# Safe Exploration of Reinforcement Learning Agents in Uncertain and Black-Box Environments

**Daniel Bethell**

*Doctor of Philosophy*

University of York
Computer Science

**March 2026**

## Abstract

Reinforcement learning (RL) is a powerful framework for sequential decision-making. However, training and deploying RL agents in safety-critical, real-world environments remains a major challenge. Many real-world applications lack a simulator or offline dataset that captures dynamics, sensing artefacts, and long-tail edge cases well enough, so effective policies often require on-system learning, adaptation, or continual calibration after deployment. In these safety-critical environments, failures are often rare but catastrophic, exploration carries real consequences, and agents must operate under uncertainty and poorly specified safety constraints. This thesis argues that safety in RL should be understood as a system-level property that emerges from how uncertainty, risk, and constraints are handled across the entire perception–decision–control pipeline.

At the perception stage, safety-relevant outputs derived from high-dimensional sensors are treated as inherently uncertain; rather than relying on point predictions, the proposed approach attaches statistically valid uncertainty to perceptual decisions using distribution-free conformal prediction combined with lightweight stochastic sampling. This helps downstream policies distinguish well-supported regions from unreliable or ambiguous sensing.

At the decision-making stage, the thesis addresses the tendency of safety-aware RL methods to become either over-confident or uniformly pessimistic. An uncertainty-aware safety evaluation mechanism is introduced to separate expected safety outcomes from uncertainty about those outcomes, enabling selective conservatism: the agent can behave cautiously when predictions are poorly supported while avoiding unnecessary avoidance in regions where experience provides reliable safety information.

Finally, at the control stage, the thesis considers environments in which system dynamics and safety constraints are not available, motivating a post-hoc enforcement mechanism that does not assume access to an analytic model or an explicit cost function. A data-driven shielding layer is developed that learns from experience to distinguish safe from unsafe behaviours in continuous black-box settings, and uses this learned notion to filter, correct, or veto candidate actions before execution.

*"Provando e riprovando"*

- Dante Alighieri

# Contents


Abstract . . . . . i
List of Figures . . . . . ix
List of Tables . . . . . xi
List of Notations . . . . . xiii
Acknowledgements . . . . . xiv
Declaration . . . . . xv

**1 Introduction** **1**
1.1 Motivation . . . . . 1
1.2 Research Hypothesis and Questions . . . . . 5
1.3 Research Contributions and Thesis Structure . . . . . 6

**2 Background and Prior Research** **9**
2.1 Background . . . . . 9
2.1.1 Reinforcement Learning . . . . . 9
2.2 Prior Research . . . . . 24
2.2.1 Uncertainty Quantification . . . . . 24
2.2.2 Safe Reinforcement Learning . . . . . 37
2.3 Gaps and Future Directions . . . . . 58
2.3.1 Comparing Uncertainty Quantification Methods . . . . . 59
2.3.2 Comparing Safe Exploration Strategies . . . . . 61
2.3.3 Open Challenges . . . . . 62

**3 Robust Uncertainty Quantification** **65**
3.1 Motivation . . . . . 65
3.2 MC-CP . . . . . 67
3.2.1 Adaptive Monte Carlo Dropout . . . . . 68
3.2.2 MC-CP for Classification . . . . . 72
3.2.3 MC-CP for Regression . . . . . 75
3.3 Evaluation . . . . . 77
3.3.1 Core Results . . . . . 78
3.3.2 Parameter Analysis . . . . . 86
3.3.3 Distribution and Covariate Shift Results . . . . . 92

3.3.4 Practical Limitations of Marginal Validity . . . 99
3.3.5 Class-wise Coverage Results . . . 102
3.3.6 Risk-Sensitive Results . . . 104
3.3.7 KDE-based Adaptive MC Dropout Results . . . 108
3.4 Safe RL Using MC-CP . . . 110
3.4.1 Perception-to-policy Framework . . . 112
3.4.2 Experimental Results . . . 117
3.5 Discussion . . . 125
3.5.1 Applicability and Impact . . . 125
3.5.2 Limitations . . . 126

**4 Reducing Over-Conservatism in Safety Critics 128**
4.1 Motivation . . . 128
4.2 USC . . . 130
4.2.1 Uncertainty-Driven Training . . . 131
4.2.2 Uncertainty Refinement . . . 135
4.3 Evaluation . . . 138
4.3.1 Core Results . . . 139
4.3.2 Deployment Results . . . 145
4.3.3 COPPOL Results . . . 146
4.4 Discussion . . . 148
4.4.1 Applicability and Impact . . . 149
4.4.2 Limitations . . . 149

**5 Adaptive Shielding for Black-Box Environments 150**
5.1 Motivation . . . 150
5.2 ADVICE . . . 152
5.2.1 Shield Construction . . . 153
5.2.2 Execution and Adaptation . . . 156
5.3 Evaluation . . . 160
5.3.1 Core Results . . . 161
5.3.2 Parameter Analysis . . . 169
5.3.3 Representation Analysis . . . 172
5.3.4 Distribution Shift Results . . . 173
5.3.5 Cautiousness Analysis . . . 175
5.4 Discussion . . . 176
5.4.1 Applicability and Impact . . . 176
5.4.2 Limitations . . . 177

**6 Conclusion 178**
6.1 Thesis Summary . . . 178
6.2 Positioning Contributions Within the Research Landscape . . . 180
6.3 Future Directions . . . 182

**Bibliography** **185**

**A Appendix for Chapter 3** **205**
A.1 Cauchy–Schwarz Derivation . . . . . 205
A.2 Mondrian Calibration Results . . . . . 205
A.3 KDE-based Adaptive MC Dropout Algorithm . . . . . 208
A.4 COPPOL Environment Examples . . . . . 208
A.5 MC-CP Experimental Setup . . . . . 211
A.5.1 Common Experimental Setup . . . . . 211
A.5.2 Distribution and Covariate Shift Experimental Setup . . . . . 213
A.5.3 Risk-Sensitive Experimental Setup . . . . . 215
A.6 COPPOL Experimental Setup . . . . . 215
A.6.1 Common Experimental Setup . . . . . 216
A.6.2 Noisy State Perturbation . . . . . 218

**B Appendix for Chapter 4** **220**
B.1 Uniform Sampling Proof . . . . . 220
B.1.1 Theorem 4.1 . . . . . 220
B.2 Full Training Results . . . . . 223
B.3 Full Deployment Results . . . . . 225
B.4 USC Experimental Setup . . . . . 226
B.4.1 Common Experimental Setup . . . . . 226

**C Appendix for Chapter 5** **232**
C.1 Theoretical Analysis . . . . . 232
C.1.1 Theorem 5.1 . . . . . 232
C.1.2 Theorem 5.2 . . . . . 233
C.1.3 Theorem 5.3 . . . . . 234
C.1.4 Theorem 5.4 . . . . . 235
C.2 Full Training Results . . . . . 236
C.3 ADVICE Experimental Setup . . . . . 236
C.3.1 Common Experimental Setup . . . . . 237

# List of Figures


1.1 Placement of the technical contributions within the embodied agent pipeline. . . 6

2.1 A typical reinforcement learning algorithm. . . . . 10
2.2 A classification of reinforcement learning algorithms. . . . . 10
2.3 A typical model-based reinforcement learning algorithm. . . . . 20
2.4 A classification of techniques for uncertainty quantification. . . . . 25
2.5 A classification of safety concerns. . . . . 38
2.6 A classification of techniques for the safe exploration of RL agents during training. 41
2.7 Architectural diagram of (a) pre-shielding and (b) post-shielding within the RL loop. . . . . 55

3.1 Illustration of the effect of imperfect perception on downstream safe policy execution. . . . . 66
3.2 High-level overview of the MC-CP framework. . . . . 67
3.3 Variance convergence behaviour of Adaptive MC Dropout. . . . . 80
3.4 The construction of prediction sets for an example image from the CIFAR-10 dataset. . . . . 81
3.5 Percentage and accuracy of prediction sets across methods for singleton (right) and mixed (left) cases. . . . . 82
3.6 Average softmax probability of the top-1 class within the constructed prediction sets across datasets of increasing complexity. . . . . 83
3.7 Predicted quantile intervals on the Boston Housing dataset for all comparative methods. . . . . 85
3.8 Visualised trends of average prediction set accuracy (left) and size (right) for all comparative CP methods on the CIFAR-10 dataset under varying target miscoverage rates $\alpha$. . . . . 87
3.9 Spearman's $\rho$ correlation between the average prediction set size and (left) the number of Adaptive MC Dropout forward passes, and (right) the resulting accuracy, across all parameter combinations of $\delta$ and $\rho_{\max}$ reported in Table 3.7. . . . 89
3.10 Non-linear relationship between average prediction set size and (left) the number of Adaptive MC Dropout forward passes and (right) achieved accuracy, modelled using a Univariate Spline fit to the reported data in Table 3.7. . . . . 89

3.11 Predicted quantile intervals on the Boston Housing dataset for MC-CP under various combinations of the threshold $\delta$ (columns) and patience $\rho_{\max}$ (rows). . . . 91
3.12 Spearman's $\rho$ correlation between the average prediction coverage and (left) the number of Adaptive MC Dropout forward passes, and (right) the resulting MAE, across all parameter combinations of $\delta$ and $\rho_{\max}$ reported in Table 3.8. . . . . 91
3.13 Non-linear relationship between average prediction coverage and (left) the number of Adaptive MC Dropout forward passes and (right) achieved MAE, modelled using a Univariate Spline fit to the reported data in Table 3.8. . . . . 92
3.14 Example ID (left) and OOD (right) samples from the CIFAR-10 → CIFAR-100 dataset pairing. . . . . 94
3.15 Qualitative examples of MC-CP prediction sets under style-based distributional shifts on the MNIST dataset. . . . . 96
3.16 Qualitative examples of MC-CP prediction sets under various covariate shifts on the CIFAR-10 dataset. . . . . 98
3.17 Variance convergence behaviour of Adaptive MC Dropout++ on CIFAR-10 dataset for an example Automobile image. . . . . 107
3.18 Variance convergence behaviour of Adaptive MC Dropout (left) and the KDE-based Adaptive MC Dropout (right). . . . . 109
3.19 High-level overview of COPPOL, our conformal-driven perception-to-policy learning approach for safe navigation over risky terrain. . . . . 112
3.20 Comparison of hazard predictions for (a) DeepMoon Crater and (b) YTU-WaterNet Water hazards. . . . . 119
3.21 Performance of comparative methods on four DeepMoon environments (vector view). . . . . 121
3.22 Performance of comparative methods on four YTU-WaterNet environments (vector view). . . . . 123
3.23 Performance of comparative methods on Environment 1 from the DeepMoon environments using the whole predictive hazard map as the state observation (image view). . . . . 124
3.24 Performance of comparative methods on a state-wise noisy variant of Environment 1 from the DeepMoon environments (vector view). . . . . 124

4.1 A qualitative comparison of the predictive cost maps in the Safety-Gymnasium CarGoal2 environments. . . . . 129
4.2 High-level USC overview. . . . . 131
4.3 Pareto plot of average episodic cost versus average episodic reward across evaluated benchmarks. . . . . 141
4.4 A qualitative comparison of the predictive cost maps in various Safety-Gymnasium environments. . . . . 144
4.5 Performance of comparative methods on Environment 1 from the DeepMoon dataset (vector view). . . . . 147

5.1 High-level ADVICE overview. . . . . 152

5.2 Average episodic reward, cumulative safety violations, cumulative goal reaches of examined approaches and example latent visualisation for all environments. . . . 163
5.3 Example trajectories of the assessed approaches on the semi-random goal environment. . . . 165
5.4 Example ADVICE shield activations during training. . . . 165
5.5 Decision regions for ADVICE's adaptive update of $K$. . . . 166
5.6 Average episodic reward, average episodic cost, cumulative goal reaches, and example latent space visualisation on the constrained randomised goal environment. 167
5.7 The average shield activations for ADVICE and the Tabular Shield for all environments. . . . 168
5.8 The rate of safety violations for all approaches in the semi-random goal, random goal, random button, and circle environments, respectively. . . . 169
5.9 The rate of safety violations, and the rate of goal reaches when ADVICE is activated at different intervals $E_{max}$. . . . 170
5.10 Average episodic reward, cumulative safety violations, and cumulative goal reaches of various values of $K$ under fixed ADVICE on the randomised goal environment. 170
5.11 Average episodic reward, cumulative safety violations, and cumulative goal reaches on the semi-randomised goal environment where some methods are pretrained on a different environment. . . . 174
5.12 Average episodic reward, cumulative safety violations, and cumulative goal reaches of various approaches on the randomised goal environment where the maximum episodic steps are doubled for ADVICE. . . . 175

6.1 A classification of techniques for uncertainty quantification, including our technical contributions. . . . 181
6.2 A classification of techniques for the safe exploration of RL agents during training, including our technical contributions. . . . 181

A.1 Qualitative visualisations of the four environments used to evaluate COPPOL. . 209
A.2 Qualitative visualisations of an example COPPOL (using MC-CP) trajectory in the four environments used to evaluate COPPOL for both datasets. . . . 210
A.3 The single global lunar digital elevation model used to create the DeepMoon dataset. 216

B.1 Average episodic reward, cost, and cumulative goal reaches of examined methods across Safety Gymnasium environments. . . . 223
B.2 Average episodic reward, cost, and cumulative goal reaches of examined methods across Gymnasium Robotics and Mujoco environments. . . . 224
B.3 Average deployment episodic reward, cost, and cumulative goal reaches of examined methods across Safety Gymnasium environments. . . . 225
B.4 Average deployment episodic reward, cost, and cumulative goal reaches of examined methods across Gymnasium Robotics and Mujoco environments. . . . 226
B.5 Different robots used within the tasks of the evaluated test-suites. . . . 227
B.6 Example navigation tasks with varying complexity levels for evaluating USC. . . 228

C.1 Average episodic reward, cumulative safety violations, cumulative goal reaches of examined approaches and example latent visualisation for all environments. . . . 237
C.2 Example navigation tasks. . . . . 239

# List of Tables


2.1 Overview of uncertainty quantification methods highlighting trade-offs. . . . . 59
2.2 Overview of safe RL methods highlighting trade-offs. . . . . 61

3.1 Evaluation of predictive uncertainty quality and efficiency across MC-Dropout variants. . . . . 79
3.2 Average prediction set accuracy and size for all comparative methods across datasets. . . . . 81
3.3 Average prediction set accuracy and size for all comparative CP methods on Tiny-ImageNet using VGG16 and VGG19 architectures. . . . . 83
3.4 Average prediction set accuracy per class for all comparative methods on the CIFAR-10 dataset. . . . . 84
3.5 Mean absolute error (MAE) and coverage for all comparative methods across datasets. . . . . 84
3.6 Average prediction set accuracy and size for all comparative CP methods on the CIFAR-10 dataset under various target miscoverage rates $\alpha$. . . . . 87
3.7 Average prediction set accuracy, Adaptive MC Dropout forward passes, and prediction set size for MC-CP on the CIFAR-10 dataset under various combinations of the threshold $\delta$ and patience $\rho_{\max}$ parameters. . . . . 88
3.8 Mean absolute error (MAE), average Adaptive MC Dropout forward passes, and coverage for MC-CP on the Boston Housing dataset under various combinations of the threshold $\delta$ and patience $\rho_{\max}$ parameters. . . . . 90
3.9 Average prediction set sizes for all comparative CP methods for two in-distribution (ID) and out-of-distribution (OOD) dataset pairings. . . . . 93
3.10 Average prediction set accuracy and set size for all comparative CP methods under two style-based distributional shifts on the MNIST dataset. . . . . 94
3.11 Average prediction set accuracy and set size for all comparative CP methods under four covariate shift types on the MNIST and CIFAR-10 datasets. . . . . 97
3.12 Average prediction set accuracy and size for all comparative CP methods on the CIFAR-10 dataset under various calibration dataset sizes $|\mathcal{D}_{\text{cal}}|$. . . . . 99
3.13 Average prediction set accuracy per class for all comparative CP methods on the CIFAR-10 dataset under two calibration set distributions. . . . . 100
3.14 Average prediction set size per class for all comparative CP methods on the CIFAR-10 dataset under two calibration set distributions. . . . . 101

3.15 Average prediction set accuracy per class for MC-CP and its sTACP variant on the CIFAR-10 dataset using the extremely imbalanced calibration dataset. . . . . 103
3.16 Average prediction set size per class for MC-CP and its sTACP variant on the CIFAR-10 dataset using the extremely imbalanced calibration dataset. . . . . 103
3.17 Average prediction set accuracy and size for all comparative CP methods on the CIFAR-10 dataset using a small calibration dataset $|\mathcal{D}_{\text{cal}}| = 50$. . . . . 105
3.18 Average prediction set accuracy for all comparative CP methods using the base nonconformity score and a risk-augmented variant on the CIFAR-10 dataset. . . 105
3.19 Average prediction size accuracy for all comparative CP methods using the base nonconformity score and a risk-augmented variant on the CIFAR-10 dataset. . . 106
3.20 Average class-wise accuracy for MC-Dropout, Adaptive MC-Dropout, and the class-sensitive Adaptive MC-Dropout++ on the CIFAR-10 dataset. . . . . 107
3.21 Evaluation of predictive uncertainty quality and efficiency across MC-Dropout variants on ID and OOD data. . . . . 109
3.22 Evaluation of hazard segmentation on DeepMoon and YTU-WaterNet. . . . . 119

4.1 Mean episodic reward and cost for comparative methods across evaluated benchmarks. . . . . 140
4.2 Predictive cost map errors in the Safety Gymnasium tasks compared to ground truth. . . . . 143
4.3 Mean episodic reward and episodic cost for comparative methods across evaluated benchmarks during 10 independent deployment runs. . . . . 145
4.4 Success rates (%) for comparative methods across evaluated benchmarks in 10 independent deployment runs. . . . . 146

5.1 Mean sensitivity analysis of $h_d$ and $h_r$ on the randomised goal environment for $E_{max} = 1000$. . . . . 171
5.2 The empirical 1-nearest-neighbour misclassification probability $\mathbb{P}$(misclass) and achieved class-separation margins in the learned embedding space for ADVICE across evaluated environments. . . . . 173

A.1 Average prediction set accuracy per class for MC-CP using various quantile construction methods on the CIFAR-10 dataset under an extremely imbalanced calibration dataset. . . . . 207
A.2 Average prediction set size per class for MC-CP using various quantile construction methods on the CIFAR-10 dataset under an extremely imbalanced calibration dataset. . . . . 207
A.3 MC-CP hyperparameters used in the experimental evaluation. . . . . 213
A.4 COPPOL hyperparameters used in the experimental evaluation. . . . . 218

B.1 USC hyperparameters used in the experimental evaluation. . . . . 229

C.1 Performance of the evaluated methods across five environments. . . . . 238
C.2 ADVICE hyperparameters used in the experimental evaluation. . . . . 241

# List of Notations

| Symbol | Meaning | Page |
|---|---|---|
| $M$ | Markov Decision Process (MDP) | 10 |
| $S$ | State space | 10 |
| $s_t$ | State at time step $t$ | 10 |
| $T$ | Maximum timesteps | 10 |
| $A$ | Action space | 10 |
| $P$ | State transition probability matrix | 11 |
| $a_t$ | Action at time step $t$ | 11 |
| $R$ | Reward function | 11 |
| $\gamma$ | Discount factor | 11 |
| $\pi$ | Policy | 11 |
| $V_\pi(s)$ | State-value function | 11 |
| $Q_\pi(s, a)$ | Action-value function | 11 |
| $E$ | Maximum episodes | 12 |
| $\eta$ | Learning rate | 13 |
| $\theta$ | Model parameters | 15 |
| $\mathcal{L}$ | Loss function | 15 |
| $\mathcal{D}$ | Dataset | 15 |
| $\epsilon$ | A small positive constant | 17 |
| $\lambda$ | Loss weighting | 19 |
| $x$ | Single input data | 26 |
| $X$ | Full set of input data | 26 |
| $y$ | Single ground truth label | 26 |
| $Y$ | Full set of ground truth labels | 26 |
| $p$ | Probability density function | 26 |
| $\mathcal{F}_\theta$ | Neural network with parameters $\theta$ | 27 |
| $N_{\text{mc}}$ | Maximum number of forward passes | 27 |
| $\alpha$ | Desired miscoverage rate | 35 |
| $\Gamma$ | Constructed prediction set | 35 |
| $\kappa$ | Nonconformity scoring function | 35 |
| $C$ | Cost function | 40 |
| $\chi$ | Maximum allowed cost per episode | 40 |

| Symbol | Meaning | Page |
|---|---|---|
| $\phi$ | Safety shield | 54 |
| $g$ | Labelling function | 56 |
| $\hat{\mu}$ | Computed Mean | 68 |
| $\hat{\sigma}^2$ | Variance | 68 |
| $\hat{y}$ | Predicted output | 70 |
| $\rho$ | Counter | 71 |
| $\rho_{\max}$ | Maximum number of consecutive count $\rho$ | 71 |
| $\delta$ | Threshold | 71 |
| $\xi$ | A single class | 73 |
| $\Xi$ | Full set of classes | 73 |
| $\mathcal{R}$ | Risk function | 104 |
| $\Omega$ | Pixel lattice | 113 |
| $\mathfrak{H}$ | Ground truth hazardous set | 113 |
| $\mathfrak{C}$ | Predicted hazard set | 113 |
| $\mathcal{M}$ | Predicted binary hazard mask | 113 |
| $B$ | Mini-batch sample | 132 |
| $\mathcal{B}$ | Replay buffer | 132 |
| $u$ | Uncertainty scalar | 132 |
| $k$ | A single nearest neighbour | 135 |
| $K$ | Set of nearest neighbours | 135 |
| $F$ | Feature space | 152 |
| $f_t$ | Feature at time step $t$ | 152 |
| $\mathfrak{v}$ | Latent embedding | 154 |
| $m$ | Contrastive separation margin | 155 |
| $z_e$ | Safety violation indicator at episode $e$ | 158 |
| $h_r$ | Recent history window | 158 |
| $h_d$ | Distant history window | 158 |

# Acknowledgements

I would like to acknowledge the many individuals and organisations that supported me throughout this PhD. In particular, I would like to thank my supervisors, Dr. Simos Gersimou, Prof. Radu Calinescu, and Dr. Calum Imrie. Their guidance, support, and expertise have been invaluable throughout this journey, and their individual mentorship has played a key role in shaping both this research and my development as an independent researcher. I feel extremely fortunate to have had such an exceptional supervisory team.

I would also like to thank my examiners, Prof. Ivana Dusparic and Dr. Peter Nightingale, for taking the time to carefully examine my thesis and for their thoughtful feedback. I am particularly grateful to Peter, who also served as my internal advisor, for his support and guidance throughout my PhD and for the care and attention he gave to reviewing my work during our regular progress meetings. I gratefully acknowledge the funding provided by the Doctoral Centre for Safe, Ethical and Secure Computing (SEtS), and in particular the support of Dr Leandro Soares Indrusiak. Without this support, I would not have been able to undertake this PhD.

Beyond the academic support I have received, I am grateful to the many friends and colleagues who have made this PhD an enjoyable and memorable experience. I particularly value the friendships formed within the department during my PhD and those made at conferences and research events across the world. I would also like to thank my friends outside academia, especially those at Jorvik Warriors Basketball, whose support, encouragement, and great laughs have been invaluable over the years.

I would like to thank my family for their loving support throughout this journey. I am especially grateful to my mum, Claire, whose love, encouragement, and belief in me have always inspired me to pursue my own path and build a career for myself. I would also like to thank my dad, Niel, for instilling in me the determination and work ethic that made completing this PhD possible, values he has embodied throughout his life and career and that I have always looked up to. I am equally thankful to the rest of my family for their continued support and encouragement over the years, and to Ned, Ted, and Marmite for keeping me company.

I like to think the Manchester *worker bee* spirit of hard work, resilience, and perseverance helped carry me through this journey. As Dante writes in the Paradiso, *"Provando e riprovando"*, to try, and try again, a sentiment that mirrors the trial-and-error nature of reinforcement learning that runs throughout this thesis and the perseverance needed to pursue a PhD.

Finally, I would like to thank my partner, Charmaine Barker, whose love, patience, and encouragement sustained me throughout this PhD. Simultaneously undertaking a PhD of her own, she understood the unique challenges of this journey better than anyone, and her emotional and intellectual support has been a constant source of strength. During the most difficult moments of this PhD, she kept me pushing forward. I am deeply grateful to have shared this experience and life with her.

# Declaration

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for a degree or other qualification at this University or elsewhere. All sources are acknowledged as references.

Parts of the research described in this thesis have been previously published in four research papers. The approach and parts of the evaluation described in Chapter 3 contributed to the papers:

Bethell, D., Gerasimou, S. and Calinescu, R. (2024) "Robust Uncertainty Quantification Using Conformalised Monte Carlo Prediction". In: *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence* (AAAI 2024). Vancouver, Canada, 20 February 2024.

and

Bethell, D., Gerasimou, S., Calinescu, R. and Imrie, C. (2026) "Learning to Navigate Under Imperfect Perception: Conformalised Segmentation for Safe Reinforcement Learning". In: *Proceedings of the 41st ACM/SIGAPP Symposium on Applied Computing* (SAC 2026). Thessaloniki, Greece, 23 March 2026.

and the approach and parts of the evaluation described in Chapter 4 contributed to the paper:

Bethell, D., Gerasimou, S., Calinescu, R. and Imrie, C. (2026) "Safe But Not Sorry: Reducing Over-Conservatism in Safety Critics via Uncertainty-Aware Modulation". In: *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems* (AAMAS 2026). Paphos, Cyprus, 25 May 2026.

and the approach and parts of the evaluation described in Chapter 5 contributed to the paper:

Bethell, D., Gerasimou, S., Calinescu, R. and Imrie, C. (2025) "Safe Reinforcement Learning in Black-Box Environments via Adaptive Shielding". In: *Proceedings of the 28th European Conference on Artificial Intelligence* (ECAI 2025). Bologna, Italy, 25 October 2025.

Signed 

# Chapter 1

# Introduction

This thesis studies how reinforcement learning agents operating in safety-critical environments can achieve improved safety and robustness. The central view is that safety cannot be secured by a single algorithmic component in isolation; it must instead be treated as a system-level property that depends on how uncertainty, risk, and constraints are handled across the entire perception–decision–control pipeline.

This chapter is structured as follows. Section 1.1 motivates the overall problem setting, including why safety is a necessity and the challenges of achieving safe behaviour for reinforcement learning agents in safety-critical domains across the perception–decision–control pipeline. Section 1.2 poses the set of research questions and hypotheses investigated, and Section 1.3 describes the contributions of this work and outlines the structure of the remainder of the thesis.

## 1.1 Motivation

Reinforcement Learning (RL) has emerged as a powerful framework for sequential decision-making, which has enabled agents to learn complex behaviours across various domains, such as game playing [1], robotic control [2], and human preference fine-tuning [3]. In RL, an agent interacts with an environment over time, repeatedly choosing actions, observing how the environment responds, and receiving feedback that reflects how desirable the resulting outcomes are [4]. By continually adjusting its behaviour in response to this feedback, the agent can, in principle, discover effective strategies without requiring explicit supervision or a hand-crafted control policy [5]. This paradigm is particularly appealing for complex systems where it is difficult to specify in advance how the agent should act in every possible situation, but where it is possible to evaluate the quality of its behaviour through experience.

RL agents are typically trained in simulation, where exploration of the environment can proceed without physical risk and thus carry no consequences in most cases. However, simulated environments can only approximate the complexity, variability, and look of the real world. Mismatches in dynamics, sensing, and unknown phenomena give rise to a *sim-to-real* gap [6], so that policies that perform well in simulation may still exhibit unexpected behaviour once deployed,

necessitating learning in the physical environment itself. In many envisioned applications, the consequences of an agent's behaviour extend far beyond the abstract performance metrics typically reported in benchmark studies. When learned controllers are embedded in physical systems, actions can have direct implications for safety and compliance with operational constraints. A single erroneous action may cause damage to equipment, violation of regulatory limits, or even harm to human users [7]. In such settings, failures cannot be treated as a benign side-effect of learning, but instead represent outcomes that may be ethically unacceptable [8] or physically prohibitive [9], even if they occur only rarely. These characteristics distinguish safety-critical domains from many of the environments in which RL methods are commonly evaluated. Standard assumptions, particularly the idea that undesirable outcomes can simply be penalised and averaged out over many episodes [10], are no longer adequate.

From the perspective of system design, this implies that learning-based controllers must satisfy not only task-related performance objectives, but also a range of explicit and implicit safety requirements. Crucially, these requirements apply both during the learning process itself and the deployment of the agent. This motivates the development of methods that treat safety as a primary design objective rather than an afterthought.

Many of the safety-critical scenarios in which RL agents are used involve embodied agents: systems that are physically instantiated in, or tightly coupled to, a real or high-fidelity simulated environment [11]. An agent's actions are realised through physical actuators and mediated by noisy, partial observations from sensors. As a consequence, the behaviour of the learning system cannot be understood solely in terms of abstract state transitions and rewards, but must also be viewed as the outcome of a closed-loop interaction between perception, decision-making, and low-level control (a high-level pipeline can be seen in Figure 1.1). At the perception stage, raw sensor streams, such as images, depth maps, or lidar scans, are transformed into a more structured representation of the environment for the agent's state of the environment it is placed within. This representation may include, for example, estimates of obstacle locations, terrain types, semantic classes, or other task-relevant features. The decision-making stage then uses this processed information, together with task objectives and any available prior knowledge, to select actions. Finally, the control stage translates these high-level commands into low-level actuator inputs, taking into account the physical dynamics of the system.

Reasoning about safety in such embodied settings, therefore, requires taking this entire loop into account. The overall reliability of the system depends not only on the quality of the learned decision-making component, but also on what the agent is able to perceive and how its decisions are ultimately executed. Perceptual errors may cause relevant hazards to be missed or misclassified; limitations in the decision-making module may lead to choices that disregard rare but critical failure modes; and constraints at the control level may restrict which safe actions can be implemented in practice. Understanding how these stages interact, and how learning-based components fit within this architecture, is essential for analysing the conditions under which RL can be deployed responsibly in safety-critical, embodied domains.

A central challenge arises already at the perception stage. In safety-critical, embodied systems,

the agent does not have direct access to the true state of the environment, but must infer it from potentially noisy, partial sensor data. Cameras and lidars provide high-dimensional streams that must be processed into a compact representation indicating, for example, the location and extent of obstacles, the presence of hazardous terrain, or the proximity of other agents. Errors in this transformation can have serious implications [12], particularly missed detections of relevant hazards may expose the agent to risks that no amount of downstream decision-making can fully mitigate. Conversely, overly pessimistic interpretations of the scene can lead the agent to treat large regions as unsafe or impassable, reducing its ability to accomplish its objectives.

These difficulties are compounded by the fact that the most critical aspects of the environment are often rare, ambiguous, or subject to substantial variability. Hazardous situations may correspond to atypical sensor configurations, challenging lighting or weather conditions, or subtle cues that are difficult to distinguish from benign patterns. As a result, the perceptual components of an embodied agent must not only achieve high average accuracy, but also behave reliably where safety-critical events occur. From the perspective of RL, this means that the agent's subsequent decisions are inherently constrained by what it can perceive and how confident it can be about those perceptions. In this sense, the overall safety of the system is tightly coupled to the quality and robustness of its perception pipeline [13].

Even if the environment were perceived perfectly, an embodied agent must still decide how to act in an environment that has potential risk. At each decision point, it must weigh not only whether an action is likely to advance its task objectives, but also how that action might influence the probability and severity of undesirable events further along the trajectory. In learning-based systems, this typically entails maintaining internal predictions about the long-term consequences of different choices, including the likelihood that they will eventually lead to hazardous situations or violations of safety requirements. If these internal predictions are systematically too optimistic, the agent may underestimate how risky certain actions truly are and select behaviours that expose it to rare but catastrophic failures [14]. Conversely, if they are excessively pessimistic, the agent may regard many potentially useful behaviours as too dangerous, leading to overly cautious strategies that avoid large regions of the environment and severely limit task performance.

Constructing reliable predictions of future safety outcomes is particularly challenging. Catastrophic, or even small, safety violations are infrequent [15]. Feedback about safety may also depend on combinations of factors that are difficult to disentangle from limited experience. Moreover, the agent must form sensible judgements about unfamiliar situations it has not yet encountered, extrapolating from past data while respecting the possibility of unobserved hazards. As a result, the component of the system that estimates how current decisions affect future safety can become a significant bottleneck: if it fails to capture important risks, the agent may behave recklessly; if it overstates uncertainty or potential danger, the agent may become unduly conservative. Achieving an appropriate balance between these extremes is essential for learning controllers that are both safe and effective in complex, safety-critical environments.

Another source of difficulty arises at the interface between the agent's decisions and the en-

vironment in which they are executed. In many systems, there exist safety requirements that can in principle be expressed as constraints on the agent's behaviour: bounds on velocities and accelerations, limits on forces or torques, or operating regions that must never be entered. When a learning-based controller proposes an action, some mechanism is typically required to ensure that this action is compatible with such requirements before it is applied to the physical system [16]. This may involve monitoring candidate actions for potential violations, adjusting them to respect known limits, or vetoing them altogether when they would lead the system into clearly unsafe regions. In this way, safety is not only a matter of what the agent decides to do, but also of how those decisions are filtered, modified, or overridden at the point of execution.

In realistic applications, however, the safety requirements themselves are often difficult to cast as precise, tractable constraints [17]. They may depend on high-dimensional sensory inputs, nuanced notions of acceptable behaviour that vary across contexts, or complex interactions between the agent and environmental factors. Domain experts may be able to articulate qualitative rules, such as "keep a safe distance from pedestrians" or "avoid configurations that place excessive stress on this component, but not be able to translate these into closed-form inequalities over the agent's state and actions. Compounding this, the environment frequently behaves as a black box from the agent's perspective: its dynamics are only partially known, subject to unmodelled disturbances, or constantly evolving. These black-box characteristics also imply that learning cannot be reduced to solving a known constrained control problem: the agent must acquire competence through interaction. As a result, safe exploration under incomplete knowledge becomes a central challenge, since the agent must gather data in uncertain regions without incurring unacceptable violations. In such settings, relying solely on hand-crafted rules or simplified models to enforce safety at execution time can be problematic. If the rules are too coarse or the models too optimistic, important failure modes may go unprotected; if they are tuned to be overly cautious, they may prevent the agent from exploiting safe but non-obvious behaviours. Ensuring that the actions produced by a learning-based controller remain safe despite incomplete knowledge of both the environment and the true safety constraints, therefore, constitutes a third fundamental challenge for the deployment of RL agents in safety-critical domains (particularly in learning-based self-adaptive systems facing unknown and evolving conditions) [18].

A challenge common to these stages is defining an acceptable balance between safety and task performance. Reward and cost may have different scales, while the consequences of violations depend on the application, making penalty weights and safety budgets difficult to specify. Conservative hazard predictions and action filtering also influence this balance by restricting potentially useful behaviour. Evaluation must therefore consider safety and performance jointly: lower cost with lower reward represents a compromise, whereas lower cost at comparable reward, or higher reward within the same safety budget, provides clearer evidence of improvement. Comparisons should account for training budgets, episode horizons, and variability across runs, alongside task completion and hazard exposure, since reward alone may penalise safer but longer trajectories.

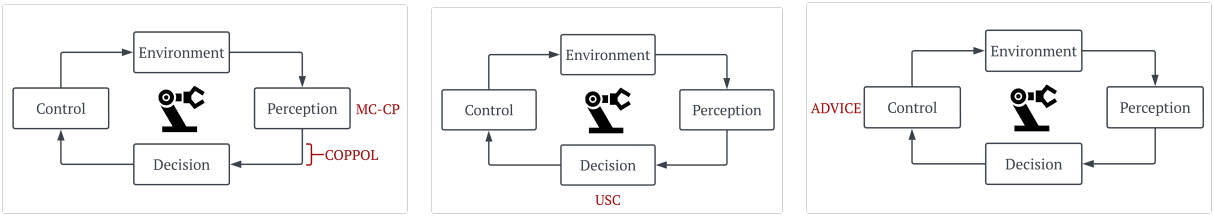
## 1.2 Research Hypothesis and Questions

Given the motivation and challenges outlined above, this thesis is guided by a research hypothesis and set of research questions that describe how safety could be enforced at different stages of the perception-decision-control pipeline in safety-critical settings. Explicitly modelling and using uncertainty across the perception–decision–control pipeline enables reinforcement learning systems to explore and operate more safely under incomplete knowledge, culminating in improved safety during exploration in black-box environments, without inducing excessive conservatism or substantial loss of task performance.

1. **To what extent can incorporating distribution-free uncertainty quantification at the perception stage mitigate the impact of imperfect sensing and improve downstream safety without degrading task performance?** When perceptual outputs are accompanied by distribution-free guarantees about their reliability and uncertainty, rather than being treated as confident predictions, the agent can better distinguish between regions where the environment is well understood and regions where its knowledge is incomplete or unreliable. Explicitly accounting for such uncertainty when making decisions is expected to reduce exposure to unobserved or misclassified hazards, thereby improving safety, while still retaining enough information about the environment for the agent to carry out its intended tasks effectively.

2. **To what extent can explicitly accounting for uncertainty in future safety predictions help an agent regulate its level of pessimism, avoiding both reckless and overly cautious behaviour?** Treating uncertainty about safety separately from the task performance, rather than conflating the two, enables an agent to be selectively cautious. It can behave conservatively in situations where little is known about the consequences of its actions, while exploiting opportunities more fully in regions of the environment where past experience provides a reliable basis for prediction. An evaluation mechanism that responds to uncertainty in this way is expected to reduce catastrophic failures compared to agents that ignore uncertainty altogether, while avoiding the excessive pessimism and corresponding loss of performance that arise when worst-case behaviour is assumed indiscriminately.

3. **How can adaptive, data-driven safety mechanisms provide meaningful safety performance when system dynamics and constraints are difficult or impossible to model explicitly?** In settings where neither the environment's dynamics nor its safety constraints can be specified accurately in advance, relying solely on fixed, handcrafted rules or simplified models to enforce safety is inherently limited. A mechanism that instead learns to filter, adjust, or veto proposed actions on the basis of accumulated experience can gradually acquire an operational notion of safe and unsafe behaviour that is better aligned with the underlying, and only partially understood, constraints. Such an adaptive enforcement mechanism is expected to reduce the frequency and severity of safety violations relative to unprotected agents, while preserving a substantial portion of the task performance that would be achievable under more idealised modelling assumptions.




(a) MC-CP and COPPOL (b) USC (c) ADVICE

Figure 1.1: Conceptual placement of the technical contributions within the embodied agent perception-decision-control pipeline: (a) MC-CP supports uncertainty-aware perception, which COPPOL incorporates into the downstream RL task; (b) USC guides policy learning through safety-cost estimation; and (c) ADVICE filters proposed actions before execution. The contributions may be used independently or in combination.

## 1.3 Research Contributions and Thesis Structure

Guided by the challenges and hypotheses outlined above, this thesis makes three primary methodological contributions, each targeting a distinct stage of the perception–decision–control loop for embodied agents in safety-critical environments. This can be visualised in Figure 1.1. Taken together, these contributions aim to improve how learning-based controllers perceive safety-relevant aspects of their environment, modulate their pessimism surrounding future actions in safety-critical environments, and enforce safe behaviour when system dynamics and constraints cannot be specified in closed form. This thesis is structured so that these contributions are developed against a common conceptual and methodological background.

Chapter 2 provides a comprehensive review of the relevant background and prior research on reinforcement learning, uncertainty quantification, and methods for the safe exploration of reinforcement learning agents during training. This chapter also highlights open problems and limitations in current methods that motivate the three research hypotheses and the corresponding contributions developed in the remainder of the thesis.

Chapter 3 addresses the issues at the perception stage discussed previously and investigates our first research hypothesis, concerning uncertainty-aware perception for safety-critical tasks. It introduces **M**onte-**C**arlo **C**onformal **P**rediction (MC-CP), a framework that combines a novel lightweight stochastic sampling procedure with a distribution-free, statistically valid uncertainty quantification method to robust obtain predictions. Its stochastic sampling component estimates epistemic uncertainty, while conformal calibration accounts for predictive errors arising from both model limitations and aleatoric variability. The chapter formalises MC-CP, analyses its finite-sample coverage guarantees under minimal assumptions, and examines its practical instantiations for identifying safety-critical objects. MC-CP can be applied to classification and regression predictors supporting stochastic dropout inference, provided suitable labelled calibration data is available; it is evaluated here on image classification and tabular regression tasks. It then integrates the resulting uncertainty-aware perceptual outputs into a perception-to-policy pipeline for embodied agents, COPPOL, evaluated using terrain imagery with hazard maps

supplied to the agent as either spatial observations or derived vector features, evaluating how exploiting these robust predictions can affect safety metrics and task performance. MC-CP is independent of the downstream RL algorithm, while COPPOL can accommodate value-based, policy-based, or actor-critic learners with compatible observation and action representations; the experiments in this chapter use the value-based DQN algorithm. This chapter provides the first technical contribution of the thesis: a principled, distribution-free approach to perception-side uncertainty quantification that directly supports safer downstream decision-making.

Chapter 4 investigates the decision-making stage and investigates our second hypothesis, which concerns utilising uncertainty quantification to help regulate an agent's pessimism in safety-critical environments to avoid careless and overly cautious behaviour. It proposes the **U**ncertain **S**afety **C**ritic (USC), an uncertainty-aware safety evaluation mechanism that modulates pessimism according to influence-based estimates of epistemic uncertainty in the safety critic. USC operates on state-action representations and requires observed safety costs during training. Its evaluation uses vector observations in navigation, robotic manipulation, and locomotion tasks, including perceptually derived vector observations within COPPOL. USC is not intrinsically restricted to actor-critic algorithms: its safety estimates can guide either policy updates or action selection in value-based methods. The presented training procedure and main benchmark evaluation use DDPG, with integration into other algorithms requiring corresponding adaptations. The chapter formalises USC as a safety value function shaped by both data and uncertainty, and analyses how this shaping influences the structure of the learned safety landscape over the environment. This yields the second technical contribution of the thesis: a decision-level mechanism that seeks to maintain or improve safety while reducing unnecessary conservatism.

Chapter 5 focuses on the control stage in settings where the environment behaves as a black box and explicit safety constraints are difficult or impossible to specify, thereby addressing the final hypothesis of this thesis. It proposes ADVICE (**AD**apti**V**e Sh**I**elding with a **C**ontrastive Auto**E**ncoder), an adaptive, data-driven mechanism that operates as a post-hoc protective layer around a reinforcement learning agent. ADVICE refines its notion of safe and unsafe behaviour based on experience, filtering, correcting, or allowing proposed actions without requiring an explicit model of the system dynamics or a hand-crafted constraint set. Its cautiousness adapts to observed safety-violation trends, without explicitly estimating uncertainty. It operates on encoded state-action pairs and requires interaction outcomes from which safe and unsafe examples can be labelled. The evaluation uses pseudo-lidar observation vectors and continuous actions in navigation tasks; application to other observation modalities would require suitable encodings and is not evaluated here. ADVICE's safety classification is independent of the policy update rule, although its replacement-action selection requires a reward action-value estimator. It is evaluated with DDPG; learners without such an estimator would require an alternative selection rule. This chapter represents the third technical contribution of the thesis: a practical mechanism for enforcing safety in black-box environments.

The contributions can be selected or combined according to the available observations and safety feedback; their placement within a common pipeline does not require their joint use. For example, uncertain perception motivates MC-CP, while directly observed states may permit

USC or ADVICE to operate without it. Section 4.3.3 evaluates COPPOL with MC-CP and USC, using calibrated hazard maps to inform the agent's observations and derive a dedicated safety cost. USC could also be paired with ADVICE when policy optimisation benefits from a cost signal but additional action filtering is desirable. This may be useful when costs are noisy or incompletely represent safety requirements, provided that the shield receives sufficiently reliable outcome feedback offering additional safety information. Combining all three would therefore be appropriate for settings involving uncertain perception, an informative cost signal for critic learning, and identifiable safe and unsafe outcomes for shield construction. In such a configuration, calibrated perception would inform a policy trained with USC, whose proposed actions would then be assessed by ADVICE.

Finally, Chapter 6 concludes the findings from the preceding chapters, reflecting on how the proposed methods collectively inform the design of uncertainty-aware reinforcement learning systems for safety-critical, embodied applications. It discusses the limitations of the current work, outlines potential avenues for integrating perception, decision-making, and enforcement components more tightly in end-to-end architectures, and identifies promising directions for future research on safe reinforcement learning under uncertainty.

# Chapter 2

# Background and Prior Research

This chapter introduces background concepts and prior research that will be used throughout this thesis. The proposed methods in this thesis will build upon the prior methods discussed. A brief introduction to background knowledge, such as RL, will be presented in Section 2.1. Subsequently, we discuss model-free reinforcement learning methods in Section 2.1.1.2 and model-based methods in Section 2.1.1.3. Next, we explore the prior research that this thesis aims to build upon in Section 2.2, including safe RL research in Section 2.2.2 and UQ research in Section 2.2.1. In Section 2.3, we compare the research presented in the previous sections, before finally analysing gaps and future directions.

## 2.1 Background

In this section, we introduce the background concepts, such as RL, that are foundational to this thesis. We summarise the agent-environment interaction, define policies and value functions, and outline the distinction between model-free and model-based learning. The aim is to provide only the conceptual and notational background needed for later chapters, rather than a survey of prior work.

### 2.1.1 Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning where an agent learns to make decisions in a dynamic environment to maximise a reward signal: a scalar feedback value that indicates the immediate desirability of an action's outcome, guiding the agent toward behaviours that yield higher long-term gains [4, pp. 2–7]. Unlike supervised learning, where models are trained on labelled datasets, RL agents learn through interaction (trial-and-error) with an environment by receiving sparse or regular feedback in the form of rewards and penalties. This unique paradigm is heavily inspired by behavioural psychology, particularly the way animals and humans learn from environmental stimuli and reinforcement signals [19].

A typical RL system consists of an agent interacting with an environment over a sequence of discrete time steps, visualised in Figure 2.1. At each iteration of the loop or *step*, the agent

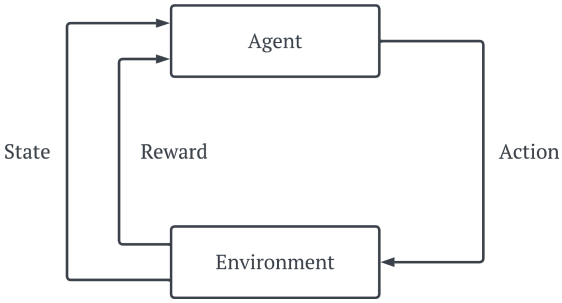



Figure 2.1: A typical reinforcement learning algorithm.

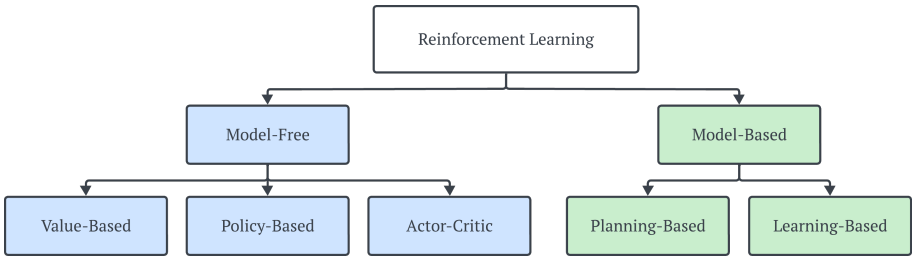



Figure 2.2: A classification of reinforcement learning algorithms.

observes the current *state* of the environment, selects the most appropriate *action*, and receives a *reward* along with the next state of the environment. The agent's goal is to learn a policy, a mapping from states to actions, that maximises the expected return, commonly formalised as the discounted sum of future rewards.

A central challenge in RL is the exploration-exploitation trade-off [20]. To maximise long-term return, the agent must balance exploring unfamiliar actions that might yield high rewards with exploiting known actions that already lead to favourable outcomes, so it can converge to an optimal policy.

Another core design decision lies in the reward specification. Reward signals can be sparse, issued only upon task completion, or dense, provided at each step based on proximity to the goal. While dense rewards can accelerate learning, they risk encoding unintended biases or restricting the solution space.

Modern RL algorithms are typically divided into categories such as model-free versus model-based. These and relevant algorithms are discussed in Sections 2.1.1.2 and 2.1.1.3, respectively. A classification of RL algorithms can be visualised in Figure 2.2.

#### 2.1.1.1 Definitions and Core Concepts

Reinforcement learning can be modelled as a Markov Decision Process (MDP) [21]: a discrete-time stochastic control process which extends Markov chains [22]. The key difference between Markov chains and MDPs is the addition of rewards and choices at each state. A finite or infinite-horizon MDP can be defined by the tuple $M = (S, A, P, R, \gamma)$, where $S$ is the set of all possible states called the state space, $s_t \in S$ is the state at timestep $t \in [0, 1, \ldots, T]$, $A$ is the set of all possible actions called the action space, $A_s$ is the set of actions available from state

$s$, $P$ is the state transition probability matrix/transition function such that $P(s_{t+1}|s_t,a_t)$ is the probability of transitioning to state $s_{t+1}$ from state $s_t$ using action $a_t$, $R$ is the reward function such that $R(s,a) \in \mathbb{R}$ is the reward for taking action $a$ in state $s$, and $\gamma \in [0,1)$ is the discount factor that determines the weight of future rewards. A higher discount factor means that future rewards are taken into consideration just as much as immediate rewards, and a lower discount factor means only immediate rewards matter more.

A policy $\pi$ is a function that specifies the agent's behaviour at each state; formally, it maps states to a probability distribution over actions, $\pi : S \to \Delta(A)$. RL solves MDPs when full knowledge of the transition function is not available; otherwise, an MDP can be solved through dynamic programming techniques [23]. The agent's goal is to find an optimal policy $\pi^*$ that maximises the expected discounted sum of rewards over the horizon:

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right] \tag{2.1}$$

To evaluate the performance of a policy $\pi$, the state-value function $V_\pi(s)$ is defined as the expected return when starting in state $s$ and following the policy $\pi$:

$$V_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid s_t = s\right] \tag{2.2}$$

Similarly, the action-value function $Q_\pi(s,a)$ gives the expected return after taking action $a$ in state $s$, and thereafter following policy $\pi$:

$$Q_\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid s_t = s, a_t = a\right] \tag{2.3}$$

These value functions satisfy the Bellman equation [24], which defines the recursive relationship between the values of states and actions. In particular, the Bellman expectation equation for the state-value function is:

$$V_\pi(s_t) = \sum_{a_t \in A} \pi(a_t \mid s_t) \left[R(s_t, a_t) + \gamma \sum_{s_{t+1} \in S} P(s_{t+1} \mid s_t, a_t) V_\pi(s_{t+1})\right] \tag{2.4}$$

The optimal value function $V^*(s)$ satisfies the Bellman optimality equation, which expresses the best achievable expected return from each state:

$$V^*(s_t) = \max_{a_t \in A} \left[R(s_t, a_t) + \gamma \sum_{s_{t+1} \in S} P(s_{t+1} \mid s_t, a_t) V_{\pi^*}(s_{t+1})\right] \tag{2.5}$$

Reinforcement learning algorithms aim to approximate these value functions and derive an optimal policy $\pi^*$, often without access to the underlying transition dynamics $P$ or reward

function $R$. The optimal value-function is the same as value function under the optimal policy $V^*(s) = V_{\pi*}(s)$. While these value functions are foundational to the MDP framework, not all reinforcement learning algorithms use them directly; some methods (e.g., policy gradient approaches) operate without explicitly estimating $V_\pi$ or $Q_\pi$.

The advantage function then captures how much better (or worse) it is to take a particular action in a given state, relative to the policy's typical expected outcome from that state. Formally, it is defined as the difference between the action-value and state-value functions:

$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s) \tag{2.6}$$

Positive advantage indicates that choosing action $a$ in state $s$ leads to a higher expected return than the policy's baseline behaviour at $s$, while negative advantage indicates that the action is expected to perform worse.

Reinforcement learning problems are typically modelled as discrete-time MDPs, where transitions occur at fixed time steps $t$. In contrast, continuous-time MDPs [25, pp. 13–16] allow transitions to occur at any point in continuous time and are more appropriate for domains with high temporal resolution or continuous control, although they are less common in standard RL settings due to their increased mathematical complexity.

In the context of episodic reinforcement learning, interaction is structured into episodes $e \in \{1, \ldots, E\}$ of finite length $T$, after which the environment resets. Each episode begins from an initial state distribution and ends in a terminal state (accepting or non-accepting). The return is computed per episode, and the agent's objective is to maximise the expected return across each episode. This contrasts with continuing tasks, where interaction is unbounded and the agent learns policies to maximise return over an indefinite horizon.

Finally, regret is a fundamental measure of learning efficiency [26]. It quantifies the difference between the cumulative reward that could have been obtained by always acting optimally, and the actual reward accumulated by the agent over time. The total reward of $T$ timesteps is defined as:

$$\text{Regret}(T) = \sum_{t=1}^{T} [V_{\pi*}(s_t) - V_\pi(s_t)] \tag{2.7}$$

In episodic reinforcement learning, where the environment resets after each episode, it is common to express regret in terms of the number of episodes $E$. Many algorithms aim to achieve sublinear regret, meaning that total regret grows slower than linearly in $E$, typically at a rate of $\mathcal{O}(\sqrt{E})$. This implies that the average regret per episode decays as $\mathcal{O}(\sqrt{E})$, indicating that the agent is learning to behave optimally as it gains more experience.

#### 2.1.1.2 Model-Free RL

Model-free reinforcement learning algorithms are a category of RL algorithms that aim to find optimal or near-optimal policies directly from interactions with the environment, without explicitly learning or using a model of environment dynamics. Unlike model-based algorithms (discussed in Section 2.1.1.3), model-free methods do not attempt to estimate transition probabilities $P(s_{t+1}|s_t, a_t)$ or reward functions explicitly. Instead, they directly estimate value functions, policy parameters, or both, based solely on the observed states, actions, and rewards encountered through experience.

The primary advantage of model-free algorithms is their simplicity and ability to scale to complex, high-dimensional tasks, as they require fewer assumptions about the underlying dynamics. However, they typically require significant interaction with the environment, leading to potentially slower convergence compared to model-based methods that use learnt or known dynamics for planning.

##### 2.1.1.2.1 Value-Based Methods

Value-based model-free algorithms aim to learn an optimal policy by approximating the action-value function $Q_\pi(s, a)$. The optimal policy can then be derived by exploiting the learned value estimates.

The core idea underpinning value-based algorithms is temporal difference (TD) learning, which enables agents to update their value estimates incrementally based on the observed rewards and the next state's estimated value. TD learning bridges the gap between Monte Carlo methods (which wait until the end of an episode to update) and dynamic programming (which requires a known model). It works by bootstrapping: updating estimates based on other learned estimates.

At each timestep $t$, the agent observes a transition $(s_t, a_t, r, s_{t+1})$, and updates its action-value estimate for $Q_\pi(s_t, a_t)$. This update is driven by the TD error, which quantifies the mismatch between the current value estimate and a one-step return estimate known as the TD target. The TD error is defined as:

$$\upsilon_t = \underbrace{r_t + \gamma Q(s_{t+1}, a_{t+1})}_{\text{TD target}} - Q(s_t, a_t) \tag{2.8}$$

The value function can then be updated $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta \cdot \upsilon_t$ where $\eta \in (0, 1]$ is the learning rate.

Q-learning [27] is the foundational value-based algorithm. It is an off-policy algorithm, meaning that it estimates the value of the optimal policy independently of the agent's behaviour during training. In Q-learning, the TD target uses the maximum action-value over all possible actions in the next state:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta \left[ r_t + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t) \right] \tag{2.9}$$

Here, the TD target represents the maximum possible return obtainable from state $s_{t+1}$ under the current value estimates. This enables the agent to learn an optimal policy even when its current behaviour includes exploration, such as under an $\epsilon$-greedy policy [28]. Q-learning is guaranteed to converge to the optimal action-value function under suitable conditions, such as sufficient exploration and a decaying learning rate, provided the environment is finite and discrete. However, Q-learning can suffer from overestimation bias due to the use of the max operator, particularly in stochastic environments.

SARSA (State-Action-Reward-State-Action) [29] is an on-policy algorithm. It updates the value function based on the action that is actually taken by the policy in the next state, rather than the one that would be selected greedily. The SARSA update is given by:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta \left[ r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right] \tag{2.10}$$

In contrast to Q-learning, SARSA includes $Q(s_{t+1}, a_{t+1})$ in the TD target to reflect the value of the next action. Since SARSA incorporates the behaviour of the actual policy during training, it learns the value of that specific policy, including any stochastic or exploratory components. This can lead to more cautious policies, especially in domains where exploratory actions may lead to unsafe or highly suboptimal outcomes. SARSA tends to be more stable than Q-learning in noisy environments, but it may converge more slowly since it does not always optimise with respect to the best possible future action.

A related method is Expected SARSA [30], which attempts to reduce the variance in value updates while retaining the benefits of an on-policy learning scheme. Rather than using the sampled action $a_{t+1}$ in the next state, Expected SARSA calculates the expected value of the action-value function under the current policy's distribution. The update rule is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta \left[ r_t + \gamma \sum_{a \in A} \pi(a|s_{t+1}) Q(s_{t+1}, a) - Q(s_t, a_t) \right] \tag{2.11}$$

The TD target is replaced with the expected value over all possible actions in the next state, weighted by the policy $\pi$. Expected SARSA, therefore, smooths the target estimate and reduces the variance introduced by stochastic action selection, while still learning on-policy. The method can be viewed as a bridge between SARSA and Q-learning, as it retains the on-policy nature of SARSA while incorporating the full action distribution, similar to the optimality-seeking behaviour of Q-learning.

Expected SARSA is particularly useful when dealing with discrete action spaces where summing over all actions is computationally tractable. In large or continuous action spaces, the summation can become infeasible, and the algorithm may require approximation methods to remain practical.

While tabular methods, such as the ones discussed, are effective in environments with small, discrete state-action spaces, they become infeasible in high-dimensional domains where explicitly

storing a table of values for each state-action pair is computationally prohibitive. To address this limitation, Deep Q-Networks (DQN) [1] approximate the action-value function $Q(s, a)$ using a deep neural network $Q_\theta(s, a)$ with parameters $\theta$. The algorithm introduces two key stabilisation techniques to address the instability that arises when combining temporal difference learning with deep networks. First, DQN maintains a separate target network with parameters $\theta^-$ and is used to compute the TD target. Specifically, the DQN target is:

$$y_t = r_t + \gamma \max_{a \in A} Q_{\theta^-}(s_{t+1}, a) \tag{2.12}$$

The parameters of the target network $\theta^-$ are periodically updated by copying from the online network $\theta^- \leftarrow \theta$. This modification reduces the risk of divergence by decoupling the target from the rapidly changing network being trained. Second, DQN employs an experience replay buffer, which stores a large number of past transitions ($s_t$, $a_t$, $r_t$, $s_{t+1}$). During training, the algorithm samples mini-batches uniformly from this buffer to perform updates, rather than relying on consecutive samples. This breaks the strong temporal correlations present in sequential interaction and improves data efficiency by reusing past experiences multiple times.

The neural network is trained by minimising the squared TD error over the sampled mini-batch:

$$\mathcal{L}_{\text{DQN}}(\theta) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}_{\text{buffer}}} \left[ (y_t - Q_\theta(s_t, a_t))^2 \right] \tag{2.13}$$

where $\mathcal{D}_{\text{buffer}}$ denotes the experience replay buffer. Through this formulation, DQN updates its value function using stochastic gradient descent, allowing it to generalise across similar states rather than learning values independently as in tabular methods.

Despite its success, DQN exhibits several limitations. The use of the max operator in the TD target can lead to overestimation of action values, particularly in stochastic settings. This issue motivated variants such as Double DQN [31], which decouples the action selection and action evaluation steps to reduce bias. Other improvements, such as Duelling DQN [32] and Prioritised Experience Replay [33], further enhance performance and stability. Nevertheless, the original DQN remains a foundational method in RL, enabling value-based methods to scale to high-dimensional observation spaces.

#### 2.1.1.2.2 Policy-Based Methods

While value-based model-free algorithms aim to estimate action-value functions and derive policies indirectly by acting greedily with respect to those estimates, policy-based model-free methods take a fundamentally different approach: they directly parameterise and optimise the policy itself. The agent's behaviour is represented as a probability distribution over actions given states $\pi_\theta(s, a)$, where $\theta$ is the parameters of a differentiable policy function (e.g., a neural network, linear function, etc.). Let Trace $= (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$ denote a trajectory sampled from the environment under policy $\pi_\theta$. The objective is defined as:

$$J(\theta) = \mathbb{E}_{\text{Trace}\sim\pi_\theta}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right] \tag{2.14}$$

Unlike value-based methods, which rely on greedy action selection, policy-based methods are naturally suited to environments with continuous or stochastic action spaces, and they allow for the direct incorporation of exploration via stochastic policies.

The base approach to optimising the objective $J(\theta)$ is via gradient ascent, using the policy gradient theorem [34]. The update rule is given by:

$$\theta \leftarrow \theta + \eta\left[\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t|s_t)\cdot(\Psi_t)\right] \tag{2.15}$$

where $\Psi_t$ can be any return estimator such as the total reward of the trajectory, the state-action value function, etc. This encourages the agent to update its policy in the direction that increases the probability of actions that led to higher returns. This algorithm is referred to as vanilla policy gradient. Despite its simplicity and generality, this method suffers from high variance in the gradient estimates, leading to unstable or slow learning, especially in sparse reward settings.

REINFORCE [35] is a Monte Carlo implementation of the vanilla policy gradient algorithm. It updates the policy parameters using complete episodes by computing the cumulative return from each timestep onward. The update rule is:

$$\theta \leftarrow \theta + \eta\left[\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t|s_t)\cdot(\sum_{i=t}^{T}\gamma^{i-t}r_i)\right] \tag{2.16}$$

This Monte Carlo estimator of return is unbiased but introduces significant variance due to the dependence on full trajectories. One common technique to reduce variance without introducing bias is the use of a baseline, typically the value function $V_\pi(s_t)$, which is subtracted from the return. This update rule becomes:

$$\theta \leftarrow \theta + \eta\left[\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t|s_t)\cdot(\sum_{i=t}^{T}\gamma^{i-t}r_i - V_\pi(s_t))\right] \tag{2.17}$$

The inclusion of this baseline maintains an unbiased gradient estimator but helps stabilise training by normalising the signal used to reinforce actions.

To address limitations in stability and sample efficiency, Trust Region Policy Optimisation (TRPO) [36], which constrains the step size in policy space to ensure monotonic improvement, was developed. Instead of performing standard gradient ascent, TRPO solves a constrained optimisation problem:

$$\begin{aligned} \theta \leftarrow \arg\max_{\theta} \mathbb{E}_{(s_t,a_t)\sim\pi_{\theta_{\text{old}}}} \left[ \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} Q_{\pi_{\theta_{\text{old}}}}(s_t,a_t) - V_{\pi_{\theta_{\text{old}}}}(s_t) \right] \\ \text{subject to} \quad \mathbb{E}_{s_t\sim\pi_{\theta_{\text{old}}}} \left[ \text{KL}\left[ \pi_{\theta_{\text{old}}}(\cdot|s_t) || \pi_\theta(\cdot|s_t) \right] \right] \leqslant \epsilon_{\text{KL}} \end{aligned} \tag{2.18}$$

The optimisation objective ensures that the new policy improves performance by increasing the expected advantage while remaining close to the current policy in distributional terms. The KL-divergence [37] constraint prevents the updated policy from deviating too drastically from the previous one, thereby avoiding large, destabilising updates. Although this constraint offers theoretical guarantees of monotonic policy improvement, it introduces additional computational complexity due to the need to solve a constrained optimisation problem at each iteration approximately.

To make trust-region methods more scalable and practical, Proximal Policy Optimisation (PPO) [38] was introduced as a first-order approximation to TRPO. PPO replaces the hard KL constraint with a clipped surrogate objective that penalises updates leading to large policy changes. The most common objective form used in PPO is:

$$\begin{aligned} \theta \leftarrow \arg\max_{\theta} \mathbb{E}_{(s_t,a_t)\sim\pi_{\theta_{\text{old}}}} \Big[ \min \Big( & \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} Q_{\pi_{\theta_{\text{old}}}}(s_t,a_t) - V_{\pi_{\theta_{\text{old}}}}(s_t), \\ & \text{clip}\left( \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) \cdot Q_{\pi_{\theta_{\text{old}}}}(s_t,a_t) - V_{\pi_{\theta_{\text{old}}}}(s_t) \Big) \Big] \end{aligned} \tag{2.19}$$

This objective discourages large deviations from the current policy by flattening the optimisation landscape outside the interval $[1-\epsilon, 1+\epsilon]$. PPO is widely used in modern RL applications due to its empirical robustness, ease of implementation, and strong performance across a range of continuous control tasks.

#### 2.1.1.2.3 Actor-Critic Methods

Actor-Critic (AC) methods represent a class of model-free reinforcement learning algorithms that combine the strengths of both value-based and policy-based approaches [39]. Unlike pure value-based methods (e.g., Q-Learning, SARSA, etc.) that learn value functions to derive a policy indirectly, or pure policy-based methods (e.g., TRPO, PPO, etc.) that optimise the policy directly, actor-critic methods maintain two distinct components: an actor, which explicitly parameterises the policy $\pi_{\theta_{\text{actor}}}(a|s)$, and a critic, which estimates value functions such as the state-value function $V_{\theta_{\text{critic}}}(s)$ or the action-value function $Q_{\theta_{\text{critic}}}(s,a)$, where $\theta_{\text{actor}}$ and $\theta_{\text{critic}}$ denote the parameters of the actor and critic networks, respectively.

The actor is responsible for selecting actions according to the current policy. At the same time, the critic evaluates the quality of these actions by providing a low-variance but possibly biased estimate of the expected return. This dual representation enables actor-critic methods to leverage the efficiency of policy gradients and the stability of value function learning.

A generic actor-critic update proceeds as follows. Given a trajectory Trace $= (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$ the actor is updated via the policy gradient:

$$\theta_{\text{actor}} \leftarrow \theta_{\text{actor}} + \eta \sum_{t=0}^{T} \nabla_{\theta_{\text{actor}}} \log \pi_{\theta_{\text{actor}}}(a_t|s_t) \cdot (Q_{\theta_{\text{critic}}}(s_t, a_t) - V_{\theta_{\text{critic}}}(s_t)) \tag{2.20}$$

Simultaneously, the critic parameters $\theta_{\text{critic}}$ are updated by minimising a regression loss between predicted values and bootstrapped targets:

$$\mathcal{L}_{\text{critic}}(\theta_{\text{critic}}) = (r_t + \gamma V_{\theta_{\text{critic}}}(s_{t+1}) - V_{\theta_{\text{critic}}}(s_t))^2 \tag{2.21}$$

This combination enables efficient policy improvement and value estimation simultaneously. The vanilla actor-critic algorithm, while conceptually sound, often suffers from high variance in policy gradients and instability during training, particularly in environments with sparse rewards. The Advantage Actor-Critic (A2C) [40] algorithm improves upon this by explicitly estimating the advantage function using the TD residual. This reduces variance compared to using full Monte Carlo returns. The actor update becomes:

$$\theta_{\text{actor}} \leftarrow \theta_{\text{actor}} + \eta \sum_{t=0}^{T} \nabla_{\theta_{\text{actor}}} \log \pi_{\theta_{\text{actor}}}(a_t|s_t) \cdot (r_t + \gamma V_{\theta_{\text{critic}}}(s_{t+1}) - V_{\theta_{\text{critic}}}(s_t)) \tag{2.22}$$

A2C further reduces variance by performing synchronous updates from multiple parallel environments, averaging gradients across workers. This stabilises learning and improves sample efficiency compared to the original actor-critic framework.

Despite A2C's improvements, it still requires synchronous computation, which may be inefficient or impractical in large-scale settings. To overcome this, the Asynchronous Advantage Actor-Critic (A3C) [40] algorithm introduces asynchronous parallelism. Multiple agents operate in independent environment instances, each maintaining a local copy of $\theta_{\text{actor}}$ and $\theta_{\text{critic}}$. After collecting trajectories, each worker computes gradients locally and applies them asynchronously to shared global parameters. Unlike A2C, which uses one-step bootstrapping, A3C uses n-step returns to improve learning signals. The actor update becomes:

$$\theta_{\text{actor}} \leftarrow \theta_{\text{actor}} + \eta \sum_{t=0}^{T} \nabla_{\theta_{\text{actor}}} \log \pi_{\theta_{\text{actor}}}(a_t|s_t) \cdot ((\sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n V_{\theta_{\text{critic}}}(s_{t+n})) - V_{\theta_{\text{critic}}}(s_t)) \tag{2.23}$$

Asynchrony reduces correlation between updates, improving stability and exploration, and it enables A3C to scale efficiently across multiple CPU cores.

A limitation of both A2C and A3C is their reliance on stochastic policies, which may be inefficient in continuous action spaces. The Deep Deterministic Policy Gradient (DDPG) [41] algorithm addresses this by adopting a deterministic actor $\mu_{\theta_{\text{actor}}}(s)$ that maps states directly to actions. This allows DDPG to avoid sampling from a stochastic policy during training and inference,

improving performance in high-dimensional continuous domains. The critic approximates the action-value function $Q_{\theta_{\text{critic}}}(s, a)$, and is trained using the loss:

$$\mathcal{L}_{\text{critic}}(\theta_{\text{critic}}) = \mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim\mathcal{D}_{\text{buffer}}}\left[((r_t + \gamma Q_{\theta'_{\text{critic}}}(s_{t+1}, \mu_{\theta'_{\text{actor}}}(s_{t+1})) - Q_{\theta_{\text{critic}}}(s_t, a_t))^2\right] \tag{2.24}$$

where $\theta'_{\text{actor}}$ and $\theta'_{\text{critic}}$ are target network parameters. The actor is updated by maximising the critic's output:

$$\theta_{\text{actor}} \leftarrow \theta_{\text{actor}} + \eta\nabla_{\theta_{\text{actor}}} Q_{\theta_{\text{critic}}}(s, \mu_{\theta_{\text{actor}}}(s)) \tag{2.25}$$

While effective, DDPG suffers from instability and overestimation bias in the Q-function. To address DDPG's limitations, Twin Delayed DDPG (TD3) [42] introduces three modifications to the base DDPG algorithm. Firstly, it maintains a second critic network and then utilises the minimum value from the two in an attempt to reduce overestimation bias. TD3 adds noise to the target actions to regularise the critic. Lastly, it updates the actor less frequently than the critic in an attempt to improve stability. These changes significantly improve performance in continuous control settings.

Finally, the Soft Actor-Critic (SAC) [43] algorithm addresses the issue of TD3 not explicitly encouraging exploration beyond the use of added noise during training by augmenting the objective with an entropy term to encourage more diverse action selection. This formulation allows the agent to remain highly exploratory, particularly during the early stages of training, while gradually focusing on high-reward behaviours as learning progresses.

More formally, SAC seeks to optimise the expected sum of rewards while also maximising the entropy of the policy $\pi_{\theta_{\text{actor}}}$. The entropy term $\mathcal{H}(\pi_{\theta_{\text{actor}}}(\cdot|s_t))$ measures the randomness of the policy's action distribution, providing an intrinsic incentive to avoid premature convergence to deterministic or suboptimal policies.

Just like TD3, SAC maintains two critics alongside a separate target value function network. The critics are updated by minimising the Bellman residual with a soft value target that incorporates both estimated return and policy entropy. The critic parameters can then be updated as follows:

$$\theta_{\text{critic}} \leftarrow \theta_{\text{critic}} - \eta\nabla_{\theta_{\text{critic}}}\Bigg(r_t + \gamma\mathbb{E}_{a_{t+1}\sim\pi_{\theta_{\text{actor}}}}\big[\min\left(Q_{\theta_{\text{critic1}}}(s_{t+1}, a_{t+1}), Q_{\theta_{\text{critic2}}}(s_{t+1}, a_{t+1})\right) - \lambda_{\text{entropy}}\log\pi_{\theta_{\text{actor}}}(a_{t+1} \mid s_{t+1})\big] - Q_{\theta_{\text{critic}}}(s_t, a_t)\Bigg)^2 \tag{2.26}$$

The actor is trained to maximise both expected action-value functions and the entropy term, effectively learning to balance exploitation of rewarding actions and exploration of uncertain alternatives. The policy update is given by:

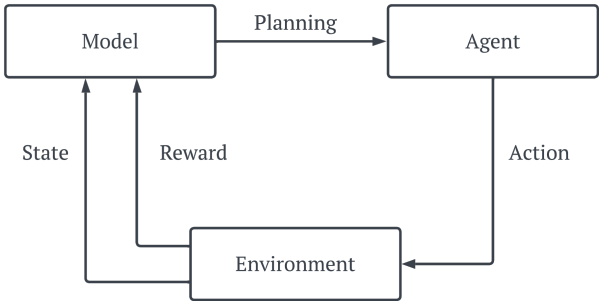



Figure 2.3: A typical model-based reinforcement learning algorithm.

$$\theta_{\text{actor}} \leftarrow \theta_{\text{actor}} + \eta \nabla_{\theta_{\text{actor}}} \mathbb{E}_{a \sim \pi_{\theta_{\text{actor}}}(\cdot|s)} \left[ \lambda_{\text{entropy}} \log \pi_{\theta_{\text{actor}}}(a \mid s) - Q_{\theta_{\text{critic}}}(s, a) \right] \tag{2.27}$$

This contributes to improved sample efficiency compared to purely on-policy algorithms such as A2C and A3C. Additionally, the inclusion of entropy regularisation yields policies that are more robust to local optima and hyperparameter sensitivity, making SAC highly effective across a wide range of continuous control tasks [44].

#### 2.1.1.3 Model-Based RL

Model-based reinforcement learning algorithms explicitly construct or utilise a model of the environment dynamics. Unlike model-free algorithms, model-based algorithms aim to predict future states and rewards by estimating transition probabilities and reward functions [45]. By leveraging this learned or predefined model, these algorithms can simulate or plan ahead, often requiring fewer interactions with the actual environment. A typical model-based algorithm can be visualised in Figure 2.3. This approach can significantly improve sample efficiency, accelerating learning, particularly in scenarios where collecting real-world data is costly or impractical.

A central challenge in model-based RL is obtaining a sufficiently accurate model of the environment. Errors in the learned model, known as model-bias, can lead to suboptimal policies, as inaccuracies accumulate when the agent simulates future trajectories. Therefore, balancing model accuracy and computational feasibility is critical to the success of model-based RL algorithms.

##### 2.1.1.3.1 Planning-Based Methods

Planning-based model-based reinforcement learning algorithms exploit full knowledge of the environment's transition dynamics $P$ and reward function $R$. In contrast to model-free approaches that learn directly from experience or model-based learning methods that first learn an approximate model, planning-based algorithms assume the model is given a priori. This enables the agent to perform explicit planning via dynamic programming to derive optimal policies [46].

A foundational approach is Value Iteration [4], which operates by directly applying the Bellman optimality operator iteratively to converge to the optimal state-value function $V^*(s)$. The process begins with an arbitrary value function $V_0(s)$, which may be initialised to zero or any

bounded estimate over the state space. At each iteration/timestep $t$, the update applies the Bellman backup:

$$V_{t+1}(s) \leftarrow \max_{a \in A} \left[ R(s,a) + \gamma \sum_{s_{t+1} \in S} P(s_{t+1}|s,a) V_t(s_{t+1}) \right] \tag{2.28}$$

Each iteration updates the value estimate of every state by using the current estimates of the future states' values, gradually improving the accuracy of the estimates. Because this update process always reduces any errors by a constant factor less than one (specifically $\gamma$), it is guaranteed to eventually converge to the unique optimal value function that solves the Bellman equation. Convergence is typically declared once the maximum difference between successive value function estimates satisfies:

$$\max_{s \in S} |V_{t+1}(s) - V_t(s)| < \epsilon_{\text{conv}} \tag{2.29}$$

ensuring that further updates will not significantly alter the policy. Once convergence is reached, the optimal policy $\pi^*$ can be derived by selecting the action that achieves the maximum in the Bellman update for each state:

$$\pi^*(s) \leftarrow \arg\max_{a \in A} \left[ R(s,a) + \gamma \sum_{s_{t+1} \in S} P(s_{t+1}|s,a) V^*(s_{t+1}) \right] \tag{2.30}$$

This procedure yields a deterministic policy that specifies, for every state, the action that maximises the expected discounted return assuming perfect knowledge of the environment dynamics. While Value Iteration is simple to implement and guarantees convergence in finite state-action spaces, it requires repeatedly sweeping over the entire state space until the value function stabilises, which can be computationally expensive in domains with large or continuous state spaces.

In contrast to Value Iteration, which updates value estimates in each iteration before deriving a policy, Policy Iteration [4] maintains and explicitly improves a candidate policy at each step. Policy Iteration begins with an initial policy $\pi_0$ and alternates between two phases: policy evaluation and policy improvement. In the policy evaluation phase, the value function $V_{\pi_t}(s)$ corresponding to the current policy $\pi_t$ is computed by solving the Bellman expectation equation:

$$V_{\pi_t}(s) = R(s, \pi_t(s)) + \gamma \sum_{s_{t+1} \in S} P(s_{t+1}|s, \pi_t(s)) V_{\pi_t}(s_{t+1}) \tag{2.31}$$

This evaluation is performed iteratively until the value function stabilises within a convergence threshold $\epsilon_{\text{conv}}$. Once the value function for the current policy has been sufficiently approximated, the policy improvement phase updates the policy by selecting the action that maximises the expected return using the newly computed value estimates:

$$\pi_{t+1}(s) \leftarrow \arg\max_{a \in A} \left[ R(s,a) + \gamma \sum_{s_{t+1} \in S} P(s_{t+1}|s,a) V_{\pi_t}(s_{t+1}) \right] \tag{2.32}$$

These phases repeat alternately, with each iteration refining the policy towards optimality. The procedure terminates once the policy stabilises, meaning that no further improvements can be made and $\pi_{t+1} = \pi_t$. Policy Iteration typically requires fewer iterations to converge compared to Value Iteration, as the explicit policy improvement step can produce larger advances in performance at each stage. However, this comes at the cost of the policy evaluation phase, which may require solving a system of equations or performing many value updates before proceeding to the next policy improvement step. As with Value Iteration, the method is guaranteed to converge in finite state-action spaces when the model is known exactly.

Both Value Iteration and Policy Iteration compute stationary policies offline, producing a policy that can then be executed throughout an episode. In contrast, Model Predictive Control (MPC) [47] takes a different approach by planning online at every timestep. Rather than computing a complete policy in advance, MPC repeatedly solves a finite-horizon optimisation problem using the known environment model to determine the optimal sequence of actions from the current state. Specifically, at each timestep $t$, MPC solves:

$$\begin{gathered} \max_{a_{0:\text{horizon}-1}} \mathbb{E}\left[ \sum_{i=0}^{\text{horizon}-1} \gamma^i R(s_i, a_i) + \gamma^{\text{horizon}} V_{\text{terminal}}(s_{\text{horizon}}) \right] \\ \text{subject to} \quad s_{i+1} \sim P(s_{i+1}|s_i, a_i) \end{gathered} \tag{2.33}$$

where horizon is the planning horizon and $V_{\text{terminal}}$ is a terminal value function that estimates the value beyond the planning horizon. The optimisation produces a sequence of actions $\{a_0^*, \ldots, a_{\text{horizon}-1}^*\}$ , but only the first action $a_0^*$ is executed. After observing the next state $s_{t+1}$, the agent re-plans from this updated state, ensuring that the plan remains responsive to new information or disturbances in the environment.

MPC provides a flexible and powerful planning mechanism that is particularly well-suited to continuous control domains and systems where conditions may change rapidly. By repeatedly solving a short-term planning problem, MPC allows the agent to adapt to new observations at each timestep. However, this advantage comes at the cost of computational demand, as solving the finite-horizon optimisation problem can be challenging in real-time applications, especially in high-dimensional or complex environments.

#### 2.1.1.3.2 Learning-Based Methods

Planning-based model-based reinforcement learning methods assume full knowledge of the transition dynamics $P$ and reward function $R$. Alternatively, learning-based approaches instead attempt to approximate these functions directly from interaction data. This paradigm seeks to construct predictive models of environment dynamics to enable planning without requiring prohibitively large numbers of real-world episodes. Learning-based methods differ substantially in

how they represent the environment model, how they integrate planning or policy optimisation, and how they address the central challenge of model bias, where prediction errors compound over imagined rollouts.

The foundational approach is the Dyna architecture [48], which can be interpreted as an early instantiation of the idea of augmenting model-free learning with synthetic experience. Dyna-Q maintains an action-value function $Q(s, a)$ updated via standard temporal-difference learning and incrementally learns an approximate model $\hat{P}_\theta$ to predict the next state. At each timestep $t$, an experience tuple $(s_t, a_t, r_t, s_{t+1})$ is used to both improve the action-value function and train the model by minimising the prediction error:

$$\mathcal{L}_{\text{model}}(\theta) = ||\hat{P}_\theta(s_t, a_t) - s_{t+1}||^2 \tag{2.34}$$

Once this model is sufficiently accurate, Dyna-Q interleaves real environment updates with planning updates, in which synthetic transitions $(\hat{s_t}, \hat{a_t}, \hat{r_t}, \hat{s_{t+1}})$ are generated by sampling from the learned model. The action-value function is then updated on these imagined transitions:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta \left[\hat{r}_t + \gamma \max_a Q(\hat{s}_{t+1}, a) - Q(s_t, a_t)\right]. \tag{2.35}$$

By combining real and imagined experience, Dyna-Q demonstrates how approximate models can be learnt from experience to reduce the need for prior knowledge. However, it was primarily applied to tabular domains with discrete state-action spaces. Extending these ideas to high-dimensional continuous environments requires function approximators with sufficient representational capacity. This motivated the development of Model-Based Policy Optimisation (MBPO) [49], which uses deep neural networks to model dynamics in continuous control tasks. MBPO learns an ensemble of probabilistic dynamics models, each parameterised by independent weights $\theta_j$, to capture epistemic uncertainty about the environment:

$$\hat{P}_\theta(s_{t+1}|s_t, a_t) = \mathcal{N}(\mu_{\theta_j}(s_t, a_t), \text{Cov}_{\theta_j}(s_t, a_t)) \tag{2.36}$$

Unlike Dyna-Q, which performs synthetic rollouts of arbitrary length, MBPO explicitly restricts synthetic trajectory lengths to a small horizon (typically 1–5 steps). This short-horizon constraint mitigates error accumulation while still providing a rich stream of imagined transitions. At each iteration, synthetic rollouts are generated by sampling the ensemble models, and the resulting transitions are stored in a model buffer $\mathcal{D}_{\text{model}}$. Policy and value functions are then updated via an off-policy RL algorithm such as Soft Actor-Critic (SAC). Concretely, the actor parameters $\theta_{\text{actor}}$ are optimised to maximise the expected Q-value across both real and synthetic experience:

$$\theta_{\text{actor}} \leftarrow \theta_{\text{actor}} + \eta \nabla_{\theta_{\text{actor}}} \mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{real}}\cup\mathcal{D}_{\text{model}}} \left[Q_{\theta_{\text{critic}}}(s, a)\right] \tag{2.37}$$

This design highlights a key evolution from Dyna-Q: rather than indiscriminately planning over long horizons, MBPO demonstrates that learning-based planning can be stable and effective when constrained to well-calibrated short rollouts.

MBPO augments its replay buffer with very short model-generated rollouts produced directly in the environment's state space. That works well when the state is a compact vector, but it becomes impractical for raw pixels. Accurately predicting future pixel arrays over even a handful of steps is notoriously hard. Dreamer [50] tackles that bottleneck differently: it first trains a recurrent latent-dynamics model that encodes each image/observation into a low-dimensional latent vector and predicts forward entirely in that latent space. During policy optimisation the agent *imagines* long sequences of latent states and rewards, computing gradients through them to improve its actor–critic networks, and only decodes pixels as an auxiliary training loss. Because control never depends on reconstructing future images, Dreamer scales far better to high-dimensional observation environments while retaining data-efficiency. Subsequent iterations, DreamerV2 [51] and DreamerV3 [52], refine this approach by improving the expressiveness of the latent dynamics model, stabilising training, and introducing scalable exploration strategies.

## 2.2 Prior Research

In this section, we review prior research on UQ and safe RL, focusing on methods most closely related to the contributions of this thesis. For UQ, we summarise representative approaches for modelling aleatoric and epistemic uncertainty. For safe RL, we outline the main families of techniques for enforcing constraints during learning, including cost-based formulations, knowledge-based safety mechanisms, and black-box approaches.

### 2.2.1 Uncertainty Quantification

Uncertainty Quantification (UQ) is the process of understanding and quantifying the uncertainty in a model's predictions [53]. In deep learning, uncertainty is typically categorised into epistemic and aleatoric uncertainty [54].

Epistemic, or model, uncertainty arises due to a lack of knowledge about the data during training or prediction and is reducible given additional data. Factors contributing to epistemic uncertainty include limited training data resulting in only partial coverage of the data distribution, out-of-distribution (OOD) inputs that differ significantly from the training data, adversarial perturbations that deliberately push inputs into regions near the decision boundary, covariate shift between training and deployment distributions, and mismatches between model complexity and the true underlying function, where either an overly simple architecture fails to capture relevant patterns or an overly complex one overfits and becomes overconfident.

Aleatoric, or data, uncertainty is inherent to the data-generating process and is irreducible regardless of the amount of additional data collected. This uncertainty can be caused by measurement noise during data acquisition, class overlap where instances share features with multiple

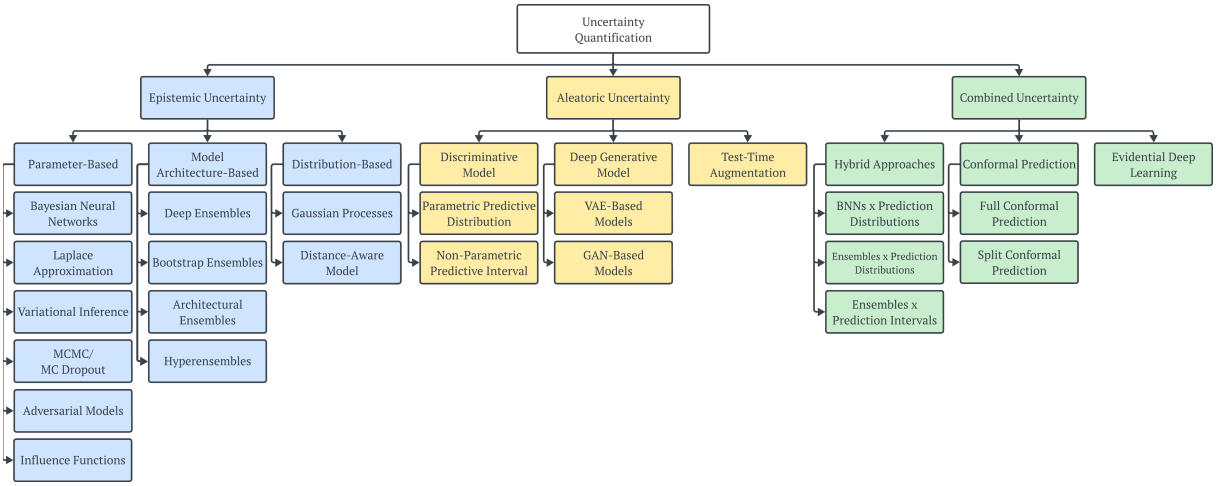



Figure 2.4: A classification of techniques for uncertainty quantification adapted from [55] decomposed into three categories: methods that quantify model uncertainty, methods that quantify data uncertainty, and methods that can model both.

classes, label ambiguity due to inconsistencies among human annotators, and heteroscedasticity where different inputs exhibit varying levels of noise.

Both types of uncertainty can coexist, and it is often necessary to quantify and address them jointly to improve model performance. Accordingly, the research landscape on UQ can be organised into approaches that primarily quantify model uncertainty, those that address data uncertainty, and methods that integrate both within a unified framework [55], as illustrated in Figure 2.4.

#### 2.2.1.1 Epistemic Uncertainty-Based Methods

Epistemic uncertainty-based methods refer to techniques that explicitly model the uncertainty arising from limited knowledge about the data-generating process and the model's parameters. Methods include: parameter-based methods focus on representing uncertainty through probabilistic modelling of parameters [56–70], model architecture-based methods estimate uncertainty by aggregating predictions from multiple models or architectural variants [71–74], and distribution-based methods quantify uncertainty by considering the distance between inputs and training data or by modelling predictive distributions over functions [75–80].

##### 2.2.1.1.1 Parameter-Based Methods

The first way to estimate epistemic uncertainty within deep learning models is to consider a distribution over the model parameters rather than treating them as deterministic.

This approach is formalised through Bayesian inference, where the posterior distribution over model parameters $\theta$ given observed data $\mathcal{D}$ is computed using Bayes' rule [56]:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int p(\mathcal{D}|\theta)p(\theta)d\theta}. \tag{2.38}$$

Here, $p(\mathcal{D}|\theta)$ is the likelihood of the data given the model parameters, $p(\theta)$ is the prior belief about $\theta$, and $p(\mathcal{D})$ is the marginal likelihood. As seen in Equation 2.38, the marginal likelihood involves an integral over all possible parameter values, making exact Bayesian inference computationally intractable for complex models.

Bayesian Neural Networks (BNNs) are a specific realisation of Bayesian inference applied to neural networks, in which weights and biases are treated as probability distributions rather than fixed values [57]. By maintaining distributions over weights, BNNs naturally quantify epistemic uncertainty. For a given data point $x \in X$ and ground truth labels $y \in Y$, BNNs produce a predictive distribution by integrating over these parameter distributions:

$$p(y|x, \mathcal{D}) = \int p(y|x, \theta)\ p(\theta|\mathcal{D})\ d\theta \tag{2.39}$$

Due to the complexity of neural networks, this integral is typically intractable. Thus, exact Bayesian inference in deep neural networks is impractical, motivating the need for approximation techniques such as those discussed next.

Laplace Approximation (LA), introduced by MacKay [58], provides a computationally efficient approximation to the posterior distribution by assuming it can be represented as a Gaussian distribution centered at the maximum a posteriori (MAP) estimate $\theta^*$:

$$p(\theta|\mathcal{D}) \approx \mathcal{N}\left(\theta \,\middle|\, \theta^*, \left[-\nabla^2 \big(\log p(\mathcal{D}|\theta) + \log p(\theta)\big)\big|_{\theta=\theta^*}\right]^{-1}\right) \tag{2.40}$$

While this approach allows epistemic uncertainty estimation without exact integration, it assumes a unimodal posterior distribution, which may limit accuracy.

To further improve scalability, Kronecker-factored LA [59] approximates the Hessian at the MAP estimate using a Kronecker product of smaller matrices. This significantly reduces computational complexity, enabling LA to scale to larger models.

Another refinement, the Linearised Laplace Approximation (LLA)[60], simplifies computations by approximating the model with a first-order Taylor expansion around the MAP estimate, effectively treating it as locally linear in parameter space. Deng et al. [61] further extend LLA by leveraging connections with neural tangent kernels, utilising Nyström approximation techniques to achieve efficient uncertainty estimation.

While Laplace Approximation and its extensions rely on local approximations of the posterior distribution around a single mode (the MAP estimate), alternative approaches aim to approximate the posterior more globally without assuming a Gaussian shape centred at a point estimate. One such class of methods is Variational Inference (VI), which transforms the problem of approximating complex posterior distributions into an optimisation task [62]. In VI, the true posterior distribution $p(\theta_{\text{model}}|\mathcal{D})$ is approximated by a simpler, tractable distribution $q(\theta_{\text{model}}; \theta_{\text{var}})$. Here, $\theta_{\text{model}}$ denotes the model parameters (for example, the weights and biases of a neural network) that are treated as random variables, while $\theta_{\text{var}}$ represents the learnable variational parameters that define the shape of the approximate distribution. For instance, if $q$

is chosen to be a Gaussian, $\theta_{\text{var}}$ would correspond to its mean and covariance. The optimisation goal is to find the parameters $\theta_{\text{var}}$ that make $q(\theta_{\text{model}}; \theta_{\text{var}})$ as close as possible to the true posterior. Closeness, in this context, is measured using the Kullback–Leibler (KL) divergence between $q$ and $p$. Minimising this divergence is equivalent to maximising the evidence lower bound (ELBO), given by:

$$\mathcal{L}_{\text{VI}}(\theta_{\text{var}}) = \text{KL}\big(q(\theta_{\text{model}}; \theta_{\text{var}}) \,\|\, p(\theta_{\text{model}})\big) - \mathbb{E}_{q(\theta_{\text{model}}\,;\,\theta_{\text{var}})}\big[\log\, p(\mathcal{D} \,|\, \theta_{\text{model}})\big] \tag{2.41}$$

where $p(\theta_{\text{model}})$ is the prior distribution. By maximising the ELBO with respect to $\theta_{\text{var}}$, the approximate posterior $q(\theta_{\text{model}}; \theta_{\text{var}})$ can be obtained that is tractable and close to the true posterior.

Significant research has been conducted to enhance the usability and efficiency of VI. Employing stochastic approximations by using mini-batches can reduce the computational burden of the VI process, therefore increasing its scalability [63]. Black-box VI leverages Monte Carlo estimates of the ELBO's gradients to allow for inference without relying on model-specific derivations; this extends VI to be usable on a broader set of models [64]. Directly mapping observed data to variation parameters allows for efficient inference for faster posterior approximations [65].

Gal et al. [66] propose Monte Carlo Dropout, a method that uses dropout layers [81] of neural networks during inference and performs multiple stochastic forward passes to approximate the posterior distribution over the network's parameters $\theta$. The predictive distribution is then defined as the average of the stochastic outputs:

$$p(y|x, \mathcal{D}) \approx \frac{1}{N_{\text{mc}}} \sum_{n=1}^{N_{\text{mc}}} \mathcal{F}_{\theta_n}(x) \tag{2.42}$$

where $\mathcal{F}_{\theta_n}(x)$ is the output of the network $\mathcal{F}_\theta$ for an input $x$ with the Bernoulli dropout applied at inference, and $N_{\text{mc}}$ is the number of stochastic forward passes. The epistemic uncertainty is the variance of the forward passes. MC dropout provides a computationally efficient way to estimate epistemic uncertainty without significantly modifying the training process like previous methods; however, it is less expressive than other Bayesian methods and is computationally costly during inference time. Brach et al. [67] attempt to approximate the whole MC Dropout process in a single forward pass through moment propagation. This provides point and uncertainty estimates of the traditional MC Dropout process with much improved computational speeds.

Similar to MC Dropout, which employs Monte Carlo sampling, Markov Chain Monte Carlo (MCMC) provides an approach to posterior inference by generating samples directly from the posterior distribution [68]. MCMC constructs a Markov chain whose stationary distribution corresponds to the true posterior, ensuring asymptotic correctness as the number of samples increases. Unlike variational inference, which relies on a parametric approximation, MCMC estimates the posterior by producing a sequence of dependent samples $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(T)}$ that explore the parameter space. A widely used variant is the Metropolis-Hastings algorithm [82],

which iteratively proposes a candidate sample $\theta^*$ from a proposal distribution and accepts it with probability:

$$\min\left(1, \frac{p(\theta^*)\, q\big(\theta^{(t-1)} \mid \theta^*\big)}{p\big(\theta^{(t-1)}\big)\, q\big(\theta^* \mid \theta^{(t-1)}\big)}\right). \tag{2.43}$$

If the proposal is rejected, the current sample is retained, ensuring that the chain eventually converges to the true posterior distribution. Although MCMC methods can approximate the posterior more accurately than many alternatives, they often suffer from slow convergence and high computational cost.

Recently, adversarial models have been used to estimate epistemic uncertainty [69]. Unlike methods such as MC Dropout, which sample models based on posterior probability, adversarial models explicitly search for plausible outcomes of the data that produce significantly different predictions. This ensures that uncertainty estimates capture not only high-posterior models but also those that contribute most to the epistemic uncertainty integral. A reference model is a trained neural network that makes predictions for a given input. However, if multiple models explain the training data equally well but yield conflicting predictions at a specific test point, uncertainty should be high. Adversarial models are models that differ significantly from the reference model in their predictions at a test point while still having a high posterior probability. Formally, an adversarial model with parameters $\theta_{\text{adv}}$ is a perturbed version of the reference model's parameters $\theta$, chosen such that it remains a plausible explanation of the training data while producing significantly different predictions at a given test point. It satisfies:

$$|\log p(\theta \mid \mathcal{D}) - \log p(\theta_{\text{adv}} \mid \mathcal{D})| \leqslant \epsilon_{\text{plaus}}, \quad \text{and} \quad \text{div}(p(y \mid x, \theta), p(y \mid x, \theta_{\text{adv}})) \geqslant \epsilon_{\text{div}} \tag{2.44}$$

where $\epsilon_{\text{plaus}}$ controls the maximum allowable decrease in posterior plausibility, $\epsilon_{\text{div}}$ specifies the minimum divergence required for an adversarial model, and $\text{div}(\cdot, \cdot)$ measures how much the predictive distributions of the reference and adversarial models differ, ensuring that adversarial models challenge the reference model's prediction at the test point. By identifying these adversarial models, uncertainty estimation can be improved because they highlight areas where different plausible models disagree. Instead of relying solely on posterior sampling, Mixture Importance Sampling is used to refine uncertainty estimates by weighting adversarial models more heavily in the estimation process. This targeted search for high-uncertainty regions allows for a more robust quantification of epistemic uncertainty than previous methods.

Finally, influence functions provide a local, retraining-free way to quantify how sensitive a fitted model is to small perturbations of the training data. The core idea is to treat learning as an optimisation problem with an optimum $\theta^*$, then approximate how $\theta^*$ would change if a particular training example was upweighted or removed [83]. Classically, influence functions originate as the derivative of an estimator viewed as a function of the data-generating distribution. This sensitivity can be expressed as a linear response: if an example $x$ is upweighted by a small $\epsilon$, then the resulting parameter shift satisfies:

$$\left.\frac{d\theta_\varepsilon}{d\varepsilon}\right|_{\varepsilon=0} = -HS_{\theta^*}^{-1}\nabla_\theta \ell(x,\theta^*) \tag{2.45}$$

where $HS_{\theta*}$ is the regularised Hessian of the training objective at $\theta^*$. This captures infinitesimal robustness: points with large gradients under directions of low curvature produce large parameter responses. Koh and Liang [70] popularised this view by using the same linear-response derivation to attribute test-time behaviour to training data in differentiable models. In particular, the estimated effect of upweighting training point $x$ on the loss at a test point $x_{\text{test}}$ is:

$$u = -\nabla_\theta \ell(x_{\text{test}},\theta^*)^\top HS_{\theta^*}^{-1}\nabla_\theta \ell(x,\theta^*) \tag{2.46}$$

which can be interpreted as an alignment between the test gradient and the inverse-curvature-preconditioned training gradient. In deep networks, the practical appeal is that one can compute the required inverse-Hessian–vector products approximately using only gradients and Hessian–vector products, avoiding explicit Hessian construction.

Because exact Hessians are ill-conditioned, indefinite, and too large to invert in neural networks, influence estimation typically replaces $HS_{\theta*}$ with a structured curvature approximation. A common choice is the Gauss–Newton [70] or the generalised Gauss–Newton matrix [84], which is positive semidefinite and often more stable numerically.

#### 2.2.1.1.2 Model-Based Methods

Another way to estimate epistemic uncertainty is through ensemble-based methods, where multiple models are trained and aggregated. Utilising the differences across models gives ensembles the ability to provide a distribution over predictions, and, therefore, uncertainty estimates.

Deep ensembles are widely utilised, where multiple neural networks are trained independently on the same dataset with different random initialisations of their weights [71]. At inference time, predictions are aggregated, typically by computing the mean across the models to obtain the final prediction and using the variance to quantify uncertainty, similar to the MC Dropout process. This aggregation can be formalised as follows: given a set of $N_{\text{models}}$ trained models $\{\mathcal{F}_n\}_{n=1}^{N_{\text{models}}}$, the predictive mean $\hat{\mu}$ and variance $\hat{\sigma}^2$ for an input $x$ are estimated as:

$$\hat{\mu} = \frac{1}{N_{\text{models}}}\sum_{n=1}^{N_{\text{models}}}\mathcal{F}_n(x), \quad \hat{\sigma}^2 = \frac{1}{N_{\text{models}}}\sum_{n=1}^{N_{\text{models}}}\big(\mathcal{F}_n(x)-\hat{\mu}\big)^2 \tag{2.47}$$

Hyperensembles extend Deep Ensembles by introducing hyperparameter diversity in addition to weight diversity, such as varying learning rates, dropout rates, and other training configurations [72]. Bootstrap aggregation, or bagging, trains multiple models on different bootstrapped subsets of the training data [73], which introduces further variability and reduces overfitting. Architectural ensembles train models with different architectures, varying the number of layers, hidden neurons, and activation functions [74].

While ensemble-based methods can produce well-calibrated uncertainty estimates and often improve predictive performance, they require training and maintaining multiple models, making them computationally expensive in terms of memory and runtime. Moreover, the quality of the uncertainty estimates strongly depends on the diversity among the ensemble members; if models are too similar, the variance can underestimate the true epistemic uncertainty.

#### 2.2.1.1.3 Distribution-Based Methods

Another research strand to estimate epistemic uncertainty within deep learning models is to consider a distribution over feature representations or function outputs rather than modelling uncertainty solely through parameter distributions.

Gaussian Processes (GPs) are non-parametric models offering a probabilistic learning approach [75]. GPs consider an entire distribution over possible functions, weighting them based on how likely they are given the observed data. This allows GPs to naturally quantify uncertainty by not committing to a single hypothesis but instead averaging over all plausible explanations. A GP is fully specified by a mean function $\mu(x)$ and a covariance function $\text{cov}(x, x')$, which together define a prior over functions before any data is observed. When conditioned on a finite set of input points, a GP models the function values as a multivariate normal distribution $\mathcal{F}(x_1), \mathcal{F}(x_2), \ldots, \mathcal{F}(x_n) \sim \mathcal{N}(\mu, \text{CovMatrix})$ where $\mu = [\mu(x_1), \mu(x_2), \ldots, \mu(x_n)]^\top$ is the vector of prior mean values, and CovMatrix is the covariance matrix with entries $[\text{CovMatrix}]_{ij} = \text{cov}(x_i, x_j)$ [85]. This matrix encodes how similar function values should be at different input locations; if two inputs $x_i$ and $x_j$ are close, the corresponding covariance $\text{cov}(x_i, x_j)$ will be high, meaning their function values are expected to be similar. The covariance function, or kernel, plays a crucial role in defining the smoothness and generalisation properties of the model. By conditioning on observed data, GPs compute a posterior distribution over functions, making predictions by averaging over all plausible function values weighted by their probability given the data. This leads to both predictive mean and predictive variance estimates. GPs inference gives lower uncertainty if the test samples are around the region where training samples are abundant, otherwise resulting in higher uncertainty. GPs can be computationally complex for larger datasets, therefore, sparse GPs [76] were developed which approximate the full Gaussian Process using a smaller set of inducing points to reduce the computational cost. This allows for efficient inference while preserving the key properties of GPs, enabling their application to large-scale problems.

Standard GPs rely on a single-layer structure, meaning their ability to represent complex hierarchical patterns is limited. Deep Gaussian Processes extend GPs by composing multiple layers of GPs, where the output of one GP serves as the input to the next [77], similar to a standard feed-forward neural network. This hierarchical structure allows Deep GPs to learn more expressive function spaces, capturing intricate dependencies and non-stationary variations that standard GPs struggle with. However, the added flexibility of Deep GPs comes at a cost: the output of a multi-layer GP is no longer Gaussian because the composition of GPs results in a highly non-Gaussian process. This makes inference in Deep GPs computationally challenging, requiring to resort to numerical approximations.

Distance-aware methods aim to produce encoded embeddings within the feature space of the neural network that reflect how far an input is from the training data. Liu et al. [78] introduce Spectral-normalised Neural Gaussian Processes (SNGPs). SNGP modifies a standard deterministic neural network by including spectral normalisation and a GP output layer. Spectral normalisation constrains the spectral norm (largest singular value) of weight matrices in each layer, effectively enforcing a bi-Lipschitz constraint. This ensures that small changes in the input space lead to proportionally small changes in the feature space, preserving the relative distances between data points and maintaining the geometry of the input space within the feature representations. The second component of SNGP replaces the traditional dense output layer with a Gaussian Process layer equipped with a Radial Basis Function (RBF) kernel. The RBF kernel computes similarity based on the Euclidean distance between feature representations, allowing the model to produce uncertainty estimates that increase for inputs further from the training data. This combination enables SNGP to provide more reliable uncertainty estimates, particularly for OOD inputs.

Building upon the concept of distance-aware uncertainty estimation, Van Amersfoort et al. [79] introduced Deterministic Uncertainty Quantification (DUQ). DUQ employs a deterministic neural network architecture that integrates elements of Radial Basis Function (RBF) networks to achieve distance awareness. In DUQ, each class is associated with a learned centroid in the feature space. The model computes the distance between an input's feature representation and these centroids, producing a prediction based on the nearest centroid and quantifying uncertainty based on the distance to this centroid.

Mukhoti et al. [80] further refined this approach with Deep Deterministic Uncertainty (DDU). This approach leverages the feature space of a deterministic neural network trained with spectral normalisation, similar to SNGP, and models the distribution of these features using a Gaussian Mixture Model (GMM).

#### 2.2.1.2 Aleatoric Uncertainty-Based Methods

Aleatoric uncertainty-based methods refer to techniques that explicitly model the uncertainty arising from inherent noise in the data-generating process, which cannot be reduced even with additional observations. Methods include: discriminative models, which estimate uncertainty by learning predictive distributions over outputs directly [86–91], deep generative models, which leverage latent variable frameworks to capture input-dependent noise [92–101], and test-time augmentation techniques, which assess uncertainty by observing variability in predictions under input perturbations [102].

##### 2.2.1.2.1 Discriminative Models

Deep discriminative models quantify uncertainty by learning a predictive distribution through a neural network. These models achieve this in one of two ways: parametric approaches impose an explicit probability distribution over the outputs, while non-parametric approaches infer uncertainty empirically without relying on predefined distributional assumptions.

These models estimate uncertainty by assuming a predefined probability distribution over the model's outputs. These models explicitly learn the parameters of a probability distribution, rather than just predicting a single point estimate, allowing them to quantify aleatoric uncertainty by modelling the inherent variability in the data through distributional assumptions over the outputs.

In classification tasks, uncertainty can be modelled through a parameterised categorical distribution over the softmax probabilities. The model outputs a probability distribution over classes, $p(y|x) = \text{Softmax}(\mathcal{F}(x))$. However, softmax probabilities do not reflect true uncertainty and can assign high confidence even when the model is wrong. Guo et al. [86] propose temperature scaling, a simple post-hoc method that smooths the output distribution by adjusting the logits, improving calibration without changing the model's predictions.

In regression tasks, uncertainty is commonly modelled using Gaussian distribution $p(y|x) = \mathcal{N}(\mu(x), \sigma^2(x))$, where the model predicts both the mean, $\mu(x)$, and the variance, $\sigma^2(x)$. This enables the model to associate each input with its own predicted variance, thereby capturing heteroscedastic uncertainty [87]. Therefore, this approach allows the model to learn input-dependent uncertainty.

For more complex or multi-modal target distributions, mixture density networks (MDNs) provide a more expressive alternative [88]. MDNs model the output as a mixture of Gaussians, allowing the prediction of multiple plausible outcomes per input, which better reflects aleatoric uncertainty in ambiguous or noisy settings.

Non-parametric approaches to aleatoric uncertainty estimation in regression tasks avoid making strong assumptions about the underlying distribution of the data. Instead of learning a full predictive distribution, these methods directly estimate prediction intervals, which aim to contain the true output value with a specified probability.

A prominent example is through Lower Upper Bound Estimation (LUBE) [89], which trains a neural network to output lower and upper bounds $[\mathcal{F}(x)_l, \mathcal{F}(x)_u]$ for each input, optimising a loss function that balances narrow and sharp intervals and coverage (the proportion of true values that fall within the predicted interval). This method bypasses the need for explicit likelihood functions and is therefore particularly suited for modelling aleatoric uncertainty in settings where the noise characteristics are unknown or difficult to parametrise.

More recently, Chen et al. [90] formalise prediction interval learning as a constrained optimisation problem, and provide theoretical guarantees for how well it generalises. They also introduce a calibration method that automatically adjusts regularisation to balance coverage and interval width, improving performance without overfitting. Tagasovska et al. [91] propose Simultaneous Quantile Regression (SQR), a method for estimating multiple conditional quantiles of the target variable. This allows the construction of calibrated prediction intervals without assuming a specific output distribution, making it a scalable and distribution-free approach to modelling aleatoric uncertainty.

#### 2.2.1.2.2 Deep Generative Models

Deep Generative Models (DGMs) estimate uncertainty by learning a data distribution and generating samples from it. These models achieve this through one of two approaches: Variational Autoencoders (VAEs) explicitly define a probabilistic latent space, while Generative Adversarial Networks (GANs) implicitly learn to generate realistic data samples through adversarial training.

Variational Autoencoders define a probabilistic latent space, enabling the modelling of variability and uncertainty within the data. They consist of two parts: an encoder, denoted $\mathcal{F}_{\text{encoder}}$, which maps input data $x$ to a latent distribution, typically Gaussian, and a decoder, which reconstructs the input from a sample of this distribution [92]. Formally, the encoder defines a variational posterior $q_{\theta_{\text{encoder}}}(\mathcal{F}_{\text{encoder}}|x)$, and the decoder defines a likelihood $p_{\theta_{\text{decoder}}}(x|\mathcal{F}_{\text{encoder}})$, where $\theta_{\text{encoder}}$ and $\theta_{\text{decoder}}$ are the learnable parameters of the encoder and decoder networks, respectively. A standard Gaussian prior $p(\mathcal{F}_{\text{encoder}}) = \mathcal{N}(0, \text{CovMatrix})$ is typically imposed over the latent space. The model is trained by maximising the evidence lower bound (ELBO):

$$\begin{aligned}\mathcal{L}_{\text{VAE}}(\theta_{\text{encoder}}, \theta_{\text{decoder}}; x) =& \mathbb{E}_{\mathcal{F}_{\text{encoder}} \sim q_{\theta_{\text{encoder}}}(\mathcal{F}_{\text{encoder}}|x)} \left[\log p_{\theta_{\text{decoder}}}(x|\mathcal{F}_{\text{encoder}})\right] \\ &- \text{KL}\left(q_{\theta_{\text{encoder}}}(\mathcal{F}_{\text{encoder}}|x) \,\|\, p(\mathcal{F}_{\text{encoder}})\right)\end{aligned} \tag{2.48}$$

The first term encourages the decoder to reconstruct the input data accurately, while the second term regularises the encoder's output distribution to remain close to the prior. This probabilistic formulation allows VAEs to capture aleatoric uncertainty by modelling data as a distribution rather than a single point estimate.

VAEs were originally popular for use in tasks related to image data, such as denoising and missing data imputation, due to their ability to generate smooth latent representations that capture underlying structures in the data. Despite their strengths, VAEs can suffer from posterior collapse, where the latent space fails to encode meaningful variability, limiting their ability to fully represent uncertainty. Recent improvements, such as hierarchical VAEs [93] and normalising flow-based VAEs [94, 95], address this issue by allowing more flexible and expressive latent distributions. However, VAEs rely on assumptions about the prior distribution, which may not always align with real-world data, affecting their ability to fully capture complex uncertainties.

Generative Adversarial Networks implicitly learn to generate realistic data samples through adversarial training [96]. They consist of two competing networks: a generator, that learns to generic realistic data samples, and a discriminator, which tries to distinguish between the real and generated samples. This adversarial setup allows GANs to generate high-quality reconstructions of missing data.

While originally designed for data generation, GANs have been adapted for uncertainty estimation. In this context, the generator maps an input to a distribution over outputs, predicting associated uncertainty using Conditional GANs (cGANs) [97]. GANs have also seen usage in OOD detection by leveraging the discriminator's ability to assess whether an input belongs to

the training distribution. An input which sits outside the learnt distribution is assigned a high uncertainty score, making GANs effective at anomaly detection [98]. Similarly, GANs can also be explored in terms of adversarial robustness research, where they learn the distribution of adversarial examples and generate new ones [99]. This allows for testing a model's vulnerability to perturbations.

The main drawbacks to this method are as follows. First, GANs are susceptible to mode collapse during training, where the generator produces limited diversity in outputs. GANs are also less interpretable than Bayesian approaches. Research into Bayesian GANs [100] and energy-based GANs [101] seeks to address these limitations. Finally, GANs are computationally heavy during training.

#### 2.2.1.2.3 Test-Time Augmentation

Another type of aleatoric uncertainty estimation approach is Test-Time Augmentation (TTA). TTA is model-agnostic and quantifies uncertainty by applying transformations to input data at inference time to assess the variability in model predictions. Instead of making a single prediction for a given input, TTA applies a set of augmentations, such as image rotations and noise perturbations, and evaluates the model's variability in predictions. Greater variability leads to a greater aleatoric uncertainty, as it captures how sensitive the model is to slight variations in the data. TTA is particularly effective in image classification and segmentation tasks. Here, image augmentations such as cropping, colour transformations and scaling can simulate real-world data variations [102].

### 2.2.1.3 Combined Uncertainty-Based Methods

Instead of focusing on either model uncertainty or inherent uncertainty within the data itself, recent research attempts to estimate both uncertainty types simultaneously. Methods include: hybrid methods, which combine an epistemic and aleatoric method into one framework [71, 87, 103–105], conformal prediction [106–116], and evidential deep learning [117–124].

#### 2.2.1.3.1 Hybrid Methods

The most straightforward approach to modelling both types of uncertainties is to select two methods that can estimate model and data uncertainty and combine them into a powerful framework.

Kendall et al. [87] developed a novel Bayesian deep learning framework that learns to map aleatoric uncertainty from the data, sitting on top of epistemic uncertainty models by combining parametric predictive distributions, and Monte Carlo Dropout. The Monte Carlo Dropout method approximates a full BNN by inducing an approximate variational distribution over the network's weights, where a stochastic prediction $y_n = \mathcal{F}_{\theta_n}(x)$ can be made with dropout-enabled weights $\theta_n$ during the $n$-th forward pass $\in N_{\text{mc}}$. Thus, the epistemic uncertainty can be estimated as the variance across these samples. To measure the aleatoric uncertainty within the proposed framework, the negative log-likelihood of a Gaussian likelihood is utilised within

the loss function of the neural network $p(y_i|x_i) = \mathcal{N}(\hat{\mu}_i, \hat{\sigma}_i^2)$ to learn to predict the aleatoric uncertainty as a function of the input $x$. This framework, however, has a computational cost and little to no sparsity control meaning it lacks the ability to ignore irrelevant features, which can reduce efficiency and increase overfitting. These limitations were overcome when MC Dropout was replaced with Bayesian Model Averaging and learning model structure uncertainty rather than just weight uncertainties [103].

Alternatively, instead of relying on Bayesian methods, ensembles of independently trained neural networks can be utilised in combination with parametric predictive distributions [71].

The issue with utilising parametric predictive distributions is that they assume the data distribution form. Instead, non-parametric prediction intervals can be combined with ensembles to provide a more flexible combined uncertainty estimation [104]. Unlike Gaussian-based parametric distributions, as seen in the previous methods in this section, prediction intervals work when the data distribution is unknown. To this end, they directly learn prediction intervals, which means there is no need for extra variance modelling, and they are more robust to outliers and heavy-tailed distributions. The ensemble is then trained to predict a prediction interval that is narrow while ensuring coverage with probability $1- \alpha$, where $\alpha$ is the chosen misscoverage rate. Using ensembles in this way assumes the ensemble mean and variance are symmetric, which does not always hold. Instead, a split normal probability density function can be fit [105] to provide separate left and right side asymmetrical variances.

#### 2.2.1.3.2 Conformal Prediction

Conformal Prediction (CP) is a powerful framework that can provide valid, distribution-free prediction sets/intervals $\Gamma(X)$ for any machine learning model [106], ensuring that the true answer $Y$ is included with a specified probability $\mathbb{P}(Y \in \Gamma(X)) \geqslant 1-\alpha$. This guarantee holds marginally over multiple predictions, making CP a robust uncertainty quantification method that adapts to the model's error distribution without assuming a specific form for the data distribution. To construct prediction sets, CP relies on a nonconformity (or conformity) score function $\kappa(X, Y)$, which quantifies how unusual a given observation $(X, Y)$ is relative to past training data. Typically, CP methods can be grouped into two categories: Full and Split CP [107].

Full CP methods construct prediction sets by evaluating a nonconformity score for each test data point using a leave-one-out approach [108]. This ensures exact finite-sample validity but is computationally expensive, as it requires model retraining or recalibration for each new test instance. In this way, full CP methods can provide coverage guarantees, but can be impractical for large-scale applications. To alleviate computational load, $k$-fold cross-validation can be utilised to reduce the number of times a new model needs to be trained [109], at the risk of reduced coverage guarantees. Influence functions can also be utilised to approximate the model training process for a new test point [125]. However, this aspect only begins to save computation at larger scales, since at small-scale datasets and model architectures, the savings are very small.

Split CP methods are a computationally efficient variant of Conformal Prediction that uses a separate calibration dataset to compute conformity scores. Unlike Full CP, they do not require

retraining for each test point, making them scalable for large models. However, their coverage guarantees depend on the size and representativeness of the calibration dataset, meaning that poorly chosen calibration sets can lead to undercoverage or overly conservative intervals [108]. Adaptive Prediction Sets (APS) extend Split CP by dynamically adjusting the size of prediction sets based on the difficulty of individual instances [110]. However, Angelopoulos et al. [111] note that APS can sometimes produce unnecessarily large prediction sets and introduce Regularized Adaptive Prediction Sets (RAPS), which regularise the tail of the prediction set to reduce its size on average.

Conformal Prediction can be extended to regression to provide valid, distribution-free prediction intervals for continuous targets [107]. Similar to classification, both Full CP and Split CP methods exist, where Full CP offers exact finite-sample coverage at the cost of high computational complexity, while Split CP provides a more scalable alternative using a separate calibration set. Notable methods include Conformalized Quantile Regression (CQR), a Split CP method that utilises a deep quantile network to provide well-calibrated prediction intervals [112]. Han et al. [113] build upon CQR and introduce Split localised conformal prediction (SLCR), a method that introduces adaptive interval sizes (seen in APS) to CQR.

Finally, beyond standard classification and regression, Conformal Prediction has been extended to a variety of specialised settings. In time-series forecasting, CP must account for temporal dependencies [114, 115]. CP has even been extended for risk-aware decision-making [116], by allowing different weights to be assigned to errors associated with particular classes, facilitating risk-aware prediction sets.

#### 2.2.1.3.3 Evidential Deep Learning

Apart from Split Conformal Prediction, the previously discussed methods that estimate both model and data uncertainty are all computationally intensive, making them unsuitable for deployment in edge devices or large-scale applications. Sensoy et al. [117] propose Evidential Deep Learning (EDL), a technique that captures both types of uncertainty using only a single forward pass of a deterministic neural network. Instead of directly producing class probabilities, EDL predicts parameters that define a probability distribution over classes, allowing it to represent uncertainty more effectively. The model gathers evidence for each class using a non-negative transformation of the network's outputs, ensuring that the evidence values are interpretable and consistent. To accurately represent uncertainty, EDL uses a specially designed loss function based on likelihood principles, often with an added regularisation term that encourages the model to express more uncertainty in the face of ambiguous or incorrect inputs. Final class probabilities are derived from the predicted distribution, and uncertainty is naturally inferred from the total amount of evidence collected. Because it requires only a single forward pass, EDL is computationally efficient. However, it does rely on a custom network architecture and a specialised loss function to ensure that its predictions and uncertainty estimates are properly calibrated.

Subsequent work has focused on reformulating the evidence-collection process to improve the

robustness and efficacy of EDL. Pandey et al. [118] identify that EDL has regions in the learning space where evidence cannot be learnt, and propose a novel regulariser to alleviate this. Information-based EDL introduces a Fisher Information Matrix to measure the informativeness of evidence into the loss function of the network [119]. This allows the model to effectively learn evidence from high uncertain examples.

Beyond architectural and uncertainty modelling refinements, recent research has explored allowing EDL to be used with pre-trained standard models, alleviating the disadvantage of requiring a custom architecture and loss function, by utilising an EDL meta-model that extracts features from the pretrained model to train itself to estimate uncertainty as the standard EDL process would [120]. EDL has also been extended to regression [121] by placing evidential priors over a Gaussian likelihood function to force the network to infer the parameters of the evidential distribution. However, this comes at a large risk of evidence contraction, where the model becomes overconfident in its predictions and collapses the uncertainty estimates, leading to underestimated predictive variance and reduced robustness to out-of-distribution samples [122–124]. This issue can be partially alleviated by adding specialised regularisation terms to the loss function that promote penalisation for overconfident incorrect predictions and encourage balanced evidence accumulation.

### 2.2.2 Safe Reinforcement Learning

Safety, as a concept, refers to eliminating risk or reducing it to a tolerable level [126]. In some contexts, safety is often defined as the antonym of risk: the lower the risk, the higher the safety, where risk can be understood as the combination of the probability and the severity of an undesirable outcome occurring; however, ISO/IEC Guide 51 emphasises that safety is *more* than the antonym of risk [127]. [127]. Safety can differ across different fields: in robotics, safety could involve harm to the hardware and surrounding humans in the environment [128], or in psychology, it may involve a human becoming distressed emotionally [129]. Beyond physical and psychological harm, safety also can encompass broader social, legal, ethical, empathetic, and cultural (SLEEC) concerns [8]. Despite these differences, the common concept that remains throughout each definition is harm and risk [126]. If there exists a risk that harm could come to an entity (e.g. human, robot, ecosystem, etc.), then mitigation techniques should be applied to reduce such risk.

AI models are no different; those deployed in physical environments, such as robots, should prioritise preventing physical harm, while AI systems operating in digital domains, such as large language models, must account for SLEEC concerns, including emotional and psychological safety.

In an effort to define safety for all AI models, research has been conducted to taxonomise issues that such models need to consider [130–132]. Seven core concerns have been identified and are illustrated in Figure 2.5: cybersecurity, negative side effects, robustness, safe exploration, scalable oversight, reward hacking, and performance.

The first concern is that an AI model must be secure from cyber threats. As AI models become

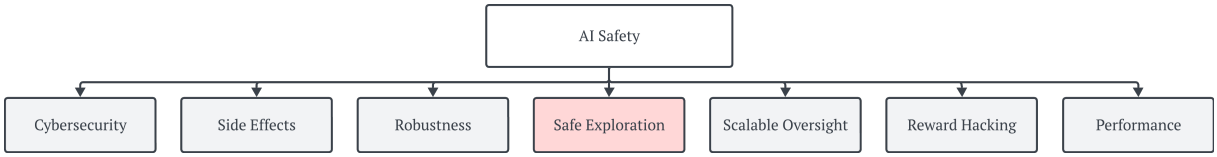



Figure 2.5: A classification of safety concerns categorising key risks that AI models should address [130–132]. Safe Exploration is highlighted in red as it is the central focus of this thesis.

increasingly integrated into critical infrastructure and sensitive applications, the risk of adversarial attacks, data breaches, and model manipulation grows [133]. A designer must ensure that the AI model is resilient to such threats, both during development and after deployment. This involves anticipating potential attacks and building safeguards into the system. Failing to do so could allow malicious actors to compromise the integrity of the model's outputs or gain access to sensitive data, leading to serious downstream consequences.

An agent operating in a multifaceted environment may have an objective function that only focuses on one aspect of the environment and therefore may engage in behaviour that can be harmful or damaging to other aspects of the environment, thus causing negative side effects [134]. For example, consider an autonomous vehicle navigating through a complex urban environment with an objective function to solely navigate to the intended destination within the shortest time possible. The designer might have intended this objective function to create a model that is efficient, but the model could ignore traffic laws and drive dangerously. According to the specified objective function, this behaviour is as intended, but as a result, it has caused unintended side effects that has caused dangerous behaviour as the objective function was only designed with a single aspect of the complex environment in mind.

The ability for an AI model to maintain robust performance when exposed to inputs or conditions that differ from those seen during training is critical. In real-world settings, models often face distributional shifts, where the data encountered at deployment diverges subtly or significantly from the training distribution. Without appropriate mechanisms in place, such shifts can lead to brittle or erratic behaviour. Robustness also encompasses resilience to adversarial perturbations, carefully crafted changes to inputs that can cause the model to produce incorrect or dangerous outputs, as well as physical-world attacks such as patches placed on objects to fool vision systems [135]. Furthermore, models must be designed to handle noisy, corrupted, or intentionally poisoned data, and to avoid reliance on confounding variables that may be spuriously correlated with target labels. A robust AI model is one that can recognise when it is uncertain, generalise beyond narrow training conditions, and resist manipulation designed to exploit its weaknesses.

Safe exploration, which is mainly relevant to reinforcement learning agents, is the ability to explore the environment or task without causing irreversible or dangerous harm to the agent or any actor within the environment [14]. This is particularly challenging when agents operate in physical domains, such as robotics or autonomous vehicles, where exploratory actions taken during training/refinement stages may lead to physical damage, safety violations, or harm to humans. While simulations can help mitigate these risks, it is often an imperfect proxy for the real

world, and discrepancies between simulated and real environments can lead to unsafe behaviours when the agent is deployed. Even in digital domains, exploration without safeguards can result in problematic outcomes. For example, consider a conversational AI model fine-tuned via reinforcement learning from human feedback (RLHF) [136] that explores new dialogue strategies during deployment. Without proper constraints, it might unintentionally generate emotionally distressing or culturally insensitive responses [137]. Designers must therefore incorporate safety mechanisms, such as constrained exploration, conservative objectives, or human oversight, to ensure that the agent can learn effectively without exposing the environment or its users to unnecessary risk.

Scalable oversight refers to the difficulty of supervising AI models when high-quality feedback is expensive, time-consuming, or otherwise limited. As models are deployed in increasingly complex tasks, it becomes infeasible for humans to monitor every decision or outcome to check for correctness and efficiency [138]. This is particularly relevant in settings such as RLHF, where annotators must evaluate the quality of model outputs or rank behaviours. Similarly, in high-stakes domains like healthcare, expert oversight may be necessary but difficult to provide continuously. Without scalable oversight mechanisms, models may drift from desired behaviours or optimise for proxies that do not reflect true human intent. Addressing this issue involves developing models that can generalise from limited supervision, defer to human judgement when uncertain, or learn from weak, indirect, or aggregated signals while still preserving alignment with human values.

Reward hacking involves the model *gaming/hacking* the objective function by maximizing it, but not in the way the designer originally intended, to a degenerate solution [139, 140]. In the autonomous vehicle example, the objective function may encourage the model to maintain speed within the legal limits. However, the model might learn to hack this reward and maintain the speed limit strictly, even in hazardous situations (e.g., icy roads, congestion), ignoring contextual safety.

The final concern is performance, the basic requirement that an AI model must reliably achieve its intended objectives. While performance may seem orthogonal to safety, a model that consistently fails to perform well can pose significant risks, particularly in safety-critical settings. For example, a medical diagnostic model that lacks accuracy may miss early signs of illness, while an autonomous drone that cannot stably navigate may endanger those around it. In such cases, poor performance is not merely a matter of inefficiency, but a direct threat to human well-being. Moreover, performance issues may compound other safety concerns: an underperforming model may be more susceptible to adversarial attacks, more reliant on confounding variables, or more prone to unsafe exploration. Thus, ensuring high performance—through rigorous evaluation, stress testing, and continual monitoring—is not just a matter of usability, but a prerequisite for safe and trustworthy AI deployment.

The prior research and proposed methods presented in this thesis from this section onward will focus on the safe exploration of reinforcement learning agents specifically, following the hypotheses presented in Section 1.2.

In the context of RL specifically, safe exploration during training is the process by which an agent learns as close to an optimal policy as possible without violating safety constraints; this may also include adhering to safety constraints during deployment. Unlike standard exploration that prioritises reward maximisation, safe exploration explicitly accounts for the risk of entering known or unknown hazardous or unrecoverable states.

The most common way to encourage safety is through reward shaping, where the reward function $R(s, a)$ is modified to penalise unsafe actions or states within the MDP. While this method can successfully prevent the agent from harm, it requires specialist knowledge of all safety concerns and exhaustive tuning to find the correct reward/penalisation balance [10]. Therefore, reward shaping is simply not enough.

To provide a more formal notation of safety for RL agents, safe exploration can be formalised using Constrained Markov Decision Processes (CMDPs) [141, p. 21]. A CMDP extends the standard MDP by incorporating a cost function and a constraint on the cumulative cost. Formally, a CMDP is defined by the tuple $(S, A, P, R, C, \chi, \gamma)$ where $C$: $S \times A \to \mathbb{R}_{\geqslant 0}$ is the cost function, similar to the reward function, that provides feedback to the agent that represents risk or undesirable behaviour, and $\chi$ is the maximum allowable cost per episode. The objective of the RL agents shifts according to this extension, now the agent must find a policy $\pi : S \to A$ that maximises the expected reward while keeping the cumulative expected cost below the threshold $\chi$. Thus:

$$\max_{\pi} \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)] \quad \text{subject to} \quad \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t C(s_t, a_t)] \leqslant \chi \tag{2.49}$$

This ensures that the agent explores the environment while adhering to constraints on unsafe behaviour. These are just two examples of how safety in reinforcement learning has been formally defined; states-of-the-art methods discussed later explore unique definitions to tailor to different settings and assumptions. Nonetheless, they share the common goal of ensuring that an agent explores its environment while adhering to constraints on unsafe behaviour.

A classification of state-of-the-art methods for safe exploration of RL agents is presented in Figure 2.6 and can be decomposed into three subcategories: knowledge-based methods, black-box methods, and methods that can be either. The methods that comprise these categories will be discussed in detail in Sections sections 2.2.2.1 to 2.2.2.3.

#### 2.2.2.1 Knowledge-Based Methods

Knowledge-based methods refer to approaches that leverage prior knowledge about the environment, task structure, or known safety constraints to guide the agent away from harmful behaviours during exploration. Rather than relying solely on trial and error, these methods incorporate human expertise, domain models, or predefined rules to restrict the agent's actions or help shape its policy. Methods include formal verification [142–148], automata [149–152], human oversight [153–158], and Bayesian techniques [159–162].

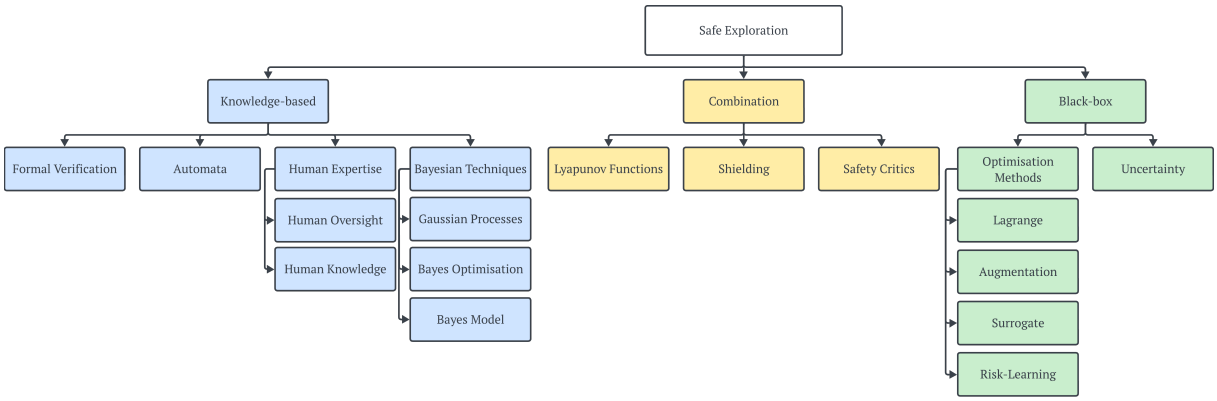



Figure 2.6: A classification of techniques for the safe exploration of RL agents during training decomposed into three categories: knowledge-based methods, black-box methods, and methods that can be either.

#### 2.2.2.1.1 Formal Verification

Formal verification methods for safe RL treat safety as a logically specified property of the agent–environment closed loop, and use verification (e.g., model checking, theorem proving, runtime monitoring) to establish, prior to or during learning, that the agent's behaviour cannot violate that property except with a bounded probability (or within a bounded quantitative budget). In contrast to penalty-style objectives that absorb safety into a scalarised loss, formal verification aims to make the safety requirement explicit and machine-checkable, typically in a temporal logic that can express ordering, inevitability, and avoidance over trajectories.

Model checking [163, pp. 7–10] is the most widely used verification primitive in safe RL, as it supports stochastic dynamics and yields quantitative guarantees. The core idea is to model the environment as a proposition-labelled MDP, express safety constraints in logic, and algorithmically verify whether the closed-loop system induced by a policy satisfies these constraints. The resulting verified artefacts, safe policies, action sets, or behavioural restrictions, are then used to constrain exploration during learning [143].

Safety specifications are commonly expressed in temporal logics. Given an MDP and a labelling function $L : S \to 2^{AP}$ assigning atomic propositions to states, a safety requirement can then be defined as a constraint over trajectories:

$$P_\pi(\text{Trace} \models \varphi) \geqslant 1 - \delta \tag{2.50}$$

where $\varphi$ encodes a temporal safety property, such as region avoidance, and the full definition above states that the policy $\pi$ satisfies the safety specification $\varphi$ with a probability of at least $1 - \delta$ over trajectories [144].

Because direct model checking is intractable in large or continuous state spaces, safe RL methods typically rely on abstraction. An abstract MDP is constructed by aggregating ground states into a smaller abstract state space, yielding a conservative approximation suitable for verification.

Model checking is then performed on this abstract model to compute satisfaction probabilities or to synthesise abstract policies that provably satisfy the safety specification [145].

While model checking provides strong, exhaustive safety guarantees, its applicability in modern RL is fundamentally limited by representational complexity. Classical verification algorithms assume access to a finite, explicitly represented transition model and a controller whose structure admits symbolic reasoning. In contrast, safe RL policies are typically parameterised by deep neural networks, inducing highly non-linear and high-dimensional closed-loop dynamics. Directly model-checking such policies is, in general, intractable, and even approximate verification scales poorly.

One mitigation is via policy extraction, which shifts verification from the learned neural policy to a structured surrogate that is amenable to formal analysis. Rather than verifying a deep policy $\pi_\theta$ directly, Bastani et al [142] train a neural oracle policy $\pi^{'}$, and then distil it into a non-parametric decision tree to support efficient downstream verification. The resulting policy representation enables explicit verification that would be infeasible for the original neural controller because decision trees partition the state space into axis-aligned regions with constant actions. Policy extraction, thus, provides a bridge between classical model checking and deep RL, retaining formal safety analysis while acknowledging the practical limits of verifying neural policies.

Formal verification methods provide the strongest form of safety assurance in RL by encoding safety requirements as explicit, machine-checkable specifications and enforcing them through exhaustive analysis or synthesis. Model checking and related techniques can guarantee constraint satisfaction by construction, offering clear interpretability and principled failure modes. However, their applicability is fundamentally constrained by the need for tractable models and verifiable policy representations, making direct integration with deep, high-dimensional RL systems challenging. As a result, formal verification is most effective when combined with abstraction, policy restriction, or representation simplification, trading expressiveness and scalability for strong, pre-runtime safety guarantees.

#### 2.2.2.1.2 Automata

Automata methods replace offline verification (e.g., model checking a fixed policy against a specification) with an automata-theoretic integration of the specification into the learning problem itself. The motivation is closely aligned with the limitations discussed above: although model checking can yield strong guarantees, modern RL controllers (in particular, deep policies) are often difficult to verify directly. Automata-based approaches compile a temporal-logic specification into a finite-state monitor and couple this monitor to the environment, guiding learning by the specification's progress. In doing so, they provide a structured mechanism for expressing long-horizon safety and task requirements in a form that can be optimised by standard RL algorithms.

Given an LTL specification $\varphi$ over $AP$ (as discussed in Section 2.2.2.1.1), automata-based methods construct an equivalent automaton $\mathcal{A}_\varphi$ whose accepting condition characterises satisfaction of $\varphi$. The standard construction then forms a product between the environment and the au-

tomaton so that each augmented state tracks both the physical state and the specification state. Learning in this product space allows the agent to optimise for the satisfaction of $\varphi$, typically by maximising the satisfaction probability $P_\pi(\tau \models \varphi)$.

A foundational example by Hasanbeig et al. [149] combines automata-guided learning with an explicit mechanism to keep exploration safe. An LTL task specification $\varphi$ is converted into a limit-deterministic Büchi automaton (LDBA) [164] $\mathcal{A} = (\mathcal{Q}, q_0, \Delta, \Sigma, \mathcal{F})$ with an alphabet $\Sigma = 2^{AP}$. A product MDP is then learnt whose states are pairs $s_\otimes = (s, q) \in S \times \mathcal{Q}$. This product construction is central because it turns satisfaction of a temporal property into a reachability/recurrence structure over an augmented Markovian state.

The reward function becomes a function of whether the next automaton state makes progress toward satisfying the Büchi condition. Formally, after observing a transition $s_\otimes = (s, q) \to^a (s', q')$, the immediate reward is assigned as:

$$R(s_\otimes, a) = \begin{cases} r_p, & \text{if } q' \in \mathcal{Q}_{\text{acc}}, \\ r_n, & \text{otherwise.} \end{cases} \tag{2.51}$$

where $\mathcal{Q}_{\text{acc}} \subseteq \mathcal{Q}$ is an accepting set. This construction is designed so that maximising the expected discounted return in the product MDP aligns with maximising the probability of satisfying the original LTL formula [150].

Even with accepting set rewards, a practical issue is sparsity: under Hasanbeig et al.'s scheme, the positive signal $r_p$ is only obtained when the automaton moves into an accepting component, so in long-horizon tasks, the agent may see no task feedback early in exploration. DRL$^2$ [151] extends this work by deriving a dense reward function from the structure of the automaton, rather than just relying on accepting states.

Formally, DRL$^2$ constructs an estimated transition kernel over automaton states $\text{TK}(q'|q)$ and constructs a high-level value function $\hat{V}(q)$ on the LBDA states that reflect *how promising* each state is to reach eventual acceptance, and then uses $\hat{V}$ for reward shaping. Given a transition $s_\otimes = (s, q) \to^a (s', q')$, the dense reward is defined as:

$$R(q, q') = \gamma(q')\hat{V}(q') - \hat{V}(q) \tag{2.52}$$

which provides guidance at every automaton transition (not only when $q' \in \mathcal{Q}_{\text{acc}}$). Intuitively, this makes progress in the specification learnable much earlier, and empirically improves sample efficiency while preserving the automata-based objective of maximising satisfaction probability.

Automata-based methods offer a principled middle ground between verification and unconstrained learning by embedding temporal-logic specifications directly into the learning dynamics. By compiling safety and task requirements into finite-state monitors and optimising behaviour in the resulting product MDP, these approaches make long-horizon, non-Markovian specifications amenable to standard RL algorithms. Their main strength lies in the expressiveness and interpretability of the specifications they support, enabling rich notions of progress and satisfaction

beyond scalar costs. However, this comes at the cost of increased state-space complexity and reliance on carefully designed reward structures, with practical performance often hinging on abstraction quality and reward shaping to mitigate sparsity.

#### 2.2.2.1.3 Human Oversight

Human-oversight approaches treat safety as a property that can be maintained during learning by incorporating a human (or human-derived signal) directly into the control loop, rather than relying solely on formal specifications or learned cost critics. The unifying idea is that humans can contribute safety in at least three complementary roles: first, as an intervening supervisor who prevents imminent violations by overriding unsafe actions; second, as a source of structured prior knowledge that shapes the learning process through curricula, warm-starting, or staged exposure to risk so that the agent encounters hazardous situations in a controlled manner; and third, as an on-demand advisor where the agent actively queries the human for guidance when uncertainty or risk is high, trading autonomy for safety only when needed. Across these threads, the common objective is not merely to learn a safe final policy, but to reduce the probability and severity of safety violations throughout training by using human feedback as an explicit safety mechanism.

Intervention-based human oversight is often formulated via the *big red button* problem first discussed by Orseau and Armstrong [153]: if a learning agent anticipates that a human operator will interrupt it during trajectories that are simultaneously high-reward and high-risk, then standard RL objectives can induce incentives to avoid interruption (e.g., by steering toward states where interruption is unlikely) or even to disable the interruption mechanism. They formalise interruption as an operator applied to the agent's policy rather than as an exogenous change to the environment. A key implication is that off-policy learners can remain well-behaved under interruption (since their value targets do not depend on the behaviour policy). In contrast, on-policy updates can inadvertently internalise the interrupt-driven behaviour unless modified. The large issue with this is that human oversight is costly and burdensome.

Where safe interruptibility focuses on making human takeover incentive-compatible, subsequent work treats human intervention as a direct mechanism for preventing catastrophic actions during training, and then reducing the burden of keeping a human constantly in the loop by distilling their decisions into a supervised policy [154]. In this framing, the agent proposes an action $a_t \sim \pi_\theta(\cdot|s_t)$ and a human-like policy evaluates whether $(s_t, a_t)$ is catastrophic. If so, a safe alternative action is replaced, and a penalty signal is applied to discourage that proposed action in the future. This can be extended to predicted futures using learned dynamics models, so that actions can be rejected/replaced if the catastrophe will occur within a short horizon [155]. This massively speeds up learning, as a policy can be induced faster and more frequently than a human.

Human-oversight via curriculum knowledge treats safety as something that can be enforced by structuring the training process, rather than by verifying the final controller or constantly intervening at the action level. The core idea is that a teacher (human-designed or automated)

provides prior knowledge about how to make learning safe early on, e.g., through training stages with stronger safeguards, easier initial conditions, or restricted exposure to hazards, and then progressively relaxes these supports as the agent becomes competent. Formally, the agent is trained over a sequence of auxiliary training problems $\{M^{(n)}\}_{n=1}^{N}$ that share $(S, A)$ but differ in dynamics, constraints, etc [157]. The simplest formulation stages the agent in progressively harder versions of the same task, leading up to the final target environment [156]. Task $m$ is a simplified version of the original MDP $M$ but with simplified hazard amounts, layouts, etc. After successive safe completions, the agent is allowed to be exposed to a higher-risk version of this target environment. This technique allows the agent to focus on simpler hazard avoidance, i.e., a single hazard, before being exposed to complex dynamics, i.e., multiple dense moving objects. This reduces safety violations and understanding, as the agent is not *overwhelmed* during early training.

Finally, human oversight via asking for help treats safety as an online-learning problem in which exploration is intrinsically risky because some mistakes are irreversible (i.e., there may be no reset and no recovery once a catastrophic transition occurs [165]). The central mechanism is to augment the RL interaction loop with an explicit query action: at each time step, the agent can either act autonomously or request guidance from a mentor, who provides the action they would take in the current state [158]. Formally, alongside choosing $a \in A$, the agent also chooses a query indicator $qry_t \in \{0, 1\}$; if $qry_t = 1$, then it uses the humans actions $a_t^h = \pi^h(s_t)$ to avoid being unsafe. The learning objective is twofold: (i) achieve performance approaching the mentor, typically expressed via regret relative to the mentor's return from the same start state, and (ii) become self-sufficient, meaning the total number of queries grows sub-linearly with the horizon $T$.

Human-oversight methods provide a safety mechanism by leveraging human judgment to prevent or mitigate catastrophic behaviour during learning, rather than relying exclusively on formal specifications or learned critics. Their primary strength lies in their flexibility: human intervention, guidance, or curriculum design can compensate for modelling errors, specification gaps, and unanticipated hazards, particularly in early or high-risk training phases. However, these approaches introduce new dependencies on human availability, consistency, and cost, and raise challenges around scalability, latency, and the faithful transfer of human intent into autonomous behaviour.

#### 2.2.2.1.4 Bayesian Techniques

Bayesian techniques address safety by making epistemic uncertainty over the environment and value predictions explicit, and then using this uncertainty to regulate exploration. The agent maintains a posterior belief over unknown quantities, such as dynamics parameters, reward/cost functions, or constraint boundaries, and selects actions that optimise expected performance while satisfying safety requirements either with high probability (chance constraints) or under worst-case credible models. Unlike broader uncertainty-based approaches that may use heuristic uncertainty scores from ensembles or learned critics primarily as auxiliary signals, Bayesian methods explicitly model uncertainty as a posterior distribution with principled credibility bounds that

enter the safety constraint and decision rule. A further distinguishing feature is the use of priors to encode domain knowledge or inductive biases (e.g., smoothness or safety margins), which are then updated into posteriors as data is collected, yielding a coherent uncertainty-to-decision pipeline.

At a technical level, GP-based Bayesian methods share a common certificate mechanism: an unknown safety-relevant quantity is modelled as a latent function over $(s, a)$ with a GP prior, and the posterior predictive mean and variance $(\hat{\mu}(s, a), \hat{\sigma}^2(s, a))$ are converted into conservative credibility bounds that act as a decision-time filter [159]. A typical construction defines a lower confidence bound and declares $(s, a)$ admissible/safe only if this bound satisfies the safety inequality. The resulting certified set is then expanded iteratively: the agent acts only within the currently certified region while selecting informative $(s, a)$ near its boundary to reduce posterior uncertainty and thereby enlarge the set without stepping outside it.

Where the safe-set expansion view uses GP credibility bounds primarily as certificates of admissibility, restricting behaviour to a region that can be declared safe and then cautiously enlarging that region as uncertainty contracts, a constrained-optimisation view uses the same posterior machinery to explicitly trade off improving performance against maintaining feasibility under the constraint as learning progresses [160]. Separate GPs model reward and safety-relevant quantities, yielding posteriors $(\hat{\mu}_R, \hat{\sigma}_R^2)$ and $(\hat{\mu}_C, \hat{\sigma}_C^2)$. Rather than selecting data purely to expand the safe region, the agent prioritises queries and actions that are expected to improve the best achievable return within the constraint, balancing uncertainty reduction in the safety model (to enlarge feasibility) against uncertainty reduction in the reward model (to locate high-value safe behaviour), while exploiting the current conservative model to act optimally in the presently certified set. This typically yields a less conservative agent.

Although the GP-based constrained-optimisation view is conceptually clean, applying it directly becomes difficult once the policy and value functions are represented by deep neural networks in large continuous spaces: the GP component must be used in a way that preserves calibrated uncertainty while avoiding the computational and modelling bottlenecks [166]. A common scaling strategy is therefore to keep the Bayesian safety signal modular, as an auxiliary estimator that constrains or steers policy improvement, while letting deep RL handle representation and optimisation. Fan and Li [161] couple an off-policy actor–critic (DDPG-style) with an online GP that estimates a trajectory-level safety quantity cast in Lyapunov form. Specifically, they learn a deep critic $G(s, a)$ that approximates the cumulative safety value $G^\pi(s, a)$, and define a one-step Lyapunov signal $g^\pi(s_t, a_t) = G(s_{t+1}, \pi(s_{t+1})) - G(s_t, a_t)$ along observed transitions; a GP posterior over $g^\pi$ yields confidence bounds which are then used to restrict action selection.

Beyond GP surrogates, a growing line of work replaces the explicit GP posterior with a Bayesian world model that can represent high-dimensional observations (e.g., images) and support planning under epistemic uncertainty. The central idea is to learn a probabilistic dynamics model $p_\theta(s_{t+1}|s_t, a_t))$ with an approximate posterior over parameters (e.g., via probabilistic ensembles), and then select policies that are pessimistic with respect to constraint satisfaction under the induced set of plausible models [162].

Bayesian techniques provide a principled framework by grounding safety decisions in explicit posterior uncertainty over safety-relevant quantities. By propagating epistemic uncertainty into credibility bounds or chance constraints, these methods offer adaptive safety mechanisms that naturally relax conservatism as knowledge improves. Their main strength lies in the coherence of the uncertainty-to-decision pipeline and the ability to incorporate prior knowledge, but this comes at the cost of increased modelling and computational complexity, as well as sensitivity to prior choice and posterior approximation quality.

#### 2.2.2.2 Black-box Methods

Black-box methods refer to approaches that assume no prior knowledge of the environment's task structure, environment dynamics, or safety constraints. These methods treat the environment as an unknown system and aim to learn safe behaviours purely from interaction, often under minimal assumptions. Since no structured information is available upfront, they may be more flexible and broadly applicable than knowledge-based techniques, but often require significantly more data to identify and avoid unsafe behaviours.

This assumption is also well aligned with many decentralised cyber-physical and self-adaptive systems (e.g., traffic networks, smart energy management, and mobility services) where individual agents typically only observe local signals and must adapt online through interaction rather than relying on an accurate global model a priori [167]. In these settings, the environment can evolve over time so decision-making must be learned under limited prior structure, motivating black-box formulations and, in particular, optimisation- and uncertainty-driven mechanisms for avoiding unsafe behaviours.

Methods in this category include optimisation techniques [168–176], and uncertainty-based techniques [177–179].

##### 2.2.2.2.1 Optimisation Techniques

Optimisation-based methods cast safety as an explicit trade-off between task performance and constraint satisfaction, typically formalised within the CMDP framework. Ways to optimise this trade-off can differ widely: through dual variables and penalties, constrained or surrogate objectives, or state augmentation.

The first set of techniques, Lagrangian (or primal–dual) methods, operationalise CMDP training by converting the constrained optimisation problem into a saddle-point objective that can be optimised. The agent seeks to find a policy $\pi$ that maximises the expected discount return subject to a safety budget on the cumulative cost. This forms the Lagrangian:

$$\mathcal{L}_\pi = -\mathbb{E}\Big[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)\Big] + \lambda \times \mathbb{E}\Big[\sum_{t=0}^{\infty}\gamma^t C(s_t,a_t)\Big] \tag{2.53}$$

where $\lambda \geqslant 0$. This yields the Lagrangian baseline (typically named PPO-Lag or DDPG-Lag) [168], where the policy update aims to maximise rewards and minimise costs, while $\lambda$ is

automatically increased or decreased according to the empirical cost observed over time during training:

$$\lambda \leftarrow \max\left(0, \lambda - \eta_\lambda(\chi - \hat{C})\right) \tag{2.54}$$

ensuring the cost budget $\chi$ is enforced in expectation. A limitation of the integral-style multiplier update in Equation 2.54 is that it often encourages oscillating behaviour between the reward and cost because $\lambda$ is adjusted only in response to accumulated constraint error, the penalty can react too slowly to rising costs and then over-correct once violations occur. Stooke et al. [172] formalise this observation and propose replacing the pure integral update with a PID controller [180] over the cost-budget error, augmenting $\lambda$ with proportional and derivative terms that respond immediately to current violation magnitude and anticipate increasing costs, thereby damping oscillations and reducing overshoot while retaining the simplicity of the original method. An integral state $\text{Int}_t \leftarrow \max(0, \text{Int}_{t-1} + \Delta_t)$ and a projected derivative term $\text{Der}_t \leftarrow \max(0, \Delta_t - \Delta_{t-1})$ are tracked, where $\Delta_t = \hat{C}_t - \chi$ is the current cost error. The cost weight is then updated as:

$$\lambda \leftarrow \max\left(0, \beta_P \Delta_t + \beta_I \text{Int}_t + \beta_D \text{Der}_t\right) \tag{2.55}$$

which falls back to the standard Lagrangian when $\beta_P = 0$ and $\beta_D = 0$. The proportional term increases the penalty immediately when costs exceed the budget, the integral term accumulates persistent violations to remove steady-state error, and the derivative term anticipates upward cost trends. Together, these terms stabilise the primal–dual dynamics, reducing overshoot and oscillation while reaching constraint satisfaction more reliably.

A further limitation of expectation-constrained Lagrangian training is that it cannot distinguish between cost distributions with the same expectation but different exposure to rare, high-severity failures, since the budget in Equation 2.53 is enforced only in expectation. Risk-sensitive approaches address this limitation by constraining Conditional Value-at-Risk (CVaR) [181], which measures the average cost within a specified upper tail of the cost-return distribution. Chow et al. [182] develop policy-gradient and actor-critic algorithms for CVaR-constrained problems using a Lagrangian formulation, jointly updating the policy and multiplier to optimise performance subject to the risk constraint.

Extending this principle to off-policy deep reinforcement learning, Yang et al. [183] propose Worst-Case Soft Actor-Critic (WCSAC). The method augments the SAC algorithm with a distributional safety critic (see Section 2.2.2.3.3) that estimates the mean and variance of cumulative costs under a Gaussian approximation. These estimates yield a closed-form CVaR measure, which replaces the expected-cost estimate in the policy objective, while an adaptive safety weight responds to violations of the risk budget. Their subsequent extension [184] introduces a quantile-regression alternative, allowing more flexible modelling of non-Gaussian cost distributions for which the Gaussian approximation may underestimate tail risk.

Although these approaches account for high-cost outcomes, risk aversion can restrict exploration and lead to overly conservative policies. ORAC [185] addresses this issue by extending WCSAC with a separate optimistic exploration policy. An ensemble of distributional cost critics provides a lower confidence bound (LCB) on the estimated CVaR, which is combined with a reward upper confidence bound (UCB) to encourage exploration of potentially rewarding actions. Concretely, for a Gaussian target policy $\pi_T(\cdot \mid s) = \mathcal{N}(\mu_T, \Sigma_T)$, the exploratory mean is shifted along the gradient of this optimistic objective:

$$\mu_E = \mu_T + \delta \frac{\Sigma_T, g}{|g|\Sigma_T} \qquad g = \nabla_a \Big[ \hat{Q}^{\mathrm{UB}} R(s,a) - \bar{\lambda} \hat{Q}^{\mathrm{LB}} C(s,a) \Big] \Big| a = \mu_T \tag{2.56}$$

where $\hat{Q}^{\mathrm{LB}} C$ denotes the optimistic CVaR estimate, $\delta$ controls the exploration shift, and $|g|\Sigma_T = \sqrt{g^\top \Sigma_T g}$. The exploration weight $\bar{\lambda}$ is reduced when this estimate falls below the budget $\chi$, increasing the relative influence of reward. This retains the underlying risk-sensitive policy updates while using optimism to mitigate excessive conservatism during exploration.

While PID-Lagrangian improves multiplier dynamics and CVaR-based extensions change the safety criterion, both retain a penalty-based optimisation mechanism whose practical constraint satisfaction depends on estimation accuracy and the interaction between policy and multiplier updates. Constrained Policy Optimisation (CPO) [169] instead incorporates an expected-cost constraint directly into a local policy-update problem, maximising a surrogate reward objective subject to a cost bound and a KL trust region. Instead of directly maximising reward and minimising cost within the actor's loss, CPO constructs separate first-order surrogate losses around the current policy $\pi_\theta$ using importance-weighted advantage estimates:

$$\mathcal{L}_R^{(t)} = \mathbb{E}_{s,a \sim \pi_{\theta_t}} \left[ \frac{\pi_\theta(a|s)}{\pi_{\theta_t}(a|s)} A_R^{\pi_{\theta_t}}(s,a) \right] \quad \mathcal{L}_C^{(t)} = \mathbb{E}_{s,a \sim \pi_{\theta_t}} \left[ \frac{\pi_\theta(a|s)}{\pi_{\theta_t}(a|s)} A_C^{\pi_{\theta_t}}(s,a) \right] \tag{2.57}$$

where $A_R$ and $A_C$ denote the advantage functions for reward and cost, respectively. The CPO update is then obtained by maximising the reward surrogate over a constrained update region: policy changes are restricted to remain within a KL ball around $\pi_{\theta_t}$, and the cost surrogate is required to stay within the remaining budget. The feasible update region is then:

$$\Theta = \{\theta : D_{\mathrm{KL}}(\pi_{\theta_t} || \pi_\theta) \leqslant \delta, \quad L_C^{(t)} \leqslant \chi - \mathbb{E}_{s \sim \mathcal{B}}[Q_C(s, \pi_{\theta_t(s)})]\} \tag{2.58}$$

CPO then selects the next parameters by maximising the reward surrogate within this constrained region $\theta_{t+1} \in arg\max_{\theta \in \Theta} L_R^{(t)}$. Thus, reward is improved only among updates that are locally predicted to respect the cost budget, while the KL trust region limits distributional shift so that the surrogates remain reliable. To solve this efficiently, the surrogate losses are linearised, and the KL constraint is approximated to the second order via a Fisher information matrix.

Unlike Lagrangian methods, which treat safety as a dual objective, surrogate methods treat safety as a sequence of locally solvable constrained updates. A seminal method by Yu et al. [174]

constructs local quadratic surrogates of the reward and cost objectives. Under the current policy $\pi_{\theta_t}$ parameters at iteration $t$, the expected discounted reward and cost returns:

$$J_R(\theta) = \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k R(s_k, a_k)\Big], \qquad J_C(\theta) = \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k C(s_k, a_k)\Big] \tag{2.59}$$

Instead of forming a Lagrangian, the method constructs quadratic surrogate losses around $\theta_t$ using first-order information and an explicit curvature term $\tau > 0$. The reward surrogate is then defined as:

$$\mathcal{L}_R^{(t)}(\theta) = -\Big(J_R(\theta_t) + \nabla_\theta J_R(\theta_t)^\top(\theta - \theta_t)\Big) + \frac{\tau}{2}\|\theta - \theta_t\|^2, \tag{2.60}$$

with an analogous surrogate for the expected cost,

$$\mathcal{L}_C^{(t)}(\theta) = J_C(\theta_t) + \nabla_\theta J_C(\theta_t)^\top(\theta - \theta_t) + \frac{\tau}{2}\|\theta - \theta_t\|^2. \tag{2.61}$$

The policy parameters are then obtained by solving a convex constrained sub-problem at each iteration:

$$\theta_{t+1} \in \arg\min_\theta \ \mathcal{L}_R^{(t)}(\theta) \quad \text{s.t.} \quad \mathcal{L}_C^{(t)}(\theta) \leqslant \chi \tag{2.62}$$

The quadratic regulariser in both the reward and cost surrogates enforces strong convexity, ensuring the sub-problem is well-posed even under stochastic gradient estimates. In contrast to Lagrangian methods, feasibility is therefore enforced directly in the surrogate update rather than indirectly via penalty tuning. While CPO enforces safety using TRPO-style likelihood-ratio surrogates and a KL trust region, Yu et al. adopt a more explicit surrogate optimisation perspective with a sequence of convex surrogate problems that are solved directly at each update.

Augmentation methods take a different stance from both Lagrangian penalties and surrogate constrained updates: rather than optimising under a constraint using an external mechanism (a dual variable $\lambda$ or a constrained sub-problem), they change the state representation so that the notion of safety becomes part of the Markov state itself. The original constraint depends on the trajectory (accumulated cost), which is non-Markov in $s_t$ alone; augmentation restores Markovian structure by introducing an auxiliary safety state that deterministically tracks remaining cost budget. This yields an unconstrained MDP over an augmented state space in which standard Bellman-style reasoning and off-the-shelf RL algorithms apply.

Sauté RL [175] defines an augmented state $\tilde{s}_t = (s_t, \bar{c}_t)$, where $\bar{c}_t$ is the remaining discounted safety budget:

$$\bar{c}_{t+1} = \frac{\bar{c}_t - C(s_t, a_t)}{\gamma} \tag{2.63}$$

where $\bar{c}_0 = \chi$ so that $\bar{c}_t \geqslant 0$. The constraint is then enforced by reshaping the task objective on the augmented MDP: transitions that enter $\bar{c}_t < 0$ incur a penalty (conceptually $+\infty$, a large constant $\beta$ in practice), so optimal policies are driven to remain in the safe subset of the augmented state space.

The key benefit is that safety becomes Markovian: the optimal action can depend on $(s_t, \bar{c}_t)$, enabling critic-based methods to learn value functions that encode both task progress and safety adherence.

Optimisation-based methods make safety tractable within gradient-based learning by expressing constraints as quantities the agent can estimate and act on during training (similar to the reward function), enabling scalable use with modern function approximators. Their central limitation is that safety is enforced through approximate optimisation signals, so constraint satisfaction can depend on the quality of estimation, optimisation dynamics, and distribution shifts during learning.

#### 2.2.2.2.2 Uncertainty

Uncertainty-based methods treat safety as a knowledge problem as much as an optimisation problem: violations arise not only from deliberately risky behaviour, but from acting confidently in regions where the agent's predictions are unreliable. Rather than enforcing safety primarily through penalties or constrained updates, these approaches quantify epistemic or predictive uncertainty and use it as a control signal to regulate exploration and execution.

Kahn et al. [177] propose an uncertainty-aware approach for collision avoidance that enforces safety through uncertainty-modulated costs. Instead of treating safety purely as an expected cumulative-cost constraint, they learn a parametric model $P_\theta(\text{COLL}|s_t, a_{t:t+H})$ that predicts the probability of a collision event COLL across a horizon $H$. The key mechanism is that the collision probability used for decision-making is risk-averse: the model outputs a Bernoulli probability via a sigmoid applied to a latent score $\mathfrak{v}$ where the agent inflates risk by adding an uncertainty margin:

$$\tilde{P}_\theta(\text{COLL}|\cdot) = \sigma\left(\mathbb{E}[\mathfrak{v}] + \lambda\sqrt{\hat{\sigma}^2[\mathfrak{v}]}\right) \tag{2.64}$$

so unfamiliar states are predicted as more dangerous. Uncertainty is measured using MC Dropout (see Section 2.2.1.1.1) to yield $\mathbb{E}[\mathfrak{v}]$ and $\hat{\sigma}^2[\mathfrak{v}]$. In effect, uncertainty becomes an explicit safety signal.

Where Kahn et al. use uncertainty to make collision avoidance more conservative at decision time, ACTSAFE [178] uses epistemic uncertainty over the dynamics to shape the learning process itself: it drives exploration toward informative regions while enforcing safety through pessimistic constraint checks. ACTSAFE addresses safe learning under unknown transition dynamics by maintaining an explicit uncertainty set over plausible models and only taking actions that are certified safe under that uncertainty. Formally, it fits a probabilistic dynamics model from collected data. From the model's epistemic uncertainty (using a Gaussian Process, see

Section 2.2.1.1.3), it constructs a confidence set $\mathcal{M}_{\sqcup}(\delta)$ of dynamics that are consistent with the data. At iteration $t$, ACTSAFE then defines a certified safe policy set:

$$\prod_{t}^{\text{safe}} = \{\pi : \max_{P \in \mathcal{M}_t(\delta)} J_C(\pi, P) \leqslant \chi\} \tag{2.65}$$

which represent possible policies whose predictive cumulative cost remains within the cost budget $\chi$ even under the worst plausible dynamics. Within this restriction, ACTSAFE uses uncertainty in two complementary ways. During safe exploration, it selects a policy that maximises an uncertainty-seeking objective, to gather informative data while staying certified safe. As uncertainty contracts and the certified safe set expands, it transitions to performance optimisation, choosing policies that maximise return while retaining the same pessimistic safety check.

Uncertainty-based methods frame safety as a consequence of limited knowledge: risk is highest when the agent acts in regions where its predictions or dynamics estimates are unreliable. These approaches regulate both exploration and execution, either by inflating predicted risk to induce conservative behaviour, or by certifying actions against worst-case plausible models. Their key advantage is adaptivity: as uncertainty shrinks with data, the method can automatically relax conservatism and recover performance. However, the resulting safety guarantees are only as strong as the uncertainty estimator and modelling assumptions, so miscalibration, approximation error, or distribution shift can translate directly into either over-conservatism or residual violations.

#### 2.2.2.3 Combination Methods

Combination methods refer to approaches whose classification can be either knowledge-based or black-box depending on the specific method proposed. These methods are often adaptable, capable of leveraging prior knowledge when available, but also can be designed specifically for scenarios where such information is unavailable. Methods include Lyapunov functions [186–192], shielding [193–207], and safety critics [208–214].

##### 2.2.2.3.1 Lyapunov Functions

Lyapunov functions originate from classical control theory in which they are used to establish the stability of a dynamical system [215, p. 11]. Given a system with a state $s \in \mathbb{R}^n$, a Lyapunov function $LF(s)$ is a scalar function designed to measure the *energy/distance* of the system state from a desired equilibrium point $s^*$. Usually, $LF$ is required to be positive definite near $s^*$, specifically, $LF(s^*) = 0$ and $LF(s) > 0$ for all $s \neq s^*$ in some neighbourhood of $s^*$. Here, if $LF(s_t)$ decreases over time, then $s_t$ converges to $s^*$, ensuring that the system remains stable.

These functions can be used to ensure the safe exploration of RL agents during training. In order to ensure the agent can explore the environment without violating safety concerns, a Lyapunov function $LF(s)$ is designed prior to training based on an expert's knowledge of the environment's dynamics, the task's goal conditions, and the safety concerns. As shown above, the function is constructed to decrease along the agent's trajectory under safe behaviour, encoding a measure

of how close the agent is to the desired behaviour. By constraining the agent to choose actions that ensure this descent property, it is possible to restrict the exploration to safe regions of the state space $S$ [186]. Formally, the Lyapunov function constructed $LF$ is defined such that $LF(s) > 0$ for all states not in the desirable safe region $s \notin S_{safe}$, and $LF(s) = 0$ for all states in the desirable safe region $s \in S_{safe}$, where $S_{safe} \subset S$.

During training, the RL agent is restricted to only select actions that take the agent into the desired safe region or lead to a decrease in the Lyapunov function:

$$A_{LF}(s) = \{a_t \in A(s_t) \mid \mathbb{E}[LF(s_t)] - \mathbb{E}[LF(s_{t+1})] \geqslant \omega \text{ or } s_{t+1} \in S_{safe}\} \tag{2.66}$$

where $\omega > 0$ is the minimum amount of *energy* that is needed to decrease for the action to be included in the constrained action set in the case that the next state $s_{t+1}$ does not bring the agent into the desired safe region.

Berkenkamp et al. [187] build on this foundation by relaxing some key assumptions. Firstly, they remove the requirement for *full* knowledge of the environment's dynamics. Instead of assuming the system is given a priori, they model the unknown aspects of the environment with Gaussian Processes [75]. The environment's dynamics are formalised as $\text{env}(s, a) = \text{env}_{prior}(s, a) + \text{GP}(s, a)$, where $\text{env}_{prior}(s, a)$ is the known prior model and $\text{GP}(s, a)$ is an unknown residual function captured by the GP. This allows the method to not require a complete environment model to work and extends it to environments with noisy dynamics or uncertain transitions. Similarly, rather than defining a fixed desirable safe region $S_{safe}$ before training, they only require an initial safe region and actively learn/expand the safe region during training. The agent finds the most uncertain state-action pairs within the current safe region, measured by the GP's predictive variance. It then collects more data at this point to reduce the uncertainty further. By collecting data points around uncertain regions and still using the Lyapunov descent condition, it allows the GPs estimates to tighten and grow the safe region incrementally without leaving the safe region.

Chow et al. [188] convert this framework into a generalisable method for safe exploration based upon CMDPs. Unlike the previous methods that enforce safety via step-wise Lyapunov descent within a GP-estimated region of attraction, Chow et al. reframe the issue as satisying a trajectory-level constraint on the cumulative constraint cost of the CMDP:

$$\mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t C(s_t, a_t) \right] \leqslant \chi \tag{2.67}$$

Then, rather than expanding and exploring a safe region, this instead enforces that every new policy $\pi$ satisfies a Lyapunov constraint:

$$S_{safe_{\pi,C}}[LF](s) \leqslant LF(s), \ \forall s \in S \tag{2.68}$$

meaning that the expected future cost from state $s$ remains bounded. This allows safe exploration in the policy space, instead of the state space [189].

Lyapunov functions offer a theoretically grounded approach for enforcing safety during exploration by ensuring that actions or trajectories remain within safe bounds. They provide strong formal guarantees for safety, and can be applied to model-based and model-free RL algorithms. However, the minimum requirement of a safe initial policy to bootstrap from may not be available in all environments/tasks.

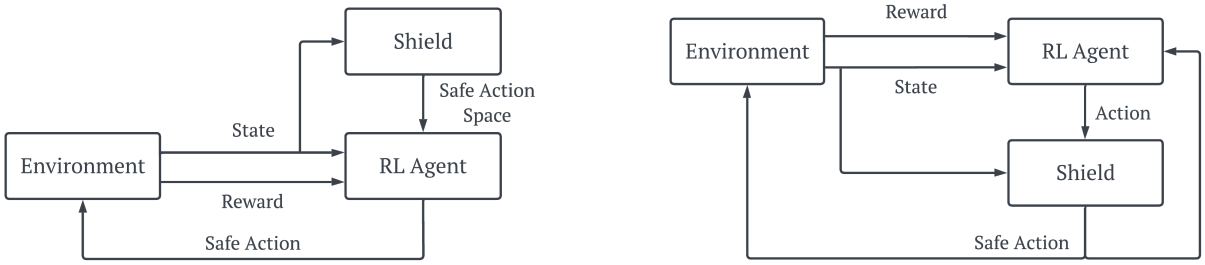
#### 2.2.2.3.2 Shielding

Shielding methods are a set of reactive safety techniques for the safe exploration of RL agents during training designed to interfere with the agent's agent selection process when a potential safety violation could occur [216]. The shield acts as a runtime filter: if the proposed action satisfies all safety requirements, it is allowed; otherwise, it is disallowed or replaced with an alternative safe action. Unlike previous methods discussed, a shield operates at runtime, meaning it does not alter the agent itself. This modularity allows it to be integrated with a wide range of RL algorithms without requiring changes to the underlying learning process.

Shielding methods can be both knowledge-based and black-box methods depending on the design. The shield can be constructed with some explicit knowledge of the environment or formalised safety constraints to ensure rigorous safety satisfaction. Conversely, a shield can be learnt during the RL agent's training to identify safety constraints on the fly and prevent further safety violations. While more flexible, they typically offer weaker safety guarantees and require more exploration.

A shield can be formally defined as a function $\phi : S \times A \rightarrow A$ that monitors the agent's action selection process and ensures adherence to safety constraints. Shielding methods can be categorised into two distinct paradigms based on when the shield intervenes relative to action selection: pre-shielding and post-shielding [217], as illustrated in Figure 2.7. In pre-shielding, the shield constrains the agent's action selection process in advance, restricting the available action space to a safe subset $A_{safe}(s) \subseteq A$ in state $s$. This ensures that unsafe actions are never considered during learning or execution as it requires the agent to operate within a limited action space throughout training. In contrast, in post-shielding, the agent selects an action according to its policy $\pi(s)$ as usual, and the shield intervenes only if the chosen action violates a safety constraint. If the post-shield detects a possible violation, then it replaces the chosen action with a safe alternative [193].

The concept of shielding was formally introduced by Alshiekh et al. [193], who proposed a framework to synthesising shields from linear temporal logic (LTL) [218] specifications. Given a safety specification, expressed as a set of LTL formulae describing allowed system behaviours, it is translated into a deterministic safety automaton $\varphi_{safety}$, where all states in it's definition are considered safe. To enable planning within the MDP and ensuring the safety specifications $\varphi_{safety}$, an abstraction of the environment $\varphi_{env}$ is constructed according to $\varphi_{safety}$ to determine the feasible safe traces that can be taken throughout the MDP. Shielding is then framed as a




(a) Pre-shielding (b) Post-shielding

Figure 2.7: Architectural diagram of (a) pre-shielding and (b) post-shielding within the RL loop.

two-player safety game, where the environment selects an observation and the system selects an action. By synthesising the shield with knowledge of both the safety requirements and an abstraction of the environment, the framework guarantees correctness during both training and execution. Both pre-emptive and post-posed implementations of shielding can be applied without compromising these guarantees.

The concept of synthesising shields from safety specifications has been extended beyond the standard MDP setting. In probabilistic shielding [194–196], shields are synthesised to perform probabilistic checks, ensuring that safety constraints are satisfied with high probability rather than guaranteeing absolute safety deterministically. In the multi-agent setting [197], shielding techniques have been adapted to coordinate multiple learners. For partially observable environments [198], shields are constructed based on beliefs over states, enabling safe exploration under uncertainty where the true system state cannot be directly observed. However, these methods are limited to discretised environments, which restricts their applicability in real-world settings. Kim et al. [199] extend shields to continuous spaces using specification written in $\text{LTL}_t$ (LTL modulo real-arithmetic theories).

While shielding offers strong correctness guarantees, it also comes with practical limitations. Requiring the shield to be invoked at every time step can significantly restrict exploration even when the agent's policy is already behaving safely in most regions of the state space. A hybrid approach that leverages formal verification to identify regions of the state space where the agent's policy is provably safe can be utilised to make sure the shield is only invoked where needed, in regions where unsafe behaviour is possible [200].

Just as restricted exploration is a limitation for previously discussed shielding approaches, so is the reliance on explicit prior knowledge of the environment and safety dynamics, which limits their applicability in complex or high-dimensional domains. Recent methods address this by replacing the need for a hand-crafted abstraction with a learned world model [201, 202]. The shield is constructed online using a latent dynamics model, typically a DreamerV2/V3 model [52], trained directly on experience from the environment. Rather than querying a formal abstraction of the environment hand-crafted by an expert, the shield performs rollouts using the latent dynamics model to verify if future trajectories satisfy the safety property $\varphi$. While a learned environment model can only provide probabilistic safety, it relaxes the assumption that the full

model and its dynamics is known a priori. Similarly, instead of a latent dynamics model Tappler et al. [203] learn an abstract MDP $\hat{M}$ automata. Just like Dreamer, this can then be queried alongside the LTL safety specification $\varphi$ to ensure safety.

While model-based methods alleviate the need for expert-defined environment dynamics, they still rely on learning abstract representations of the environment and require safety priors. A further step towards black-box shielding is to construct the shield directly from experience, with no requirement to model transition dynamics explicitly. One way is to learn a shield by identifying non-accepting terminal state-action pairs during training, and then forbid them if the agent attempts the same pair in the future [205]. As the agent interacts with the environment, when a non-accepting terminal state-action pair is identified, the episode is reset and the pair is stored within the shield and labelled $g_\phi : S \times A \to \{0, 1\}$ as either safe (1) or unsafe (0). Then, at every timestep, the shield checks to see if the current state-action pair $(s_t, a_t)$ has been encountered before. If it has, and the label is 1, then it disallows the action and replaces it with a safe action, if the label is 0 then the action is permitted. If the pair has never been encountered before, the action is also permitted and labelled according to the transition observed. However, this struggles at continuous or higher-dimensional data as the likelihood that all state and action features align with previously seen data becomes improbable.

Safety shields offer a flexible and reactive framework in which unsafe actions can be rejected and replaced with safe actions, working analogous to the actual RL algorithm. They can be synthesised from full formal knowledge of the environment and safety constraints, or learned entirely from interaction data without prior assumptions. This makes shielding applicable across a wide range of tasks, from discrete, structured domains to high-dimensional, continuous environments. While formally constructed shields provide strong safety guarantees, learned or black-box variants offer greater scalability and adaptability at the cost of probabilistic or empirical guarantees.

#### 2.2.2.3.3 Safety Critics

Safety critics constitute a family of methods that augment the learning process with a secondary critic network, aptly named the safety critic, explicitly optimised to predict safety-related signals, such as the expected cost, violation probability, or risk of entering unsafe states. Unlike standard critics, which estimate value functions relative to reward objectives, safety critics estimate alternative value functions relative to safety metrics, often defined through the cost function $C(s, a)$ of a CMDP and constrained via a budget $\chi$. These safety-specific estimates enable the agent to anticipate potential violations and adjust its policy accordingly, even when such violations are rare or delayed.

In contrast to shielding-based approaches that act as external safety mechanisms on agent behaviour, safety critics are integrated directly into the learning loop and influence gradient-based updates of the agent's parameters. This tight coupling allows safety signals to guide both exploration and exploitation by shaping the optimisation landscape through auxiliary loss terms or constrained policy updates.

The foundational work by Dalal et al. [208] introduced the first safety critic, enforcing state-

wise safety constraints via direct policy correction. At the heart of this method is a safety layer, which intercepts the raw actions produced by a policy and minimally modifies them to satisfy predefined safety constraints. The critic network $\mathcal{F}_\theta$ is pretrained using a dataset $\mathcal{D}_{\text{pretrain}} = \{(s_t, a_t, s_{t+1})\}$, collected via random exploration in the same environment. Although the dataset contains no explicit cost labels, it is assumed that the cost function $C_{\text{constraint}}(s, a)$ is known and can thus be evaluated directly at states $s_t$ and $s_{t+1}$, providing targets for training.

Specifically, for each constraint, the critic learns a linear approximation of how the safety signal changes with the chosen action. Given state $s$, the critic network outputs coefficients that define this first-order model:

$$C_{\text{constraint}}(s, a) \approx C_{\text{constraint}}(s) + \mathcal{F}_\theta(s)^\top a \tag{2.69}$$

Using this linear approximation, the safety layer solves a quadratic optimisation problem at each timestep to project the raw policy action $\pi(s)$ onto the closest feasible action set. Under the simplifying assumption that at most one constraint is active at a time, this optimisation has an analytic closed-form solution, involving only a few basic arithmetic operations. This simplicity ensures differentiability and introduces minimal computational overhead.

Importantly, the approach does not assume knowledge of the policy used during pretraining, making it well-suited for practical scenarios where legacy datasets may come from unknown or complex controllers. Nonetheless, this method assumes immediate observability of constraint violations, fully specified cost functions, and the feasibility of enforcing safety through single-step action corrections, similar to shielding techniques discussed previously.

An alternative approach to safety critics is presented by Srinivasan et al. [211], who propose a method for learning the safety critic itself rather than relying on predefined and explicitly known constraints. This work departs from Dalal et al.'s assumption of known cost functions and instead employs a learned safety value function $Q_{\text{safe}}(s, a)$, parametrised by a neural network trained to predict the likelihood of future constraint violations. The learned critic acts as a generalised safety specification, enabling safety knowledge transfer across tasks and environments.

The core of this method involves two phases. Initially, the safety critic is pretrained in environments where constraint violations can occur in a controlled manner. Specifically, the critic is trained by labelling actions retrospectively as safe or unsafe based on whether they lead to safety violations within a fixed horizon. Formally, the safety critic $\mathcal{F}_{\text{safe}}$ predicts the probability that an action $a$ in state $s$ leads to constraint violations, enabling a constrained policy $\overline{\pi}$ to be defined as follows:

$$\overline{\pi}(s) = \arg \max_{a : \mathcal{F}_{\text{safe}}(s,a) \leqslant \epsilon_{\text{safe}}} \pi(s, a) \tag{2.70}$$

In the second phase, the pretrained critic is transferred to new, more challenging environments, acting as a learned safety constraint that guides exploration and significantly reduces constraint

violations during training. This relaxes a lot of the assumptions of known knowledge of the prior method, but still requires a pre-training phase that may not be feasible in all situations.

Building upon this direction, Zhang et al. [212] eliminate the need for a pretraining phase by jointly learning the safety critic and policy online. The core idea is to treat cost signals $C(s,a)$ and reward signals $R(s,a)$ as parallel feedback from the environment and to integrate safety into the learning loop by modulating the reward function based on safety critic predictions. The safety critic $\mathcal{F}_{\text{safe}}$ estimates the expected future cumulative cost $Q_\theta^C(s,a)$ and this estimate is used to scale the reward via a cost-aware weighting function, resulting in a modulated reward.

This reward transformation effectively converts the constrained optimisation problem into an unconstrained one, enabling the use of standard policy gradient updates while adhering to a safety budget. To ensure stability and smooth learning, the authors employ a differentiable exponential weighting scheme and implement a dynamic cost threshold schedule that allows more exploration early in training before gradually tightening safety adherence.

A more cautious variant of safety critics is introduced by Bharadhwaj et al. [213], who propose a conservative safety critic that explicitly overestimates the risk of the state-action space. The goal is not to estimate the true expected cost $Q^C(s,a)$ as accurately as possible, but rather to learn a deliberately conservative approximation that systematically upper bounds it across the space, thereby promoting safety through pessimism. This overestimation discourages the agent from taking actions that could potentially incur high costs, even when empirical data may suggest otherwise. Formally the safety critic is trained via the following objective:

$$\begin{aligned}\hat{Q}^C \leftarrow \arg\min_{Q^C}\ \eta \cdot \left(-\mathbb{E}_{s\sim\mathcal{D}_{\text{buffer}},\, a\sim\pi_\theta(a|s)}\left[Q^C(s,a)\right] + \mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{buffer}}}\left[Q^C(s,a)\right]\right) \\ + \frac{1}{2}\mathbb{E}_{(s,a,s_{t+1},c)\sim\mathcal{D}_{\text{buffer}}}\left[\left(Q^C(s,a) - \mathcal{B}^{\pi_\theta}\hat{Q}^C(s,a)\right)^2\right]\end{aligned} \tag{2.71}$$

The first term of this objective encourages the safety critic to assign higher costs to new actions that the policy might want to try. The second term encourages the safety critic to learn the expected cumulative cost under the current policy using the Bellman target. Together, these terms produce a safe, accurate, yet conservative safety critic.

Safety critics provide a principled mechanism for integrating constraint satisfaction directly into the reinforcement learning objective by estimating expected cumulative cost signals alongside rewards. They can be trained using pre-existing data or learned jointly with the policy, allowing for a range of instantiations with varying assumptions. Pretrained safety critics provide strong safety performance; however, learnt variants offer greater adaptability.

## 2.3 Gaps and Future Directions

The preceding sections have surveyed three closely related but often independently developed bodies of work: RL methods in Section 2.1.1, UQ methods in Section 2.2.1, and safe RL methods in Section 2.2.2. This section summarises the reviewed material to identify key gaps that arise

Table 2.1: Overview of uncertainty quantification methods, highlighting trade-offs between expressiveness, computational cost, usability, interpretability, and formal guarantees.

| Method | Uncertainty Type | Computation (H/M/L) | Post-hoc | Interpretability | Guarantees |
|---|---|---|---|---|---|
| BNN | Epistemic | High | ✗ | ✗ | - |
| LA | Epistemic | Med | ✓ | ✗ | - |
| VI | Epistemic | Med | ✗ | ✗ | - |
| MCMC | Epistemic | High | ✗ | ✗ | Asymptotically exact posterior |
| MC Dropout | Epistemic | Med | ✓ | ✗ | - |
| Adversarial Models | Epistemic | High | ✓ | ✗ | - |
| Influence Functions | Epistemic | Med/High | ✓ | ✗ | - |
| Ensembles | Epistemic | High | ✗ | ✗ | - |
| GPs | Epistemic | High | ✗ | ✓ | Exact posterior |
| Distance-Aware Models | Epistemic | Med | ✗ | ✗ | - |
| Parametric Predictive Distributions | Aleatoric | Low | ✗ | ✓ | - |
| Non-parametric Predictive Intervals | Aleatoric | Low/Med | ✗ | ✓ | - |
| VAE-based models | Aleatoric | Med | ✗ | ✗ | - |
| GAN-based models | Aleatoric | High | ✗ | ✗ | - |
| TTA | Aleatoric | Med | ✓ | ✗ | - |
| Hybrid Approaches | Both | High | ✗ | ✓ | - |
| Full CP | Both | High | ✓ | ✓ | Finite-sample coverage (IID) |
| Split CP | Both | Low | ✓ | ✓ | Finite-sample coverage (IID) |
| EDL | Both | Low | ✗ | ✓ | - |

when these methods are considered in isolation, as well as open challenges that emerge to motivate future work.

### 2.3.1 Comparing Uncertainty Quantification Methods

Section 2.2.1 reviewed a diverse set of UQ methods that differ not only in their modelling assumptions, but also in the form and reliability of the uncertainty information they produce. To consolidate these differences, Table 2.1 provides a high-level comparison of the principal method families along six axes that are particularly salient for use in safety-critical settings: the type of uncertainty targeted (epistemic, aleatoric, or both), computational burden (High/Medium/Low), whether the technique can be applied post-hoc to an existing model, whether the resulting uncertainty representation is directly actionable/interpretable (e.g., bounded scores, prediction intervals, or prediction sets), and whether the method offers any meaningful formal guarantee. Importantly, Table 2.1 is not intended to rank techniques by empirical predictive performance, nor to claim that one method family is universally superior. Reported accuracy–uncertainty trade-offs are highly dependent on architecture, dataset, and training setup, and are therefore difficult to compare fairly without a controlled experimental study. Instead, the objective of this comparison is to surface trade-offs that recur across the research landscape, most notably, the tension between expressiveness and computational tractability, the gap between producing an uncertainty signal and producing one that is operationally interpretable for downstream decisions, and the scarcity of approaches that couple practical usability with reliability guarantees.

Epistemic approaches aim to capture uncertainty arising from limited data or model indeterminacy. The comparisons highlight a persistent divide within this class. On one side, Bayesian posterior methods and approximations (e.g., BNNs, VI, Laplace, and MCMC) provide a principled route to epistemic uncertainty, but incur substantial computational overhead and typically

yield predictive variance outputs that are not straightforward to operationalise under. Moreover, the strongest theoretical characterisations (e.g., asymptotic exactness for MCMC) are rarely attainable in modern deep settings. On the other side, scalable alternatives such as MC Dropout, ensembles, distance-aware surrogates, adversarial-model formulations, and influence-functions can often be deployed more readily, sometimes even post-hoc, but the resulting uncertainty estimates are largely proxy-based and lack formal reliability guarantees. Collectively, these trends suggest that epistemic UQ methods often trade either computational tractability or usability, leaving a gap for uncertainty signals that are both practical and dependable in downstream use.

Aleatoric approaches instead model uncertainty that is intrinsic to the data generation process. In the comparison table, these methods are generally more computationally tractable than epistemic techniques, since they can often be implemented through predictive distribution parameterisation or interval estimation. These methods also contain some of the most readily interpretable uncertainty representations, particularly when methods output explicit prediction intervals. However, the comparison also indicates that aleatoric estimates are typically tied to modelling assumptions. Parametric predictive distributions are only meaningful to the extent that the assumed likelihood family reflects the true noise structure. Generative and augmentation-based approaches, such as VAE or GAN sampling and TTA, can capture forms of ambiguity and input sensitivity, yet their uncertainty outputs are often expressed as uncalibrated dispersion scores whose operational meaning may vary across datasets and shift conditions. As a result, while aleatoric UQ is useful for representing data noise and ambiguity, it does not by itself resolve uncertainty arising from novelty or insufficient knowledge, and it rarely provides reliability guarantees that can be carried through to downstream decisions.

Methods that model both epistemic and aleatoric uncertainty attempt to provide a more complete account of predictive reliability. The comparison table indicates that achieving this combined modelling often comes with a non-trivial cost, either because the underlying epistemic component remains computationally heavy or because the resulting uncertainty is still sensitive to approximation choices and calibration quality. The notable exception is CP, which can be applied post-hoc to a wide range of base models and produces interpretable prediction sets or intervals with an explicit user-controlled error level. Importantly, conformal methods also stand out as one of the few families that provide a meaningful formal statement of reliability, namely, finite-sample coverage under exchangeability assumptions. At the same time, these guarantees are not unconditional: they are typically marginal, depend on the calibration data being representative of deployment conditions, and may be strained under distributional shift. Consequently, the combined-uncertainty category illustrates both the promise and the remaining limitations of current UQ practice, in which actionable outputs and formal reliability can be obtained, but often under assumptions that are difficult to maintain in realistic deployment settings.

Taken together, Table 2.1 suggests that UQ methods that are most attractive for deployment under imperfect perception are those that produce uncertainty representations that are both actionable and reliable, while remaining computationally feasible within modern perception pipelines. Specifically, the comparison highlights the importance of (i) uncertainty outputs

Table 2.2: Overview of safe RL methods, highlighting trade-offs between priors, where they apply safety, usability, and formal guarantees.

| Method | System Priors | Safety Priors | What it Constraints | Post-hoc | Requires Cost-Signal | Guarantees |
|---|---|---|---|---|---|---|
| Formal Verification | ✓ | ✓ | Trajectories | ✓ | ✗ | Exact satisfaction |
| Automata | ✗ | ✓ | Trajectories | ✗ | ✗ | Finds satisfying policy |
| Human Knowledge | ✓ | ✓ | Actions | ✗ | ✗ | - |
| Human Intervention | ✓ | ✓ | Actions | ✓ | ✗ | - |
| Bayesian Techniques | ✓ | ✗ | States | ✗ | ✗ | Certifiable bounds |
| Lagrangian | ✗ | ✗ | Policy updates | ✗ | ✓ | - |
| State Augmentation | ✗ | ✗ | States | ✓ | ✓ | - |
| Surrogates | ✗ | ✗ | Policy updates | ✗ | ✓ | - |
| Risk-learning | ✗ | ✗ | Policy updates | ✗ | ✓ | - |
| Uncertainty | ✗ | ✗ | Policy updates | ✗ | ✗ | - |
| Lyapunov Functions | ✓/✗ | ✗ | Policy updates | ✗ | ✗ | Stability |
| Shielding | ✓/✗ | ✓/✗ | Actions | ✓ | ✗ | Certifiable safety/- |
| Safety Critics | ✓/✗ | ✓/✗ | Policy updates | ✗ | ✓ | - |

that admit consistent operational use, such as prediction sets or calibrated intervals rather than unbounded dispersion scores; (ii) post-hoc applicability, since many practical systems rely on pre-trained deep models for perception; and (iii) a notion of reliability that extends beyond heuristic ranking, ideally through calibrated uncertainty with an explicit error control parameter. At the same time, the table makes clear that these desiderata are not simultaneously satisfied by many existing approaches, because methods with strong reliability statements typically rely on assumptions or calibration regimes that may be challenged by distribution shift, while scalable epistemic proxies often lack decision-level guarantees.

### 2.3.2 Comparing Safe Exploration Strategies

Section 2.2.2 reviewed a diverse set of safe RL methods that differ not only in their need for priors, but also in how they constrain the RL agent and their formal safety guarantees. To consolidate these differences, Table 2.2 provides a high-level comparison of the principal method families along six axes that are particularly important for safety-critical settings: the requirement for system priors and safety priors, how they constrain the RL agent (e.g., actions, policy updates, etc.), if they can be applied to a base RL agent with zero modifications, if they require a cost-signal, and whether the method offers any meaningful formal guarantees. Importantly, Table 2.2 is not intended to rank techniques by empirical performance, nor to claim that one family is superior, but rather is intended to expose structural trade-offs and show how different approaches operationalise safety. These trade-offs determine not only the type of safety violations a method can prevent, but also its practical usability.

Knowledge-based methods explicitly rely on prior knowledge about the system or safety structure in order to enforce safety. This includes formal verification and automata-based approaches, which require a formal specification of acceptable behaviour and, in many cases, an explicit or abstracted model of the environment. These methods constrain safety at the trajectory or state-visitation level and can provide strong guarantees, such as verified satisfaction of a specification, when their assumptions hold. Bayesian techniques similarly impose safety by leveraging structured modelling assumptions, for example, through probabilistic dynamics or safety function

estimates with confidence bounds. While these approaches offer principled mechanisms for reasoning about safety and uncertainty, their reliance on system priors or carefully specified safety models limits their applicability in complex domains where accurate modelling is infeasible or where safety constraints cannot be fully specified a priori.

The second class of approaches enforces safety without assuming access to explicit system models or formal specifications. This includes Lagrangian and primal–dual methods, state-augmentation techniques, surrogate-based constraint learning, risk-learning frameworks, and uncertainty-driven safety mechanisms. As reflected in Table 2.2, these methods typically require an explicit cost signal (an unknown cost signal, that is to be optimised upon; similar to a reward function), and constrain safety by shaping policy updates or restricting the learned state distribution during training. Their primary advantage lies in their generality and compatibility with modern deep RL pipelines. However, safety in this setting is inherently data-driven and approximate: constraint satisfaction is achieved only to the extent that costs are accurately observed and learned, and formal guarantees are generally absent. Consequently, these methods may exhibit unsafe exploration, delayed constraint satisfaction, or sensitivity to distributional shift, particularly in early training or under imperfect state estimation.

Some approaches occupy an intermediate position as they can be initialised with prior knowledge, or can be entirely learnt during training. Lyapunov-based methods, shields, and safety critics fall into this category, differing in the stage at which safety is enforced and the strength of their assumptions. Lyapunov-based approaches encode stability or safety conditions into the learning process, typically requiring assumptions about the existence and learnability of a suitable Lyapunov function, while shields constrain executed actions directly and can therefore be applied post-hoc to a trained policy. Safety critics integrate learned estimates of risk or cost into policy optimisation, offering flexibility at the expense of robustness. These hybrid methods can have higher usability than black-box/knowledge-based approaches as they can be initialised or learnt.

Taken together, Table 2.2 shows a fundamental trade-off in safe RL between reliance on prior knowledge, practical deployability, and the strength of safety guarantees. Methods that leverage rich system or safety priors can enforce safety at the level of trajectories with formal assurances, but at the cost of restrictive assumptions. In contrast, black-box learning-based approaches scale to complex environments but typically offer only empirical safety. Hybrid methods partially bridge this gap by allowing for initialisation with priors or learning the safety constraint on-the-fly. These observations highlight the difficulty of achieving safety mechanisms that are simultaneously assumption-light, post-hoc deployable, adaptive, and reliable, motivating the open challenges discussed in the following section.

### 2.3.3 Open Challenges

The comparisons in Tables 2.1 and 2.2 highlight that both UQ and safe RL have made substantial progress, but largely under different assumptions and evaluation criteria. UQ methods primarily focus on producing uncertainty signals that reflect predictive reliability, whereas safe RL methods

focus on constraining behaviour during learning and deployment. In safety-critical settings, these two objectives must ultimately be reconciled: uncertainty estimates must be not only statistically meaningful but also usable as inputs to downstream decision-making, while safety mechanisms must remain reliable when the state information they depend on is imperfect, and the agent's behaviour induces non-stationary data.

Motivated by these considerations, the following points summarise key open challenges emerging from the UQ research landscape, which also inform the broader system-level challenges discussed later in this section:

- **Interpretable uncertainty representations.** A recurring limitation across most UQ methods is that uncertainty is often expressed as predictive variance, entropy, disagreement, reconstruction error, or other unbounded scores whose numerical scale is not comparable across models, datasets, or deployment conditions. A key open problem is to design uncertainty representations that are intrinsically interpretable and operationally stable, such as bounded reliability scores with consistent semantics, or set-based outputs (prediction sets and prediction intervals) that directly encode uncertainty in a form that can be propagated through downstream system components.
- **Reliability under distribution shift and sensor corruption.** Many UQ approaches appear well calibrated in-distribution yet exhibit systematic failures under covariate or distributional shift. This gap is especially pronounced in safety-critical perception, where the most consequential errors occur precisely in regimes that differ from training. A gap is to create UQ methods that can still perform reliably under such failures.
- **Compute-bounded uncertainty suitable for deployment.** There is a trade-off between efficacy and computational performance. Methods with strong empirical performance often require multiple models or large compute, while methods that are inexpensive at inference time may rely on weaker uncertainty proxies. An open challenge is to obtain uncertainty estimates that are both reliable and computationally light, without sacrificing interpretability or robustness under shift.
- **Temporal uncertainty in non-stationary, closed-loop settings.** Most UQ evaluations implicitly assume static prediction tasks with independent test inputs, whereas deployed agents generate temporally correlated observations and induce non-stationary state distributions through their own actions. This creates two related gaps: first, how to aggregate and smooth uncertainty over time in a principled way, and second, how to maintain reliability when the data distribution evolves online. This challenge is particularly relevant for RL systems, where policy updates change the visited state distribution during training and where deployment feedback can shift the operating regime, invalidating assumptions underlying static calibration.

Motivated by the considerations above, the following points summarise key open challenges emerging from the safe RL research landscape:

- **Adaptive conservatism rather than uniform caution.** Many existing approaches

are either uniformly conservative or have no level of conservatism. This creates an undesirable trade-off between constraint satisfaction and performance, and can prevent efficient exploration or lead to over-restriction even in regions that are well understood. An open challenge is to develop safety mechanisms that can relax conservatism when evidence supports safe behaviour and increase conservatism when uncertainty is high, while remaining stable and avoiding oscillatory behaviour.

- **Closing the gap between black-box scalability and knowledge-based guarantees.** Knowledge-based methods can provide strong safety assurances but require system models, abstractions, or formal specifications that are often unavailable in complex environments. Black-box approaches scale more naturally, but typically offer only empirical safety and may violate constraints during learning. A central research direction is to strengthen black-box safe RL by developing safety mechanisms that approach the reliability of knowledge-based methods under weaker assumptions.
- **Safety without explicit cost functions in continuous environments.** Many black-box safe RL methods still assume access to a cost signal, which in practice functions as a safety oracle provided by the environment or designer, even if unknown during training. In continuous control tasks, truly cost-free safety learning, where the agent must infer safety structure without any explicit cost signal, remains an unexplored challenge.
- **Compositional safety for multi-constraint settings.** Real systems typically face multiple interacting constraints, such as collision avoidance, actuator limits, energy budgets, and mission-level requirements. Extending beyond single-cost CMDP formulations to robustly handle multiple constraints, heterogeneous timescales, and competing objectives remains an open challenge, particularly when constraints interact non-linearly or are only partially observed.

# Chapter 3

# Robust Uncertainty Quantification

This chapter introduces an approach for the robust uncertainty quantification of deep learning models that can provide statistically valid coverage of predicted outputs. We assess the proposed method across different learning tasks, including classification, regression, and semantic segmentation. This chapter describes the motivation for the approach in Section 3.1. We then describe the methodology of the approach in Section 3.2, and fully evaluate it for classification and regression tasks in Section 3.3. We then present how the approach can be used within a perception-to-policy framework to enhance safety in Section 3.4, before finally summarising the chapter in Section 3.5.

## 3.1 Motivation

A wide variety of techniques have been proposed to enhance the safety of RL in real-world applications, see Section 2.2.2, each exhibiting differing levels of effectiveness and reliability. However, in many practical systems where safe RL is deployed, such as robotics, autonomous vehicles, and other embodied agents, the overall decision-making architecture is typically modular [219]. In particular, the perception and control subsystems are often developed and trained independently. The perception module processes raw sensory inputs, whether that is an image or LIDAR data, to produce a representation of the environment, while the policy/control module (in some cases, an RL agent) acts upon this representation to select actions that are intended to satisfy safety and performance objectives.

This common architectural separation introduces a critical limitation in such systems: the safety of the learned policy is inherently bounded by the efficacy of perception [220]. If the perception module provides incomplete, noisy, or incorrect observations of the environment, the downstream safe RL policy can only be as safe as the hazards it is able to perceive. As illustrated in Figure 3.1, when the perception module fails to correctly identify all relevant obstacles in the scene, the downstream policy, despite being trained with safety constraints, may generate an action that leads to a collision in the true environment. This coupling between perception uncertainty and policy safety highlights a fundamental vulnerability in existing safety-critical systems, motivating the need for principled methods that can quantify and propagate uncertainty from perception



(a) Input Image (b) Accurate Perception (c) Imperfect Perception

Figure 3.1: Illustration of the effect of imperfect perception on downstream safe policy execution. Image (a) shows the raw sensory observation captured by the agent. In image (b), the perception module provides an accurate segmentation of the scene, enabling the downstream safe policy to generate a trajectory that successfully avoids all obstacles. In contrast, image (c) depicts a case of imperfect perception in which a vehicle is missed by the segmentation network, causing the policy to plan a straight trajectory that would result in a collision. (Image adapted from [221].)

to decision-making in a statistically reliable manner.

While UQ techniques, like those discussed in Section 2.2.1, can partially mitigate this limitation by enabling the downstream policy to reason about perceptual confidence, they do not solve the full issue discussed. Although such methods can highlight regions of high uncertainty in the input space and allow the policy to act more cautiously in ambiguous situations [222], they cannot compensate for systematic perception errors or entirely unobserved hazards. In scenarios where hazards are misdetected or absent from the perceptual representation, even a well-calibrated uncertainty estimate fails to provide actionable information about unseen risks.

Recently, conformal prediction, discussed in Section 2.2.1.3.2, emerged as a promising framework for achieving distribution-free, statistically valid UQ. By constructing prediction sets/intervals that guarantee a desired coverage level under minimal assumptions, conformal methods provide a principled approach to quantify uncertainty in deep learning models. However, despite their advantages, conformal predictors can produce excessively large prediction sets in practice. Such over-conservatism reduces the practical utility of the predictions for downstream decision-making.

Building on the limitations of conventional UQ and the conservatism of CP, we propose **M**onte-**C**arlo **C**onformal **P**rediction (MC-CP), a hybrid UQ framework that combines a novel adaptive Monte Carlo (MC) Dropout method with CP to produce compact prediction sets. Empirical evaluations across standard classification and regression benchmarks demonstrate that MC-CP consistently outperforms state-of-the-art UQ methods. Furthermore, when integrated into a perception-to-policy framework for safe reinforcement learning, MC-CP provides calibrated hazard maps that propagate uncertainty from perception to decision-making. This enables downstream policies to reason over statistically valid, risk-aware representations of their environment, leading to substantial improvements in both perception and navigation safety relative to baseline and alternative conformal variants.

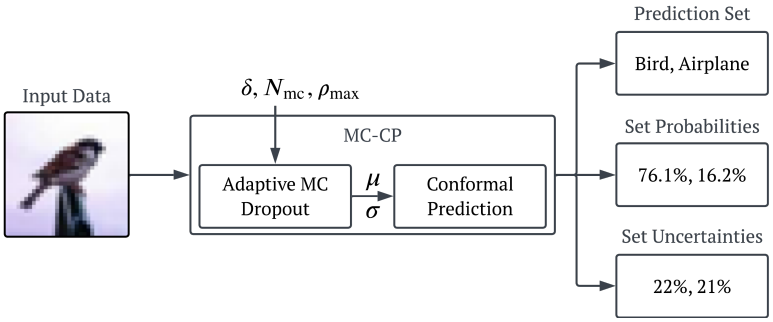



Figure 3.2: High-level overview of the MC-CP framework showing the interaction between adaptive Monte Carlo Dropout and conformal prediction. Given an input sample, the Adaptive MC Dropout module performs stochastic forward passes and monitors variance convergence, producing the mean prediction $\mu$ and uncertainty $\sigma$. These estimates are then passed to the conformal prediction stage, which applies calibration statistics to yield statistically valid prediction sets with associated probabilities and (unconformalised) uncertainties, resulting in compact and well-calibrated estimates.

## 3.2 MC-CP

Building on the conceptual overview introduced above, this section formalises the **M**onte-**C**arlo **C**onformal **P**rediction (MC-CP) framework and details its components as well as how it operates across classification and regression tasks. First, we present a high-level overview, visualised in Figure 3.2, to illustrate the interaction between its two principal components, Adaptive Monte Carlo (MC) Dropout and CP.

Given an input sample, a novel adaptive MC Dropout module performs stochastic forward passes through the neural network to obtain a predictive distribution. Unlike standard MC Dropout, this module monitors the convergence of the predictive variance and terminates sampling once stability is reached, thereby reducing redundant computations. The resulting mean and variance estimates are then passed to the conformal calibration stage, which constructs statistically valid prediction sets (for classification) or prediction intervals (for regression) according to the desired coverage level $1 - \alpha$. The two-stage process of MC-CP allows it to inherit the distribution-free coverage guarantees of CP while regularising both the head and the tail of the predictive distribution: the adaptive MC component stabilises the head by reducing stochastic variance in high-confidence regions, whereas the conformal calibration constrains the tail to achieve statistically valid yet compact prediction sets. Regularising both ends of the predictive distribution is essential, as errors/hidden uncertainties in either region can lead to negative effects: lack of regularisation in the head can lead to overconfidence in the top predictions, while no regularisation in the tail can lead to overly large prediction sets.

In the remainder of this section, we formally define the Adaptive MC Dropout algorithm in Section 3.2.1, and then describe its integration with CP for classification and regression in Section 3.2.2 and Section 3.2.3, respectively.

### 3.2.1 Adaptive Monte Carlo Dropout

Simply combining standard MC Dropout and CP could yield effective results, but such a naive combination would be computationally ineffective. MC Dropout requires a large, fixed number of stochastic forward passes $N_{\text{mc}}$, often in the range of hundreds to thousands per input sample, to obtain a stable estimate of epistemic uncertainty. When naively combined with CP, which itself incurs an additional cost over a held-out calibration dataset and when crafting prediction sets, the overall computation at design- and inference-time scales. This computational overhead becomes prohibitive in real-time or resource-constrained settings such as autonomous vehicles, where a decision/prediction needs to be made quickly [223].

Moreover, using a uniform sampling budget for all inputs disregards the heterogeneity of epistemic uncertainty across the input space [224]. Samples that lie in well-explored/optimised regions of the training distribution typically yield stable predictive estimates after only a few stochastic passes, whereas out-of-distribution or ambiguous samples may require substantially more evaluations before convergence. Fixing the number of passes $N_{\text{mc}}$ for every input $x$ therefore wastes computation on confident cases and can even risk under-sampling uncertain ones depending on the chosen value of $N_{\text{mc}}$. To address this imbalance, we introduce an Adaptive MC Dropout mechanism that dynamically adjusts the number of stochastic forward passes per input according to an online estimate of predictive-variance convergence. This adaptive scheme preserves the fidelity of Bayesian uncertainty estimation while substantially reducing redundant computations.

Building upon the definition of MC Dropout, given in Section 2.2.1.1.1, we recall a neural network is defined as $\mathcal{F}_\theta : X \to Y$. For an input $x \in X$ during inference, dropout layers [81] remain active, and multiple stochastic forward passes are performed to approximate sampling from a posterior distribution over the network's weights $\theta$. The resulting predictive distribution is approximated as:

$$p(y|x, \mathcal{D}) \approx \frac{1}{N_{\text{mc}}} \sum_{n=1}^{N_{\text{mc}}} \mathcal{F}_{\theta_n}(x) \tag{3.1}$$

where $\mathcal{F}_{\theta_n}(x)$ denotes the output of the network with the parameter realisation after applying a dropout mask during the $n$-th stochastic forward pass. The empirical predictive mean and variance are then given by:

$$\begin{aligned} \hat{\mu}(x) &= \frac{1}{N_{\text{mc}}} \sum_{n=1}^{N_{\text{mc}}} \mathcal{F}_{\theta_n}(x) \\ \hat{\sigma}^2(x) &= \frac{1}{N_{\text{mc}} - 1} \sum_{n=1}^{N_{\text{mc}}} (\mathcal{F}_{\theta_n}(x) - \hat{\mu}(x))^2 \end{aligned} \tag{3.2}$$

Here, $\hat{\mu}(x)$ is the model's expected prediction, and $\hat{\sigma}^2(x)$ is the epistemic uncertainty arising from variability in the sampled network weights.

The values $\hat{\mu}$ and $\hat{\sigma}^2$ are formed from repeated evaluations at inference with dropout applied. As $n \in \{1, \ldots, N_{\text{mc}}\}$ increases, the repeated stochastic evaluations with dropout can be viewed as draws from a Bernoulli sampling process. Under this view, the law of large numbers [225, pp. 370–374] and the central limit theorem [226] guarantee that the empirical mean and variance of these draws converge to their population values. In practice, this means that after a certain point, further samples contribute negligibly to the uncertainty estimate, suggesting that the computation can be stopped adaptively once convergence is detected. The following Theorem 3.1 formalises this conclusion.

**Theorem 3.1.** *When the dropout process is modelled as a sequence of independent Bernoulli-distributed masks applied to the neural network $\mathcal{F}_\theta : X \to Y$, the corresponding sequence of outputs obtained from repeated forward passes of a fixed input $x \in X$ constitutes an independent and identically distributed (i.i.d.) sample from the dropout-induced predictive distribution. The empirical mean and covariance of the sample converge almost surely to the true predictive mean and covariance. Moreover, the empirical mean is asymptotically normal with covariance equal to that of the underlying predictive distribution.*

*Proof.* During inference with active dropout, each stochastic forward pass corresponds to applying a random dropout mask $\text{Mask}_i$, whose entries are i.i.d. Bernoulli($p$) variables determining which neurons are dropped. The network output for the $i$-th pass is then $\hat{y}_n = \mathcal{F}_\theta(x; \text{Mask}_n) \in \mathbb{R}^d$ where the only source of randomness is the mask $\text{Mask}_n$. Because the masks are independent and identically distributed [66], the sequence $(\hat{y}_n)_{n \geqslant 1}$ forms an i.i.d. collection of random vectors on $\mathbb{R}^d$, defined on the probability space associated with the dropout mechanism.

Let $\mu := \mathbb{E}[\hat{y}_1]$ and $\mathbb{C} := \text{Cov}(\hat{y}_1)$ be the true predictive mean and covariance of the MC Dropout distribution. These expectations are finite since the network outputs are bounded (e.g., softmax activations lie in $[0, 1]^d$), ensuring $\mathbb{E}||\hat{y}_1||^2 < \infty$. Since we cannot compute the true expectations, we define the empirical estimators:

$$\begin{aligned} \bar{\mu}_{N_{\text{mc}}} &= \frac{1}{N_{\text{mc}}} \sum_{n=1}^{N_{\text{mc}}} \hat{y}_n \\ \bar{\mathbb{C}}_{N_{\text{mc}}} &= \frac{1}{N_{\text{mc}} - 1} \sum_{n=1}^{N_{\text{mc}}} (\hat{y}_n - \bar{\mu}_{N_{\text{mc}}})(\hat{y}_n - \bar{\mu}_{N_{\text{mc}}})^\top \end{aligned} \tag{3.3}$$

To establish the strong consistency of $\bar{\mu}_{N_{\text{mc}}}$ and $\bar{\mathbb{C}}_{N_{\text{mc}}}$, consider first the sample mean. By the vector-valued Kolmogorov Strong Law of Large Numbers [227, pp. 81–83], since the first moment $\mathbb{E}||\hat{y}_1|| < \infty$ is finite, then $\bar{\mu}_{N_{\text{mc}}} \to \mu$ almost surely as $N_{\text{mc}} \to \infty$. This guarantees that the empirical mean computed from repeated dropout passes converges pointwise to the expected predictive output of the stochastic network.

In terms of the covariance estimator, we can rewrite it equivalently as:

$$\bar{\mathbb{C}}_{N_{\text{mc}}} = \frac{N_{\text{mc}}}{N_{\text{mc}} - 1} \left( \frac{1}{N_{\text{mc}}} \sum_{n=1}^{N_{\text{mc}}} \hat{y}_n \hat{y}_n^\top - \bar{\mu}_{N_{\text{mc}}} \bar{\mu}_{N_{\text{mc}}}^\top \right) \tag{3.4}$$

Since the second moment $\mathbb{E}||\hat{y}_1||^2 < \infty$ is finite, each entry of $\hat{y}_n\hat{y}_n^\top$ has a finite expectation (see Lemma A.1), and hence the strong law applies once again:

$$\frac{1}{N_{\text{mc}}}\sum_{n=1}^{N_{\text{mc}}}\hat{y}_n\hat{y}_n^\top \xrightarrow{\text{a.s.}} \mathbb{E}[\hat{y}_1\hat{y}_1^\top] \tag{3.5}$$

Combining this with $\bar{\mu}_{N_{\text{mc}}} \to \mu$ almost surely and using the continuous mapping theorem yields:

$$\bar{\mathbb{C}}_{N_{\text{mc}}} \xrightarrow{\text{a.s.}} \mathbb{E}[\hat{y}_1\hat{y}_1^\top] - \mu\mu^\top = \mathbb{C} \tag{3.6}$$

Hence, both the sample mean and covariance computed from MC-dropout predictions are strongly consistent estimators of their true counterparts.

Having established the strong consistency of the empirical mean and covariance estimators, we now characterise their asymptotic distribution. While strong consistency guarantees that MC Dropout estimates converge almost surely to their true counterparts as $N_{\text{mc}} \to \infty$, it does not describe the fluctuations of these estimators for finite sample sizes. To capture this behaviour, we invoke the multivariate Central Limit Theorem (CLT), which provides a Gaussian approximation to the sampling distribution of the empirical mean. Specifically, for the i.i.d. sequence $\hat{y}_n$ with finite covariance $\bar{\mathbb{C}}$:

$$\sqrt{N_{\text{mc}}}(\hat{\mu}_{N_{\text{mc}}} - \mu) \Rightarrow \mathcal{N}_D(0, \bar{\mathbb{C}}) \tag{3.7}$$

where $\Rightarrow$ denotes convergence in distribution. This means that, for large $N_{\text{mc}}$, the distribution of the sample mean around the true mean is approximately Gaussian with covariance $\bar{\mathbb{C}}/N_{\text{mc}}$. Intuitively, as the number of stochastic passes increases, the random fluctuations in the empirical mean shrink at the rate $1/\sqrt{N_{\text{mc}}}$, while its variability across repeated sampling behaves according to a normal law. □

In summary, Theorem 3.1 and the corresponding proof confirm that the randomness induced by dropout during inference behaves like a well-defined sampling process: as the number of stochastic passes increases, the empirical mean and variance settle to their true predictive values. Once this stabilisation occurs, further forward passes contribute negligibly. This key insight motivates the concept that a variant of MC Dropout can be created that tracks the variance during execution to stop the procedure early when this stabilisation occurs; we refer to this approach as Adaptive MC Dropout.

To turn this insight into a practical method, we operationalise this previous theorem in Algorithm 1 with the Adaptive MC Dropout extension. Firstly, as standard in traditional MC Dropout, for each input $x$, we perform multiple stochastic forward passes through the network. At each stochastic iteration $n \in \{1, \ldots, N_{\text{MC}}\}$, the network performs a forward pass under a new dropout mask, yielding the prediction $\hat{y}$. This prediction is added to the set of all predictions $\mathcal{P}$ for the whole MC Dropout process. The empirical mean and variance are then estimated

**Algorithm 1** Adaptive Monte Carlo Dropout

**Require:** Model $\mathcal{F}_\theta$, Input $x$, Maximum forward passes $N_{\text{mc}}$, Threshold $\delta$ and Patience $\rho_{\max}$

1: $\rho \leftarrow 0; \quad \mathcal{P} \leftarrow 0; \quad \hat{\sigma}^2_{n-1} \leftarrow 0$
2: **for** $n \in \{1, \ldots, N_{\text{MC}}\}$ **do**
3: $\quad \hat{y} \leftarrow \mathcal{F}_{\theta_n}(x)$
4: $\quad \mathcal{P} \leftarrow \mathcal{P} \cup \hat{y}$
5: $\quad \hat{\mu} \leftarrow \frac{1}{|\mathcal{P}|} \sum_{i=1}^{|\mathcal{P}|} \hat{y}_i$
6: $\quad \hat{\sigma}^2_n \leftarrow \begin{cases} \dfrac{1}{|P|-1} \displaystyle\sum_{i=1}^{|P|} (\hat{y}_i - \hat{\mu})^2, & |P| > 1 \\ 0, & \text{otherwise} \end{cases}$
7: $\quad \rho \leftarrow \begin{cases} \rho + 1, & \text{if } |\mathcal{P}| > 2 \ \wedge \ (\hat{\sigma}^2_n - \hat{\sigma}^2_{n-1}) \leqslant \delta, \\ 0, & \text{otherwise.} \end{cases}$
8: $\quad$ **if** $\rho > \rho_{\max}$ **then**
9: $\quad\quad$ terminate loop
10: $\quad$ **end if**
11: $\quad \hat{\sigma}^2_{n-1} \leftarrow \hat{\sigma}^2_n$
12: **end for**
13: **return** $\overline{\mathcal{P}}$, $\hat{\sigma}^2_n$

using the set of predictions so far (lines 5 and 6), which correspond directly to the estimators formalised in Equation 3.3 in Theorem 3.1. In practice, the covariance estimator is applied in its diagonal form to yield per-output variances, as this reduces computational cost and remains sufficient for monitoring uncertainty convergence in each output dimension. However, if $\hat{y}$ is multidimensional, e.g. in the case of binary or multi-class classification tasks, then the variance estimator (line 6) should use the element-wise product $(\hat{y}_i - \hat{\mu}) \odot (\hat{y}_i - \hat{\mu})$. To operationalise the convergence guarantee established in Theorem 3.1, Adaptive MC Dropout introduces a convergence counter $\rho$, a patience value $\rho_{\max}$, and a threshold $\delta$. After the empirical estimators have been updated, the change in variance is measured $\hat{\sigma}^2_n - \hat{\sigma}^2_{n-1}$ between the variance of this forward pass and the previous one. If the change in variance (for all dimensions if $\hat{y}$ is multidimensional) is below the threshold $\delta$, then the counter $\rho$ is incremented, indicating empirical stability of the estimator for this forward pass. If the change exceeds $\delta$, the counter is reset to zero. Once the counter exceeds the patience value ($\rho_{\max}$ consecutive forward passes), the process is deemed to have converged, indicating continued empirical stability, and the MC Dropout sampling is terminated early. The process then returns the mean of the predictions $\bar{\mathcal{P}}$ from the whole MC Dropout process, and the last estimate of variance $\hat{\sigma}^2_n$.

The user-defined hyperparameters introduced by Adaptive MC Dropout, threshold $\delta \in (0, 1)$ and patience $\rho_{\max} \in \mathbb{Z}_+$, control the sensitivity of the method. As $\delta \to 1$, the method becomes less sensitive since larger changes in the estimated variance $\hat{\sigma}^2_n$ are considered for convergence. In contrast, as $\delta \to 0$, the method becomes more sensitive to fluctuations in variance, requiring the variance to stabilise more tightly before convergence is determined. This, as a result, typically

increases the number of stochastic forward passes but allows the method to converge closer to the mean and variance. The patience parameter $\rho_{\max}$ further regulates this behaviour by determining the number of consecutive iterations that must satisfy the convergence condition; larger values enforce stricter convergence at the cost of additional computation.

While the theoretical analysis ensures asymptotic consistency of the estimators, in practice, the adaptive stopping of the MC Dropout process directly influences the computational complexity of the method. For the standard MC Dropout process, we denote $d_{\hat{y}}$ as the dimensionality of the network's output. Given the computational cost of each forward pass $\text{cost}(\mathcal{F}_{\theta_n})$, which depends on the network's architecture and input size, the total complexity of the standard MC Dropout process is:

$$\mathcal{O}(N_{\text{mc}} \cdot \text{cost}(\mathcal{F}_{\theta_n})) + \mathcal{O}(d_{\hat{y}} \cdot N_{\text{mc}}) \tag{3.8}$$

where the first term represents the repeated forward passes of the network, and the second term accounts for the incremental updates of the estimated mean and variance across $d_{\hat{y}}$ output dimensions. In comparison, Adaptive MC Dropout retains the same worst-case complexity but reduces the expected computational cost by introducing an adaptive stopping mechanism. We let $N^*_{\text{mc}}(x)$ be the number of stochastic forward passes actually required for convergence for input $x$, such that $\rho_{\max} \leqslant N^*_{\text{mc}}(x) \leqslant N_{\text{mc}}$. The complexity is therefore:

$$\mathcal{O}(N^*_{\text{mc}}(x) \cdot \text{cost}(\mathcal{F}_{\theta_n})) + \mathcal{O}(d_{\hat{y}} \cdot N^*_{\text{mc}}(x)) \tag{3.9}$$

The additional operations introduced by the adaptive stopping (e.g., computing the variance difference, counter updates, etc) are $\mathcal{O}(d_{\hat{y}})$ per iteration and are therefore encapsulated within the second term. In the worst-case scenario, where convergence is not reached within $N_{\text{mc}}$ passes, then Adaptive MC Dropout reduces to standard MC Dropout. In practice, however, $N^*_{\text{mc}}(x)$ is typically much smaller than $N_{\text{mc}}$, particularly for in-distribution or high-confidence inputs (this is shown explicitly in Section 3.3.1. Therefore, our proposed variant achieves substantial computational savings.

### 3.2.2 MC-CP for Classification

The next stage of MC-CP is to integrate the estimates yielded from Adaptive MC Dropout into the CP framework. While Adaptive MC Dropout stabilises the head of the predictive distribution by reducing stochastic variance in high-confidence regions, conformal calibration constrains the tail by enforcing valid coverage over uncertain samples. Put together, these components create prediction sets that are both statistically grounded and computationally efficient.

Building upon the CP formulation introduced in Section 2.2.1.3.2, this section details the methodology for MC-CP for classification. The objective of MC-CP is to leverage the regularised predictions and epistemic uncertainty estimates from Adaptive MC Dropout to construct statistically valid, conservative prediction sets. To do this, MC-CP is carried out in two stages:

**Algorithm 2** MC-CP calibration for classification

**Require:** Model $\mathcal{F}_\theta$, Calibration dataset $\mathcal{D}_{\text{cal}}$, Maximum forward passes $N_{\text{mc}}$, Threshold $\delta$, Patience $\rho_{\max}$, Regularisation coefficient $\beta$, Number of unpenalised classes $N_{\text{reg}}$, Stochastic smoothing boolean rand, and Miscoverage rate $\alpha$

1: $\mathcal{K} \leftarrow 0$;
2: **for** $(x_i, y_i) \in \mathcal{D}_{\text{cal}}$ **do**
3: $\quad \hat{y}, \hat{\sigma}^2 \leftarrow \text{AdaptiveMCDropout}(\mathcal{F}_\theta, x_i, N_{\text{mc}}, \delta, \rho_{\max})$
4: $\quad \hat{y}^{\text{sorted}} \leftarrow [\hat{y}_1^{\text{sorted}}, \ldots, \hat{y}_\Xi^{\text{sorted}}] \quad \text{s.t.} \quad \hat{y}_1^{\text{sorted}} \geqslant \ldots \geqslant \hat{y}_\Xi^{\text{sorted}}$
5: $\quad \text{rank}_{y_i} \leftarrow \min\{\ \xi | \hat{y}_k^{\text{sorted}} = \hat{y}_{y_i}\}$
6: $\quad \kappa_i \leftarrow \sum_{\xi=1}^{\text{rank}_{y_i}} \hat{y}_\xi^{\text{sorted}} + \beta(\text{rank}_{y_i} - N_{\text{reg}})_+$
7: $\quad$ **if** rand **then**
8: $\quad\quad \kappa_i \leftarrow \kappa_i - \text{Unif}(0,1) \cdot \hat{y}_{\xi,\text{rank}_{y_i}}^{\text{sorted}}$
9: $\quad$ **end if**
10: $\quad \mathcal{K} \leftarrow \mathcal{K} \ \cup \ \{\kappa_i\}$
11: **end for**
12: $\tau_\alpha \leftarrow$ the $\lceil (1-\alpha)(1+|\mathcal{D}_{\text{cal}}|) \rceil$-th smallest value in $\mathcal{K}$
13: **return** $\tau_\alpha$

a calibration stage, which determines a threshold from a held-out calibration dataset, and an inference stage, which uses the calibrated threshold to construct prediction sets with a guaranteed coverage that the true label lies within the set.

Starting with the calibration stage, it is assumed that a network $\mathcal{F}_\theta$ has been trained prior on a training dataset $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{|D_{train}|}$ where $x_i \in X$ and $y_i \in Y$, and that a separate hold-out calibration dataset $\mathcal{D}_{\text{cal}} = \{(x_i, y_i)\}_{i=1}^{|D_{cal}|}$ is available. Both datasets satisfy $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{cal}} \subset X \times Y$ and are from the same underlying data distribution. The calibration step, detailed in Algorithm 2, starts by iterating over each data point in the calibration dataset and retrieving the mean prediction $\hat{y}$ and estimated variance $\hat{\sigma}^2$ using the Adaptive MC Dropout method (line 3). Both quantities are vectors, computed elementwise across stochastic forward passes: for each class, the mean and sample variance are calculated from that class's predicted probabilities across the passes. All CP methods (both full and split) use a nonconformity score to quantify how atypical/nonconforming a prediction is relative to the model's confidence. By sorting the predictive probabilities by descending probability for each class $\xi \in \Xi$ and identifying the index for the true label $\text{rank}_{y_i}$ from these sorted probabilities, it can be measured how confidently the model predicts the true label. In the case of MC-CP, $\kappa_i$ (line 5) quantifies how much cumulative probability mass must be gathered before the true label appears in the sorted predictions, effectively capturing the model's uncertainty for that example. The regularisation term $\beta(\text{rank}_{y_i} - N_{\text{reg}})_+$ penalises instances where the true label ranks low among the predicted classes, effectively discouraging large prediction sets by assigning higher nonconformity to uncertain examples. As a result, this score encodes both model confidence and a regularisation bias toward compact, informative prediction sets used during calibration. If chosen by the user, via the rand boolean, a small random perturbation is applied by subtracting a uniform fraction

**Algorithm 3** MC-CP inference for classification

**Require:** Model $\mathcal{F}_\theta$, Input $x$, Maximum forward passes $N_{\text{mc}}$, Threshold $\delta$, Patience $\rho_{\max}$, Regularisation coefficient $\beta$, Number of unpenalised classes $N_{\text{reg}}$, Stochastic smoothing boolean rand, and the empirical quantile $\tau_\alpha$

1: $\hat{y}, \hat{\sigma}^2 \leftarrow \text{AdaptiveMCDropout}(\mathcal{F}_\theta, x, N_{\text{mc}}, \delta, \rho_{\max})$
2: $\hat{y}^{\text{sorted}}, \hat{\sigma}^{2,\text{sorted}} \leftarrow [(\hat{y}_1, \hat{\sigma}_1^2), \ldots, (\hat{y}_\Xi, \hat{\sigma}_\Xi^2)] \quad \text{s.t.} \quad \hat{y}_1^{\text{sorted}} \geqslant \ldots \geqslant \hat{y}_\Xi^{\text{sorted}}$
3: $\text{rank} \leftarrow |\{\xi \in \Xi : \sum_{i=1}^{\xi} \hat{y}_i^{\text{sorted}} + \beta(\xi - N_{\text{reg}})_+ \leqslant \tau_\alpha\}| + 1$
4: **if** rand **then**
5: $\quad \text{rank} \leftarrow \text{rank} - \mathbb{I}\left\{ \frac{(\sum_{i=1}^{\text{rank}} \hat{y}_i^{\text{sorted}} + \beta(\text{rank} - N_{\text{reg}})_+ - \tau_\alpha)}{\hat{y}_{\text{rank}^{\text{sorted}}} + \beta \mathbb{I}(\text{rank} > N_{\text{reg}})} \leqslant \text{Unif}(0,1) \right\}$
6: **end if**
7: $\Gamma \leftarrow \{\xi \in \Xi | \xi \leqslant \text{rank}\}$
8: $\Gamma_{\text{probs}} \leftarrow (\hat{y}_i^{\text{sorted}})_{i=1}^{\text{rank}}$
9: $\Gamma_{\text{vars}} \leftarrow (\hat{\sigma}_i^{2,\text{sorted}})_{i=1}^{\text{rank}}$
10: **return** $(\Gamma,\ \Gamma_{\text{probs}},\ \Gamma_{\text{vars}})$

of the true label's probability mass. This randomisation serves as a tie-breaking mechanism that smooths the nonconformity distribution and ensures exact finite-sample coverage. Finally, $\tau_\alpha$ is obtained by sorting the calibration scores in $\mathcal{K}$ in ascending order and selecting the score at rank $\lceil (1-\alpha)(1+|\mathcal{D}_{\text{cal}}|) \rceil$. This estimates the $(1-\alpha)$-quantile with a finite-sample correction: the threshold is chosen so that most calibration examples have true-label scores at or below it, and is then used to construct prediction sets for unseen inputs. For example, with 99 calibration examples and $\alpha = 0.1$, the threshold is the 90th smallest score.

After the threshold $\tau_\alpha$ has been computed, it can be used during inference on unseen data points to construct compact prediction sets. As outlined in Algorithm 3, for an unseen input $x$, the predictive mean $\hat{y}$ and associated estimated variance $\hat{\sigma}^2$ are obtained using the Adaptive MC Dropout method, consistent with the procedure applied during calibration. Subsequently, the predictive probabilities and corresponding variance estimates for all classes are sorted in descending order according to the predicted probabilities. To determine how many classes should be included in the prediction set for a new sample, the calibrated threshold $\tau_\alpha$ is used to identify the smallest number of top-ranked classes whose cumulative probability mass, adjusted by the same regularisation term from the calibration stage, does not exceed $\tau_\alpha$ (line 3). This ensures that the resulting prediction set satisfies the desired coverage level established during calibration. If chosen by the user, via the rand boolean, a randomisation step analogous to that used during calibration is applied to handle boundary cases where the cumulative conformity score is close to the threshold. Specifically, a uniform random variable determines whether the last included class, the one that causes the cumulative score to exceed $\tau_\alpha$, is retained or excluded from the prediction set. Finally, the prediction set $\Gamma$ is constructed by including all classes up to the computed rank (line 7). The corresponding predicted probabilities and variances for these classes are stored as $\Gamma_{\text{probs}}$ and $\Gamma_{\text{vars}}$ respectively.

The user-defined hyperparameters introduced by MC-CP, such as the regularisation coefficient $\beta$, number of unpenalised classes $N_{\text{reg}}$, stochastic smoothing boolean rand, and miscoverage rate

$\alpha$, control the prediction set size sensitivity and the regularisation of the method. The configuration of these hyperparameters closely follows the conventions established in the Regularised Adaptive Prediction Sets (RAPS) [111] framework and the standard split CP framework [107]. Specifically, $\beta$ determines the magnitude of the additive penalty applied to lower-ranked probabilities, thereby trading off compactness and coverage (visualisation of this additive penalty can be seen in Figure 3.4). $N_{\text{reg}}$ defines how many of the top-ranked classes are exempt from this penalty. rand enables stochastic tie-breaking among equally ranked logits to maintain exchangeability, and $\alpha$ specifies the target miscoverage rate, which directly controls the coverage level $1-\alpha$. MC-CP introduces no additional calibration-specific hyperparameters beyond those inherited from RAPS; its only additional parameters, $\delta$ and $\rho_{\max}$, belong to the Adaptive MC Dropout component discussed in Section 3.2.1. It should be noted that the introduction of Adaptive MC Dropout does not alter the functional behaviour of the RAPS hyperparameters. Because conformal calibration depends only on the relative ordering of non-conformity scores within the calibration set, the parameters $\beta$, $N_{\text{reg}}$, and $\alpha$ retain the same effect on coverage and regularisation. The Adaptive MC Dropout component merely stabilises the underlying probability estimates by reducing stochastic variance, which may influence the magnitude of the resulting prediction set sizes but not the qualitative behaviour or tuning dynamics of these hyperparameters.

MC-CP directly inherits the finite-sample coverage guarantees of split conformal prediction, as formalised in RAPS [111]. Under the standard assumption of exchangeability between the calibration and test samples, CP ensures that for any target miscoverage rate $\alpha$, the constructed prediction set $\Gamma$ satisfies:

$$\mathbb{P}(y_i \in \Gamma | x_i) \geqslant 1-\alpha \tag{3.10}$$

This guarantee is model-agnostic and holds irrespective of the stochasticity introduced by Adaptive MC Dropout, since the dropout realisations are independent across samples and identically applied to both calibration and test data. The Adaptive MC Dropout component influences only the efficiency and stability of the underlying predictive distributions, affecting the expected size of the prediction sets, but does not alter the statistical validity of the conformal calibration.

### 3.2.3 MC-CP for Regression

The MC-CP framework is generalisable across tasks, and we extend it to regression tasks where the objective is to construct prediction intervals over continuous targets rather than discrete label sets over categorical targets. MC-CP for regression operates in the same two stages as the classification variant: calibration and inference.

In the regression setting, the neural network $\mathcal{F}_\theta$ is a deep quantile regression (DQR) network [228], which learns to approximate conditional quantiles $\tau_{\text{upper}}, \tau_{\text{lower}} \in (0,1)$ of the target distribution given an input $x$, instead of a single point estimate as in standard regression. Although MC-CP can be paired with a conventional regression model, the resulting intervals are typically less well-calibrated. By contrast, DQR provides a more expressive approximation of

**Algorithm 4** MC-CP calibration for regression

**Require:** Model $\mathcal{F}_\theta$, Calibration dataset $\mathcal{D}_{\text{cal}}$, Maximum forward passes $N_{\text{mc}}$, Threshold $\delta$, Patience $\rho_{\max}$, and Miscoverage rate $\alpha$

1: $\mathcal{K} \leftarrow 0$;
2: **for** $(x_i, y_i) \in \mathcal{D}_{\text{cal}}$ **do**
3: $\quad(\hat{y}_{\text{low}}, \hat{y}_{\text{high}}), (\hat{\sigma}^2_{\text{low}}, \hat{\sigma}^2_{\text{high}}) \leftarrow \text{AdaptiveMCDropout}(\mathcal{F}_\theta, x_i, N_{\text{mc}}, \delta, \rho_{\max})$
4: $\quad\kappa_i \leftarrow \max\{\hat{y}_{\text{low}} - y_i, y_i - \hat{y}_{\text{high}}\}$
5: $\quad\mathcal{K} \leftarrow \mathcal{K} \cup \{\kappa_i\}$
6: **end for**
7: $\tau_\alpha \leftarrow$ the $\lceil (1-\alpha)(1+|\mathcal{D}_{\text{cal}}|)\rceil$-th smallest value in $\mathcal{K}$
8: **return** $\tau_\alpha$

the conditional distribution, yielding adaptive, heteroscedastic intervals that better align with the conformal calibration objective.

The calibration stage, detailed in Algorithm 4, closely parallels that of the classification variant, but is adapted to handle continuous-valued targets. Iterating over each data point in the calibration dataset, the mean prediction $(\hat{y}_{\text{low}}, \hat{y}_{\text{high}})$ and estimated variance $(\hat{\sigma}^2_{\text{low}}, \hat{\sigma}^2_{\text{high}})$ for both the predicted upper and lower quantiles of the DQR network is retrieved using the Adaptive MC Dropout method. The nonconformity score $\kappa_i$ is calculated (line 3), where it measures how small the adjustment of the quantiles should be to expand the predicted interval so it would include $y_i$. If the ground truth value $y_i$ lies within the predicted interval, both terms are non-positive, and therefore $\kappa_i \leqslant 0$. If the ground truth value lies below the lower bound, then $\hat{y}_{\text{low}} - y_i > 0$ captures how far below the quantile the true value is. Inversely, if above $y_i$ is above the upper bound, then $y_i - \hat{y}_{\text{high}} > 0$ captures how far above the quantile the true value is. Finally, $\tau_\alpha$ is computed from the calibration scores in $\mathcal{K}$ using the same finite-sample-adjusted quantile selection as in Algorithm 2.

The threshold $\tau_\alpha$ can be used during inference on unseen data to calibrate the predicted quantiles to guarantee coverage. Detailed in Algorithm 5, the mean prediction $(\hat{y}_{\text{low}}, \hat{y}_{\text{high}})$ and estimated variance $(\hat{\sigma}^2_{\text{low}}, \hat{\sigma}^2_{\text{high}})$ for both the predicted upper and lower quantiles of the DQR network is retrieved using the Adaptive MC Dropout method just as in the calibration stage. To compute the prediction interval $\Gamma$, the threshold is applied symmetrically to the predicted quantiles, expanding the interval to ensure the desired coverage level $1-\alpha$, following the principle inherited from Conformalised Quantile Regression (CQR) [112] framework. The variance of this interval $\Gamma_{\text{vars}}$ captures epistemic uncertainty in both quantile estimates.

The user-defined miscoverage rate $\alpha$ plays the same role as in the classification variant, determining the desired coverage level $1 - \alpha$ of the conformal intervals. However, its effect on the regression formulation is more directly observable in the interval width. As $\alpha \to 0$, it results in a larger threshold $\tau_\alpha$, thereby expanding the prediction interval. Conversely, as $\alpha \to 1$, it results in a smaller threshold $\tau_\alpha$, reducing the expansion of the prediction interval.

**Algorithm 5** MC-CP inference for regression

**Require:** Model $\mathcal{F}_\theta$, Input $x$, Maximum forward passes $N_{\text{mc}}$, Threshold $\delta$, Patience $\rho_{\max}$, and the empirical quantile $\tau_\alpha$

1: $(\hat{y}_{\text{low}}, \hat{y}_{\text{high}}), (\hat{\sigma}^2_{\text{low}}, \hat{\sigma}^2_{\text{high}}) \leftarrow \text{AdaptiveMCDropout}(\mathcal{F}_\theta, x, N_{\text{mc}}, \delta, \rho_{\max})$
2: $\Gamma \leftarrow [\hat{y}_{\text{low}} - \tau_\alpha, \hat{y}_{\text{high}} + \tau_\alpha]$
3: $\Gamma_{\text{vars}} \leftarrow \hat{\sigma}^2_{\text{high}} + \hat{\sigma}^2_{\text{low}}$
4: **return** $(\Gamma, \Gamma_{\text{vars}})$

## 3.3 Evaluation

This section empirically evaluates the proposed MC-CP framework, including the Adaptive MC Dropout component, across both classification and regression tasks. The experimental evaluation aims to verify that MC-CP achieves the dual objectives established throughout Section 3.2: maintaining statistically valid coverage while producing compact prediction sets that remain robust under distributional uncertainty and perceptual noise. To this end, the following hypotheses are investigated.

The first hypothesis is that the Adaptive MC Dropout method improves inference efficiency by terminating sampling early for confident inputs based on online predictive variance monitoring, thereby reducing the number of stochastic forward passes required without significantly affecting predictive accuracy.

The second hypothesis is that MC-CP retains the $1 - \alpha$ coverage guarantee of split CP while producing smaller and more informative prediction sets than state-of-the-art methods. By regularising both the head and tail of the prediction set, the resulting sets should remain compact yet statistically valid.

The final hypothesis is that MC-CP maintains valid coverage and reasonable prediction-set size under imperfect perception and distributional shift. By integrating uncertainty quantification directly into the calibration process, the method is hypothesised to remain robust when the underlying perception model operates on noisy or shifted inputs.

The subsequent sections evaluate these hypotheses systematically. Section 3.3.1 presents the core quantitative results for the Adaptive MC Dropout method, as well as classification and regression results for MC-CP. Section 3.3.2 analyses the sensitivity to user-defined hyperparameters, followed by coverage behaviour under distributional shift and input corruption in Section 3.3.3. We then analyse the conditions in which MC-CP can fail in practical settings in Section 3.3.4, before examining potential ways to mitigate these conditions in Section 3.3.5. Section 3.3.6 investigates ways to incorporate risk into the prediction set creation, then finally, Section 3.3.7 evaluates a kernel-density-based Adaptive MC Dropout variant to improve saliency of the stopping mechanism.

### 3.3.1 Core Results

This section presents the core experimental results used for validating the proposed MC-CP framework. We begin by evaluating the behaviour of the Adaptive MC Dropout method in isolation, before examining the performance of MC-CP in classification and regression tasks.

#### 3.3.1.1 Adaptive MC Dropout Analysis

We first evaluate the behaviour of the proposed Adaptive MC Dropout method in isolation to verify its efficiency and efficacy in practice. This analysis aims to empirically test our first hypothesis: that the online predictive variance monitoring preserves the predictive fidelity of standard MC Dropout while reducing the number of required forward passes.

To assess this, we run a variety of experiments on a handful of image classification datasets: MNIST [229], Fashion-MNIST [230], CIFAR-10 [231], and CIFAR-100 [231]. Full details of the experimental setup and dataset information can be found in Appendix A.5.1. All methods employ identical network architectures and training procedures for a fair comparison. In-distribution (ID) inputs are held-out samples from the same distribution as the training data, whereas out-of-distribution (OOD) inputs come from a different distribution.

Firstly, to evaluate whether Adaptive MC Dropout achieves comparable predictive performance and uncertainty estimates while improving computational efficiency, we compare it against standard MC Dropout [66] and a single-shot variant [232] that leverages moment propagation [233] to approximate the predictive mean and variance. Performance is assessed across a range of UQ metrics, including accuracy, AUROC, Spearman's $\rho$, and negative log-likelihood (NLL), under both ID and OOD settings, as well as in terms of computational efficiency. These metrics were chosen to capture complementary aspects of predictive quality and efficiency: accuracy measures overall task performance, ID AUROC and Spearman's $\rho$ quantify the discriminative power and ranking consistency of uncertainty estimates, NLL evaluates probabilistic calibration, and OOD AUROC evaluates the separation between ID and OOD samples. Concretely, ID AUROC is computed by ranking ID examples by uncertainty and measuring how well this separates incorrect from correct predictions, while Spearman's $\rho$ is the rank correlation between uncertainty and error magnitude (e.g., $|y - \hat{y}|$). The computational metrics, average inference time per sample (ms) and the number of forward passes, assess the efficiency gains achieved by the adaptive mechanism. The results are summarised in Table 3.1.

Across both dataset pairings (MNIST → FMNIST, and CIFAR-10 → CIFAR-100), we observe that Adaptive MC Dropout achieves near-identical predictive accuracy and uncertainty estimates to traditional MC Dropout, despite performing substantially fewer stochastic forward passes. For example, on the CIFAR-10 → CIFAR-100 pairing, traditional MC Dropout achieves $72.61 \pm 0.76$ ID AUROC, showing good ability to distinguish between correct and incorrect predictions based on its uncertainty. Adaptive MC Dropout achieves a comparable $72.11 \pm 1.29$ ID AUROC, while in contrast, Single-Shot MC Dropout only reaches $64.15 \pm 1.26$. This indicates two things: the adaptive termination of the sampling process does not degrade the quality of epistemic uncertainty estimation, and that the deterministic moment propagation of the single-shot variant

Table 3.1: Evaluation of predictive uncertainty quality and efficiency across MC-Dropout variants on in-distribution (ID) and out-of-distribution (OOD) data. Light blue highlighted cells indicate the best result.

| | ID (MNIST) | | | | OOD (FMNIST) | | Mixed | |
|---|---|---|---|---|---|---|---|---|
| **Technique** | **Accuracy ↑** | **AUROC ↑** | **Spearman's $\rho$ ↑** | **NLL ↓** | **AUROC ↑** | **Spearman's $\rho$ ↑** | **Time/sample (ms) ↓** | **Forward Passes ↓** |
| MC-Dropout | $98.96 \pm 0.06$ | $\mathbf{0.98 \pm 0.01}$ | $\mathbf{0.17 \pm 0.01}$ | $\mathbf{0.03 \pm 0.01}$ | $\mathbf{0.99 \pm 0.01}$ | $\mathbf{0.86 \pm 0.00}$ | $8.05 \pm 0.01$ | $1000.00 \pm 0.00$ |
| Single-Shot MC-Dropout | $98.68 \pm 0.09$ | $0.93 \pm 0.01$ | $\mathbf{0.17 \pm 0.02}$ | $0.48 \pm 0.03$ | $\mathbf{0.99 \pm 0.02}$ | $\mathbf{0.86 \pm 0.03}$ | $\mathbf{1.48 \pm 0.02}$ | $\mathbf{1.00 \pm 0.00}$ |
| Adaptive MC-Dropout | $\mathbf{98.97 \pm 0.06}$ | $\mathbf{0.98 \pm 0.00}$ | $\mathbf{0.17 \pm 0.01}$ | $\mathbf{0.03 \pm 0.00}$ | $\mathbf{0.99 \pm 0.01}$ | $\mathbf{0.86 \pm 0.01}$ | $1.58 \pm 0.04$ | $76.60 \pm 3.93$ |

| | ID (CIFAR-10) | | | | OOD (CIFAR-100) | | Mixed | |
|---|---|---|---|---|---|---|---|---|
| **Technique** | **Accuracy ↑** | **AUROC ↑** | **Spearman's $\rho$ ↑** | **NLL ↓** | **AUROC ↑** | **Spearman's $\rho$ ↑** | **Time/sample (ms) ↓** | **Forward Passes ↓** |
| MC-Dropout | $\mathbf{61.00 \pm 1.09}$ | $\mathbf{72.61 \pm 0.76}$ | $\mathbf{0.38 \pm 0.01}$ | $\mathbf{1.12 \pm 0.04}$ | $\mathbf{64.73 \pm 1.34}$ | $\mathbf{0.25 \pm 0.02}$ | $9.26 \pm 0.22$ | $1000.00 \pm 0.00$ |
| Single-Shot MC-Dropout | $46.46 \pm 4.51$ | $64.15 \pm 1.26$ | $0.24 \pm 0.02$ | $1.60 \pm 0.06$ | $55.22 \pm 2.13$ | $0.09 \pm 0.04$ | $2.25 \pm 0.21$ | $\mathbf{1.00 \pm 0.00}$ |
| Adaptive MC-Dropout | $60.60 \pm 1.81$ | $72.11 \pm 1.29$ | $0.37 \pm 0.02$ | $1.13 \pm 0.05$ | $64.05 \pm 1.92$ | $0.24 \pm 0.03$ | $\mathbf{2.12 \pm 0.00}$ | $105.93 \pm 2.47$ |

fails to capture the full variability of the posterior distribution. This gap in performance is less pronounced in easy dataset pairings, such as MNIST → FMNIST, but exists nonetheless.

This trend continues to exist on the OOD metrics, where Adaptive MC Dropout maintains parity with standard MC Dropout in both AUROC and Spearman's $\rho$, confirming that adaptivity preserves the model's ability to discriminate between in- and out-of-distribution samples. In contrast, the single-shot variant again exhibits a drop in uncertainty performance. For example, in OOD Spearman's $\rho$, which quantifies how well uncertainty correlates with prediction correctness, both traditional MC Dropout and Adaptive MC Dropout reach $0.24 \pm 0.02$ and $0.24 \pm 0.03$ respectively on the CIFAR-10 → CIFAR-100 pairing, showing moderate positive correlation. Single-Shot MC Dropout only achieves $0.09 \pm 0.04$, showing very weak positive correlation.

The computational benefits are Adaptive MC Dropout are significant. On average, it performs 10x fewer forward passes than traditional MC Dropout, translating to a 4-5x reduction in per-sample inference time depending on the dataset. It achieves per-sample inference time on par with the single-shot variant but with the performance of the heavier traditional method.

Taken together, these results empirically aid us in verifying the first hypothesis, showing that Adaptive MC Dropout preserves the predictive fidelity of standard MC Dropout while substantially reducing computational cost. The negligible effect of additional forward passes after convergence confirms that the adaptive rule provides a principled trade-off between efficiency and statistical reliability.

The variance convergence tracking in Adaptive MC Dropout can be visualised in Figure 3.3 for ID and OOD data. We can observe, in the ID data sample, that the variance for all classes rapidly decreases during the initial stochastic forward passes. After $\approx 100$ forward passes, the variance begins to stabilise, with the occasional forward pass contributing small amounts of additional information and thereby increasing variance. However, at $\approx 242$ forward passes, the variance decreases below the threshold $\delta$ and after $\rho_{\max}$ passes later, the Adaptive MC Dropout method terminates early. A similar trend is observed under OOD conditions, although convergence is slightly slower due to the input data being OOD, and thus, stochastic forward passes are highly stochastic. We see this as variance spikes are higher and more frequent, especially around the $200 - 250$ forward passes mark. The OOD data sample converges much later, after around 362

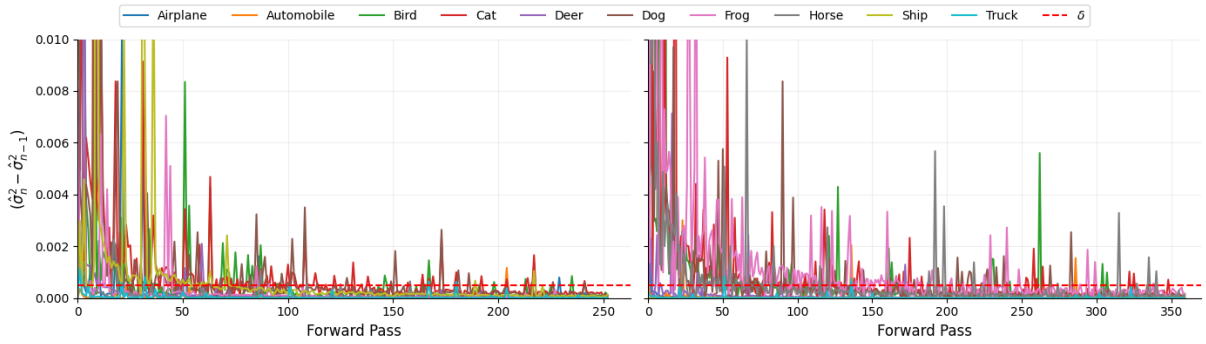



Figure 3.3: Variance convergence behaviour of Adaptive MC Dropout on CIFAR-10 (left, ID) and CIFAR-100 (right, OOD). Each curve represents the per-class variance difference $\hat{\sigma}_n^2 - \hat{\sigma}_{n-1}^2$ across stochastic forward passes, with the red dashed line indicating the convergence threshold $\delta$ used for early termination.

forward passes. These results qualitatively confirm the theoretical findings in Theorem 3.1, and the empirical results in Table 3.1.

#### 3.3.1.2 Classification Results

We next evaluate the proposed MC-CP framework in the classification setting to empirically test our second hypothesis: the MC-CP achieves $1-\alpha$ coverage guarantee while producing more compact prediction sets.

We conduct a variety of experiments on an expanded suite of image classification datasets used previously, namely: MNIST [229], Fashion-MNIST [230], CIFAR-10 [231], CIFAR-100 [231], and Tiny-ImageNet [234]. MC-CP is evaluated against a handful of state-of-the-art UQ and CP methods: a baseline neural network, MC-Dropout [66], Naive CP [111], and Regularised Adaptive Prediction Sets (RAPS) [111]. Full details of the experimental setup and dataset information can be found in Appendix A.5. All methods employ identical network architectures as well as calibration and training procedures for a fair comparison.

Firstly, we evaluate the base predictive performance of MC-CP and the comparative methods by examining the average prediction-set accuracy and average set size. An effective split conformal method should maximise the likelihood that the ground-truth label is contained within the prediction set while minimising the number of labels included. A naive approach would trivially achieve full coverage by including all classes, but at the cost of excessively large prediction sets. Conversely, restricting the set to only the top-1 prediction would reduce the set size but severely degrade coverage. Table 3.2 summarises these trade-offs across datasets of increasing difficulty, where $\alpha = 0.05$.

Across all datasets, the baseline model provides a reference for the accuracy achievable by the underlying CNN when trained in the standard manner and producing singleton predictions. On simpler datasets, the model attains competitive performance (e.g., $98.90 \pm 0.06$ on MNIST), but its efficacy declines substantially on more complex datasets (e.g., $18.93 \pm 1.05$ on Tiny-ImageNet). When the same network is evaluated with MC Dropout at inference time,

Table 3.2: Average prediction set accuracy and size for all comparative methods (a baseline neural network, MC-Dropout, Naive CP, and RAPS) across datasets of increasing difficulty.

| | MNIST | | Fashion-MNIST | | CIFAR-10 | | CIFAR-100 | | Tiny-ImageNet | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Technique** | **Accuracy ↑** | **Set Size ↓** | **Accuracy ↑** | **Set Size ↓** | **Accuracy ↑** | **Set Size ↓** | **Accuracy ↑** | **Set Size ↓** | **Accuracy ↑** | **Set Size ↓** |
| Baseline | $98.90 \pm 0.06$ | $1.00 \pm 0.00$ | $87.99 \pm 0.26$ | $1.00 \pm 0.00$ | $61.95 \pm 0.36$ | $1.00 \pm 0.00$ | $27.86 \pm 0.87$ | $1.00 \pm 0.00$ | $18.93 \pm 1.05$ | $1.00 \pm 0.00$ |
| MC | $98.89 \pm 0.04$ | $1.00 \pm 0.00$ | $87.92 \pm 0.23$ | $1.00 \pm 0.00$ | $64.46 \pm 0.36$ | $1.00 \pm 0.00$ | $30.54 \pm 0.60$ | $1.00 \pm 0.00$ | $21.40 \pm 2.37$ | $1.00 \pm 0.00$ |
| Naive | $94.99 \pm 0.58$ | $0.95 \pm 0.22$ | $95.07 \pm 0.40$ | $1.20 \pm 0.42$ | $94.93 \pm 0.28$ | $3.53 \pm 1.63$ | $95.08 \pm 0.42$ | $39.25 \pm 11.43$ | $95.15 \pm 0.34$ | $97.53 \pm 29.64$ |
| RAPS | $98.90 \pm 0.04$ | $1.08 \pm 0.35$ | $98.88 \pm 0.07$ | $1.80 \pm 0.99$ | $96.71 \pm 0.30$ | $4.35 \pm 1.86$ | $95.17 \pm 0.25$ | $41.65 \pm 4.01$ | $95.43 \pm 0.09$ | $107.78 \pm 2.06$ |
| MC-CP | $99.68 \pm 0.01$ | $1.06 \pm 0.33$ | $99.18 \pm 0.06$ | $1.76 \pm 1.00$ | $98.53 \pm 0.15$ | $4.11 \pm 1.81$ | $96.46 \pm 0.20$ | $39.26 \pm 3.95$ | $96.01 \pm 0.41$ | $97.17 \pm 3.67$ |

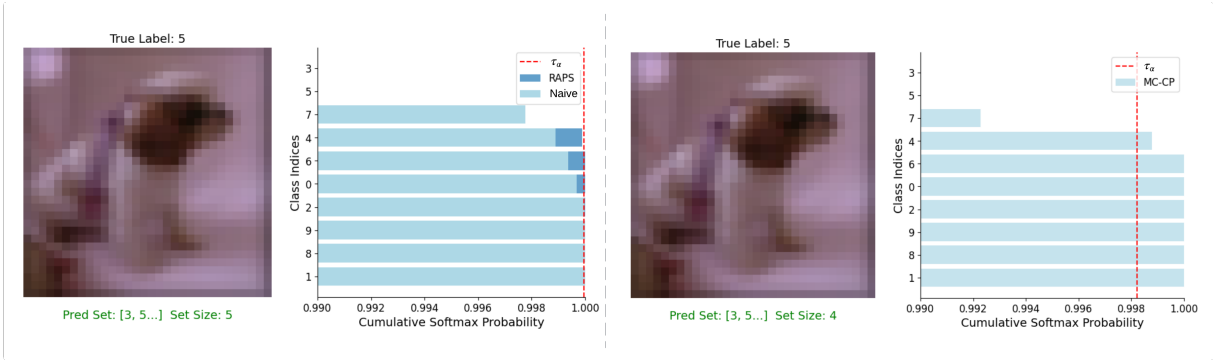



Figure 3.4: The construction of prediction sets for an example image from the CIFAR-10 dataset. The ground truth label and constructed prediction set are above and below the image, respectively. The cumulative softmax probabilities and the calibrated $\tau_\alpha$ visualises how classes are added to the prediction set. (Left) Naive CP, and Regularised Adaptive Prediction Sets (RAPS). (Right) The proposed Monte Carlo Conformal Prediction (MC-CP).

accuracy improves slightly due to the regularising effect of averaging over multiple stochastic predictions—for instance, a $2.47 \pm 1.32$ increase on Tiny-ImageNet compared to the baseline. However, this improvement is insufficient to overcome the limitations of producing only singleton predictions. Although the resulting predictive variance offers some indication of the model's uncertainty, its absolute magnitude (e.g., 0.12 on the top-1 prediction for a given sample) is difficult to interpret meaningfully. The foundational CP method, Naive CP, which ranks class probabilities and selects the minimum threshold required to satisfy the target coverage, approximately achieves the desired coverage across all datasets but falls slightly short in some cases; for example, $94.93 \pm 0.28$ on CIFAR-10. While its prediction set sizes accurately reflect uncertainty across datasets, performance degrades as task complexity increases, and the method struggles to produce compact and stable sets; for example, $97.53 \pm 29.64$ on Tiny-ImageNet. RAPS introduces regularisation to the tail of the prediction distribution, which stabilises the resulting prediction sets but comes at the cost of slightly larger set sizes, for instance, a 10.25 increase in the mean set size, while substantially improving stability, as reflected by the reduction in standard deviation from 29.64 to 2.06. RAPS also manages to achieve over the target miscoverage rate across all datasets evaluated. Finally, MC-CP achieves the best overall performance, attaining the highest accuracy and coverage across all datasets while producing small and stable prediction sets. For example, on Tiny-ImageNet, MC-CP attains a compact mean set size comparable to Naive CP while matching the stability of RAPS, achieving $97.17 \pm 3.67$.

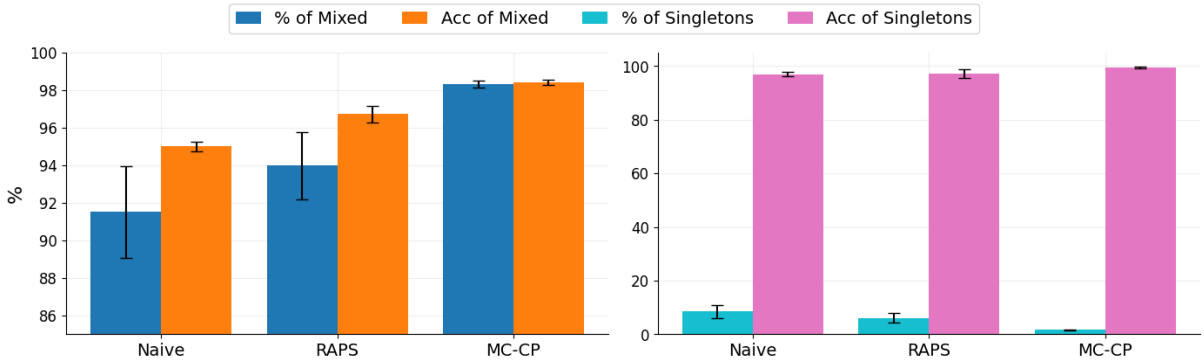



Figure 3.5: Percentage and accuracy of prediction sets across methods for singleton (right) and mixed (left) cases. Singleton sets correspond to highly confident predictions where the set size equals one, while mixed sets represent uncertain samples with multiple possible labels.

Alongside quantitative results, Figure 3.4 provides a qualitative illustration of how the comparative CP methods construct their prediction sets. All methods sort class probabilities in descending order and sequentially add classes until the calibrated cumulative threshold is exceeded. On the left, Naive CP (light blue bars) simply adds classes until this threshold is met, yielding a set of size five in this example. RAPS applies a small regularisation penalty (represented by dark blue bars on top of the base probabilities in light blue) after a fixed number of top classes $N_{\text{reg}}$, which slightly smooths the inclusion process and stabilises the set, though its size remains the same. In contrast, MC-CP, on the right, integrates predictive uncertainty into the calibration stage, resulting in a lower threshold $\tau_\alpha$. Combined with the regularisation of both the head and tail of the predictive distribution, spurious low-probability classes are suppressed, producing a more compact and calibrated prediction set, a size of four in this example.

Figure 3.5 further analyses the prediction-set creation by separating cases where the CP method outputs a singleton (high-confidence, set size = 1) or a mixed set (low-confidence, set size > 1). The proportion and corresponding accuracy of each case reveal how effectively each method calibrates uncertainty. As shown, Naive CP produces a relatively smaller proportion of mixed sets, suggesting over-confidence, and its accuracy on these uncertain samples is noticeably lower. RAPS increases the proportion of mixed sets, indicating improved uncertainty awareness, but their accuracy remains variable due to less precise calibration of the predictive tail. In contrast, MC-CP achieves both a higher proportion and higher accuracy of mixed sets, implying that it correctly identifies and defers uncertain predictions. At the same time, it maintains near-perfect accuracy on singleton sets, confirming that confident predictions remain reliable and are used in cases where the model will most likely be accurate. The lower accuracy and proportion of singletons observed in both Naive CP and RAPS suggest that, in cases where a singleton prediction was incorrect, increasing the set size could have yielded a more accurate and calibrated prediction.

Figure 3.6 illustrates the average softmax probability of the top-1 class across datasets for the CP methods, highlighting the effect of Adaptive MC Dropout on confidence regularisation. As expected, overall confidence decreases as task complexity grows, reflecting greater epistemic

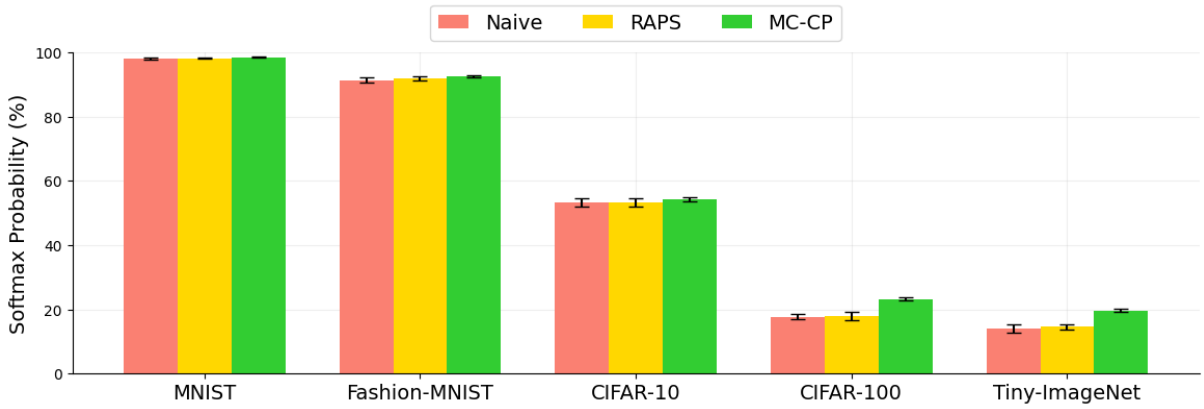



Figure 3.6: Average softmax probability of the top-1 class within the constructed prediction sets across datasets of increasing complexity. Higher values indicate greater model confidence in the most probable class.

Table 3.3: Average prediction set accuracy and size for all comparative CP methods on Tiny-ImageNet using VGG16 and VGG19 architectures.

| | **VGG16** | | **VGG19** | |
|---|---|---|---|---|
| **Technique** | **Accuracy ↑** | **Set Size ↓** | **Accuracy ↑** | **Set Size ↓** |
| Naive | $94.84 \pm 0.28$ | $32.39 \pm 23.44$ | $95.16 \pm 0.34$ | $28.04 \pm 22.53$ |
| RAPS | $95.10 \pm 0.37$ | $42.18 \pm 5.17$ | $95.23 \pm 0.19$ | $36.50 \pm 4.90$ |
| MC-CP | $96.14 \pm 0.26$ | $40.44 \pm 5.09$ | $96.22 \pm 0.14$ | $32.48 \pm 5.22$ |

uncertainty. However, MC-CP consistently maintains slightly higher and more stable top-1 probabilities compared to Naive CP and RAPS, particularly on challenging datasets such as CIFAR-100 and Tiny-ImageNet. These results empirically support that Adaptive MC Dropout stabilises the head of the predictive distribution, enabling MC-CP to produce more calibrated and reliable predictions.

To evaluate how the CP methods perform when paired with a well-trained and fine-tuned backbone, we repeat the analysis on the Tiny-ImageNet dataset using VGG16 and VGG19 architectures [235], results shown in Table 3.3. Across both backbones, the relative ordering of methods mirrors the core results (shown in Table 3.2): MC-CP attains the highest accuracy, followed by RAPS, with Naive CP performing slightly worse. Moving from the lightly tuned CNNs to stronger VGG models compresses prediction sets dramatically; average set sizes fall from $\approx 100$ labels to around $30 - 40$, representing nearly a threefold reduction. This demonstrates that better-trained backbones reduce epistemic noise, and the conformal methods scale accordingly. RAPS continues to stabilise prediction sets over Naive CP but incurs a modest increase in set size, while MC-CP combines high accuracy with compactness.

To verify that the overall improvement in predictive accuracy observed with MC-CP is not dominated by a single class, we evaluate the average prediction-set accuracy per class on CIFAR-10, as shown in Table 3.4. The results clearly show that MC-CP achieves consistently high accuracy across all ten classes, with notably low variance, indicating that its performance gains

Table 3.4: Average prediction set accuracy per class for all comparative methods on the CIFAR-10 dataset.

| Technique | Airplane | Automobile | Bird | Cat | Deer | Dog | Frog | Horse | Ship | Truck |
|---|---|---|---|---|---|---|---|---|---|---|
| Baseline | $62.91 \pm 4.57$ | $76.28 \pm 6.94$ | $39.83 \pm 4.12$ | $40.79 \pm 2.86$ | $59.82 \pm 9.23$ | $49.46 \pm 6.18$ | $87.76 \pm 2.92$ | $61.24 \pm 5.27$ | $76.84 \pm 5.09$ | $66.73 \pm 3.41$ |
| MC | $65.93 \pm 5.31$ | $75.39 \pm 7.94$ | $45.32 \pm 5.07$ | $39.37 \pm 3.19$ | $57.28 \pm 3.34$ | $59.21 \pm 7.88$ | $76.43 \pm 6.87$ | $72.07 \pm 3.13$ | $77.08 \pm 6.41$ | $73.12 \pm 6.89$ |
| Naive | $93.47 \pm 1.02$ | $96.79 \pm 0.69$ | $92.38 \pm 2.19$ | $93.15 \pm 2.24$ | $97.63 \pm 0.79$ | $93.08 \pm 0.83$ | $99.28 \pm 0.41$ | $91.36 \pm 2.85$ | $96.57 \pm 0.82$ | $94.83 \pm 1.38$ |
| RAPS | $95.63 \pm 1.19$ | $97.72 \pm 0.68$ | $94.69 \pm 1.25$ | $97.83 \pm 0.48$ | $98.57 \pm 0.64$ | $97.18 \pm 0.93$ | $98.83 \pm 0.42$ | $95.48 \pm 0.87$ | $97.25 \pm 1.03$ | $95.86 \pm 1.72$ |
| MC-CP | $97.82 \pm 0.91$ | $99.27 \pm 0.39$ | $98.19 \pm 0.79$ | $98.33 \pm 0.51$ | $98.97 \pm 0.33$ | $98.18 \pm 0.29$ | $98.95 \pm 0.59$ | $98.56 \pm 0.37$ | $98.21 \pm 0.47$ | $98.48 \pm 0.31$ |

Table 3.5: Mean absolute error (MAE) and coverage for all comparative methods across datasets of increasing difficulty.

| | Boston Housing | | Concrete | | Abalone | | Protein | | Blog Feedback | |
|---|---|---|---|---|---|---|---|---|---|---|
| Technique | MAE ↓ | Coverage ↑ | MAE ↓ | Coverage ↑ | MAE ↓ | Coverage ↑ | MAE ↓ | Coverage ↑ | MAE ↓ | Coverage ↑ |
| Baseline | $0.30 \pm 0.02$ | $23.52 \pm 3.18$ | $0.37 \pm 0.01$ | $20.55 \pm 1.41$ | $0.62 \pm 0.04$ | $47.86 \pm 2.34$ | $1.35 \pm 0.01$ | $49.10 \pm 1.75$ | $2.12 \pm 0.08$ | $70.32 \pm 5.70$ |
| MC | $0.37 \pm 0.02$ | $72.83 \pm 2.75$ | $0.54 \pm 0.01$ | $71.54 \pm 5.51$ | $0.64 \pm 0.02$ | $85.96 \pm 1.82$ | $1.49 \pm 0.02$ | $81.87 \pm 0.21$ | $2.61 \pm 0.08$ | $86.09 \pm 5.80$ |
| CQR | $0.31 \pm 0.61$ | $95.97 \pm 5.10$ | $0.37 \pm 0.02$ | $90.34 \pm 3.69$ | $0.62 \pm 0.11$ | $92.94 \pm 2.36$ | $1.40 \pm 0.02$ | $94.79 \pm 0.01$ | $2.21 \pm 0.10$ | $90.73 \pm 0.34$ |
| MC-CP | $0.35 \pm 0.20$ | $98.46 \pm 4.83$ | $0.44 \pm 0.01$ | $93.36 \pm 2.49$ | $0.64 \pm 0.04$ | $95.98 \pm 3.07$ | $1.45 \pm 0.01$ | $96.06 \pm 0.73$ | $2.40 \pm 0.12$ | $95.73 \pm 0.34$ |

are not confined to specific categories. In contrast, Naive CP and RAPS exhibit larger inter-class fluctuations, particularly on difficult classes such as *Bird* and *Cat*. For example, Naive CP exhibits per-class accuracies between $\approx 91 - 99\%$, which shows some accuracy gains are very class-specific. Inversely, MC-CP exhibits per-class accuracies between $\approx 97 - 99\%$, which shows gains are more uniform.

These results on the classification setting aid us in empirically verifying the second hypothesis: that MC-CP maintains the target miscoverage rate and produces smaller prediction sets than the state-of-the-art methods.

#### 3.3.1.3 Regression Results

Next, we evaluate the MC-CP framework in the regression setting to further validate its generalisability and efficacy.

We conduct our experiments on a suite of tabular regression datasets: Boston Housing [236], Concrete [237], Abalone [238], Protein [239], and Blog Feedback [240]. MC-CP is evaluated against a handful of state-of-the-art UQ and CP methods for regression: a baseline neural network (which is a deep quantile regression architecture), MC-Dropout [66], and Conformalised Quantile Regression (CQR) [112]. Full experimental details and dataset information can be found in Appendix A.5. All methods employ identical architectures as well as calibration and training procedures for a fair comparison.

First, we evaluate the predictive performance of MC-CP and the comparative methods by examining the mean absolute error of the predictive quantiles and the empirical coverage across the regression datasets. Table 3.5 summarises these results, where $\alpha = 0.1$.

As described in Section 3.2.3, we employ the DQR network architecture to best demonstrate the competitiveness of the CP methods. Across all datasets, the baseline network achieves a strong MAE, indicating that its predicted quantiles are close to the true targets. However, this comes at the cost of limited coverage, meaning that the ground truth frequently lies outside the

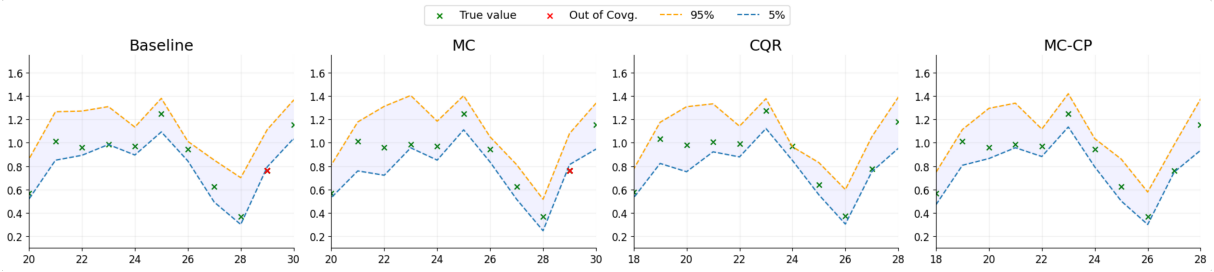



Figure 3.7: Predicted quantile intervals on the Boston Housing dataset for all comparative methods. The dashed lines denote the 5th and 95th predicted quantiles, with the shaded region representing the corresponding prediction interval. Green markers indicate true target values falling within the interval, while red markers denote values outside coverage.

predicted quantile intervals. For example, on the Boston Housing dataset, the baseline achieves the lowest MAE with $0.30 \pm 0.02$, but a coverage of $23.52 \pm 3.18$ is extremely low, indicating that the predicted quantile intervals are overly narrow and frequently fail to contain the true target values. Employing MC Dropout can marginally increase coverage compared to the baseline, for example, a $50.99 \pm 4.09$ increase in coverage on the Concrete dataset. Across all datasets, MC Dropout yields the worst MAE, indicating that its intervals are obtained by broadly widening predictions rather than improving fit. For example, on the Concrete dataset, MC Dropout achieves an MAE $\approx 22.73\%$ higher than the next highest. In practice, this degradation makes the method unreliable despite the higher nominal coverage. CQR is the state-of-the-art CP method for regression and manages to achieve the target miscoverage for every dataset while incurring only a marginal increase in MAE relative to the baseline. In all cases, the empirical coverage is $> 90\%$, demonstrating reliable calibration, and in some cases, the MAE remains comparable to the baseline (e.g., $0.37 \pm 0.01$ for the baseline and $0.37 \pm 0.02$ for CQR on the Concrete dataset), indicating that the improved coverage is achieved with minimal loss in predictive accuracy and through adaptive calibration of the quantiles rather than a uniform widening of prediction intervals. MC-CP achieves a substantial improvement in coverage across all datasets, often exceeding $> 95\%$, while incurring only a modest increase in MAE. Notably, this increase in error is smaller than that observed for the uncalibrated MC Dropout model, indicating that the adaptive calibration in MC-CP effectively compensates for the variance introduced by stochastic sampling.

To aid the quantitative results shown in Table 3.5, Figure 3.7 visualises the predicted quantile bands for a sequence of test points on the Boston Housing dataset. The baseline intervals are too narrow, yielding low coverage, as indicated by the out-of-coverage red crosses. MC Dropout increases coverage primarily by widening the predictive bands through stochastic sampling, which reduces misses but at the expense of inflated intervals. CQR calibrates the quantiles to the data, improving coverage with more structured and data-aligned widths. In contrast, MC-CP produces bands that are consistently well-scaled—most targets fall within the intervals while maintaining narrow widths.

Across both classification and regression tasks, MC-CP improves predictive reliability by enhanc-

ing coverage relative to its respective baselines. However, the manner in which this improvement presents differs interestingly between settings. In classification, MC-CP achieves higher coverage while simultaneously reducing prediction set sizes. As discussed in Section 3.3.1.2, this arises because the RAPS style regularisation regularises the tail of the probability distribution, suppressing overconfident low-probability classes and yielding more compact yet statistically valid prediction sets. The Adaptive MC Dropout component stabilises variance in the head of the distribution, ensuring that confident predictions remain sharp while uncertainty is well-calibrated in ambiguous regions.

In contrast, in regression tasks, MC-CP increases coverage at the cost of slightly wider predictive intervals. This difference stems from the nature of the predictive distributions in each setting. In classification, softmax outputs are bounded $[0, 1]$ and normalised, producing well-behaved probability distributions whose uncertainty estimates are relatively stable under Monte Carlo sampling. Regression outputs, on the other hand, are unbounded real values $(-\infty, \infty)$, meaning small perturbations in the dropout masks can induce large variations in the predicted mean and variance. Consequently, MC Dropout tends to exhibit greater instability in regression compared to classification, often producing noisier variance estimates. This was empirically shown via MC Dropout, producing quantiles with the largest MAE in Table 3.5.

Because MC-CP calibrates directly on these stochastic predictions, the resulting conformal intervals expand to accommodate this additional uncertainty, trading a modest increase in interval width for significantly higher coverage. This behaviour highlights a key distinction between classification and regression under MC-CP: in discrete prediction spaces, adaptive regularisation can compress prediction sets without sacrificing validity, whereas in continuous spaces it must instead compensate for epistemic instability by broadening quantile bounds. Together, these results underscore that MC-CP flexibly adapts to the structure of the predictive space, tightening or widening uncertainty regions as necessary to achieve the highest coverage results.

### 3.3.2 Parameter Analysis

We next perform a detailed parameter analysis to examine the sensitivity and robustness of the proposed MC-CP framework with respect to its key hyperparameters. This analysis aims to further analyse how the Adaptive MC Dropout and CP parameters interact and to quantify their influence on predictive calibration, coverage, and efficiency.

We share the same experimental setup and evaluation protocol described in Section 3.3.1.2 and Section 3.3.1.3, ensuring direct comparability with the classification experiments. Unless stated otherwise, all models are trained and evaluated on the CIFAR-10 dataset [231] for classification, and the Boston Housing dataset [236] for regression, using identical architectures and calibration procedures.

Firstly, we analyse the coverage and average prediction set size of all CP methods on the CIFAR-10 dataset across a range of target miscoverage rates $\alpha$. This evaluation examines whether the performance trends observed in the core results persist under varying miscoverage rates, thereby assessing the stability and reliability of each method. Table 3.6 summarises these results.

Table 3.6: Average prediction set accuracy and size for all comparative CP methods on the CIFAR-10 dataset under various target miscoverage rates $\alpha$.

| | $\alpha = 0.25$ | | $\alpha = 0.20$ | | $\alpha = 0.15$ | | $\alpha = 0.10$ | | $\alpha = 0.05$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Technique** | **Accuracy ↑** | **Set Size ↓** | **Accuracy ↑** | **Set Size ↓** | **Accuracy ↑** | **Set Size ↓** | **Accuracy ↑** | **Set Size ↓** | **Accuracy ↑** | **Set Size ↓** |
| Naive | $84.28 \pm 0.91$ | $2.17 \pm 1.30$ | $84.81 \pm 0.87$ | $2.20 \pm 1.49$ | $86.57 \pm 0.67$ | $2.83 \pm 1.58$ | $90.06 \pm 1.12$ | $3.34 \pm 1.79$ | $94.93 \pm 0.28$ | $3.53 \pm 1.63$ |
| RAPS | $89.99 \pm 0.13$ | $2.87 \pm 1.02$ | $92.00 \pm 0.62$ | $3.18 \pm 1.14$ | $94.21 \pm 0.29$ | $3.44 \pm 1.30$ | $95.80 \pm 0.12$ | $4.04 \pm 1.52$ | $96.71 \pm 0.30$ | $4.35 \pm 1.86$ |
| MC-CP | $90.05 \pm 0.07$ | $2.71 \pm 0.98$ | $92.50 \pm 0.19$ | $3.00 \pm 1.09$ | $92.11 \pm 0.09$ | $3.16 \pm 1.20$ | $96.71 \pm 0.11$ | $3.94 \pm 1.47$ | $98.53 \pm 0.15$ | $4.11 \pm 1.81$ |

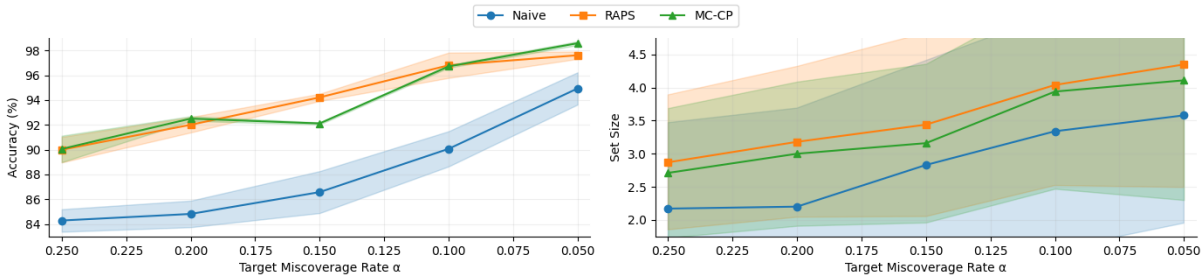



Figure 3.8: Visualised trends of average prediction set accuracy (left) and size (right) for all comparative CP methods on the CIFAR-10 dataset under varying target miscoverage rates $\alpha$.

Across all target miscoverage levels $\alpha$, the expected monotonic behaviour is observed: as $\alpha$ decreases (stricter coverage requirements), both prediction set accuracy and set size increase for all methods. This confirms that each CP method is correctly calibrated with respect to its nominal confidence level. Among the comparative methods, Naive CP consistently produces the smallest prediction sets but at the cost of noticeably lower accuracy than the RAPS and MC-CP, and in one case, it fails to achieve the target miscoverage ($94.93 \pm 0.28$ accuracy at $\alpha = 0.05$). In contrast, RAPS and MC-CP exhibit strong conformity to the target miscoverage rates, with MC-CP consistently achieving better accuracy while maintaining smaller prediction sets. For instance, at $\alpha = 0.1$, MC-CP achieves $96.71 \pm 0.11$ accuracy with an average set size of $3.94 \pm 1.47$, compared to $95.80 \pm 0.12$ and $4.04 \pm 1.52$ with RAPS. This trend is observed across all values of $\alpha$, validating that the performance trend observed in the core results persists with the expected monotonic behaviour.

To further illustrate these trends, Figure 3.8 visualises the relationship between target miscoverage rate $\alpha$, prediction set accuracy, and set size. As expected, Naive CP consistently exhibits the lowest accuracy across all $\alpha$ values, confirming its tendency toward under-coverage. Both RAPS and MC-CP maintain the highest accuracy, while MC-CP produces smaller prediction set sizes.

Next, we ablate the Adaptive MC Dropout parameters, particularly the threshold $\delta$ and the patience $\rho_{\max}$, to examine their effect on MC-CP's efficiency and predictive performance. Results can be seen in Table 3.7. As expected, relaxing either parameter (larger $\delta$ or smaller $\rho_{\max}$) reduces the number of stochastic forward passes, thereby improving computational efficiency at the expense of marginally lower accuracy and larger set size. Conversely, stricter convergence criteria (smaller $\delta$ or larger $\rho_{\max}$) lead to increased sampling and higher predictive fidelity, reflecting the trade-off between efficiency and stability inherent in adaptive stochastic inference.

Table 3.7: Average prediction set accuracy, Adaptive MC Dropout forward passes, and prediction set size for MC-CP on the CIFAR-10 dataset under various combinations of the threshold $\delta$ and patience $\rho_{\max}$ parameters.

| | | **Threshold $\delta$** | | | | |
|---|---|---|---|---|---|---|
| **Patience $\rho_{\max}$** | **Metric** | **0.1** | **0.01** | **0.001** | **0.0001** | **0.00001** |
| 1 | Accuracy ↑ | $93.75 \pm 0.43$ | $96.00 \pm 0.50$ | $96.40 \pm 0.48$ | $98.00 \pm 0.50$ | $98.00 \pm 0.50$ |
| | Forward Passes | $3.77 \pm 0.82$ | $9.22 \pm 3.16$ | $52.71 \pm 18.59$ | $387.71 \pm 150.13$ | $985.02 \pm 75.78$ |
| | Set Size ↓ | $5.99 \pm 1.32$ | $5.35 \pm 1.39$ | $5.34 \pm 1.21$ | $5.01 \pm 1.50$ | $4.21 \pm 1.33$ |
| 10 | Accuracy ↑ | $96.20 \pm 0.40$ | $96.60 \pm 0.49$ | $97.60 \pm 0.49$ | $98.33 \pm 0.47$ | $98.67 \pm 0.47$ |
| | Forward Passes | $13.10 \pm 1.50$ | $31.45 \pm 10.84$ | $142.36 \pm 56.79$ | $812.34 \pm 211.21$ | $1000.00 \pm 0.00$ |
| | Set Size ↓ | $4.80 \pm 1.44$ | $5.30 \pm 1.21$ | $5.26 \pm 1.34$ | $4.11 \pm 1.81$ | $4.06 \pm 1.48$ |
| 100 | Accuracy ↑ | $97.33 \pm 0.47$ | $97.50 \pm 0.50$ | $98.25 \pm 0.43$ | $98.62 \pm 0.48$ | $98.50 \pm 0.48$ |
| | Forward Passes | $103.09 \pm 1.60$ | $156.77 \pm 35.56$ | $672.57 \pm 180.16$ | $1000.00 \pm 0.00$ | $1000.00 \pm 0.00$ |
| | Set Size ↓ | $5.01 \pm 1.43$ | $5.27 \pm 1.20$ | $4.75 \pm 1.32$ | $4.08 \pm 1.33$ | $4.19 \pm 1.20$ |

The only combination of the two hyperparameters that doesn't manage to reach the desired miscoverage rate of $\alpha = 0.05$ is $\delta = 0.1$ and $\rho_{\max} = 1$. In this case, MC-CP achieves an accuracy of $93.75 \pm 0.43$, approximately 1.25% under the target rate. It does so with an average of $3.77 \pm 0.82$ forward passes, which suggests that the severe early termination of the sampling process causes insufficient averaging over the stochastic dropout masks. As a result, the predictive mean becomes biased and the estimated variance under-representative of true epistemic uncertainty. This effectively *poisons* the predictive distribution, yielding perturbed predictions, thus harming the prediction set creation process.

However, even under relatively loose stopping conditions, MC-CP maintains strong accuracy and hits the target miscoverage rate while performing an order of magnitude fewer forward passes compared to the full-pass baseline. For example, with $\delta = 0.01$ and $\rho_{\max} = 1$, MC-CP achieves an accuracy of $96.00 \pm 0.50$ all with $9.22 \pm 3.16$ forward passes. Under stricter convergence criteria, the number of forward passes naturally increases, leading to a slight improvement in accuracy and a gradual reduction in prediction set size. This indicates that additional samples provide marginal calibration benefits and modestly refine the resulting prediction sets.

To further examine the relationship between adaptive sampling behaviour and predictive performance presented in Table 3.7, Figure 3.9 presents the Spearman's $\rho$ correlations between the average prediction set size and both the number of forward passes and achieved accuracy across all combinations of $\delta$ and $\rho_{\max}$.

The observed strong negative correlations ($\rho = -0.77$ and $\rho = -0.76$, both with $p < 0.001$) confirm the trends discussed previously: as the variance stabilisation threshold becomes stricter, more forward passes are required, which modestly increases accuracy while yielding smaller, more precise prediction sets. This relationship reinforces that adaptive termination effectively balances computational efficiency with predictive refinement, where additional stochastic samples improve calibration and compactness.

Given the strong correlations observed and the theoretical convergence properties established in Section 3.2.1, it is useful to visualise how these diminishing returns manifest empirically.

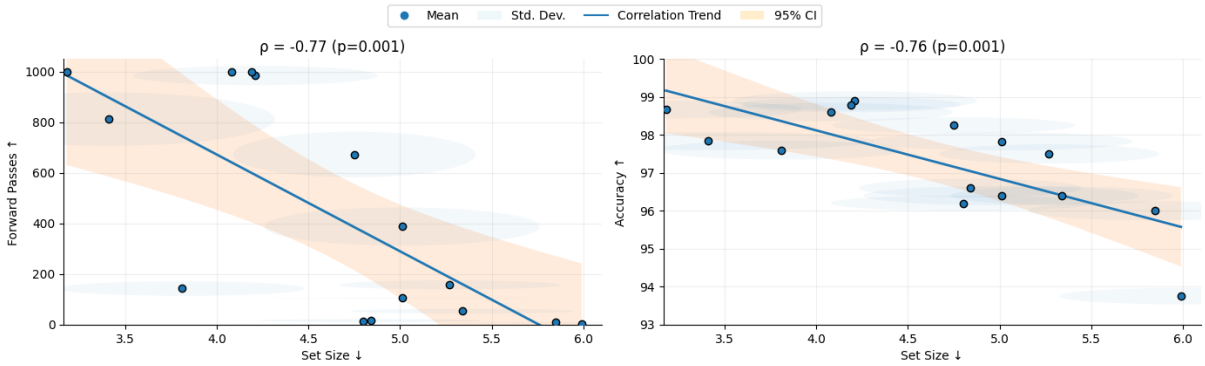



Figure 3.9: Spearman's $\rho$ correlation between the average prediction set size and (left) the number of Adaptive MC Dropout forward passes, and (right) the resulting accuracy, across all parameter combinations of $\delta$ and $\rho_{\max}$ reported in Table 3.7.

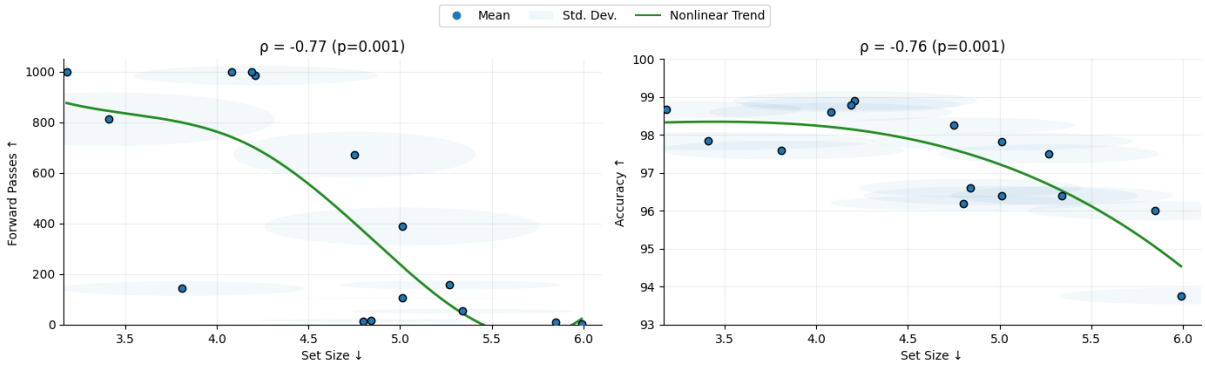



Figure 3.10: Non-linear relationship between average prediction set size and (left) the number of Adaptive MC Dropout forward passes and (right) achieved accuracy, modelled using a Univariate Spline fit to the reported data in Table 3.7.

Figure 3.10 presents a non-linear Univariate Spline fit to the data reported in Table 3.7, capturing the relationship between the average prediction set size, number of forward passes, and resulting accuracy across all parameter configurations. Both trends exhibit clear monotonic behaviour: as the adaptive sampling process progresses and the number of forward passes increases, the predictive accuracy improves while the average set size gradually decreases. It can be observed that, beyond approximately 500 forward passes, both curves begin to flatten, indicating that additional sampling yields negligible benefits in either calibration or predictive refinement. For example, between $10-500$ forward passes, the accuracy increases from $\approx 93.75-98\%$. However, for the remaining $501-1000$ forward passes, accuracy improves marginally from $\approx 98-98.67\%$. This behaviour empirically corroborates Theorem 3.1, which states that the mean and variance of MC Dropout converge as sampling continues, validating that Adaptive MC Dropout successfully identifies a practical convergence point that balances efficiency and reliability.

Following the analysis of the Adaptive MC Dropout parameters in the classification setting, we repeat the experiments on the Boston Housing regression task to assess their effect on MC-CP's efficacy. Results are reported in Table 3.8. As discussed in Section 3.2.3, MC-CP exhibits

Table 3.8: Mean absolute error (MAE), average Adaptive MC Dropout forward passes, and coverage for MC-CP on the Boston Housing dataset under various combinations of the threshold $\delta$ and patience $\rho_{\max}$ parameters.

| | | **Threshold $\delta$** | | | | |
|---|---|---|---|---|---|---|
| **Patience $\rho_{\max}$** | **Metric** | **0.1** | **0.01** | **0.001** | **0.0001** | **0.00001** |
| 1 | MAE ↓ | $0.41 \pm 0.02$ | $0.40 \pm 0.01$ | $0.40 \pm 0.02$ | $0.40 \pm 0.01$ | $0.41 \pm 0.03$ |
| | Forward Passes | $3.00 \pm 0.00$ | $3.15 \pm 0.84$ | $4.83 \pm 2.70$ | $5.03 \pm 3.03$ | $5.14 \pm 3.47$ |
| | Coverage ↑ | $84.85 \pm 3.81$ | $87.83 \pm 2.17$ | $88.85 \pm 1.63$ | $83.70 \pm 9.78$ | $87.50 \pm 2.72$ |
| 10 | MAE ↓ | $0.39 \pm 0.02$ | $0.38 \pm 0.02$ | $0.37 \pm 0.01$ | $0.36 \pm 0.02$ | $0.36 \pm 0.01$ |
| | Forward Passes | $82.38 \pm 209.43$ | $339.28 \pm 368.87$ | $382.57 \pm 393.45$ | $403.24 \pm 399.50$ | $459.85 \pm 383.65$ |
| | Coverage ↑ | $84.78 \pm 14.13$ | $89.67 \pm 0.54$ | $97.28 \pm 0.54$ | $96.74 \pm 2.17$ | $97.28 \pm 2.72$ |
| 100 | MAE ↓ | $0.37 \pm 0.01$ | $0.34 \pm 0.01$ | $0.35 \pm 0.01$ | $0.35 \pm 0.03$ | $0.35 \pm 0.01$ |
| | Forward Passes | $484.33 \pm 432.50$ | $984.32 \pm 98.02$ | $974.85 \pm 127.46$ | $977.25 \pm 113.95$ | $982.10 \pm 108.46$ |
| | Coverage ↑ | $97.83 \pm 2.17$ | $98.37 \pm 1.63$ | $93.76 \pm 5.98$ | $92.93 \pm 5.98$ | $95.65 \pm 2.17$ |

slightly different behaviour in regression compared to classification; MC-CP improves coverage in both cases, but through wider quantile intervals rather than smaller prediction sets. The following analysis, therefore, investigates how the adaptive sampling dynamics manifest under this distinct behaviour in the regression setting.

We observe three similar trends. Firstly, under stricter convergence criteria, the number of forward passes increases. As a result, the coverage increases and MAE decreases, showing that more sampling allows for better calibration and prediction of the conformalised quantiles. Finally, looser convergence criteria result in fewer forward passes, which can actually harm/poison the predictive mean of the Adaptive MC Dropout process, and as a result, harm coverage guarantees. For example, for $\delta = 0.1$ and $\rho_{\max} = 1$, MC-CP only performs $3.00 \pm 0.00$ forward passes and as a result achieves a meagre $84.85 \pm 3.81$ coverage of the ground truth data, $\approx 5.15\%$ below the target miscoverage rate of $\alpha = 0.1$. Under stricter criteria, $\delta = 1e-5$ and $\rho_{\max} = 100$, MC-CP overachieves on the target coverage, covering $95.65 \pm 2.17$ of the ground truth data whilst performing $982.10 \pm 108.46$ on average. Just as in the classification analysis, we can partially observe regions in the parameter combinations where diminishing returns begin to emerge. For example, a moderate combination of $\delta = 1e-3$ and $\rho_{\max} = 10$, performs $382.57 \pm 393.45$ forward passes on average but achieves a coverage and MAE of $97.28 \pm 0.54$ and $0.37 \pm 0.01$ respectively, results on par with the stricter combinations that perform nearly $\approx 1000$ forward passes.

To further illustrate these quantitative trends, Figure 3.11 visualises the predicted quantile intervals for MC-CP for the results reported in Table 3.8. We can visually observe the trends already discussed, as the convergence threshold tightens or the patience increases, the intervals become progressively tighter, leading to more consistent containment of the true target values due to better calibration of the quantiles and less chance of poisoning the Adaptive MC Dropout process.

The correlations in Figure 3.12 mirror the regression trends observed earlier. Coverage is positively associated with the number of stochastic forward passes ($\rho = 0.75$ with $p < 0.001$), indicating that stricter adaptive sampling leads to better-calibrated intervals. At the same time, coverage is negatively associated with MAE ($\rho = -0.73$ with $p < 0.001$), suggesting that

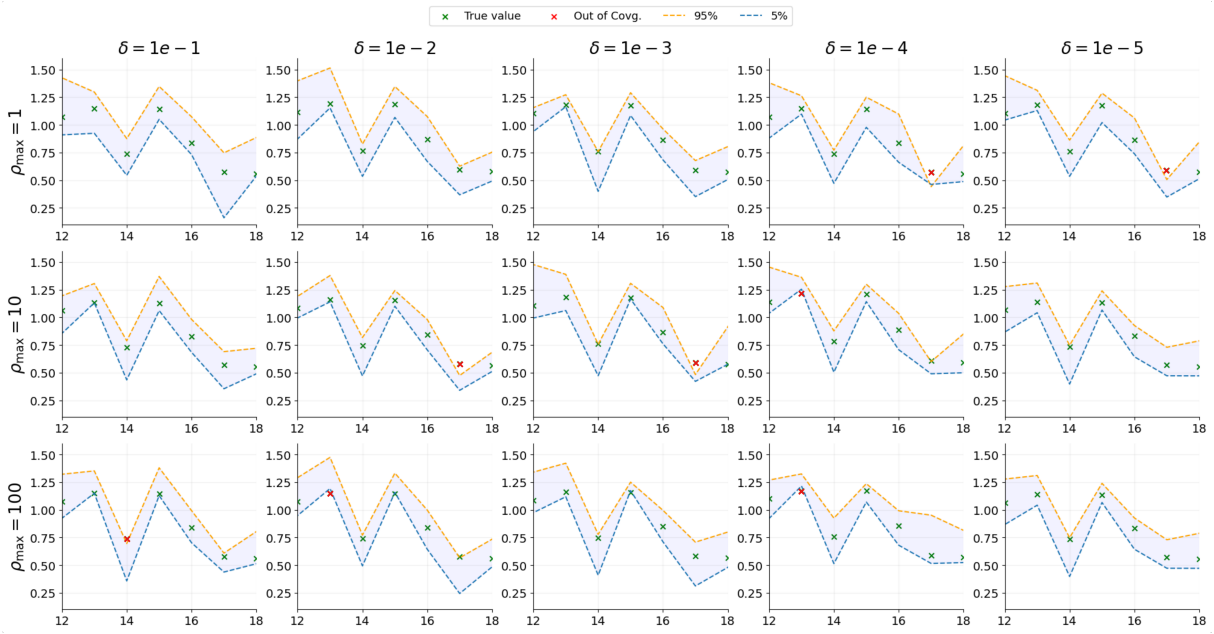



Figure 3.11: Predicted quantile intervals on the Boston Housing dataset for MC-CP under various combinations of the threshold $\delta$ (columns) and patience $\rho_{\max}$ (rows) for the results reported in Table 3.8. The dashed lines denote the 5th and 95th predicted quantiles, with the shaded region representing the corresponding prediction interval. Green markers indicate true target values falling within the interval, while red markers denote values outside coverage.

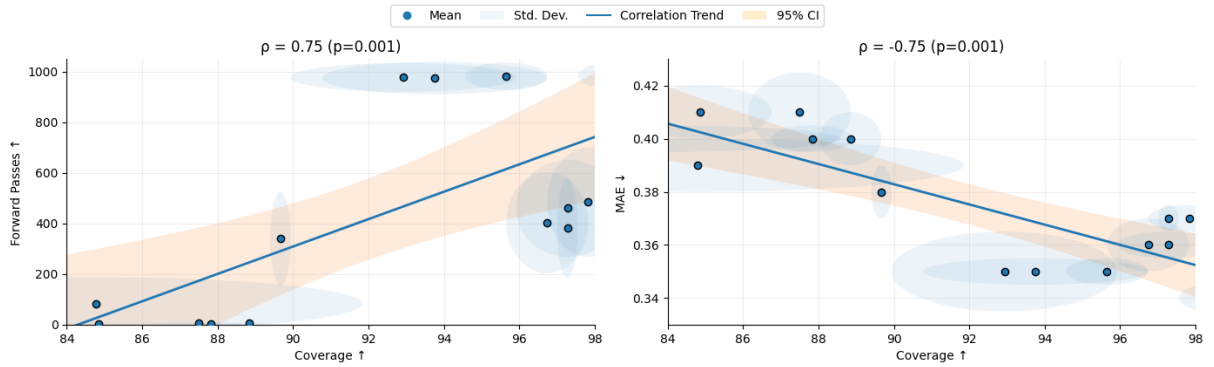



Figure 3.12: Spearman's $\rho$ correlation between the average prediction coverage and (left) the number of Adaptive MC Dropout forward passes, and (right) the resulting MAE, across all parameter combinations of $\delta$ and $\rho_{\max}$ reported in Table 3.8.

when MC-CP gains higher coverage, it typically comes with better accuracy, showing that better calibration both improves coverage and accuracy. Together, these relationships confirm that additional sampling primarily enhances calibration quality.

Figure 3.13 visualises the non-linear relationship between coverage with both computation and accuracy. As coverage increases, the number of forward passes initially rises sharply, reflecting the stricter sampling required for tighter calibration, but eventually plateaus and begins to decline slightly, indicating diminishing returns beyond approximately 400 forward passes. A

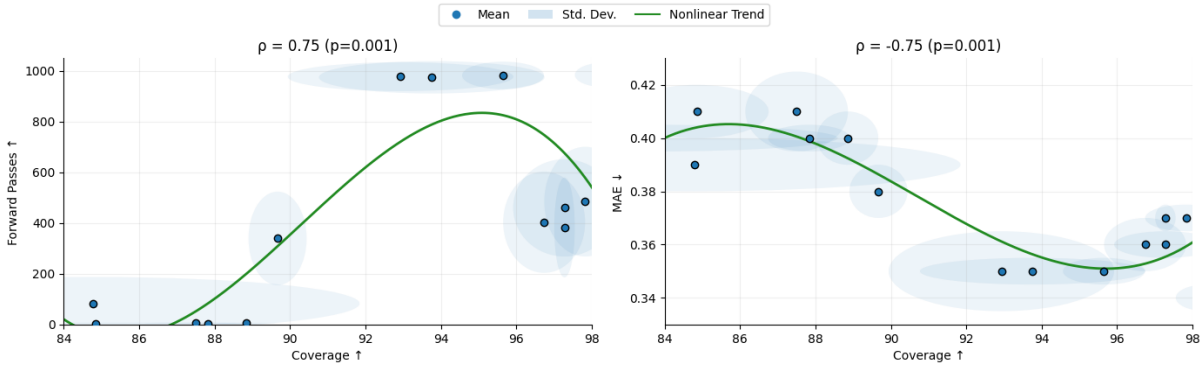



Figure 3.13: Non-linear relationship between average prediction coverage and (left) the number of Adaptive MC Dropout forward passes and (right) achieved MAE, modelled using a Univariate Spline fit to the reported data in Table 3.8.

similar pattern is observed for accuracy: MAE decreases steadily with increasing coverage before stabilising, suggesting that additional sampling yields marginal improvements once the model reaches near-nominal calibration.

For completeness, we note that MC-CP inherits some hyperparameter structure overlaps as RAPS, including the regularisation strength $\beta$ and the number of unpenalised classes $N_{\text{reg}}$. A detailed sensitivity analysis of these parameters is presented in their paper [111], specifically Table 4 for the ablation of $\beta$ and $N_{\text{reg}}$. Given the shared conformal calibration foundation, we expect qualitatively similar trends in MC-CP. However, due to the Adaptive MC Dropout Component, the reported set sizes for MC-CP would typically be reduced (as shown empirically in our results), as both the head and tail of the predictive distribution are regularised instead of solely the tail in RAPS.

Having validated MC-CP across both classification and regression tasks, we next investigate its robustness, interpretability, and decision utility in greater depth within the classification domain.

### 3.3.3 Distribution and Covariate Shift Results

We next evaluate the robustness of the proposed MC-CP framework under distributional and covariate shift to empirically test the third hypothesis: whether MC-CP maintains valid coverage and stable prediction set sizes when the test data deviate from the training distribution. This analysis focuses on the classification setting, where such shifts are known to significantly degrade predictive accuracy.

To ensure comparability, we retain the same experimental configuration and hyperparameters as in Section 3.3.1.2. Additional experimental information surrounding the construction of the distribution and covariate shifts can be found in Appendix A.5.2.

Firstly, we examine how MC-CP and the comparative CP methods behave under distributional shift, specifically in the presence of OOD data. In realistic deployment scenarios, models are frequently exposed to OOD data samples that differ from the training distribution, and thus,

Table 3.9: Average prediction set sizes for all comparative CP methods for two in-distribution (ID) and out-of-distribution (OOD) dataset pairings: MNIST → Fashion-MNIST and CIFAR-10 → CIFAR-100. Δ Set Size indicates the relative change in prediction set size between ID and OOD conditions.

| | **MNIST → Fashion-MNIST** | | | **CIFAR-10 → CIFAR-100** | | |
|---|---|---|---|---|---|---|
| **Technique** | **Set Size (ID) ↓** | **Set Size (OOD) ↑** | **Δ Set Size ↑** | **Set Size (ID) ↓** | **Set Size (OOD) ↑** | **Δ Set Size ↑** |
| Naive | $0.95 \pm 0.22$ | $1.01 \pm 0.08$ | $0.06 \pm 0.14$ | $3.53 \pm 1.63$ | $4.24 \pm 1.29$ | $0.71 \pm 0.34$ |
| RAPS | $1.08 \pm 0.35$ | $5.67 \pm 1.83$ | $4.59 \pm 1.48$ | $4.35 \pm 1.86$ | $4.96 \pm 1.52$ | $0.61 \pm 0.34$ |
| MC-CP | $1.06 \pm 0.33$ | $6.60 \pm 1.72$ | $5.54 \pm 1.39$ | $4.11 \pm 1.81$ | $5.57 \pm 1.56$ | $1.46 \pm 0.25$ |

a reliable UQ method should reflect this uncertainty such that ID data samples have lower uncertainty than OOD. For example, an image classifier trained to classify types of vehicles may still confidently produce predictions when presented with images of dogs or pedestrians at test time, even though these categories were never seen during training, because the model cannot selectively abstain from prediction and must assign each input to one of the known classes. We employ two dataset pairings that reflect increasing levels of semantic and visual shift: MNIST → Fashion-MNIST and CIFAR-10 → CIFAR-100. The MNIST → Fashion-MNIST pairing introduces a moderate domain shift while preserving a similar greyscale structure, making it suitable for evaluating calibration under mild distributional shift. In contrast, the CIFAR-10 → CIFAR-100 pairing represents a substantially more challenging scenario, where both the visual complexity and class granularity increase; CIFAR-100 contains finer-grained subclasses that overlap semantically with those in CIFAR-10 (e.g., truck vs. pickup truck), providing a more realistic approximation of near OOD conditions encountered in practice. In the case of CP methods, the average set size for the OOD data should be significantly higher than that of the ID data to reflect well-calibrated uncertainty estimates. Table 3.9 summarises these results.

Across both dataset pairings, all methods exhibit the expected increase in prediction set size between ID and OOD data, correctly reflecting higher uncertainty. However, the magnitude of this increase varies considerably between methods. Naive CP shows minimal responsiveness to the OOD shift, with only marginal changes in set size (e.g., $\Delta = 0.06 \pm 0.14$ on MNIST → Fashion-MNIST), indicating limited sensitivity. In contrast, both RAPS and MC-CP adapt their prediction sets more appropriately, expanding them to better reflect the uncertainty that comes with OOD samples.

Crucially, MC-CP outperforms both comparative methods across both dataset pairings, with the highest increase in set size (e.g., $\Delta = 1.46 \pm 0.25$ on CIFAR-10 → CIFAR-100) compared to RAPS ($\Delta = 0.61 \pm 0.34$), suggesting that the integration of uncertainty into the calibration stage enables greater detection of OOD samples. This behaviour aligns with the theoretical design of MC-CP: while RAPS regularises only the tail of the predictive distribution, MC-CP regularises both the head and tail, yielding prediction sets that remain compact on ID data yet expand proportionally as uncertainty increases under OOD conditions.

To qualitatively illustrate this behaviour, Figure 3.14 presents example ID and OOD predictions from the CIFAR-10 → CIFAR-100 pairing. As expected, MC-CP produces compact sets for




Figure 3.14: Example ID (left) and OOD (right) samples from the CIFAR-10 → CIFAR-100 dataset pairing. The true label and corresponding prediction set are shown above and below each image, respectively.

Table 3.10: Average prediction set accuracy and set size for all comparative CP methods under two style-based distributional shifts on the MNIST dataset: Line Thickness and Ink Bleed/Feathering, each applied at two severity levels. Δ Set Size indicates the relative change in prediction set size between the in-distribution (Base) and shifted (Severity 1–2) conditions.

| | **Line Thickness** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **Base** | | **Severity 1** | | | **Severity 2** | | |
| **Technique** | **Accuracy ↑** | **Set Size ↓** | **Accuracy ↑** | **Set Size ↓** | **Δ Set Size ↓** | **Accuracy ↑** | **Set Size ↓** | **Δ Set Size ↓** |
| Naive | $94.99 \pm 0.58$ | $0.95 \pm 0.22$ | $92.61 \pm 0.54$ | $1.01 \pm 0.05$ | $0.06 \pm 0.23$ | $56.48 \pm 5.93$ | $1.05 \pm 0.71$ | $0.10 \pm 0.74$ |
| RAPS | $98.90 \pm 0.04$ | $1.08 \pm 0.35$ | $99.56 \pm 0.02$ | $2.26 \pm 0.70$ | $1.18 \pm 0.78$ | $89.16 \pm 0.37$ | $3.53 \pm 1.42$ | $2.45 \pm 1.46$ |
| MC-CP | $99.68 \pm 0.01$ | $1.06 \pm 0.33$ | $99.67 \pm 0.04$ | $2.22 \pm 0.36$ | $1.16 \pm 0.49$ | $93.61 \pm 0.38$ | $4.04 \pm 1.73$ | $2.98 \pm 1.76$ |

| | **Ink Bleed/Feathering** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **Base** | | **Severity 1** | | | **Severity 2** | | |
| **Technique** | **Accuracy ↑** | **Set Size ↓** | **Accuracy ↑** | **Set Size ↓** | **Δ Set Size ↓** | **Accuracy ↑** | **Set Size ↓** | **Δ Set Size ↓** |
| Naive | $94.99 \pm 0.58$ | $0.95 \pm 0.22$ | $97.01 \pm 0.97$ | $1.04 \pm 0.63$ | $0.09 \pm 0.67$ | $89.32 \pm 0.97$ | $1.05 \pm 0.55$ | $0.10 \pm 0.59$ |
| RAPS | $98.90 \pm 0.04$ | $1.08 \pm 0.35$ | $99.97 \pm 0.35$ | $2.23 \pm 0.75$ | $1.15 \pm 0.83$ | $99.74 \pm 0.56$ | $4.02 \pm 1.89$ | $2.94 \pm 1.92$ |
| MC-CP | $99.68 \pm 0.01$ | $1.06 \pm 0.33$ | $100.00 \pm 0.00$ | $2.19 \pm 0.36$ | $1.13 \pm 0.49$ | $99.98 \pm 0.45$ | $3.51 \pm 1.63$ | $2.45 \pm 1.66$ |

ID samples, while appropriately broadening the prediction sets for OOD samples to preserve coverage under heightened uncertainty.

Next, we examine how MC-CP and the comparative CP methods behave under style-based distributional shifts, where the visual appearance of inputs is altered while the underlying semantic class remains unchanged. Unlike OOD data, where prediction sets are expected to expand substantially to reflect epistemic uncertainty over unseen categories, style-based shifts probe the model's robustness. In these scenarios, a well-calibrated conformal predictor should maintain high coverage of the ground-truth class with only minimal expansion of the prediction set. We investigate two common types of visual degradation: line thickness and ink bleed/feathering, which mimic realistic artefacts arising from handwritten digits. Both perturbations are applied at two levels of severity to progressively increase distortion. This experiment is conducted exclusively on the MNIST dataset, where the simplicity and structural consistency of the digits allow for the generation of interpretable and controlled style perturbations. Results are presented in Table 3.10.

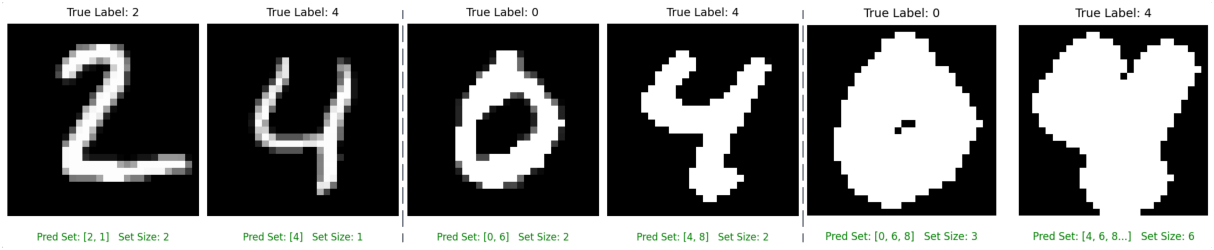
Across both types of style-shifts, all methods exhibit the expected trend of increasing prediction set size with greater shift severity, reflecting rising uncertainty as the input deviates from the training distribution. However, MC-CP consistently achieves the best balance between accuracy and set compactness across all shift severities.

Across both style-based shifts, it can be observed that Naive CP struggles to maintain the target miscoverage rate of $\alpha = 0.05$ in most cases, failing to achieve the expected 95% coverage in three out of four severity settings. For instance, under the line thickness shift at severity 2, Naive CP attains an accuracy of only $56.48 \pm 5.93$, falling short of the target despite an increase in average set size to $1.05 \pm 0.71$. Across all shifts and severities, it at most increases its set size between the base and shifted data by $0.10 \pm 0.74$, showing its inability to correctly adapt to uncertain inputs. RAPS, by contrast, maintains valid coverage across three out of four perturbation severities but does so at the cost of substantially larger prediction sets. It exhibits the highest average set size in three out of the four severity conditions, indicating that it compensates for increased uncertainty primarily through over-expansion of prediction sets. For example, under ink bleed at severity 2, RAPS achieves $99.74 \pm 0.56$ accuracy but with an average set size of $4.02 \pm 1.89$. MC-CP consistently provides the most balanced trade-off between accuracy and set compactness. It achieves the highest coverage in all four severity conditions (only missing out on the target miscoverage rate in one setting, however, it still achieves the highest coverage of all the methods) while maintaining moderate set sizes. For ink bleed at severity 2, MC-CP achieves an accuracy of $99.98 \pm 0.45$ with a compact set size of $3.51 \pm 1.63$, massively outperforming both Naive CP ($89.32 \pm 0.97$, set size $1.05 \pm 0.55$) and RAPS ($99.74 \pm 0.56$, set size $4.02 \pm 1.89$).

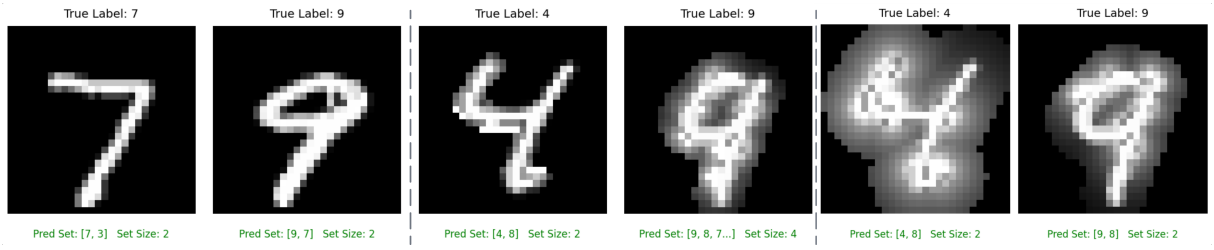
To further illustrate these quantitative findings, Figure 3.15 presents qualitative examples of MC-CP prediction sets under the line thickness and ink bleed/feathering style-shifts. In both types of shifts, prediction sets expand gradually as the distortions intensify, reflecting MC-CP's adaptive response to higher epistemic uncertainty. Importantly, the true labels remain contained within the prediction sets even at the highest severity levels, demonstrating that MC-CP effectively preserves coverage.

Next, we examine how MC-CP and the comparative CP methods behave under covariate shift, where the input distribution changes due to environmental or sensor-level perturbations while the underlying label distribution remains the same. Such shifts commonly occur in real-world deployments, for instance, when a trained classifier is applied to images captured under different lighting conditions, camera lenses, or compression settings. Despite these input changes, the model continues to produce confident predictions even when its learned representations are no longer well-aligned with the altered data, potentially leading to miscalibration or degraded coverage.

To simulate these conditions, we evaluate four covariate shifts that reflect realistic sources of input variation encountered in practical deployment settings. Blob occlusion models partial visibility or sensor obstruction (e.g., raindrops, dirt, or foreground clutter on a camera lens), while defocus/lens blur captures loss of sharpness due to focus errors or motion blur. JPEG-Compression noise/artifacts approximates degradation introduced by lossy image transmission




(a) Line Thickness




(b) Ink Bleed/Feathering

Figure 3.15: Qualitative examples of MC-CP prediction sets under style-based distributional shifts on the MNIST dataset: (a) Line Thickness, and (b) Ink Bleed/Feathering. In each row, the two leftmost images correspond to ID (Base) samples, the middle two to Severity 1, and the rightmost two to Severity 2. The true label and corresponding prediction set are shown above and below each image, respectively. Green labels indicate that the ground-truth class is contained within the prediction set.

or storage, which is particularly relevant in bandwidth-limited or edge settings. Finally, elastic distortions emulate local geometric deformations arising from imperfect digitisation, printing artifacts, or natural object warping. We perform this analysis on both the MNIST and CIFAR-10 datasets to assess performance under varying dataset complexity and visual diversity. The results are summarised in Table 3.11.

Under covariate shift, an effective CP method should expand its prediction sets in proportion to the induced uncertainty while maintaining stable accuracy/coverage. MC-CP exhibits this behaviour consistently across all four distortions, adapting its prediction set size accordingly whilst maintaining strong accuracy. In contrast, similar to the style-shift results, Naive CP displays limited responsiveness; set sizes increase only marginally. RAPS tends to over-expand, compensating for uncertainty but at the cost of larger prediction sets. MC-CP therefore achieves the most balanced trade-off, particularly on the more complex CIFAR-10 dataset, where covariates can harm smaller, fine-grained features needed for accurate classification.

In the case of blob occlusion, the distortion introduces a large disruption by masking key object regions, such as occluding the dog in an image and leaving only the background context, making class prediction highly difficult/uncertain. On MNIST, Naive CP's accuracy drops sharply from $94.99 \pm 0.58$ to $70.91 \pm 0.79$ on average, with the average set size increasingly minimally by $0.09 \pm 0.67$. Both RAPS and MC-CP respond appropriately by expanding sets to approximately 3 labels on average, thus keeping accuracy higher ($70.91 \pm 0.79$ and $93.67 \pm 0.56$ for Naive CP and MC-CP, respectively, on MNIST) but still marginally missing out on the target miscoverage.

Table 3.11: Average prediction set accuracy and set size for all comparative CP methods under four covariate shift types on the MNIST and CIFAR-10 datasets: Blob Occlusion, Defocus/Lens Blur, JPEG Compression Noise/Artifacts, and Elastic Distortions. Δ Set Size indicates the relative change in prediction set size between the in-distribution (Base) and shifted (Covariate) conditions.

| | **Blob Occlusion** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **MNIST** | | | | | **CIFAR-10** | | | | |
| | **Base** | | **Covariate** | | | **Base** | | **Covariate** | | |
| **Technique** | **Acc. ↑** | **Set Size ↓** | **Acc. ↑** | **Set Size ↓** | **Δ Set Size ↓** | **Acc. ↑** | **Set Size ↓** | **Acc. ↑** | **Set Size ↓** | **Δ Set Size ↓** |
| Naive | $94.99 \pm 0.58$ | $0.95 \pm 0.22$ | $70.91 \pm 0.79$ | $1.03 \pm 0.62$ | $0.08 \pm 0.66$ | $94.93 \pm 0.28$ | $3.53 \pm 1.63$ | $82.01 \pm 0.73$ | $4.06 \pm 1.37$ | $0.53 \pm 2.13$ |
| RAPS | $98.90 \pm 0.04$ | $1.08 \pm 0.35$ | $93.14 \pm 0.68$ | $3.15 \pm 1.49$ | $2.07 \pm 1.53$ | $96.71 \pm 0.30$ | $4.35 \pm 1.86$ | $89.11 \pm 0.88$ | $5.70 \pm 1.69$ | $1.35 \pm 2.51$ |
| MC-CP | $99.68 \pm 0.01$ | $1.06 \pm 0.33$ | $93.67 \pm 0.56$ | $3.07 \pm 1.44$ | $2.01 \pm 1.48$ | $98.53 \pm 0.15$ | $4.11 \pm 1.81$ | $91.09 \pm 0.65$ | $5.71 \pm 1.60$ | $1.60 \pm 2.42$ |

| | **Defocus/Lens Blur** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **MNIST** | | | | | **CIFAR-10** | | | | |
| | **Base** | | **Covariate** | | | **Base** | | **Covariate** | | |
| **Technique** | **Acc. ↑** | **Set Size ↓** | **Acc. ↑** | **Set Size ↓** | **Δ Set Size ↓** | **Acc. ↑** | **Set Size ↓** | **Acc. ↑** | **Set Size ↓** | **Δ Set Size ↓** |
| Naive | $94.99 \pm 0.58$ | $0.95 \pm 0.22$ | $96.97 \pm 0.78$ | $1.04 \pm 0.63$ | $0.09 \pm 0.67$ | $94.93 \pm 0.28$ | $3.53 \pm 1.63$ | $89.04 \pm 0.42$ | $4.08 \pm 1.31$ | $0.55 \pm 2.09$ |
| RAPS | $98.90 \pm 0.04$ | $1.08 \pm 0.35$ | $100.00 \pm 0.00$ | $3.29 \pm 1.81$ | $2.21 \pm 1.84$ | $96.71 \pm 0.30$ | $4.35 \pm 1.86$ | $91.94 \pm 0.48$ | $5.28 \pm 1.47$ | $0.93 \pm 2.37$ |
| MC-CP | $99.68 \pm 0.01$ | $1.06 \pm 0.33$ | $100.00 \pm 0.00$ | $2.99 \pm 1.65$ | $1.93 \pm 1.68$ | $98.53 \pm 0.15$ | $4.11 \pm 1.81$ | $94.98 \pm 0.14$ | $5.47 \pm 1.48$ | $1.36 \pm 2.34$ |

| | **JPEG-Compression Noise/Artifacts** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **MNIST** | | | | | **CIFAR-10** | | | | |
| | **Base** | | **Covariate** | | | **Base** | | **Covariate** | | |
| **Technique** | **Acc. ↑** | **Set Size ↓** | **Acc. ↑** | **Set Size ↓** | **Δ Set Size ↓** | **Acc. ↑** | **Set Size ↓** | **Acc. ↑** | **Set Size ↓** | **Δ Set Size ↓** |
| Naive | $94.99 \pm 0.58$ | $0.95 \pm 0.22$ | $99.92 \pm 0.43$ | $1.03 \pm 0.51$ | $0.08 \pm 0.56$ | $94.93 \pm 0.28$ | $3.53 \pm 1.63$ | $92.93 \pm 0.32$ | $3.54 \pm 1.32$ | $0.01 \pm 2.10$ |
| RAPS | $98.90 \pm 0.04$ | $1.08 \pm 0.35$ | $100.00 \pm 0.00$ | $2.04 \pm 0.45$ | $0.96 \pm 0.57$ | $96.71 \pm 0.30$ | $4.35 \pm 1.86$ | $97.52 \pm 0.64$ | $4.73 \pm 1.65$ | $0.38 \pm 2.49$ |
| MC-CP | $99.68 \pm 0.01$ | $1.06 \pm 0.33$ | $100.00 \pm 0.00$ | $2.02 \pm 0.39$ | $0.96 \pm 0.51$ | $98.53 \pm 0.15$ | $4.11 \pm 1.81$ | $96.78 \pm 0.76$ | $5.03 \pm 1.64$ | $0.92 \pm 2.44$ |

| | **Elastic Distortions** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **MNIST** | | | | | **CIFAR-10** | | | | |
| | **Base** | | **Covariate** | | | **Base** | | **Covariate** | | |
| **Technique** | **Acc. ↑** | **Set Size ↓** | **Acc. ↑** | **Set Size ↓** | **Δ Set Size ↓** | **Acc. ↑** | **Set Size ↓** | **Acc. ↑** | **Set Size ↓** | **Δ Set Size ↓** |
| Naive | $94.99 \pm 0.58$ | $0.95 \pm 0.22$ | $82.17 \pm 0.57$ | $1.04 \pm 0.62$ | $0.09 \pm 0.66$ | $94.93 \pm 0.28$ | $3.53 \pm 1.63$ | $92.43 \pm 0.95$ | $4.00 \pm 1.28$ | $0.47 \pm 2.07$ |
| RAPS | $98.90 \pm 0.04$ | $1.08 \pm 0.35$ | $96.79 \pm 0.25$ | $2.76 \pm 0.83$ | $1.68 \pm 0.90$ | $96.71 \pm 0.30$ | $4.35 \pm 1.86$ | $96.34 \pm 0.52$ | $5.06 \pm 1.54$ | $0.71 \pm 2.41$ |
| MC-CP | $99.68 \pm 0.01$ | $1.06 \pm 0.33$ | $96.81 \pm 0.24$ | $2.73 \pm 1.18$ | $1.67 \pm 1.22$ | $98.53 \pm 0.15$ | $4.11 \pm 1.81$ | $97.46 \pm 0.84$ | $5.58 \pm 1.53$ | $1.47 \pm 2.37$ |

For the defocus/lens blur, the degradation primarily preserves global structure while removing fine texture, leading to moderate uncertainty rather than semantic ambiguity. On CIFAR-10, Naive CP falls to $89.04 \pm 0.42$ accuracy, once again showing poor coverage. In this experiment, even RAPS struggles to correctly adapt its set sizes and achieves an accuracy of $91.94 \pm 0.48$, while MC-CP barely misses out on the target miscoverage, reaching $94.98 \pm 0.14$ coverage. Even though MC-CP has the largest set size expansion between the clean and covariate data, a $1.36 \pm 2.34$ increase, it manages to do so with the highest coverage, showing a more accurate and controlled expansion to uncertainty.

JPEG-Compression noise/artifacts primarily affect local texture statistics rather than overall object identity. Consequently, this covariate is the least effective at harming the predictive performance of all the CP methods. On CIFAR-10, Naive CP once again fails to adapt to uncertainty and, evidently, is the only method to fail to hit the target miscoverage with an accuracy of $92.93 \pm 0.32$ and a tiny set size expansion of $0.01 \pm 2.10$ on average. MC-CP and RAPS both achieve the target coverage and expand their set sizes appropriately, with MC-CP

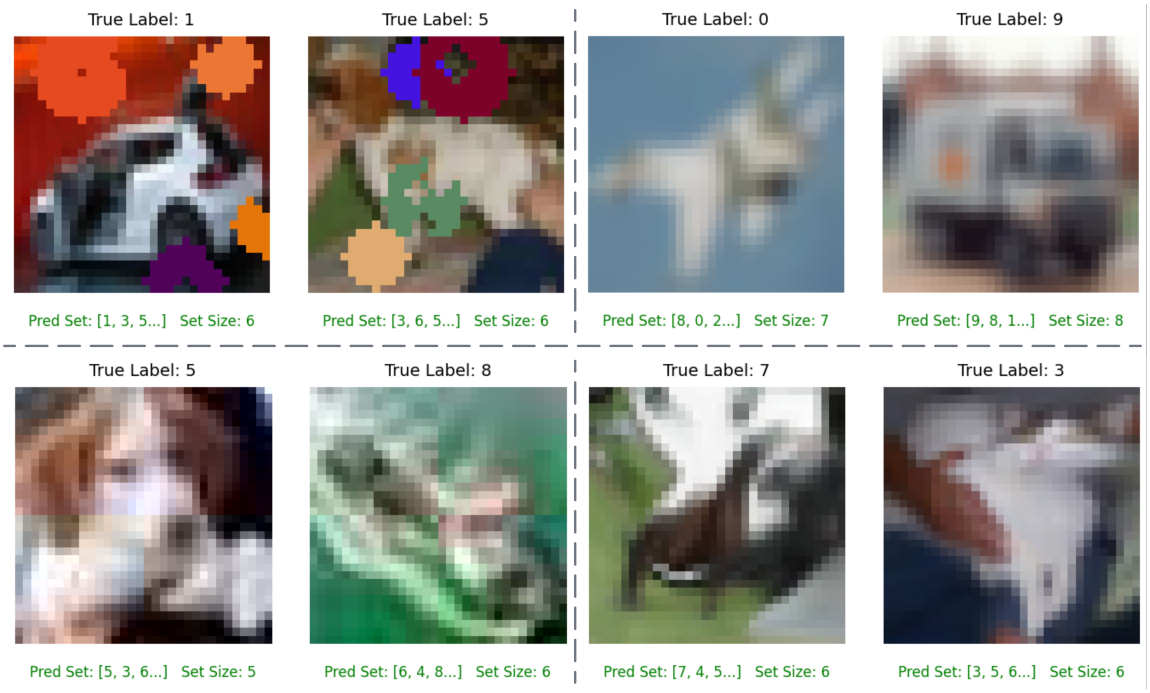



Figure 3.16: Qualitative examples of MC-CP prediction sets under various covariate shifts on the CIFAR-10 dataset: Blob Occlusion (top left), Defocus/Lens Blur (top right), JPEG Compression Noise/Artifacts (bottom left), and Elastic Distortions (bottom right). The true label and corresponding prediction set are shown above and below each image, respectively. Green labels indicate that the ground-truth class is contained within the prediction set.

expanding the most with $0.92 \pm 2.44$ on average.

Finally, the elastic distortion covariate introduces non-linear geometric warping, such as stretching a digit or elongating an animal's limbs, which alters object geometry and can blur class boundaries. We observe similar trends to the JPEG-Compression results, where Naive CP fails to hit the target miscoverage, whereas RAPS and MC-CP successfully achieve it. MC-CP once again has the larger set size expansion whilst achieving a greater accuracy of all comparative methods.

Across all four covariate shifts, MC-CP consistently achieves the highest or near-highest accuracy, with prediction sets that expand proportionally to the severity of the distortion. Interestingly, in several covariate conditions, MC-CP attains both the highest accuracy and the largest prediction set size, contrasting the behaviour observed in the base results, where MC-CP achieved superior accuracy with more compact sets. This effect arises because the regularisation applied to the predictive head via Adaptive MC Dropout redistributes probability mass away from the most confident classes under covariate shift. As a result, the calibrated cumulative probability threshold $\tau_\alpha$ is reached more slowly, necessitating inclusion of additional classes in the prediction set to preserve valid coverage. In contrast, RAPS, which lacks such head regularisation, often remains overconfident even under distributional perturbations, producing smaller sets but suffering reduced accuracy due to premature truncation of the prediction set.

Table 3.12: Average prediction set accuracy and size for all comparative CP methods on the CIFAR-10 dataset under various calibration dataset sizes $|\mathcal{D}_{\text{cal}}|$.

| | $\lvert\mathcal{D}_{\text{cal}}\rvert = 50$ | | $\lvert\mathcal{D}_{\text{cal}}\rvert = 100$ | | $\lvert\mathcal{D}_{\text{cal}}\rvert = 1000$ | | $\lvert\mathcal{D}_{\text{cal}}\rvert = 2500$ | | $\lvert\mathcal{D}_{\text{cal}}\rvert = 10,000$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Technique** | **Accuracy ↑** | **Set Size ↓** | **Accuracy ↑** | **Set Size ↓** | **Accuracy ↑** | **Set Size ↓** | **Accuracy ↑** | **Set Size ↓** | **Accuracy ↑** | **Set Size ↓** |
| Naive | $92.73 \pm 0.45$ | $3.25 \pm 1.28$ | $93.87 \pm 0.73$ | $3.61 \pm 1.39$ | $94.34 \pm 0.11$ | $3.59 \pm 1.32$ | $94.93 \pm 0.28$ | $3.53 \pm 1.63$ | $95.01 \pm 0.11$ | $3.42 \pm 1.44$ |
| RAPS | $99.67 \pm 0.87$ | $7.13 \pm 1.34$ | $98.83 \pm 0.40$ | $5.67 \pm 1.76$ | $97.85 \pm 0.62$ | $5.23 \pm 1.69$ | $96.71 \pm 0.30$ | $4.35 \pm 1.86$ | $97.89 \pm 0.83$ | $3.91 \pm 1.71$ |
| MC-CP | $99.67 \pm 0.65$ | $7.10 \pm 1.29$ | $97.91 \pm 0.05$ | $4.76 \pm 1.70$ | $98.72 \pm 0.57$ | $5.24 \pm 1.79$ | $98.53 \pm 0.15$ | $4.11 \pm 1.81$ | $98.65 \pm 0.58$ | $3.73 \pm 1.72$ |

To further illustrate these quantitative trends, Figure 3.16 presents qualitative examples of MC-CP prediction sets under each covariate shift on CIFAR-10. The visualisations clearly reflect the adaptive behaviour observed in Table 3.11: as the severity of the distortion increases, prediction sets expand to capture higher uncertainty while still containing the true class.

Overall, these results empirically validate our third hypothesis: MC-CP maintains valid coverage and stable predictive behaviour under distributional and covariate shift. Across all distortion types, the method adapts its prediction set size to reflect increased uncertainty while preserving high accuracy, demonstrating robustness to deviations from the training distribution.

### 3.3.4 Practical Limitations of Marginal Validity

While the previous analyses have demonstrated that MC-CP achieves robust UQ, even under various distributional and covariate shifts. MC-CP, like other CP methods, relies on the notion of marginal validity (the guarantee the ground truth label is with the prediction set with probability of at least $1-\alpha$). However, marginal validity represents an aggregate notion of reliability and does not necessarily imply uniform coverage across all subpopulations. In practical deployment, data can be scarce, imbalanced, or (as we tested in Section 3.3.3) systematically shifted. Under such conditions, the guarantee may mask heterogeneity in coverage, some classes may be over covered while others are under-covered, or compromise the usefulness of prediction sets. Evidently, in this section, we examine the practical limitations of marginal validity.

To ensure comparability, we retain the same experimental configuration and hyperparameters as in Section 3.3.1.2.

Firstly, we analyse the effect of the size of the calibration dataset size $|\mathcal{D}_{\text{cal}}|$ on the prediction set accuracy and size on the CIFAR-10 dataset. Since split CP relies on empirical quantiles estimated from the calibration nonconformity scores, smaller calibration datasets naturally introduce higher variance in the quantile estimate. Results for this experiment are summarised in Table 3.12.

Across all calibration set sizes, a clear trend emerges: smaller calibration datasets ($|\mathcal{D}_{\text{cal}}| \to 50$) lead to inflated prediction sets and reduced practical utility. When $|\mathcal{D}_{\text{cal}}| = 50$, both RAPS and MC-CP exhibit near-perfect accuracy ($99.67 \pm 0.97$ and $99.67 \pm 0.65$, respectively) but achieve this through substantial over-expansion of the prediction sets, with average sizes exceeding 7. This behaviour arises because limited calibration data (in the case of $|\mathcal{D}_{\text{cal}}| = 50$, each class has 5 data points) produce noisy quantile estimates, prompting overly conservative set thresholds to maintain marginal validity. Although the expansion guarantees coverage, it renders the

Table 3.13: Average prediction set accuracy per class for all comparative CP methods on the CIFAR-10 dataset under two calibration set distributions: balanced, where the calibration dataset is evenly partitioned, and extremely imbalanced. Highlighting denotes class-wise miscoverage.

| | Balanced $(0.10 \times 10)$ | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Technique** | **Airplane** | **Automobile** | **Bird** | **Cat** | **Deer** | **Dog** | **Frog** | **Horse** | **Ship** | **Truck** | **Average** |
| Naive | $93.47 \pm 1.02$ | $96.79 \pm 0.69$ | $92.38 \pm 2.19$ | $93.15 \pm 2.24$ | $97.63 \pm 0.79$ | $93.08 \pm 0.83$ | $99.28 \pm 0.41$ | $91.36 \pm 2.85$ | $96.57 \pm 0.82$ | $94.83 \pm 1.38$ | $94.93 \pm 0.28$ |
| RAPS | $95.63 \pm 1.19$ | $97.72 \pm 0.68$ | $94.69 \pm 1.25$ | $97.83 \pm 0.48$ | $98.57 \pm 0.64$ | $97.18 \pm 0.93$ | $98.83 \pm 0.42$ | $95.48 \pm 0.87$ | $97.25 \pm 1.03$ | $95.86 \pm 1.72$ | $96.71 \pm 0.30$ |
| MC-CP | $97.82 \pm 0.91$ | $99.27 \pm 0.39$ | $98.19 \pm 0.79$ | $98.33 \pm 0.51$ | $98.97 \pm 0.33$ | $98.18 \pm 0.29$ | $98.95 \pm 0.59$ | $98.56 \pm 0.37$ | $98.21 \pm 0.47$ | $98.48 \pm 0.31$ | $98.53 \pm 0.15$ |

| | Extreme Imbalance $(0.90, 0.011 \times 9)$ | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Technique** | **Airplane** | **Automobile** | **Bird** | **Cat** | **Deer** | **Dog** | **Frog** | **Horse** | **Ship** | **Truck** | **Average** |
| Naive | $87.61 \pm 0.43$ | $93.46 \pm 0.37$ | $75.72 \pm 0.55$ | $83.37 \pm 0.48$ | $87.43 \pm 0.39$ | $80.35 \pm 0.52$ | $96.83 \pm 0.31$ | $81.14 \pm 0.46$ | $95.84 \pm 0.34$ | $84.93 \pm 0.44$ | $86.87 \pm 0.45$ |
| RAPS | $94.41 \pm 0.34$ | $95.53 \pm 0.29$ | $95.14 \pm 0.33$ | $95.78 \pm 0.28$ | $96.32 \pm 0.27$ | $96.24 \pm 0.31$ | $97.91 \pm 0.22$ | $88.69 \pm 0.44$ | $96.88 \pm 0.25$ | $91.43 \pm 0.38$ | $94.83 \pm 0.31$ |
| MC-CP | $96.20 \pm 0.28$ | $97.11 \pm 0.25$ | $97.33 \pm 0.26$ | $96.67 \pm 0.29$ | $99.40 \pm 0.18$ | $96.90 \pm 0.27$ | $98.90 \pm 0.20$ | $95.73 \pm 0.31$ | $96.67 \pm 0.29$ | $94.90 \pm 0.36$ | $96.78 \pm 0.27$ |

prediction sets impractical for deployment, as an average set containing most of the label space (7 out of 10 classes) offers little value for ID data.

As the calibration dataset size increases, all methods average set size slowly reduce. For example, MC-CP has an average set size of $4.76 \pm 1.70$ at $|\mathcal{D}_{\text{cal}}| = 100$ but a set size of $3.73 \pm 1.72$ at $|\mathcal{D}_{\text{cal}}| = 10,000$. RAPS observes a similar trend but at a slightly higher magnitude. In contrast, Naive CP fails to reach the target miscoverage rate (accuracy $< 95\%$) at every calibration dataset size, except for $|\mathcal{D}_{\text{cal}}| = 10,000$, demonstrating its inability to adjust to calibration uncertainty.

While MC-CP reduces prediction set sizes and improves accuracy relative to other conformal methods, its quantile estimates also become fragile when calibration data are limited. In such cases, excessive set inflation, though theoretically ensuring coverage, undermines the interpretability and practical usefulness of the resulting prediction sets in real-world settings.

Next, we investigate how class imbalance within the calibration dataset influences both overall and class-wise performance, examining its effects on prediction set size, accuracy. In this experiment, the calibration dataset size is fixed at $|\mathcal{D}_{\text{cal}}| = 250$, but the class distribution is varied between a stratified (balanced) configuration (each class covers 10% of the total samples) and an extremely imbalanced configuration where the Frog class constitutes approximately 90% of the calibration samples and each of the remaining nine classes contributes only about 1.1%. The frog class was selected as the majority because, as shown in Table 3.4, it is the most accurately classified class in the base CNN model. This setup, therefore, creates a challenging and realistic form of bias, favouring a majority class with reliable accuracy and nonconformity estimates while depriving minority classes of sufficient calibration coverage. Evaluating class-wise coverage is crucial, as marginal validity alone may conceal systematic miscoverage across underrepresented classes, leading to unreliable UQ in imbalanced data regimes. Results for per-class prediction set accuracy and set size are summarised in Table 3.13 and Table 3.14, respectively.

Across both regimes, we observe that Naive CP struggles to maintain consistent class-wise coverage, especially under imbalance. In the balanced setting, its per-class accuracy varies substantially across classes ($91.36 \pm 2.85$ for Horse compared to $99.28 \pm 0.41$ for Frog), already indicating class-dependent calibration quality. Under imbalance, this disparity widens sharply: minority classes such as Bird drop to $75.72 \pm 0.55$, while the majority frog class remains well

Table 3.14: Average prediction set size per class for all comparative CP methods on the CIFAR-10 dataset under two calibration set distributions: balanced, where the calibration dataset is evenly partitioned, and extremely imbalanced.

| | **Balanced** ($0.10 \times 10$) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Technique** | **Airplane** | **Automobile** | **Bird** | **Cat** | **Deer** | **Dog** | **Frog** | **Horse** | **Ship** | **Truck** | **Average** |
| Naive | $3.70 \pm 1.24$ | $2.94 \pm 1.21$ | $3.96 \pm 1.15$ | $4.09 \pm 1.12$ | $3.78 \pm 1.12$ | $3.85 \pm 1.12$ | $2.94 \pm 1.13$ | $3.50 \pm 1.38$ | $2.93 \pm 1.21$ | $3.61 \pm 1.33$ | $3.53 \pm 1.63$ |
| RAPS | $4.58 \pm 1.69$ | $4.02 \pm 2.02$ | $4.62 \pm 1.32$ | $4.89 \pm 1.24$ | $4.38 \pm 1.31$ | $4.67 \pm 1.31$ | $3.23 \pm 1.61$ | $4.41 \pm 1.74$ | $3.82 \pm 1.92$ | $4.88 \pm 1.89$ | $4.35 \pm 1.86$ |
| MC-CP | $4.33 \pm 1.73$ | $3.48 \pm 1.91$ | $4.50 \pm 1.44$ | $4.81 \pm 1.38$ | $4.32 \pm 1.43$ | $4.63 \pm 1.35$ | $2.81 \pm 1.72$ | $4.17 \pm 1.91$ | $3.85 \pm 1.88$ | $4.20 \pm 1.93$ | $4.11 \pm 1.81$ |

| | **Extreme Imbalance** ($0.90, 0.011 \times 9$) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Technique** | **Airplane** | **Automobile** | **Bird** | **Cat** | **Deer** | **Dog** | **Frog** | **Horse** | **Ship** | **Truck** | **Average** |
| Naive | $2.43 \pm 0.77$ | $2.18 \pm 0.79$ | $2.70 \pm 0.81$ | $2.85 \pm 0.79$ | $2.60 \pm 0.79$ | $2.76 \pm 0.78$ | $2.06 \pm 0.76$ | $2.59 \pm 0.95$ | $2.03 \pm 0.73$ | $2.40 \pm 0.85$ | $2.46 \pm 0.85$ |
| RAPS | $3.58 \pm 1.36$ | $3.60 \pm 1.49$ | $3.93 \pm 1.15$ | $4.20 \pm 1.08$ | $3.99 \pm 1.08$ | $3.92 \pm 1.09$ | $3.08 \pm 1.25$ | $4.04 \pm 1.44$ | $3.03 \pm 1.44$ | $3.94 \pm 1.53$ | $3.73 \pm 1.36$ |
| MC-CP | $4.67 \pm 1.60$ | $4.41 \pm 1.84$ | $4.79 \pm 1.29$ | $5.07 \pm 1.27$ | $4.61 \pm 1.31$ | $4.84 \pm 1.24$ | $3.63 \pm 1.54$ | $4.47 \pm 1.69$ | $4.20 \pm 1.80$ | $4.88 \pm 1.71$ | $4.56 \pm 1.59$ |

covered at $96.83 \pm 0.31$. In fact, Naive CP fails to achieve coverage on 8 of 10 classes when under imbalance.

RAPS partially improves upon the poor performance. Compared to Naive CP, which fails to achieve coverage on 6 classes on the balanced calibration dataset, RAPS only fails on 1 class, highlighting its more accurate calibration. However, under imbalance, it fails to achieve coverage on 3 classes; at worst, it gets an accuracy of $88.69 \pm 0.44$ on the Horse class. This extreme imbalance also makes RAPS fail to achieve the target coverage across the whole test dataset, just like Naive CP, reaching an average accuracy of $94.83 \pm 0.31$. The improved overall and per-class accuracies under the imbalanced regime also coincide with a relative expansion of the prediction sets, where the average set increases from $2.46 \pm 0.85$ to $3.73 \pm 1.36$ compared to Naive CP.

MC-CP, by contrast, displays the most stable and class-balanced coverage across both regimes. In the balanced setting, it achieves $98.53 \pm 0.15$ average accuracy with minimal class-wise deviation (ranging from $97.82 \pm 0.91$ to $99.27 \pm 0.39$). Notably, under the balanced setting, MC-CP is the only method to achieve class-wise coverage. Under imbalance, MC-CP once again achieves overall coverage, with $96.78 \pm 0.27$ accuracy, with very minor inter-class deviation compared to other methods. It only fails to cover 1 class, the fewest of any method, where the failed class is Truck with $94.90 \pm 0.36$ accuracy, barely missing out on accurate coverage. This robustness extends to efficiency: MC-CP's mean prediction set size was smaller compared to RAPS on the balanced dataset, but actually increased on the imbalanced dataset, showing the extra regularisation can aid calibration stabilisation.

These results underscore a fundamental limitation of marginal validity in split CP: class-wise coverage is not theoretically guaranteed, even though MC-CP achieved it empirically in our experiment. Under balanced calibration, all methods maintain nominal average coverage, yet subtle per-class deviations already emerge, indicating that validity holds only in expectation across the dataset. When the calibration data become highly imbalanced, this issue is further amplified; no method achieves per-class coverage, with at least one class for each method falling below the 95% target.

### 3.3.5 Class-wise Coverage Results

Building on the findings from Section 3.3.4, which show that MC-CP improves overall coverage but does not ensure uniform reliability across individual classes, we next explore how it can be modified to promote more balanced, class-wise coverage. In many real-world classification settings, such as safety-critical perception or medical diagnosis, this heterogeneity in coverage can be undesirable, as systematic under-coverage for specific classes translates to unequal confidence and potentially biased decision-making.

Several methods have been proposed to extend conformal prediction toward class-wise or conditional coverage, yet each comes with practical limitations. Mondrian Conformal Prediction [241] achieves formal class-conditional validity by partitioning the calibration data by class and computing a separate quantile within each subset. While this guarantees per-class coverage, it adds an extra assumption: there should be *enough* data in each partition. Thus, when calibration samples are scarce, the quantile estimates for minority classes become unreliable. Moreover, because Mondrian calibration splits the calibration data across partitions, it no longer leverages the full calibration dataset, resulting in reduced efficiency and greater sensitivity to tail-sample variability. As shown in Appendix A.2, these effects cause MC-CP utilising Mondrian calibration to degrade substantially under imbalance, confirming that the theoretical guarantee does not always translate to practical reliability.

More recently, Augmented Label Rank Calibration (ALRC) [242] has been proposed as a more data-efficient alternative. Rather than partitioning the calibration set, ALRC augments the standard conformal ranking framework with label-wise regularisation terms that explicitly correct class-dependent miscoverage. In doing so, it provides approximate per-class coverage while retaining the marginal validity of standard conformal prediction and utilising nearly all available calibration data. However, ALRC achieves this improvement only partially; its class-wise coverage remains empirical rather than guaranteed, and its effectiveness diminishes when tail classes have extremely few examples.

Based on this, we introduce a variant of MC-CP that directly addresses class-wise miscoverage by integrating a soft tail-aware calibration penalty inspired by the soft tail-aware CP (sTACP) framework [243]. The core idea is to introduce a smooth, class-aware penalty that down-weights over-represented classes and compensates under-represented ones during calibration, thereby promoting balanced empirical coverage without requiring partitioned calibration data or additional supervision.

Formally, sTACP introduces an additional regularisation penalty to the nonconformity scores during the calibration stage of MC-CP to correct class-wise miscoverage. The modified calibration score is defined as:

$$\kappa_i \leftarrow \sum_{\xi=1}^{\text{rank}_{y_i}} \hat{y}_{\xi}^{\text{sorted}} + \beta_{\text{reg}}(\text{rank}_{y_i} - N_{\text{reg}})_+ + \underbrace{\beta_{\text{pen}}(\text{rank}_{y_i} - N_{\text{pen}})_+ \hat{p}(y_i)}_{\text{sTACP Regularisation}} \tag{3.11}$$

where the first two terms correspond to the standard MC-CP formulation, and the final term

Table 3.15: Average prediction set accuracy per class for MC-CP and its sTACP variant on the CIFAR-10 dataset using the extremely imbalanced calibration dataset from Section 3.3.4. Highlighting denotes class-wise miscoverage.

| Technique | Airplane | Automobile | Bird | Cat | Deer | Dog | Frog | Horse | Ship | Truck | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|
| MC-CP | $96.20 \pm 0.28$ | $97.11 \pm 0.25$ | $97.33 \pm 0.26$ | $96.67 \pm 0.29$ | $99.40 \pm 0.18$ | $96.90 \pm 0.27$ | $98.90 \pm 0.20$ | $95.73 \pm 0.31$ | $96.67 \pm 0.29$ | $94.90 \pm 0.36$ | $96.78 \pm 0.27$ |
| MC-CP$^{\text{sTACP}}$ | $96.30 \pm 0.41$ | $98.90 \pm 0.22$ | $95.60 \pm 0.39$ | $98.90 \pm 0.21$ | $99.30 \pm 0.18$ | $95.20 \pm 0.44$ | $99.10 \pm 0.20$ | $97.40 \pm 0.31$ | $95.90 \pm 0.36$ | $95.90 \pm 0.34$ | $97.10 \pm 0.27$ |

Table 3.16: Average prediction set size per class for MC-CP and its sTACP variant on the CIFAR-10 dataset using the extremely imbalanced calibration dataset from Section 3.3.4.

| Technique | Airplane | Automobile | Bird | Cat | Deer | Dog | Frog | Horse | Ship | Truck | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|
| MC-CP | $4.67 \pm 1.60$ | $4.41 \pm 1.84$ | $4.79 \pm 1.29$ | $5.07 \pm 1.27$ | $4.61 \pm 1.31$ | $4.84 \pm 1.24$ | $3.63 \pm 1.54$ | $4.47 \pm 1.69$ | $4.20 \pm 1.80$ | $4.88 \pm 1.71$ | $4.56 \pm 1.59$ |
| MC-CP$^{\text{sTACP}}$ | $5.17 \pm 1.74$ | $3.58 \pm 2.16$ | $5.33 \pm 1.53$ | $5.63 \pm 1.43$ | $4.95 \pm 1.53$ | $3.89 \pm 1.44$ | $3.49 \pm 2.05$ | $4.37 \pm 2.16$ | $4.51 \pm 2.09$ | $4.53 \pm 2.14$ | $4.69 \pm 1.98$ |
| $\Delta$ Set Size | $0.50 \pm 2.36$ | $-0.83 \pm 2.84$ | $0.54 \pm 2.00$ | $0.56 \pm 1.91$ | $0.34 \pm 2.02$ | $-0.95 \pm 1.90$ | $-0.14 \pm 2.56$ | $-0.10 \pm 2.74$ | $0.31 \pm 2.76$ | $-0.35 \pm 2.74$ | $0.13 \pm 2.54$ |

represents the soft tail-aware calibration penalty. The coefficient $\beta_{\text{pen}}$ controls the magnitude of the class-aware adjustment, while $N_{\text{pen}}$ defines the rank threshold beyond which the penalty activates. The empirical class prior $\hat{p}(y_i)$ is estimated from the calibration set and serves as a frequency-dependent weight, amplifying the penalty for majority (head) classes and weakening it for under-represented (tail) classes. This modification inflates the nonconformity scores of head-class examples in proportion to both their calibration frequency and their rank position, thereby raising the estimated threshold $\tau_\alpha$ during calibration. This adjustment mitigates the dominance of head-class residuals in the global threshold, leading to empirically more balanced per-class coverage without explicit partitioning. Because the modification only redefines the nonconformity score, the standard marginal validity guarantee of conformal prediction remains theoretically preserved.

Using the same extremely imbalanced calibration set from Section 3.3.4 (Frog $\approx 90\%$ of $|\mathcal{D}_{\text{cal}}|$, others $\approx 1.1\%$), we compare standard MC-CP against the variant with an additional sTACP variant, denoted MC-CP$^{\text{sTACP}}$, to assess whether a soft, class-aware rank penalty can correct minority miscoverage without over-expanding prediction sets. Results are summarised in Table 3.15 and in Table 3.16, respectively.

Across all classes, MC-CP$^{\text{sTACP}}$ shifts coverage toward the target coverage of 95% while keeping sets essentially as compact as MC-CP. In the standard MC-CP, the only class that fails to reach the target coverage is Truck at $94.90 \pm 0.36$, but with the additional regularisation, this improves to $95.90 \pm 0.34$. This results in no class failing to reach the target miscoverage rate in the MC-CP$^{\text{sTACP}}$ variant. At the aggregate level, the average accuracy increases from $96.78 \pm 0.27$ to $97.10 \pm 0.27$, all whilst the mean set size only increases by $+0.13 \pm 2.54$. Interestingly, we see some individual classes reduce in accuracy, closer to the target miscoverage of 95%, and also some display a reduction in set size. For example, the Dog class reduces in accuracy from $96.90 \pm 0.27$ to $95.20 \pm 0.44$ with an average set size reduction of $-0.95 \pm 1.90$. This shows set-size shifts are small and directionally aligned with the coverage corrections (expansions where coverage needed lifting; contractions where it was excessive).

Overall, these results demonstrate that integrating soft tail-aware calibration into MC-CP effectively mitigates the class-wise miscoverage observed under data-starved conditions. Although

this extension provides no formal guarantees, like the previously discussed methods, it remains more practical and adaptive to severe class imbalance.

### 3.3.6 Risk-Sensitive Results

While the previous sections focused on improving the validity and uniformity of MC-CP's coverage, these methods implicitly treat all prediction errors as equally costly. In practice, however, the consequences of miscoverage are often asymmetric, as certain classes, regions, or uncertainty conditions may carry higher risk than others. The notion of risk offers a broader perspective on reliability by capturing not only how often a predictive failure occurs but also how significant its impact is within the specific application context.

From this perspective, coverage violations, excessive set sizes, or over-confident predictions in critical regions can all be interpreted as distinct manifestations of risk. A method that consistently meets nominal coverage may still be risky if its sets are too large to be actionable, or if it systematically underestimates uncertainty in classes that matter most; both of which we have seen in Section 3.3.4.

In this section, we extend MC-CP to account for this broader notion of risk. Rather than treating coverage as the sole indicator of reliability, we view prediction set quality through the discussed concept of risk, in which different forms of uncertainty, inefficiency, or misallocation of confidence contribute differently to the prediction set, rather than solely coverage. To operationalise this, we integrate the calibration from Conformal Risk Control (CSC) [116], which replaces the traditional coverage-only guarantee with a general constraint on a chosen risk function, into MC-CP. This extension provides a principled mechanism for aligning MC-CP's UQ with the operational demands of the downstream task.

Formally, to incorporate a broader notion of risk into the calibration procedure, we augment the nonconformity scores $\kappa_i$ with an additional risk-modulation term. The resulting score is defined as:

$$\kappa_i \leftarrow \sum_{\xi=1}^{\text{rank}_{y_i}} \hat{y}_{\xi}^{\text{sorted}} + \beta(\text{rank}_{y_i} - N_{\text{reg}})_+ + \underbrace{\lambda_{\text{risk}}\mathcal{R}(x_i, y_i)}_{\text{Risk Calibration}} \tag{3.12}$$

where $\lambda_{\text{risk}} \geqslant 0$ is a user-chosen weighting parameter, and $\mathcal{R}(x_i, y_i)$ is a user-chosen risk function that can penalise any notion of risk, such as large prediction sets, or class-weighted costs.

To ensure comparability, we retain the same experimental configuration and hyperparameters as in Section 3.3.1.2. We therefore focus solely on the effect of introducing risk-aware calibration into MC-CP, assessing how different choices of $\mathcal{R}(x_i, y_i)$ influence coverage behaviour, set-size efficiency, and class-wise reliability. The specific instantiations of the risk functions considered in this section, as well as their complete mathematical definitions are provided in Appendix A.5.3.

Firstly, we analyse the effect of a risk function that penalises large prediction sets using the same experimental setup as in Table 3.13, focusing on the small calibration dataset $|\mathcal{D}_{\text{cal}}| = 50$. This

Table 3.17: Average prediction set accuracy and size for all comparative CP methods on the CIFAR-10 dataset using a small calibration dataset $|\mathcal{D}_{\text{cal}}| = 50$. Results are shown for the base nonconformity score and for a risk-augmented variant incorporating a set-size penalty.

| | Base - $\mathbb{I}\{y \notin \Gamma\}$ | | $\mathbb{I}\{y \notin \Gamma\} + \lambda_{\text{size}}\lvert\Gamma\rvert$ | | |
|---|---|---|---|---|---|
| **Technique** | **Accuracy ↑** | **Set Size ↓** | **Accuracy ↑** | **Set Size ↓** | **Δ Set Size ↓** |
| Naive | $92.73 \pm 0.45$ | $3.25 \pm 1.28$ | $96.20 \pm 0.46$ | $4.16 \pm 1.58$ | $0.91 \pm 2.03$ |
| RAPS | $99.67 \pm 0.87$ | $7.13 \pm 1.34$ | $96.43 \pm 0.76$ | $4.46 \pm 1.57$ | $-2.67 \pm 2.06$ |
| MC-CP | $99.67 \pm 0.65$ | $7.10 \pm 1.29$ | $98.39 \pm 0.31$ | $5.03 \pm 1.71$ | $-2.07 \pm 2.14$ |

Table 3.18: Average prediction set accuracy for all comparative CP methods on the CIFAR-10 dataset. Results are shown for the base nonconformity score and for a risk-augmented variant incorporating class-specific weights $\lambda_y$, where automobile and truck are assigned higher cost ($\lambda_y = 2.0$), airplane, bird, and ship a lower cost ($\lambda_y = 0.5$), and the remaining classes have a standard cost ($\lambda_y = 1.0$). For clarity, only classes with altered weights are displayed.

| | Base - $\mathbb{I}\{y \notin \Gamma\}$ | | | | | $\lambda_y\mathbb{I}\{y \notin \Gamma\}$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Technique** | **Airplane** | **Automobile** | **Bird** | **Ship** | **Truck** | **Airplane** | **Automobile** | **Bird** | **Ship** | **Truck** |
| Naive | $93.47 \pm 1.02$ | $96.79 \pm 0.69$ | $92.38 \pm 2.19$ | $96.57 \pm 0.82$ | $94.83 \pm 1.38$ | $92.04 \pm 0.87$ | $96.89 \pm 0.46$ | $91.12 \pm 0.52$ | $96.60 \pm 0.61$ | $95.75 \pm 0.24$ |
| RAPS | $95.63 \pm 1.19$ | $97.72 \pm 0.68$ | $94.69 \pm 1.25$ | $97.25 \pm 1.03$ | $95.86 \pm 1.72$ | $95.12 \pm 0.19$ | $97.38 \pm 0.82$ | $91.81 \pm 0.59$ | $97.20 \pm 0.46$ | $96.33 \pm 0.72$ |
| MC-CP | $97.82 \pm 0.91$ | $99.27 \pm 0.39$ | $98.19 \pm 0.79$ | $98.21 \pm 0.47$ | $98.48 \pm 0.31$ | $94.71 \pm 0.06$ | $99.12 \pm 0.29$ | $95.27 \pm 0.67$ | $96.76 \pm 0.55$ | $99.31 \pm 0.64$ |

setting is particularly informative, as the base results exhibited substantially inflated prediction set sizes, rendering them impractical for real-world use and making it an ideal scenario to assess the impact of a set-size penalty. Results are reported in Table 3.17.

As discussed previously, when the calibration dataset is extremely small, the result is very large prediction sets for both RAPS and MC-CP, $\approx 7$ labels per prediction set, even if they both achieve near-perfect accuracy on average. Introducing the set-size penalty substantially reduces this inefficiency. For MC-CP, the average set size shrinks by $-2.07 \pm 2.14$ while keeping accuracy higher than the target miscoverage at $98.39 \pm 0.31$. RAPS exhibits an even stronger contraction, with sets decreasing $-2.67 \pm 2.06$ on average, but it reduces the accuracy a lot more than MC-CP at $96.43 \pm 0.76$. Simply adding this one additional risk term to the nonconformity scores during calibration can recover substantially more compact and practically useful sets in the low-calibration regime, trading only a small amount of coverage for a large gain in efficiency.

Secondly, we analyse the effect of a class-weighted risk function on MC-CP and the comparative methods. This experiment is designed to emulate a road-scene perception system, where miscoverage on vehicles is substantially more costly than on classes that rarely appear in driving scenarios. To capture this asymmetry, we assign a higher cost to the Automobile and Truck classes ($\lambda_y = 2.0$), lower costs to the Airplane, Bird, and Ship classes ($\lambda_y = 0.5$), and unit cost to all remaining classes. Table 3.18 and Table 3.19 report, respectively, the resulting class-wise accuracies and prediction set sizes for the base nonconformity score and its risk-weighted counterpart.

Across all methods, the class-weighted risk tends to reallocate coverage toward the high-cost vehicle classes, while allowing modest reductions for low-importance categories. For MC-CP,

Table 3.19: Average prediction set size for all comparative CP methods on the CIFAR-10 dataset. Results are shown for the base nonconformity score and for a risk-augmented variant incorporating class-specific weights $\lambda_y$, where automobile and truck are assigned higher cost ($\lambda_y = 2.0$), airplane, bird, and ship a lower cost ($\lambda_y = 0.5$), and the remaining classes have a standard cost ($\lambda_y = 1.0$). For clarity, only classes with altered weights are displayed.

| | Base - $\mathbb{I}\{y \notin \Gamma\}$ | | | | | $\lambda_y\mathbb{I}\{y \notin \Gamma\}$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Technique** | **Airplane** | **Automobile** | **Bird** | **Ship** | **Truck** | **Airplane** | **Automobile** | **Bird** | **Ship** | **Truck** |
| Naive | $3.70 \pm 1.24$ | $2.94 \pm 1.21$ | $3.96 \pm 1.15$ | $2.93 \pm 1.21$ | $3.61 \pm 1.33$ | $3.23 \pm 1.43$ | $4.03 \pm 1.60$ | $3.29 \pm 1.29$ | $2.61 \pm 1.61$ | $4.02 \pm 1.62$ |
| RAPS | $4.58 \pm 1.69$ | $4.02 \pm 2.02$ | $4.62 \pm 1.32$ | $3.82 \pm 1.92$ | $4.88 \pm 1.89$ | $4.05 \pm 1.66$ | $1.08 \pm 1.85$ | $3.86 \pm 1.21$ | $4.06 \pm 1.72$ | $5.65 \pm 1.78$ |
| MC-CP | $4.33 \pm 1.73$ | $3.48 \pm 1.91$ | $4.50 \pm 1.44$ | $3.85 \pm 1.88$ | $4.20 \pm 1.93$ | $4.19 \pm 1.47$ | $4.33 \pm 1.59$ | $4.28 \pm 1.31$ | $3.64 \pm 1.60$ | $4.91 \pm 1.61$ |

the Automobile and Truck classes maintain or improve their already high coverage. Specifically, Automobile remains $\approx 99\%$, whilst Truck improves from $98.49 \pm 0.31$ to $99.31 \pm 0.64$. Whereas Airplane and Bird drop slightly from very high coverage ($\approx 98\%$) towards accuracies closer to the target miscoverage level. A similar pattern is observed for RAPS: coverage for Automobile and Truck remains high, while Bird, treated as low-risk, experiences a noticeable decrease. This is exactly what the risk function forces the CP methods to adhere to; errors on cars and trucks become rarer, at the expense of a small increase in risk for classes that are less relevant in the driving context.

In terms of efficiency/set size, the impact is milder. For most methods and classes, average set sizes change only marginally when moving from the base to the class-weighted configuration. MC-CP, for example, slightly increases the set sizes for Automobiles and Trucks, while reducing set sizes for low-importance classes, indicating that the set sizes match the analysis seen in the per-class coverage levels.

Alternatively to applying risk control within the calibration procedure, risk can be incorporated even earlier, directly into the Adaptive MC-Dropout method that underpins MC-CP's uncertainty estimates. Instead of modifying the nonconformity scores, this experiment adjusts the uncertainty convergence behaviour of MC-Dropout in a class-dependent manner, allowing the model to spend more computational effort stabilising predictions for safety-critical classes. This is particularly relevant in perception systems for autonomous driving, where vehicle-related classes (e.g., automobiles and trucks) are far more consequential than rarely observed categories such as airplanes, birds, or ships.

To operationalise this, we adapt the Adaptive MC Dropout method to allow for a per-class $\delta$ rather than a globally defined one; we denote this changed variant as Adaptive MC-Dropout++. High-importance classes, automobile and truck, are given a stricter threshold $\delta = 1e-5$, forcing more stochastic forward passes to achieve stable variance, while all other classes retain the standard $\delta = 1e-4$. Table 3.20 reports the resulting class-wise accuracies, and Figure 3.17 visualises the per-class variance convergence for an example image.

The results demonstrate that class-specific uncertainty modulation influences behaviour in a direction aligned with the intended risk priorities. As expected, the base Adaptive MC Dropout produces a slightly lower average accuracy of $60.12 \pm 2.32$ compared to standard MC-Dropout, which achieves an average accuracy of $60.61 \pm 2.01$, due to the adaptive termination procedure.

Table 3.20: Average class-wise accuracy for MC-Dropout, Adaptive MC-Dropout, and the class-sensitive Adaptive MC-Dropout++ on the CIFAR-10 dataset. In Adaptive MC-Dropout++, the variance threshold $\delta$ is assigned per class rather than globally: all classes use $\delta = 1e-4$, except for Automobile and Truck, which use $\delta = 1e-5$ to reflect the higher importance of vehicle classes in road-scene perception.

| Technique | Airplane | Automobile | Bird | Cat | Deer | Dog | Frog | Horse | Ship | Truck | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|
| MC-Dropout | $54.72 \pm 3.75$ | $78.08 \pm 4.87$ | $37.76 \pm 8.03$ | $41.82 \pm 8.50$ | $54.70 \pm 5.94$ | $48.02 \pm 3.82$ | $89.92 \pm 3.55$ | $58.44 \pm 2.51$ | $81.62 \pm 3.26$ | $61.06 \pm 4.82$ | $60.61 \pm 2.01$ |
| Adaptive MC-Dropout | $54.26 \pm 4.74$ | $76.58 \pm 4.43$ | $39.84 \pm 7.24$ | $40.48 \pm 8.71$ | $53.32 \pm 6.41$ | $47.24 \pm 4.60$ | $90.42 \pm 3.60$ | $56.60 \pm 2.71$ | $83.00 \pm 3.39$ | $59.48 \pm 5.89$ | $60.12 \pm 2.32$ |
| Adaptive MC-Dropout++ | $54.68 \pm 1.81$ | $77.12 \pm 5.69$ | $39.14 \pm 9.14$ | $40.40 \pm 8.65$ | $53.44 \pm 6.37$ | $47.52 \pm 4.85$ | $89.50 \pm 3.40$ | $58.66 \pm 1.64$ | $81.72 \pm 4.02$ | $61.78 \pm 6.68$ | $60.39 \pm 2.22$ |

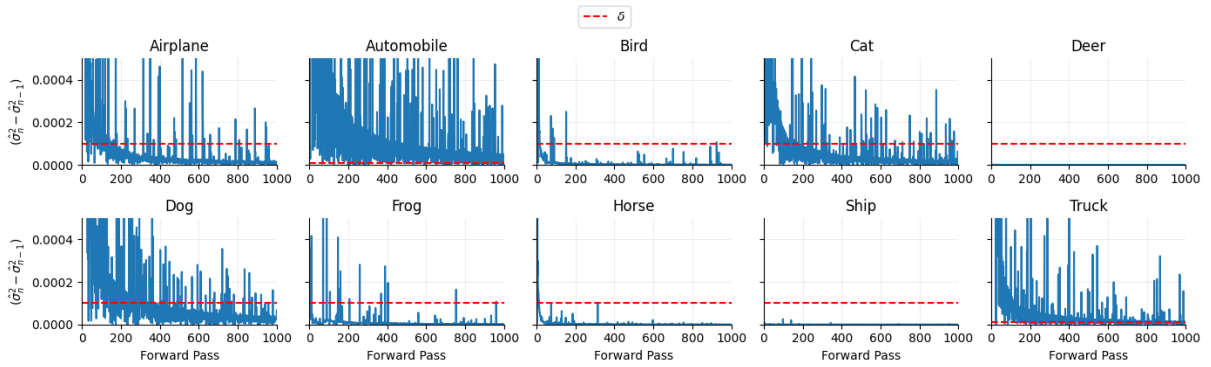



Figure 3.17: Variance convergence behaviour of Adaptive MC Dropout++ on CIFAR-10 dataset for an example Automobile image. Each curve represents the per-class variance difference $\hat{\sigma}_n^2 - \hat{\sigma}_{n-1}^2$ across stochastic forward passes, with the red dashed line indicating the per-class specific convergence threshold $\delta$ used for early termination.

Introducing a class-specific $\delta$ balances the computational efficiency of the Adaptive variant with the uncertainty refinement of the standard variant. More specifically, the class-specific $\delta$ enables Adaptive MC Dropout to terminate early for classes unrelated to Automobile and Truck, but for analogous classes, it forces extra computation/forward passes to be spent to ensure a more accurate variance convergence. As a result, this shrinks the small accuracy gap for these classes whilst keeping a similar accuracy to the base Adaptive MC Dropout for the remaining classes. For the Automobile class, the base Adaptive MC Dropout achieves an accuracy of $76.58 \pm 4.43$, which is marginally lower than MC-Dropout, which achieves an accuracy of $78.08 \pm 4.87$. Adaptive MC Dropout++ balances these and achieves an accuracy of $77.12 \pm 5.69$, maintaining computational efficiency and salient accuracy.

The variance convergence curves in Figure 3.17 illustrate this mechanism clearly on an example Automobile image. For the Automobile and Truck classes, the variance difference $\hat{\sigma}_n^2 - \hat{\sigma}_{n-1}^2$ remains above the tighter threshold for longer, requiring more sampling steps before termination. In contrast, low-importance classes reach the larger $\delta = 1e-4$ threshold much sooner, truncating sampling earlier. If both the Automobile and Truck classes had the global $\delta = 1e-4$, there would have been several instances during this example where the early termination of the Adaptive MC Dropout method could have occurred, even though the variance had not yet converged.

Overall, the experiments in this section demonstrate that risk-aware extensions of MC-CP provide a flexible and effective means of tailoring UQ to application-specific priorities. Whereas the

base MC-CP framework focuses purely on marginal coverage, the introduction of additional risk terms allows a practitioner to shape the behaviour of prediction sets along dimensions beyond mere validity. Across all variants, the resulting behaviour aligns with the intended notions of operational risk—allocating coverage, compactness, and computational effort in proportion to class importance, while preserving the core strength of MC-CP: its reliable, coverage-controlled treatment of uncertainty, now more strategically aligned with the downstream demands of real-world decision-making.

### 3.3.7 KDE-based Adaptive MC Dropout Results

The proposed Adaptive MC-Dropout method relies on monitoring the stability of the MC predictive samples to determine when sufficient stochastic forward passes have been collected. In the original formulation, this stability is assessed through the predictive variance: once the difference in variance $\hat{\sigma}_n^2 - \hat{\sigma}_{n-1}^2$ of the MC estimate falls below a fixed threshold $\delta$, sampling is terminated and the final predictive distribution is formed. While effective in expectation, variance-based stopping can be noisy and slow to stabilise, particularly for inputs whose predictive distributions are highly uncertain. In such cases, the variance signal can fluctuate, leading to inconsistent stopping times and less interpretable convergence behaviour.

To investigate this further, we explore an alternative, non-parametric approach based on Kernel Density Estimation (KDE) for tracking convergence of the MC predictive distribution. Rather than measuring uncertainty through a single second-order statistic, KDE allows the estimated empirical density of successive MC samples to be compared. This yields a richer and potentially more sensitive indicator of convergence, capturing changes not only in spread but also in shape, mode concentration, and distributional drift across iterations. Consequently, KDE-based tracking can provide a more salient and stable signal for determining when further sampling becomes unnecessary.

Formally, the KDE-based variant of Adaptive MC Dropout replaces the variance–stability stopping rule with a non-parametric density–stability criterion. In the original procedure, convergence is detected by monitoring the change in the variance estimate:

$$\hat{\sigma}_n^2 \leftarrow \frac{1}{|\mathcal{P}|} \sum_{i=1}^{|\mathcal{P}|} (\hat{y}_i - \hat{\mu})^2, \quad \Delta_n = \hat{\sigma}_n^2 - \hat{\sigma}_{n-1}^2 \tag{3.13}$$

and terminating once $\Delta_n \leqslant \delta$ for $\rho_{\max}$ consecutive steps. In the KDE-based version, this moment-based test is replaced by a distributional stability check. After each MC sample, the output is flattened to $z_n = \mathrm{vec}(\hat{y}_n)$ and used to update a cumulative kernel density estimate on a fixed grid $\{t_g\}_{g=1}^{G}$, which tracks how the entire predictive distribution evolves across iterations rather than only its second moment. Convergence is then assessed through the stability of the KDE peak values:

$$\Delta_n^{\mathrm{KDE}} = \max_j |(\mathrm{peaks}_n)_j - (\mathrm{peaks}_{n-1})_j| \tag{3.14}$$

Table 3.21: Evaluation of predictive uncertainty quality and efficiency across MC-Dropout variants on ID and OOD data.

| | ID (MNIST) | | | | OOD (FMNIST) | | Mixed | |
|---|---|---|---|---|---|---|---|---|
| **Technique** | **Accuracy ↑** | **AUROC ↑** | **Spearman's $\rho$ ↑** | **NLL ↕** | **AUROC ↑** | **Spearman's $\rho$ ↑** | **Time/sample (ms) ↓** | **Forward Passes ↓** |
| MC-Dropout | $98.96 \pm 0.06$ | $0.98 \pm 0.01$ | $0.17 \pm 0.01$ | $0.03 \pm 0.01$ | $0.99 \pm 0.01$ | $0.86 \pm 0.00$ | $8.05 \pm 0.01$ | $1000.00 \pm 0.00$ |
| Single-Shot MC-Dropout | $98.68 \pm 0.09$ | $0.93 \pm 0.01$ | $0.17 \pm 0.02$ | $0.48 \pm 0.03$ | $0.99 \pm 0.02$ | $0.86 \pm 0.03$ | $1.48 \pm 0.02$ | $1.00 \pm 0.00$ |
| Adaptive MC-Dropout | $98.97 \pm 0.06$ | $0.98 \pm 0.00$ | $0.17 \pm 0.01$ | $0.03 \pm 0.00$ | $0.99 \pm 0.01$ | $0.86 \pm 0.01$ | $1.58 \pm 0.04$ | $76.60 \pm 3.93$ |
| Adaptive MC-Dropout (KDE) | $98.96 \pm 0.08$ | $0.98 \pm 0.01$ | $0.17 \pm 0.03$ | $0.03 \pm 0.00$ | $0.99 \pm 0.01$ | $0.86 \pm 0.02$ | $3.17 \pm 0.01$ | $54.20 \pm 0.75$ |

| | ID (CIFAR-10) | | | | OOD (CIFAR-100) | | Mixed | |
|---|---|---|---|---|---|---|---|---|
| **Technique** | **Accuracy ↑** | **AUROC ↑** | **Spearman's $\rho$ ↑** | **NLL ↕** | **AUROC ↑** | **Spearman's $\rho$ ↑** | **Time/sample (ms) ↓** | **Forward Passes ↓** |
| MC-Dropout | $61.00 \pm 1.09$ | $72.61 \pm 0.76$ | $0.38 \pm 0.01$ | $1.12 \pm 0.04$ | $64.73 \pm 1.34$ | $0.25 \pm 0.02$ | $9.26 \pm 0.22$ | $1000.00 \pm 0.00$ |
| Single-Shot MC-Dropout | $46.46 \pm 4.51$ | $64.15 \pm 1.26$ | $0.24 \pm 0.02$ | $1.60 \pm 0.06$ | $55.22 \pm 2.13$ | $0.09 \pm 0.04$ | $2.25 \pm 0.21$ | $1.00 \pm 0.00$ |
| Adaptive MC-Dropout | $60.60 \pm 1.81$ | $72.11 \pm 1.29$ | $0.37 \pm 0.02$ | $1.13 \pm 0.05$ | $64.05 \pm 1.92$ | $0.24 \pm 0.03$ | $2.12 \pm 0.00$ | $105.93 \pm 2.47$ |
| Adaptive MC-Dropout (KDE) | $60.49 \pm 2.07$ | $72.54 \pm 1.17$ | $0.38 \pm 0.02$ | $1.13 \pm 0.06$ | $64.45 \pm 1.87$ | $0.25 \pm 0.03$ | $3.32 \pm 0.04$ | $68.34 \pm 2.56$ |

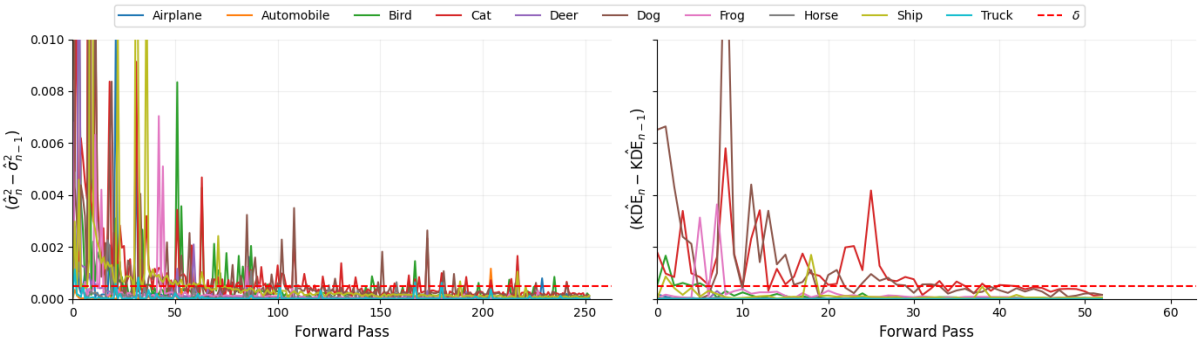



Figure 3.18: Variance convergence behaviour of Adaptive MC Dropout (left) and the KDE-based Adaptive MC Dropout (right). Each curve represents the per-class variance difference $(\hat{\sigma}_n^2 - \hat{\sigma}_{n-1}^2$ for the base method, and $(\text{peaks}_n)_j - (\text{peaks}_{n-1})_j$ for the KDE-based method) across stochastic forward passes, with the red dashed line indicating the convergence threshold $\delta$ used for early termination.

and sampling stops once $\Delta_n^{\text{KDE}} \leqslant \delta$ for $\rho_{\max}$ consecutive steps. This KDE peak criterion provides a finer-grained and more responsive indicator of convergence, capturing changes in distributional shape (e.g., sharpening, shifting, or splitting of modes) that variance alone may fail to detect, while leaving the overall Adaptive MC Dropout procedure unchanged.

To ensure comparability, we retain the same experimental configuration and hyperparameters as in Section 3.3.1.1. The full KDE-based Adaptive MC Dropout algorithm can be found in Appendix A.3. Results are reported in Table 3.21.

We compare the KDE-based Adaptive MC-Dropout variant to the previously analysed MC-Dropout, Single-Shot MC-Dropout and variance-based Adaptive MC-Dropout. Across both MNIST→Fashion-MNIST and CIFAR-10→CIFAR-100 dataset pairings, the KDE variant preserves essentially the same predictive and uncertainty quality as standard Adaptive MC-Dropout, and thus the standard MC Dropout. This indicates that replacing the variance-based stopping rule with KDE peak stability does not degrade the saliency or calibration of the resulting uncertainty estimates, even on the more complex datasets.

The main differences arise in efficiency. In terms of forward passes, the KDE variant sub-

stantially reduces the number of stochastic samples required before declaring convergence: on MNIST→Fashion-MNIST, the average forward passes drop from $76.60 \pm 3.93$ to $54.20 \pm 0.75$, and on CIFAR-10→CIFAR-100 from $105.93 \pm 2.47$ to $68.34 \pm 2.56$. This behaviour is clearly reflected in Figure 3.18, where the KDE-based method exhibits a much sharper and more decisive drop towards the convergence threshold compared to the noisier and slower-to-stabilise variance differences of standard Adaptive MC Dropout. In the visualised example, the variance-based method converges after $\approx 252$ forward passes, whilst the KDE-based method only takes $\approx 53$. However, this benefit comes at the cost of a heavier convergence criterion: computing and updating the KDE is more expensive per forward pass than tracking a scalar variance. On the MNIST→Fashion-MNIST pairing, Adaptive MC Dropout take approximately $1.58\text{ms} \pm 0.04$ per sample while the KDE variant takes $3.17\text{ms} \pm 0.01$ per sample. Whilst this is substantially faster than the standard MC Dropout method, it is more than twice as slow compared to the variance-based Adaptive MC Dropout method.

Overall, KDE-based Adaptive MC-Dropout offers comparable uncertainty quality and allows for early stopping during the MC Dropout process, but the extra density-estimation overhead makes it a more computationally involved option, trading off speed for early termination saliency.

Taken together, the results presented throughout Section 3.3 demonstrate that empirically, MC-CP validates the three hypotheses set out at the start of this evaluation. First, the Adaptive MC Dropout method reliably improves sampling efficiency by terminating early on confident inputs without compromising predictive accuracy, which we showed in Section 3.3.1.1. Second, across both classification and regression tasks, MC-CP consistently maintains the $1-\alpha$ coverage guarantee of split conformal prediction while producing smaller and more informative prediction sets than existing baselines, which we showed in Section 3.2. Third, the method exhibits robust behaviour under a wide range of challenging conditions, including distributional and covariate shift, limited calibration data, and severe class imbalance, highlighting both the strengths and limitations of marginal validity and motivating subsequent refinements, which we showed in Section 3.3.3, and 3.3.4, respectively. Extensions incorporating class-aware regularisation, risk-sensitive calibration, and improved convergence tracking further demonstrate that MC-CP can be adapted to meet the requirements of downstream tasks and real-world issues while preserving its baseline empirical results. Overall, Section 3.3 shows that MC-CP forms a calibrated, flexible, and practically scalable UQ framework, laying the groundwork for its downstream application in safe RL, which we will explore in Section 3.4.

## 3.4 Safe RL Using MC-CP

The previous sections evaluated MC-CP in isolation, focusing on its computational efficiency, coverage performance, and robustness under varying distributional/covariate shifts. Although these results demonstrate that MC-CP can produce calibrated and compact prediction sets/intervals across a range of tasks, its practical value ultimately depends on how these uncertainty-aware predictions interact with a downstream decision-making component. In safety-critical systems, perception rarely constitutes the terminal objective; rather, it forms the first stage of a sequen-

tial pipeline in which an autonomous agent must interpret sensory information, reason about its reliability, and select actions that achieve long-term goals while avoiding hazardous outcomes. Consequently, it is essential to examine how uncertainty-aware perception influences online RL, where the agent continually updates its behaviour through interaction with a dynamic environment.

Prior approaches to uncertainty-aware navigation have sought to enrich individual components of the autonomy stack but have typically treated perception and control as methodologically separate. UQ techniques (discussed in more detail in Section 2.2.1) such as Bayesian neural networks [57] and MC Dropout [66] aim to represent epistemic ambiguity within the perception module. These uncertainty estimates have been propagated into mapping and planning pipelines, for example by encoding per-cell confidence scores in occupancy grids, to encourage planners to prefer routes that appear reliable under the model's uncertainty [244–246]. While such methods reduce overconfident decisions, they remain fundamentally probabilistic: calibrated probabilities or variance estimates serve as useful heuristics but do not provide finite-sample guarantees, leaving residual risk that critical hazards may still be missed in practice [53]. CP offers a complementary line of work by providing distribution-free, finite-sample guarantees for perceptual tasks such as semantic segmentation and object detection [116, 247–249]. Beyond perception, conformal decision theory extends these guarantees to the action-selection layer [250], calibrating policy behaviour under imperfect predictions.

In online RL, the coupling between perception and control is particularly pronounced. The agent does not merely rely on a fixed dataset of observations but repeatedly encounters new sensory inputs whose quality may fluctuate due to noise, environmental complexity, and domain shift. Any imperfections in the perception module therefore propagate into the policy's decision-making process, potentially altering the trajectory distribution, the experienced rewards, and ultimately the learned policy itself. A theoretically well-calibrated perception model does not automatically guarantee improved behaviour unless the policy can meaningfully incorporate its uncertainty estimates into the control loop. This raises a fundamental question: to what extent can calibrated prediction sets mitigate the adverse effects of perceptual uncertainty during online learning, and how does this uncertainty shape the agent's exploration, safety, and performance over time?

Addressing this question requires shifting from an isolated assessment of perceptual uncertainty to a systems-level perspective. The role of MC-CP is no longer confined to quantifying prediction reliability; rather, its outputs become part of the agent's state representation, influencing action selection and subsequent learning updates. This perception-to-policy dependency introduces a bidirectional relationship: the policy relies on uncertainty-aware perception to guide safe exploration, while the data distribution encountered by the agent is itself a product of the policy's past decisions. Understanding this interaction is therefore critical for determining whether MC-CP merely improves perceptual calibration or whether it substantively enhances the safety and stability of the overall autonomous system.

We propose **Co**nformal **P**erception-to-**Po**licy **l**earning (COPPOL), a unified framework that

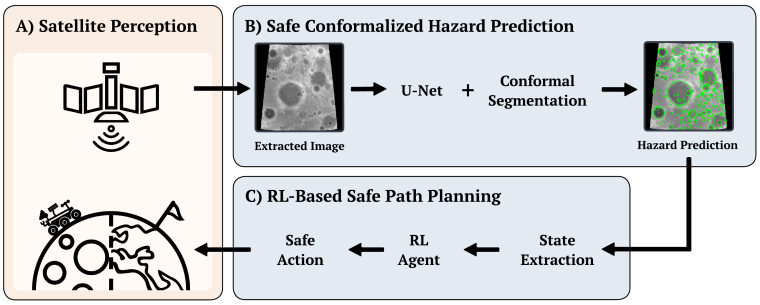



Figure 3.19: High-level overview of COPPOL, our conformal-driven perception-to-policy learning approach for safe navigation over risky terrain. COPPOL combines **(A)** real-world image capture via satellites of the rover's surrounding terrain, **(B)** U-Net-based segmentation of hazards with uncertainty calibration through conformal prediction, and **(C)** an RL agent to plan a safe trajectory to the goal. Orange modules represent the environment; blue modules are learned components of our approach.

incorporates MC-CP into the entire autonomy pipeline. COPPOL couples MC-CP–calibrated hazard perception with RL, transforming raw sensory data into set-valued, risk-aware representations that explicitly bound missed hazards under distribution-free, finite-sample guarantees. By exposing these calibrated hazard maps to the policy, COPPOL enables the agent to reason over spatially grounded uncertainty rather than uncalibrated probabilities, aligning perception and control under a common statistical risk model. This perception-to-policy integration addresses the long-standing disconnect between uncertainty-aware perception and downstream decision-making, providing a principled mechanism for propagating uncertainty through the control loop and enabling safer, more reliable behaviour in online RL settings.

In the remainder of this section, we formalise the COPPOL framework. First, we provide a high-level overview to illustrate the perception-to-policy pipeline in Figure 3.19. We then formally detail the interaction between the perception and policy in Section 3.4.1, before finally performing a thorough experimental evaluation of the framework in Section 3.4.2 to assess the impact of MC-CP-driven perception on downstream policy learning and safety.

### 3.4.1 Perception-to-policy Framework

In this section, we formalise COPPOL by lifting MC-CP from a stand-alone UQ method to the front-end of a perception-to-policy pipeline. At a high level, COPPOL is a conformal-driven perception-to-policy learning approach that converts raw imagery into calibrated hazard maps for downstream navigation. COPPOL comprises (A): real-world imagery captured via satellites of a rover's local terrain, (B): a convolutional U-Net that segments hazardous terrain at the pixel level with uncertainty calibration through conformal prediction, and (C): an RL agent that plans safe trajectories across the risky terrain. It should be noted that we use space navigation throughout the remaining sections as a motivating example, but COPPOL can comfortably generalise to any perception-to-policy pipeline. Unlike traditional pipelines, where the RL agent consumes raw or thresholded segmentation outputs, our framework treats

calibrated hazard regions as integral information, allowing the policy to reason over safety-aware spatial representations rather than uncalibrated predictions. This decoupling of perception and planning under explicit risk guarantees enables principled navigation in environments where perception errors can have harmful consequences.

We formalise hazard perception in COPPOL as set-valued, distribution-free inference over the spatial domain of an observation. This reflects the fact that the RL agent ultimately needs a region of space to treat as unsafe, and that our primary safety requirement is that this predicted region contains the true hazardous area with high probability, without relying on strong assumptions about the underlying data distribution. Each labelled observation $(x_i, y_i)$ consists of an image $x_i \in \mathbb{R}^{H\times W\times C}$ accompanied by a binary hazard mask $y_i \in \{0,1\}^{H\times W}$ that marks which pixels correspond to unsafe terrain. The image is defined over the pixel lattice $\Omega = \{1,\dots,H\} \times \{1,\dots,W\}$, where the true hazardous set for the $i$-th image is $\mathfrak{H}_i = \{(m,n) \in \Omega : y_{m,n} = 1\}$. The goal of the perception module is not merely to assign a label to each pixel independently, but to recover a set-valued estimate of $\mathfrak{H}_i$ that we can treat as the predicted hazardous region in the scene.

Hazard scores are produced by a convolutional U-Net [251] $f_\theta : \mathbb{R}^{H\times W\times C} \to [0,1]^{H\times W}$, trained on a dataset $\mathcal{D}_{\text{train}}$ of labelled images using a standard binary cross-entropy loss. Given an input image $x$, the network outputs a per-pixel hazard probability map $p_\theta(x)$, where each entry $p_\theta(x)_{m,n}$ represents the model's belief that pixel $(m,n)$ is hazardous. This step converts raw imagery into a dense field of probabilistic hazard scores, but by itself it does not specify which pixels should ultimately be treated as unsafe. In the baseline setting, a single probability threshold $\lambda \in [0,1]$ is chosen a priori (for instance, by validation tuning), and the final binary hazard mask is obtained by declaring all pixels with $p_\theta(x)_{m,n} \geqslant \lambda$ as hazardous. This setup is the standard approach for pixel-wise semantic segmentation. While simple, this approach effectively fixes one operating point on the precision–recall curve and offers no formal control over the rate at which true hazards are missed, which is precisely the safety-critical quantity of interest.

To move beyond this ad-hoc choice and obtain hazard maps that are calibrated to a desired risk level, we expose the full family of candidate regions induced by the network's scores and treat $\lambda$ as a data-driven parameter. Conformal calibration then selects the smallest threshold whose induced hazard regions satisfy a prescribed bound on missed hazards. Let $p_\theta(x) \in [0,1]^{H\times W}$ denote the hazard score map produced by the Adaptive MC Dropout module (i.e., the mean of the $N_{\text{MC}}$ stochastic forward passes; see Section 3.2.1). For any $\lambda \in [0,1]$, we define the set of pixels whose hazard score exceeds the corresponding cut-off:

$$\mathfrak{C}_\lambda(x) = \{(m,n) \in \Omega : p_\theta(x)_{(m,n)} \geqslant 1-\lambda\} \tag{3.15}$$

and the associated binary mask:

$$\mathcal{M}_\lambda(x)_{(m,n)} = \mathbf{1}\{(m,n) \in \mathfrak{C}_\lambda(x)\} \tag{3.16}$$

As $\lambda$ increases, the set $\mathfrak{C}_\lambda(x)$ expands monotonically, progressively covering more of the terrain. Small values of $\lambda$ typically yield compact, high-precision hazard regions that risk missing parts of the true hazardous set, whereas larger values of $\lambda$ inflate the predicted region and act as a safety buffer at the cost of additional false positives. In the context of risky-terrain perception and navigation, this nested family $\{\mathfrak{C}_\lambda\}_{\lambda\in[0,1]}$ therefore encodes an explicit trade-off between efficiency and safety.

MC-CP turns the intuitive safety–efficiency trade-off into a CP problem by treating the false negative rate (missed hazards) as the risk to control (risk control like we explored in Section 3.3.6 but for semantic segmentation), and then using calibration data to pick the least conservative threshold that keeps that risk below a desired level. Let a calibration dataset $\mathcal{D}_{\text{cal}} = \{(x_i, y_i)\}_{i=1}^{|\mathcal{D}_{\text{cal}}|}$, where both datasets satisfy $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{cal}} \subset X \times Y$ and are from the same underlying data distribution. Using the corresponding ground truth hazardous sets $\mathfrak{H}_i \subseteq \Omega$, we measure for each candidate $\lambda \in [0, 1]$ how many hazardous pixels are not captured by $\mathfrak{C}_\lambda(x_i)$. Concretely, we define the pixel-level false negative rate (FNR) for the $i$-th calibration image as:

$$L_i^{\text{FNR}}(\lambda) = 1 - \frac{|\mathfrak{H}_i \cap \mathfrak{C}_\lambda(x_i)|}{|\mathfrak{H}_i|} \tag{3.17}$$

with the convention $L_i^{\text{FNR}}(\lambda) = 0$ whenever $|\mathfrak{H}_i| = 0$. This quantity measures the fraction of truly hazardous pixels that the prediction fails to cover: a value of zero indicates perfect hazard coverage for image $i$, whereas values closer to one indicate that most hazards were missed. This will act as our risk function throughout, but can be swapped out for alternative choices (e.g., weighted false negatives or instance-level criteria) without changing the surrounding framework, simply by replacing the FNR term in the subsequent definitions. For a given threshold $\lambda$, we then aggregate these scores across the calibration set to obtain the empirical FNR risk:

$$\hat{\mathcal{R}}_{|\mathcal{D}_{\text{cal}}|}(\lambda) = \frac{1}{|\mathcal{D}_{\text{cal}}|} \sum_{i=1}^{|\mathcal{D}_{\text{cal}}|} L_i^{\text{FNR}}(\lambda) \tag{3.18}$$

The conformal calibration step selects the least conservative threshold that still satisfies the target risk level $\alpha$. Following the conformal risk control formulation, we choose:

$$\hat{\lambda} = \inf\left\{\lambda \in [0, 1] : \frac{|\mathcal{D}_{\text{cal}}|}{|\mathcal{D}_{\text{cal}}| + 1}\hat{\mathcal{R}}_{|\mathcal{D}_{\text{cal}}|}(\lambda) + \frac{1}{|\mathcal{D}_{\text{cal}}| + 1} \leqslant \alpha\right\} \tag{3.19}$$

Intuitively, the term $\hat{\mathcal{R}}_{|\mathcal{D}_{\text{cal}}|}(\lambda)$ estimates the average FNR at threshold $\lambda$, and the additional factor $\frac{1}{|\mathcal{D}_{\text{cal}}|+1}$ accounts for finite-sample uncertainty due to using a finite calibration set. The optimisation over $\lambda$ searches along the nested family $\{\mathfrak{C}_\lambda\}$, increasing $\lambda$ only as far as needed to ensure that the empirical FNR, plus this finite-sample correction, remains below the desired risk level $\alpha$. The resulting $\hat{\lambda}$ therefore corresponds to the smallest threshold that is compatible with the target bound on missed hazards.

Once $\hat{\lambda}$ has been identified for the user-chosen desired risk level $\alpha$, it is fixed and applied uniformly at test time. Given a new image $x$, COPPOL applies $\hat{\lambda}$ to the probability map

$p_\theta(x)$ to obtain a binary, conformally calibrated hazard mask $\mathcal{M}_{\hat{\lambda}}(x)$. Under the standard exchangeability assumption between calibration and deployment data, this procedure guarantees that the expected pixel-level FNR on unseen images is controlled by $\alpha$:

$$\mathbb{E}\left[L^{\mathrm{FNR}}_{|\mathcal{D}_{\mathrm{cal}}|+1}(\hat{\lambda})\right] \leqslant \alpha \tag{3.20}$$

In other words, on average across future scenes, the fraction of missed hazardous pixels will not exceed the prescribed risk level. From a navigation perspective, the calibrated mask $\mathcal{M}_{\hat{\lambda}}(x)$ can therefore be interpreted as a statistically grounded safety envelope: it expands around predicted hazards just enough to satisfy the desired coverage guarantee, acting as a principled buffer in regions where the segmentation model is uncertain.

Having instantiated a conformally calibrated hazard predictor, COPPOL embeds it into an online perception–action loop rather than treating it as a one-off preprocessing step. At each decision step, the robot/rover acquires a local overhead terrain image (e.g., from an onboard camera or satellite-derived patch). The MC-CP segmentation module processes this image and returns a calibrated hazard mask that marks pixels predicted to be unsafe under a finite-sample bound on missed hazards. The RL agent then acts as a high-level path planner: it takes as input both the current image and its associated hazard mask, selects a motion command (e.g., move to the next waypoint), and this command is executed by the underlying robotics stack using conventional low-level control. Executing the action updates the rover's pose and may also change the visual scene (for instance, because of a new viewpoint, illumination changes, or partial occlusions such as dust or cloud cover), leading to a new image and a newly calibrated hazard mask at the next time step. From the agent's perspective, conformal perception therefore provides a time-varying, risk-aware view of the world that directly informs its control decisions, closing the loop between perception and safe navigation.

From this perspective, it is natural to cast navigation as an MDP defined on the pixel lattice $\Omega$. The rover's pose at time $t$ is the lattice coordinate $q_t = (m_t, n_t) \in \Omega$ with initial coordinate $q_0 \in \Omega$ and the target/goal location $q_{\mathrm{goal}} \in \Omega$. The action space consists of the four cardinal moves on the grid, each corresponding to a fixed displacement:

$$A = \{\uparrow, \rightarrow, \downarrow, \leftarrow\}, \quad \begin{array}{ll} \Delta_\uparrow = (-1, 0), & \Delta_\rightarrow = (0, 1), \\ \Delta_\downarrow = (1, 0), & \Delta_\leftarrow = (0, -1) \end{array} \tag{3.21}$$

At each timestep, the agent chooses an action $a_t \in A$, and the pose is updated deterministically as $q_{t+1} = q_t + \Delta_{a_t}$. In a realistic deployment, actions that would move the rover beyond the current image footprint would trigger a recentering of the observation window (for example, by requesting an adjacent satellite tile so that the overhead view keeps covering the robot). For ease of implementation and to remain consistent with the fixed-size image patches provided by empirical datasets, we do not model this recentering process explicitly. Instead, we assume a static planning image and treat actions that would leave $\Omega$ as invalid (or equivalently, clip them to the nearest valid boundary pixel). An episode terminates either when the agent reaches the

goal $q_{t+1} = q_{\text{goal}}$ or when a fixed horizon $T$ is exceeded. In a realistic deployment, navigation would not stop at a single image patch: once the rover reaches its current goal, a satellite or onboard sensor could be retasked to image the next region of interest while the rover carries out low-level tasks such as sample collection or close-up imaging. High-level navigation would then resume over the newly acquired terrain, leading to an effectively infinite-horizon problem defined over a sequence of overlapping terrain segments. However, the datasets used in this work provide isolated, fixed-size image patches with a single start–goal pair, so we restrict ourselves to a finite-horizon MDP with episodic termination.

The observable state $s_t \in S$ presented to the agent can take different forms, depending on the desired trade-off between spatial richness and sample efficiency. In an image view, the state may consist of the full planning image (the terrain with the conformal mask overlaid), optionally augmented with a representation of the current pose and goal location. In a more compact vector view, the state can instead encode a set of features derived from $q_t$, $q_{\text{goal}}$, and $\mathcal{M}_{\hat{\lambda}}(x)$, such as the Manhattan offset to the goal, the distance to the nearest predicted hazard boundary, a binary inside-hazard indicator, and action-legality flags. In both cases, the essential property is that the state encodes the same conformally calibrated world: the RL agent always reasons over hazard information that inherits the finite-sample coverage guarantees of MC-CP, whether this information is presented as pixels or as structured features.

To encourage efficient yet safe navigation, we design a reward function that combines goal-directed progress with penalties for entering conformally predicted hazards. Let $\text{dist}_t := ||q_t - q_{\text{goal}}||_1$ denote the Manhattan distance from the agent to the goal at time $t$. We define the per-step reward $r_t$ as:

$$r_t = \begin{cases} -1 - \epsilon_{\text{step}}, & \text{if } q_{t+1} \in \mathcal{M}_{\hat{\lambda}}(x), \\ \operatorname{sgn}\big(\text{dist}_t - \text{dist}_{t+1}\big) - \epsilon_{\text{step}} + \beta_{\text{goal}}\, \mathbf{1}\{q_{t+1} = q_{\text{goal}}\}, & \text{otherwise}, \end{cases} \tag{3.22}$$

where $\epsilon_{\text{step}} \geqslant 0$ is an optional small per-step cost, $\beta_{\text{goal}} \gg 0$ is a bonus for reaching the goal, and sgn is the sign function returning 1 if its argument is positive, -1 if negative, and 0 otherwise. The first case assigns a strong penalty whenever the agent enters or remains in pixels labelled as hazardous by the calibrated mask, thereby translating MC-CP's coverage guarantees into explicit costs for predicted hazards. The second case rewards $A^*$-like progress toward the goal, penalises unnecessarily long paths, and provides a large positive reward upon successful arrival at the goal.

COPPOL is agnostic to the choice of discrete-control algorithm: value-based methods that approximate a state–action value function $Q^\pi(s, a)$ and act via $\arg\max_a Q^\pi(s_t, a)$, as well as actor–critic methods that directly parameterise $\pi_\theta(a|s)$, can be used without modification.

Taken together, these components fully specify COPPOL as an online RL framework in which MC-CP–calibrated hazard maps shape both the state representation and the reward structure. The agent therefore learns policies over a risk-aware abstraction of the environment that inherits finite-sample guarantees on missed hazards from the perception module. In the next subsection,

we instantiate this formulation on hazardous-terrain benchmarks and empirically evaluate how conformalised perception affects navigation performance and safety relative to non-conformal baselines.

### 3.4.2 Experimental Results

This section empirically evaluates COPPOL instantiated with MC-CP in the online navigation setting. The following experimental analysis is designed to assess whether conformally calibrated hazard perception can translate into tangible benefits for both perception and control when compared to deterministic and probabilistic baselines. In particular, we examine whether MC-CP–driven hazard maps improve coverage of unsafe terrain, whether these improvements propagate to safer trajectories without degrading task performance, and whether the resulting perception-to-policy pipeline remains robust under degraded sensing and distributional shift. To this end, the following hypotheses are investigated.

The first hypothesis is that MC-CP–based hazard perception substantially increases coverage of true hazards relative to non-conformal baselines, while maintaining reasonable precision. By explicitly controlling the pixel-level false-negative rate during calibration, MC-CP is expected to recover a larger fraction of hazardous regions at both pixel and instance levels, thereby providing a more faithful and conservative representation of unsafe terrain. Crucially, it should achieve this without collapsing into trivially conservative masks that flood the scene with false hazards or label visually implausible pixels as unsafe.

The second hypothesis is that exposing these conformally calibrated hazard maps to the RL agent enables safer navigation without compromising task efficiency. When the calibrated mask is used to shape the state and reward structure, the agent should learn policies that achieve similar reward and success rates to baseline agents, but spend less time in hazardous regions and maintain greater clearance from unsafe terrain.

The final hypothesis is that these safety benefits persist under imperfect perception and distributional shift (as seen with MC-CP in Section 3.3.3). Because MC-CP enforces finite-sample guarantees on missed hazards at the perception level, the induced cost field should continue to steer the policy away from unsafe regions even when observations are corrupted by noise or drawn from shifted terrain distributions.

We evaluate COPPOL on two satellite-derived hazard segmentation benchmarks: the DeepMoon crater detection dataset [252] and the YTU-WaterNet water segmentation dataset [253]. These datasets are representative of visually distinct but safety-critical terrain types: craters in planetary exploration and water bodies in terrestrial navigation. Both environments pose different perceptual challenges, i.e., subtle, low-contrast features in lunar imagery versus large, heterogeneous structures in aerial water segmentation, making them well-suited to assess COPPOL under varied conditions. For each dataset, four terrain patches and start-goal configurations provide varied hazard layouts and routing challenges, as illustrated in Appendix A.4. These environments are suitable for assessing whether improved hazard perception translates into safer navigation under a common RL setup. Safety is encouraged through the hazard penalty in the

shaped reward, rather than a separate cost constraint, and evaluated using clearance from and time spent within ground-truth hazards alongside task success. This distinguishes empirical improvements in navigation safety from satisfaction of a prescribed safety budget.

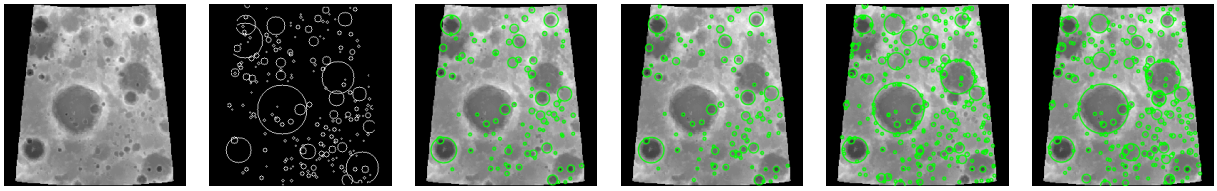
We instantiate COPPOL with two conformal prediction algorithms: CRC [116] and MC-CP. This design not only evaluates COPPOL as a general perception-to-policy framework, but also isolates the specific impact of the underlying conformal predictor on both hazard perception and downstream navigation. Moreover, this diversity in CP instances demonstrates that our perception-to-policy pipeline is agnostic to the selected CP method and can accommodate different conformal segmentation approaches. For completeness, we also compared our approach against non-conformal baselines, i.e., a standard U-Net segmentation model (as discussed at the start of Section 3.4.1) and a U-Net with MC-Dropout [66]. Taken together, these baselines encompass deterministic, probabilistic, and conformal approaches, allowing for a controlled assessment of how distribution-free calibration affects both hazard perception and safe navigation. Full details of the experimental setup and dataset information can be found in Appendix A.6. COPPOL uses a Deep Q-Network (DQN) agent, trained for 200,000 timesteps per run across five runs, with a maximum episode length of 1,000 steps. Training uses a replay buffer of 200,000 transitions, mini-batches of 64, a discount factor of $\gamma = 0.95$, and updates at every timestep. The learning rate decreases linearly from $10^{-3}$ to $10^{-7}$ over the first 90% of training, while exploration is annealed over the first 75% to a final $\epsilon = 0.05$. All methods employ identical network architectures and training procedures for a fair comparison.

Firstly, we evaluate whether MC-CP–based conformal segmentation improves hazard perception relative to deterministic and probabilistic baselines. Performance is assessed on the DeepMoon and YTU-WaterNet benchmarks in terms of pixel-wise and instance-level metrics, including precision, coverage (recall), F1 (Dice) score, and IoU. These metrics are chosen to capture complementary aspects of hazard perception quality: precision penalises spurious hazards, coverage quantifies how completely true hazardous regions are recovered, F1 balances these two effects, and IoU measures spatial overlap between predictions and ground truth. By reporting results at both pixel and instance level, we assess not only local segmentation accuracy but also how reliably each method identifies entire hazardous structures (e.g., craters or water bodies). The results of this perception study are summarised in Table 3.22, with qualitative examples shown in Figure 3.20.

Across both datasets, the deterministic and MC-Dropout baselines exhibit a systematic tendency to under-cover hazards. On the DeepMoon dataset, the Baseline model attains pixel-level precision of $44.59 \pm 21.68$ but only $10.42 \pm 8.77$ coverage, with F1 and IoU of $15.71 \pm 11.91$ and $8.98 \pm 7.18$, respectively. This reflects the intrinsic difficulty of crater segmentation on orbital imagery, where craters often appear as only slightly darker grey-scale depressions with highly variable scale, from tiny holes to large basins, so the baseline model struggles to distinguish them from normal texture/brightness variations and to capture their full extent, a challenge also noted in prior work on lunar crater detection [254]. MC-Dropout slightly sharpens predictions, achieving $48.26 \pm 28.19$, but further reduces coverage to $6.15 \pm 6.01$, yielding similarly low F1 and IoU scores. This behaviour is consistent with the stochastic sampling effect of MC-

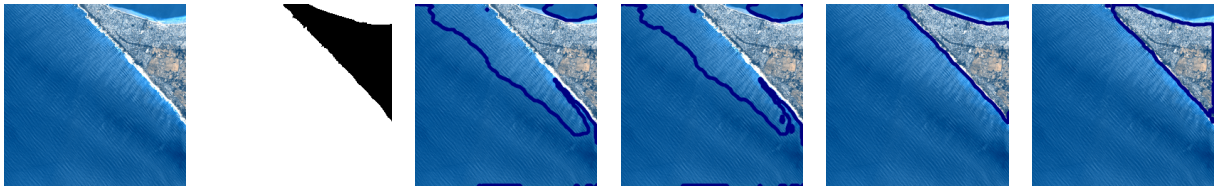
Table 3.22: Evaluation of hazard segmentation on DeepMoon and YTU-WaterNet. Metrics are reported at the pixel and instance level for all methods.

| | | Pixel-level Metrics | | | | Instance-level Metrics | | |
|---|---|---|---|---|---|---|---|---|
| **Dataset** | **Technique** | **Precision** | **Coverage** | **F1 (Dice)** | **IoU** | **Precision** | **Coverage** | **F1 Score** |
| DeepMoon | Baseline | $44.59 \pm 21.68$ | $10.42 \pm 8.77$ | $15.71 \pm 11.91$ | $8.98 \pm 7.18$ | $64.30 \pm 20.76$ | $60.49 \pm 16.95$ | $59.59 \pm 14.28$ |
| | MC | $48.26 \pm 28.19$ | $6.15 \pm 6.01$ | $10.38 \pm 9.56$ | $5.75 \pm 5.49$ | $67.69 \pm 20.38$ | $53.95 \pm 19.60$ | $56.29 \pm 14.83$ |
| | CRC | $26.63 \pm 8.08$ | $42.97 \pm 18.34$ | $30.38 \pm 10.76$ | $18.35 \pm 7.02$ | $43.53 \pm 16.14$ | $72.14 \pm 13.93$ | $52.43 \pm 11.68$ |
| | MC-CP | $25.72 \pm 4.92$ | $69.29 \pm 16.16$ | $25.36 \pm 7.34$ | $14.72 \pm 4.70$ | $35.67 \pm 6.35$ | $79.66 \pm 10.40$ | $45.54 \pm 8.60$ |
| YTU-WaterNet | Baseline | $95.88 \pm 16.53$ | $94.78 \pm 15.91$ | $94.90 \pm 16.04$ | $92.93 \pm 17.54$ | | | |
| | MC | $95.86 \pm 16.60$ | $94.84 \pm 15.72$ | $94.94 \pm 16.00$ | $92.97 \pm 17.44$ | | | |
| | CRC | $86.04 \pm 19.91$ | $98.71 \pm 6.72$ | $90.21 \pm 15.94$ | $85.13 \pm 20.59$ | | | |
| | MC-CP | $72.88 \pm 27.22$ | $99.69 \pm 1.14$ | $80.77 \pm 22.39$ | $72.80 \pm 27.22$ | | | |



Input Ground Truth Baseline MC CRC MC-CP

(a) DeepMoon Crater Detection



Input Ground Truth Baseline MC CRC MC-CP

(b) YTU-WaterNet Water Detection

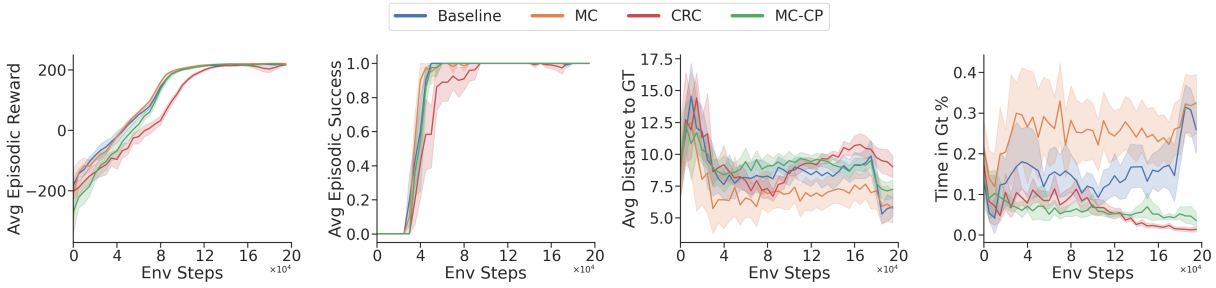
Figure 3.20: Comparison of hazard predictions for (a) DeepMoon Crater and (b) YTU-WaterNet Water hazards. Each row shows the input image captured via satellite, ground truth mask of the hazards, and the output of different methods: Baseline U-Net, MC, CRC, and MC-CP.

Dropout: averaging multiple noisy forward passes concentrates probability mass on pixels that are consistently high-scoring and suppresses ambiguous regions, effectively shrinking the predicted hazard set. As a result, the model becomes more selective (higher precision) but misses even more truly hazardous pixels, reducing coverage. At the instance-level of the DeepMoon dataset, the Baseline and MC methods detect only around $60.49 \pm 16.95$ and $53.95 \pm 19.60$ percent of crater instances, respectively, despite relatively high instance-level precision. In other words, when these models do predict a hazard, they are often correct, but they simply miss a large fraction of hazardous pixels and even entire hazardous structures. On the YTU-WaterNet dataset, the same methods perform much better, with pixel-level precision and coverage both around the mid-90s (e.g., Baseline precision $95.88 \pm 16.53$, coverage $94.78 \pm 15.91$), indicating that the underlying segmentation task is easier in this domain (most likely due to the stronger colour separation of the hazards, i.e., strong blue water in contrast to green land). Nevertheless, even here, a small but non-negligible fraction of hazardous pixels remains undetected.

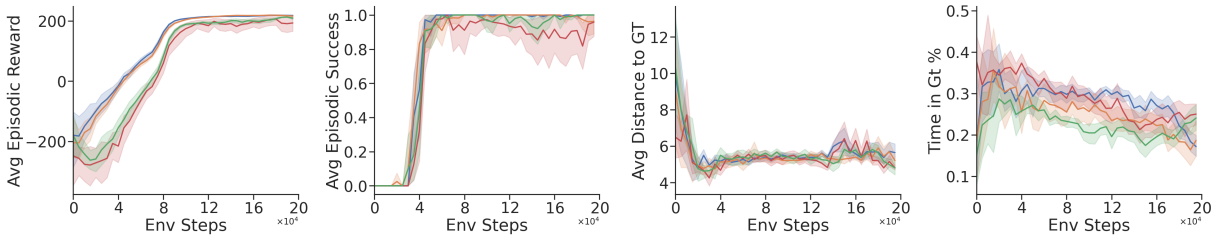
The COPPOL variants (both with CRC or MC-CP) explicitly correct this under-coverage by sacrificing some precision in favour of higher recall, as dictated by the FNR-based calibration objective. On the DeepMoon dataset, CRC boosts pixel-level coverage from $10.42 \pm 8.77$, compared to the baseline, to $42.97 \pm 18.34$, and MC-CP further increases it to $69.29 \pm 16.16$. Instance-level coverage shows a similar pattern, rising from $60.49 \pm 16.95$ to $72.14 \pm 13.93$ for the CRC variant and $79.66 \pm 10.40$ for the MC-CP variant. These gains come with reduced precision (e.g., MC-CP pixel precision $25.72 \pm 4.92$ and instance precision $35.67 \pm 6.35$), and consequently lower IoU, reflecting the fact that the conformal masks expand beyond the tight rims produced by the baseline network. This behaviour is expected: by optimising for a bound on the false negative rate, MC-CP inflates predicted hazard regions to ensure that true hazardous pixels are rarely missed, even if this means marking additional safe pixels as hazardous. For safety-critical navigation, this is the desired direction of approximation: it is preferable to slightly overestimate the extent of hazards than to leave dangerous regions undetected. A similar trend is observed on the YTU-WaterNet dataset, where the Baseline already achieves strong performance. The COPPOL variants still provide measurable improvements in coverage. CRC increases pixel-level coverage from $94.78 \pm 15.91$ to $98.71 \pm 6.72$, and MC-CP pushes it further to $99.69 \pm 1.14$, effectively saturating recall. As before, this is accompanied by a drop in precision. Importantly, however, these degradations are modest relative to the absolute level of performance, and the conformal masks remain spatially close to the ground truth: they tend to slightly dilate the hazardous regions rather than generating entirely spurious structures. This can be qualitatively observed in Figure 3.20, where on the YTU-WaterNet dataset, MC-CP slightly encroaches onto the land to guarantee hazard coverage, but this slight dilation drops precision somewhat. These results allow us to empirically validate our first hypothesis: MC-CP–based hazard perception substantially increases coverage of true hazards relative to non-conformal baselines, while maintaining reasonable precision. In both datasets, MC-CP achieved the highest coverage, even higher than the CRC variant, whilst not reducing the predicted hazard map to trivial/overinflated solutions.

To assess how conformally calibrated perception affects downstream decision-making, we next evaluate COPPOL in the online RL setting on four DeepMoon environments using the vector-state representation. Performance is measured over training steps in terms of average episodic reward and success rate, capturing task efficiency and learnability, as well as average distance to ground-truth hazards and time spent within ground-truth hazardous regions, capturing safety and coverage. These metrics together indicate whether conformalised perception leads to safer trajectories without degrading overall navigation performance. Results for the DeepMoon dataset are reported in Figure 3.21.

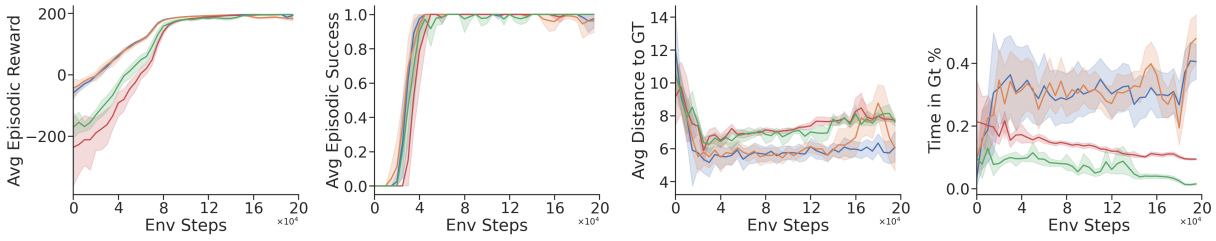
Across all four environments, the reward and success curves show that introducing conformal perception does not hinder the ability of the RL agent to learn effective navigation policies. All methods eventually converge to similar average episodic rewards of $\approx 200$ and near-saturated success rates close to 1.0. Although the COPPOL variants sometimes exhibit a small lag in reward early in training, their convergence speed is comparable to, and occasionally indistinguishable from, that of the Baseline and MC agents. This indicates that the more conservative hazard maps produced by conformal calibration do not significantly slow down learning or pre-



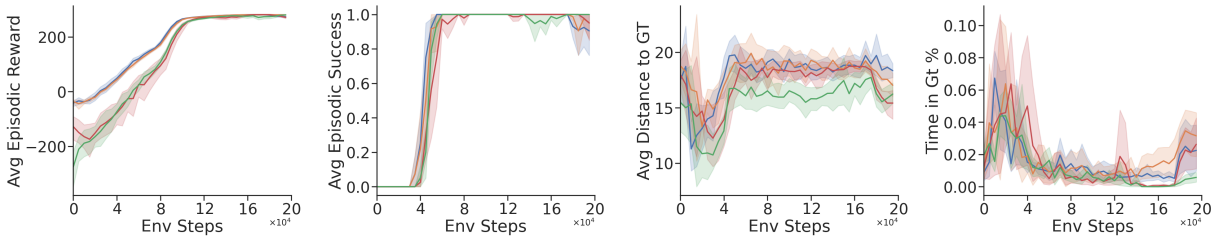

(a) Environment 1 results




(b) Environment 2 results




(c) Environment 3 results




(d) Environment 4 results

Figure 3.21: Performance of comparative methods on four DeepMoon environments (vector view). Curves show average episodic reward, success rate, distance to ground truth hazard, and time spent in ground-truth hazards for Baseline, MC, CRC, and MC-CP (shaded bands indicate variability).

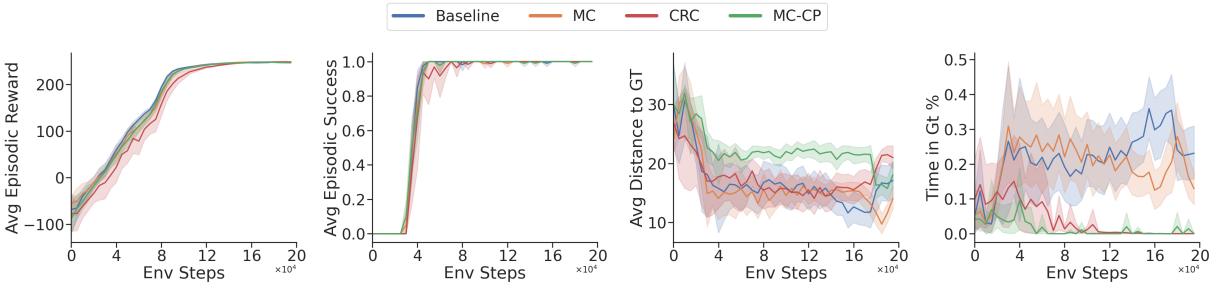
vent the agent from discovering efficient paths to the goal.

The main differences emerge in the safety metrics: average distance to the ground truth craters and average time spent within ground truth craters. In terms of distance to ground-truth hazards, the COPPOL agents consistently maintain larger clearance than the non-conformal baselines in all four environments. After the initial learning phase, CRC and MC-CP stabilise at higher distances to the nearest crater, with reduced variance across training. In contrast, the Baseline and MC agents tend to drift closer to crater rims as training progresses. This effect is particularly clear in Environment 3 (Figure 3.21c), where the Baseline and MC agents average

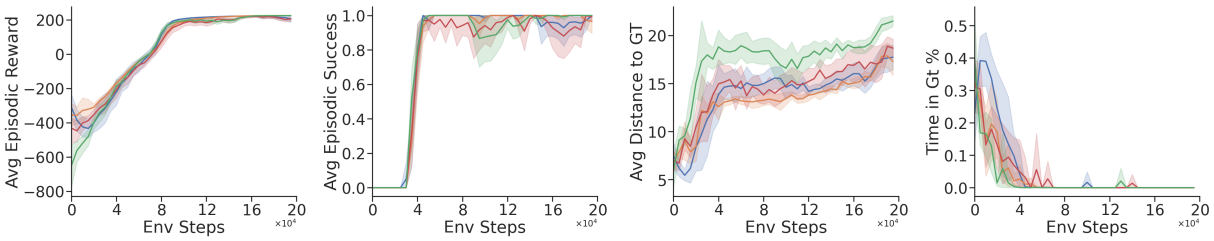
$\approx 6$ distance from the nearest crater, whereas CRC and MC-CP maintain distances closer to $\approx 7-8$ distance. The increased clearance arises because the calibrated masks capture crater rims more completely, and MC-CP in particular expands them with an additional safety buffer, shaping a cost field that more faithfully reflects the true hazardous geometry. In terms of time spent inside ground-truth hazards, it further highlights the safety benefits of COPPOL. In a few environments, the Baseline and MC agents spend a substantial portion of their trajectories within hazardous regions. For example, in Environment 3, the Baseline and MC trajectories spend on average about one-third of their training time in hazardous areas. CRC already reduces this fraction significantly by better aligning predicted hazards with the ground truth; MC-CP goes further, often driving the time-in-hazard percentage towards zero. In the same environment, MC-CP reduces the average fraction of time spent in hazards to approximately $\approx 0.075\%$. These improvements directly reflect the increased hazard coverage of COPPOL: by marking a larger portion of the true crater rims (and a small buffer around them in the case of MC-CP) as unsafe, the induced cost discourages paths that cut through or skim dangerously close to hazardous regions.

Similar trends are observed in the four YTU-WaterNet environments, reported in Figure 3.22. Across all methods, agents rapidly attain near-identical average rewards and success rates, confirming that conformal perception again does not impede learning or final task performance and that the more conservative/slightly inflated predicted hazard regions do not trap the agent into overly cautious behaviour or excessively long paths. As on DeepMoon, CRC and MC-CP consistently maintain greater clearance to ground-truth water regions and reduce the fraction of time spent in hazardous areas relative to the Baseline and MC agents. Taken together with the DeepMoon results, this indicates that the safety benefits of COPPOL with MC-CP are robust across distinct terrain types and environment layouts, allowing us to validate our second hypothesis: conformally calibrated perception yields safer policies while preserving high task efficiency.

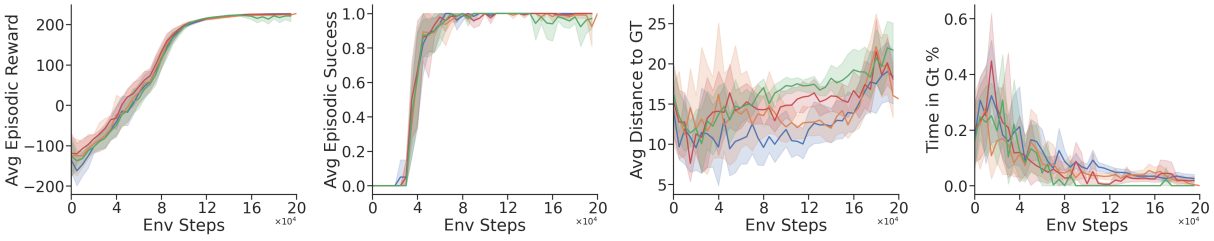
Next, we test whether our conclusions hold when the agent operates directly on the calibrated hazard map, where the full predictive hazard map is provided as the state observation. Results can be visualised in Figure 3.23. As in the vector view, all four methods, Baseline, MC, CRC, and MC-CP, achieve similar asymptotic average reward and near-saturated success rates, indicating that conformal perception does not hinder learning even when the state is high-dimensional and more complex. Convergence is slightly slower and noisier than in the vector view, reflecting the increased state size and the additional computational overhead of processing image inputs at each step, but the final task performance remains comparable across methods. The safety trends mirror those observed with the vector representation. COPPOL variants, CRC and MC-CP, maintain greater average distance to ground-truth hazards and spend less time in hazardous regions than the Baseline and MC agents, with MC-CP typically achieving the lowest hazard exposure. Thus, even when the agent consumes the full calibrated hazard map rather than engineered features, MC-CP still yields policies that are at least as effective in terms of reward and success while exhibiting consistently safer navigation, at the cost of a modest increase in computational load due to the larger state representation.



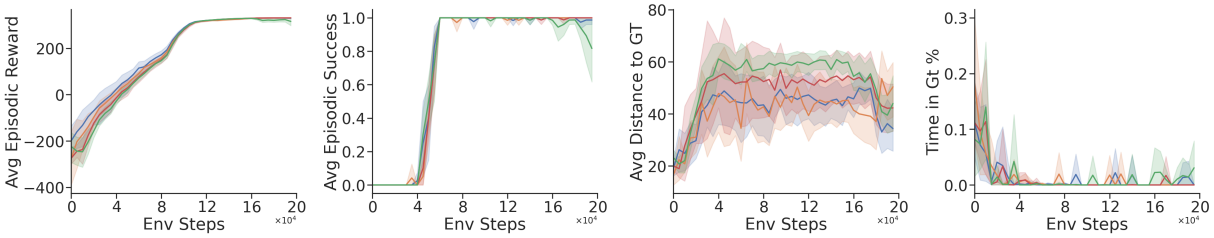

(a) Environment 1 results




(b) Environment 2 results




(c) Environment 3 results




(d) Environment 4 results

Figure 3.22: Performance of comparative methods on four YTU-WaterNet environments (vector view). Curves show average episodic reward, success rate, distance to ground truth hazard, and time spent in ground-truth hazards for Baseline, MC, CRC, and MC-CP (shaded bands indicate variability).

To assess robustness under degraded sensing (e.g., radiation-induced sensor noise in spaceborne cameras or corrupted range estimates derived from noisy satellite imagery), we evaluate a state-wise noisy variant of DeepMoon Environment 1 (Figure 3.21a). Noise is injected at the observation level by perturbing the four distance-derived features (in the vector view) that describe the goal distance and the predicted-hazard distance (and their companion distance features). Details for how this noise is injected can be found in Appendix A.6.2. Results can be observed in Figure 3.24.

Despite the injected noise at every time step $t$, all methods eventually recover high episodic

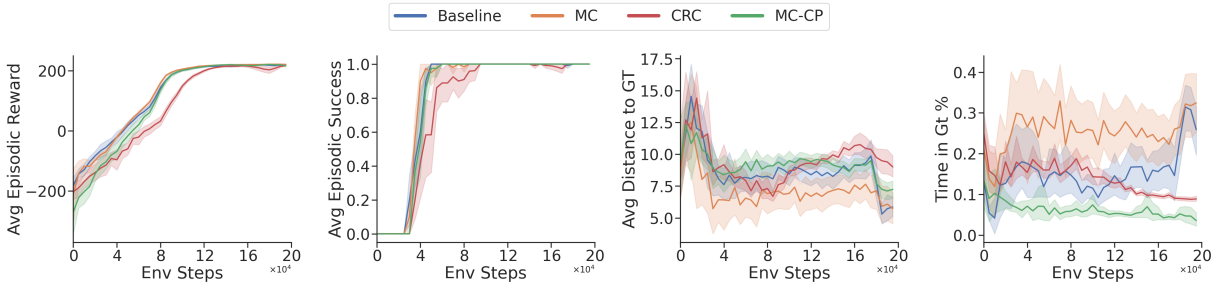



Figure 3.23: Performance of comparative methods on Environment 1 from the DeepMoon environments using the whole predictive hazard map as the state observation (image view). Curves show average episodic reward, success rate, distance to ground truth hazard, and time spent in ground-truth hazards for Baseline, MC, CRC, and MC-CP (shaded bands indicate variability).

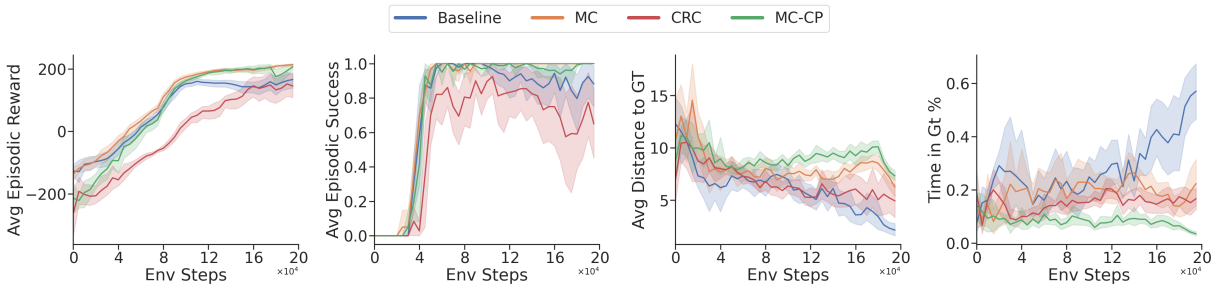



Figure 3.24: Performance of comparative methods on a state-wise noisy variant of Environment 1 from the DeepMoon environments (vector view). Curves show average episodic reward, success rate, distance to ground truth hazard, and time spent in ground-truth hazards for Baseline, MC, CRC, and MC-CP (shaded bands indicate variability).

reward and success rates, although learning is noticeably less stable than in the normal vector view. In particular, the reward curves show that all agents converge to similar final returns, but the Baseline and CRC variants do so more slowly and with larger oscillations than MC and MC-CP. The success-rate plots make this instability even clearer: while MC and MC-CP climb relatively quickly to high success and remain there, the Baseline and CRC agents struggle to maintain a consistently high success rate, exhibiting prolonged plateaus and occasional drops before convergence. The Baseline agent converges to $\approx$ 90% average success rate, and CRC performs worse with an average success rate of $\approx$ 70%.

The safety metrics again favour COPPOL with MC-CP. For the average distance to ground-truth hazards, MC-CP maintains the largest and most stable clearance throughout training (matching the results from the clean vector view environment), whereas CRC and especially the Baseline agent tend to drift closer to crater rims and display higher variance under noise. The time spent in ground-truth hazards shows the most significant result: the Baseline agent ends training with a substantial fraction of each episode spent in hazardous regions. At the end of training, it spends $\approx$ 50% of its time in a hazard. CRC improves on this but still incurs frequent incursions alongside MC-Dropout, while MC-CP spends very little time in hazards. Overall, these results show an interesting finding: Monte Carlo sampling, either used directly (MC) or

within MC-CP, helps to reduce the effect of noisy state features. Moreover, conformal calibration with MC-Dropout (COPPOL using MC-CP) yields the most robust behaviour: preserving task performance while substantially limiting hazardous exposure even when the state is corrupted at every step.

Taken together, the noisy-state experiments empirically validate the third and final hypothesis: COPPOL with MC-CP maintains high task performance while substantially improving safety under degraded sensing.

## 3.5 Discussion

In this chapter, we introduced **M**onte-**C**arlo **C**onformal **P**rediction (MC-CP), a hybrid framework for robust UQ designed to address the challenge of imperfect perception in perception-to-policy pipelines for safety-critical domains. MC-CP combines a novel Adaptive Monte Carlo Dropout with CP, regularising both the head and tail of the predictive distribution. Formulated as a general, model-agnostic procedure for constructing well-calibrated prediction sets and intervals, it is applicable to both classification and regression tasks. Empirically, MC-CP is able to yield stronger coverage than state-of-the-art methods and perform more robustly under distributional/covariate shifts, directly tackling the central problem of unreliable and over-confident perception.

MC-CP only address one half of the perception-to-policy pipelines; as a result, we proposed a unified **Co**nformal **P**erception-to-**Po**licy **l**earning (COPPOL) framework that integrates MC-CP into the policy learning stage. Whereas standard pipelines pass raw or thresholded segmentations to an RL agent, COPPOL transforms sensory observations into conformally calibrated hazard maps that explicitly bound missed hazards under distribution-free, finite-sample guarantees. Empirically, across multiple datasets, COPPOL retained the task performance of baseline agents while consistently increasing hazard clearance, reducing time spent in unsafe terrain, and maintaining stability under degraded/noisy sensing.

### 3.5.1 Applicability and Impact

MC-CP is designed as a general, model-agnostic framework for robust UQ, and its applicability extends well beyond the specific instantiations considered in this chapter. The construction of MC-CP requires only access to a base predictor that produces real-valued scores and a held-out calibration set drawn from the deployment distribution. Under these mild requirements, MC-CP can be wrapped around a wide variety of architectures, including convolutional networks, transformers, and other modern deep models [255], and applied to both classification and regression problems. In Sections 3.3.1.2 and 3.3.1.3, we demonstrated this flexibility by applying MC-CP to standard benchmark tasks, showing that it yields prediction sets and intervals that satisfy the target coverage while remaining substantially more compact than existing conformal baselines. Since the calibration step is performed post hoc and does not require retraining the underlying network, MC-CP can, in principle, be integrated into existing pipelines with minimal engineering effort, serving as a drop-in mechanism for upgrading probabilistic predictions with

finite-sample guarantees. This is particularly attractive in safety-critical perception settings, where practitioners benefit from an explicit and interpretable control parameter, the risk level $\alpha$, that directly bounds the expected rate of missed hazards, rather than relying on heuristic threshold tuning or probabilistic scores whose calibration may degrade under shift.

The perception-to-policy experiments, which introduced COPPOL in Section 3.4, show how these calibrated predictions can be made actionable for downstream decision-making. Within COPPOL, MC-CP transforms conventional segmentation outputs into conformally calibrated hazard maps that are embedded into the state and reward structure of an RL agent. Empirically, COPPOL with MC-CP achieved markedly higher coverage of unsafe regions than deterministic and MC-dropout baselines while retaining competitive task performance, both in terms of reward and success rate and in terms of safety metrics such as clearance from hazards and time spent in hazardous regions. A key aspect of COPPOL's applicability is that it remains largely agnostic to the choice of RL algorithm and state representation: in this chapter, we considered both vector- and image-based state encodings, yet the qualitative safety improvements of MC-CP persisted across these design choices. In more complex systems, the same conformally calibrated hazard information could be consumed by actor–critic methods, model-based planners, or safe RL filters such as shields and reachability-based controllers. Because COPPOL only assumes that the downstream agent can access the calibrated hazard map (or features derived from it) and incorporate hazard penalties into its objective, the framework is compatible with a broad class of decision-making architectures. As a result, the combination of MC-CP and COPPOL offers a general recipe for propagating statistically grounded uncertainty from perception into control, with potential impact wherever perception-driven autonomy must balance performance and safety under imperfect sensing.

### 3.5.2 Limitations

While MC-CP and COPPOL address several shortcomings of existing uncertainty-aware perception and control methods, they also come with important limitations and assumptions that constrain their guarantees and practical applicability.

From a statistical perspective, the guarantees provided by MC-CP are inherently marginal rather than worst-case. The conformal calibration procedure ensures that, under the exchangeability assumption, the expected false negative rate over future samples does not exceed the target level $\alpha$. These guarantees depend on having calibration data that is from the same underlying data distributions as the training data; which can be either difficult to find or will be partitioned from the training data, potentially reducing base model performance. Computationally, Adaptive MC Dropout adds stochastic forward passes, and the conformal step requires an additional calibration sweep; this overhead grows with input resolution and latency constraints. In resource-constrained settings, this extra computational overhead may not be as feasible.

When MC-CP outputs are consumed by COPPOL, pixel-marginal guarantees do not automatically imply spatial/instance or trajectory-level safety: specific regions (e.g., thin rims) or individual episodes can still violate the bound even if the marginal objective is met, and the

RL agent's closed-loop behaviour is not covered by perception-level calibration. There is also a mismatch between CP's exchangeability assumption and RL's non-stationary data distribution: COPPOL mitigates this by calibrating perception per-image, but maintaining guarantees inside a feedback loop remains open. COPPOL's current implementation is further limited by the simplified experimental setting. The environments are static, planar, and discretised as pixel lattices with discrete actions and short horizons, which simplifies both dynamics and planning. In practice, it may be preferable to operate on continuous grids so that the agent can optimise smooth trajectories and reason about sub-pixel geometry. Likewise, many applications require 3D perception, where conformal calibration must extend to spatial fields. The hazard representation is also binary; a richer formulation would model continuous risk fields (e.g., per-cell hazard intensity or likelihood), enabling cost shaping that reflects varying severities rather than a hard safe/unsafe split. Finally, feeding full hazard maps (i.e., when using the image view state representation) increases state dimensionality and computation, which may stress real-time deployments even though asymptotic performance is comparable.

# Chapter 4

# Reducing Over-Conservatism in Safety Critics

This chapter introduces an approach for reducing over-conservatism in safety critics through uncertainty-aware modulation and refinement, enabling policies to achieve more precise reward–safety trade-offs. We evaluate the proposed method across a variety of continuous-control environments and safety-critical benchmarks. This chapter describes the motivation for the approach in Section 4.1. We then describe the methodology of the approach in Section 4.2, and fully evaluate it across various test suites in Section 4.3. Finally, we summarise the chapter in Section 4.4 where we discuss the method's impact, limitations, and areas for future work.

## 4.1 Motivation

The previous chapter examined how safety in RL can be fundamentally limited by imperfect perception. There, the central observation was that an otherwise well-behaved safe RL agent can only avoid hazards that are visible in its perceptual representation; if obstacles are misdetected or entirely absent, no amount of downstream optimisation/safety features can recover safety. The previous analysis showed how uncertainty-aware perception with distribution-free, finite-sample coverage guarantees (via our proposed MC-CP) can mitigate this issue by exposing uncertainty to downstream decision-making. This mitigates the perception-side bottleneck in the perception-to-policy pipeline, but only addresses one half of the overall problem.

As discussed previously, in practical systems/embodied agents, perceptual outputs are fed into a downstream decision-making module that includes explicit safety mechanisms, such as shields or safety critics, intended to regulate the learned policy's behaviour. One source of failure in this module is misaligned or poorly specified safety constraints, which can, in principle, be reduced through careful design and domain modelling. A more subtle and less easily addressed issue arises when this safety component is implemented as a learned safety critic (a widely used approach in safe RL, as discussed in Section 2.2.2.3.3): in that case, the quality and structure of the critic's cost estimates can potentially become a new bottleneck for safety. The safety critic

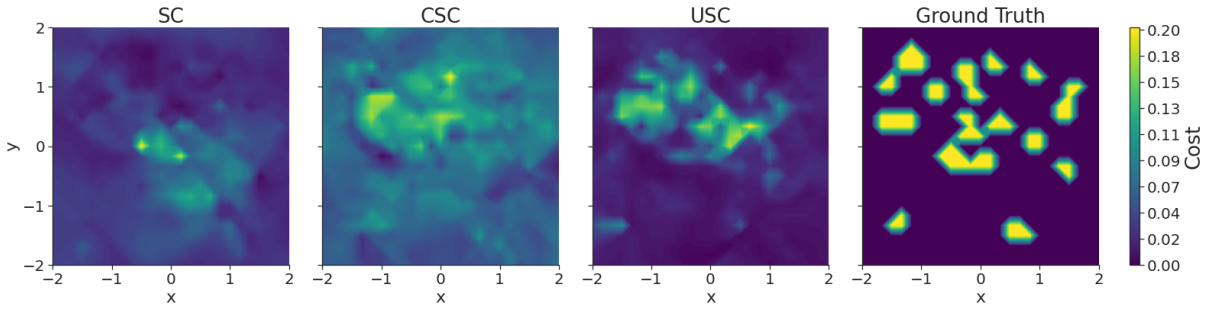



Figure 4.1: A qualitative comparison of the predictive cost maps from the vanilla Safety Critic (SC), Conservative Safety Critic (CSC), and proposed Uncertain Safety Critic (USC) methods in the Safety-Gymnasium CarGoal2 environments against the ground truth cost map.

converts sparse, delayed cost feedback into dense guidance for the policy: predicting future safety violations from state–action pairs and to penalising unsafe behaviour through a cost-based objective. This offers a way to embed safety directly into the learning process, since the critic provides a continuous safety signal that can shape both exploration and exploitation and can be learned from interaction, only requiring a cost function. In practice, however, the effectiveness of this approach is tightly coupled to the structure of the learned cost landscape; if the critic's predictions are noisy, biased, or poorly structured, the gradients passed to the policy no longer reflect the true risk of different behaviours, undermining both safety and task performance.

Standard/vanilla safety critics [208], which aim to approximate expected cumulative cost as accurately as possible, face substantial challenges [213]. Cost signals in safety-critical domains are often sparse, heavily skewed towards zero cost, and may only be observed when rare failures occur. Under these conditions, the critic is prone to underestimating risk in poorly explored regions. When this happens, the policy is encouraged to exploit these regions despite limited cost feedback, leading to frequent constraint violations or unstable behaviour under distributional shift. Instability in the critic's predictions propagates directly into the optimisation process: if the critic's gradients are misaligned with true hazard boundaries, the mechanisms enforcing constraints can oscillate or diverge, producing training instability and unreliable policies.

To counteract systematic underestimation, conservative variants [213] of safety critics have been proposed that deliberately bias predictions upward. Instead of learning an unbiased estimate of cost, these critics are trained to overestimate it. This pessimistic stance is intended to *err on the side of caution*, so that the policy becomes conservative in its exploration. However, this shift towards pessimism introduces a different point of failure rooted in over-conservatism. By inflating cost predictions uniformly across the state–action space, conservative critics tend to produce diffuse cost landscapes in which large regions appear unsafe. The boundary between genuinely hazardous and safe regions becomes blurred. From the policy's perspective, the resulting gradients can be weak, saturated, or poorly aligned with the actual structure of risk. In truly safe regions, inflated cost signals interfere with reward optimisation, suppressing the gradients that would otherwise refine behaviour and improve task performance. Evidently, the optimisation

landscape can collapse: the policy receives little meaningful information about how to trade off reward and cost, and learning may stall entirely. Qualitative examples of inaccurate/diffuse cost gradients can be observed in Figure 4.1.

These phenomena reflect a deeper conceptual issue: existing safety critics tend to conflate uncertainty with risk. Under both standard and conservative formulations, underexplored regions are often treated as hazardous, independent of the actual cost structure. Uncertainty arising from limited data, dynamic policies, or complex environment dynamics is grouped into a global pessimistic bias.

In the context of imperfect perception, uncertainty was not simply a warning flag, but a structured signal that could be calibrated, propagated, and leveraged by the policy to inform risk assessment. Similarly, the safety critic should not treat uncertainty solely as a justification for uniform overestimation. Instead, uncertainty could be used to shape the pattern of conservatism across the state–action space, concentrating pessimism where data are scarce or observed costs are high and relaxing it where repeated experience indicates that the environment is safe.

Building on this perspective, we propose the **U**ncertain **S**afety **C**ritic (USC), a novel safety critic that shapes its conservative bias through uncertainty-weighted objectives and further sharpens predictions via an uncertainty refinement procedure that interpolates poorly represented regions of the state–action space, preserving the protective benefits of pessimism in genuinely risky or poorly understood regions while avoiding the diffuse, flattened cost landscapes that otherwise cripple learning. Empirical evaluations across multiple safety test suites demonstrate that USC produces sharper and more reliable cost maps, enabling policies to achieve scalable and precise reward–cost trade-offs.

## 4.2 USC

Building on the conceptual overview introduced above, this section formalises the **U**ncertain **S**afety **C**ritic (USC) methodology, and details its components. First, we present a high-level overview, visualised in Figure 4.2, to illustrate its integration into a standard safe RL training loop.

During training, USC employs influence functions (discussed in Section 2.2.1.1.1) to estimate the sensitivity of critic parameters to specific training samples, quantifying uncertainty at the level of parameter attribution. These uncertainty estimates modulate the safety critic's cost predictions by reducing the underestimation of costs in uncertain or costly regions and preventing their overestimation in adequately covered (safe) regions. This distinction preserves conservative behaviour where needed, and maintains informative reward gradients from the reward critic $Q_R$. To further improve the safety critic's reliability, USC identifies replay buffer samples (data) with the highest estimated uncertainty. For each such sample, USC performs linear interpolation with its nearest confidently predicted neighbour, generating synthetic samples that span the critic's decision boundary. These interpolated transitions are used in a constrained update to reduce uncertainty in sparsely represented regions.

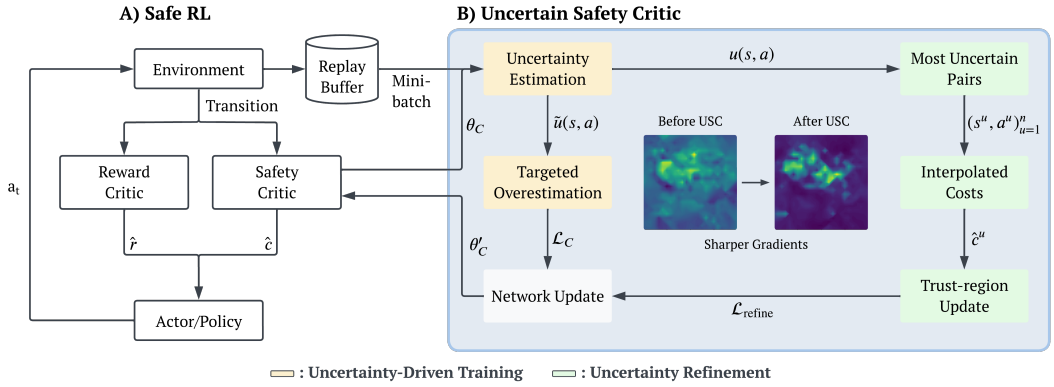



Figure 4.2: High-level USC overview, showing its integration into the safe RL training loop. The safety critic receives transitions from the environment and, alongside the reward critic, guides policy updates. Within USC, uncertainty estimation is used to compute influence scores $\tilde{u}(s,a)$, which drive targeted overestimation in the conservative loss $\mathcal{L}_C$ to preserve sharp, informative gradients. In parallel, the most uncertain state–action pairs are identified, interpolated with confident neighbours to form synthetic costs $\tilde{c}^u$, and incorporated into a trust-region update $\mathcal{L}_{\text{refine}}$. The Actor and Reward Critic Networks are updated as standard. Together, these mechanisms reduce over-conservatism while ensuring that gradient structure remains accurate and actionable.

In the remainder of this section, we formally define the uncertainty-aware training process in Section 4.2.1, and then describe the uncertainty refinement procedure in Section 4.2.2.

### 4.2.1 Uncertainty-Driven Training

USC departs from widely-used failure-based safety formulations that rely on binary constraint goals $c_t \in \{0,1\}$ [211, 213]. In those settings, the critic only observes whether a catastrophic constraint violation occurred, instead of a fine-grained measure of non-catastrophic constraint violations or violation severity. The concept of how close the agent is to a catastrophic failure can only then be inferred indirectly from the agent's trajectory via the value function instead of via environment feedback. Instead, USC operates under a continuous cost-based setting [169]. Each transition incurs a non-negative real-valued cost $C(s,a) \in \mathbb{R}_+$, which can encode both hard and soft violations. The cost function therefore acts as a second scalar feedback signal alongside the reward function: reward measures task performance, whereas cost measures safety violations or their severity. For example, costs may scale with collision intensity or distance to a hazard, which the binary setting cannot capture. This formulation provides denser supervision for the safety critic, exposes meaningful gradients that are essential for scalable safety under tight constraints, and avoids ambiguity in how "failure" is defined, thereby enhancing applicability across diverse safety specifications. Moreover, binary safety signals arise as a special case of this framework (e.g., by restricting $C(s,a) \in \{0,1\}$), whereas the purely binary setting cannot express graded notions of risk.

USC works in the CMDP formulation [141, p. 21] (as described in Section 2.2.2), where $M =$

$(S, A, P, R, C, \chi, \gamma)$, and $\chi$ is the maximum allowed cost. The objective is to find a policy $\pi$ that maximises the expected reward while satisfying the expected cost according to the value of $\chi$. The safety critic is a parametric function:

$$\begin{aligned} Q_C &: S \times A \to \mathbb{R}_{\geqslant 0} \\ Q_C(s, a; \theta_C) &\approx \mathbb{E}\Big[ \sum_{n=0}^{\infty} \gamma^n C(s_{t+n}, a_{t+n}) | s_t = s, a_t = a, \pi_\theta \Big] \end{aligned} \tag{4.1}$$

with parameters $\theta_C$. The goal of the safety critic $Q_C$ is to satisfy three properties: its predictions reflect the true expected cumulative cost induced by the current policy; it does not underestimate cost in regions where underestimation would lead to constraint violations; and its gradients are sharp and aligned with hazard boundaries, so that policy updates receive informative safety directions rather than flat or saturated signals.

Standard safety critics focus primarily on accuracy (through Bellman regression). Conservative safety critics add a global pessimistic bias that encourages overestimation across the state–action space. USC aims to provide all three simultaneously, which has not been done before, by modulating conservatism by epistemic uncertainty, concentrating pessimism where the critic is both uncertain and costly, while easing it where data supports low risk.

To quantify epistemic uncertainty in the safety critic, we incorporate upper bound estimates on $Q_C$ to overestimate alternative actions from the action space and weight them based on their confidence using Gauss-Newton influence [70], thus preventing overestimation of well-predicted data. We adopt Gauss-Newton over alternative influence functions (discussed in Section 2.2.1.1.1) as it provides stable, interpretable, and scalable uncertainty estimates using only first-order gradients and a positive semi-definite curvature [256].

The replay buffer stores previously observed transitions containing the state, action, reward, safety cost, and next state, allowing experience to be reused across training updates. Our configuration is detailed in Appendix B.4. Let $B = \{(s_i, a_i, r_i, c_i, s_{t+1,i})\}_{i=1}^{|B|}$ denote a mini-batch of transitions sampled from the replay buffer $\mathcal{B}$, where $B \subseteq \mathcal{B}$, and let $\theta_C^*$ denote the safety critic's parameters after the latest gradient update, which are treated as fixed when estimating uncertainty. For each state-action pair $(s, a)$, Gauss-Newton influence yields a scalar $u \geqslant 0$ that quantifies the safety critic's certainty:

$$\begin{aligned} u(s, a) = \operatorname{diag}\Big( & \big[\nabla_{\theta_C} Q_C(s, a; \theta_C^*)\big] \\ & \times \Big( \sum_{i=1}^{B} \nabla_{\theta_C} Q_C(s_i, a_i; \theta_C^*) \nabla_{\theta_C} Q_C(s_i, a_i; \theta_C^*)^\top + \delta I \Big)^{-1} \\ & \times \big[\nabla_{\theta_C} Q_C(s, a; \theta_C^*)\big]^\top \Big) \end{aligned} \tag{4.2}$$

where $\delta I$ is a small dampening term ensuring invertibility of the Gauss-Newton matrix. Intuitively, $u(s, a)$ measures the sensitivity of the safety critic's parameters are to each training

sample; higher $u(s,a)$ values indicate greater uncertainty, meaning that small changes to those samples would cause large shifts in the safety critic's parameters.

The raw influence scores $u(s,a)$ can vary over several orders of magnitude. Naively using them to scale the safety critic's loss term would make training unstable, as a few highly uncertain samples could dominate the gradient. We therefore map the raw uncertainty scalars into an uncertainty-adjusted weight:

$$\tilde{u}(s,a) = \log(1 + u(s,a)) \times (1 + \mathbf{1}\{Q_C(s,a) > \frac{1}{|B|}\sum_{i=1}^{|B|} Q_C(s_i,a_i)\}) \tag{4.3}$$

This transformation serves two purposes. The logarithmic term $\log(1+u(s,a))$ compresses the range of influence scores, preventing a small subset of scalars from producing excessively large gradients while still preserving the ordering of uncertainty. Samples with higher influence remain more heavily weighted, but their effect grows sub-linearly. The second term boosts the weight for transitions whose predicted cost is above the batch mean. In other words, conservatism is emphasised in regions that are both uncertain and currently predicted to be relatively costly. Samples that are uncertain but more safe than others (below-average cost) or costly but certain are treated with some caution, avoiding unnecessary inflation that would flatten gradients in already-safe regions. Finally, samples that are low cost and certain are given a very small weight $\tilde{u}(s,a) \to 0$, to remove nearly all pessimism.

The safety critic is then updated by minimising:

$$\begin{aligned}\mathcal{L}_C = \mathbb{E}_{(s_t,a_t,r_t,c_t,s_{t+1})\sim\mathcal{B}}\Bigg[&\frac{1}{2}\Big(Q_C(s_t,a_t) - \big(c_t + \gamma Q_C(s_{t+1},\pi(s_{t+1}))\big)\Big)^2 \\ &+ \frac{1}{2}\,\tilde{u}(s_t,a_t)\;\times \log \sum_{a'\sim\mathrm{Unif}(A)} \exp\left(\frac{Q_C(s_t,a') - \mu}{\sigma+\epsilon}\right) - \frac{Q_C(s_t,a_t)-\mu}{\sigma+\epsilon}\Bigg]\end{aligned} \tag{4.4}$$

where $\mu$ and $\sigma$ denote the batch mean and standard deviation of the safety critic outputs, and $\epsilon$ is a small constant for numerical stability. The first term is a standard Bellman regression loss. It encourages $Q_C$ to match the observed continuous costs and propagate them forward in time through the dynamics, ensuring that predictions remain consistent with the environment. The second term introduces USC's uncertainty-weighted conservative bias. The log-exp-sum compares the predicted cost of the current action $a_t$ with a sampled set of alternatives $a'$. When any alternative action appears safer (lower cost) than the current action, the penalty increases, encouraging the critic to raise the estimated cost of $a_t$ relative to the alternatives. Sampling alternatives from a uniform distribution over $A$ rather than from the current policy [213] is an important addition to USC: under a policy that already avoids hazards, sampling from $\pi_\theta(\cdot|s_t)$ would rarely visit truly risky actions, making the conservative term ineffective; uniform sampling is more likely to draw out-of-distribution, high-cost actions, thereby exerting stronger pressure to maintain pessimism around potential hazards.

**Theorem 4.1.** *The log-exp-sum (LSE) penalty, used in Equation 4.4 and standard in CSC [213], is more effective at encouraging conservatism when the m alternative actions are sampled uniformly over the action space, rather than from the current policy [213]. If the policy aims to avoid high-cost regions, then the uniform sampling is more likely to draw from high-cost out-of-distribution actions. Formally, for any fixed state s, the conservative term is strictly larger under uniform scaling.*

The corresponding proof is in Appendix B.1. The safety critic's parameters are updated by

$$\theta_C \leftarrow \theta_C - \eta_C \nabla_{\theta_C} \mathcal{L}_C \tag{4.5}$$

where $\eta_C > 0$ is the learning rate for the safety critic. The reward critic $Q_R$ is trained in parallel with a standard Bellman error on the reward signal [41], leaving the task-value estimates unaffected since the uncertainty-based modulation is only applied to $Q_C$. The actor is optimised with a Lagrangian objective that balances reward maximisation, cost penalisation, and stability under policy shifts. Its loss is given by:

$$\begin{aligned} \mathcal{L}_\pi = & - \mathbb{E}_{s_t \sim B}\Big[ Q_R\big(s_t, \pi_{\theta_\pi}(s_t)\big) \Big] \\ & + \lambda\, \mathbb{E}_{s_t \sim B}\Big[ Q_C\big(s_t, \pi_{\theta_\pi}(s_t)\big) \Big] \\ & + \delta\, D_{\mathrm{KL}}\Big( \pi_{\theta_\pi}(\cdot|s_t) \,\big\|\, \pi^{\mathrm{old}}_{\theta_\pi}(\cdot|s_t) \Big) \end{aligned} \tag{4.6}$$

where $\lambda \geqslant 0$ is the dual variable, and the KL term regularises policy shifts between updates with the coefficient $\delta > 0$. The actor parameters are updated by:

$$\theta_\pi \leftarrow \theta_\pi - \eta_\pi \nabla_{\theta_\pi} \mathcal{L}_\pi \tag{4.7}$$

and the dual variable is adjusted after each episode using the empirical average cost $\hat{C}$:

$$\lambda \leftarrow \max\Big(0, \lambda - \eta_\lambda\big(\chi - \hat{C}\big)\Big) \tag{4.8}$$

ensuring that the cost budget $\chi$ is enforced in expectation. Because the actor's updates depend on $Q_C$, the shape of the cost landscape induced by the uncertainty-driven training objective directly determines how the policy navigates the reward–safety trade-off. By using continuous costs and concentrating conservatism where uncertainty is high, USC constructs a safety critic that is both protective and structurally informative. In safe, well-understood regions, gradients from $Q_C$ remain sharp and allow the policy to continue improving reward; in risky or poorly explored regions, the critic becomes deliberately pessimistic, steering the policy away from potential violations.

### 4.2.2 Uncertainty Refinement

The uncertainty-weighted conservative objective shapes the safety critic by amplifying overestimation in regions where the critic is epistemically uncertain, while suppressing unnecessary inflation in well-covered, low-cost regions. This modulation concentrates conservatism where it matters most and preserves sharper gradients elsewhere. However, the objective by itself does not reduce epistemic uncertainty: state–action pairs that are rarely observed or lie near poorly explored decision boundaries can remain permanently high-cost and uncertain. In such regions, the critic may continue to treat *unknown* as *unsafe*, leading to persistently diffuse cost maps, overly cautious action filtering, and weak policy gradients. To address this, USC augments uncertainty-driven training with an explicit refinement mechanism to help reduce uncertainty in these regions and allow the agent to explore more of its environment.

Firstly, after the Gauss-Newton influence produces a set of scalars $u(s,a)$ for each transition in the mini-batch $B$, USC ranks the current mini-batch in descending order of $u(s,a)$ and selects the $n$ most uncertain samples $U^n \in B$. To refine these uncertain cost estimates around these inputs, we construct synthetic cost targets by interpolating each $(s^u, a^u) \in U^n$ with nearby samples. Let $\{(s^k, a^k, c^k)\}_{k=1}^K \subset \mathcal{B}$ be its $K$ nearest neighbours in the replay buffer $\mathcal{B}$. We define an interpolated target cost as:

$$\begin{aligned} \hat{c}^u &= \sum_{k=1}^{K} w_k c^k \\ w_k &= \frac{1/\left\|(s^u,a^u)-(s^k,a^k)\right\|_2}{\sum_{j=1}^{K} 1/\left\|(s^u,a^u)-(s^j,a^j)\right\|_2} \end{aligned} \tag{4.9}$$

where $\|\cdot\|_2$ denotes the Euclidean ($L_2$) norm on the concatenated state-action vectors, so the weights $w_k$ are inversely proportional to their distance from the uncertain sample. This interpolation encourages smoothness across the critic's decision boundary, enabling uncertain regions to inherit structure from their nearest confident neighbours, thus reducing epistemic uncertainty in sparsely sampled regions.

To incorporate these synthetic targets, we adopt a trust region update on the safety critic's parameters. Given the old predictions $Q_C(s^u, a^u; \theta_C)$ and the new predictions under parameters $\theta'_C$, we compute a regularised loss

$$\begin{aligned} \mathcal{L}_{\text{refine}} = \frac{1}{|U^n|} \sum_{(s^u,a^u)\in U^n} \Bigg[ &\frac{1}{2}\big(Q_C(s^u,a^u) - \hat{c}^u\big)^2 \\ &+ \beta\Big( \max\big\{0, D_{\text{KL}}\big(Q_C(s^u,a^u)||Q_C(s^u,a^u;\theta^{\text{old}}_{\text{safe}})\big) - \epsilon\big\}\Big)^2 \Bigg] \end{aligned} \tag{4.10}$$

where the first term enforces consistency with interpolated costs, and the second penalises large deviations from the previous critic predictions beyond a threshold $\epsilon$, and $\beta$ is a trust-region penalty coefficient controlling the strength of the KL regularisation term. The quadratic penalty ensures that updates remain local in function space, preventing destabilising shifts in

low-data regions. We perform this step as a trust-region update because refinement relies on interpolated rather than ground-truth cost signals. Although interpolation is constrained to local neighbourhoods and weighted by distance, the resulting $\hat{c}^u$ values may not perfectly match the true environment's cost function $C(s, a)$. Without additional safeguards, aggressively fitting to these synthetic targets could distort the critic in ways that harm both safety and performance. The KL-regularised loss $\mathcal{L}_{\text{refine}}$ only allows small measured adjustments towards the interpolated costs. In effect, refinement acts as a cautious, guided smoothing step that sharpens uncertain regions while preserving global consistency. The critic's parameters are then refined by:

$$\theta_C \leftarrow \theta_C - \eta_C \nabla_{\theta_C} \mathcal{L}_{\text{refine}} \tag{4.11}$$

Combined with the uncertainty-weighted conservative objective, uncertainty refinement equips USC with a mechanism to reshape the cost landscape in precisely those regions where the critic is least reliable. The critic continues to *err on the side of caution* in uncertain or costly regions, but refinement allows it to progressively distinguish genuinely hazardous areas from merely under-sampled, potentially high-reward regions. This, in turn, preserves sharp and informative gradients for the actor, enabling the policy to recover meaningful reward–cost trade-offs instead of being trapped behind flat or saturated cost barriers.

All updates described above are embedded within the standard RL loop (Algorithm 6 with the USC extension). At each step, the actor proposes a candidate action, which is then screened by the safety critic. If the predicted cost exceeds that safety tolerance $\varepsilon_{\text{safe}}$, additional candidate actions are produced using Ornstein–Uhlenbeck noise [41] (lines 6-9). Among these candidates, the action with the lowest predicted cost is selected, provided it satisfies the tolerance $\varepsilon_{\text{safe}}$; otherwise, the least risky action is chosen. This mechanism ensures that the policy not only maximises reward but also respects the safety critic's guidance. Transitions are stored in the replay buffer, from which the actor and both critics are iteratively updated. The dual variable $\lambda$ is dynamically updated to maintain the cost budget (line 31). In this way, the safety critic is integral to both training and execution, shaping policy improvement while directly filtering unsafe actions.

The user-defined hyperparameters introduced by USC, the Gauss-Newton dampening term $\delta$, the trust-region penalty $\beta$, the trust threshold $\epsilon$, and the safety tolerance $\varepsilon_{\text{safe}}$, control the sensitivity/conservativeness of the method. The remaining hyperparameters, the number of $m$ alternative actions, the various learning rates $\eta$, are inherited/common among all safety critics, and thus, retain the same functional behaviour. It should be noted that any additional procedures from USC do not alter the functional behaviour of inherited hyperparameters. Small values of the Gauss–Newton dampening term $\delta$ influence scores more sensitive to local curvature and thus amplify the effect of uncertainty modulation, whereas larger values stabilise the curvature approximation and smooth out extreme influence values. The trust-region penalty coefficient $\beta$ scales the KL-based penalty in the refinement loss; increasing $\beta$ tightens the trust region, yielding smaller, more conservative refinement steps, while a smaller $\beta$ allows faster adaptation to interpolated targets. The trust threshold $\epsilon$ defines the dead zone of this penalty. For

**Algorithm 6** Uncertain Safety Critic (USC) inside the Safe RL Loop

**Require:** Environment $M$, replay buffer $\mathcal{B}$, policy $\pi_{\theta_\pi}$, reward critic $Q_R$, safety critic $Q_C$, discount factor $\gamma$, safety budget $\chi$, dual variable $\lambda \geqslant 0$, trust-region penalty coefficient $\beta$, trust threshold $\epsilon$, learning rates $\eta_\pi, \eta_R, \eta_C, \eta_\lambda$, safety tolerance $\epsilon_{\text{safe}}$

Initialise $\theta_\pi, \theta_R, \theta_C$, targets, $\lambda \leftarrow 1$
**for** $e = 1, 2, \ldots, E_{\max}$ **do**
  $s_1 \leftarrow \textsc{Reset}()$, $t \leftarrow 1$,  episode cost accumulator $C_e \leftarrow 0$
  **while** $t \leqslant T$ and $s_t$ not terminal **do**
    $a_t \leftarrow \pi_{\theta_\pi}(s_t)$
    **if** $Q_C(s_t, a_t) > \epsilon_{\text{safe}}$ **then**
      $\mathcal{N}(s_t) = \{a_t + \text{OU}(0, \sigma^2_{\text{OU}}), k = 1, \ldots, N_{\text{samples}}\}$
      $a_t \leftarrow \arg\min_{a' \in \{a_t\} \cup \mathcal{N}(s_t)} Q_C(s_t, a')$
    **end if**
    $(r_t, c_t, s_{t+1}) \leftarrow \textsc{Step}(a_t)$;  $C_e \leftarrow C_e + c_t$
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, a_t, r_t, c_t, s_{t+1})\}$;  $t \leftarrow t + 1$
    $B \leftarrow \{(s_i, a_i, r_i, c_i, s'_i)\}_{i=1}^{|B|} \overset{\text{iid}}{\sim} \mathcal{U}(\mathcal{B})$
    **for** $\{(s_i, a_i, r_i, c_i, s'_i)\} \in \mathcal{B}$ **do**
      $U \leftarrow U \bigcup \textsc{CalculateUncertainty}(B, s_i, a_i)$ {Eq. 4.2}
      $\tilde{U} \leftarrow \tilde{U} \bigcup \textsc{UncertaintyAdjustment}(B, u(s_i, a_i))$ {Eq. 4.3}
    **end for**
    $\mu \leftarrow \frac{1}{B} \sum_i Q_C(s_i, a_i)$,  $\sigma \leftarrow \text{Std}\big(\{Q_C(s_i, a_i)\}_{i=1}^{|B|}, \mu\big)$
    $\mathcal{L}_C \leftarrow \textsc{CalculateQ}_C\textsc{Loss}(B, \tilde{U}, \mu, \sigma)$ {Eq. 4.4}
    $\theta_C \leftarrow \theta_C - \eta_C \nabla_{\theta_C} L_C$ {Eq.4.5}
    $\mathcal{L}_R \leftarrow \textsc{CalculateQ}_R\textsc{Loss}(B, \gamma)$
    $\theta_R \leftarrow \theta_R - \eta_R \nabla_{\theta_R} L_R$
    $L_\pi \leftarrow \textsc{CalculateActorLoss}(B, \delta, \pi_{\theta_\pi})$ {Eq. 4.6}
    $\theta_\pi \leftarrow \theta_\pi - \eta_\pi \nabla_{\theta_\pi} L_\pi$ {Eq.4.7}
    $U^n \leftarrow \textsc{SelectTop}(U, n)$
    **for** $u \in U^n$ **do**
      $u^K \leftarrow \textsc{FindNeighbours}(u, \mathcal{B}, K)$
      $\hat{C}^u \leftarrow \hat{C}^u \bigcup \textsc{CalculateTargetCost}(u^K)$ {Eq.4.9}
    **end for**
    $\mathcal{L}_{\text{refine}} \leftarrow \textsc{CalculateRefineLoss}(\hat{C}^u, U^n, \beta, \epsilon)$ {Eq. 4.10}
    $\theta_C \leftarrow \theta_C - \eta_C \nabla_{\theta_C} L_{\text{refine}}$ {Eq.4.11}
  **end while**
  $\lambda \leftarrow \max\big(0,\ \lambda - \eta_\lambda (\chi - C_e)\big)$ {Eq.4.8}
**end for**

$D_{\text{KL}} \leqslant \epsilon$ refinement updates are effectively unpenalised, whereas for $D_{\text{KL}} > \epsilon$ the penalty grows quadratically. So, reducing $\epsilon$ makes even modest divergence from the previous gradients more costly, while enlarging $\epsilon$ permits bolder refinements before the trust region engages. Finally, the safety tolerance $\varepsilon_{\text{safe}}$ specifies how unsafe the preliminary action can be before it is rejected.

Finally, it is useful to consider limiting cases in which USC's extensions fail and examine how the method behaves. In the limiting case where the Gauss–Newton influence estimation fails to provide informative uncertainty signals, USC naturally degenerates to the underlying conservative or even vanilla safety critic. In the case the curvature approximation in $(G + \delta I)^{-1}$ is poor or $\delta$ is chosen sufficiently large, the resulting influence scores $u(s, a)$ could tend towards a near-constant scalar across the batch, so that the uncertainty weights $\tilde{u}(s, a)$ become effectively uniform. In this case, the uncertainty-weighted conservative loss reduces to the original CSC uniform conservative objective [213]. In the same scenario, the influence scores could tend towards zero $u(s, a) \to 0$. In this case, the scalars would cancel out any conservatism, and the safety critic objective would reduce to a standard safety critic [211]. In this sense, USC is backwards compatible with existing safety critic formulations: even when influence-based uncertainty or refinement are ineffective, the method collapses gracefully to the behaviour of the underlying safety critic rather than introducing new failure modes.

An analogous set of arguments applies to the refinement component. Suppose the KL trust region is set too tight (e.g., very large $\beta$ or very small $\epsilon$). In that case, any deviation from the old gradients becomes heavily penalised and refinement updates are effectively blocked, so $\mathcal{L}_{\text{refine}}$ contributes almost no gradient and training behaves like the conservative or standard safety critic. Conversely, if the trust region is too loose (e.g., small $\beta$, large $\epsilon$), refinement may smooth hazard boundaries, but it still behaves as a data-driven regression towards interpolated costs, rather than introducing qualitatively new gradient behaviour. Similar degeneracies arise when the influence scores $u(s, a)$ become uninformative (e.g., nearly constant or close to zero): the top-$n$ set of *uncertain* states then becomes an arbitrary selection, so the critic is refined on random data points which would not affect training. Finally, when the replay buffer is small or dominated by early, noisy rollouts, the local neighbourhoods used to construct $\hat{c}_u$ may be poorly representative, reducing the accuracy of interpolated costs and limiting the benefit of refinement; in this case, the main Bellman updates driven by ground-truth costs dominate, so any oversmoothing introduced by refinement is outweighed by supervised corrections.

A simpler hypothetical alternative would be to estimate safety directly from neighbouring state-action samples using K-nearest neighbours (KNN). USC already uses this principle locally in Equation 4.9, but retains a learned critic to estimate cumulative discounted cost and provide differentiable estimates for actor optimisation. Although KNN can capture nonlinear relationships, its predictions depend on the distance metric and the availability of representative neighbours. A neural critic can learn shared representations across experiences, potentially supporting generalisation where local samples are sparse. Direct interpolation of immediate costs also differs from estimating long-term consequences, although KNN could be extended to use return or Bellman targets.

## 4.3 Evaluation

This section empirically evaluates the proposed USC method across a suite of continuous-control and safety benchmark tasks. The experimental evaluation aims to verify that USC achieves the

objectives established throughout Section 4.2: reducing over-conservatism and diffuse cost landscapes to achieve a better balance of safety and task performance. In particular, the experiments are designed to assess whether uncertainty-driven training and refinement yield sharper, more informative cost gradients that support better reward–safety trade-offs. To this end, the following hypotheses are investigated.

The first hypothesis is that uncertainty-driven training and refinement enables USC to achieve strictly better reward–safety trade-offs than existing safety critics: by utilising uncertainty-modulated pessimism and uncertainty-refinement, USC should achieve the best balance of both cost and reward over training. This should be the case under tight and loose safety budgets, as well as across a mix of safety suites.

The second hypothesis is that incorporating epistemic-uncertainty modulation and refinement improves the quality of the learned safety critic. USC should produce cost landscapes whose gradients more closely align with the underlying ground truth hazard structure than existing safety critics. Concretely, regions of high predicted cost should concentrate around genuinely hazardous states and actions, and gradient information should remain non-saturated in safe regions, resulting in lower gradient error, better calibration of cost levels against empirical violation frequencies, and reduced diffuseness of the cost map.

The final hypothesis is that the deployment behaviour of USC should remain at least as safe during deployment as it is during training. In many safe RL methods, critics and policies appear safe under training but exhibit degraded safety when deployed, due to distributional shift, non-stationarity in the replay buffer, or overfitting of the critic to the training distribution. Since USC is hypothesised to improve the quality of the learned safety critic, fully on-policy deployment is expected to remain closely aligned to training results.

The subsequent sections evaluate these hypotheses systematically. Section 4.3.1 presents the core quantitative results for USC, and ablates the optional refinement procedure. Section 4.3.2 analyses how USC performs during deployment, followed by combining USC with MC-CP's predictive hazard maps in the COPPOL framework in Section 4.3.3.

### 4.3.1 Core Results

This section presents the core experimental results used for validating the proposed USC methods. We begin by evaluating the performance of USC, followed by the ablation of the refinement procedure, before finally examining the quality of the learned safety critic.

We first evaluate the base performance of USC to assess how effective it is at achieving a strong reward-cost trade-off. To assess this, we run experiments on a handful of environments from various safety test-suites: Safety Gymnasium [257], Gymnasium Robotics [258], and Mujoco [259]. We compare against several state-of-the-art safe RL methods: a baseline DDPG agent [41], PID-Lag [172], CPO [169], RCPO [170], SC [211], and CSC [213]. We also compare against a variant of USC that does not use the optional refinement stage to ablate how the refinement stage affects performance, named USC (NR). Safety and task performance are assessed across a range of relevant metrics, including average reward, average cost, and the total number of times

Table 4.1: Mean episodic reward and cost for comparative methods across evaluated benchmarks. Light blue and light green highlighted cells indicate the lowest cost and highest reward for each task, respectively.

| | | Safety Gymnasium | | | | Gymnasium Robotics | Mujoco |
|---|---|---|---|---|---|---|---|
| **Technique** | **Metric** | **Goal1** ($\chi = 1.0$) | **Goal2** ($\chi = 5.0$) | **Button1** ($\chi = 5.0$) | **Button2** ($\chi = 5.0$) | **FetchReach** ($\chi = 5.0$) | **HalfCheetah** ($\chi = 0.1$) |
| DDPG | Reward ↑ | $6.57 \pm 0.47$ | $8.74 \pm 0.72$ | $6.20 \pm 0.74$ | $7.51 \pm 0.67$ | $-4.36 \pm 16.04$ | $5.71 \pm 2.48$ |
| | Cost ↓ | $1.05 \pm 0.39$ | $5.36 \pm 1.47$ | $9.50 \pm 3.01$ | $10.30 \pm 3.32$ | $8.19 \pm 7.70$ | $21.82 \pm 23.20$ |
| PID-Lag | Reward ↑ | $6.48 \pm 0.49$ | $5.88 \pm 3.39$ | $2.49 \pm 2.57$ | $1.30 \pm 2.61$ | $-1.43 \pm 5.96$ | $4.69 \pm 1.52$ |
| | Cost ↓ | $1.14 \pm 0.51$ | **$4.94 \pm 2.27$** | **$4.50 \pm 4.26$** | **$3.40 \pm 4.77$** | $3.45 \pm 3.12$ | $0.36 \pm 0.35$ |
| CPO | Reward ↑ | $6.40 \pm 0.76$ | $8.74 \pm 0.81$ | $5.99 \pm 1.20$ | $7.56 \pm 0.79$ | $1.80 \pm 4.49$ | $3.69 \pm 1.89$ |
| | Cost ↓ | $1.21 \pm 0.51$ | $5.08 \pm 1.26$ | $7.47 \pm 2.62$ | $9.05 \pm 2.80$ | $4.24 \pm 1.70$ | $1.71 \pm 1.93$ |
| RCPO | Reward ↑ | $6.56 \pm 0.38$ | $8.28 \pm 0.95$ | $4.21 \pm 2.38$ | $3.67 \pm 3.18$ | $-0.77 \pm 6.57$ | $6.28 \pm 2.13$ |
| | Cost ↓ | $1.02 \pm 0.40$ | $5.13 \pm 1.71$ | $5.46 \pm 3.35$ | $4.58 \pm 3.72$ | $3.84 \pm 2.36$ | $0.94 \pm 0.84$ |
| SC | Reward ↑ | **$6.60 \pm 0.45$** | $7.54 \pm 1.38$ | $5.12 \pm 1.89$ | $2.48 \pm 3.32$ | $1.98 \pm 4.65$ | $3.19 \pm 3.43$ |
| | Cost ↓ | $1.02 \pm 0.37$ | $5.65 \pm 2.00$ | $5.96 \pm 2.68$ | $4.04 \pm 4.46$ | **$3.18 \pm 1.05$** | $0.32 \pm 0.63$ |
| CSC | Reward ↑ | $6.56 \pm 0.40$ | $8.65 \pm 0.75$ | $5.48 \pm 1.83$ | $6.45 \pm 2.44$ | $-1.53 \pm 6.51$ | $4.13 \pm 1.29$ |
| | Cost ↓ | $1.03 \pm 0.39$ | $5.28 \pm 1.44$ | $5.57 \pm 2.63$ | $6.49 \pm 3.19$ | $3.91 \pm 3.07$ | $2.71 \pm 5.70$ |
| USC (NR) | Reward ↑ | $6.56 \pm 0.39$ | $8.74 \pm 0.78$ | $6.31 \pm 0.55$ | $7.49 \pm 0.79$ | $0.55 \pm 4.00$ | **$6.64 \pm 2.26$** |
| | Cost ↓ | $1.06 \pm 0.41$ | $5.53 \pm 1.43$ | $6.37 \pm 2.05$ | $8.02 \pm 2.52$ | $3.34 \pm 1.36$ | $0.62 \pm 0.51$ |
| USC | Reward ↑ | $6.55 \pm 0.39$ | **$8.84 \pm 0.62$** | **$6.38 \pm 0.62$** | **$7.69 \pm 0.62$** | **$3.92 \pm 2.51$** | $3.76 \pm 2.70$ |
| | Cost ↓ | **$0.96 \pm 0.33$** | $5.05 \pm 1.27$ | $6.02 \pm 2.13$ | $6.32 \pm 2.23$ | $3.20 \pm 1.11$ | **$0.20 \pm 0.40$** |

the agent reached the goal. Full details of the experimental setup and environment information can be found in Appendix B.4. All methods employ identical network architectures as well as training procedures for a fair comparison. Core cost-reward trade-off results for the comparative methods across the range of test suites are reported in Table 4.1, while Figure 4.3 visualises the same results as reward–cost Pareto plots with one-standard-deviation ellipses. Full training plots for all methods for each environment can be found in Appendix B.2.

On the Safety Gymnasium goal-navigation tasks, USC achieves consistently favourable cost-reward trade-offs. In CarGoal1, SC attains the highest mean reward, with $6.60 \pm 0.45$, with the majority of other safe RL methods achieving a similar reward. However, the cost performance is more varied across methods and highlights which methods achieve the best cost-reward balance. USC achieves the lowest average cost at $0.96 \pm 0.33$, compared with $1.02 \pm 0.37$ for SC, which achieves the second lowest cost, slightly above the approved safety budget $\chi = 1.0$. The remaining comparative methods observe a much higher cost; for example, the PID-Lag agent averages $1.14 \pm 0.51$ cost, and CPO averages $1.21 \pm 0.51$ cost. This effect becomes more pronounced in CarGoal2, where USC achieves the highest reward at $8.84 \pm 0.62$, where other methods struggle to scale. For example, the standard safety critic reaches $7.54 \pm 1.38$, and the PID-Lag agent only reaches $5.88 \pm 3.39$ average reward. USC achieves the second lowest cost in this environment with $5.05 \pm 1.27$, closely behind the PID-Lag agent, which averages $4.94 \pm 2.27$ cost. Compared with PID-Lag, USC achieves substantially higher mean reward at a slightly higher mean cost. This represents a different reward-cost compromise: PID-Lag remains below the specified budget of $\chi = 5.0$ on average, whereas USC slightly exceeds it.

On CarButton1 and CarButton2, USC achieves the highest mean rewards of $6.38 \pm 0.62$ and $7.69 \pm 0.62$, respectively. However, these rewards are accompanied by mean costs of $6.02 \pm 2.13$ and $6.32 \pm 2.23$, both exceeding the specified budget of $\chi = 5.0$. PID-Lag and RCPO achieve lower mean costs within the budget, but also substantially lower rewards. These results therefore demonstrate different reward-cost compromises rather than an unqualified advantage for USC.

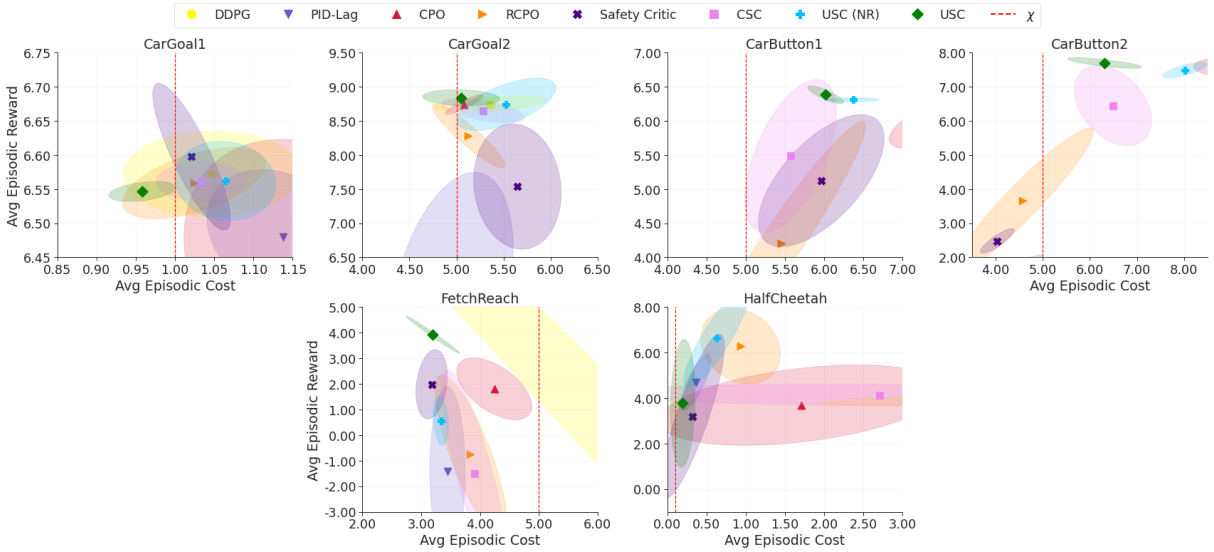



Figure 4.3: Pareto plot of average episodic cost versus average episodic reward across evaluated benchmarks. Each point denotes the mean performance, with shaded ellipses representing the standard deviation.

Relative to CPO, however, USC achieves both higher mean reward and lower mean cost on both tasks.

The Gymnasium Robotics FetchReach task is particularly challenging: instead of costs arising from obstacles/hazards, they are incurred by movement that may harm the robot's internal equipment, such as high-velocity movements and jitter, costs which are harder to reduce. The DDPG baseline achieves an average reward of $-4.36 \pm 16.04$ with a very high cost of $8.19 \pm 7.70$, showcasing the difficulty of performing well in the task, regardless of simultaneously keeping costs low. Most safe RL methods struggle to attain a reward greater than zero, but USC achieves the highest reward at $3.92 \pm 2.51$. It does so with an average cost of $3.20 \pm 1.11$, which is only $\approx 0.02$ above the SC agent, which achieved the lowest cost of $3.18 \pm 1.05$; however, it did so with a much lower reward of $1.98 \pm 4.65$.

Finally, on the Mujoco HalfCheetah task, the cost-reward trade-off is uniquely challenging; reward is given via speed of the agent; however, if the agent goes too fast, then it incurs a cost penalty. The unconstrained DDPG agent achieves a strong reward of $5.71 \pm 2.48$, but with a considerable average cost of $21.82 \pm 23.20$, showcasing the difficulty of this task. The USC (NR) agent achieves the highest reward of $6.64 \pm 2.36$, whilst the USC agent achieves the lowest cost of $0.20 \pm 0.40$.

Overall, USC does not provide a uniformly superior reward-cost balance. The clearest evidence of improvement at comparable cost occurs in FetchReach, where USC achieves higher mean reward than SC with nearly identical mean cost. CarGoal2 also shows improvements in both mean reward and mean cost relative to most comparators. Conversely, the button tasks and HalfCheetah illustrate compromises whose desirability depends on the safety requirements. These findings suggest that USC may be particularly useful where improved cost estimation

allows productive behaviour without a corresponding increase in violations. Where reward and cost are directly coupled, as through speed in HalfCheetah, improved estimation still requires a choice between competing objectives. However, methods achieving lower costs than USC often obtain much lower rewards, while those achieving higher rewards often incur greater costs, indicating alternative reward–cost compromises.

The ablation between USC and USC (NR) aids us in isolating the effect of the optional uncertainty refinement component. Across the evaluated tasks, USC (NR) typically achieves a higher reward than its safety critic counterparts, but often at a slightly elevated cost. For example, on CarGoal2, USC (NR) achieves a reward of $8.74 \pm 0.78$ with a cost of $5.53 \pm 1.43$, whereas refinement shifts USC to $8.84 \pm 0.62$ reward and $5.05 \pm 1.27$ cost, moving it leftwards towards the desired cost budget line. In the Pareto plots, this qualitatively shows as USC points that are consistently closer to the upper-left frontier than their USC (NR) counterparts, indicating that refinement successfully resolves high-uncertainty regions that would otherwise remain diffuse or arbitrarily pessimistic. This reinforces the claim that reducing epistemic uncertainty in targeted regions leads to cleaner cost landscapes and tighter constraint satisfaction.

Table 4.2 evaluates how accurately each safety critic method reconstructs the spatial structure of the cost landscape, rather than only reporting scalar episodic cost. Concretely, we compare predicted cost maps against a ground-truth cost map using three metrics. First, Gradient MSE computes the mean-squared error between predicted and true gradient magnitudes. This metric directly captures whether the critic places cost boundaries in the right locations and with the right sharpness, which is crucial because policy updates depend on informative cost gradients: if gradients are spatially misaligned or flattened, the actor receives weak or misleading safety signals even if the average cost is low. Second, Contrast Error measures the difference between the mean predicted cost in hazardous versus safe regions and compares it to the corresponding ground-truth contrast. This quantifies whether the critic preserves the relative separation between safe and hazardous areas (i.e., whether hazards are *visibly* more costly than safe space), which affects how reliably the policy can distinguish safe regions from unsafe ones. Third, Entropy Error compares the Shannon entropy of predicted and ground-truth cost values. This measures whether the critic matches the local spread/complexity of the true cost distribution: overly conservative or diffuse critics tend to blur costs across the map (or saturate them), distorting the distribution of cost values and thereby producing cost landscapes that are either uninformatively flat or uniformly inflated. Details on how these metrics are calculated can be found in Section B.4.1.4.

Across all tasks, USC improves the structural fidelity of the learned cost maps. For Gradient MSE, USC is best on three of four tasks, reducing error on Goal1 to $0.11 \pm 0.06$ compared to $0.42 \pm 0.45$ by SC and $0.29 \pm 0.23$ by CSC; the only exception is Button2 where SC attains the lowest gradient error of $0.16 \pm 0.10$, but USC is close behind with $0.19 \pm 0.14$. In terms of Contrast Error, the differences in results are drastically smaller, so it cannot be said that any method fails or performs notably worse. The CSC agent achieves the lowest error on three of the four tasks. The clearest gains for USC appear in Entropy Error, which directly reflects diffuse or saturated maps: USC reduces and achieves the lowest entropy mismatch on all Safety

Table 4.2: Predictive cost map errors in the Safety Gymnasium tasks compared to ground truth, showing gradient alignment, safe–unsafe contrast, and hazard boundary sharpness.

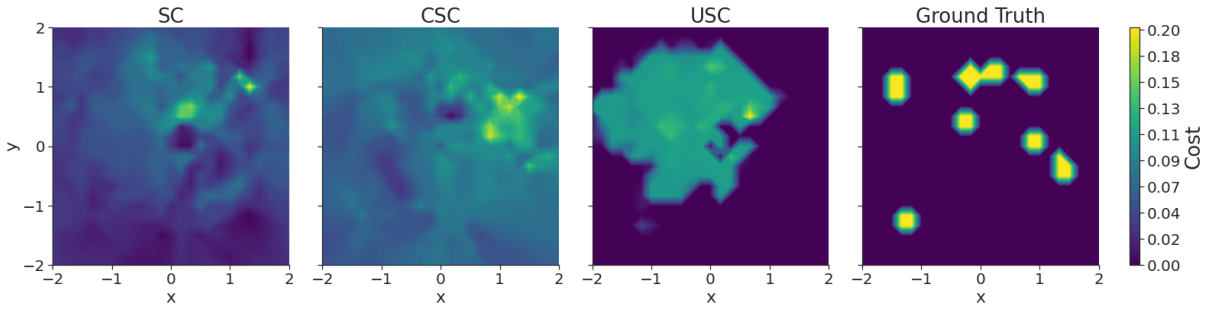
| | | Safety Gymnasium | | | |
|---|---|---|---|---|---|
| **Technique** | **Metric** | **Goal1** ($\chi = 1.0$) | **Goal2** ($\chi = 5.0$) | **Button1** ($\chi = 5.0$) | **Button2** ($\chi = 5.0$) |
| SC | Gradient MSE ↓ | $0.42 \pm 0.45$ | $0.58 \pm 0.29$ | $0.82 \pm 0.57$ | $\mathbf{0.16 \pm 0.10}$ |
| | Contrast Error ↓ | $0.12 \pm 0.09$ | $0.09 \pm 0.09$ | $0.12 \pm 0.03$ | $0.08 \pm 0.10$ |
| | Entropy Error ↓ | $3.04 \pm 0.20$ | $2.50 \pm 0.67$ | $2.98 \pm 0.26$ | $2.49 \pm 0.26$ |
| CSC | Gradient MSE ↓ | $0.29 \pm 0.23$ | $1.95 \pm 1.37$ | $0.27 \pm 0.32$ | $0.27 \pm 0.23$ |
| | Contrast Error ↓ | $\mathbf{0.08 \pm 0.05}$ | $\mathbf{0.04 \pm 0.03}$ | $\mathbf{0.08 \pm 0.08}$ | $0.11 \pm 0.06$ |
| | Entropy Error ↓ | $3.11 \pm 0.26$ | $3.02 \pm 0.27$ | $2.67 \pm 0.57$ | $2.72 \pm 0.52$ |
| USC | Gradient MSE ↓ | $\mathbf{0.11 \pm 0.06}$ | $\mathbf{0.10 \pm 0.06}$ | $\mathbf{0.10 \pm 0.07}$ | $0.19 \pm 0.14$ |
| | Contrast Error ↓ | $0.08 \pm 0.08$ | $0.08 \pm 0.05$ | $0.10 \pm 0.09$ | $\mathbf{0.08 \pm 0.07}$ |
| | Entropy Error ↓ | $\mathbf{1.03 \pm 0.63}$ | $\mathbf{2.16 \pm 0.26}$ | $\mathbf{0.28 \pm 0.20}$ | $\mathbf{1.34 \pm 0.35}$ |

Gymnasium tasks. As expected, CSC typically inflates this error compared to the SC agent. For example, on Button2, SC achieves an error of $2.49 \pm 0.26$ and CSC inflates this to $2.72 \pm 0.52$. USC, then reduces this diffusion significantly, for example on Goal1, SC achieves $3.04 \pm 0.20$, CSC inflates this to $3.11 \pm 0.26$, but USC dramatically reduces this to $1.03 \pm 0.63$.

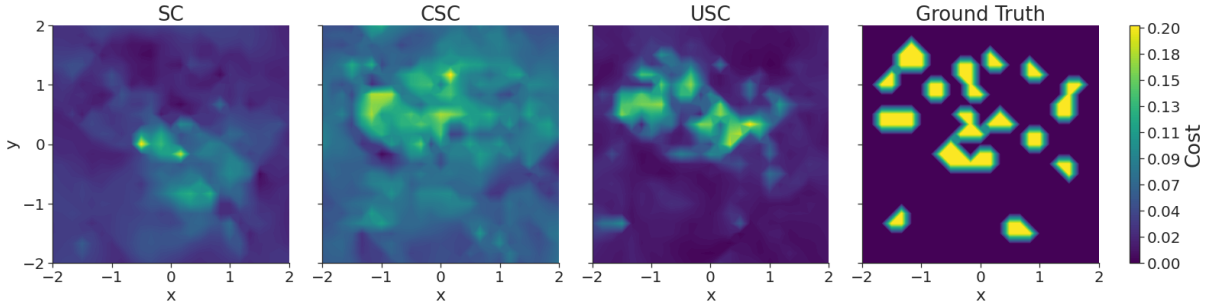
Figure 4.4 provides qualitative cost-map examples for SC, CSC, and USC alongside the ground-truth hazard cost-map, and the visual trends closely mirror the quantitative error patterns in Table 4.2. Across the goal-navigation tasks, SC tends to produce noisy, spatially diffuse maps with scattered hotspots, while CSC often smears elevated cost across broad regions, effectively blurring the boundary between genuinely hazardous regions and safe regions; this is particularly evident in CarGoal2, where CSC yields a large, high-cost blur that obscures the fine-grained hazard structure. USC, in contrast, produces maps with a markedly cleaner background and more coherent high-cost structure, concentrating elevated cost into a smaller accurate regions rather than distributing low-amplitude cost everywhere; qualitatively, this corresponds to the improved gradient fidelity and reduced diffuseness. On the button tasks, the differences are even clearer: SC frequently under-structures the map (either missing large portions of the hazardous set or representing it as weak, inconsistent blobs), whereas CSC can collapse into an overly conservative or poorly localised representation that does not reflect the true hazard geometry. USC produces sharper, higher-contrast cost regions that better reflect where hazards lie and, importantly, avoids assigning widespread intermediate cost across the map. However, reconstruction quality varies across tasks. USC's elevated predictions extend across a broad region in CarGoal1, while much of the hazard structure remains unresolved in CarButton1. Differences in hazard geometry and the coverage of collected experience may contribute to this variation. The quantitative gains therefore coexist with local reconstruction errors, indicating that USC does not recover hazard boundaries perfectly in all environments; it simply improves upon it.

Combined, these results allow us to empirically validate our second hypothesis regarding USC: that the quality of the learned safety critic will be more accurate, avoiding inaccurate or overly diffuse predictive cost-maps.

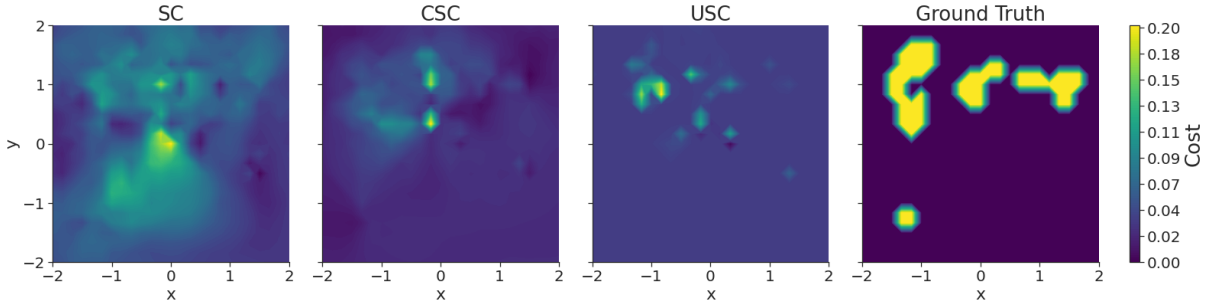
Finally, we profiled the computational overhead of USC's additional components. The Gauss-



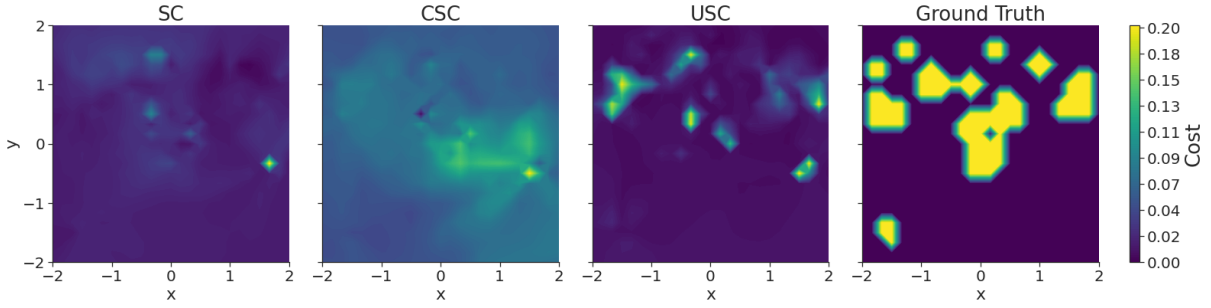

(a) CarGoal1 results




(b) CarGoal2 results




(c) CarButton1 results




(d) CarButton2 results

Figure 4.4: A qualitative comparison of the predictive cost maps from the vanilla Safety Critic (SC), Conservative Safety Critic (CSC), and Uncertain Safety Critic (USC) methods in various Safety-Gymnasium environments against the ground truth cost map.

Newton influence estimation adds approximately ≈50 ms per update, increasing the CSC's update time from around ≈64 ms to ≈112 ms (about 75% overhead), which is consistent with the fact that it requires only a small number of additional Jacobian–vector products. The optional refinement step is more expensive (approximately 1.1s when invoked), increasing the

Table 4.3: Mean episodic reward and episodic cost for comparative methods across evaluated benchmarks during 10 independent deployment runs. Light blue and light green highlighted cells indicate the lowest cost and highest reward for each task, respectively. Light red highlighted cells denote models that do not achieve 100% success rate.

| | | Safety Gymnasium | | | | Gymnasium Robotics | Mujoco |
|---|---|---|---|---|---|---|---|
| **Technique** | **Metric** | **Goal1** ($\chi = 1.0$) | **Goal2** ($\chi = 5.0$) | **Button1** ($\chi = 5.0$) | **Button2** ($\chi = 5.0$) | **FetchReach** ($\chi = 5.0$) | **HalfCheetah** ($\chi = 0.1$) |
| DDPG | Reward ↑ | **7.90 ± 2.40** | 9.13 ± 2.88 | **5.36 ± 1.87** | 9.09 ± 2.69 | 4.74 ± 0.10 | 9.24 ± 2.17 |
| | Cost ↓ | 3.10 ± 4.11 | 41.80 ± 40.59 | **71.40 ± 126.01** | 31.30 ± 24.58 | 3.09 ± 1.22 | 69.80 ± 27.32 |
| PID-Lag | Reward ↑ | 6.50 ± 2.11 | 8.73 ± 3.51 | **0.07 ± 0.19** | **0.07 ± 0.20** | **3.68 ± 3.06** | 5.21 ± 0.47 |
| | Cost ↓ | 4.60 ± 7.00 | 25.10 ± 16.59 | **0.00 ± 0.00** | **0.00 ± 0.00** | **2.61 ± 1.73** | **0.20 ± 0.40** |
| CPO | Reward ↑ | 6.76 ± 2.52 | 8.04 ± 2.98 | 6.22 ± 2.06 | 7.73 ± 4.52 | 4.80 ± 0.11 | 6.57 ± 0.26 |
| | Cost ↓ | 7.80 ± 11.74 | 21.50 ± 20.41 | 43.70 ± 64.24 | 47.30 ± 84.94 | 2.80 ± 1.17 | 10.90 ± 3.59 |
| RCPO | Reward ↑ | 6.74 ± 2.21 | 8.02 ± 3.76 | **−0.01 ± 0.26** | **−5.30 ± 12.53** | 4.72 ± 0.16 | **9.48 ± 1.68** |
| | Cost ↓ | 2.40 ± 4.98 | 33.20 ± 53.33 | **0.00 ± 0.00** | **101.80 ± 147.89** | 2.53 ± 1.61 | 0.80 ± 1.17 |
| SC | Reward ↑ | 7.57 ± 1.89 | **11.48 ± 2.51** | **2.63 ± 2.14** | **0.11 ± 0.17** | **4.81 ± 0.09** | 6.91 ± 0.47 |
| | Cost ↓ | 4.80 ± 5.93 | 31.20 ± 33.28 | **0.00 ± 0.00** | **0.00 ± 0.00** | 3.12 ± 1.09 | 0.50 ± 0.92 |
| CSC | Reward ↑ | 7.21 ± 2.64 | 9.44 ± 2.79 | **3.96 ± 1.62** | **6.33 ± 2.95** | 4.78 ± 0.10 | 8.66 ± 1.57 |
| | Cost ↓ | 2.00 ± 6.00 | 27.00 ± 17.49 | **7.90 ± 15.93** | **14.10 ± 17.39** | 2.98 ± 1.17 | 16.00 ± 10.11 |
| USC (NR) | Reward ↑ | 5.79 ± 2.34 | 9.38 ± 3.95 | **7.65 ± 2.66** | **9.59 ± 2.59** | 4.78 ± 0.13 | 5.74 ± 0.28 |
| | Cost ↓ | 2.60 ± 5.66 | 26.30 ± 26.71 | 22.20 ± 25.96 | 24.00 ± 41.23 | 2.97 ± 1.75 | 0.60 ± 0.49 |
| USC | Reward ↑ | 7.19 ± 1.97 | 9.48 ± 3.30 | 6.41 ± 2.53 | 8.51 ± 4.79 | 4.78 ± 0.07 | 5.44 ± 0.42 |
| | Cost ↓ | **0.70 ± 2.10** | **20.30 ± 24.63** | **15.90 ± 28.40** | **13.60 ± 13.71** | **2.45 ± 1.25** | **0.20 ± 0.40** |

update time from roughly ≈112 ms to ≈1.2s. Since USC with refinement yields the strongest safety–performance gains, refinement can be treated as an adjustable compute–performance knob and need not be executed at every update when resources are constrained.

### 4.3.2 Deployment Results

We next evaluate the deployment behaviour of USC and the comparative methods to determine, when the policy is fixed, if the safety results match those seen in training. We share the same experimental setup and evaluation protocol described in Section 4.3.1, ensuring direct compatibility with core results. Table 4.3 and Table 4.4 report these results, showing both mean episodic reward/cost and task success rates, respectively.

Straight away, a key observation is that deployment induces a pronounced train–deploy gap for several baselines, particularly on the Safety Gymnasium tasks, where the mean episodic cost often increases substantially relative to Table 4.1. This behaviour is consistent with a shift in the visited state–action distribution once exploration noise is removed and rollouts are executed end-to-end under a fixed policy: critic errors that are tolerable during training can compound at test time, leading to higher realised costs even when training-time averages appeared close to the budget.

Across the Safety Gymnasium test suite, several methods achieve deceptively low deployment costs precisely because they fail to solve the task. For example, PID-Lag attains $0.00 \pm 0.00$ cost on both Button1 and Button2, but has 0% success on both tasks, indicating collapse to an overly cautious or inactive policy. However, this is very similar to the behaviour we see during training, where the method achieves a very low reward over both of these tasks. In contrast, RCPO achieves a respectable task reward across Button1 and Button2 during training, but during deployment sees a success rate of 0% and 30% respectively. This is a better example of the train-deploy gap that can sometimes be observed. CPO, USC (NR), and USC are the only

Table 4.4: Success rates (%) for comparative methods across evaluated benchmarks in 10 independent deployment runs. Light red highlighted cells denote models that do not achieve 100% success rate.

| | Safety Gymnasium | | | | Gymnasium Robotics | Mujoco |
|---|---|---|---|---|---|---|
| **Technique** | **Goal1** ($\chi = 1.0$) | **Goal2** ($\chi = 5.0$) | **Button1** ($\chi = 5.0$) | **Button2** ($\chi = 5.0$) | **FetchReach** ($\chi = 5.0$) | **HalfCheetah** ($\chi = 0.1$) |
| DDPG | 100% | 100% | **90**% | 100% | 100% | - |
| PID-Lag | 100% | 100% | **0**% | **0**% | **90**% | - |
| CPO | 100% | 100% | 100% | 100% | 100% | - |
| RCPO | 100% | 100% | **0**% | **30**% | 100% | - |
| SC | 100% | 100% | **50**% | **0**% | 100% | - |
| CSC | 100% | 100% | **70**% | **80**% | 100% | - |
| USC (NR) | 100% | 100% | 100% | 100% | 100% | - |
| USC | 100% | 100% | 100% | 100% | 100% | - |

methods to achieve 100% success on all four Safety Gymnasium tasks, and among these, USC achieves the lowest costs and relatively high rewards. For example, on Button2, CPO averages an extremely high cost of $47.30 \pm 84.94$, USC (NR) achieves the highest reward of $9.59 \pm 2.59$ and a lower average cost of $24.00 \pm 41.23$, however, USC balances these best with a reward of $8.51 \pm 4.78$ and a cost of $13.60 \pm 13.71$.

On the FetchReach environment, most methods maintain high success rates (except for PID-Lag), and the SC agent achieves the highest reward of $4.81 \pm 0.09$. However, nearly every method only falls short of that by a negligible margin. USC achieves the lowest cost of all methods, averaging $2.45 \pm 1.25$ cost across the deployment episodes. On the HalfCheetah environment, DDPG's deployment cost explodes to $69.80 \pm 27.32$, whilst achieving the second highest reward of $9.24 \pm 2.17$. CSC incurs a surprisingly high cost of $16.00 \pm 10.11$ despite attempting to be conservative during training; this may be explained via its diffuse cost landscape. If the critic fails to accurately measure cost, the agent may enter hazardous regions of the state space. USC, by contrast, preserves the same low-cost regime observed during training, with deployment cost $0.20 \pm 0.40$.

Overall, these results aid us in verifying our third and final hypothesis: the deployment behaviour of USC should remain as safe as it was during training. We observed that some methods struggled to reduce the train-deploy gap; however, across all environments, USC was able to maintain a similar strong reward-cost trade-off and even observed the lowest cost across all environments.

### 4.3.3 COPPOL Results

Previously, we established MC-CP (seen in Chapter 3) as a mechanism for calibrated hazard perception, and, in this chapter, we established USC as a mechanism for uncertainty-aware safety critics. The next question is whether their benefits are additive in a single perception-to-policy pipeline. In Section 3.4, we observed through the proposed COPPOL method that we can improve the safety of a downstream policy/RL agent via MC-CP's calibrated hazard maps. Since USC sits alongside the policy, helping shape its decision-making through uncertainty-aware cost prediction, it should be questioned if the combination can further improve safety, more so than the base COPPOL results observed in Section 3.4.2.

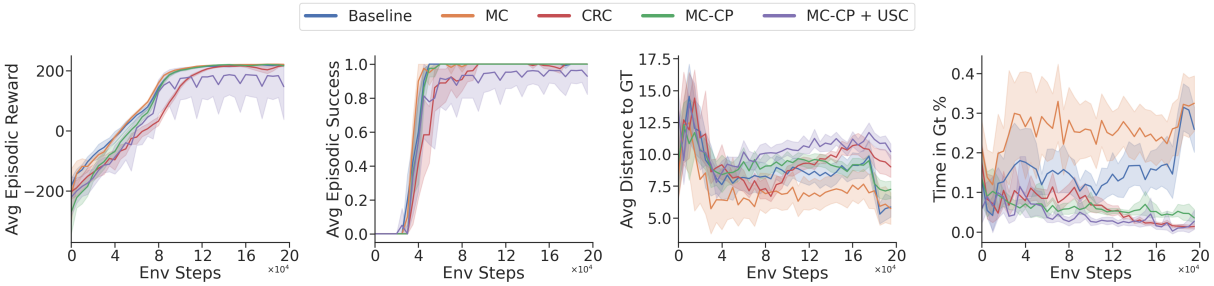



Figure 4.5: Performance of comparative methods on Environment 1 from the DeepMoon dataset (vector view). Curves show average episodic reward, success rate, distance to ground truth hazard, and time spent in ground-truth hazards for Baseline, MC, CRC, MC-CP, and MC-CP + USC (shaded bands indicate variability).

A further motivation for this combined evaluation is that, in the COPPOL evaluated environments, hazard supervision provides enough prior knowledge to define an explicit cost signal in addition to the task reward. In the original COPPOL formulation, safety is encouraged primarily through the hazard-aware reward shaping induced by the calibrated perception outputs; if the agent is within a perceived hazard, then it incurs a negative reward. Here, we additionally exploit the fact that hazards are explicitly represented in the dataset, meaning that *being in a hazard* can be treated as a safety violation signal. This enables a more CMDP-like decomposition in which the agent optimises reward for task completion while simultaneously minimising a hazard-derived cost, allowing USC to play its intended role as a safety critic trained on a dedicated cost channel rather than relying on safety to be implicitly encoded in the reward.

To make this combination meaningful, we first move from an MDP formulation in the original COPPOL instantiation and switch to a CMDP that now has a cost function $C$, and a safety budget $\chi$. Next, to avoid conflating safety with general performance shaping, we modify the environment's scalar feedback relative to the base COPPOL experiments. At each step, the reward $r_t$ is now:

$$r_t = \operatorname{sgn}\big(\mathrm{dist}_t - \mathrm{dist}_{t+1}\big) - \epsilon_{\text{step}} + \beta_{\text{goal}}\,\mathbf{1}\{q_{t+1} = q_{\text{goal}}\} \tag{4.12}$$

Then, the penalty associated with being in predicted hazards is reinterpreted as a cost signal used to train the safety critic:

$$c_t = \mathbf{1}\{q_{t+1} \in \mathcal{M}_{\hat{\lambda}}(x)\} \tag{4.13}$$

This separation is important because it forces the safety mechanism to specialise: the safety critic is trained to predict and penalise hazard exposure, while the actor remains free to optimise the full task objective and does not treat every negative reward component as *unsafe* or, inversely, safety as bad task performance. In effect, MC-CP continues to provide calibrated, uncertainty-aware hazard maps to the policy, while USC adds an uncertainty-aware behavioural safety signal that explicitly targets hazard occupancy.

Figure 4.5 reports the resulting learning curves on Environment 1 from the DeepMoon environments for the original results (originally seen in Figure 3.21) and the new results for MC-CP + USC. For a fair comparison with the reward-centric results, we set $\chi = 0$.

The reward and success curves show that introducing a safety critic into the COPPOL framework does not compromise task learning. MC-CP + USC reaches a similar high-reward performance and near-perfect task success as the original baselines. It does exhibit a small reduction in task performance; however, this is consistent with the behaviour of primal–dual constrained optimisation, where oscillations in the learned dual variable periodically increase the effective weight on the cost term, temporarily prioritising constraint satisfaction over reward maximisation. The slow climb and oscillation in the reward is similar to the slow decrease and oscillation seen with the time spent within the ground truth hazards.

The benefits appear most clearly in the hazard-centric metrics. While all methods reduce distance to hazards early in training as competent behaviour emerges, MC-CP + USC maintains one of the strongest late-stage distances, suggesting that the combined approach more reliably preserves separation from hazardous regions after convergence. Most notably, MC-CP + USC yields the lowest time spent in ground-truth hazards over the course of training, improving over MC-CP alone and substantially outperforming methods without calibrated hazard perception. This pattern is consistent with the intended complementarity: MC-CP reduces perceptual failure by producing calibrated hazard representations, while USC reduces behavioural failure by discouraging risky decisions in uncertain regions of the state–action space through selective conservatism. Taken together, the results suggest that calibrated hazard perception and uncertainty-aware safety critics address distinct sources of uncertainty in a perception-to-policy pipeline, and that coupling them yields additional safety benefits beyond those achieved by COPPOL with MC-CP alone.

## 4.4 Discussion

In this chapter, we introduced the **U**ncertain **S**afety **C**ritic (USC), a novel safety-critic designed to reduce the overly conservative and reckless behaviour that commonly arises in current safety critic variants. USC departs from uniformly pessimistic safety critics by explicitly tying the strength of conservatism to epistemic uncertainty, so that pessimism is concentrated where the critic is poorly supported by data and relaxed in well-covered regions. This design directly targets the diffuse cost landscapes and flattened gradients observed with conservative safety critics, which can stall policy improvement and produce poor reward–safety trade-offs even when constraint satisfaction is the goal. Empirically, across Safety Gymnasium, Gymnasium Robotics, and Mujoco benchmarks, USC consistently achieved the strongest reward-cost trade-off, and it produced cost maps with improved gradient structure and reduced diffuseness relative to both standard and conservative baselines. These results support the central claim that uncertainty-aware modulation and targeted refinement can preserve the protective benefits of conservative estimation without inheriting its most damaging optimisation pathologies.

### 4.4.1 Applicability and Impact

USC is designed as a general mechanism for improving safety critics in cost-based safe RL, and its applicability extends beyond the specific environments evaluated in this chapter. The method assumes only that a cost signal can be defined and observed during interaction, which is standard in CMDP-style formulations and is often natural in safety-critical domains where proximity to hazards, violation margins, or severity can be quantified. Under this assumption, USC can be applied as a drop-in replacement for existing cost critics, with no requirement for environment-specific shielding logic or formal models of dynamics. Since safety-critics are widely applicable to any RL algorithm, USC thus inherits the applicability of existing safety critics.

The main practical impact of USC lies in how it reshapes the optimisation landscape faced by the policy. By preventing the critic from becoming uniformly inflated in safe regions, USC preserves informative gradients that allow the agent to continue improving reward under tight constraints, rather than collapsing into overly cautious behaviour. This effect is particularly important in complex environments where the policy must navigate narrow safe corridors or make precise trade-offs near hazard boundaries. Consequently, USC provides a scalable alternative to existing conservative safety critics, improving the critic structure and downstream policy performance under tight safety budgets.

### 4.4.2 Limitations

While USC addresses several shortcomings of existing safety critics, it also introduces a few potential limitations.

First, USC relies on the availability of a meaningful cost function. While such costs are natural in many domains, they may not be available in domains where the safety constraint is unknown a priori. In such cases, USC can only be applied once an appropriate cost surrogate has been engineered or learned, and the quality of this surrogate directly bounds the usefulness of the resulting safety critic.

Second, USC introduces additional computational overhead relative to a standard critic. Influence-based uncertainty estimation requires extra gradient computations to approximate the Gauss–Newton curvature, and the refinement step adds further cost when invoked. Although these components are tractable in the benchmark settings considered, the overhead may become non-trivial for very large critics, high-dimensional state–action spaces, or time-constrained training regimes. In practice, this motivates treating refinement as an optional component that can be applied intermittently when compute is limited, trading wall-clock time for improved critic structure and downstream safety–performance balance.

# Chapter 5

# Adaptive Shielding for Black-Box Environments

This chapter introduces a data-driven approach for shielding an RL agent against unknown safety constraints during training through a learned latent safety representation to discriminate unsafe regions of the state–action space. We evaluate the proposed shielding mechanism across a range of continuous-control environments. This chapter describes the motivations for the approach in Section 5.1. We then describe the methodology of the approach in Section 5.2, and fully evaluate it across various safety-critical environments in Section 5.3. Finally, we summarise the chapter in Section 5.4, where we discuss the method's impact, limitations, and areas for future work.

## 5.1 Motivation

The preceding chapters address two major failure points that arise when training RL in safety-critical settings. Chapter 3 focused on perception, showing that even a well-designed downstream safety mechanism can fail when hazards are not reliably represented in the agent's state, and that uncertainty-aware statistically valid perception can expose and partially mitigate this mismatch. Chapter 4 then moved to decision-making, analysing the role of learned safety critics and showing how over-conservatism can distort reward–safety trade-offs when the critic becomes pessimistic in regions that are, in fact, safe. Taken together, these chapters have mitigated safety bottlenecks in the perception-decision-control pipeline.

This chapter focuses on the third component: control. In embodied agents, decision-making does not act directly on the world; it produces proposed actions that must be translated into physical commands and applied to a system whose dynamics and operating conditions may be only partially understood. In many practical architectures, this final stage is guarded by explicit enforcement mechanisms, such as action filters/shields [193], that monitor proposed actions and modify or veto them when they would violate safety requirements. In the ideal case, if the relevant constraints are known and can be evaluated, the control layer can prevent clear violations even while the policy continues learning. This is the perspective implicitly

assumed by much of classical shielding: safety is enforced by checking a proposed action against a specification and, if it is inadmissible, modifying or replacing it with a permissible action.

However, as highlighted in the previous chapter, one source of failure in the decision-making module is misaligned or poorly specified safety constraints, which can, in principle, be reduced through careful design and domain modelling. The issue, which we address in this chapter, begins where that assumption breaks down. In many realistic deployments, safety requirements are not available as a precise, closed-form constraint set. They may be implicit, context-dependent, high-dimensional, or simply too costly to specify exhaustively. Moreover, the environment is treated as a black box: the agent and shield can observe transitions, rewards, and outcomes indicating success or failure, but are not provided with a dynamics model or an explicit safety specification. Safety-relevant structure must therefore be inferred from interaction rather than checked against known constraints. In such settings, *better modelling* at design time is not merely an engineering task left to the user; it is a structural limitation of the problem. The system still needs an execution-time safety mechanism, but the mechanism cannot rely on a complete and correct statement of what *unsafe* means.

This limitation becomes particularly critical during training. Learning-based safety mechanisms, most notably safety critics [211] and cost-aware policies [169], depend on observing safety-relevant feedback. In the simplest and most common formulation, the feedback is only revealed when a violation occurs: the agent discovers that a region of behaviour is unsafe by entering it and suffering a safety event. Even when costs are shaped more densely, meaningful supervision is still concentrated around the boundary between safe and unsafe behaviour, and obtaining that supervision requires visiting the very states and actions that the training process should avoid. This creates an inherent tension: to learn an accurate internal model of risk, the agent must collect experience that includes unsafe outcomes; yet in safety-critical domains, unsafe outcomes are precisely what should be prevented. Put differently, a purely learning-driven safety mechanism is often *data hungry*, exactly where data collection is most unacceptable.

If the safety constraints were fully known, the control layer could, at least in principle, resolve this tension by preventing truly unsafe execution while still providing the learning system with a meaningful training signal. For example, a shield could block a proposed unsafe action and substitute it with a safe alternative, while the learning algorithm is trained with an informative penalty that reflects what would have happened without intervention. But this strategy depends on having a trustworthy way to label the proposed behaviour as unsafe in the first place. When the constraints are unknown or only partially understood, the control layer cannot reliably decide when to intervene, and the learning system cannot rely on consistent cost feedback. The result is a second, more subtle training-time failure mode: even with accurate perception and a well-shaped safety critic, the overall system can still execute unsafe actions simply because neither the agent nor the controller has a sufficiently precise operational notion of the constraint boundary early in training.

Building on this perspective, this chapter proposes ADVICE (**AD**apti**V**e Sh**I**elding with a **C**ontrastive Auto**E**ncoder): an adaptive, data-driven, post-shielding mechanism that uses repre-

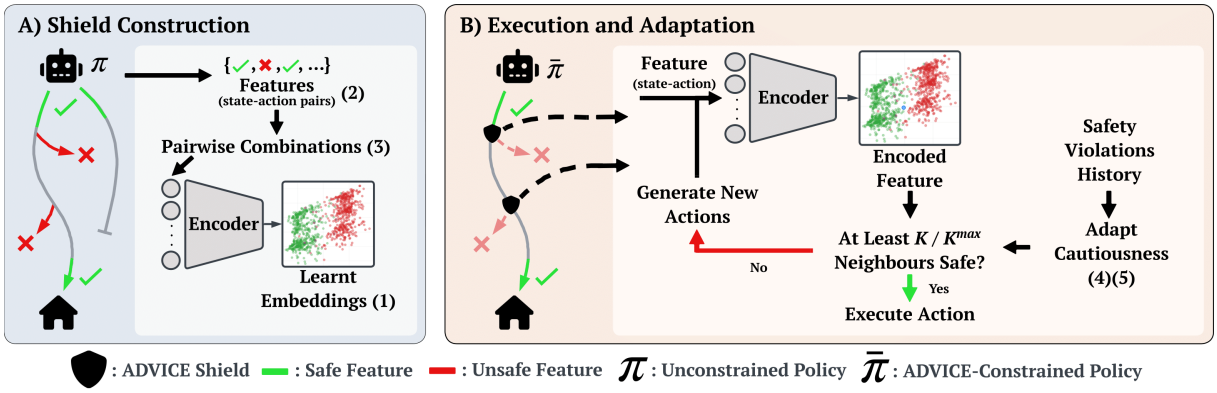



Figure 5.1: High-level ADVICE overview, showing its shield construction (A), and execution and adaptation (B) stages. During shield construction, safe and unsafe features (signifying accepting and non-accepting terminal states, respectively) are collected and used by a contrastive autoencoder to learn latent representations (embeddings). During shield execution and adaptation, a new feature is encoded into the contrastive latent space, and the nearest neighbours guide its classification as safe or unsafe. ADVICE's cautiousness is adapted dynamically based on the RL agent's recent safety performance.

sentation learning to infer a latent notion of safety directly from interaction data. Rather than assuming an explicit constraint specification is known at design time, like in previous safety shields [193, 217], ADVICE learns to discriminate safe and unsafe behaviour from experience in a way that is compatible with black-box environments. This enables the shield to filter, correct, or veto proposed actions in regions where experience indicates elevated risk. In doing so, the method targets the control-stage bottleneck that remains when perception is reliable, and decision-making is well-shaped, but where domain modelling is not feasible, and safety constraints cannot be assumed to be known in advance.

Whereas USC guides policy learning through estimates of cumulative safety cost, ADVICE learns from safe and unsafe experience to filter proposed actions before execution without requiring an explicit cost function. The methods could therefore complement one another through cost-guided policy improvement and additional action filtering, although their joint use is not evaluated in this thesis.

## 5.2 ADVICE

Building on the conceptual overview introduced above, this section formalises the ADVICE methodology and details its components. First, we present a high-level overview, visualised in Figure 5.1, to illustrate its integration into the standard RL training loop.

The ADVICE post-shielding approach empowers safe RL exploration in black-box environments without requiring a system model, domain knowledge or handcrafted rules via a contrastive autoencoder (CA) model that can efficiently distinguish between safe and unsafe features from the feature space $F : S \times A$ (representing all possible state-action pairs). A feature $f_t = (s_t, a_t)$,

$f_t \in F$, denotes the state-action pair at timestep $t$. Throughout this chapter, a feature refers to this complete state-action pair, rather than an individual component of the input vector; its encoded representation is termed a latent embedding. The CA model leverages a unique loss function where similar and dissimilar features are compared, enabling the systematic identification of meaningful latent feature representations. ADVICE employs these latent representations and trains an unsupervised nearest neighbours model in the learnt embedding space, thus enabling the classification of new features. Formally, the devised ADVICE shield is defined by a function $\phi : F \times \mathbb{Z}_+ \to A$, such that at time step $t$ during the execution of an episode, ADVICE evaluates the agent's desired action $a_t$ by computing $\phi(f_t, K)$, where $K$ is a parameter that controls the risk aversion level of the shield. This function returns $a_t$ if the desired action is deemed safe (thus endorsing its execution), or returns (and enforces the execution of) another, safe action $a'_t$ instead. ADVICE considers the agent's performance and automatically adapts the value of $K$, thus supporting the dynamic calibration of ADVICE's cautiousness level during learning.

In the remainder of this section, we formally define the ADVICE construction in Section 5.2.1, and then the execution and adaptation of the shield in Section 5.2.2.

### 5.2.1 Shield Construction

As a black-box post-shield, ADVICE does not rely on any domain knowledge about the RL agent or its environment. Instead, the shield is synthesised directly from experience by learning a representation in which safety-relevant structure becomes separable. Unlike USC (Section 4), which estimates cumulative safety cost from an explicit cost signal to guide policy optimisation, ADVICE uses observed outcomes to classify proposed state-action pairs as safe or unsafe. This difference motivates the contrastive autoencoder: its contrastive objective brings features with matching safety labels closer together and separates those with opposing labels, while reconstruction preserves information from the inputs. A nearest-neighbour classifier then queries this latent space to determine whether a proposed action should be allowed or overridden, without requiring an explicit cost function.

The shield construction stage begins with collecting a feature set $F_E$ of state-action pairs during an initial, unshielded interaction period of $E_{max}$ episodes, where the RL agent is allowed to interact with and collect experience from its environment. Because the setting of ADVICE is explicitly black-box, ADVICE has no prior basis for deciding which regions of the state–action space are safe; collecting experience is therefore necessary to obtain any empirical signal that can ground a learned notion of safety. Furthermore, the learned shield must generalise across high-dimensional continuous spaces: it is not sufficient to memorise individual unsafe tuples [205], and the representation must capture similarity structures so that nearby but unseen features can be classified reliably later. This also relaxes the safety definition from assuming a cost-based MDP framework (like the approaches from [170, 213, 260]) to the standard finite-horizon MDP (as in [193, 196]), thereby enabling greater generalisability.

Given the features $f_t \in F_E$ collected during the $E_{max}$ unshielded episodes, ADVICE assigns to

one of three categories via a labelling function $g(\cdot)$. The label is determined by the outcome observed after executing the action $a_t$, i.e., by inspecting the next state $s_{t+1}$ and whether the next state is terminal (accepting or non-accepting):

$$g(f_t) = \begin{cases} \text{safe}, & \text{if } t = 0 \text{ or } s_{t+1} \text{ is an accepting state} \\ \text{unsafe}, & \text{if } s_{t+1} \text{ is a non-accepting terminal state} \\ \text{inconclusive}, & \text{otherwise} \end{cases} \tag{5.1}$$

This definition captures the minimal safety information reliably available in a black-box RL loop. Features that immediately precede an accepting terminal transition are treated as safe because they are empirically associated with successful task termination. Features that immediately precede a failure termination are treated as unsafe because they are empirically associated with hazardous or invalid outcomes. All remaining features are marked inconclusive because, without an explicit constraint function or dense cost, the episode has not yet revealed whether those intermediate transitions lie on a genuinely safe trajectory or whether they will eventually lead to failure. Importantly, assigning safety labels to trajectory prefixes can be fundamentally ambiguous: two rollouts may be identical for many time steps and differ only at the final decision, where one last action (e.g., turning right) reaches the goal, and another (e.g., turning left) transitions into a failure terminal state. In such cases, treating the shared prefix as definitively *safe* or *unsafe* would inject label noise, since the same intermediate feature can appear in both successful and failing executions depending on how the episode resolves. ADVICE therefore anchors supervision to the most reliable point of evidence: terminal outcomes, while leaving intermediate, non-terminal features inconclusive until sufficient experience makes their risk profile separable in the learned latent space. The special case $t = 0$ treats the initial state–action context as safe, which anchors the *safe* partition and reflects the standard assumption that environment reset places the agent in a feasible initial configuration.

The set $F_E$ is then partitioned into disjoint subsets of safe, unsafe and inconclusive features: $F_{safe} = \{f_t \in F_E | g(f_t) = \text{safe}\}$, $F_{unsafe} = \{f_t \in F_E | g(f_t) = \text{unsafe}\}$, and $F_{inc} = F_E \backslash (F_{safe} \cup F_{unsafe})$. This partitioning facilitates the contrastive representation learning of: (i) similar feature pairs (i.e., two safe or two unsafe features); and (ii) dissimilar feature pairs (i.e., a safe and an unsafe feature), allowing the CA model to discern between safe and unsafe features effectively. This key ADVICE characteristic allows it to focus on finding meaningful representations in a lower-dimensional latent space that adequately reflects the problem domain similarities and differences. We assemble a training dataset $\mathcal{D}_{\text{train}}$ for the CA model by automatically labelling each combination of distinct features from $F_{safe} \cup F_{unsafe}$ as similar (1) or dissimilar (0):

$$\begin{aligned} \mathcal{D}_{\text{train}} = &\{((f_t, f_{t'}), 1) \mid f_t, f_{t'} \in F_{safe} \cup F_{unsafe} \wedge g(f_t) = g(f_{t'}) \wedge f_t \neq f_{t'}\} \cup \\ &\{((f_t, f_{t'}), 0) \mid f_t, f_{t'} \in F_{safe} \cup F_{unsafe} \wedge g(f_t) \neq g(f_{t'})\} \end{aligned} \tag{5.2}$$

Using this dataset, let $\mathcal{F}_{\text{encoder};\, \theta_{\text{encoder}}}$ be an encoder neural network that maps a feature to a latent embedding $\mathfrak{v}$, and let $\mathcal{F}_{\text{decoder};\, \theta_{\text{decoder}}}$ be a decoder neural network producing a reconstruction of the feature:

$$\begin{aligned} \mathfrak{v} &= \mathcal{F}_{\text{encoder};\,\theta_{\text{encoder}}}(f) \\ \hat{f} &= \mathcal{F}_{\text{decoder};\,\theta_{\text{decoder}}}(\mathfrak{v}) \end{aligned} \tag{5.3}$$

The CA is then the combination of the encoder and decoder networks. Formally, let $\mathcal{F}_{\text{AE};\,\theta_{\text{AE}}} := \mathcal{F}_{\text{encoder};\,\theta_{\text{encoder}}} \circ \mathcal{F}_{\text{decoder};\,\theta_{\text{decoder}}}$ be the autoencoder network with parameters $\theta_{\text{AE}} := (\theta_{\text{encoder}}, \theta_{\text{decoder}})$. For any feature $f_t$, the mapping to the reconstruction is defined as $\hat{f}_t = \mathcal{F}_{\text{AE};\,\theta_{\text{AE}}}(f_t)$.

ADVICE trains the CA with two coupled objectives: a mean squared error (MSE) loss that measures how accurately the model reconstructs input features, and a contrastive loss (CL) that refines the latent space by clustering similar features and separating dissimilar ones based on Euclidean distance. This encourages cohesion among similar features and separation among dissimilar ones.

For a training sample $((x_i, x_j), y)$ from the training dataset $\mathcal{D}_{\text{train}}$, where $y \in \{0, 1\}$ and $x = f$, the contrastive loss is:

$$\mathcal{L}_{\text{CL}}(x_i, x_j, y, \theta_{\text{AE}}) = y \cdot \|\mathfrak{v}_i - \mathfrak{v}_j\|^2 + (1 - y) \cdot \max\left(0, m - \|\mathfrak{v}_i - \mathfrak{v}_j\|^2\right) \tag{5.4}$$

where $m > 0$ is a margin parameter. When $y = 1$, the loss encourages the embeddings to move closer together. When $y = 0$, the loss encourages the squared distance between embeddings to be at least $m$; otherwise, a penalty is incurred. The margin prevents trivial solutions where all embeddings collapse to a point, and it explicitly enforces separation between safe and unsafe clusters once the model has enough evidence to do so. In parallel, the MSE between each input feature and its reconstruction is defined as:

$$\mathcal{L}_{\text{MSE}}(x_i, x_j, \theta_{\text{AE}}) = \|x_i - \hat{x_i}\|^2 + \|x_j - \hat{x_j}\|^2 \tag{5.5}$$

and averaged over training samples. While $\mathcal{L}_{\text{CL}}$ shapes the relative geometry, $\mathcal{L}_{\text{MSE}}$ stabilises training and ensures the encoder does not discard information needed to embed unseen features consistently. The combined objective used in training is therefore:

$$\mathcal{L}_{\text{total}}(x_i, x_j, y, \theta_{\text{AE}}) = \mathcal{L}_{\text{MSE}} + \lambda_{\text{CL}} \cdot \mathcal{L}_{\text{CL}} \tag{5.6}$$

with $\lambda_{\text{CL}}$ controlling the trade-off between geometry shaping and reconstruction fidelity. Optimising these losses yields an encoder that both (i) preserves the core structure of state–action features and (ii) arranges them so that *safe* and *unsafe* form separable regions in latent space. This is precisely the representation property needed for an inference-time safety test: the classifier will only be reliable if local neighbourhoods in latent space correspond to meaningful safety similarity.

Once trained, ADVICE encodes all labelled features from the construction stage and embeds a nearest-neighbours model directly in the learned latent space. Operationally, the shield stores the latent vectors for the encoded training features along with their safe/unsafe labels, and uses

efficient nearest-neighbour lookup to retrieve local neighbourhoods around a new encoded query feature. This non-parametric choice is deliberate: it avoids fitting an additional parametric decision boundary that could be brittle under distributional shift, and it exploits the main artefact ADVICE has learned; the geometry of the embedding space itself.

### 5.2.2 Execution and Adaptation

Upon completing the shield construction phase, ADVICE, similar to [205], is deployed as a post-shield inside the standard RL loop. In this configuration, the underlying RL agent (policy and learning algorithm) remains unchanged: at each time step, it proposes an action according to its current policy, while ADVICE sits *downstream* and decides whether that proposed action should actually be executed in the environment. This design mirrors the role of classical action filters [261] and post-shields [193], but replaces an explicit, hand-coded safety test with a learned, latent-space safety discriminator. The ADVICE execution and adaptation stage can be seen in Algorithm 7, which shows the standard RL loop with the ADVICE extension.

Assuming the shield has been built partway through, training proceeds for a maximum of $E$ episodes. During an episode $e$, at each time step $t$, the agent observes the current state $s_t$ and proposes an action $a_t \sim \pi(s_t)$. In an unshielded system, this action would be executed directly, which could lead to an unsafe action. Using ADVICE, the action is first assessed by the shield.

The key part of ADVICE's inference is the *IsSafe* function (lines 25-28), which is based on the learned latent geometry. Given a candidate feature $f = (s, a)$, ADVICE encodes it using the trained contrastive encoder to obtain its latent embedding $\mathfrak{v} = \mathcal{F}_{\text{encoder}}(f)$. To classify the new feature in the latent space as either *safe* or *unsafe*, ADVICE uses the nearest neighbours model trained in the shield construction stage and retrieves the $K_{max}$ nearest latent points from the stored training embeddings. It then filters those neighbours to retain only those labelled *safe*, producing a set of safe neighbours $N_{\mathfrak{v}}$. The function then returns a boolean: true if $|N_{\mathfrak{v}}|$ contains at least $K$ neighbouring samples, thus providing sufficient empirical support for safety; otherwise, the action is deemed unsafe. Rather than fitting an explicit classifier head that could be brittle or miscalibrated, ADVICE relies on the geometry learned by the contrastive objective: if the encoder has successfully clustered safe outcomes and separated unsafe outcomes, then nearest-neighbour structure becomes a meaningful proxy for safety.

**Theorem 5.1.** *The probability of ADVICE misclassifying a feature is bounded by* $\exp(-\frac{m^2}{2\sigma^2})$*, where* $m$ *is the contrastive separation margin and* $\sigma^2$ *is the variance of the Gaussian noise within the latent space.*

**Theorem 5.2.** *The probability of ADVICE misclassifying a feature decreases exponentially with improved data entropy and is bounded by* $\exp(-\frac{\mathcal{H}(F_E)}{2\sigma^2})$.

The corresponding proofs are in Appendix C.1. If the proposed action is deemed safe, it is executed. If the agent-selected action $a_t$ is deemed unsafe, ADVICE intervenes and generates a set of valid candidate actions $A_{f_t}$ by quantising each action dimension $A^d$, where $d \in D$ is the dimension, and extracting the Cartesian product across the $D$ action dimensions. ADVICE then filters $A_{f_t}$ into a subset $A'_{f_t}$ containing only actions that are both (i) valid (within

**Algorithm 7** ADVICE Execution and Adaptation

**Require:** Contrastive encoder $\mathcal{F}_{\text{encoder}}$, Max neighbours $K_{max}$, Safety threshold $K$, Safety violation indicators $(z_i)_{i=1}^{E}$, Recent/distant time windows $h_r$, $h_d$, Shield construction episodes $E_{max}$, Max episodes $E$, Max timesteps $T$, Policy $\pi$

**for** $e = E_{max} + 1, E_{max} + 2, \ldots, E$ **do**
  $s_1 \leftarrow$ Observe(), $t \leftarrow 1$
  **while** $t \leqslant T \wedge s_t$ not terminal **do**
    $a_t \leftarrow \pi(s_t)$
    $f_t \leftarrow (s_t, a_t)$
    **if** !IsSafe$(f_t, K, K_{max}, \mathcal{F}_{\text{encoder}})$ **then**
      $A_{f_t} \leftarrow \left\{ \times_{d \in D} A^d_{f_t} \mid \forall d \in D \,.\, (A^d_{f_t} \subset A^d) \right\}$
      $A'_{f_t} \leftarrow \{a \in A_{f_t} \mid \text{IsSafe}((s_t, a), K, K_{max}, \mathcal{F}_{\text{encoder}})\}$
      **if** $A'_{f_t} \neq \varnothing$ **then**
        $a_t \leftarrow \arg\max_{a \in A'_{f_t}} Q_R(s_t, a)$
      **end if**
    **end if**
    $r_t, s_{t+1} \leftarrow$ Execute$(a_t)$
    $\pi \leftarrow$ UpdatePolicy$(a_t, s_t, r_t, s_{t+1})$
    $t \leftarrow t + 1$
  **end while**
  **if** $s_t$ is non-accepting terminal **then**
    $z_e \leftarrow 1$
  **else**
    $z_e \leftarrow 0$
  **end if**
  $K \leftarrow$ UpdateK$(K, K_{max}, (z_i)_{i=1}^{e}, h_d, h_r)$
**end for**

**function** IsSafe$(f, K, K_{max}, \mathcal{F}_{\text{encoder}})$ )
  $\mathfrak{v} = \mathcal{F}_{\text{encoder}}(f)$
  $N_{\mathfrak{v}} \leftarrow$ GetSafeNeighbours$(\mathfrak{v}, K_{max})$
**return** $|N_{\mathfrak{v}}| \geqslant K$ {Action safe (unsafe otherwise)}

the environment's admissible bounds and any immediate feasibility checks) and (ii) predicted safe by the *IsSafe* function when paired with the current state $s_t$ (that is, actions $a$ such that *IsSafe*$(s_t, a)$ returns true). If $A'_{f_t}$ is non-empty, ADVICE selects among these safe actions using the agent's own reward model. Concretely, each candidate action is evaluated under the agent's value function $Q_R(s_t, a)$, and the candidate with the highest predicted return is chosen as the replacement. This ensures that the shield does not behave as a purely safety-maximising controller that ignores the task, thereby preserving the agent's optimisation objective as much as possible under the safety constraint. In the case that $A'_{f_t}$ is empty, then no candidate action is predicted safe by the current latent model. In that case, ADVICE executes the original action

$a_t$. If the shield cannot identify any action that it can justify as safer, then overriding the agent becomes arbitrary and may degrade learning or induce unintended behaviours. Retaining $a_t$ is therefore treated as the least harmful option given the shield's current uncertainty.

A distinctive component of ADVICE is that the user-defined safety threshold $K$ is not static. ADVICE, optionally, can adapt its cautiousness over time based on recent safety performance. The agent's safety performance is not consistent; realistically, the agent should slowly achieve better safety performance over time as the policy learns to avoid terminal states that prematurely truncate episodes and therefore yield lower expected return. Evidently, the appropriate level of shielding cautiousness depends on the agent's current behaviour and competence. Early in training, the agent tends to be erratic and unsafe; later, it may behave safely for longer periods and benefit from less restrictive shielding to improve exploration and performance. A fixed $K$ cannot accommodate both regimes well, so ADVICE adjusts $K$ online via a double sliding window, commonly used in the field of anomaly detection [262, 263]. By comparing the current trend of safety violations against a broader historical view, ADVICE can discern whether the current trend deviates from the pattern seen historically. This analysis informs ADVICE to strengthen or relax its cautiousness level $K$ during the execution of an episode. To do this, ADVICE tracks whether each episode ends with a safety violation. Let $z_e$ be a binary indicator for episode $e$, where $z_e = 1$ if the episode ends in a non-accepting terminal (i.e., a safety violation), and $z_e = 0$ otherwise. At the end of each episode $e$, the adaptation function is given by:

$$K = \begin{cases} \min(K+1, K_{max}), & \text{if } \mu_r > \mu_d + \sigma_d \\ \max(K-1, \lceil K_{max}/2 \rceil), & \text{if } \mu_r < \mu_d - \sigma_d \\ K, & \text{otherwise} \end{cases} \tag{5.7}$$

where

$$\mu_r = \frac{\sum_{i=e-h_r+1}^{e} z_i}{h_r}, \quad \mu_d = \frac{\sum_{i=e-h_d+1}^{e} z_i}{h_d}, \quad \sigma_d = \sqrt{\mu_d(1-\mu_d)} \tag{5.8}$$

represents the moving probabilities of an episode ending with a safety violation $z_e$ in the recent $h_r$ and distant $h_d$ past ($h_d > h_r$), and the standard deviation for the former, respectively.

If the $\mu_r$ value exceeds the sum of $\mu_d + \sigma_d$, this signifies that the agent incurs safety violations more often than expected. Consequently, $K$ is automatically incremented to adopt a more cautious position. Conversely, if $\mu_r$ is less than $\mu_d - \sigma_d$, it is an indication that the agent acts more cautiously than expected. As a result, $K$ is automatically reduced to allow the agent to explore more freely. The safety threshold $K$ remains the same for any other scenario. ADVICE deliberately only considers one standard deviation, as two or more standard deviations would make the adaptation slow to respond to emerging safety risks.

**Theorem 5.3.** *The expected regret of ADVICE with a fixed $K$ is upper bounded by a function of order $O\left(\frac{1}{\sqrt{E}}\right) + O\left((E - E_{max}) \cdot \Delta Q \cdot \exp\left(-\frac{m^2}{2\sigma^2}\right) \cdot \frac{1}{\sqrt{K}}\right)$.*

**Theorem 5.4.** *The expected long-term regret of ADVICE with an adaptive $K$ is lower than that of ADVICE with a fixed $K$, and is upper bounded by a function of order* $O\left(\frac{1}{\sqrt{E}}\right) + O\left((E - E_{max}) \cdot \Delta Q \cdot \exp\left(-\frac{m^2}{2\sigma^2}\right) \cdot \frac{1}{\sqrt{K*}}\right) + O\left(\frac{1}{E-E_{max}} \sum_{e=1}^{E-E_{max}} |K_e - K^*| \cdot \Delta Q_{K_e}\right)$.

The corresponding proofs are in Appendix C.1. Adaptive $K$ adjusts the evidence required to accept an action within the existing latent representation; it does not update the encoder or correct inaccurate safety labels. It can tighten shielding when recent violations exceed the historical tolerance band and relax it when they fall below that band. However, the rule responds to relative changes rather than enforcing an absolute violation limit, so a persistently high but stable violation rate may leave $K$ unchanged. Adaptation also follows observed outcomes and therefore cannot prevent the violations that first reveal a change. Its effectiveness remains dependent on the learned representation and the availability of suitable replacement actions.

The user-defined hyperparameters introduced by ADVICE, the contrastive margin $m$, the unshielded episodes $E_{max}$, the maximum neighbours $K_{max}$, the safety threshold $K$, and the recent/distant time windows $h_r$ and $h_d$, control the sensitivity/conservativeness of the method. Any remaining hyperparameters (e.g., learning rates, optimiser settings) are inherited from the underlying RL agent and retain their standard functional behaviour; critically, ADVICE does not modify how those inherited parameters influence policy learning, but instead controls which actions are executed during interaction with the environment. The contrastive margin $m$ controls how strongly the latent space is shaped to separate safe and unsafe features. As $m \to 0$, the separation requirement for dissimilar pairs becomes weak, and the embedding may not develop a clear safety boundary. A $m \to \infty$ enforces a stronger gap between safe and unsafe clusters; however, excessively large margins can over-constrain training, potentially degrading reconstruction and harming generalisation. The unshielded period $E_{max}$ determines the amount of evidence available for constructing the shield. Smaller $E_{max}$ reduces exposure to risk during the bootstrap phase but produces fewer terminally labelled examples, which can bias or sparsify the dataset and lead to an unreliable latent geometry early on. Increasing $E_{max}$ improves coverage and diversity of labelled outcomes, stabilising representation learning and downstream safety discrimination, but at the explicit cost of allowing more unprotected interaction before shielding begins. Thus, $E_{max}$ primarily trades initial risk for later shield quality. The fixed ADVICE (not using adaptation) neighbour parameters $K_{max}$ and $K$ determine how conservative the shield is. Increasing $K_{max}$ makes the decision less sensitive to individual noisy neighbours by averaging over a broader local region, whereas a smaller $K_{max}$ yields a more local but potentially higher-variance decision near the boundary. Increasing $K$ makes ADVICE more risk-averse by requiring stronger local support from safe neighbours before an action is permitted, while decreasing $K$ relaxes the shield and favours exploration. Values of $K$ below $\lceil K_{max}/2 \rceil$ should be avoided, as they can admit actions that are locally supported by more unsafe than safe neighbours, contradicting the intended semantics of the safety vote. Setting $K = 0$ effectively removes the shield entirely, reverting to an unshielded RL agent. Finally, the adaptation windows $h_r$ and $h_d$ control how quickly ADVICE (when using adaptation and not a fixed $K$) adjusts cautiousness over training. A smaller $h_r$ makes the method respond rapidly to emerging safety violations (by tightening the threshold), while a larger $h_r$ smooths stochastic fluctuations but delays response. The distant

window $h_d$ provides a longer-term baseline: larger $h_d$ stabilises the historical estimate, whereas smaller $h_d$ causes the baseline to track short-term behaviour more closely, reducing the contrast between *recent* and *historical* trends and therefore reducing adaptation sensitivity.

Finally, it is useful to consider limiting cases in which ADVICE becomes ineffective and examine how the method behaves. The first limiting case is when the learned latent representation fails to become safety-discriminative. If the contrastive autoencoder does not produce a latent space in which safe and unsafe features are meaningfully separated (e.g., due to insufficient or biased terminal supervision, heavy stochasticity, or feature aliasing), then the KNN-based safety test will be unreliable. In this regime, ADVICE naturally degenerates towards an almost non-informative filter: actions may be accepted or rejected based on noisy neighbourhood structure, and the shield's decisions become effectively indistinguishable from a weak heuristic. The second limiting case arises even when the safety test flags an action as unsafe, but no safe alternative can be identified within the candidate set generated. When the filtered safe set is empty, ADVICE retains the original policy action. This is a deliberate fallback: with no alternative predicted to be safe, overriding the agent would be arbitrary and may degrade both learning and performance. In this sense, ADVICE is backwards compatible with standard RL execution: when the learned representation or candidate search is unhelpful, the method collapses gracefully to unshielded behaviour rather than forcing unsafe or low-value substitutions.

## 5.3 Evaluation

This section empirically evaluates the proposed ADVICE method across a suite of continuous-control and safety benchmark tasks. The experimental evaluation aims to verify that ADVICE achieves the objectives established throughout Section 5.2: shielding an RL agent against unknown safety constraints during training by learning a data-driven latent representation that supports reliable safety discrimination at execution time. The experiments, therefore, assess both whether ADVICE improves safety during exploration and whether the learned representation is sufficiently structured to support robust, generalisable shielding decisions. To this end, the following hypotheses are investigated.

The first hypothesis is that ADVICE improves safety during training while preserving task performance: by filtering or replacing actions that are predicted to be unsafe in latent space, ADVICE should reduce the frequency of safety violations relative to unshielded RL, and should do so with only a minimal degradation in reward. This improvement should remain competitive even against baselines that benefit from stronger assumptions, such as access to explicit constraint information or prior knowledge of the safety specification, since ADVICE directly intervenes at the point of control rather than relying solely on learned cost modelling.

The second hypothesis is that ADVICE learns an effective safety-discriminative representation. If this contrastive latent space is learned successfully, then simple local neighbourhood tests, such as KNN voting with a safety threshold, should classify the safety of previously unseen features reliably. Empirically, this should manifest as a clear separation between the embedded safe and unsafe partitions and consistent agreement between predicted safety labels and observed

outcomes when the shield is queried online. In other words, the representation should not merely reconstruct state–action features, but should encode a geometry that aligns with safety-relevant similarity.

The final hypothesis is that ADVICE exhibits robustness under transfer and distributional shift. Since the shield is learned from experience rather than from explicit domain modelling, an important practical question is whether a shield learned in one setting retains utility when the environment changes (e.g., via different goal/obstacle configurations, or density/sparsity of hazards). ADVICE is hypothesised to maintain improved safety relative to unshielded training under such shifts, while remaining competitive in task performance. This hypothesis directly tests whether the latent safety representation captures generalisable structure rather than memorising idiosyncratic trajectories from the construction environment.

The subsequent sections evaluate these hypotheses systematically. Section 5.3.1 presents the core quantitative results for ADVICE, comparing safety and performance against unshielded RL and relevant baselines. Section 5.3.2 ablates ADVICE's hyperparameters to quantify their effect on safety performance, followed by analysis of the learnt latent representations in Section 5.3.3. Section 5.3.4 analyses ADVICE's ability to achieve strong safety performance under distributional shift, before finally, the cautiousness of ADVICE is analysed in Section 5.3.5.

### 5.3.1 Core Results

This section presents the core experimental results used for validating the proposed ADVICE shielding method. We begin by evaluating the safety and task performance of ADVICE, followed by qualitative results of its shielding behaviour.

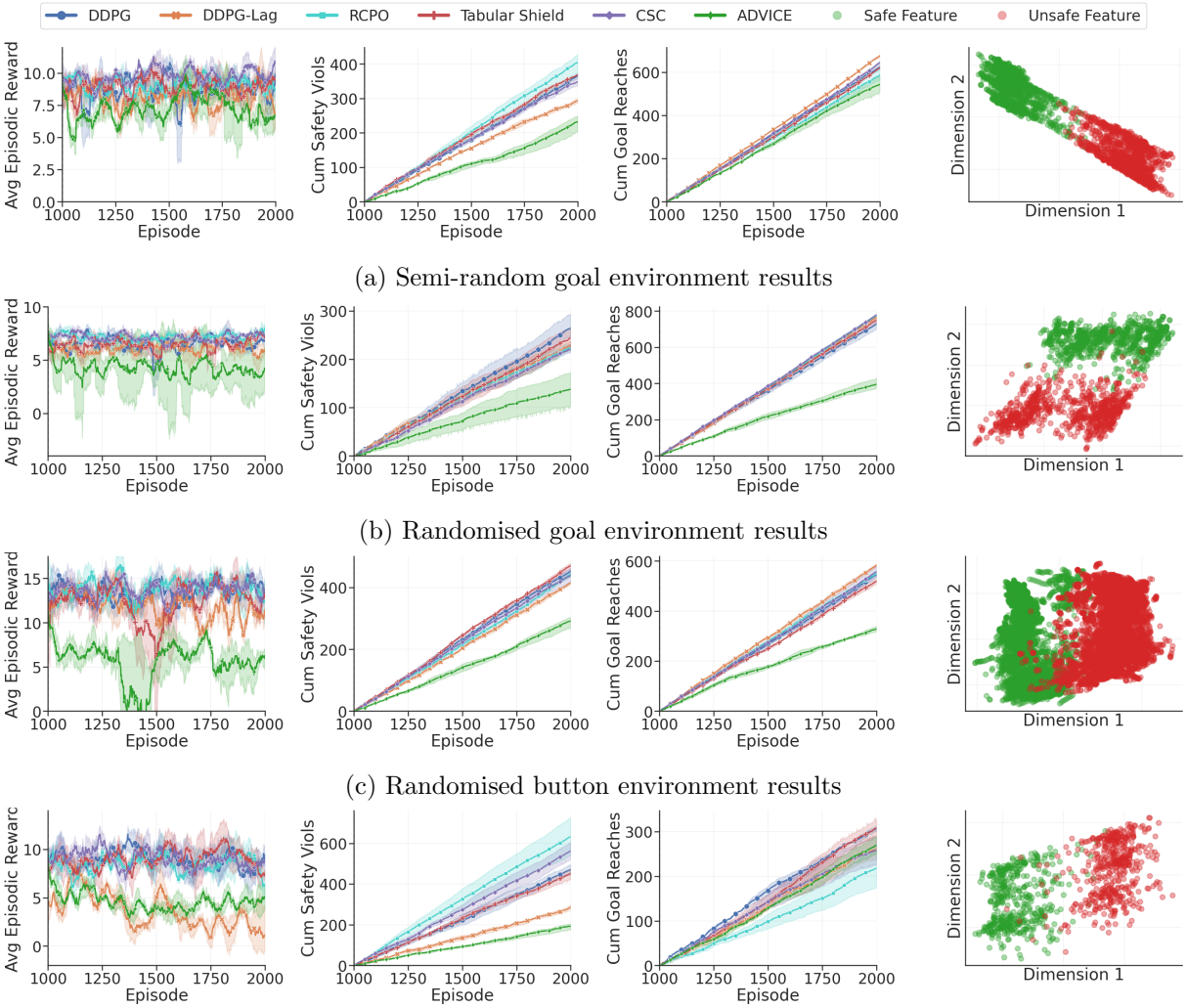
We first evaluate the base performance of ADVICE to assess how effective it is at learning the safety constraints during training, shielding the RL agent from safety violations, and keeping task performance to a competitive level. To assess this, we run experiments on a handful of environments from the Safety Gymnasium [257] test suite. We compare against several state-of-the-art safe RL methods: a baseline DDPG agent [41], DDPG-Lag [168], RCPO [170], Tabular Shield [205], and CSC [213]. The baseline selection reflects ADVICE's black-box setting, which does not assume a known dynamics model or an explicit safety specification. These assumptions limit the range of directly comparable shielding methods, motivating a diverse comparison: DDPG provides an unshielded reference, DDPG-Lag and RCPO represent cost-based policy optimisation, and CSC represents safety-critic learning. Tabular Shield provides a shielding comparator that similarly learns from observed unsafe outcomes, allowing comparison between storing previously unsafe state-action pairs and generalising through a learned representation. However, its dependence on revisiting the same discretised pairs limits its suitability for continuous spaces. Its results therefore characterise this tabular adaptation in the evaluated setting, rather than its effectiveness in its intended discrete setting. By contrast, Chapter 4 evaluates USC's modifications to cost-based policy optimisation and therefore emphasises safety critics and constrained optimisation methods. DDPG, RCPO, and CSC provide overlap between the evaluations, although differences in tasks and protocols preclude a direct ranking across chap-

ters. Safety and task performance are assessed across a range of relevant metrics, including average reward, cumulative safety violations, and the total number of times the agent reached the goal. Full details of the experimental setup and environment information can be found in Appendix C.3. All methods employ identical network architectures as well as training procedures for a fair comparison. Core training results for the comparative methods can be visualised in Figure 5.2. Full training plots for all methods can be found in Appendix C.2.

Across the environments shown, ADVICE consistently achieves the strongest reduction in safety violations during training, often outperforming baselines that either rely on explicit constraint structure through CMDP modelling.

In the easiest semi-random goal environment (the simplest of all environments), the unshielded and cost-based methods achieve high task performance but incur a large number of safety violations. For example, the CSC and RCPO agents achieve high average episodic rewards across training of $\approx 10.0$, but both accumulate large amounts of safety violations at around $\approx 350-400$ violations. The DDPG-Lag agent manages to reduce violations by $\approx 50$ compared to the base DDPG agent, but this trade-off results in a small reduction of reward; as a result, it averages $\approx 7.5$ reward across training. By contrast, ADVICE substantially reduces violations to approximately $\approx 200-225$, a reduction of nearly a third relative to the unshielded DDPG agent. This safety improvement is achieved through aggressive action filtering, which reduces the average episodic reward to $\approx 7.0$ and accumulates $\approx 50$ fewer goal reaches across training. While this illustrates a conservative bias towards safety, it shows how ADVICE can substantially reduce safety violations compared to other methods, even more so than methods that model safety via a priori cost stream. These results demonstrate a reduction in safety violations at the expense of reward and task completion. This compromise may be preferable where violations carry substantial consequences, although its suitability depends on the application's safety requirements.

As task difficulty increases and safety-focused features semantically overlap with task-relevant features (e.g., pressing the wrong button in the Randomised button environment results in a safety violation), the limitations of existing approaches become more apparent. In the Randomised button environment, all comparative methods fail to reduce violations past a negligible amount. The Tabular Shield and DDPG-Lag agents do not reduce the violations, bar some deviation, but do reduce the average task reward from $\approx 14$ across training to $\approx 11$, showing a failure at the safety/reward trade-off. This shows that current cost-based methods struggle to disentangle what it means to be safe when there is strong overlap between safety- and reward-relevant features. By contrast, ADVICE's contrastive objective learns a better representation of safety and thus accumulates $\approx 125$ fewer safety violations compared to all other methods. While it averages the lowest reward of $\approx 6.5$ across training, it maintains a very competitive amount of goal reaches at $\approx 325$. This is a repeating trend across all environments, while ADVICE's reward is much lower, the task completion remains competitive, showing that the reward design/structure itself penalises safe task completion (e.g., Safety Gymnasium heavily rewards Euclidean movement towards the goal, thus, circling the environment towards the goal only achieves positive reward in the last part of the trajectory, rather than across the whole trajectory if the path was a straight line).




(a) Semi-random goal environment results

(b) Randomised goal environment results

(c) Randomised button environment results

(d) Randomised circle environment results

Figure 5.2: Average episodic reward, cumulative safety violations, cumulative goal reaches of examined approaches and example latent visualisation for the semi-random goal (a), randomised goal (b), randomised button (c), and randomised circle (d) environments. Results are shown for episodes 1000–2000 to highlight ADVICE's execution and adaptation.

The Randomised circle environment is the most challenging because goal-relevant features may be present throughout most of an episode trajectory, yet only contribute to task success if they are realised at the final timestep. As a result, many intermediate states are ambiguous with respect to eventual success or failure. This difficulty is further compounded by the fact that obstacles are spatially co-located with the goal region, tightly coupling goal achievement with safety violations. Across comparative methods, the DDPG-Lag agent is the most competitive, accumulating $\approx 275$ safety violations with an average reward of $\approx 4.0$, but as many goal reaches as the base DDPG agent (again showing reward does not always correlate with task completion). Interestingly, the RCPO agent achieves the most safety violations, $\approx 150$ more than the base DDPG agent, with $\approx 50$ fewer task completions, highlighting how difficult this environment can be for some cost-based methods. ADVICE, again, achieves the fewest safety violations, at $\approx 200$,

compared to $\approx 450$ with the base DDPG agent. It does so with just as many task completions and an average reward of $\approx 5.5$.

Across these environments, ADVICE consistently reduces cumulative safety violations, while the associated loss of task performance varies in magnitude. In the randomised goal and button environments, fewer violations are accompanied by substantially fewer goal reaches and lower reward, indicating a conservative trade-off rather than an improvement across all metrics, which is beneficial in environments with stricter safety-performance weighting. The reduction in goal completion is smaller in the semi-random goal environment. In the randomised circle environment, ADVICE achieves fewer violations while maintaining goal completion comparable to DDPG-Lag and CSC, although its reward remains below several baselines. These results suggest that ADVICE is most attractive when avoiding violations takes priority over task efficiency. Reward shaping may contribute to its lower return, but cannot alone explain the reduced goal completion observed in some tasks. The desirability of this trade-off therefore depends on the application's tolerance for violations and reduced task progress.

Finally, the latent-space visualisations (final column) support these quantitative trends. Across tasks, ADVICE consistently learns embeddings in which safe and unsafe features form separable clusters, even in the more challenging environments. This separation underpins the effectiveness of the KNN-based safety inference and provides qualitative evidence that the shield's decisions are grounded in a meaningful safety representation rather than brittle heuristics.

Together, these results allow us to empirically validate our first hypothesis: that ADVICE improves safety during training while preserving task performance. Across all four environments, ADVICE achieves the lowest safety violations by a significant margin, and it does so with competitive task completion against methods that have cost functions that are designed around known safety constraints.

Figure 5.3 provides qualitative evidence that is consistent with the quantitative trends in Figure 5.2. In the semi-random goal environment, the base DDPG agent exhibits the most direct, return-seeking behaviour, typically cutting through the centre of the obstacle wall to minimise path length. While this is efficient in expectation, it is brittle under action noise: small deviations are more likely to produce contacts when the trajectory threads through narrow gaps, which aligns with the higher cumulative violation counts observed for unshielded learning. Cost-based baselines, such as RCPO, display a more moderated version of this behaviour, generally maintaining a wider berth around obstacles than pure DDPG. This reflects their tendency to internalise safety through reward–cost trade-offs: rather than explicitly preventing risky actions, they learn policies that reduce expected violations by increasing clearance. However, these trajectories still frequently pass close to obstacle boundaries, suggesting that balancing cost via the policy alone does not reliably enforce conservative execution when the shortest paths remain only marginally risky. In contrast, ADVICE adopts the most cautious navigation strategy, favouring routes that move around the obstacle wall with noticeably larger clearance. This behaviour matches the role of the post-shield: action proposals that place the agent in locally risk-elevated regions are more likely to be filtered and replaced, biasing execution toward trajectories sup-



DDPG DDPG-Lag RCPO Tabular Shield CSC ADVICE

Figure 5.3: Example trajectories of the assessed approaches on the semi-random goal environment; obstacles are purple, goals are green circles, and the RL agent is the red vehicle.

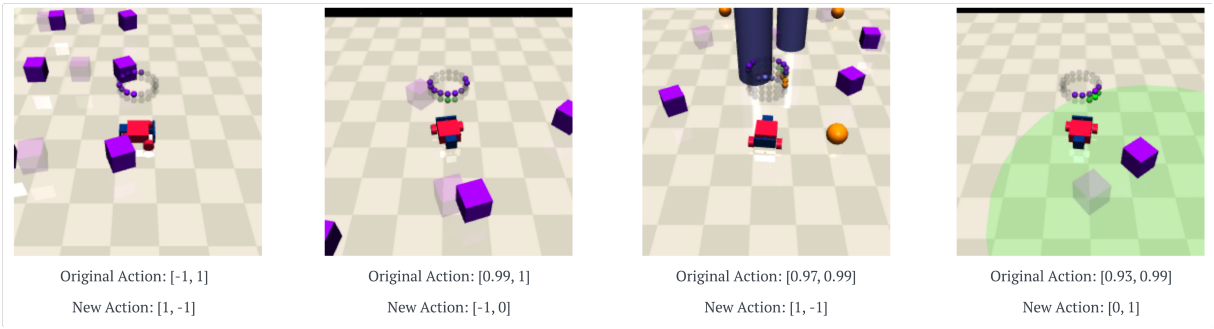



Figure 5.4: Example ADVICE shield activations during training showing the environment around the agent when the shield declared the proposed feature unsafe, the policy's proposed action, and the safer alternative selected from the candidate action set after safety filtering and value-based ranking.

ported by dense safe neighbourhood evidence in latent space. The resulting paths are longer and sometimes initially move away from the goal, which helps explain why ADVICE can exhibit a lower average episodic reward despite achieving substantially fewer safety violations: the reward function penalises detours and increases in goal distance, whereas the shield explicitly optimises for risk reduction rather than shortest-path efficiency.

Figure 5.4 illustrates moments where ADVICE intervenes in the control loop by overriding the policy's proposed action. In each example, the original action points toward a locally risky region, typically steering the vehicle into close proximity with obstacles, clipping the edge of an obstacle cluster, or moving into configurations where small action noise could plausibly produce a collision. For example, in the first example, the policy's action would turn the agent to the left, into the obstacle, the replaced action turns the agent to the right, to move out of the way of the hazard.

A consistent qualitative pattern is that the replacement actions preserve the high-level intent (continuing forward motion or goal-directed steering) but reduce the likelihood of contact by adjusting the most risk-critical component of the control signal. In practice, this often appears as dampening an aggressive turn or straightening the trajectory when the policy would cut too close to an obstacle. These interventions provide an intuitive explanation for the core quantitative results: ADVICE reduces cumulative safety violations by preventing the specific local control decisions that most frequently lead to terminal failures, while still enabling the agent to make steady task progress. At the same time, because the shield sometimes replaces a shortest-path action with a safer detour action, these activations also help explain why ADVICE can incur a

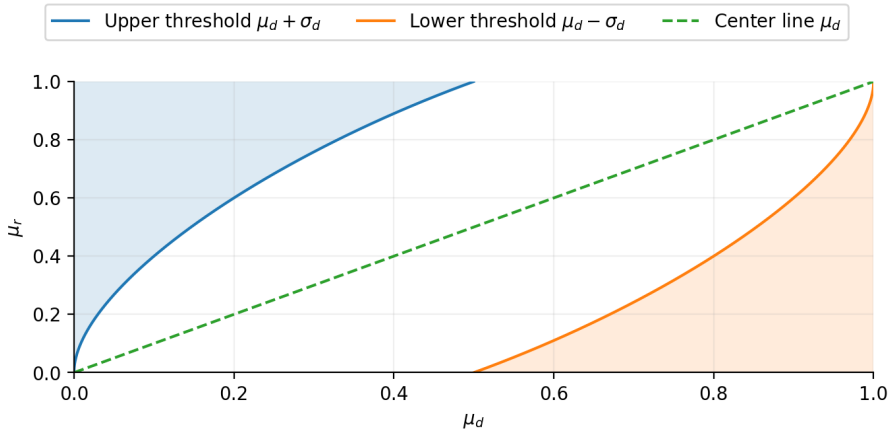



Figure 5.5: Decision regions for ADVICE's adaptive update of $K$ in the $(\mu_d, \mu_r)$ plane. The shaded blue region corresponds to $\mu_r > \mu_d + \sigma_d$ (triggering an increase in $K$), and the shaded orange region corresponds to $\mu_r < \mu_d - \sigma_d$ (triggering a decrease in $K$); otherwise $K$ is unchanged. The reference line $\mu_r = \mu_d$ is shown dashed.

modest reward penalty in environments where reward primarily reflects goal-directed efficiency rather than risk-aware behaviour.

While the preceding results demonstrate that ADVICE can substantially reduce safety violations in practice, it is also important to clarify how the adaptive cautiousness mechanism that drives this behaviour is formally bounded. Figure 5.5 illustrates the decision regions governing ADVICE's adaptive update of the safety threshold $K$ as a function of recent and historical safety violation rates. The horizontal axis represents the long-term (distant) violation probability $\mu_d$, while the vertical axis captures the short-term (recent) violation probability $\mu_r$. The dashed diagonal corresponds to the reference line $\mu_r = \mu_d$, indicating stationary safety behaviour where recent outcomes are consistent with historical performance. When the recent violation rate exceeds the upper threshold $\mu_d + \sigma_d$, the system enters the shaded blue region, triggering an increase in $K$ and thereby making the shield more conservative. Conversely, when $\mu_r$ falls below the lower threshold $\mu_d - \sigma_d$, ADVICE enters the shaded orange region and decreases $K$, allowing for more exploratory behaviour. Importantly, the gap between these thresholds defines a stability band in which no update is performed, preventing oscillatory behaviour in response to noise or isolated safety events. This bounded update rule ensures that ADVICE responds only to statistically meaningful deviations in safety performance: sustained increases in violations lead to tighter control, while consistent safe behaviour gradually relaxes conservatism. As a result, the adaptive mechanism provides a principled explanation for the empirical trends observed earlier, namely, ADVICE's ability to prevent unsafe behaviour without permanently locking the agent into an overly conservative/restrictive behaviour.

In the core results, safety violations induce automatic episodic termination, which makes costs intrinsically sparse: once a violation occurs, no further cost can accrue within that episode. While appropriate for evaluating shielding behaviour, this differs from the operating regime assumed by CMDP and Lagrangian methods such as DDPG-Lag and RCPO, where costs accu-

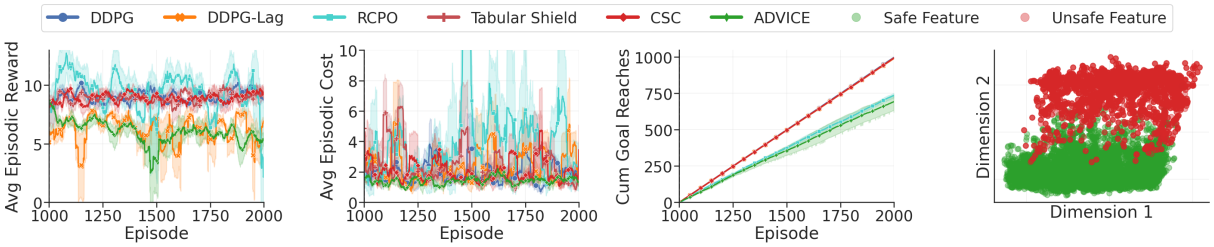



Figure 5.6: Average episodic reward, average episodic cost, cumulative goal reaches, and example latent space visualisation on the constrained randomised goal environment.

mulate over time, and the policy must manage a long-horizon reward–cost trade-off. To ensure a fair comparison with these approaches, we therefore also evaluate ADVICE in a CMDP-style setting on the constrained randomised goal environment, where safety violations contribute to an episodic cost signal without terminating the rollout. The safety budget is set to $\chi = 0$, so any incurred cost constitutes a constraint violation. In this environment, ADVICE's notion of an unsafe transition is adapted in the minimal way required: rather than using non-accepting terminal states as unsafe evidence, any transition that induces cost is treated as unsafe. This preserves ADVICE's black-box nature while aligning its supervision signal with that optimised by cost-based baselines; the remainder of the method, including latent-space safety inference and adaptive modulation of the threshold $K$, remains unchanged.

Figure 5.6 show that this CMDP-style evaluation largely preserves the results observed in Figure 5.2, while making the cost comparison more directly meaningful. The base DDPG agent averages a high reward across training of $\approx 9.0$, but sustains persistent episodic costs, averaging $\approx 2.5$ across training, reflecting the expected behaviour when reward maximisation continues to exploit risky shortcuts. Cost-based baselines reduce this cost to varying degrees, but their behaviour is often unstable: RCPO in particular exhibits high-variance cost trajectories with large spikes that reach $\approx 6.0$ cost in some periods, indicating that cost optimisation can exhibit severe oscillation when trying to dually maximise reward and minimise cost. This can be seen by its ability to reach the highest reward at points, sometimes $\approx 13.0$, however, this then leads to an increase in cost, which in turn forces the reward to come down as the agent tries to minimise risky behaviour again. The CSC agent is the most effective at this balance, as it averages the same large reward as the base DDPG agent, but averages a low cost of $\approx 2.0$. It does so whilst accumulating the task completion as the base agent. By contrast, ADVICE yields the lowest and most stable episodic cost curve in this regime at $\approx 1.75$, with comparatively little oscillation across training. The cost reduction comes with a modest reward penalty, as it averages a similar reward to DDPG-Lag at $\approx 6.5$. Nevertheless, ADVICE continues to make competitive task progress, as reflected by its approximately linear growth in cumulative goal reaches.

The constrained-environment experiment demonstrates that ADVICE does not require violations to terminate an episode: an observed safety cost can instead identify unsafe transitions. This supports its applicability to the non-terminating, cost-based setting of Safety Gymnasium. However, ADVICE uses this feedback to classify actions rather than to optimise a cumulative cost budget, and binary safety labels do not distinguish violations by severity.

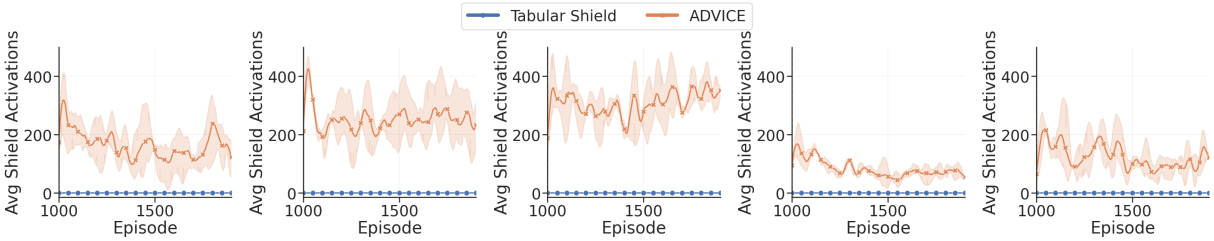



Figure 5.7: The average shield activations for ADVICE and the Tabular Shield in the semi-random goal, random goal, random button, random circle, and constrained random goal environments, respectively.

In an environment containing both known and unknown hazards, explicit safety checks could handle specified constraints while ADVICE addresses additional hazards represented in its labelled experience. The intuition is that the learned representation can generalise from observed unsafe outcomes to similar state-action pairs without requiring an explicit description of each hazard. This requires the observations to contain relevant information and the feedback to identify unsafe outcomes; absence of a known cost alone would not establish safety with respect to unknown hazards. Any replacement action would also need to satisfy the known constraints. Hazards first encountered after construction may require additional labelled data and retraining, since adapting $K$ changes cautiousness within the existing representation. This combined configuration has not been evaluated here.

Figure 5.7 compares the frequency of shield activations across environments for ADVICE and the Tabular Shield agents. A consistent pattern emerges: the Tabular Shield exhibits very few activations. This highlights its limited capacity to generalise in high-dimensional continuous state–action spaces, rather than superior safety behaviour. Because the Tabular Shield relies on recording previously unsafe transitions at a fixed discretisation, it can only intervene on state–action tuples that have been observed exactly (up to rounding) [205]. In continuous domains, this causes most locally risky actions to appear novel and therefore admissible, leading to minimal intervention even in environments with dense hazards.

By contrast, ADVICE activates its shield more frequently and more consistently across environments, particularly in the semi-random ($\approx 200$ activations) and constrained settings ($\approx 175$ activations). This reflects the fact that ADVICE does not memorise individual unsafe transitions, but instead reasons over neighbourhood structure in a learned latent space. As a result, once a region of the state–action space is identified as unsafe, ADVICE can generalise this information to nearby configurations and proactively intervene before repeated violations occur. The higher activation rate should therefore be interpreted as a sign of effective generalisation rather than excessive conservatism, and it helps explain why ADVICE achieves substantially lower cumulative safety violations than tabular shielding in continuous control tasks.

Finally, Figure 5.8 plots the rate of safety violations over training for all approaches across the core environments. For ADVICE, a clear and consistent pattern emerges once the shield is activated. The violation rate drops sharply at the point of shield deployment, falling well below

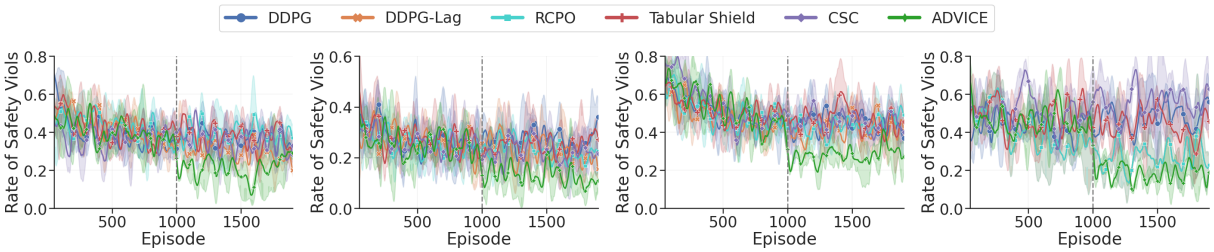



Figure 5.8: The rate of safety violations for all approaches in the semi-random goal, random goal, random button, and circle environments, respectively.

that of all comparative methods and remaining suppressed thereafter. This abrupt reduction contrasts with the behaviour of unshielded and cost-based approaches, whose violation rates fluctuate over time but do not exhibit a comparable discontinuity. For example, in the circle environment, ADVICE averages $\approx 0.5$ safety violations an episode (roughly a violation every other episode), but once the shield is switched on, this reduces to $\approx 0.2$ violations per episode (roughly a violation every 5 episodes). Importantly, the form of this reduction provides insight into its cause. The sudden decrease in violations is not indicative of the policy gradually learning safer behaviour; such learning would be expected to manifest as a smoother, progressive decay in violation rates. Instead, the sharp transition aligns with the activation of the post-shield and reflects direct intervention at execution time. This supports the interpretation that ADVICE's safety improvements arise primarily from its shielding mechanism rather than from changes in the underlying policy.

### 5.3.2 Parameter Analysis

We next perform a detailed parameter analysis to examine the sensitivity and robustness of ADVICE with respect to its key hyperparameters and quantify their influence on reward maximisation and safety performance.

We share the same experimental setup and evaluation protocol described in Section 5.3.1, ensuring direct compatibility with core results. All models are trained on the randomised goal environment, using identical architecture and parameters for a fair comparison.

Firstly, we analyse the sensitivity of ADVICE to the choice of $E_{max}$, which determines the duration of the initial unshielded data-collection phase before the shield is activated. This parameter directly controls the amount and diversity of experience available to the contrastive autoencoder during construction, and therefore induces a clear trade-off between short-term safety and long-term shielding effectiveness. Results are shown in Figure 5.9.

When the shield is activated early (e.g. $E_{max} = 500$), ADVICE rapidly suppresses safety violations to $\approx 0.2$ per episode. However, this conservative behaviour comes at the cost of a less informative training set for the latent model. With fewer unsafe and safe examples observed before shielding, the learned representation is necessarily coarser, which limits the shield's ability to discriminate nuanced risk patterns. This results in a slightly reduced rate of goal reaches, indicating that the shield intervenes more aggressively.

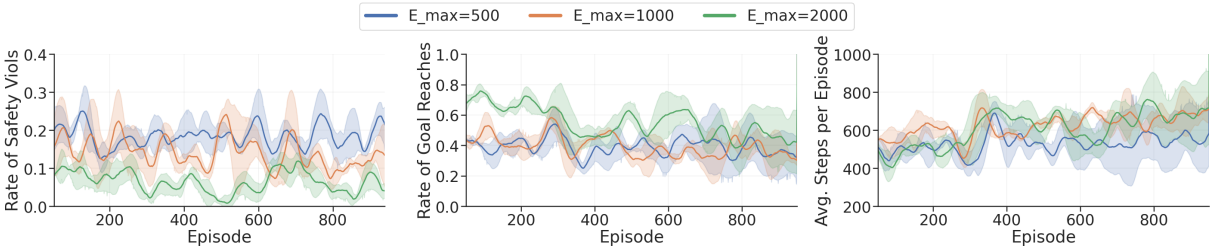



Figure 5.9: The rate of safety violations, and the rate of goal reaches when ADVICE is activated at different intervals $E_{max}$.

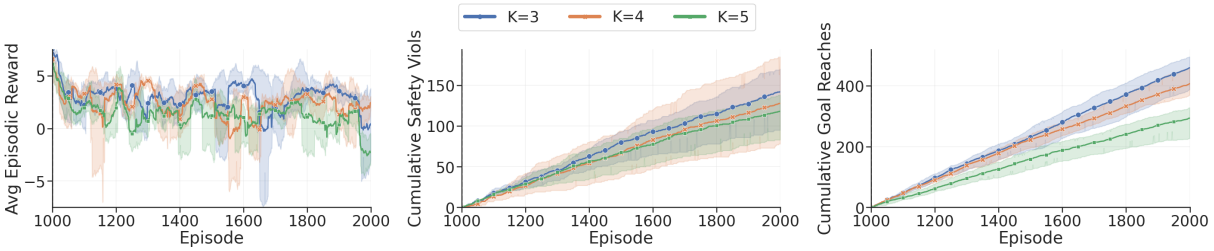



Figure 5.10: Average episodic reward, cumulative safety violations, and cumulative goal reaches of various values of $K$ under fixed ADVICE on the randomised goal environment.

Delaying shield activation (e.g. $E_{max} = 2000$) allows ADVICE to collect a much richer set of state–action features before any intervention occurs. This improves the quality of the learned latent space, resulting in higher goal-reaching rates and much lower safety violation rate, from $\approx 0.2$ per episode at $E_{max} = 500$ to $\approx 0.075$ per episode at $E_{max} = 2000$. However, safety violations accumulate at a substantially higher rate while the agent explores without protection. In safety-critical settings, this extended exposure to unsafe behaviour may be unacceptable, even if it yields a stronger shield later on.

Next, we analyse the effect of the fixed safety threshold $K$ when ADVICE is deployed without adaptive cautiousness. The safety threshold $K$ controls how much local evidence is required for a proposed state–action feature to be treated as safe in latent space: larger values require a stronger consensus among the nearest neighbours and therefore make the shield more risk-averse, whereas smaller values relax the decision rule and permit more exploratory (but potentially unsafe) actions. The results in Figure 5.10 make this trade-off explicit and show that, under a fixed-threshold shield, safety and task performance can be tuned for preference.

The most relaxed setting, $K = 3$ (where $K_{max} = 5$), yields an agent with the strongest task progress but the weakest safety performance. This agent incurs the most safety violations, indicating that allowing actions with only a minimal safe-majority in the neighbourhood leads to more frequent violations in unsafe regions. However, as it is the most relaxed, it also achieves the highest reward, an average of $\approx 4.5$ and the most task completions.

At the other extreme, $K = 5$ (where $K_{max} = 5$), produces the safest but most conservative agent. This agent produces the fewest safety violations during training, averaging $\approx 40$ fewer violations than the agent with $K = 3$, indicating that requiring near-unanimous safe neighbour

Table 5.1: Mean sensitivity analysis of $h_d$ and $h_r$ on the randomised goal environment for $E_{max} = 1000$.

| | | $h_d$ | |
|---|---|---|---|
| $h_r$ | Metrics | 10 | 25 |
| 2 | Consecutive Episodes ($K = 3$) | $24.49 \pm 9.59$ | $31.09 \pm 4.53$ |
| | Consecutive Episodes ($K = 4$) | $1.00 \pm 0.00$ | $7.89 \pm 0.77$ |
| | Consecutive Episodes ($K = 5$) | $32.97 \pm 7.95$ | $151.60 \pm 3.09$ |
| | Changes of $K$ | $116.33 \pm 9.87$ | $27.67 \pm 4.62$ |
| 3 | Consecutive Episodes ($K = 3$) | $166.81 \pm 5.71$ | $117.35 \pm 8.07$ |
| | Consecutive Episodes ($K = 4$) | $21.10 \pm 6.86$ | $28.76 \pm 9.26$ |
| | Consecutive Episodes ($K = 5$) | $13.49 \pm 5.40$ | $149.68 \pm 8.71$ |
| | Changes of $K$ | $28.67 \pm 1.41$ | $19.33 \pm 2.83$ |
| 4 | Consecutive Episodes ($K = 3$) | $954.49 \pm 9.59$ | $169.07 \pm 9.02$ |
| | Consecutive Episodes ($K = 4$) | $22.60 \pm 6.71$ | $22.10 \pm 7.26$ |
| | Consecutive Episodes ($K = 5$) | $0.00 \pm 0.00$ | $181.35 \pm 4.92$ |
| | Changes of $K$ | $2.33 \pm 0.47$ | $17.77 \pm 2.72$ |

evidence effectively suppresses risky actions. However, this conservatism comes with a clear performance cost. This agent accumulates task completions more slowly than the other values, suggesting that the shield intervenes frequently enough to prevent the policy from executing many goal-efficient actions, instead forcing safer but longer trajectories.

Finally, we analyse the effect of the distant and recent history windows $h_d$ and $h_r$ when ADVICE is deployed with adaptive cautiousness. Table 5.1 reports two complementary indicators of adaptation behaviour: (i) how long ADVICE tends to remain at each cautiousness level (mean consecutive episodes at $K \in \{3, 4, 5\}$), and (ii) how frequently the mechanism changes its cautiousness (the number of changes of $K$). Together, these metrics make it possible to distinguish three qualitatively different regimes: overly reactive switching of $K$, stable but conservative *locking* of the value of $K$, and balanced adaptation.

When the recent window is very short ($h_r = 2$), the update is highly sensitive to short-term fluctuations in violation outcomes. This is most apparent for $h_d = 10$, where $K$ changes extremely frequently ($\approx 116.33$ changes on average across training) and ADVICE spends only negligible time at the intermediate threshold $K = 4$ (essentially one episode on average) while oscillating between relaxed ($K = 3$) and conservative ($K = 5$) values. This pattern is characteristic of an adaptation rule that is responding to noise rather than persistent trends: minor increases in the recent violation rate are sufficient to push the mechanism upward, and similarly tiny decreases pull it back down. Increasing the distant window to $h_d = 25$ dampens this behaviour substantially (changes drop to around $\approx 27.67$), but it also shifts the system toward long conservative runs, with $K = 5$ persisting for roughly $\approx 151.60$ consecutive episodes on average. Intuitively, it can be observed that a larger $h_d$ makes the baseline safety level slower to move, so once the

recent rate deviates enough to trigger an increase in cautiousness, it can take longer for the historical estimate to catch up and allow $K$ to relax again.

When the recent window becomes longer ($h_r = 4$), the update can become conservative and, in some cases, effectively inactive. The clearest example is the combination of $h_r = 4, h_d = 10$, where ADVICE remains at $K = 3$ for the vast majority of training ($\approx 954.49$ consecutive episodes on average) and only changes $K$ about 2–3 times. This indicates that the recent estimate is so heavily smoothed that deviations rarely exceed the decision thresholds, so the mechanism largely fails to respond even when the agent's behaviour changes. With $h_d = 25$, the system does adapt more often ($\approx 17.77$ changes of $K$), but it also spends extended periods at the conservative value $K = 5$ ($\approx 181.35$ consecutive episodes), again reflecting the tendency of a long historical window to slow the return from cautious mode once it has been entered.

At intermediate values ($h_r = 3$), the mechanism exhibits a more balanced and predictable behaviour. The number of $K$ updates is an order of magnitude lower than the most reactive setting ($\approx 28.67$ changes for $h_d = 10$, and $\approx 19.33$ for $h_d = 25$), indicating that the recent window is long enough to suppress spurious triggers while still being responsive to meaningful shifts in safety performance. Inversely, it also has many more updates of $K$ than the most conservative setting. The time spent at $K = 4$ is non-trivial, and ADVICE does not collapse into a constant oscillation between the most relaxed and the most conservative values, showing the adaptation is more focused and less noisy.

### 5.3.3 Representation Analysis

We next investigate the quality of the learned latent space that ADVICE uses to discriminate safe and unsafe features, and assess whether the contrastive embedding provides a reliable basis for neighbourhood-based safety decisions beyond the reported violation and reward metrics. We utilise the same experimental setup and evaluation protocol described in Section 5.3.1 to ensure direct comparability with the core results.

In particular, we analyse the latent space constructed by the contrastive objective by measuring how effective the separation between safe and unsafe features is, and we estimate ADVICE's feature misclassification rate to empirically assess the bound in Theorem 5.1. Specifically, $\mathbb{P}(\text{misclass})$ is estimated as the empirical 1-nearest-neighbour error in the latent space: for each embedding $\mathfrak{v}_i$ with label $y_i$, we compute the nearest within-class distance $\zeta_{\text{same}}(\mathfrak{v}_i) = \min_{j \neq i : y_j = y_i} |\mathfrak{v}_i - \mathfrak{v}_j|_2$ and the nearest cross-class distance $\zeta_{\text{diff}}(\mathfrak{v}_i) = \min_{j : y_j \neq y_i} |\mathfrak{v}_i - \mathfrak{v}_j|_2$, and set $\mathbb{P}(\text{misclass}) = \frac{1}{N} \sum i = 1^N \mathbf{1}! [\zeta_{\text{diff}}(\mathfrak{v}_i) < \zeta_{\text{same}}(\mathfrak{v}_i)]$. The effective margin for a sample is defined as $m_{\text{achieved}} = \zeta_{\text{diff}}(\mathfrak{v}_i) - \zeta_{\text{same}}(\mathfrak{v}_i)$ (positive when $\mathfrak{v}_i$ is closer to its own class than the opposite class), and we summarise separation by the 5% quantile margin $m_{\text{achieved}} = Q_{0.05}(m_i i = 1^N)$.

Table 5.2 provides a direct, embedding-level explanation for why ADVICE's safety test remains reliable across environments. Across all tasks, the estimated 1-nearest-neighbour misclassification probability is low, remaining below 3% in four of the five environments and peaking in the randomised circle environment at $0.059 \pm 0.015$. The three goal environments (semi-random, randomised, and constrained goal) yield the lowest misclassification rates ($0.013 \pm 0.005$,

Table 5.2: The empirical 1-nearest-neighbour misclassification probability $\mathbb{P}$(misclass) and achieved class-separation margins in the learned embedding space for ADVICE across evaluated environments.

| | Semi-random Goal | Randomised Goal | Randomised Button | Randomised Circle | Constrained Randomised Goal |
|---|---|---|---|---|---|
| $\mathbb{P}$(misclass) | $0.013 \pm 0.005$ | $0.022 \pm 0.007$ | $0.026 \pm 0.003$ | $0.059 \pm 0.015$ | $0.009 \pm 0.002$ |
| Effective Margin $m_{\text{achieved}}$ | $0.084 \pm 0.0021$ | $-0.047 \pm 0.029$ | $-0.023 \pm 0.018$ | $-0.091 \pm 0.046$ | $-0.130 \pm 0.076$ |
| Effective Margin $m_{\text{achieved}}$ (5%) | $0.168 \pm 0.097$ | $0.122 \pm 0.067$ | $0.018 \pm 0.076$ | $0.131 \pm 0.014$ | $0.070 \pm 0.075$ |

$0.022 \pm 0.007$, and $0.009 \pm 0.002$, respectively). This is most likely because the safe-relevant and task-relevant features do not exhibit heavy overlap, which is observable in both the randomised button and circle tasks.

The margin statistics then clarify how misclassification arises and why it remains rare despite difficult regimes. In the semi-random goal environment, the mean achieved margin is positive ($0.084 \pm 0.0021$), indicating that embeddings are, on average, closer to their own class than to the opposite class, which is the expected geometry for reliable nearest-neighbour inference. However, in the randomised goal, randomised button, randomised circle, and constrained randomised goal environments, the mean achieved margin becomes negative ($-0.047 \pm 0.029$, $-0.023 \pm 0.018$, $-0.091 \pm 0.046$, and $-0.130 \pm 0.076$, respectively). At face value, a negative mean margin might suggest a systematically entangled latent space, but this would contradict the consistently low misclassification probabilities. The most informative way to reconcile these observations is to treat the margin distribution as heavy-tailed: the majority of samples are correctly separated with positive margin, while a small number of outliers exhibit strongly negative margins, pulling the mean below zero without substantially increasing the fraction of actual nearest-neighbour errors. In other words, the latent space is broadly separable, but contains a minority of highly ambiguous points that are placed deep inside the opposite class neighbourhood. This interpretation is supported directly by the lower-tail margin summary. When separation is summarised by the 5% quantile margin, the achieved margins are largely positive in every environment ($0.168 \pm 0.097$, $0.122 \pm 0.067$, $0.018 \pm 0.076$, $0.131 \pm 0.014$, and $0.070 \pm 0.075$). Even in environments where the mean margin is negative, the embedding still exhibits positive separation for the overwhelming majority of points, and the poor mean is driven by a relatively small set of extreme, ambiguous points.

These results allow us to empirically validate the second hypothesis: that ADVICE learns an effective safety-discriminative latent representation thanks to the safety-driven data collection and contrastive objective.

### 5.3.4 Distribution Shift Results

We next evaluate ADVICE under transfer and distribution shift, testing whether a shield learned in one training regime can remain effective when the environment dynamics and hazard–goal configurations change. We utilise the same experimental setup and evaluation protocol described in Section 5.3.1 to ensure direct comparability with the core results.

Figure 5.11 created a distributional shift by training the ADVICE, RCPO, Tabular Shield,

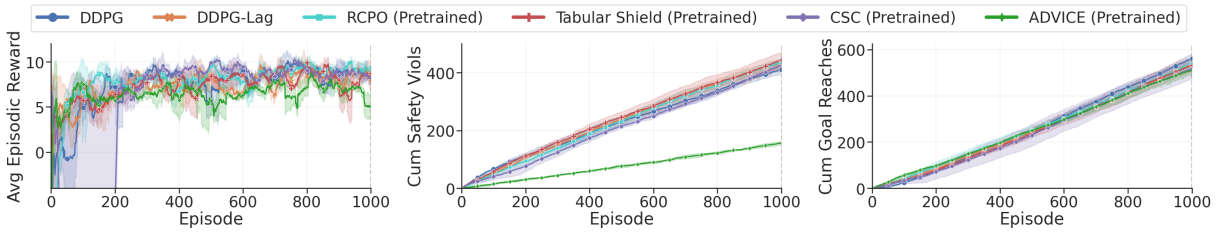



Figure 5.11: Average episodic reward, cumulative safety violations, and cumulative goal reaches on the semi-randomised goal environment where ADVICE, RCPO, Tabular Shield, and CSC have been pre-trained on the randomised goal environment to show transfer learning and distributional shift robustness capabilities.

and CSC agents on the randomised goal environment, where the hazards are sparse and then training is continued on the semi-random environment where hazards are more densely packed. This experiment ablates whether the comparative methods are able to learn robust, effective meanings of safety.

Results show that pretraining yields broadly comparable reward learning trends across methods, but only ADVICE translates this warm start into a meaningful reduction in unsafe behaviour under shift. After a short *catch-up* period of a dozen episodes, all methods stabilise to similar average episodic reward levels of $\approx 7-8$, indicating that the shift does not prevent the policy from recovering competent goal-directed behaviour. However, the cumulative safety violations show a much sharper separation between approaches. The RCPO (pretrained), the Tabular Shield (pretrained), and CSC (pretrained) agents remain tightly clustered, accumulating $\approx 400$ violations by the end of training. This indicates that, for these baselines, pretraining on the randomised goal task does not produce a safety mechanism that generalises reliably to the semi-random goal task. In contrast, the pretrained ADVICE agent exhibits significantly lower violations, $\approx 150$ at the end of training, around a $2-3\times$ reduction relative to the other pretrained methods.

Notably, this reduction in violations does not compromise task progress, as in the core results (seen in Figure 5.2), ADVICE accumulates slightly fewer task completions, but in this case it does not. The cumulative goal reaches for all methods grows linearly and remains close in magnitude, each method achieving $\approx 500-550$ goal reaches.

Overall, these results support the interpretation that ADVICE transfers more robustly because it stores and queries safety knowledge in the geometry of a learned representation, rather than depending on a finely tuned cost–reward balance or a brittle discrete memory of previously observed unsafe tuples. Under distribution shift, the baselines largely behave as though safety must be relearned from scratch, whereas ADVICE retains a usable notion of risk that continues to generalise in the target environment, producing an immediate and sustained reduction in violations. This also allows us to empirically validate the third hypothesis: that ADVICE exhibits robustness under transfer and distributional shift thanks to learning the *idea* of safety in the latent space.

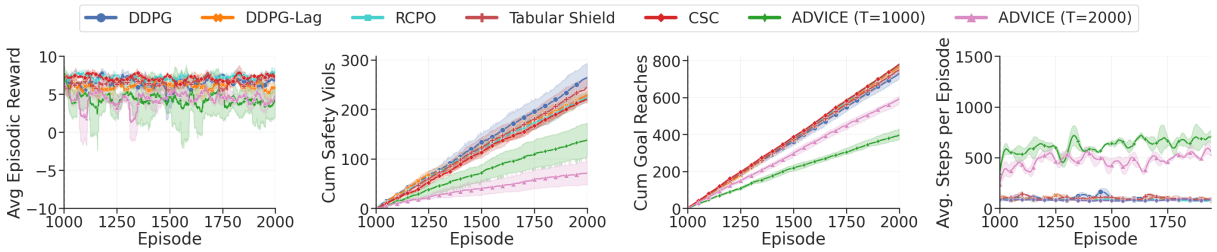



Figure 5.12: Average episodic reward, cumulative safety violations, and cumulative goal reaches of various approaches on the randomised goal environment where the maximum episodic steps are doubled for ADVICE.

### 5.3.5 Cautiousness Analysis

We next test whether the lower reward and reduced goal-reaching frequency observed for ADVICE in the core results are primarily a horizon artefact, rather than a failure to learn competent goal-directed behaviour or harm goal-directed behaviour via action filtering. In Section 5.3.1, we hypothesised that ADVICE can appear to underperform on reward because (i) the reward function penalises detours and temporary increases in goal distance, and (ii) the shield biases the agent toward longer, higher-clearance trajectories around obstacle fields, which may simply require more time steps to reach the goal (observed qualitatively in Figure 5.3). To directly evaluate this hypothesis, we repeat the randomised goal experiment setup and protocol described in Section 5.3.1, while doubling ADVICE's maximum episode length from $T = 1000$ to $T = 2000$, keeping the remainder of the training setup unchanged to ensure direct comparability with the core results.

Figure 5.12 reports the results for this experiment. With the standard horizon ($T = 1000$), ADVICE exhibits the exact same qualitative pattern as in the core results: it accumulates significantly fewer safety violations than all baselines, but achieves a slightly lower average episodic reward and fewer cumulative goal reaches. When the horizon is doubled ($T = 2000$), ADVICE's performance shifts in exactly the way predicted by our hypothesis for this experiment. In particular, ADVICE ($T = 2000$) attains markedly higher cumulative goal reaches than ADVICE ($T = 1000$) by the end of training (increasing from $\approx 400$ to $\approx 600$), narrowing much of the gap to the comparative methods, which average $\approx 700$. The average episodic reward stays the same as the ADVICE ($T = 1000$) agent, which supports the notion that the reward function penalises longer routes to the goal, as ADVICE ($T = 2000$) achieves $\approx 200$ more task completions with no increase in average reward. Interestingly

Interestingly, the notable reduction in safety violations for the ADVICE ($T = 2000$) agent (from $\approx 125$ to $\approx 75$) can be explained purely as a replay-buffer composition effect. With the shorter horizon ($T = 1000$), many ADVICE rollouts that are behaviourally safe still fail to reach the goal within the episode because the shield steers the agent along longer, higher-clearance routes. Those episodes disproportionately contribute either timeouts or late-stage risky corrections, so the replay buffer is populated more heavily by truncated/unsuccessful trajectories and the transitions leading up to them. When the horizon is doubled, a much larger fraction

of these same conservative trajectories now terminate successfully, meaning the buffer becomes dominated by goal-reaching rollouts that (by construction) contain long stretches of safe, high-return behaviour. For an off-policy learner, this changes the training signal: the critic is trained more often on state–action pairs that lead to successful outcomes without violations, and the actor is updated toward actions that match those successful rollouts. Over-training, the policy therefore proposes fewer inherently risky actions in the first place, so the shield is triggered less often, and fewer violations occur. In this sense, the improved safety is not because *more steps make the agent safer*, but because the longer horizon shifts the dataset from being failure-/timeout-heavy to being success-heavy, which trains a safer and more effective policy through the buffer.

Overall, this experiment validates the additional proposed hypothesis: when ADVICE is given adequate time to execute its safer, longer-horizon plans, its goal-reaching rate rises substantially while its safety advantage is preserved and even improved upon.

## 5.4 Discussion

In this chapter, we introduced ADVICE (**AD**apti**V**e Sh**I**elding with a **C**ontrastive Auto**E**ncoder), a black-box post-shielding method for training RL agents under unknown or difficult-to-model safety constraints. Unlike cost-based safe RL approaches that assume an explicit constraint signal, ADVICE constructs a shield directly from experience and enforces safety at execution time via a safety-discriminative representation of state–action features, such that safe and unsafe outcomes become separable in a compact latent space. Empirically, ADVICE consistently reduced safety violations across Safety Gymnasium environments and demonstrated robustness under distribution shift, where a shield learned in one environment retained safety benefits when transferred to another, indicating that the learned latent geometry captures reusable safety structure rather than memorising exact unsafe tuples.

### 5.4.1 Applicability and Impact

ADVICE is designed for settings where domain modelling is infeasible or where safety constraints are not available as a hand-engineered cost. Its applicability is therefore complementary to CMDP-style approaches: while cost-based methods excel when a dense and meaningful cost function exists, ADVICE remains applicable when safety is only observable through outcomes (e.g., failure terminations or induced constraint events), or when integrating a correct cost signal would require significant domain-specific engineering. Because ADVICE is a post-shield, it is agnostic to the underlying RL algorithm and can be deployed with standard continuous-control learners without modifying their optimisation objectives, architectures, or update rules. This makes it a practical safety wrapper.

The main practical impact of ADVICE lies in shifting safety from a purely *learned preference* to an enforced control constraint. Instead of hoping that the policy internalises safety through reward shaping or critic learning, ADVICE prevents unsafe actions at the point of execution, which is particularly valuable during exploration when policies are most volatile. This is reflected

in the violation-rate results: safety improves abruptly when the shield is activated, indicating direct intervention. Additionally, ADVICE's adaptive cautiousness mechanism adds a useful practical property: the system is not locked to a single conservatism setting. By modulating $K$ based on recent versus historical safety trends, ADVICE can become more conservative when violations rise and relax when behaviour stabilises. This makes the shield more usable across training phases, avoiding the need to hand-tune a fixed safety threshold that would otherwise be too strict early on or too permissive later.

### 5.4.2 Limitations

Despite its broad applicability, ADVICE introduces several limitations that are important in safety-critical deployment.

First, ADVICE requires an initial unshielded data-collection period to construct its latent safety representation. This is a fundamental cost of black-box shielding: without prior knowledge of constraints, some unsafe experience may be necessary to obtain informative unsafe examples. While the results suggest that ADVICE can still perform well with moderate $E_{max}$, the existence of any unshielded phase may be unacceptable in domains where violations are catastrophic, motivating either tighter initial safeguards or alternative ways of obtaining safety labels. If some data is already collected, shown in Section 5.3.4, ADVICE can be pretrained with some initial data, even if the safety requirement is unknown within that data; potentially reducing this assumption.

Second, is that ADVICE effectively *discards* a large fraction of collected experience by treating non-terminal transitions as inconclusive and excluding them from the supervised contrastive pairing stage. While this choice is principled in a black-box setting, it creates a data-efficiency gap: much of the agent's state–action visitation distribution is never used to shape the embedding. In long-horizon tasks, this can be substantial, since only a small subset of transitions occur immediately before accepting or failing outcomes.

Finally, there is practical overhead. Contrastive training requires assembling many labelled pairs, storing feature sets, and running nearest-neighbour queries during execution. While these costs are manageable in the evaluated benchmarks, they may become non-trivial in very high-dimensional observation spaces, long-horizon tasks with large feature buffers, or time-constrained control settings.

# Chapter 6

# Conclusion

This chapter concludes the thesis by summarising the findings from the technical chapters and reflecting on their implications for the design of safe RL agents operating under uncertainty. In Section 6.1, we revisit the research questions set out in Chapter 1 and examine how the proposed methods collectively address them. In Section 6.2, we position our contributions within the research landscape detailed in Chapter 2, before outlining the broader research directions that have been identified in this thesis in Section 6.3.

## 6.1 Thesis Summary

This thesis aimed to investigate how RL agents can achieve improved safety and robustness when operating in safety-critical environments characterised by uncertainty and partial knowledge. Although RL agents are often trained safely in simulation, deployment exposes them to the sim-to-real mismatch; thus, agents are sometimes trained on the physical system itself (e.g., on-robot/vehicle training) in real-world test beds rather than purely in simulation. In these settings, training cannot tolerate the trial-and-error failures that standard training regimes implicitly assume, as a single erroneous action may result in a serious violation or harm to a person or the agent. Many target applications involve embodied agents whose behaviour is shaped by a closed-loop interaction between perception, decision-making, and low-level control, each of which can introduce distinct failure modes: perceptual errors can hide hazards, predictive errors can distort risk estimates, and execution constraints may be unknown, implicit, or difficult to formalise. Accordingly, the central view of this thesis proposed that safety must be treated as a systems property of the perception–decision–control pipeline, and that achieving safe behaviour requires addressing uncertainty and risk across the full loop rather than within any single component in isolation.

Given this motivation, three research questions were proposed concerning how the safety of RL agents can be strengthened at distinct points in the pipeline. The first question asked whether incorporating distribution-free uncertainty quantification at the perception stage can mitigate imperfect sensing and improve downstream safety without degrading task performance. To address this, Chapter 3 introduced **M**onte-**C**arlo **C**onformal **P**rediction (MC-CP), combining

lightweight stochastic sampling with statistically valid conformal calibration to produce perceptual outputs with distribution-free reliability guarantees. By shifting perception from point predictions toward conformally calibrated predictions that statistically guarantee the coverage of the ground truth, the agent gains an explicit mechanism for reducing imperfect perception under ambiguous or shifted inputs. In the embodied navigation setting, these calibrated hazard representations were then shown to support safer downstream behaviour, e.g., by increasing clearance from ground-truth hazards and reducing time spent in ground-truth hazardous regions, while preserving the ability to learn effective policies.

The second question investigated whether explicitly accounting for uncertainty in future safety predictions can help an agent regulate its internal pessimism, avoiding both reckless behaviour driven by optimistic misprediction and overly cautious behaviour induced by uniform conservatism. Chapter 4 addressed this question by proposing the **U**ncertain **S**afety **C**ritic (USC), which modulates conservative safety evaluation according to principled uncertainty estimates rather than enforcing pessimism uniformly across the state–action space. The core aim was to concentrate conservatism where the safety critic is epistemically uncertain about its predictions and/or where there is high cost, while relaxing it in certain and/or low-cost regions so that the agent can exploit safe, confident regions of the environment and be cautious elsewhere. Empirically, this uncertainty-aware shaping was shown to produce strong, scalable reward-cost trade-offs and produced a safety critic that matched the ground truth hazard map more accurately.

The third question asked whether adaptive, data-driven safety mechanisms can protect the agent from executing unsafe actions when the environment behaves as a black box and explicit safety constraints are difficult or impossible to specify. This question reflects the practical reality that many safety requirements are implicit, context-dependent, or high-dimensional, and that simplified models may be insufficiently trustworthy for enforcement. Chapter 5 addressed this by proposing ADVICE (**AD**apti**V**e Sh**I**elding with a **C**ontrastive Auto**E**ncoder), a post-shielding mechanism that learns to filter, correct, or allow proposed actions based on experience, without requiring an explicit dynamics model or a hand-crafted constraint set. In this formulation, safety enforcement becomes an adaptive process: the shield refines its cautiousness as the agent interacts with the environment, aiming to reduce violations while retaining substantial task performance.

Taken together, the thesis provides a coherent answer to its overarching question: safe exploration under uncertainty or partial knowledge is achievable when uncertainty is represented explicitly and used to actively modulate perception, decision-making, and execution, rather than being treated as an implicit by-product of learning.

Several broader observations/insights emerge from this thesis. The first is that safety improvements are often non-substitutable across the perception–decision–control loop: strengthening one stage does not reliably compensate for brittleness elsewhere. This is most clearly illustrated by the combined COPPOL evaluation in Section 4.3.3, where calibrated hazard perception (MC-CP) and uncertainty-aware safety evaluation (USC) deliver additive safety benefits when

coupled in the same pipeline, reinforcing that safety must be treated as a systems property of the closed-loop architecture: it cannot be secured by optimising the policy alone, but requires making uncertainty and risk actionable at each stage where failures can originate, from perception through decision-making to execution-time filtering.

Safety mechanisms should be treated as controllers over a non-stationary learning process, not as static constraints applied uniformly throughout training. In practice for RL agents, the distribution of visited states, the reliability of learned predictors, and the agent's propensity for unsafe behaviour all evolve as learning progresses. Methods that can modulate their protective strength over time are therefore better aligned with the realities of exploration: early training often benefits from stronger safeguards to prevent catastrophic excursions, whereas later training benefits from relaxing unnecessary conservatism to avoid suppressing performance once competence is established. The adaptive behaviour shown across all proposed methods in this thesis (MC-CP, USC, and ADVICE) operationalise this principle directly, either through adaptive cautiousness, pessimism, or prediction set/interval/region size.

Finally, distribution shift should be treated as an inevitable operating condition for RL agents rather than as an auxiliary robustness scenario introduced only at evaluation time. In RL, the data distribution is intrinsically non-stationary: it is shaped by the evolving policy during training and is further perturbed at deployment by changes in sensing, dynamics, and environment composition. From this perspective, methods that merely test robustness to shift after training address the symptom rather than the cause. Instead, safety mechanisms must be constructed so that deviations from the training distribution are directly reflected in the signals that guide behaviour. This principle is reflected in the design of the proposed methods. MC-CP is explicitly formulated to regularise both ends of the predictive distribution, so that uncertainty naturally expands as inputs depart from the training support, rather than relying on post-hoc detection of out-of-distribution states. Similarly, ADVICE aims to learn a generalised notion of safety through experience-driven shielding, such that action filtering is governed by accumulated evidence of unsafe behaviour rather than by narrow assumptions tied to a fixed environment configuration. In both cases, distribution shift is not treated as an external stressor but as a first-class consideration that shapes how uncertainty and safety are represented and acted upon.

## 6.2 Positioning Contributions Within the Research Landscape

Figure 6.1 and Figure 6.2 situate the proposed methods within the UQ and safe RL classifications introduced in Chapter 2. Rather than redefining these classifications, the figures highlight how MC-CP, COPPOL, USC, and ADVICE occupy specific methodological categories while addressing limitations that remain unresolved in prior work.

Within UQ, MC-CP is situated under combined uncertainty methods and, more specifically, within split CP. Classical split CP provides finite-sample, distribution-free coverage guarantees but typically operates on deterministic predictors, implicitly assuming that the base model's uncertainty is either negligible or adequately captured by point estimates. In practice, even state-of-the-art split CP variants can be brittle under distributional and covariate shift: while

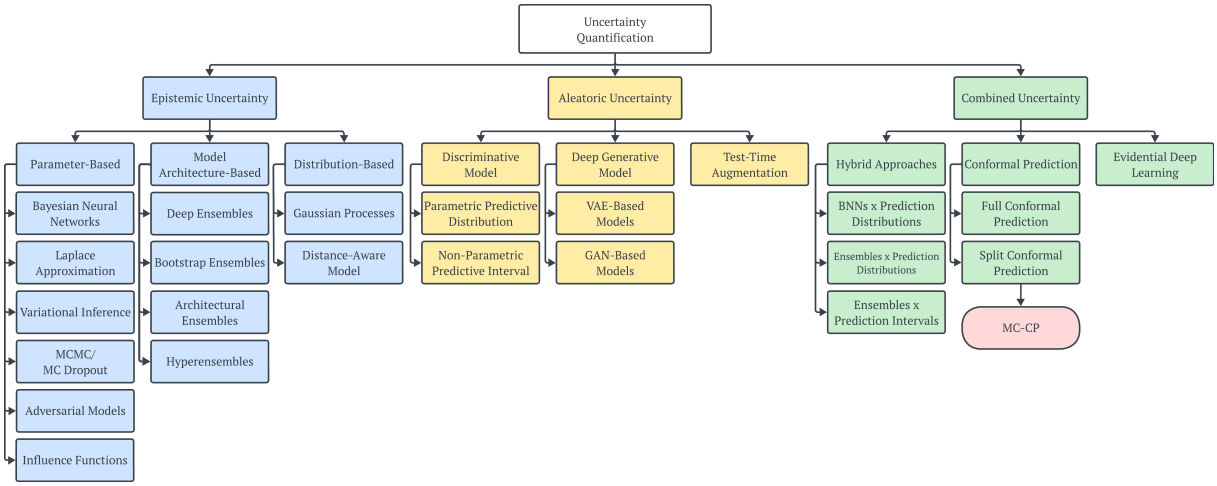



Figure 6.1: A classification of techniques for uncertainty quantification adapted from [55] decomposed into three categories: methods that quantify model uncertainty, methods that quantify data uncertainty, and methods that can model both. Light red denotes our contributions.

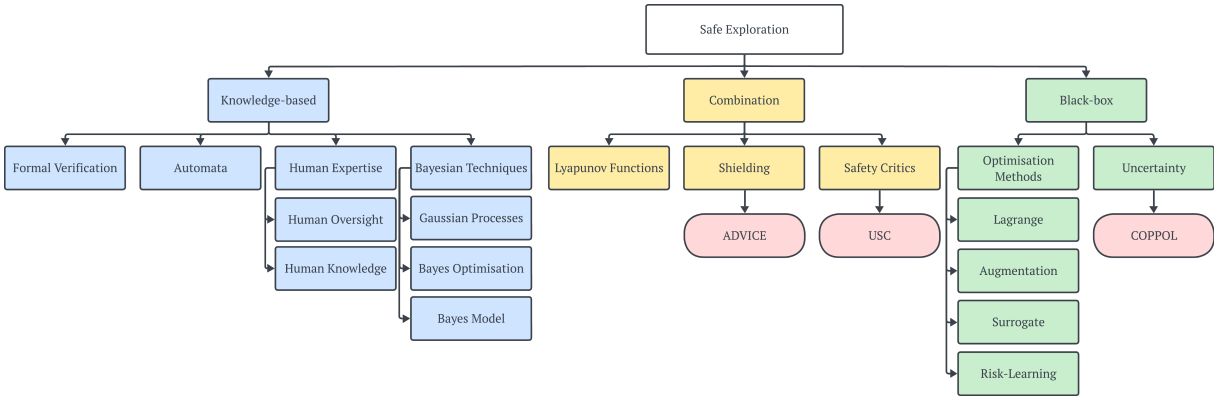



Figure 6.2: A classification of techniques for the safe exploration of RL agents during training decomposed into three categories: knowledge-based methods, black-box methods, and methods that can be either. Light red denotes our contributions.

marginal coverage may still hold in theory under exchangeability, the constructed sets can become poorly calibrated in a useful sense (e.g., failing to expand appropriately on shifted or ambiguous inputs), undermining their role as actionable uncertainty estimates in high-dimensional perception. MC-CP addresses this limitation by explicitly integrating stochastic epistemic sampling into the conformal calibration process, so that prediction sets and intervals expand in regions of epistemic uncertainty induced by limited data, shift, or model misspecification while retaining distribution-free validity. MC-CP also extends MC-Dropout approaches by introducing a computationally lightweight stochastic sampling procedure that preserves predictive uncertainty estimates without incurring the full cost of large numbers of forward passes. In doing so, MC-CP bridges stochastic uncertainty estimation and CP, yielding uncertainty representations that remain statistically grounded yet more responsive to shift and imperfect perception.

COPPOL is situated within black-box safe RL methods that leverage uncertainty to regulate

agent behaviour. Existing approaches in this category typically incorporate uncertainty estimates directly into the learning process, for example, by modifying the optimisation objective to discourage exploration in uncertain regions of the environment. While effective in some settings, this tightly couples safety to a particular learning algorithm and can limit modularity and reuse. COPPOL departs from this paradigm by leaving the underlying RL algorithm unchanged and instead acting at the perception–state interface: uncertainty-aware hazard predictions are used to alter the agent's state representation itself, ensuring that uncertainty about the environment is reflected in the features upon which decisions are made. As a result, COPPOL can be paired with standard, unmodified RL algorithms as well as specialised safe RL methods, providing a modular mechanism for propagating perceptual uncertainty into control without entangling safety enforcement with the learning objective.

USC is situated within the class of safety critic methods. In safe RL as a whole, existing methods that utilise conservative behaviour, enforce it uniformly across the state–action space, biasing cost estimates upward to reduce the risk of safety violations. While effective at preventing catastrophic failures, this uniform pessimism does not account for variations in uncertainty and often leads to excessive conservatism in regions where safety predictions are reliable. Moreover, existing safety critic approaches frequently struggle to shape cost gradients in a manner that accurately reflects the underlying hazard structure, resulting in blurred or misaligned safety boundaries that can hinder effective policy improvement. USC addresses these limitations by modulating conservative bias according to principled uncertainty estimates and by explicitly refining the learned safety landscape, enabling pessimism to be concentrated where safety predictions are unreliable while preserving sharp, informative cost gradients in confident regions.

ADVICE is situated within shielding-based methods for safe RL, operating as a post-hoc mechanism that filters or replaces proposed actions at execution time. A key limitation of existing shielding approaches is that they typically assume some form of explicit safety specification, such as a known cost function, constraint boundary, or tractable abstraction derived from environment structure, which makes them difficult to apply in continuous black-box settings where constraints are implicit, high-dimensional, or unavailable. ADVICE addresses this gap by learning a notion of safe versus unsafe behaviour directly from experience, without requiring access to an explicit cost function or hand-crafted constraint set, enabling shielding in continuous environments under minimal modelling assumptions. More broadly, whereas safe RL methods commonly apply fixed levels of conservatism or constraint strength throughout training, ADVICE adapts its intervention strength online according to observed safety performance, treating safety enforcement as a controller over a non-stationary learning process rather than a static rule applied uniformly.

## 6.3 Future Directions

Safety-critical RL ultimately demands methods that remain dependable when operating conditions, sensing, and behaviour distributions depart from those observed during training, and it therefore motivates safety mechanisms that can generalise beyond the specific regimes in

which they are developed and tuned. While this thesis introduced methods that generalise across regimes and deliver significant safety improvements by making uncertainty actionable at the perception, decision, and control stages, it also establishes a clear foundation for the next generation of safety architectures that unify these principles within a single, end-to-end design.

Firstly, MC-CP could be extended by trying to further reduce computational overheads brought on by the Adaptive MC Dropout module via approximations of the sampling process. Also, extending MC-CP to richer perceptual outputs and modalities (e.g., 3D images, multi-modal sensing, etc.) would broaden applicability while preserving clear, tunable coverage guarantees at the perception layer. A simpler direction is to evaluate MC-CP for regression under distribution shift, since the distributional shift experiments in Section 5.3.4 focus on classification. Stochastic quantile predictions may become more variable on unfamiliar inputs, and may behave differently to classification sets. Intuitively, shifts may alter both the location and spread of the target values, requiring prediction intervals to move or widen accordingly.

Next, COPPOL could be extended to couple calibrated hazard maps with constraint/risk-sensitive control (i.e., CVaR [181], shields [193], reachability) to obtain path-level assurances. Another direction could be feedback-aware calibration: periodic on-policy recalibration of MC-CP to preserve the target miscoverage rate $\alpha$ as the RL policy shifts the data distribution. Scaling COPPOL to richer settings, such as continuous control, longer horizons, and 3D environments, will likely require replacing hand-crafted penalties with principled objectives and operating on continuous geometric representations, but would significantly improve the applicability of COPPOL.

A natural extension is to broaden USC to domains in which an explicit cost signal is unavailable or difficult to specify manually (i.e., the environments in which ADVICE targets). One can learn a surrogate cost model from safety-related data (e.g., human annotations, system feedback, or inferred hazard labels) rather than assuming that the cost function is given a priori [264]. These learned cost models act as continuous approximations of safety risk and can be used as the cost signal for USC. A second direction is to improve the scalability of uncertainty estimation through cheaper approximations to influence scores [265] or by amortising uncertainty computation across updates [266]. Extending USC to richer settings, such as partial observability [267] and multi-objective settings [268], would clarify how uncertainty-aware conservatism interacts with complex exploration dynamics and whether critic uncertainty can be used as a principled signal for when to tighten or relax safety constraints during learning.

To improve ADVICE, a first direction is to reduce reliance on unsafe, unshielded exploration by incorporating safer bootstrapping strategies, such as conservative initial shields [194], human-in-the-loop labelling [154], or curriculum schedules [156] that gradually expand exploration as the representation improves. Relatedly, one can explore active data-collection policies that target informative boundary regions to learn separations more efficiently with fewer violations. Another direction is to exploit inconclusive features rather than throwing them away, while still avoiding hard, noisy labels by assigning soft labels based on empirical outcomes observed from similar prefixes (e.g., the fraction of rollouts in which that prefix, or a local neighbourhood of

it, eventually leads to failure/cost). These probabilities can then be used to weight contrastive pair construction and losses. Deeper theoretical and empirical studies into richer settings would clarify the boundaries of ADVICE's robustness. Extending to partial observability [198], multi-agent interaction [197], or tasks with non-trivial constraint geometry would test whether latent neighbourhood separability remains a reliable proxy for safety.

Looking at the higher level focus of this thesis, a natural next step is to move from purely statistical or purely learned safeguards toward hybrid safety scaffolds that combine rapid, structured protection with flexible data-driven refinement. One particularly promising direction is neuro-symbolic safety [269] for early-stage competence, where an explicit symbolic component provides conservative but intelligible safety constraints at the beginning of training, when the agent's data coverage is poorest and errors are most costly, while a neural component progressively relaxes, refines, or contextualises these constraints as evidence accumulates. In contrast to many current neuro-symbolic pipelines that treat the symbolic layer as static or complete, the safety setting motivates symbolic structures that are explicitly allowed to be incomplete and to evolve: rules can be treated as provisional hypotheses, updated via counterexamples and uncertainty signals, and parameterised to admit controlled violation only when the system has strong empirical support. An evolving neuro-symbolic layer could therefore serve as a principled bridge between cautious initial behaviour and performant late-stage policies, while offering an auditable account of why interventions occur.

An additional direction is to develop explicit composition principles for uncertainty and safety across the perception–decision–control loop. In the current pipeline, downstream components typically consume perceptual outputs as if they were ordinary observations, at most augmented with heuristic confidence, even though methods such as conformal prediction provide interpretable reliability statements about those outputs. A promising research direction is therefore to formalise interface contracts in which perception supplies not only a hazard estimate (or set-valued map), but also a calibrated certificate of coverage and error structure, and in which the decision-making component is designed to condition its behaviour on that certificate. Concretely, a policy or safety critic could treat regions marked by low guaranteed coverage as epistemically fragile and increase caution or trigger safeguards, while exploiting regions with strong guarantees more aggressively without resorting to blanket conservatism. More broadly, contracts of this form would enable end-to-end safety reasoning that respects where uncertainty originates and how it propagates, shifting safe RL from isolated component guarantees to systems that intentionally use those guarantees to regulate risk throughout learning and deployment.

Finally, given that distribution shift is a default condition for embodied RL agents, an important direction is shift-aware safety mechanisms that actively seek evidence rather than merely reacting to it. Modules can be extended into a closed-loop *safety probing* paradigm, where the system intentionally performs low-risk tests to reduce epistemic uncertainty in safety-relevant regions, updates its symbolic guardrails accordingly, and only then permits higher-reward behaviours. For example, carefully exploring the safety region boundary to acquire additional data, or observing hazards from different perspectives. This would benefit data-hungry mechanisms and potentially increase the robustness of the agent.

# Bibliography


[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing Atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[2] Jens Kober, J Andrew Bagnell, and Jan Peters. "Reinforcement learning in robotics: A survey". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1238–1274.

[3] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. "Deep reinforcement learning from human preferences". In: *Advances in neural information processing systems* 30 (2017).

[4] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. Vol. 1. 1. MIT press Cambridge, 1998.

[5] Csaba Szepesvári. *Algorithms for reinforcement learning*. Springer nature, 2022.

[6] Andrew Wagenmaker, Kevin Huang, Liyiming Ke, Kevin Jamieson, and Abhishek Gupta. "Overcoming the Sim-to-Real Gap: Leveraging Simulation to Learn to Explore for Real-World RL". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 78715–78765.

[7] Puze Liu, Haitham Bou-Ammar, Jan Peters, and Davide Tateo. "Safe reinforcement learning on the constraint manifold: Theory and applications". In: *IEEE Transactions on Robotics* (2025).

[8] Beverley Townsend, Colin Paterson, TT Arvind, Gabriel Nemirovsky, Radu Calinescu, Ana Cavalcanti, Ibrahim Habli, and Alan Thomas. "From pluralistic normative principles to autonomous-agent rules". In: *Minds and Machines* 32.4 (2022), pp. 683–715.

[9] Javier García and Diogo Shafie. "Teaching a humanoid robot to walk faster through safe reinforcement learning". In: *Engineering Applications of Artificial Intelligence* 88 (2020), p. 103360.

[10] Bharat Prakash, Nicholas R Waytowich, Ashwinkumar Ganesan, Tim Oates, and Tinoosh Mohsenin. "Guiding Safe Reinforcement Learning Policies Using Structured Language Constraints." In: *SafeAI@ AAAI*. 2020, pp. 153–161.

[11] Tristan Tomilin, Meng Fang, and Mykola Pechenizkiy. "HASARD: A Benchmark for Vision-Based Safe Reinforcement Learning in Embodied Agents". In: *The Thirteenth International Conference on Learning Representations*. 2025.

[12] Elad Osherov and Michael Lindenbaum. "Increasing CNN robustness to occlusions by reducing filter support". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 550–561.

[13] John Molloy and John McDermid. "Safety Assessment for Autonomous Systems' Perception Capabilities". In: *arXiv preprint arXiv:2208.08237* (2022).

[14] Javier Garcıa and Fernando Fernández. "A comprehensive survey on safe reinforcement learning". In: *Journal of Machine Learning Research* 16.1 (2015), pp. 1437–1480.

[15] Jordan Frank, Shie Mannor, and Doina Precup. "Reinforcement learning in the presence of rare events". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 336–343.

[16] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. "Robot Operating System 2: Design, architecture, and uses in the wild". In: *Science robotics* 7.66 (2022), eabm6074.

[17] Andrei Marinescu, Ivana Dusparic, Adam Taylor, Vinny Cahill, and Siobhán Clarke. "Decentralised multi-agent reinforcement learning for dynamic and uncertain environments". In: *arXiv preprint arXiv:1409.4561* (2014).

[18] Nicolás Cardozo and Ivana Dusparic. "Adaptation to unknown situations as the holy grail of learning-based self-adaptive systems: Research directions". In: *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE. 2021, pp. 252–253.

[19] Kenji Doya. "Reinforcement learning: Computational theory and biological mechanisms". In: *HFSP journal* 1.1 (2007), p. 30.

[20] Apostolos N Burnetas and Michael N Katehakis. "Optimal adaptive policies for Markov decision processes". In: *Mathematics of Operations Research* 22.1 (1997), pp. 222–255.

[21] Richard Bellman. "A Markovian decision process". In: *Journal of mathematics and mechanics* (1957), pp. 679–684.

[22] Paul A Gagniuc. *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017.

[23] Dimitri Bertsekas and John N Tsitsiklis. *Introduction to probability*. Vol. 1. Athena Scientific, 2008.

[24] "Bellman Equation". In: *Encyclopedia of Machine Learning*. Boston, MA: Springer US, 2010, pp. 97–97. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_71. URL: https://doi.org/10.1007/978-0-387-30164-8_71.

[25] Xianping Guo, Onésimo Hernández-Lerma, Xianping Guo, and Onésimo Hernández-Lerma. *Continuous-time Markov decision processes*. Springer, 2009.

[26] Donald A Berry and Bert Fristedt. "Bandit problems: sequential allocation of experiments (Monographs on statistics and applied probability)". In: *London: Chapman and Hall* 5.71-87 (1985), pp. 7–7.

[27] Christopher JCH Watkins and Peter Dayan. "Q-learning". In: *Machine learning* 8 (1992), pp. 279–292.

[28] Fanghui Liu, Luca Viano, and Volkan Cevher. "Understanding deep neural function approximation in reinforcement learning via $\epsilon$-greedy exploration". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 5093–5108.

[29] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*. Vol. 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[30] Harm Van Seijen, Hado Van Hasselt, Shimon Whiteson, and Marco Wiering. "A theoretical and empirical analysis of expected sarsa". In: *2009 ieee symposium on adaptive dynamic programming and reinforcement learning*. IEEE. 2009, pp. 177–184.

[31] Hado Van Hasselt, Arthur Guez, and David Silver. "Deep reinforcement learning with double q-learning". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 30. 1. 2016.

[32] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. "Dueling network architectures for deep reinforcement learning". In: *International conference on machine learning*. PMLR. 2016, pp. 1995–2003.

[33] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. "Distributed Prioritized Experience Replay". In: *International Conference on Learning Representations*. 2018.

[34] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. "Policy gradient methods for reinforcement learning with function approximation". In: *Advances in neural information processing systems* 12 (1999).

[35] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8 (1992), pp. 229–256.

[36] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. "Trust region policy optimization". In: *International conference on machine learning*. PMLR. 2015, pp. 1889–1897.

[37] Solomon Kullback and Richard A Leibler. "On information and sufficiency". In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[38] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[39] Vijay Konda and John Tsitsiklis. "Actor-critic algorithms". In: *Advances in neural information processing systems* 12 (1999).

[40] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. "Asynchronous methods for deep reinforcement learning". In: *International conference on machine learning*. PmLR. 2016, pp. 1928–1937.

[41] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[42] Scott Fujimoto, Herke Hoof, and David Meger. "Addressing function approximation error in actor-critic methods". In: *International conference on machine learning*. PMLR. 2018, pp. 1587–1596.

[43] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International conference on machine learning*. Pmlr. 2018, pp. 1861–1870.

[44] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. "Soft actor-critic algorithms and applications". In: *arXiv preprint arXiv:1812.05905* (2018).

[45] Thomas M Moerland, Joost Broekens, Aske Plaat, Catholijn M Jonker, et al. "Model-based reinforcement learning: A survey". In: *Foundations and Trends® in Machine Learning* 16.1 (2023), pp. 1–118.

[46] Richard Bellman. "Dynamic programming". In: *science* 153.3731 (1966), pp. 34–37.

[47] KS Holkar and Laxman M Waghmare. "An overview of model predictive control". In: *International Journal of control and automation* 3.4 (2010), pp. 47–63.

[48] Richard S Sutton. "Integrated modeling and control based on reinforcement learning and dynamic programming". In: *Advances in neural information processing systems* 3 (1990).

[49] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. "When to trust your model: Model-based policy optimization". In: *Advances in neural information processing systems* 32 (2019).

[50] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. "Learning latent dynamics for planning from pixels". In: *International conference on machine learning*. PMLR. 2019, pp. 2555–2565.

[51] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. "Mastering atari with discrete world models". In: *arXiv preprint arXiv:2010.02193* (2020).

[52] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. "Mastering diverse domains through world models". In: *arXiv preprint arXiv:2301.04104* (2023).

[53] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. "A review of uncertainty quantification in deep learning: Techniques, applications and challenges". In: *Information fusion* 76 (2021), pp. 243–297.

[54] Eyke Hüllermeier and Willem Waegeman. "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods". In: *Machine learning* 110.3 (2021), pp. 457–506.

[55] Wenchong He and Z Jiang. "A survey on uncertainty quantification methods for deep neural networks: An uncertainty source perspective". In: *perspective* 1 (2023), p. 88.

[56] Vincent Fortuin. "Priors in bayesian deep learning: A review". In: *International Statistical Review* 90.3 (2022), pp. 563–591.

[57] Ethan Goan and Clinton Fookes. "Bayesian neural networks: An introduction and survey". In: *Case Studies in Applied Bayesian Data Science: CIRM Jean-Morlet Chair, Fall 2018* (2020), pp. 45–87.

[58] David JC MacKay. "Bayesian interpolation". In: *Neural computation* 4.3 (1992), pp. 415–447.

[59] Hippolyt Ritter, Aleksandar Botev, and David Barber. "A scalable laplace approximation for neural networks". In: *6th international conference on learning representations, ICLR 2018-conference track proceedings*. Vol. 6. International Conference on Representation Learning. 2018.

[60] Alexander Immer, Maciej Korzepa, and Matthias Bauer. "Improving predictions of Bayesian neural nets via local linearization". In: *International conference on artificial intelligence and statistics*. PMLR. 2021, pp. 703–711.

[61] Zhijie Deng, Feng Zhou, and Jun Zhu. "Accelerated linearized Laplace approximation for Bayesian deep learning". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 2695–2708.

[62] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. "Variational inference: A review for statisticians". In: *Journal of the American statistical Association* 112.518 (2017), pp. 859–877.

[63] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. "Stochastic variational inference". In: *the Journal of machine Learning research* 14.1 (2013), pp. 1303–1347.

[64] Rajesh Ranganath, Sean Gerrish, and David Blei. "Black box variational inference". In: *Artificial intelligence and statistics*. PMLR. 2014, pp. 814–822.

[65] Ankush Ganguly, Sanjana Jain, and Ukrit Watchareeruetai. "Amortized variational inference: A systematic review". In: *Journal of Artificial Intelligence Research* 78 (2023), pp. 167–215.

[66] Yarin Gal and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning". In: *international conference on machine learning*. PMLR. 2016, pp. 1050–1059.

[67] Kai Brach, Beate Sick, and Oliver Dürr. "Single Shot MC Dropout Approximation". In: *ICML 2020 Workshop on Uncertainty and Ro-bustness in Deep Learning, July 17, 2020, virtual*. 2020.

[68] Charles J Geyer. "Practical markov chain monte carlo". In: *Statistical science* (1992), pp. 473–483.

[69] Kajetan Schweighofer, Lukas Aichberger, Mykyta Ielanskyi, Günter Klambauer, and Sepp Hochreiter. "Quantification of uncertainty with adversarial models". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 19446–19484.

[70] Pang Wei Koh and Percy Liang. "Understanding black-box predictions via influence functions". In: *International conference on machine learning*. PMLR. 2017, pp. 1885–1894.

[71] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles". In: *Advances in neural information processing systems* 30 (2017).

[72] Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. "Hyperparameter ensembles for robustness and uncertainty quantification". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6514–6527.

[73] Jeremy Nixon, Balaji Lakshminarayanan, and Dustin Tran. "Why are bootstrapped deep ensembles not better?" In: *"I Can't Believe It's Not Better!"NeurIPS 2020 workshop*. 2020.

[74] Konstantinos Kamnitsas, Wenjia Bai, Enzo Ferrante, Steven McDonagh, Matthew Sinclair, Nick Pawlowski, Martin Rajchl, Matthew Lee, Bernhard Kainz, Daniel Rueckert, et al. "Ensembles of multiple models and architectures for robust brain tumour segmentation". In: *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: Third International Workshop, BrainLes 2017, Held in Conjunction with MICCAI 2017, Quebec City, QC, Canada, September 14, 2017, Revised Selected Papers 3*. Springer. 2018, pp. 450–462.

[75] Carl Edward Rasmussen. "Gaussian processes in machine learning". In: *Summer school on machine learning*. Springer, 2003, pp. 63–71.

[76] Edward Snelson and Zoubin Ghahramani. "Sparse Gaussian processes using pseudo-inputs". In: *Advances in neural information processing systems* 18 (2005).

[77] Andreas Damianou and Neil D Lawrence. "Deep gaussian processes". In: *Artificial intelligence and statistics*. PMLR. 2013, pp. 207–215.

[78] Jeremiah Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax Weiss, and Balaji Lakshminarayanan. "Simple and principled uncertainty estimation with deterministic deep learning via distance awareness". In: *Advances in neural information processing systems* 33 (2020), pp. 7498–7512.

[79] Joost Van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. "Uncertainty estimation using a single deep deterministic neural network". In: *International conference on machine learning*. PMLR. 2020, pp. 9690–9700.

[80] Jishnu Mukhoti, Andreas Kirsch, Joost van Amersfoort, PH Torr, and Yarin Gal. "Deterministic neural networks with inductive biases capture epistemic and aleatoric uncertainty". In: *arXiv preprint arXiv:2102.11582* 2 (2021).

[81] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[82] Christos Karras, Aristeidis Karras, Markos Avlonitis, and Spyros Sioutas. "An overview of mcmc methods: From theory to applications". In: *IFIP international conference on artificial intelligence applications and innovations*. Springer. 2022, pp. 319–332.

[83] R v Mises. "On the asymptotic distribution of differentiable statistical functions". In: *The annals of mathematical statistics* 18.3 (1947), pp. 309–348.

[84] James Martens. "Deep learning via Hessian-free optimization". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel, 2010, pp. 735–742.

[85] Simon Bachstein. "Uncertainty quantification in deep learning". In: *Master Thesis* (2019).

[86] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. "On calibration of modern neural networks". In: *International conference on machine learning*. PMLR. 2017, pp. 1321–1330.

[87] Alex Kendall and Yarin Gal. "What uncertainties do we need in bayesian deep learning for computer vision?" In: *Advances in neural information processing systems* 30 (2017).

[88] Axel Brando Guillaumes. "Mixture density networks for distribution and uncertainty estimation". MA thesis. Universitat Politècnica de Catalunya, 2017.

[89] Abbas Khosravi, Saeid Nahavandi, Doug Creighton, and Amir F Atiya. "Lower upper bound estimation method for construction of neural network-based prediction intervals". In: *IEEE transactions on neural networks* 22.3 (2010), pp. 337–346.

[90] Haoxian Chen, Ziyi Huang, Henry Lam, Huajie Qian, and Haofeng Zhang. "Learning prediction intervals for regression: Generalization and calibration". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 820–828.

[91] Natasa Tagasovska and David Lopez-Paz. "Single-model uncertainties for deep learning". In: *Advances in neural information processing systems* 32 (2019).

[92] Diederik P Kingma, Max Welling, et al. *Auto-encoding variational bayes*. 2013.

[93] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. "Ladder variational autoencoders". In: *Advances in neural information processing systems* 29 (2016).

[94] Danilo Rezende and Shakir Mohamed. "Variational inference with normalizing flows". In: *International conference on machine learning*. PMLR. 2015, pp. 1530–1538.

[95] Rogan Morrow and Wei-Chen Chiu. "Variational autoencoders with normalizing flow decoders". In: *arXiv preprint arXiv:2004.05617* (2020).

[96] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets". In: *Advances in neural information processing systems* 27 (2014).

[97] Mehdi Mirza and Simon Osindero. "Conditional generative adversarial nets". In: *arXiv preprint arXiv:1411.1784* (2014).

[98] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Georg Langs, and Ursula Schmidt-Erfurth. "f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks". In: *Medical image analysis* 54 (2019), pp. 30–44.

[99] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. "Generating adversarial examples with adversarial networks". In: *arXiv preprint arXiv:1801.02610* (2018).

[100] Yunus Saatci and Andrew G Wilson. "Bayesian GAN". In: *Advances in neural information processing systems* 30 (2017).

[101] Junbo Zhao, Michael Mathieu, and Yann LeCun. "Energy-based generative adversarial network". In: *arXiv preprint arXiv:1609.03126* (2016).

[102] Murat Seckin Ayhan and Philipp Berens. "Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks". In: *Medical Imaging with Deep Learning*. 2018.

[103] Aliaksandr Hubin and Geir Storvik. "Combining model and parameter uncertainty in Bayesian neural networks". In: *arXiv preprint arXiv:1903.07594* (2019).

[104] Tim Pearce, Alexandra Brintrup, Mohamed Zaki, and Andy Neely. "High-quality prediction intervals for deep learning: A distribution-free, ensembled approach". In: *International conference on machine learning*. PMLR. 2018, pp. 4075–4084.

[105] Tárik S Salem, Helge Langseth, and Heri Ramampiaro. "Prediction intervals: Split normal mixture from quality-driven deep ensembles". In: *Conference on Uncertainty in Artificial Intelligence*. PMLR. 2020, pp. 1179–1187.

[106] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Vol. 29. Springer, 2005.

[107] Anastasios N Angelopoulos, Stephen Bates, et al. "Conformal prediction: A gentle introduction". In: *Foundations and Trends® in Machine Learning* 16.4 (2023), pp. 494–591.

[108] Matteo Fontana, Gianluca Zeni, and Simone Vantini. "Conformal prediction: a unified review of theory and new challenges". In: *Bernoulli* 29.1 (2023), pp. 1–23.

[109] Vladimir Vovk. "Cross-conformal predictors". In: *Annals of Mathematics and Artificial Intelligence* 74 (2015), pp. 9–28.

[110] Yaniv Romano, Matteo Sesia, and Emmanuel Candes. "Classification with valid and adaptive coverage". In: *Advances in neural information processing systems* 33 (2020), pp. 3581–3591.

[111] Anastasios Angelopoulos, Stephen Bates, Jitendra Malik, and Michael I Jordan. "Uncertainty sets for image classifiers using conformal prediction". In: *arXiv preprint arXiv:2009.14193* (2020).

[112] Yaniv Romano, Evan Patterson, and Emmanuel Candes. "Conformalized quantile regression". In: *Advances in neural information processing systems* 32 (2019).

[113] Xing Han, Ziyang Tang, Joydeep Ghosh, and Qiang Liu. "Split localized conformal prediction". In: *arXiv preprint arXiv:2206.13092* (2022).

[114] Margaux Zaffran, Olivier Féron, Yannig Goude, Julie Josse, and Aymeric Dieuleveut. "Adaptive conformal predictions for time series". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 25834–25866.

[115] Chen Xu and Yao Xie. "Conformal prediction interval for dynamic time-series". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 11559–11569.

[116] Anastasios N Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. "Conformal risk control". In: *arXiv preprint arXiv:2208.02814* (2022).

[117] Murat Sensoy, Lance Kaplan, and Melih Kandemir. "Evidential deep learning to quantify classification uncertainty". In: *Advances in neural information processing systems* 31 (2018).

[118] Deep Shankar Pandey and Qi Yu. "Learn to accumulate evidence from all training samples: theory and practice". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 26963–26989.

[119] Danruo Deng, Guangyong Chen, Yang Yu, Furui Liu, and Pheng-Ann Heng. "Uncertainty estimation by fisher information-based evidential deep learning". In: *International conference on machine learning*. PMLR. 2023, pp. 7596–7616.

[120] Maohao Shen, Yuheng Bu, Prasanna Sattigeri, Soumya Ghosh, Subhro Das, and Gregory Wornell. "Post-hoc uncertainty learning using a dirichlet meta-model". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 37. 8. 2023, pp. 9772–9781.

[121] Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. "Deep evidential regression". In: *Advances in neural information processing systems* 33 (2020), pp. 14927–14937.

[122] Nis Meinert, Jakob Gawlikowski, and Alexander Lavin. "The unreasonable effectiveness of deep evidential regression". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 8. 2023, pp. 9134–9142.

[123] Yuefei Wu, Bin Shi, Bo Dong, Qinghua Zheng, and Hua Wei. "The evidence contraction issue in deep evidential regression: Discussion and solution". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. 19. 2024, pp. 21726–21734.

[124] Kai Ye, Tiejin Chen, Hua Wei, and Liang Zhan. "Uncertainty regularized evidential regression". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. 15. 2024, pp. 16460–16468.

[125] Dharmesh Tailor, Alvaro Correia, Eric Nalisnick, and Christos Louizos. "Approximating Full Conformal Prediction for Neural Network Regression with Gauss-Newton Influence". In: *The Thirteenth International Conference on Learning Representations* (2025).

[126] Niklas Möller, Sven Ove Hansson, and Martin Peterson. "Safety is more than the antonym of risk". In: *Journal of Applied Philosophy* 23.4 (2006), pp. 419–432.

[127] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC). *ISO/IEC Guide 51:2014: Safety aspects – Guidelines for their inclusion in standards*. International Organization for Standardization. 2014. URL: `https://www.iso.org/obp/ui/#iso:std:iso-iec:guide:51:ed-3:v1:en`.

[128] Adam Haskard and Damith Herath. "Secure Robotics: Navigating Challenges at the Nexus of Safety, Trust, and Cybersecurity in Cyber-Physical Systems". In: *ACM Computing Surveys* (2025).

[129] Audrey Lyndon, Dána-Ain Davis, Anjana E Sharma, and Karen A Scott. *Emotional safety is patient safety*. 2023.

[130] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. "Concrete problems in AI safety". In: *arXiv preprint arXiv:1606.06565* (2016).

[131] Inioluwa Deborah Raji and Roel Dobbe. "Concrete problems in AI safety, revisited". In: *arXiv preprint arXiv:2401.10899* (2023).

[132] David Leslie. "Understanding artificial intelligence ethics and safety". In: *arXiv preprint arXiv:1906.05684* (2019).

[133] Jiaju Qi, Qihao Zhou, Lei Lei, and Kan Zheng. "Federated reinforcement learning: techniques, applications, and open challenges". In: *Intelligence & Robotics* 1.1 (2021), pp. 18–57.

[134] Sandhya Saisubramanian, Shlomo Zilberstein, and Ece Kamar. "Avoiding negative side effects due to incomplete knowledge of AI systems". In: *AI magazine* 42.4 (2022), pp. 62–71.

[135] Zeyang Li, Chuxiong Hu, Yunan Wang, Yujie Yang, and Shengbo Eben Li. "Safe reinforcement learning with dual robustness". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).

[136] Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. "A Survey of Reinforcement Learning from Human Feedback". In: *arXiv e-prints* (2023), arXiv–2312.

[137] Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. "Safe rlhf: Safe reinforcement learning from human feedback". In: *arXiv preprint arXiv:2310.12773* (2023).

[138] Tom Yan and Zachary Lipton. "A theoretical case-study of Scalable Oversight in Hierarchical Reinforcement Learning". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 27295–27339.

[139] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. "Inverse reward design". In: *Advances in neural information processing systems* 30 (2017).

[140] Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. "Defining and characterizing reward gaming". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 9460–9471.

[141] Eitan Altman. *Constrained Markov decision processes*. Routledge, 2021.

[142] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. "Verifiable reinforcement learning via policy extraction". In: *Advances in neural information processing systems* 31 (2018).

[143] Dennis Gross, Nils Jansen, Sebastian Junges, and Guillermo A Pérez. "COOL-MC: a comprehensive tool for reinforcement learning and model checking". In: *International Symposium on Dependable Software Engineering: Theories, Tools, and Applications*. Springer. 2022, pp. 41–49.

[144] Dennis Gross and Helge Spieker. "Probabilistic model checking of stochastic reinforcement learning policies". In: *arXiv preprint arXiv:2403.18725* (2024).

[145] George Rupert Mason, Radu Constantin Calinescu, Daniel Kudenko, and Alec Banks. "Assured reinforcement learning with formally verified abstract policies". In: *9th International Conference on Agents and Artificial Intelligence (ICAART)*. York. 2017.

[146] Nathan Fulton and André Platzer. "Safe reinforcement learning via formal methods: Toward safe control through proof and learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[147] TD Simão, Nils Jansen, and MTJ Spaan. "AlwaysSafe: Reinforcement Learning without Safety Constraint Violations during Training". In: *20th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems. 2021, pp. 1226–1235.

[148] Joshua Riley, Radu Calinescu, Colin Paterson, Daniel Kudenko, and Alec Banks. "Reinforcement learning with quantitative verification for assured multi-agent policies". In: *13th International Conference on Agents and Artificial Intelligence*. York. 2021.

[149] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. "Cautious Reinforcement Learning with Logical Constraints". In: *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. 2020, pp. 483–491.

[150] Mohammadhosein Hasanbeig, Daniel Kroening, and Alessandro Abate. "LCRL: Certified policy synthesis via logically-constrained reinforcement learning". In: *International Conference on Quantitative Evaluation of SysTems*. Springer. 2022, pp. 217–231.

[151] Marco Bagatella, Andreas Krause, and Georg Martius. "Directed Exploration in Reinforcement Learning from Linear Temporal Logic". In: *Seventeenth European Workshop on Reinforcement Learning*. 2025.

[152] Mingyu Cai, Shaoping Xiao, Junchao Li, and Zhen Kan. "Safe reinforcement learning under temporal logic with reward design and quantum action selection". In: *Scientific reports* 13.1 (2023), p. 1925.

[153] Laurent Orseau and M Armstrong. "Safely interruptible agents". In: *Conference on Uncertainty in Artificial Intelligence*. Association for Uncertainty in Artificial Intelligence. 2016.

[154] S Saunders, G Sastry, A Stuhlmüller, and O Evans. "Trial without error: Towards safe reinforcement learning via human intervention". In: *17th International Conference on Autonomous Agents and MultiAgent Systems*. ACM Digital Library. 2018.

[155] Yunkun Xu, Zhenyu Liu, Guifang Duan, Jiangcheng Zhu, Xiaolong Bai, and Jianrong Tan. "Look before you leap: Safe model-based reinforcement learning with human intervention". In: *Conference on Robot Learning*. PMLR. 2022, pp. 332–341.

[156] Kelvin Toonen and Thiago Simão. "Making Reinforcement Learning Safer via Curriculum Learning". In: *International Conference on Agents and Artificial Intelligence*. 2025.

[157] Matteo Turchetta, Andrey Kolobov, Shital Shah, Andreas Krause, and Alekh Agarwal. "Safe reinforcement learning via curriculum induction". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 12151–12162.

[158] Benjamin Plaut, Juan LiÃŠvano-Karim, and Stuart Russell. "Asking for Help Enables Safety Guarantees Without Sacrificing Effectiveness". In: *arXiv preprint arXiv:2502.14043* (2025).

[159] Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. "Safe exploration in finite markov decision processes with gaussian processes". In: *Advances in neural information processing systems* 29 (2016).

[160] Akifumi Wachi, Yanan Sui, Yisong Yue, and Masahiro Ono. "Safe exploration and optimization of constrained mdps using gaussian processes". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[161] Jiameng Fan and Wenchao Li. "Safety-guided deep reinforcement learning via online gaussian process estimation". In: *arXiv preprint arXiv:1903.02526* (2019).

[162] Yarden As, Bhavya Sukhija, and Andreas Krause. "Safe exploration using bayesian world models and log-barrier optimization". In: *arXiv preprint arXiv:2405.05890* (2024).

[163] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[164] Salomon Sickert, Javier Esparza, Stefan Jaax, and Jan Křetínskỳ. "Limit-deterministic Büchi automata for linear temporal logic". In: *International Conference on Computer Aided Verification*. Springer. 2016, pp. 312–332.

[165] Teodor Mihai Moldovan and Pieter Abbeel. "Safe exploration in Markov decision processes". In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*. 2012, pp. 1451–1458.

[166] Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. "When Gaussian process meets big data: A review of scalable GPs". In: *IEEE transactions on neural networks and learning systems* 31.11 (2020), pp. 4405–4423.

[167] Ivana Dusparic and Vinny Cahill. "Using reinforcement learning for multi-policy optimization in decentralized autonomic systems–an experimental evaluation". In: *International Conference on autonomic and trusted computing*. Springer. 2009, pp. 105–119.

[168] Alex Ray, Joshua Achiam, and Dario Amodei. "Benchmarking safe exploration in deep reinforcement learning". In: *arXiv preprint arXiv:1910.01708* 7.1 (2019), p. 2.

[169] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. "Constrained policy optimization". In: *International conference on machine learning*. PMLR. 2017, pp. 22–31.

[170] Chen Tessler, Daniel J Mankowitz, and Shie Mannor. "Reward constrained policy optimization". In: *arXiv preprint arXiv:1805.11074* (2018).

[171] Santiago Paternain, Miguel Calvo-Fullana, Luiz FO Chamon, and Alejandro Ribeiro. "Safe policies for reinforcement learning via primal-dual methods". In: *IEEE Transactions on Automatic Control* 68.3 (2022), pp. 1321–1336.

[172] Adam Stooke, Joshua Achiam, and Pieter Abbeel. "Responsive safety in reinforcement learning by pid lagrangian methods". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 9133–9143.

[173] Sergio Rozada, Dongsheng Ding, Antonio G Marques, and Alejandro Ribeiro. "Deterministic policy gradient primal-dual methods for continuous-space constrained mdps". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 39. 19. 2025, pp. 20200–20208.

[174] Ming Yu, Zhuoran Yang, Mladen Kolar, and Zhaoran Wang. "Convergent policy optimization for safe reinforcement learning". In: *Advances in Neural Information Processing Systems* 32 (2019).

[175] Aivar Sootla, Alexander I Cowen-Rivers, Taher Jafferjee, Ziyan Wang, David H Mguni, Jun Wang, and Haitham Ammar. "Sauté rl: Almost surely safe reinforcement learning using state augmentation". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 20423–20443.

[176] Zhuo Li, Derui Zhu, and Jens Grossklags. "Safe Reinforcement Learning via Episodic Control". In: *IEEE Access* (2025).

[177] Gregory Kahn, Adam Villaflor, Vitchyr Pong, Pieter Abbeel, and Sergey Levine. "Uncertainty-aware reinforcement learning for collision avoidance". In: *arXiv preprint arXiv:1702.01182* (2017).

[178] Yarden As, Bhavya Sukhija, Lenart Treven, Carmelo Sferrazza, Stelian Coros, and Andreas Krause. "ActSafe: Active Exploration with Safety Constraints for Reinforcement Learning". In: *The Thirteenth International Conference on Learning Representations*. 2024.

[179] Arushi Jain, Khimya Khetarpal, and Doina Precup. "Safe option-critic: learning safety in the option-critic architecture". In: *The Knowledge Engineering Review* 36 (2021), e4.

[180] Rakesh P Borase, DK Maghade, SY Sondkar, and SN Pawar. "A review of PID control, tuning methods and applications". In: *International Journal of Dynamics and Control* 9.2 (2021), pp. 818–827.

[181] R Tyrrell Rockafellar, Stanislav Uryasev, et al. "Optimization of conditional value-at-risk". In: *Journal of risk* 2 (2000), pp. 21–42.

[182] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. "Risk-constrained reinforcement learning with percentile risk criteria". In: *Journal of Machine Learning Research* 18.167 (2018), pp. 1–51.

[183] Qisong Yang, Thiago D Simão, Simon H Tindemans, and Matthijs TJ Spaan. "WCSAC: Worst-case soft actor critic for safety-constrained reinforcement learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 12. 2021, pp. 10639–10646.

[184] Qisong Yang, Thiago D Simão, Simon H Tindemans, and Matthijs TJ Spaan. "Safety-constrained reinforcement learning with a distributional safety critic: Q. Yang et al." In: *Machine Learning* 112.3 (2023), pp. 859–887.

[185] Radu Marinescu, Elizabeth Daly, and Ivana Dusparic. "Optimistic Exploration for Risk-Averse Constrained Reinforcement Learning". In: *European Conference on Artificial Intelligence*. 2025.

[186] Theodore Perkins and Andrew Barto. "Lyapunov design for safe reinforcement learning control". In: *Safe Learning Agents: Papers from the 2002 AAAI Symposium*. 2002, pp. 23–30.

[187] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. "Safe model-based reinforcement learning with stability guarantees". In: *Advances in neural information processing systems* 30 (2017).

[188] Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. "A lyapunov-based approach to safe reinforcement learning". In: *Advances in neural information processing systems* 31 (2018).

[189] Yinlam Chow, Ofir Nachum, Aleksandra Faust, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. "Lyapunov-based safe policy optimization for continuous control". In: *arXiv preprint arXiv:1901.10031* (2019).

[190] Subin Huh and Insoon Yang. "Safe reinforcement learning for probabilistic reachability and safety specifications: A Lyapunov-based approach". In: *arXiv preprint arXiv:2002.10126* (2020).

[191] Ashkan B Jeddi, Nariman L Dehghani, and Abdollah Shafieezadeh. "Lyapunov-based uncertainty-aware safe reinforcement learning". In: *arXiv preprint arXiv:2107.13944* (2021).

[192] Pol Mestres, Arnau Marzabal, and Jorge Cortés. "Anytime safe reinforcement learning". In: *arXiv preprint arXiv:2504.16417* (2025).

[193] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. "Safe reinforcement learning via shielding". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[194] Nils Jansen, Bettina Könighofer, Sebastian Junges, Alexandru C Serban, and Roderick Bloem. "Safe reinforcement learning via probabilistic shields". In: *arXiv preprint arXiv:1807.06096* (2018).

[195] Wen-Chi Yang, Giuseppe Marra, Gavin Rens, and Luc De Raedt. "Safe reinforcement learning via probabilistic logic shields". In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. 2023, pp. 5739–5749.

[196] Francesco Belardinelli, Alexander W Goodall, et al. "Probabilistic Shielding for Safe Reinforcement Learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 39. 15. 2025, pp. 16091–16099.

[197] Ingy ElSayed-Aly, Suda Bharadwaj, Christopher Amato, Rüdiger Ehlers, Ufuk Topcu, and Lu Feng. "Safe Multi-Agent Reinforcement Learning via Shielding". In: *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. 2021, pp. 483–491.

[198] Steven Carr, Nils Jansen, Sebastian Junges, and Ufuk Topcu. "Safe reinforcement learning via shielding for pomdps". In: *arXiv preprint arXiv:2204.00755* (2022).

[199] Kyungmin Kim, Davide Corsi, Andoni Rodriguez, JB Lanier, Benjami Parellada, Pierre Baldi, César Sánchez, and Roy Fox. "Realizable Continuous-Space Shields for Safe Reinforcement Learning". In: *arXiv preprint arXiv:2410.02038* (2024).

[200] Davide Corsi, Guy Amir, Andoni Rodríguez, César Sánchez, Guy Katz, and Roy Fox. "Verification-Guided Shielding for Deep Reinforcement Learning". In: *arXiv preprint arXiv:2406.06507* (2024).

[201] Alexander W Goodall and Francesco Belardinelli. "Approximate shielding of atari agents for safe exploration". In: *arXiv preprint arXiv:2304.11104* (2023).

[202] Alexander W Goodall and Francesco Belardinelli. "Approximate model-based shielding for safe reinforcement learning". In: *ECAI 2023*. IOS Press, 2023, pp. 883–890.

[203] Martin Tappler, Stefan Pranger, Bettina Könighofer, Edi Muškardin, Roderick Bloem, and Kim Larsen. "Automata learning meets shielding". In: *International Symposium on Leveraging Applications of Formal Methods*. Springer. 2022, pp. 335–359.

[204] Masaki Waga, Ezequiel Castellano, Sasinee Pruekprasert, Stefan Klikovits, Toru Takisaka, and Ichiro Hasuo. "Dynamic shielding for reinforcement learning in black-box environments". In: *International Symposium on Automated Technology for Verification and Analysis*. Springer. 2022, pp. 25–41.

[205] Shahaf S Shperberg, Bo Liu, and Peter Stone. "Learning a shield from catastrophic action effects: never repeat the same mistake". In: *arXiv preprint arXiv:2202.09516* (2022).

[206] Haonan Yu, Wei Xu, and Haichao Zhang. "Towards safe reinforcement learning with a safety editor policy". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 2608–2621.

[207] Greg Anderson, Swarat Chaudhuri, and Isil Dillig. "Guiding safe exploration with weakest preconditions". In: *arXiv preprint arXiv:2209.14148* (2022).

[208] Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. "Safe exploration in continuous action spaces". In: *arXiv preprint arXiv:1801.08757* (2018).

[209] Zhaorun Chen, Zhuokai Zhao, Tairan He, Binhao Chen, Xuhao Zhao, Liang Gong, and Chengliang Liu. "Safe reinforcement learning via hierarchical adaptive chance-constraint safeguards". In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2024, pp. 12378–12385.

[210] Jonas Günster, Puze Liu, Jan Peters, and Davide Tateo. "Handling long-term safety and uncertainty in safe reinforcement learning". In: *arXiv preprint arXiv:2409.12045* (2024).

[211] Krishnan Srinivasan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn. "Learning to be safe: Deep rl with a safety critic". In: *arXiv preprint arXiv:2010.14603* (2020).

[212] Hanping Zhang and Yuhong Guo. "Safety Modulation: Enhancing Safety in Reinforcement Learning through Cost-Modulated Rewards". In: *arXiv preprint arXiv:2504.03040* (2025).

[213] Homanga Bharadhwaj, Aviral Kumar, Nicholas Rhinehart, Sergey Levine, Florian Shkurti, and Animesh Garg. "Conservative Safety Critics for Exploration". In: *International Conference on Learning Representations*. 2020.

[214] Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. "Recovery rl: Safe reinforcement learning with learned recovery zones". In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 4915–4922.

[215] W. Hahn. *Theory and Application of Liapunov's Direct Method*. Dover Books on Mathematics. Dover Publications, 2019. ISBN: 9780486833606. URL: `https://books.google.co.uk/books?id=9wNBvQEACAAJ`.

[216] Haritz Odriozola-Olalde, Maider Zamalloa, and Nestor Arana-Arexolaleiba. "Shielded Reinforcement Learning: A review of reactive methods for safe learning". In: *2023 IEEE/SICE International Symposium on System Integration (SII)*. IEEE. 2023, pp. 1–8.

[217] Bettina Könighofer, Florian Lorber, Nils Jansen, and Roderick Bloem. "Shield synthesis for reinforcement learning". In: *Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles: 9th International Symposium on Leveraging Applications of Formal Methods, ISoLA 2020, Rhodes, Greece, October 20–30, 2020, Proceedings, Part I 9*. Springer. 2020, pp. 290–306.

[218] Amir Pnueli. "The temporal logic of programs". In: *18th annual symposium on foundations of computer science (sfcs 1977)*. ieee. 1977, pp. 46–57.

[219] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. "A survey of deep learning techniques for autonomous driving". In: *Journal of field robotics* 37.3 (2020), pp. 362–386.

[220] Sarah Dean, Andrew Taylor, Ryan Cosner, Benjamin Recht, and Aaron Ames. "Guaranteeing safety of learned perception modules via measurement-robust control barrier functions". In: *Conference on Robot Learning*. PMLR. 2021, pp. 654–670.

[221] Sheng Li. *[CS231N] Semantic Segmentation in the Traffic Environment using the DeepLabv3+ Model*. https://sheng-li.me/cs-231n-semantic-segmentation-in-the-traffic-environment-using-the-deeplabv3-model/. Accessed: 2025-10-06. 2018.

[222] Yarden As, Bhavya Sukhija, Lenart Treven, Carmelo Sferrazza, Stelian Coros, and Andreas Krause. "Actsafe: Active exploration with safety constraints for reinforcement learning". In: *arXiv preprint arXiv:2410.09486* (2024).

[223] Sam Garlick and Andrew Bradley. "Real-time optimal trajectory planning for autonomous vehicles and lap time simulation using machine learning". In: *Vehicle System Dynamics* 60.12 (2022), pp. 4269–4289.

[224] Janis Postels, Hermann Blum, Yannick Strümpler, Cesar Cadena, Roland Siegwart, Luc Van Gool, and Federico Tombari. "The hidden uncertainty in a neural networks activations". In: *arXiv preprint arXiv:2012.03082* (2020).

[225] Geoffrey Grimmett and David Stirzaker. *Probability and random processes*. Oxford university press, 2020.

[226] Sang Gyu Kwak and Jong Hae Kim. "Central limit theorem: the cornerstone of modern statistics". In: *Korean journal of anesthesiology* 70.2 (2017), p. 144.

[227] Heinz Bauer. *Probability theory*. Vol. 23. Walter de Gruyter, 1996.

[228] Guohao Shen, Yuling Jiao, Yuanyuan Lin, Joel L Horowitz, and Jian Huang. "Deep quantile regression: Mitigating the curse of dimensionality through composition". In: *arXiv preprint arXiv:2107.04907* (2021).

[229] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[230] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. 2017. arXiv: 1708.07747 [cs.LG].

[231] A. Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". PhD thesis. University of Tront, 2009. URL: https://www.semanticscholar.org/paper/Learning-Multiple-Layers-of-Features-from-Tiny-Krizhevsky/5d90f06bb70a0a3dced62413346235c02b1a

[232] Kai Brach, Beate Sick, and Oliver Dürr. "Single-shot Bayesian approximation for neural networks". In: *arXiv preprint arXiv:2308.12785* (2023).

[233] RA Dorfman. "A note on the! d-method for finding variance formulae." In: *Biometric Bulletin* (1938).

[234] Yann Le and Xuan Yang. "Tiny imagenet visual recognition challenge". In: *CS 231N* 7.7 (2015), p. 3.

[235] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[236] David Harrison Jr and Daniel L Rubinfeld. "Hedonic housing prices and the demand for clean air". In: *Journal of environmental economics and management* 5.1 (1978), pp. 81–102.

[237] I-C Yeh. "Modeling of strength of high-performance concrete using artificial neural networks". In: *Cement and Concrete research* 28.12 (1998), pp. 1797–1808.

[238] Warwick J Nash, Tracy L Sellers, Simon R Talbot, Andrew J Cawthorn, and Wes B Ford. "The population biology of abalone (haliotis species) in tasmania. i. blacklip abalone (h. rubra) from the north coast and islands of bass strait". In: *Sea Fisheries Division, Technical Report* 48 (1994), p411.

[239] Ning Qian and Terrence J Sejnowski. "Predicting the secondary structure of globular proteins using neural network models". In: *Journal of molecular biology* 202.4 (1988), pp. 865–884.

[240] Krisztian Buza. "Feedback prediction for blogs". In: *Data analysis, machine learning and knowledge discovery*. Springer, 2013, pp. 145–152.

[241] Vladimir Vovk, David Lindsay, Ilia Nouretdinov, and Alex Gammerman. "Mondrian confidence machine". In: *Technical Report* (2003).

[242] Yuanjie Shi, Subhankar Ghosh, Taha Belkhouja, Jana Doppa, and Yan Yan. "Conformal prediction for class-wise coverage via augmented label rank calibration". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 132133–132178.

[243] Shuqi Liu, Jianguo Huang, and Luke Ong. "Conformal Prediction Meets Long-tail Classification". In: *arXiv preprint arXiv:2508.11345* (2025).

[244] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding". In: *arXiv preprint arXiv:1511.02680* (2015).

[245] Daniel Flögel, Marcos Gómez Villafane, Joshua Ransiek, and Sören Hohmann. "Disentangling Uncertainty for Safe Social Navigation using Deep Reinforcement Learning". In: *arXiv preprint arXiv:2409.10655* (2024).

[246] Jacopo Banfi, Lindsey Woo, and Mark Campbell. "Is it worth to reason about uncertainty in occupancy grid maps during path planning?" In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 11102–11108.

[247] Luca Mossina, Joseba Dalmau, and Léo Andéol. "Conformal semantic image segmentation: Post-hoc quantification of predictive uncertainty". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 3574–3584.

[248] Léo Andéol, Thomas Fel, Florence De Grancey, and Luca Mossina. "Confident object detection via conformal prediction and conformal risk control: an application to railway signaling". In: *Conformal and Probabilistic Prediction with Applications*. PMLR. 2023, pp. 36–55.

[249] Alexander Timans, Christoph-Nikolas Straehle, Kaspar Sakmann, and Eric Nalisnick. "Adaptive bounding box uncertainties via two-step conformal prediction". In: *European Conference on Computer Vision*. Springer. 2024, pp. 363–398.

[250] Jordan Lekeufack, Anastasios N Angelopoulos, Andrea Bajcsy, Michael I Jordan, and Jitendra Malik. "Conformal decision theory: Safe autonomous decisions from imperfect predictions". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, pp. 11668–11675.

[251] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[252] Ari Silburt, Mohamad Ali-Dib, Chenchong Zhu, Alan Jackson, Diana Valencia, Yevgeni Kissin, Daniel Tamayo, and Kristen Menou. "Lunar crater identification via deep learning". In: *Icarus* 317 (2019), pp. 27–38.

[253] Firat Erdem, Bulent Bayram, Tolga Bakirman, Onur Can Bayrak, and Burak Akpinar. "An ensemble deep learning based shoreline segmentation approach (WaterNet) from Landsat 8 OLI images". In: *Advances in Space Research* 67.3 (2021), pp. 964–974.

[254] Lena Downes, Ted J Steiner, and Jonathan P How. "Deep learning crater detection for lunar terrain relative navigation". In: *AIAA SciTech 2020 Forum*. 2020, p. 1838.

[255] Hamed Karimi and Reza Samavi. "Quantifying deep learning model uncertainty in conformal prediction". In: *Proceedings of the AAAI Symposium Series*. Vol. 1. 1. 2023, pp. 142–148.

[256] Aleksandar Botev, Hippolyt Ritter, and David Barber. "Practical Gauss-Newton optimisation for deep learning". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 557–565.

[257] Jiaming Ji, Borong Zhang, Jiayi Zhou, Xuehai Pan, Weidong Huang, Ruiyang Sun, Yiran Geng, Yifan Zhong, Josef Dai, and Yaodong Yang. "Safety gymnasium: A unified safe reinforcement learning benchmark". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 18964–18993.

[258] Rodrigo de Lazcano, Kallinteris Andreas, Jun Jet Tai, Seungjae Ryan Lee, and Jordan Terry. *Gymnasium Robotics*. Version 1.4.0. 2024. URL: `http://github.com/Farama-Foundation/Gymnasium-Robotics`.

[259] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. "Gymnasium: A standard interface for reinforcement learning environments". In: *arXiv preprint arXiv:2407.17032* (2024).

[260] Vivek S Borkar. "An actor-critic algorithm for constrained Markov decision processes". In: *Systems & control letters* 54.3 (2005), pp. 207–213.

[261] Craig Boutilier, Thomas Dean, and Steve Hanks. "Decision-theoretic planning: Structural assumptions and computational leverage". In: *Journal of Artificial Intelligence Research* 11 (1999), pp. 1–94.

[262] Degang Wang, Lina Zhuang, Lianru Gao, Xu Sun, Xiaobin Zhao, and Antonio Plaza. "Sliding dual-window-inspired reconstruction network for hyperspectral anomaly detection". In: *IEEE Transactions on Geoscience and Remote Sensing* 62 (2024), pp. 1–15.

[263] Bing Tu, Nanying Li, Zhuolang Liao, Xianfeng Ou, and Guoyun Zhang. "Hyperspectral anomaly detection via spatial density background purification". In: *Remote Sensing* 11.22 (2019), p. 2618.

[264] Shashank Reddy Chirra, Pradeep Varakantham, and Praveen Paruchuri. "Safety through feedback in Constrained RL". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 139938–139967.

[265] Zayd Hammoudeh and Daniel Lowd. "Training data influence analysis and estimation: A survey". In: *Machine Learning* 113.5 (2024), pp. 2351–2403.

[266] Aurick Zhou and Sergey Levine. "Amortized conditional normalized maximum likelihood: Reliable out of distribution uncertainty estimation". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 12803–12812.

[267] Steven Carr, Nils Jansen, Sebastian Junges, and Ufuk Topcu. "Safe reinforcement learning via shielding under partial observability". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 37. 12. 2023, pp. 14748–14756.

[268] Shangding Gu, Bilgehan Sel, Yuhao Ding, Lu Wang, Qingwei Lin, Alois Knoll, and Ming Jin. "Safe and balanced: A framework for constrained multi-objective reinforcement learning". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2025).

[269] Greg Anderson, Abhinav Verma, Isil Dillig, and Swarat Chaudhuri. "Neurosymbolic reinforcement learning with formally verified exploration". In: *Advances in neural information processing systems* 33 (2020), pp. 6172–6183.

[270] D. Joyce. *Cauchy's Inequality*. Math 130 Linear Algebra, Clark University. Accessed 2025-10-09. 2015. URL: `http://math.clarku.edu/~ma130/`.

[271] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. "Stable-baselines3: Reliable reinforcement learning implementations". In: *Journal of machine learning research* 22.268 (2021), pp. 1–8.

# Appendix A

# Appendix for Chapter 3

## A.1 Cauchy–Schwarz Derivation

**Lemma A.1.** *Let $\hat{y}_n \in \mathbb{R}^D$ be the network output under dropout on the n-th forward pass, and the second-moment $\mathbb{E}||\hat{y}_1||^2 < \infty$ is finite. For every pair of coordinates $i, j \in \{1, \ldots, D\}$, the entry $(\hat{y}_n \hat{y}_n^\top)_{ij} = \hat{y}_{n,i}\hat{y}_{n,j}$ has a finite expectation; in particular $\mathbb{E}|\hat{y}_{1,i}\hat{y}_{1,j}| < \infty$. Hence the second-moment matrix $\mathbb{E}[\hat{y}_1\hat{y}_1^\top]$ exists entry-wise and is finite. Defining $\mu := \mathbb{E}[\hat{y}_1]$, it follows that the covariance:*

$$\mathbb{C} := Cov(\hat{y}_1) = \mathbb{E}[(\hat{y}_1 - \mu)(\hat{y}_1 - \mu)^\top] = \mathbb{E}[\hat{y}_1\hat{y}_1^\top] - \mu\mu^\top \tag{A.1}$$

*is well-defined and finite (entry-wise).*

*Proof.* Since $||\hat{y}_1||^2 = \sum_{i=1}^D \hat{y}_{1,i}^2$ with non-negative summands, $\mathbb{E}||\hat{y}_1||^2 < \infty$ implies $\mathbb{E}[\hat{y}_{1,i}^2] < \infty$ for each $i$. With fixed $i, j$, the Cauchy-Schwarz [270] inequality is:

$$\mathbb{E}[|\hat{y}_{1,i}\ \hat{y}_{1,j}|] \leqslant (\mathbb{E}[\hat{y}_{1,i}^2])^{\frac{1}{2}}(\mathbb{E}[\hat{y}_{1,j}^2])^{\frac{1}{2}} \leqslant \infty \tag{A.2}$$

so every entry $(\hat{y}_1\hat{y}_1^\top)_{ij}$ has a finite expectation and hence $\mathbb{E}[\hat{y}_1\hat{y}_1^\top]$ exists with finite entries. Given the finiteness of $\mu$ established in Theorem 3.1, expanding the covariance definition gives:

$$\mathbb{E}[(\hat{y}_1 - \mu)(\hat{y}_1 - \mu)^\top] = \mathbb{E}[\hat{y}_1\hat{y}_1^\top] - \mathbb{E}[\hat{y}_1]\mu^\top - \mu\mathbb{E}[\hat{y}_1]^\top + \mu\mu^\top = \mathbb{E}[\hat{y}_1\hat{y}_1^\top] - \mu\mu^\top \tag{A.3}$$

and both terms on the right are finite entry-wise. Therefore, $\mathbb{C}$ is well-defined and finite. □

## A.2 Mondrian Calibration Results

As demonstrated in Section 3.3.4, the marginal validity guarantee underlying standard CP doesn't ensure uniform or class-wise coverage, particularly under data imbalance in the calibration dataset. Even though MC-CP exhibits strong empirical robustness, achieving target coverage across most classes, its validity remains aggregate in nature.

To address this limitation, a seemingly natural extension is to adopt the Mondrian Conformal Prediction [241] quantile construction, which partitions the calibration data into disjoint subsets

according to a chosen conditioning variable: for instance, the class label (e.g. Truck in CIFAR-10), a higher-level semantic category (e.g. animals or vehicles in CIFAR-10), or any other discrete attribute relevant to the prediction task (e.g. lighting conditions of the image). Then, a separate threshold $\tau$ is computed for each subset, and the corresponding nonconformity scores are only attributed to that subset's quantile. Importantly, when the conditioning variable is the task class labels, Mondrian CP can guarantee class-conditional coverage, ensuring that each class individually attains the target miscoverage $1-\alpha$. In principle, this modification can enable MC-CP to mitigate the imbalance effects observed in Section 3.3.4 by simply altering the threshold construction.

However, the guarantee given by Mondrian CP comes with additional assumptions compared to Split CP. Specifically, it additionally requires that the calibration data within each partition be independently and identically distributed and that each class subset contain a sufficient number of samples to yield a stable quantile estimate. These conditions are often difficult to satisfy in practice, especially in highly imbalanced regimes where minority classes may contain few calibration points. These conditions can be difficult to satisfy in practice, especially in highly imbalanced regimes where minority classes may contain few calibration points, which we show in this section.

Formally, we recall $\mathcal{D}_{\text{cal}} = \{(x_i, y_i)\}_{i=1}^{|\mathcal{D}_{\text{cal}}|}$ is the calibration dataset, and introduce a conditioning variable $\mathfrak{z} = g(x, y) \in \mathfrak{Z}$ that partitions the calibration data into disjoint strata $\{\mathcal{D}_{\text{cal}_\mathfrak{z}}\}_{\mathfrak{z}\in\mathfrak{Z}}$, where $\mathcal{D}_{\text{cal}_\mathfrak{z}} = \{i : g(x_i, y_i) = \mathfrak{z}\}$. For each stratum $\mathfrak{z}$, we compute a stratum-specific quantile from the corresponding calibration scores $\mathcal{K}_\mathfrak{z} = \{\kappa_i : i \in \mathcal{D}_{\text{cal}_\mathfrak{z}}\}$, defined as:

$$\tau_{\alpha_\mathfrak{z}} \leftarrow \text{ the } \lceil (1-\alpha)(1+|\mathcal{D}_{\text{cal}_\mathfrak{z}}|)\rceil \text{ largest value in } \{\kappa_{\mathfrak{z},i}\}_{i=1}^{|\mathcal{D}_{\text{cal}_\mathfrak{z}}|} \tag{A.4}$$

Prediction sets are then formed conditionally, by comparing the nonconformity score of each candidate label against the quantile corresponding to the stratum to which that label belongs. In the class-conditional case, this means that each class is evaluated using its own calibration quantile, ensuring that inclusion in the prediction set depends on class-specific calibration statistics rather than a single global threshold. More generally, the conditioning variable can represent any meaningful partition of the data; for example, grouping classes into broader semantic categories such as animals and vehicles. In this case, all animal classes (e.g., dog, cat, bird, frog) would share a common quantile derived from their collective calibration scores, while vehicle classes (e.g., car, truck, airplane, ship) would be calibrated under a separate quantile. During inference, candidate labels are therefore compared against the quantile of their respective group, ensuring that coverage is enforced within each semantic partition rather than uniformly across the entire dataset. The prediction set construction is formally defined as:

$$\begin{aligned} \text{rank} &\leftarrow |\{\xi \in \Xi : \sum_{i=1}^{\xi} \hat{y}_i^{\text{sorted}} + \beta(\xi - N_{\text{reg}})_+ \leqslant \tau_{\alpha_\mathfrak{z}}\}| + 1 \quad \text{s.t.} \quad \xi \in \mathfrak{z} \\ \Gamma &\leftarrow \{\xi \in \Xi | \xi \leqslant \text{rank}\} \end{aligned} \tag{A.5}$$

Table A.1: Average prediction set accuracy per class for MC-CP using various quantile construction methods on the CIFAR-10 dataset under an extremely imbalanced calibration dataset. Highlighting denotes class-wise miscoverage

| Technique | Airplane | Automobile | Bird | Cat | Deer | Dog | Frog | Horse | Ship | Truck | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|
| MC-CP | $96.20 \pm 0.28$ | $97.11 \pm 0.25$ | $97.33 \pm 0.26$ | $96.67 \pm 0.29$ | $99.40 \pm 0.18$ | $96.90 \pm 0.27$ | $98.90 \pm 0.20$ | $95.73 \pm 0.31$ | $96.67 \pm 0.29$ | $94.90 \pm 0.36$ | $96.78 \pm 0.27$ |
| MC-CP$^{\text{perclass}}$ | $81.24 \pm 0.87$ | $91.23 \pm 0.14$ | $76.61 \pm 0.68$ | $99.63 \pm 0.24$ | $64.25 \pm 0.72$ | $98.94 \pm 0.82$ | $97.35 \pm 0.61$ | $82.52 \pm 0.19$ | $85.18 \pm 0.69$ | $72.38 \pm 0.83$ | $84.93 \pm 0.38$ |
| MC-CP$^{\text{group}}$ | $86.80 \pm 1.05$ | $92.09 \pm 0.68$ | $97.15 \pm 0.26$ | $97.56 \pm 0.97$ | $99.43 \pm 0.72$ | $96.78 \pm 0.66$ | $98.71 \pm 0.29$ | $96.16 \pm 0.95$ | $93.80 \pm 0.68$ | $85.51 \pm 1.07$ | $94.44 \pm 0.90$ |

Table A.2: Average prediction set size per class for MC-CP using various quantile construction methods on the CIFAR-10 dataset under an extremely imbalanced calibration dataset.

| Technique | Airplane | Automobile | Bird | Cat | Deer | Dog | Frog | Horse | Ship | Truck | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|
| MC-CP | $4.67 \pm 1.60$ | $4.41 \pm 1.84$ | $4.79 \pm 1.29$ | $5.07 \pm 1.27$ | $4.61 \pm 1.31$ | $4.84 \pm 1.24$ | $3.63 \pm 1.54$ | $4.47 \pm 1.69$ | $4.20 \pm 1.80$ | $4.88 \pm 1.71$ | $4.56 \pm 1.59$ |
| MC-CP$^{\text{perclass}}$ | $3.09 \pm 1.58$ | $2.48 \pm 1.57$ | $3.62 \pm 1.11$ | $3.31 \pm 0.96$ | $3.70 \pm 1.19$ | $3.12 \pm 0.99$ | $2.69 \pm 1.03$ | $3.40 \pm 1.31$ | $2.46 \pm 1.57$ | $2.99 \pm 1.64$ | $3.09 \pm 1.39$ |
| MC-CP$^{\text{group}}$ | $4.54 \pm 1.77$ | $3.79 \pm 2.17$ | $4.77 \pm 1.16$ | $5.04 \pm 1.02$ | $4.62 \pm 1.10$ | $4.93 \pm 1.01$ | $3.38 \pm 1.61$ | $4.67 \pm 1.47$ | $3.57 \pm 2.11$ | $4.68 \pm 1.94$ | $4.40 \pm 1.69$ |

Under the exchangeability assumption within each stratum and with sufficiently large calibration samples for each stratum $|\mathcal{D}_{\text{cal}_\mathfrak{z}}|$, this Mondrian MC-CP calibration guarantees class-conditional coverage:

$$\mathbb{P}\{Y \in \Gamma | X, g(X,Y) = \mathfrak{z}\} \geqslant 1 - \alpha \tag{A.6}$$

for all $\mathfrak{z} \in \mathfrak{Z}$.

Next, we empirically evaluate two variants of MC-CP that incorporate the Mondrian calibration procedure. The first, denoted MC-CP$^{\text{perclass}}$, uses the true class label as the conditioning variable, thereby enforcing marginal validity within each class. The second, denoted MC-CP$^{\text{group}}$, generalises this idea to structured groupings of classes in the CIFAR-10 dataset: the dominant class Frog, a vehicle group comprising Truck and related classes, and an animal group containing all remaining non-frog classes. These experiments directly mirror those presented in Section 3.3.4 on dataset imbalance, but employ Mondrian calibration to investigate whether stratified quantile estimation can recover balanced class-wise coverage while preserving predictive efficiency. Results are summarised in Table A.1 and Table A.2, respectively.

Under the same extreme imbalance ($|\mathcal{D}_{\text{cal}}| = 250$ with $\approx 90\%$ of the labels being frog and the remaining evenly shared across the remaining classes), incorporating Mondrian calibration into MC-CP does not recover reliable class-wise coverage, and in both variants, worsens results. When the conditioning variable is the class-labels $\mathfrak{z} = y$, each minority class has only $\approx 2-3$ calibration samples; evidently, the per-class quantiles are corrupted by small-sample noise. This results in overconfident, small prediction sets and a failure to achieve class-wise coverage. For example, the average set size reduces from $4.56 \pm 1.59$ to $3.09 \pm 1.39$ compared to standard MC-CP, and as a result, the average accuracy reduces to $84.93 \pm 0.38$. Grouping the classes by semantics lifts the calibration count to $\approx 12 - 15$ for all groups except for the Frog group, and thus, partially stabilises the quantiles. The average set size increases to $4.40 \pm 1.69$, and several minority classes' accuracy increases to $> 90\%$ but still fail to achieve class-wise coverage. Both variants actually underperform compared to the standard MC-CP. Overall, when the calibration dataset is small and severely imbalanced, Mondrian calibration yields poorly calibrated thresholds and

**Algorithm 8** KDE-based Adaptive Monte Carlo Dropout

**Require:** Model $\mathcal{F}_\theta$, Input $x$, Maximum forward passes $N_{\text{mc}}$, KDE threshold $\delta$, Patience $\rho_{\max}$, Grid $\{t_g\}_{g=1}^{G}$, Bandwidth $h$, Kernel $K_h$

$\rho \leftarrow 0$; $\mathcal{P} \leftarrow \varnothing$; prev_peaks $\leftarrow$ None
Initialise cumulative kernel sums $C \leftarrow 0_{\Xi \times G}$
**for** $n \in \{1, \ldots, N_{\text{mc}}\}$ **do**
  $\hat{y} \leftarrow \mathcal{F}_{\theta_n}(x)$
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{\hat{y}\}$
  $z \leftarrow \text{vec}(\hat{y}) \in [0,1]^{\Xi}$
  $C \leftarrow C + K_h(t_g - z) \qquad (\forall g = 1, \ldots, G)$
  peaks $\leftarrow \max_g \frac{C}{n}$
  $\rho \leftarrow \begin{cases} \rho + 1, & \text{if prev_peaks} \neq \text{None} \wedge \max_j |\text{peaks}_j - \text{prev_peaks}_j| \leqslant \delta, \\ 0, & \text{otherwise} \end{cases}$
  **if** $\rho \geqslant \rho_{\max}$ **then**
    terminate loop
  **end if**
  prev_peaks $\leftarrow$ peaks
**end for**
**return** $\mathcal{P}$

is therefore ineffective for achieving class-wise coverage.

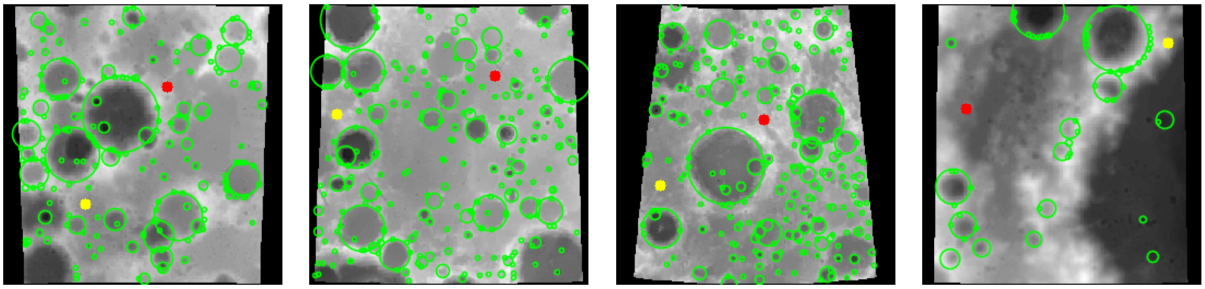
## A.3 KDE-based Adaptive MC Dropout Algorithm

In Section 3.3.7, we introduced a KDE-based variant of the Adaptive MC Dropout Algorithm. Algorithm 8 implements this, in which the stochastic forward passes are stopped online using a kernel-density stabilisation criterion. After each dropout sample $\hat{y} = \mathcal{F}_{\theta_n}(x)$, we update a discretised KDE over a fixed grid $\{t_g\}_{g=1}^{G}$ for each output dimension, maintaining cumulative kernel sums so that the estimated density can be updated in $\mathcal{O}(\Xi G)$ per sample without recomputing all pairwise contributions. Let peaks $\in \mathbb{R}^{\Xi}$ denote the per-dimension KDE peak statistic computed from the current density estimate. Sampling terminates early once the peak statistic changes by at most $\delta$ for $\rho_{\max}$ consecutive iterations, indicating that additional Monte Carlo samples are unlikely to change the estimated predictive distribution. In the worst case, the method reduces to standard MC Dropout with $N_{\text{mc}}$ forward passes.

## A.4 COPPOL Environment Examples

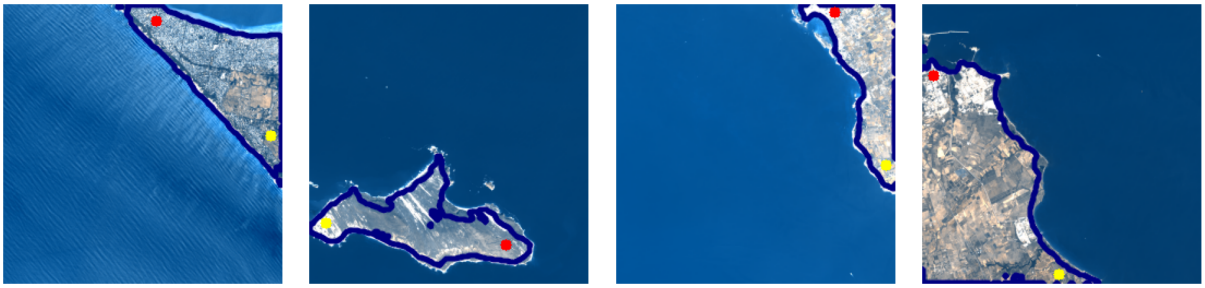
This section provides qualitative visualisations of the eight environments used to evaluate COPPOL, comprising four environments per dataset, together with representative agent trajectories.

Figure A.1 shows the four environments for each dataset. The DeepMoon environments vary in the size and arrangement of detected craters:



Environment 1 Environment 2 Environment 3 Environment 4

(a) DeepMoon Crater Detection



Environment 1 Environment 2 Environment 3 Environment 4

(b) YTU-WaterNet Water Detection

Figure A.1: Qualitative visualisations of the four environments used to evaluate COPPOL in (a) DeepMoon Crater and (b) YTU-WaterNet Water hazards. Hazard Detection is constructed via the MC-CP variant. Yellow is the starting point, red is the goal point, green circles are the detected craters, and blue is the detected edge of a body of water.

**DeepMoon 1:** A large central crater and neighbouring craters obstruct direct progress from the lower start position towards the upper-right goal, requiring a route around the central cluster.

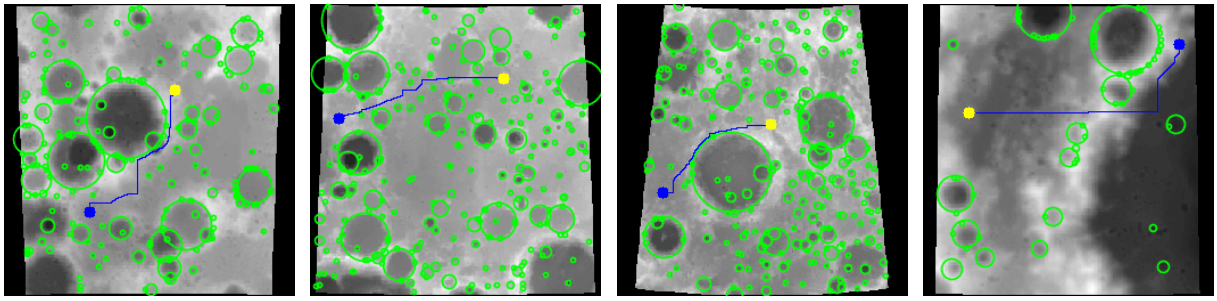
**DeepMoon 2:** Numerous small and medium-sized craters are distributed between the left-hand start and upper-right goal, testing navigation through successive gaps between hazards.

**DeepMoon 3:** A large crater lies directly between the lower-left start and the goal near the centre, requiring a detour while avoiding the surrounding smaller craters.

**DeepMoon 4:** Craters are more sparsely distributed, but a large crater beside the upper-right start obstructs direct progress towards the left-hand goal, testing local hazard avoidance before entry into more open terrain.

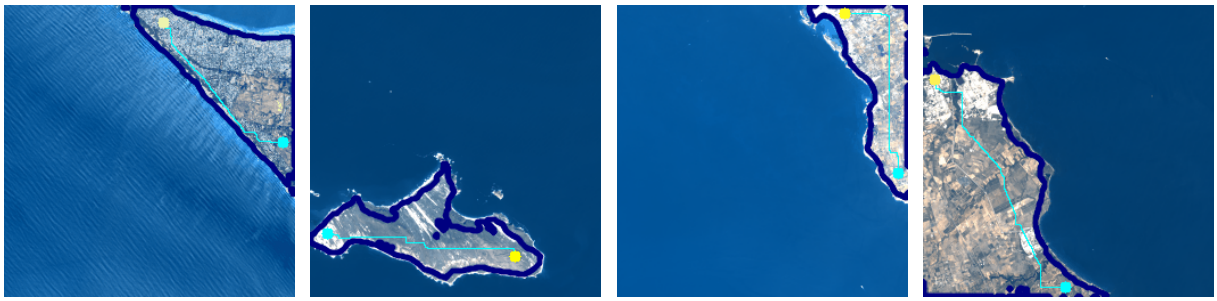
Together, these layouts test navigation around large isolated obstacles and through more densely distributed hazards, allowing assessment of how calibrated crater detections influence route selection and clearance across different spatial configurations. The YTU-WaterNet environments instead test navigation within land regions bounded by different shoreline geometries:

**Water 1:** The start and goal lie within a tapering coastal land region bounded by water on multiple sides, testing whether the agent maintains clearance while progressing along the



Environment 1 Environment 2 Environment 3 Environment 4

(a) DeepMoon Crater Detection



Environment 1 Environment 2 Environment 3 Environment 4

(b) YTU-WaterNet Water Detection

Figure A.2: Qualitative visualisations of an example COPPOL (using MC-CP) trajectory in the four environments used to evaluate COPPOL in (a) DeepMoon Crater and (b) YTU-WaterNet Water hazards. Hazard Detection is constructed via the MC-CP variant. Yellow is the starting point, purple and cyan are the agent's trajectories, respectively, green circles are the detected craters, and blue is the detected edge of a body of water.

land.

**Water 2:** The start and goal occupy opposite ends of an elongated island with an irregular shoreline, requiring traversal through narrower sections while avoiding coastal indentations.

**Water 3:** A narrow coastal strip follows the right-hand image boundary, with a curved shoreline between the lower start and upper goal, limiting the space available for maintaining clearance from water.

**Water 4:** A broader land region lies to the left of an irregular coastline, allowing inland detours between the lower start and upper-left goal and testing whether the agent uses this space to maintain hazard clearance.

Together, these layouts vary the shape and width of navigable land and the space available for inland detours, allowing assessment of how calibrated water boundaries influence hazard avoidance under different routing constraints.

Figure A.2 complements these environment snapshots by overlaying an example agent trajectory for each environment (using the MC-CP variant of COPPOL), illustrating the behavioural effect

of the hazard detections on planning and execution. The plotted paths demonstrate how the agent adapts its route relative to the predicted hazardous regions: in DeepMoon, trajectories tend to arc around dense crater clusters or thread through locally safer gaps between detections; in YTU-WaterNet, trajectories typically track along the coastline while maintaining separation from the water boundary.

## A.5 MC-CP Experimental Setup

This section details the experimental setup used to evaluate MC-CP in Section 3.3. All experiments share a common experimental backbone, which is described first, followed by experiment-specific subsections that document any deviations or additional configuration details required for individual evaluations.

### A.5.1 Common Experimental Setup

This subsection describes the experimental configuration shared across all evaluations of MC-CP in Section 3.3. Unless otherwise stated, all experiments follow this setup exactly.

#### A.5.1.1 Datasets

MC-CP is evaluated on a diverse set of standard benchmarks covering both image classification and tabular regression tasks.

For classification, we consider five widely used image datasets: MNIST [229], Fashion-MNIST [230], CIFAR-10 and CIFAR-100 [231], and Tiny-ImageNet [234]. MNIST and Fashion-MNIST each consist of $60,000$ grey-scale images of size $28 \times 28$ across 10 classes. CIFAR-10 and CIFAR-100 contain $60,000$ colour images of size $32 \times 32$, with 10 and 100 classes respectively. Tiny-ImageNet is a reduced-scale variant of ImageNet comprising $100,000$colour images of size $64 \times 64$ spanning 200 classes. These datasets provide increasing levels of visual complexity and class granularity, allowing MC-CP to be evaluated across a broad range of classification regimes.

For regression, we evaluate MC-CP on five standard tabular benchmarks: Boston Housing [236], Concrete Compressive Strength [237], Abalone [238], Physicochemical Properties of Protein Tertiary Structure [239], and Blog Feedback [240]. The Boston Housing dataset contains 506 samples with 14 attributes, Abalone consists of $4,180$ samples with 9 attributes, Concrete contains $1,030$ samples with 9 attributes, the Protein dataset includes $45,730$ samples with 9 attributes, and Blog Feedback comprises $60,021$ samples with 281 attributes. Together, these datasets span a wide range of sample sizes, dimensionalities, and noise characteristics.

For datasets with established train–test splits available in prior work or common benchmark implementations, we follow these splits directly. In all cases, split CP is implemented by holding out a dedicated calibration set disjoint from training and testing. For classification datasets without a standard calibration split, $2,500$ samples are reserved from the test set for calibration, with the remaining test samples used for evaluation. For regression datasets, where test sets are typically smaller, half of the test data is used for calibration and the remaining half

for evaluation. This procedure ensures strict adherence to the requirements of split CP while maintaining comparability with prior work.

#### A.5.1.2 Models and Training

For all experiments, MC-CP is applied post hoc to trained predictive models, which are treated as fixed black-box predictors during conformal calibration and evaluation. Separate base models are used for classification and regression tasks, reflecting standard practice for each setting.

For classification, we employ a convolutional neural network (CNN) architecture commonly used in image classification benchmarks. The network consists of two convolutional blocks, each comprising a $3 \times 3$ convolutional layer followed by max-pooling and dropout, and a fully connected classification head. Specifically, the first convolutional layer uses 32 filters and the second uses 64 filters, both followed by $2 \times 2$ max-pooling. Dropout with a rate 0.5 is applied after each pooling layer to induce stochasticity for Monte Carlo sampling. The convolutional feature extractor is followed by a dense layer with 128 hidden units and ReLU activation, and a softmax output layer producing class probabilities. Models are trained using categorical cross-entropy loss and optimised with stochastic gradient descent with momentum 0.9 and learning rate 0.01. Training is performed for 10 epochs with a batch size of 128. The model follows a lightweight LeNet-inspired architecture [229], augmented with dropout layers to enable Monte Carlo sampling at test time.

For regression, we employ a deep quantile neural network (DQNN) that directly predicts conditional quantiles of the target variable. The model consists of a sequence of fully connected layers with ReLU activations, preceded by an optional input normalisation layer adapted to the training data. Unless otherwise stated, two hidden layers with 128 units each are used. Dropout with a rate 0.1 is applied between hidden layers. The output layer comprises separate linear heads corresponding to each target quantile.

Quantile regression is performed using the pinball loss, with lower and upper quantiles $\tau_{\text{lower}} = 0.05, \tau_{\text{upper}} = 0.95$. The model is trained using the Adam optimiser with a learning rate 0.001 for 100 epochs and a batch size of 32. This formulation yields prediction intervals directly from the base model, allowing MC-CP to operate on quantile-based nonconformity scores without altering the training procedure.

Across both classification and regression settings, model parameters are learned exclusively during the training phase. No retraining, fine-tuning, or architectural modification is performed during conformal calibration or evaluation, ensuring that MC-CP remains strictly post hoc and model-agnostic.

#### A.5.1.3 MC-CP Hyperparameters

The hyperparameters reported in Table A.3 are fixed across all experiments and were selected to reflect standard, well-established practice rather than being tuned per dataset. For classification, the hyperparameters are guided by choices commonly adopted in RAPS-style prediction

Table A.3: MC-CP hyperparameters used in the experimental evaluation.

| **Parameter** | **Classification** | **Regression** |
|---|---|---|
| Target miscoverage level $\alpha$ | 0.05 | 0.10 |
| Maximum MC samples $N_{\text{mc}}$ | 1000 | 1000 |
| Threshold $\delta$ | $5 \times 10^{-4}$ | $5 \times 10^{-4}$ |
| Patience $\rho_{\max}$ | 10 | 10 |
| Regularisation $\beta$ | 0.01 | 0.01 |
| Number of unpenalized classes $N_{\text{reg}}$ | 5 | 5 |
| Stochastic smoothing boolean rand | True | - |

sets [111], while the regression configuration follows conventions used in CQR [112]. The Adaptive MC Dropout parameters are set according to the parameter analysis in Section 3.3.2 and are used uniformly to balance estimator stability against computational cost.

### A.5.2 Distribution and Covariate Shift Experimental Setup

To evaluate the robustness of MC-CP under distribution and covariate shifts, we construct shifted evaluation distributions by applying label-preserving transformations to inputs.

Let $(x, y) \sim \mathcal{D}$ denote an in-distribution test example, and let $\tau_\rho : X \to X$ be a corruption operator indexed by a severity parameter $\rho$. The shifted distribution $\mathcal{D}_\rho$ is defined as:

$$x' = \tau_\rho(x), \qquad (x', y) \sim \mathcal{D}_\rho \tag{A.7}$$

so that the shift acts on $p(x)$ while preserving labels. All shifts are applied at evaluation time only; training and calibration follow the common setup in Section A.5. Covariate shifts operate without a severity parameter $\rho$.

#### A.5.2.1 Distribution Shift Setup

Stroke-style shifts require identifying a foreground (stroke) region. Two candidate masks are obtained by thresholding the greyscale image at 0.5 where pixels in the image are bounded between $[0, 1]$. Line thickness variations are implemented via binary morphology. Given a foreground mask $fm$, a morphed mask:

$$fm_\rho \in \{fm \oplus B_r,\ fm \ominus B_r\} \tag{A.8}$$

is obtained by dilation or erosion with a disk structuring element $B_r$ of radius $r = \rho$. The greyscale image is then reconstructed by hard assignment: pixels inside the morphed mask are set to the stroke extreme (1 for digit strokes and 0 for no stroke), while all others retain their original values. This produces controlled changes in stroke width while preserving digit identity.

Ink bleed simulates outward diffusion of stroke intensity, with emphasis near stroke endpoints. Endpoint seeds are identified by thinning the foreground mask and selecting pixels that have exactly one adjacent foreground pixel when considering the 8-connected neighbourhood (including diagonals); additional bleed seeds are sampled as a fixed fraction of boundary pixels. A distance-based envelope:

$$w(p) = \exp\left( - \frac{d(p, S)}{\max(1, R)} \right) \tag{A.9}$$

Here $d(p, S)$ denotes the distance from pixel $p$ to the nearest bleed seed (endpoint or boundary hotspot; identified using the 8-connected neighbours), used to control the spatial decay of the ink-bleed effect and $R = \rho$ controls the spread/bleed severity. After computing $w(p)$, bleed is applied only outside the stroke mask and only within an $R$-pixel shell around the digit (implemented via an $R$-pixel dilation of the stroke mask), so that the corruption locally expands the stroke appearance without globally modifying the background. Endpoint regions are emphasised to produce stronger *ballooning* at stroke tips, with additional mild bleed seeded by a fixed fraction of boundary hotspots. Affected pixels are then pulled toward the inferred stroke extreme (1 for digit strokes, 0 for no stroke) and clipped to $[0, 1]$; the severity parameter $\rho$ controls the spread via $R$.

#### A.5.2.2 Covariate Shift Setup

Blob occlusion overlays one or more opaque masks onto the image. Each blob is initialised as a circle with a radius sampled as a fraction of the image size; irregular blobs are formed by random additions and subtractions. Occluded pixels are overwritten with a fill value (by default, the estimated background colour). For MNIST-like images, blob centres are biased toward the foreground to ensure meaningful occlusion.

Defocus blur is modelled as a convolution with a circular point spread function:

$$x = K_r * x \tag{A.10}$$

where the kernel $K_r \in \mathbb{R}^{(2r+1)\times(2r+1)}$ is a normalised circular kernel with radius $r$ to simulate lens-induced blur while preserving global structure.

JPEG artefacts are simulated via a standard $8 \times 8$ block Discrete Cosine Transform (DCT) compression pipeline with quality-controlled quantisation. For RGB inputs, images are converted to YCbCr and optionally chroma-subsampled before compression. Lower quality factors induce stronger blocking and ringing artefacts.

Elastic distortions apply a smooth random spatial deformation: a $4 \times 4$ grid of control points is perturbed by i.i.d Gaussian noise with standard deviation 0.35 times the grid cell size, and the resulting sparse displacement field is bilinearly interpolated to a dense warp.

### A.5.3 Risk-Sensitive Experimental Setup

To evaluate MC-CP under explicit notions of risk, we augment the base nonconformity score with an additional risk term, inspired by CSC formulations that encode asymmetric costs directly into the calibration objective. Rather than treating all miscoverage events as equally undesirable, this augmentation allows the conformal predictor to prioritise specific failure modes, such as overly large prediction sets or miscoverage on safety-critical classes. As stated in Section 3.3.6, we augment the nonconformity scores $\kappa_i$ with an additional risk-modulation term. The resulting score is defined as:

$$\kappa_i \leftarrow \sum_{\xi=1}^{\text{rank}_{y_i}} \hat{y}_{\xi}^{\text{sorted}} + \beta(\text{rank}_{y_i} - N_{\text{reg}})_+ + \underbrace{\lambda_{\text{risk}}\mathcal{R}(x_i, y_i)}_{\text{Risk Calibration}} \tag{A.11}$$

where $\lambda_{\text{risk}} \geqslant 0$ is a user-chosen weighting parameter, and $\mathcal{R}(x_i, y_i)$ is a user-chosen risk function that can penalise any notion of risk, such as large prediction sets, or class-weighted costs. When $\lambda_{\text{risk}} = 0$, the score reduces to the standard MC-CP formulation. We evaluated two instantiations of $\mathcal{R}$.

To discourage overly conservative prediction sets, we define a set-size penalty that directly penalises the cardinality of the prediction set:

$$\mathcal{R}(x_i, y_i) \triangleq |\Gamma(x_i)| \tag{A.12}$$

This formulation biases calibration toward sharper prediction sets by increasing the nonconformity score for large sets, thereby tightening the empirical quantile threshold used at test time.

To model asymmetric miscoverage costs across classes, we further consider a class-weighted risk that penalises miscoverage events according to the semantic importance of the true label:

$$\mathcal{R}(x_i, y_i) \triangleq \lambda_{\text{y}}\mathbb{I}\{y_i \notin \Gamma(x_i)\} \tag{A.13}$$

where $\lambda_{\text{y}} : Y \to \mathbb{R}_{>0}$ assigns a cost to each class. In our CIFAR-10 experiments, higher costs are assigned to automobile and truck classes, lower costs to airplane, bird, and ship, and unit cost to all remaining classes, reflecting application-dependent risk preferences.

## A.6 COPPOL Experimental Setup

This section details the experimental setup used to evaluate COPPOL in Section 3.4.2. All experiments share a common experimental backbone, which is described first, followed by experiment-specific subsections that document any deviations or additional configuration details required for individual evaluations.

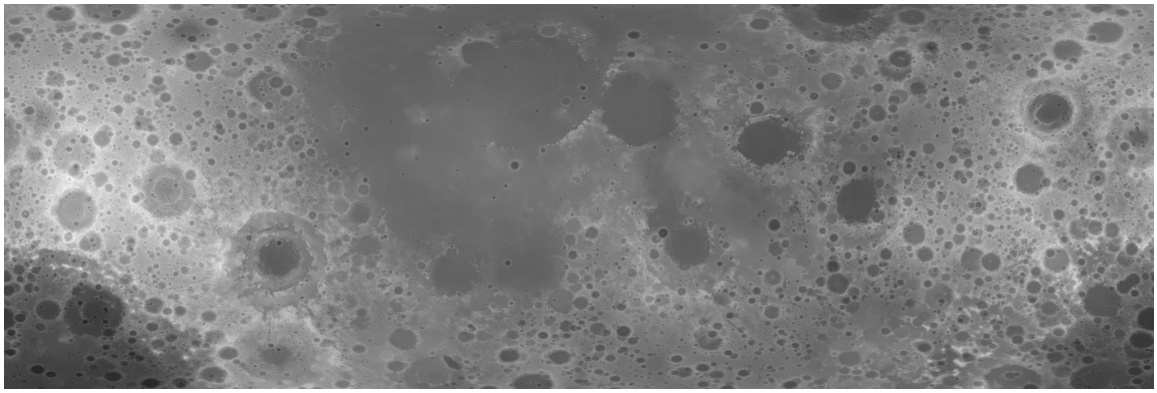


Figure A.3: The single global lunar digital elevation model used to create the DeepMoon dataset.

### A.6.1 Common Experimental Setup

This subsection describes the experimental configuration shared across all evaluations of COPPOL in Section 3.4.2. Unless otherwise stated, all experiments follow this setup exactly.

#### A.6.1.1 Datasets

COPPOL is evaluated on two satellite imaging datasets: DeepMoon crater detection [252] and the YTU-WaterNet water segmentation dataset [253]. Both tasks are framed as supervised pixel-wise prediction problems in which each input patch is paired with a binary target mask.

We follow the DeepMoon data-generation procedure [252] in which a single global lunar digital elevation model (DEM) is used as the underlying source image (seen in Figure A.3), and training examples are produced by repeatedly sampling random geographic crops and converting them into fixed-size patches. Specifically, square crops are extracted from the global DEM at random locations (uniformly distributed in position), with crop side lengths sampled log-uniformly over a wide range to ensure multiscale coverage. Each crop is then downsampled to a fixed spatial resolution (256×256), transformed into an orthographic projection to reduce distortion, and intensity-rescaled to improve contrast. Targets are generated at the same spatial resolution as binary ring masks indicating crater rims, constructed from merged human crater catalogues (with sub-pixel radii mapped into pixel space and very small craters excluded).

In our implementation, we store DeepMoon images as single-channel tensors of shape (256, 256, 1). Raw DEM values are normalised to $[0, 1]$ via division by 255. To avoid the zero-valued padded/background regions dominating the dynamic range, we apply a foreground-only linear rescaling: for each patch, let $m = \min\{x > 0\}$ and $M = \max\{x > 0\}$ over non-zero pixels; then non-zero intensities are mapped into a fixed interval $[low, hi]$ (we use $low = 0.1$, $hi = 1.0$), while zeros are left unchanged. This preserves the missing/padded semantics of zeros while making elevation contrast comparable across patches despite changing crop scale and terrain statistics. For experiments, we use a split of train $= 30,000$, cal $= 5,000$, test $= 5,000$ patches, derived from the publicly released DeepMoon corpus (the original DeepMoon construction also uses geographically disjoint splits to avoid train–test leakage across nearby longitudes).

The YTU-WaterNet dataset is introduced alongside the WaterNet ensemble segmentation study and is built from Landsat 8 OLI coastal imagery with labels produced using OpenStreetMap-derived shoreline/water information. The paper reports 63 Landsat 8 OLI full-frame scenes drawn from multiple coastal regions, which are subdivided into 1008 image subsets of size $512 \times 512$, with an 80%, 10%, 10% train/validation/test split used for model development and evaluation.

We repackage YTU-WaterNet into a DeepMoon-style HDF5 format for consistent loading across datasets. Each original subset is read as a multi-band patch with a paired binary mask. We convert the source channels into a natural-looking 3-channel composite in $[0, 1]$: the patch is first reordered into a consistent channel layout, then mapped to an RGB-like representation (red band as $R$, a synthetic green computed as a convex combination of red and the *blue proxy* channel, and the blue proxy as $B$). To stabilise appearance across scenes and acquisition conditions, we apply a global percentile stretch (2nd–98th percentile) to the composite and clip to $[0, 1]$. Each image–mask pair is then resized to $256 \times 256$ to match the DeepMoon resolution; we use area interpolation for the image and nearest-neighbour interpolation for the mask to avoid introducing mixed-label boundary artefacts.

#### A.6.1.2 Models and Training

For both datasets, we train a U-Net–style fully convolutional segmentation model that maps a fixed-size image patch to a per-pixel probability mask. The model follows an encoder–decoder design with skip connections: the encoder extracts increasingly abstract spatial features via repeated $3 \times 3$ convolutions and $2 \times 2$ max-pooling, and the decoder progressively restores spatial resolution via upsampling and feature fusion with the corresponding encoder activations. The encoder comprises three resolution levels, each containing two $3 \times 3$ convolutional layers with ReLU activations and same-padding, using 112, 224, and 448 feature channels respectively, followed by max-pooling. A bottleneck block applies two additional $3 \times 3$ convolutions at 448 channels. The decoder mirrors the encoder with three upsampling stages; at each stage, the upsampled feature map is concatenated with the corresponding encoder feature map (skip connection), followed by dropout and two further $3 \times 3$ convolutions. Dropout is applied after each concatenation with a rate 0.15. A final $1 \times 1$ convolution with sigmoid activation produces a single-channel output mask representing per-pixel foreground probabilities. The only dataset-specific adjustment is the input channel dimensionality (single-channel versus three-channel patches), while all other architectural choices are held fixed to ensure comparability.

Models are trained with the Adam optimiser (learning rate $10^{-4}$) using binary cross-entropy with sigmoid outputs. We train for up to 15 epochs with mini-batches of size 8, selecting the final model via early stopping on validation loss with patience 3. The best-performing checkpoint (lowest validation loss) is saved and used for all subsequent evaluations.

#### A.6.1.3 COPPOL Hyperparameters

The hyperparameters reported in Table A.4 are fixed across all experiments and were selected to reflect standard, well-established practice rather than being tuned per dataset. Some hyper-

Table A.4: COPPOL hyperparameters used in the experimental evaluation.

| **Parameter** | **Model** | **RL agent** |
|---|---|---|
| Target miscoverage $\alpha$ | 0.7 | – |
| Max MC samples $N_{\text{mc}}$ | 1000 | – |
| Threshold $\delta$ | $5 \times 10^{-4}$ | – |
| Patience $\rho_{\max}$ | 10 | – |
| Number of runs | – | 5 |
| Episode horizon | – | 1000 |
| Total timesteps | – | 200,000 |
| Training frequency | – | Every timestep |
| Replay buffer size | – | 200,000 |
| Batch size | – | 64 |
| Discount factor $\gamma$ | – | 0.95 |
| Gradient clipping | – | 2.5 |
| Exploration fraction | – | 0.75 |
| Final exploration $\varepsilon_{\text{final}}$ | – | 0.05 |
| Learning-rate schedule $\eta$ | – | linear: $10^{-3} \rightarrow 10^{-7}$ (end fraction 0.9) |

parameters, such as the miscoverage rate $\alpha$, were selected via manual tuning.

For the RL agent, we use the widely adopted Stable Baselines 3 [271] library to implement a Deep Q Network agent.

### A.6.2 Noisy State Perturbation

In Section 3.4.2, we introduced COPPOL and evaluated it under a controlled observation-noise setting designed to assess robustness to corrupted hazard-related state information. To this end, we constructed a noisy variant of the DeepMoon environment in which additive stochastic perturbations are applied to a subset of distance-derived state features at every timestep.

Formally, let $s_t \in \mathcal{S}$ denote the true environment state at time $t$, and let $S_{\text{noise}} \subset \{1, \ldots, |s_t|\}$ index the distance-based components of the observation, including goal distance and hazard proximity measures. The noisy observation $\tilde{s}_t$ is constructed element-wise as:

$$\tilde{s}_{t,i} = \begin{cases} \text{clip}\big(s_{t,i} + \varepsilon_{t,i},\, [\text{low}_i, \text{high}_i]\big), & \text{if } i \in S_{\text{noise}}, \\ s_{t,i}, & \text{if } i \notin S_{\text{noise}}, \end{cases} \qquad \varepsilon_{t,i} \sim \mathcal{N}(0, \sigma^2), \tag{A.14}$$

where noise is injected independently across timesteps and state dimensions, and only affects the semantically meaningful distance-derived features.

To ensure consistency between the perturbed observations and the declared observation space, we expand the corresponding bounds by $3\sigma$ prior to clipping. This guarantees that noisy ob-

servations remain valid elements of the environment's observation space and avoids introducing artificial constraint violations due solely to noise injection.

In our experimental setup, we fix $\sigma = 1.0$ and apply perturbations deterministically at every timestep ($p = 1.0$), so that all distance-based features are corrupted throughout each episode. This represents a deliberately challenging regime in which the agent must act under persistent, unbiased observation noise. Importantly, the perturbation is applied exclusively at the observation level and does not modify the underlying environment dynamics, reward function, or termination conditions, allowing the effect of state uncertainty on policy behaviour to be isolated.

# Appendix B

# Appendix for Chapter 4

## B.1 Uniform Sampling Proof

In this section, we theoretically analyse USC and show that USC is more conservative than other alternatives under certain conditions.

### B.1.1 Theorem 4.1

**Theorem 2.** The log-exp-sum (LSE) penalty, used in Equation 4.4 and standard in CSC [213], is more effective at encouraging conservatism when the $m$ alternative actions are sampled uniformly over the action space, rather than from the current policy [213]. If the policy aims to avoid high-cost regions, then the uniform sampling is more likely to draw from high-cost out-of-distribution actions. Formally, for any fixed state $s$, the conservative term is strictly larger under uniform scaling.

*Proof.* For a fixed state $s \in S$, and the action space $A$, the safety critic satisfies:

$$Lo(s) \leqslant Q_C(s,a) \leqslant Up(s) \quad \forall a \in A \tag{B.1}$$

where $Lo(\cdot)$ and $Up(\cdot)$ are the lower and upper bounds of the safety critic's output at state $s$. We let $\delta(s) \in (Lo(s), Up(s))$ be a cost threshold and:

$$\text{Risk}_{\tau(s)} := \{a \in A | Q_C(s,a) \geqslant \delta(s)\} \tag{B.2}$$

be the most costly set of actions above the cost threshold $\tau(s)$. For a distribution $P$ over actions (either $\pi(\cdot|s)$ or uniform), the probability of hitting that set is:

$$p_P(s) := \mathbb{P}_{A\sim P}(A \in \text{Risk}_{\delta(s)}) = \mathbb{P}(Q_C(s,a) \geqslant \delta(s))) \tag{B.3}$$

We assume the standard safe RL condition: the policy is less likely to select risky actions than the uniform sampler:

$$p_\pi(s) \leqslant p_{\text{unif}}(s) - \eta_s \tag{B.4}$$

for some $\eta_s > 0$. This assumption reflects the idea that the agent's current policy tends to avoid high-cost actions, whereas the uniform sampler does not. Let $a_1, \ldots, a_m \sim P$ be independent

and identically distributed $m$ sampled actions. For shared scalars, temperature $c(s) > 0$ and batch mean $\mu(s)$, the normalised critic value for each sampled action and the LSE is defined as:

$$\begin{aligned} z_i &:= c(s)(Q_C(s, a_i) - \mu(s)) \\ \mathrm{LSE}_m^{(P,c)}(s) &:= log \sum_{i=1}^{m} e^{z_i} \end{aligned} \tag{B.5}$$

We define the conservative margin at state $s$ under sampler $P$ as:

$$\Delta_p^{(c)}(s) := \mathrm{LSE}_m^{(P,c)}(s) - c(s)(Q_C(s, a_{\text{real}}) - \mu(s)) \tag{B.6}$$

where $a_{\text{real}}$ is the ground truth action from the data. This gives us the gap between the sampled actions and the ground truth under any sampler $P$. Since the data (second term) term is identical under both samplers, $P \in \{\pi(\cdot|s), \mathrm{Unif}\}$, the difference $\Delta_{\mathrm{Unif}}^{(c)} - \Delta_\pi^{(c)}$ reduces to comparing the two LSE terms.

To perform this comparison, we recall that LSE is a smooth version of the maximum: it always lies between the hard max and the hard max plus $\log m$. Evidently, if we can show the expected hard max of $Q_C$ values is larger under uniform sampling, then the same must hold for the LSE- up to $\log m$.

As stated, LSE acts like a smoothed max over $m$ inputs. This is formalised by the bound:

$$\max_i z_i \leqslant \log \sum_{i=1}^{m} e^{z_i} \leqslant \max_i z_i + \log m \tag{B.7}$$

So, this shows that LSE differs from the true max by at most $\log m$. So, if the expected max of $z_i$ is larger under uniform sampling, then the expected LSE will also be larger up to $\log m$. Applying this logic to both samplers gives:

$$\mathbb{E}[\mathrm{LSE}_m^{(\mathrm{Unif},c)}(s)] - \mathbb{E}[\mathrm{LSE}_m^{(\pi,c)}(s)] \geqslant c(s)(\mathbb{E}[\max_i Q_C(s, a_i)]_{\mathrm{Unif}} - \mathbb{E}[\max_i Q_C(s, a_i)]_\pi) - \log m \tag{B.8}$$

To show that the expected max of a uniform sampling is larger, we need to find the lower bound of $\mathbb{E}[\max_i Q_C(s, a_i)]$ for either sampler $P$. Let:

$$K_P := \sum_{i=1}^{m} \mathbf{1}\{a_i \in \mathrm{Risk}_{\delta(s)}\} \tag{B.9}$$

be a Binomial random variable counting how many of the sampled actions fall into the risky set. Thus,

$$\mathbb{P}(\max_i Q_C(s, a_i) \geqslant \delta(s)) = \mathbb{P}(K_P \geqslant 1) = 1 - (1 - p_P(s))^m \tag{B.10}$$

is the probability that at least one sample is risky. On the event that $K_P = 0$, the max is at least $Lo(s)$ as all sampled values are below $\delta(s)$, but the safety critic is still bounded by $Lo(s)$. Therefore:

$$\mathbb{E}[\max_i Q_C(s, a_i)]_P \geqslant Lo(s) + (\delta(s) - Lo(s))(1 - (1 - p_P(s))^m) \tag{B.11}$$

This shows that the expected maximum depends only on the probability of sampling a risky action. This, whichever sampler assigns more probability mass to the risky set $\text{Risk}_{\delta(s)}$ will yield the larger expected maximum. By assumption, $p_{\text{Unif}}(s) > p_\pi(s)$, so:

$$(1 - p_\pi(s))^m > (1 - p_{\text{Unif}}(s))^m \tag{B.12}$$

Therefore, the uniform sampler is more likely to include at least one risky action, leading to a larger expected maximum:

$$\mathbb{E}[\max Q_C]_{\text{Unif}} - \mathbb{E}[\max Q_C]_\pi \geqslant (\delta(s) - Lo(s))[(1 - p_\pi(s))^m - (1 - p_{\text{Unif}}(s))^m] \tag{B.13}$$

Plugging this into the LSE inequality:

$$\mathbb{E}[\text{LSE}_m^{(\text{Unif},c)}(s) - \text{LSE}_m^{(\pi,c)}(s)] \geqslant c(s)(\delta(s) - Lo(s))[(1-p_\pi(s))^m - (1-p_{\text{Unif}}(s))^m] - \log m \tag{B.14}$$

As shown earlier, because both gaps $\Delta_P^{(c)}(s)$ share the same data term, we conclude:

$$\mathbb{E}[\Delta_{\text{Unif}}^{(c)}(s) - \Delta_\pi^{(c)}(s)] \geqslant c(s)(\delta(s) - Lo(s))[(1 - p_\pi(s))^m - (1 - p_{\text{Unif}}(s))^m] - \log m \tag{B.15}$$

This shows us that, in expectation, drawing actions uniformly instead of from the policy always gives a larger conservative penalty. The margin is proportional to: how high the risky region is above the baseline ($\delta(s) - Lo(s)$), and how many risky hits you expect to get under uniform sampling versus policy sampling. □

#### B.1.1.0.1 Remark (Effect on the loss).

From the theorem, for any fixed state $s$ and sample size $m \geqslant 1$:

$$\mathbb{E}\Big[\Delta_{\text{Unif}}^{(c)}(s) - \Delta_\pi^{(c)}(s)\Big] \geqslant c(s)(\delta(s) - Lo(s))\big((1 - p_\pi(s))^m - (1 - p_{\text{Unif}}(s))^m\big) - \log m. \tag{B.16}$$

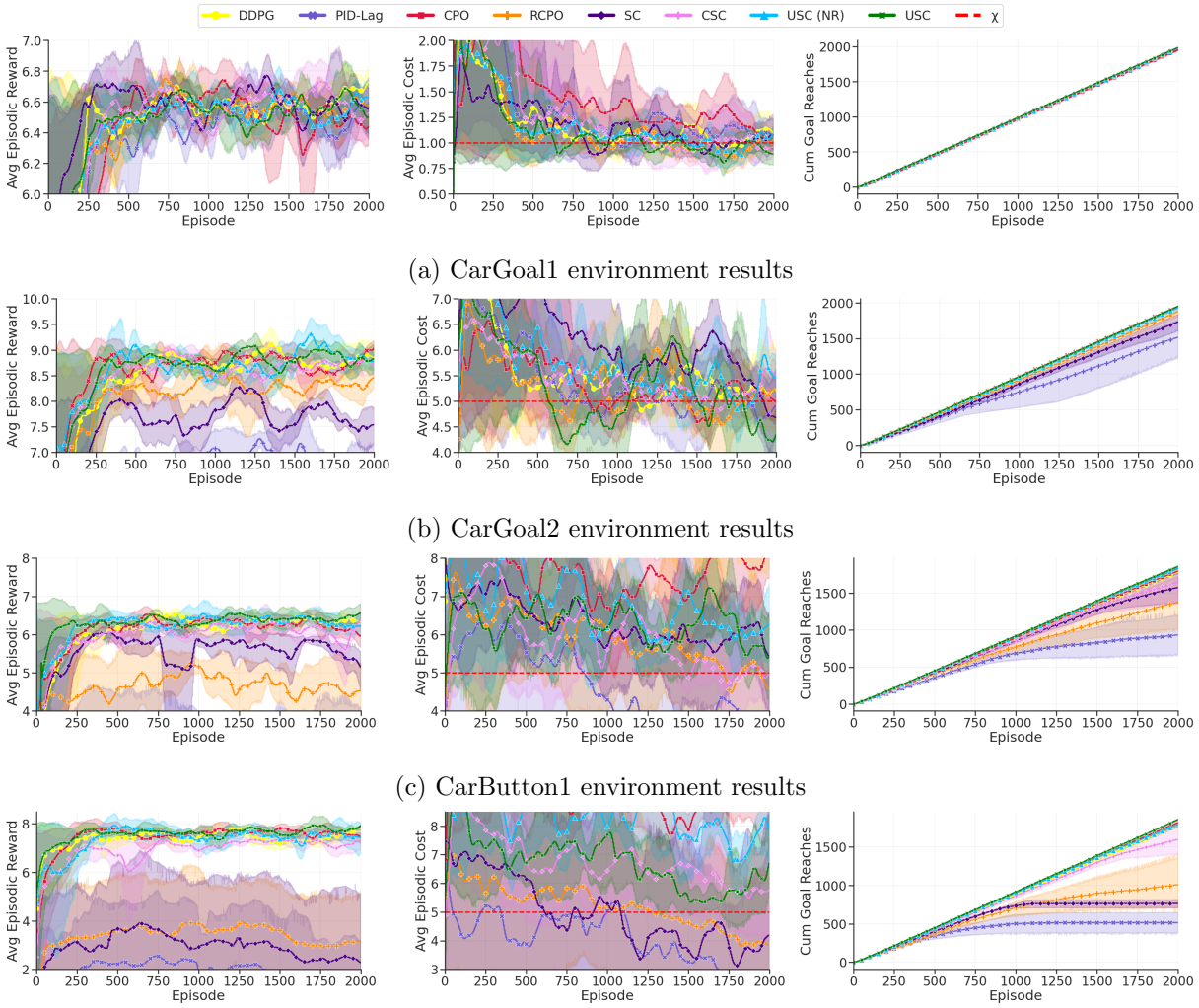
Hence, whenever the right-hand side of Equation B.16 is positive, the conservative penalty term inside Equation 4.4 is, in expectation, strictly larger under uniform sampling than under policy sampling:

$$\mathbb{E}\Big[\Delta_{\text{Unif}}^{(c)}(s)\Big] \geqslant \mathbb{E}\Big[\Delta_\pi^{(c)}(s)\Big] + c(s)(\delta(s) - Lo(s))\big((1 - p_\pi(s))^m - (1 - p_{\text{Unif}}(s))^m\big) - \log m. \tag{B.17}$$

When the uncertainty modulation $\tilde{u}(s, a_{\text{data}}) \geqslant 0$ is applied as in Equation 4.4, $L_C^{(c)}(s, a_{\text{data}}) = \tilde{u}(s, a_{\text{data}})\, \Delta^{(c)}(s)$, the inequality is preserved (it is multiplied by the same nonnegative factor on both sides):

$$\mathbb{E}\Big[L_C^{(c)} \big|\, \text{Unif}\Big] - \mathbb{E}\Big[L_C^{(c)} \big|\, \pi\Big] \geqslant \tilde{u}(s, a_{\text{data}})\Big(c(s)(\delta(s) - Lo(s))\big((1-p_\pi(s))^m - (1-p_{\text{Unif}}(s))^m\big) - \log m\Big). \tag{B.18}$$

Consequently, if $\tilde{u}(s, a) > 1$, then the conservative bias is *amplified*, and uniform sampling yields strictly stronger pessimism than policy sampling. If $\tilde{u}(s, a) < 1$, the conservative bias is weakened, but the loss remains more conservative than under policy sampling alone, only scaled down by $\tilde{u}(s, a)$. This scaling typically results in less conservatism than the policy sampling without uncertainty modulation, thus being less conservative. Evidently, the loss will create sharper gradients than alternative methods due to increased conservatism in high-cost, high-uncertainty regions and reduced conservatism in low-uncertainty regions.




(a) CarGoal1 environment results

(b) CarGoal2 environment results

(c) CarButton1 environment results

(d) CarButton2 environment results

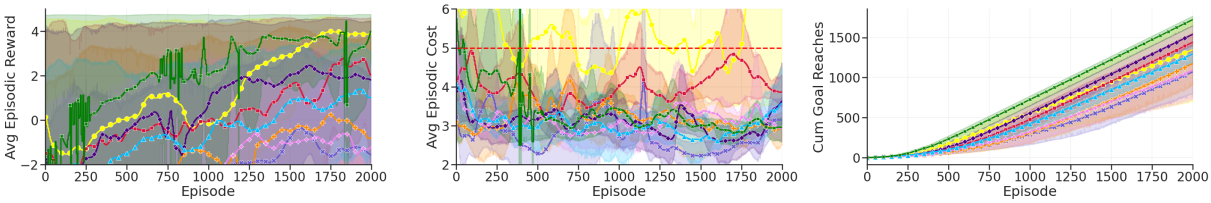
Figure B.1: Average episodic reward, cost, and cumulative goal reaches of examined methods (DDPG, Safety Critic, Conservative Safety Critic, Uncertain Safety Critic) for the CarGoal1 (a), CarGoal2 (b), CarButton1 (c), and CarButton2 (d) environments.

## B.2 Full Training Results

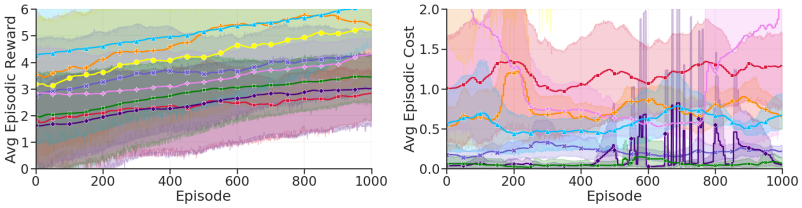
In Section 4.3.1, we show the main results for all approaches in a set of tasks. In Figure B.1, we show the full training plots for all results/methods shown in Table 4.1.

Across all four Safety Gymnasium tasks, the training curves show that USC delivers the sharpest reward–cost trade-off and the most stable learning dynamics. In CarGoal1 and CarGoal2, USC maintains a high average reward while keeping the average cost close to (or below) the competing safety critic baselines, with visibly tighter confidence bands than SC and CSC. This mirrors the summary statistics in Table 4.1, where USC attains the top reward in CarGoal2 ($\approx 8.84$) while achieving the lowest cost among the learning methods ($\approx 5.05$ vs. $\approx 5.36$ for SC), indicating that the improved critic gradients translate directly into better online learning traces.

As task difficulty scales (CarGoal → CarButton), the limitations of prior critics become pro-




(a) FetchReach environment results



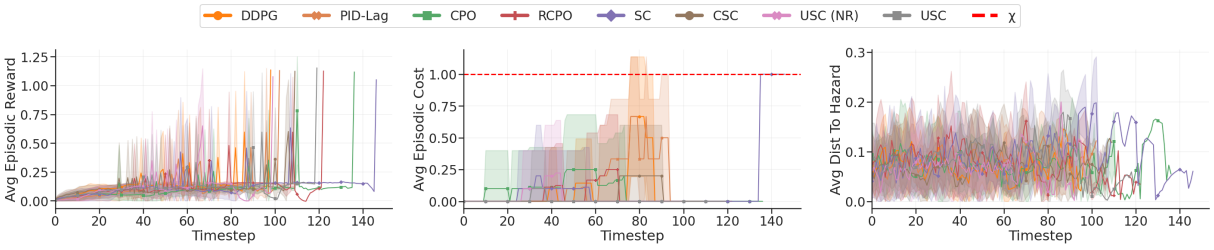

(b) HalfCheetah environment results

Figure B.2: Average episodic reward, cost, and cumulative goal reaches of examined methods (DDPG, Safety Critic, Conservative Safety Critic, Uncertain Safety Critic) for the CarGoal1 (a), CarGoal2 (b), CarButton1 (c), and CarButton2 (d) environments.

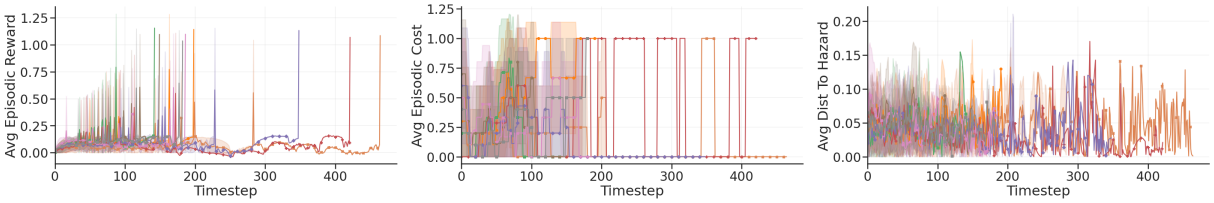
nounced. The standard Safety Critic suffers reward collapse and erratic cost spikes; in the cumulative goal-reach plots, its curve frequently flattens: evidence that the policy stops solving episodes for extended stretches. In CarButton2, this stagnation is particularly evident: SC's cumulative successes plateau early ($\approx$ episode 1000), whereas USC continues to accrue goals nearly linearly through training, indicating sustained policy improvement rather than oscillation. The Conservative Safety Critic agent remains more stable than SC but still degrades at larger scales: it avoids the worst collapses yet learns noticeably slower and accumulates substantially fewer successes (as seen at $\approx$ episode 1750 for both CarButton environments), consistent with its over-conservatism and the reward underperformance reported in Table 1 (e.g., CarButton2 reward $\approx 6.45$ for CSC vs. $\approx 7.69$ for USC).

In FetchReach, seen in Figure B.2, USC stabilises around the highest reward levels while maintaining the lowest episodic costs, yielding the steepest cumulative goal-reach curve among all methods. Safety Critic shows highly erratic cost spikes and inconsistent reward, whereas CSC avoids catastrophic collapse but underperforms in both reward and goal completion compared to USC. In fact, USC achieves the highest goal success rate in the environment. These dynamics confirm that USC can scale beyond Safety Gym to more structured manipulation tasks without losing stability.

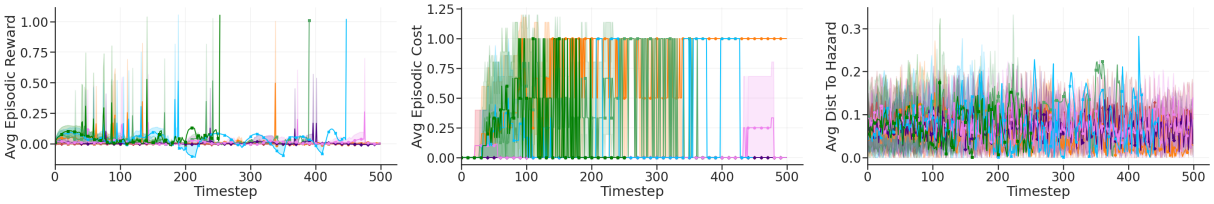
In HalfCheetah, the divergence between methods is even more pronounced. SC rapidly destabilises, with its cost curve oscillating wildly, while CSC remains overly conservative and is never is able to stabilise costs. In contrast, USC achieves monotonic reward growth coupled with suppressed cost variance, demonstrating both efficient exploration and safe policy improvement.



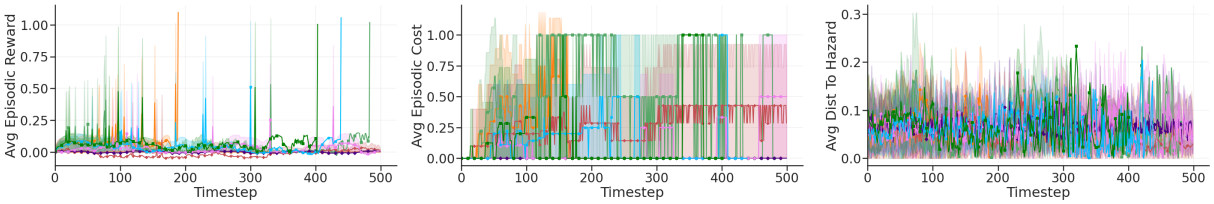

(a) CarGoal1 environment results




(b) CarGoal2 environment results




(c) CarButton1 environment results



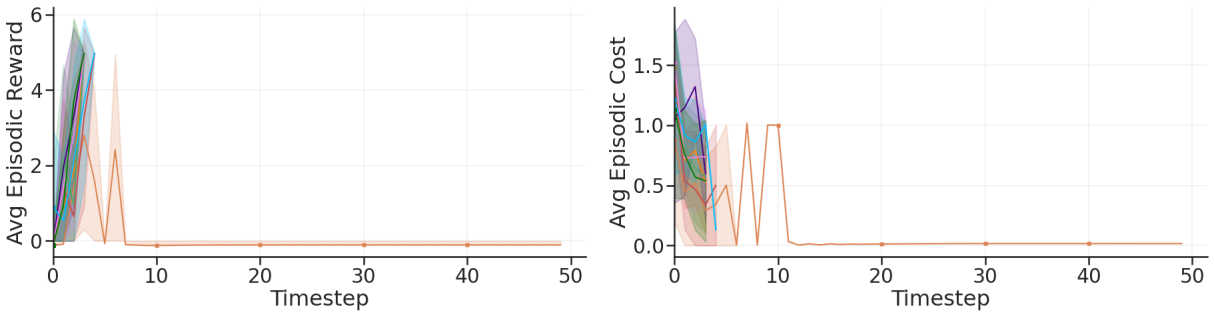

(d) CarButton2 environment results

Figure B.3: Average deployment episodic reward, cost, and cumulative goal reaches of examined methods (DDPG, Safety Critic, Conservative Safety Critic, Uncertain Safety Critic) for the CarGoal1 (a), CarGoal2 (b), CarButton1 (c), and CarButton2 (d) environments.

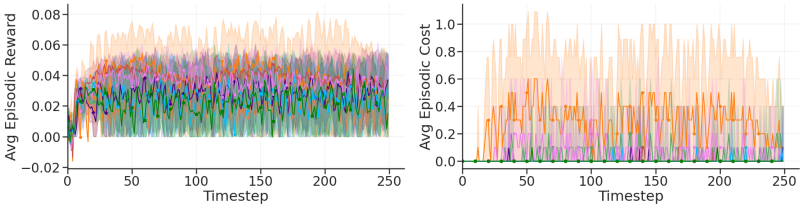
## B.3 Full Deployment Results

In Section 4.3.2, we show the main results for all approaches during deployment under a fixed policy. In Figure B.3, we show the full deployment plots for all results/methods shown in Table 4.3.

The deployment-phase evaluations confirm the trends reported. Across all CarGoal and CarButton tasks, USC consistently delivers higher reward while keeping costs relatively low compared to all other methods, and it maintains the highest average distance to hazards. Safety Critic again proves brittle, with frequent cost spikes and poor hazard avoidance, while CSC is more stable but overly conservative, often suppressing reward. Notably, in the more difficult CarButton tasks, USC is the only safety-aware method that sustains balanced performance without collapsing into unsafe or overly cautious behaviour and achieves 100% success rate, whereas




(a) FetchReach environment results




(b) HalfCheetah environment results

Figure B.4: Average episodic reward, cost, and cumulative goal reaches of examined methods (DDPG, Safety Critic, Conservative Safety Critic, Uncertain Safety Critic) for the CarGoal1 (a), CarGoal2 (b), CarButton1 (c), and CarButton2 (d) environments.

the other safety-aware methods fail to reach the goal every time. These results reinforce that USC's uncertainty-modulated training advantage translates into reliable and safe deployment performance.

## B.4 USC Experimental Setup

This section details the experimental setup used to evaluate USC in Section 4.3. All experiments share a common experimental backbone, which is described first, followed by experiment-specific subsections that document any deviations or additional configuration details required for individual evaluations.

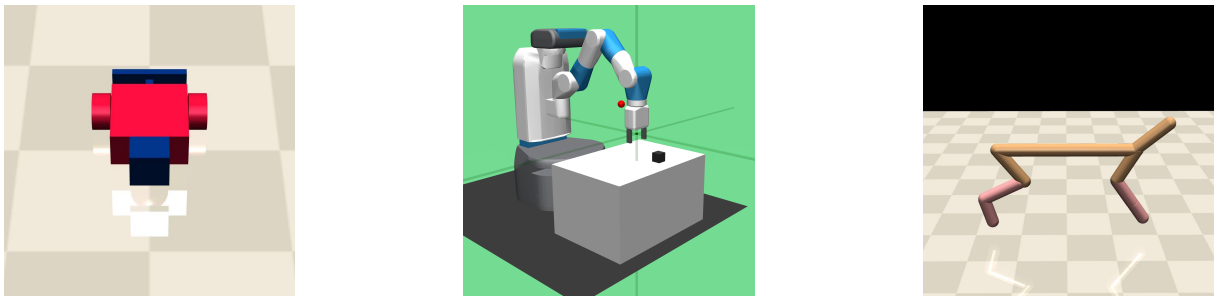
### B.4.1 Common Experimental Setup

This subsection describes the experimental configuration shared across all evaluations of USC in Section 4.3. Unless otherwise stated, all experiments follow this setup exactly.

#### B.4.1.1 Environments

We evaluate USC and the comparative methods in various tasks across three separate test suites. In our experiments, found in Section 4.3, we chose to use:

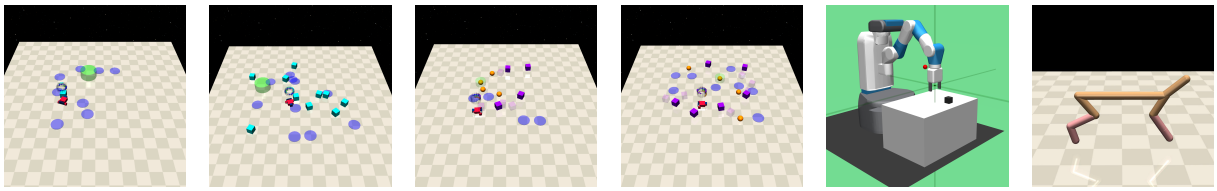
- **Car robot:** This robot has two wheels on the rear that the agent can control with one free-rolling front wheel. Steering and movement require nuanced coordination. The action space for the car is $[-1, 1]^2$, and the agent is shown in Figure B.5a.



**(a)** Car **(b)** Fetch Manipulator **(c)** Half Cheetah

Figure B.5: Different robots used within the tasks of the evaluated test-suites.

- **Fetch Manipulator:** A 7-DoF arm with a two-finger parallel gripper, controlled via small end-effector displacements in Cartesian space, with inverse kinematics handled internally. The action space is $[-1, 1]^4$, and the agent is shown in Figure B.5b.
- **HalfCheetah:** This robot is a planar bipedal agent with six actuated hinge joints. The action space is $[-1, 1]^6$, with each dimension applying torque to one joint. The agent is shown in Figure B.5c.
- **CarGoal1**: In this environment, the car-robot must steer to a goal region while avoiding nine hazards whose locations are randomised each episode. The agent, the goal, and hazards are placed randomly (within bounds) at reset. At each timestep, the agent observes LIDAR readings to the goal and to hazards (each with separate lidar observations), along with its own pose and velocity (forming a vector observation). The reward is composed of a "distance" term that gives $r_t = (D_{\text{last}} - D_{\text{now}})\,\beta$, plus a bonus $R_{\text{goal}}$ for reaching the goal. Costs are incurred when the agent comes into contact with any hazard (each hazard contact incurs a cost of 0.2). Episodes terminate after reaching the goal or exceeding a time limit (1000 steps).
- **CarGoal2**: This environment extends CarGoal1 by increasing the number of hazards to twenty, all with randomised positions at each episode reset. As in CarGoal1, the car-robot, the goal, and the hazards are placed randomly (within bounds). The agent receives observations consisting of separate LIDAR readings to the goal and to hazards, together with its own pose and velocity. The reward function is identical, with a distance-based shaping term $r_t = (D_{\text{last}} - D_{\text{now}})\,\beta$ and a goal-reaching bonus $R_{\text{goal}}$. Costs are incurred whenever the agent makes contact with hazards (each hazard contact incurs a cost of 0.2). Episodes terminate either upon reaching the goal or after 1000 steps.
- **CarButton1**: In this environment, the car-robot must drive to and press a correct button while avoiding nine hazards (including three incorrect buttons) whose positions are randomised each episode. At reset, the agent, buttons, and hazards are randomly placed within bounds. At each timestep, the agent observes separate LIDAR readings to the goal, buttons and to hazards, as well as its own pose and velocity (vector observation). The reward consists of a shaping term $r_t = (D_{\text{last}} - D_{\text{now}})\,\beta$ plus a bonus $R_{\text{button}}$ for successfully pressing the correct button. A cost (e.g. 0.2) is incurred whenever the agent contacts any hazard or an incorrect button. Episodes terminate upon pressing the button or exceeding



**(a)** Goal1 **(b)** Goal2 **(c)** Button1 **(d)** Button2 **(e)** Fetch Reach **(f)** Half Cheetah

Figure B.6: Example navigation tasks with varying complexity levels for evaluating USC.

a timestep limit (e.g. 1000 steps).

- **CarButton2**: This environment extends CarButton1 by increasing the number of hazards to fourteen (including three incorrect buttons), all with randomised positions at each episode reset. As in CarButton1, the car-robot, buttons, and hazards are placed randomly within bounds. The agent observes separate LIDAR readings to the goal, to buttons, and to hazards, together with its own pose and velocity. The reward function is identical, with a shaping term $r_t = (D_{\text{last}} - D_{\text{now}})\,\beta$ and a bonus $R_{\text{button}}$ for pressing the correct button. A cost (0.2) is incurred whenever the agent contacts any hazard or an incorrect button. Episodes terminate either upon pressing the correct button or after 1000 steps.
- **FetchReach**: In this environment, a 7-DoF Fetch manipulator robot must move its end effector to a randomly selected 3D goal position and maintain it. At reset, the desired goal is sampled uniformly within the reachable workspace. At each timestep, the agent receives a reduced observation consisting of the end effector position and velocity (6 dimensions), concatenated with the 3D goal position. The action space is $\text{Box}(-1, 1)^3$, representing Cartesian displacements $(dx, dy, dz)$ of the end effector (the gripper action is fixed to zero in this task). The reward is the negative Euclidean distance between the achieved and desired goal, with an additional bonus of $+5.0$ when the goal is reached within a tolerance of 0.05 m. Safety costs are imposed through two terms: (1) a velocity cost of 1.0 whenever the magnitude of the end effector velocity exceeds a threshold $v_{\max} = 0.01$, and (2) a jerk penalty $\lambda_{\text{jerk}} \sum_i (\Delta v_i)^2$ with $\lambda_{\text{jerk}} = 1000.0$. The total episodic cost is the sum of these penalties across the trajectory. Episodes terminate upon reaching the goal or after a maximum horizon of 500 steps.
- **HalfCheetah**: The HalfCheetah robot is a planar serial-link agent with six actuated hinge joints, controlled by torques in the action space $\text{Box}(-1, 1)^6$. The observation is the 17-dimensional vector of generalised positions (excluding root $x$) and velocities. The nominal reward is the forward velocity (distance travelled per timestep). In the Safe Velocity variant, a safety cost of 1.0 is incurred on any timestep where the magnitude of the agent's velocity exceeds a predefined threshold (set to 50% of its maximum speed under converged PPO). Episodic length is 500 steps.

In the Safety Gymnasium tasks evaluated here, obstacle configurations are randomised at each episode reset, including their positions and orientations where applicable. Obstacles remain stationary throughout each episode; the randomisation therefore varies the layout between episodes

Table B.1: USC hyperparameters used in the experimental evaluation.

| Parameter | Value |
|---|---|
| Lambda learning rate $\eta_\lambda$ | $1e-3$ |
| KL coefficient $\delta$ | $1e-6$ |
| Gauss-Newton dampening $\delta I$ | $1e-3$ |
| Constant $\epsilon$ | $1e-6$ |
| Sampled alternative actions $a'$ | 64 |
| Most uncertain samples $n$ | 5 |
| KL threshold $\beta$ | $1e-2$ |
| Safety tolerance $\epsilon_{safe}$ | 0.3 |

rather than introducing moving obstacles during navigation.

#### B.4.1.2 Model and Training

All USC experiments are built on top of a standard DDPG agent, which serves as the underlying continuous-control backbone.

The actor and critic networks are implemented as fully connected multilayer neural networks with identical architectures, each consisting of two hidden layers of size $(256, 256)$ and ReLU activations. The actor outputs deterministic continuous actions scaled to the environment's action bounds, while the critic estimates state–action values using a standard Q-function parameterisation. Target networks are maintained for both actor and critic and updated via soft Polyak averaging. Optimisation is performed using the Adam optimiser for both networks. The actor is trained with a learning rate of $2 \times 10^{-3}$, while the critic uses a learning rate of $1 \times 10^{-3}$, reflecting the standard practice of training the critic slightly more conservatively to stabilise value estimation. The discount factor is fixed to $\gamma = 0.95$, and target networks are updated using a soft update coefficient of $5 \times 10^{-3}$.

Experience replay is used throughout training, with a replay buffer capacity of $2 \times 10^5$ transitions. Mini-batches of size 64 are sampled uniformly from the buffer, and gradient updates are performed at every environment step once sufficient experience has been collected. To stabilise training, gradients are clipped with a maximum norm of 2.5.

Exploration during training is induced by adding temporally correlated Ornstein–Uhlenbeck noise to the actor's actions, with noise scale 0.2. This encourages smoother exploration in continuous action spaces, particularly in the early phases of training.

#### B.4.1.3 USC Hyperparameters

The hyperparameters reported in Table B.1 are fixed across all experiments and were selected to reflect standard, well-established practice rather than being tuned per environment. Parameters are set according to previous methods, which share parameters [213], as well as manual tuning.

#### B.4.1.4 Costmap Metrics

We evaluated USC on three costmap metrics: gradient MSE, contrast error, and entropy error. In this section, we formally define the calculation of these metrics.

Let $\hat{C}_{\text{env}} \in R^{H \times W}$ be the estimated cost map of the corresponding environment and $C^* \in R^{H \times W}$ be the ground truth costmap from the same environment. We split both maps into non-overlapping square patches of side length *patch* (we use $patch = 10$ in our experiments). Gradient MSE compares local spatial structure by measuring discrepancies in edge strength. For a map $\hat{C}$, we compute a gradient magnitude image using Sobel filters. Let $So_x$ and $So_y$ denote the horizontal and vertical Sobel kernels and $*$ denote a 2D convolution:

$$\begin{aligned} G_x(\hat{C}) &= So_x * \hat{C} \qquad G_y(\hat{C}) = So_y * \hat{C} \\ g(\hat{C})_{ij} &= \sqrt{G_x(\hat{C})_{ij}^2 + G_y(\hat{C})_{ij}^2} \end{aligned} \tag{B.19}$$

We then compute, for each patch $\Omega_{patch}$, the MSE between the gradient magnitudes:

$$err_k^{grad}(\hat{C}, C^*) = \frac{1}{|\Omega_{patch}|} \sum_{(i,j) \in \Omega_{patch}} (g(\hat{C})_{ij} - g(C^*)_{ij})^2 \tag{B.20}$$

Lower values indicate that the predicted map captures similar boundaries and transitions to the ground truth.

Contrast error evaluates whether the predicted map preserves the relative separation between hazardous and safe regions. We define a hazard mask from the ground truth using a fixed threshold $\delta$ (in our experiments, we use $\delta = 0.1$):

$$haz_{ij} = \mathbb{1}\{C^*_{ij} > \delta\} \tag{B.21}$$

For each patch $\Omega_{patch}$, we define the hazard and non-hazard sets:

$$\Omega_{patch}^{haz} = \{(i,j) \in \Omega_{patch} : haz_{ij} = 1\}, \quad \Omega_{patch}^{non-haz} = \{(i,j) \in \Omega_{patch} : haz_{ij} = 0\} \tag{B.22}$$

When a patch contains at least one hazard pixel and at least one safe pixel, we compute the within-patch contrast of a map $\hat{C}$ as the difference between the mean hazard cost and the mean non-hazard cost:

$$\Delta_{patch}(\hat{C}) = \frac{1}{|\Omega_{patch}^{haz}|} \sum_{(i,j) \in \Omega_{patch}^{haz}} \hat{C}_{ij} - \frac{1}{|\Omega_{patch}^{non-haz}|} \sum_{(i,j) \in \Omega_{patch}^{non-haz}} \hat{C}_{ij} \tag{B.23}$$

The patch contrast error is then

$$err_{patch}^{cont}(\hat{C}, C^*) = |\Delta_{patch}(\hat{C}) - \Delta_{patch}(C^*)| \tag{B.24}$$

This measures whether the model reproduces the hazard-vs-non-hazard gap observed in the ground truth. Lower values indicate the model preserves the intended separation.

The final metric, entropy error, compares the distributional complexity of predicted costs within each patch. For each patch $\Omega_{patch}$, we approximate the empirical distribution of values using a histogram with $\mathfrak{B}$ bins (we use $\mathfrak{B} = 50$ in our experiments). Let $\tilde{p}_{patch,b}(\hat{C})$ denote the normalised probability mass in bin $b$ for patch *patch*. We compute the Shannon entropy:

$$\mathcal{H}_{patch}(\hat{C}) = -\sum_{b=1}^{\mathfrak{B}} \tilde{p}_{patch,b}(\hat{C}) \log(\tilde{p}_{patch,b}(\hat{C}) + \epsilon) \tag{B.25}$$

with a small $\epsilon > 0$ for numerical stability. The patch entropy error is:

$$err^{ent}_{patch}(\hat{C}, C^*) = |\mathcal{H}_{patch}(\hat{C}) - \mathcal{H}_{patch}(C^*)| \tag{B.26}$$

Lower values indicate that the predicted map has a similar level of within-patch variability (e.g., uniform regions vs. mixed/high-frequency regions) to the ground truth.

# Appendix C

# Appendix for Chapter 5

## C.1 Theoretical Analysis

In this section, we theoretically analyse ADVICE and show that the expected probability of ADVICE misclassifying an unseen feature is bounded and can be decreased by diversifying the data collected before episode $E_{max}$, as well as the bounded regret that ADVICE adds to the underlying RL agent.

### C.1.1 Theorem 5.1

**Theorem 3.** The probability of ADVICE misclassifying a feature is bounded by $\exp(-\frac{m^2}{2\sigma^2})$, where $m$ is the contrastive separation margin and $\sigma^2$ is the variance of the Gaussian noise within the latent space.

*Proof.* The contrastive separation margin in the latent space is given by:

$$m = \min_{f_s \in F_{\text{safe}}, f_u \in F_{\text{unsafe}}} \|\mathcal{F}_{\text{encoder}}(f_s) - \mathcal{F}_{\text{encoder}}(f_u)\|_2 \tag{C.1}$$

The noise in the latent space is assumed to follow a Gaussian distribution $\epsilon \sim N(0, \sigma^2)$. In ADVICE, an unseen feature $f$ is classified as safe if the K-nearest neighbours of $\mathcal{F}_{\text{encoder}}(f)$ contain safer than unsafe features. In the following, we analyse the 1-nearest-neighbour decision rule, which provides an upper bound on the misclassification probability of the K-NN classifier used by ADVICE. Let:

$$\begin{aligned} dist_s &= min_{f_s \in F_{safe}} \|\mathcal{F}_{\text{encoder}}(f) - \mathcal{F}_{\text{encoder}}(f_s)\|_2 \\ dist_u &= min_{f_u \in F_{unsafe}} \|\mathcal{F}_{\text{encoder}}(f) - \mathcal{F}_{\text{encoder}}(f_u)\|_2 \end{aligned} \tag{C.2}$$

define the Euclidean distance to the nearest safe and unsafe feature, respectively, given a new feature $f$.

An incorrect decision (misclassification) occurs when $f \in F_{safe}$ and $dist_u < dist_s$ or $f \in F_{unsafe}$ and $dist_u > dist_s$. The contrastive separation margin $m$ ensures that, in a noise-free case $\|\mathcal{F}_{\text{encoder}}(f_u) - \mathcal{F}_{\text{encoder}}(f_s)\|_2 \geqslant m \; \forall f_s \in F_{safe} \; \forall f_u \in F_{unsafe}$. In RL environments such as those targeted by ADVICE, where noise exists, the distances $dist_s$ and $dist_u$ are perturbed by noise. Therefore:

$$dist_u - dist_s = \|\mathcal{F}_{\text{encoder}}(f) - \mathcal{F}_{\text{encoder}}(f_u)\|_2 - \|\mathcal{F}_{\text{encoder}}(f) - \mathcal{F}_{\text{encoder}}(f_s)\|_2 \approx m + \epsilon \tag{C.3}$$

where $\epsilon \sim N(0, \sigma^2)$ denotes aggregate latent-space noise and is independent and Gaussian. The noise $\sigma$ in the latent space can come from: noisy data, imperfect model training, randomness in batch sampling, etc. The probability of $dist_u < dist_s$ (misclassification) when $f \in F_{safe}$ is given by:

$$P(dist_u < dist_s) = P(m + \epsilon < 0) = P(\epsilon < -m) \tag{C.4}$$

Since $\epsilon \sim N(0, \sigma^2)$, we can normalise it so that:

$$P(dist_u < dist_s) = P(Z < -\frac{m}{\sigma}) \tag{C.5}$$

where $Z \sim N(0, 1)$ is the standard normal random variable. Using the cumulative distribution function of the standard normal distribution $\Phi$, we get:

$$P(dist_u < dist_s) = \Phi(-\frac{m}{\sigma}) \tag{C.6}$$

ADVICE uses K-nearest neighbours to classify an unseen feature $f$. If $m$ is large relative to $\sigma$, the probability of misclassifying an unseen feature decreases exponentially. Using the standard Gaussian tail bound $\Phi(-x) \leqslant \exp(-x^2/2)$ for $x \geqslant 0$, we can define the misclassification probability is:

$$E[\text{feature misclassification}] \leqslant \exp\left(-\frac{m^2}{2\sigma^2}\right) \tag{C.7}$$

□

### C.1.2 Theorem 5.2

**Theorem 4.** The probability of ADVICE misclassifying a feature decreases exponentially with improved data entropy and is bounded by $\exp(-\frac{\mathcal{H}(F_E)}{2\sigma^2})$.

*Proof.* Let $m_{achieved}$ express the effective achieved separation margin between sets $F_{safe}$ and $F_{unsafe}$, where $m_{achieved} \leqslant m$. Equality only holds under ideal conditions, such as perfectly

diverse data, perfect model training, no data noise, etc. The diversity of the feature set $F_E$ collected before episode $E_{max}$ can be quantified using entropy:

$$\mathcal{H}(F_E) = -\sum_{f \in F_E} p(f)\ log\ p(f) \tag{C.8}$$

where $p(f)$ is the probability distribution of features $f \in F_E$. Higher entropy corresponds to a broader set of features, ensuring greater diversity. Greater diversity results in more representative embeddings, given good model training, allowing the contrastive loss function to achieve better separation and clusterings of sets $F_{safe}$ and $F_{unsafe}$. The effective separation margin $m_{achieved}$ depends on $\mathcal{H}(F_E)$. As diversity increases, the embeddings for $F_{safe}$ and $F_{unsafe}$ become more separable, thereby $m_{achieved} \propto k \cdot \sqrt{\mathcal{H}(F_E)}$ where $k > 0$ is a proportionality factor that links the entropy $\mathcal{H}(F_E)$ of the feature set to the effective separation margin $m_{achieved}$. It encapsulates the influence of latent space geometry, scaling properties, and model-specific parameters. While $k$ may vary depending on the training process and feature distribution, it is assumed to be stable for a given setup. Empirically, $k$ can be estimated by observing the relationship between $m_{achieved}$ and $\mathcal{H}(F_E)$ across diverse datasets or configurations. From Theorem 1, with the substitution of $m_{achieved}$ for $m$, the probability of misclassifying a feature is bounded by:

$$\begin{aligned} P(misclassification) &\leqslant \exp\left(-\frac{m_{achieved}^2}{2\sigma^2}\right) \\ &\leqslant \exp\left(-\frac{k \cdot \mathcal{H}(F_E)}{2\sigma^2}\right) \end{aligned} \tag{C.9}$$

therefore showing that increasing the diversity of the feature set reduces the misclassification probability of a feature exponentially.

□

### C.1.3 Theorem 5.3

**Theorem 5.** The expected regret of ADVICE with a fixed $K$ is upper bounded by a function of order $O\left(\frac{1}{\sqrt{E}}\right) + O\left((E - E_{max}) \cdot \Delta Q \cdot \exp\left(-\frac{m^2}{2\sigma^2}\right) \cdot \frac{1}{\sqrt{K}}\right)$.

*Proof.* The total regret of ADVICE can be decomposed into two individual components: the baseline regret from the RL process, and regret introduced from the shielding mechanism with a fixed $K$ accounting for misclassification. The baseline regret arises from the agent's standard exploration-exploitation trade-off in RL [26]. This term is defined as:

$$= O\left(\frac{1}{\sqrt{E_{max}}}\right) \tag{C.10}$$

which reflects the improvement in the agent's policy quality over $E_{max}$ episodes. To construct the regret term for regret introduced from the shielding mechanism with a fixed $K$ accounting

for misclassification, we can use the probabilistic bound of misclassification from Theorem 1, $exp\left(-\frac{m^2}{2\sigma^2}\right)$. Let $\Delta Q = \sum_{t=1}^{T} \frac{Q^*(s_t,a_t)-Q^*(s_t,a_t^{'})}{T}$ denote the average regret per timestep caused by taking the misclassified action $a_t^{'}$ instead of the optimal action $a_t$. The shield also restricts the agent's action choices into a feasible set $\mathcal{F}_\mathcal{K}$ determined by $K$ where a higher $K$ (more cautious shield) reduces $\mathcal{F}_\mathcal{K}$, whilst a lower $K$ (more relaxed shield) increases $\mathcal{F}_\mathcal{K}$. Thus, the size of $\mathcal{F}_\mathcal{K}$ scales inversely with $\sqrt{K}$. We can define our regret for the shielding term as:

$$= O\left((E - E_{max}) \cdot \Delta Q \cdot exp\left(-\frac{m^2}{2\sigma^2}\right) \cdot \frac{1}{\sqrt{K}}\right) \tag{C.11}$$

□

### C.1.4 Theorem 5.4

**Theorem 6.** The expected long-term regret of ADVICE with an adaptive $K$ is lower than that of ADVICE with a fixed $K$, and is upper bounded by a function of order $O\left(\frac{1}{\sqrt{E}}\right) + O\left((E - E_{max}) \cdot \Delta Q \cdot \exp\left(-\frac{m^2}{2\sigma^2}\right) \cdot \frac{1}{\sqrt{K^*}}\right) + O\left(\frac{1}{E-E_{max}} \sum_{e=1}^{E-E_{max}} |K_e - K^*| \cdot \Delta Q_{K_e}\right)$.

*Proof.* The total regret of ADVICE with an adaptive $K$ can be decomposed into three individual components: the baseline regret from the reinforcement learning process, the regret introduced from the shielding mechanism with at the optimal $K^*$ accounting for misclassification, and the short-term regret incurred as $K$ deviates from the optimal $K^*$. We can use the baseline regret and the regret from the shielding mechanism from Theorem 5 with the substitution of $K^*$. The adaptive $K$ dynamically adjusts based on recent safety violations, thus leading to deviations from $K^*$. Therefore:

$$= O\left(\frac{1}{E - E_{max}} \sum_{e=1}^{E-E_{max}} |K_e - K^*| \cdot \Delta Q_{K_e}\right) \tag{C.12}$$

reflects the cost of dynamically adjusting $K$ as the agent converges to $K^*$, where $|K_e - K^*|$ is the deviation of $K_e$ from $K^*$ at episode $e$ and $\Delta Q_{K_e}$ is the regret introduces by $K_e$ at episode $e$.

Next, we prove that the long-term regret of adaptive $K$ is lower than that of fixed $K$. Adaptive $K$ adjusts dynamically based on the observed safety violations; this ensures that $K_e$ converges to an optimal $K^*$ as $E \to \infty$, thus minimising the misclassification and shielding regret. Let $K^*$ minimise the combined regret:

$$K^* = argmin_K \left(\Delta Q \cdot exp\left(-\frac{m^2}{2\sigma^2}\right) \cdot \frac{1}{\sqrt{K}} + \Delta Q_K \cdot \frac{1}{\sqrt{K}}\right) \tag{C.13}$$

where the first term represents misclassification regret, which decreases with increasing K, and the second term represents shielding regret, which increases with increasing K. The adaptive mechanism ensures that $K_e$ increases if safety violations are too frequent, and $K_e$ decreases

if safety violations are rare. As $E \to \infty$, the mechanism stabilizes $K_e$ near $K^*$, leading to $|K_e - K^*| \to 0$ as $e \to E$. The additional adjustment penalty from the adaptive $K$ mechanism (Equation C.12) vanishes as $|K_e - K^*| \to 0$, leaving only:

$$= O\left(\frac{1}{\sqrt{E}}\right) + O\left((E - E_{max}) \cdot \Delta Q \cdot exp\left(-\frac{m^2}{2\sigma^2}\right) \cdot \frac{1}{\sqrt{K^*}}\right) \tag{C.14}$$

For fixed $K$, the regret includes suboptimal shielding and misclassification effects: if $K > K^*$ then $K$ is too cautious and leads to high shielding regret, if $K < K^*$ then $K$ is too relaxed and leads to increased misclassification regret. Adaptive $K$, by converging to $K^*$, ensures that neither of these extremes persists in the long term, achieving a lower total regret.

While adaptive $K$ introduces early added regret, but diminishes as $K_e \to K^*$. This ensures that the long-term regret is dominated by the optimal $K^*$, unlike a fixed $K$, which does not adapt to the environment. Let $\epsilon$ represent the difference between $K$ and $K^*$. For a fixed $K$, $\epsilon$ is constant leading to regret terms proportional to $|\epsilon|$. For adaptive $K$, $|\epsilon_e|$ decreases over episodes due to convergence $|\epsilon_e| \propto \frac{1}{\sqrt{e}}$. This makes the additional regret introduced by the adaptive $K$ approach 0 as $E \to \infty$. □

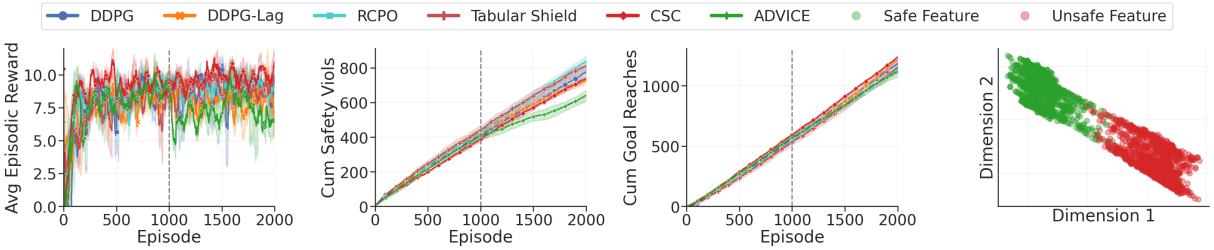
## C.2 Full Training Results

In Section 5.3.1, we show the main results for all approaches in a set of tasks. For fair comparison, we show results from episode 1000 and standardise all metrics to zero. Below, in Figure C.1, we show the unstandardised results for the same experiments.

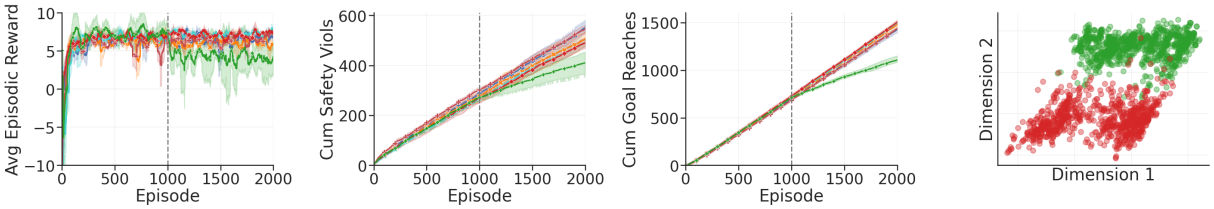
Across all environments, the full-horizon results exhibit the same qualitative trends as those discussed in Section 5.3.1. In particular, ADVICE consistently achieves a lower cumulative safety violation rate than baseline safe RL methods, including approaches that benefit from explicit cost signals or early safety guidance, where ADVICE's shield is *switched on* at episode 1000. While several methods obtain an initial advantage through predefined constraints, ADVICE adapts over time and converges to safer behaviour without sacrificing goal achievement. This confirms that the safety improvements observed after episode 1000 persist over the entire training horizon, demonstrating stable long-term adaptation rather than transient effects. Numerical values of all training results can be found in Table C.1.

## C.3 ADVICE Experimental Setup

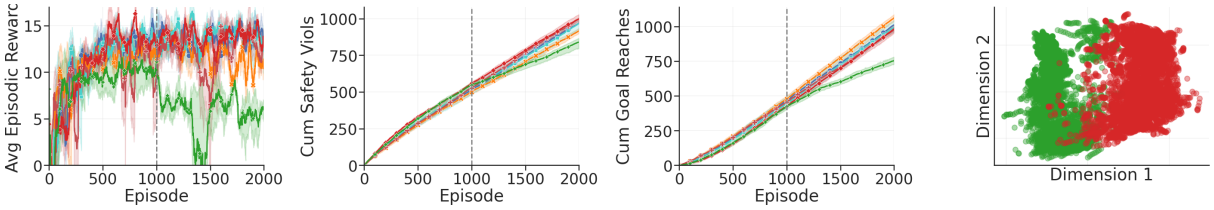
This section details the experimental setup used to evaluate ADVICE in Section 5.3. All experiments share a common experimental backbone, which is described first, followed by experiment-specific subsections that document any deviations or additional configuration details required for individual evaluations.



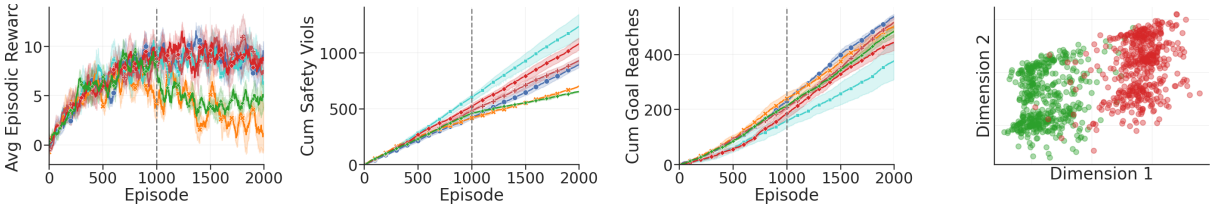

(a) Semi-random goal environment results




(b) Randomised goal environment results




(c) Randomised button environment results




(d) Randomised circle environment results

Figure C.1: Average episodic reward, cumulative safety violations, cumulative goal reaches of examined approaches and example latent visualisation for the semi-random goal (a), randomised goal (b), randomised button (c), and randomised circle (d) environments. Results are shown for episodes 0–2000 to highlight ADVICE's execution and adaptation.

## C.3.1 Common Experimental Setup

This subsection describes the experimental configuration shared across all evaluations of ADVICE in Section 5.3. Unless otherwise stated, all experiments follow this setup exactly.

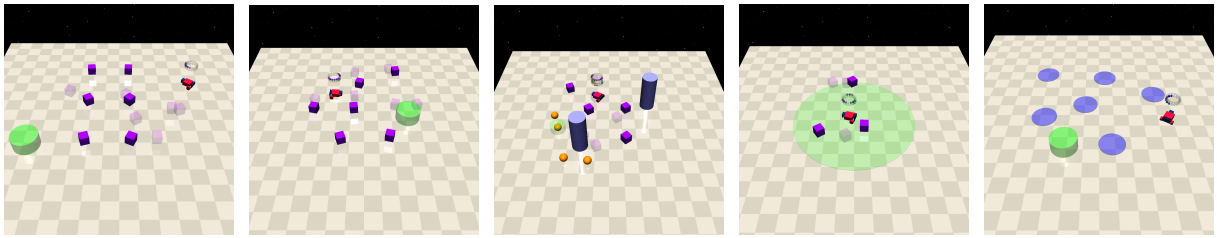
### C.3.1.1 Environments

We evaluate ADVICE and the comparative methods in various tasks across the Safety Gymnasium[257] test suite. In it, a robot with Lidar sensors has to navigate through environments with obstacles to complete a given task. The test suite comes with a multitude of robots (e.g. Point, Ant, Car) and a set of tasks (e.g. Goal, Circle, Button) that can be evaluated. In our experiments, found in Section 5.3, we chose to use:

Table C.1: Performance of the evaluated methods across five environments.

| Method | Average Reward ↑ | Cumulative violations ↓ | Cumulative goal reaches ↑ |
|---|---|---|---|
| **Semi-random goal** | | | |
| DDPG | $9.0 \pm 1.0$ | $366.0 \pm 15.4$ | $626.0 \pm 15.1$ |
| DDPG-Lag | $6.9 \pm 1.5$ | $292.3 \pm 7.0$ | $\mathbf{675.0 \pm 7.3}$ |
| RCPO | $8.8 \pm 1.1$ | $405.0 \pm 20.8$ | $587.4 \pm 22.6$ |
| Tabular Shield | $9.5 \pm 0.7$ | $369.0 \pm 4.0$ | $622.0 \pm 8.5$ |
| CSC | $\mathbf{10.9 \pm 1.2}$ | $350.0 \pm 15.0$ | $643.4 \pm 10.0$ |
| ADVICE | $6.7 \pm 1.5$ | $\mathbf{234.0 \pm 20.0}$ | $547.7 \pm 36.6$ |
| **Randomised goal** | | | |
| DDPG | $6.8 \pm 0.6$ | $264.0 \pm 29.9$ | $726.0 \pm 31.3$ |
| DDPG-Lag | $5.9 \pm 0.6$ | $228.3 \pm 14.2$ | $765.3 \pm 20.7$ |
| RCPO | $\mathbf{7.8 \pm 0.7}$ | $223.4 \pm 10.4$ | $773.0 \pm 12.0$ |
| Tabular Shield | $7.3 \pm 0.5$ | $243.4 \pm 23.7$ | $750.0 \pm 19.4$ |
| CSC | $7.4 \pm 0.5$ | $221.0 \pm 6.6$ | $\mathbf{778.0 \pm 8.8}$ |
| ADVICE | $4.2 \pm 2.0$ | $\mathbf{138.6 \pm 32.5}$ | $395.0 \pm 26.4$ |
| **Randomised button** | | | |
| DDPG | $\mathbf{14.0 \pm 1.3}$ | $452.6 \pm 7.8$ | $547.9 \pm 15.9$ |
| DDPG-Lag | $10.6 \pm 1.1$ | $415.0 \pm 6.2$ | $\mathbf{582.8 \pm 9.2}$ |
| RCPO | $12.3 \pm 1.7$ | $442.3 \pm 6.4$ | $550.0 \pm 3.7$ |
| Tabular Shield | $12.8 \pm 2.0$ | $472.0 \pm 8.9$ | $521.1 \pm 16.4$ |
| CSC | $13.7 \pm 1.5$ | $439.0 \pm 24.0$ | $559.0 \pm 25.6$ |
| ADVICE | $6.2 \pm 1.0$ | $\mathbf{294.0 \pm 16.0}$ | $328.2 \pm 9.0$ |
| **Randomised circle** | | | |
| DDPG | $\mathbf{9.5 \pm 1.0}$ | $473.0 \pm 21.9$ | $307.0 \pm 4.7$ |
| DDPG-Lag | $1.0 \pm 2.2$ | $284.4 \pm 12.2$ | $261.7 \pm 23.3$ |
| RCPO | $6.3 \pm 1.2$ | $634.6 \pm 89.3$ | $220.5 \pm 43.2$ |
| Tabular Shield | $7.4 \pm 1.0$ | $450.0 \pm 27.7$ | $\mathbf{309.3 \pm 18.0}$ |
| CSC | $8.7 \pm 1.1$ | $563.0 \pm 42.4$ | $258.5 \pm 31.2$ |
| ADVICE | $5.1 \pm 1.5$ | $\mathbf{193.1 \pm 14.5}$ | $269.5 \pm 18.4$ |
| **Constrained randomised goal** | | | |
| DDPG | $8.8 \pm 0.5$ | $2.0 \pm 0.4$ | $998.7 \pm 1.1$ |
| DDPG-Lag | $6.6 \pm 2.0$ | $2.0 \pm 1.0$ | $999.2 \pm 0.7$ |
| RCPO | $3.8 \pm 4.5$ | $2.4 \pm 1.5$ | $738.8 \pm 19.3$ |
| Tabular Shield | $9.1 \pm 0.6$ | $\mathbf{3.5 \pm 0.4}$ | $998.9 \pm 0.9$ |
| CSC | $\mathbf{9.3 \pm 0.5}$ | $1.8 \pm 1.7$ | $997.4 \pm 0.3$ |
| ADVICE | $5.3 \pm 1.0$ | $1.4 \pm 0.3$ | $696.7 \pm 54.5$ |

- **Car robot:** This robot has two wheels on the rear that the agent can control with one free-rolling front wheel. Steering and movement require nuanced coordination. The action space for the car is $[-1, 1]^2$
- **Semi-random Goal**: A standard goal environment, where the agent aims to reach the goal at the end of the episode whilst navigating through six obstacles. The six obstacles have a static spawn, and the agent and the goal have randomised positions every episode. We placed the six obstacles to form a large *wall*, where the agent can fit through to reach



**(a)** Semi-random Goal **(b)** Randomised Goal **(c)** Randomised Button **(d)** Randomised Circle **(e)** Constrained Randomised Goal

Figure C.2: Example navigation tasks with varying complexity levels for evaluating ADVICE. The purple blocks are the obstacles, the green circles are the goals, orange spheres are the buttons, and the red vehicle is the RL agent.

the goal but with an increased risk of crashing. In this instance, we want to determine if ADVICE and other safe RL exploration approaches will learn to avoid the *wall* or risk going through it. Deployment trajectories in Figure 5.3, show the learnt trajectories of ADVICE and other approaches.

- **Randomised Goal**: This environment is similar to the semi-random goal environment, with the additional complexity of the obstacles also having random spawns. This extra randomised aspect adds increased difficulty as the agent and safety mechanisms cannot memorise the positions of the obstacles to avoid.
- **Randomised Button**: The environment has two pillar obstacles, four regular obstacles, and four buttons all with random positions and rotations every episode. The agent has to press the correct button which is selected randomly per episode. If the agent collides with an obstacle or presses the incorrect button, the episode will terminate. This scenario increases the task difficulty as the safe and unsafe features collected share a significant overlap.
- **Randomised Circle**: The agent has to circle in a given zone in this environment. The aim is to maximise speed and distance from the centre of the zone whilst navigating through three randomised obstacles. This scenario element further increases the task difficulty as now the obstacles to avoid are directly within the area where the agent can maximise its reward.
- **Constrained Randomised Goal**: This environment is similar to the random goal environment, but instead the task is set up as a constrained MDP. Instead of obstacles that terminate the episode, the task includes hazards that give the agent 0.2 cost per step when inside them. The agent cannot terminate in this task and instead has to minimise cost whilst maximising rewards.

In all environments for all the tasks above, the agent uses *pseudo* Lidar to perceive objects in the environment. Each type of object (e.g. goal, obstacles) in the environment has its own separate Lidar observation, where a Lidar vector has 16 bins. All vectors are flattened into one observational vector and then given to the agent as the current state. For example, in the semi-random goal environment, there is a goal and a set of obstacles. Here the observational

space is $[0, 1]^{32}$. All lidar vectors have a max distance of 3 meters. In both goal environments, an episode has a maximum timestep of 1000. In the circle environment, the maximum timestep is 500. A *goal reach* in both goal environments is defined as reaching the goal before the episode truncates; in the circle environment, it is defined as being within the circle when the episode truncates. A *crash* is defined as the agent colliding with an obstacle; when this occurs, the episode is terminated shortly. Anything else is considered the episode *timing out*. For the cost-based approaches in the unconstrained environments, the approaches receive a cost of 0.2 when entering a non-accepting terminal state. This was chosen via a mixture of manual tuning, the recommended values from Safety Gymnasium, and comparative approaches.

Each task has a separate reward function for the agent to maximise. Whenever the agent comes into contact with an obstacle, a constraint cost of $-1$ is given (the exception being the Constrained Randomised Goal task). The reward functions for each task are:

- **Semi-random, Randomised Goal, Randomised Button, & Constrained Randomised Goal**: $R_t = (D_{last} - D_{now})\beta$, where $D_{last}$ is the distance between the agent and the goal in timestep $t_{-1}$, $D_{now}$ is the distance between the agent and the goal in timestep $t$, and $\beta$ is the discount factor. Simply, the agent moving towards the goal, in terms of Euclidean distance, gains a positive reward. The agent moving away from the goal gains a negative reward. Reaching the goal gains a static reward of $+1$.
- **Randomised Circle**: $R_t = \frac{1}{1+|r_{agent}-r_{circle}|} * \frac{(-uy+vx)}{r_{agent}}$ where $(u, v)$ is the x-y velocity coordinates of the agent, $(x, y)$ are the x-y coordinates, $r_{agent}$ is the Euclidean distance of the agent from the origin of the circle, and $r_{circle}$ is the radius of the circle. Simply, the agent is rewarded for moving at speed along the circumference of the circle.

It should be emphasised that the agent or approaches used in the evaluation have no prior knowledge of the task/environment/safety concern. Thus, we can define this environment and all tasks within as *black box.*

#### C.3.1.2 Model and Training

All ADVICE experiments are built on top of a standard DDPG agent, which serves as the underlying continuous-control backbone.

The actor and critic networks are implemented as fully connected multilayer neural networks with identical architectures, each consisting of two hidden layers of size $(256, 256)$ and ReLU activations. The actor outputs deterministic continuous actions scaled to the environment's action bounds, while the critic estimates state–action values using a standard Q-function parameterisation. Target networks are maintained for both actor and critic and updated via soft Polyak averaging. Optimisation is performed using the Adam optimiser for both networks. The actor is trained with a learning rate of $2 \times 10^{-3}$, while the critic uses a learning rate of $1 \times 10^{-3}$, reflecting the standard practice of training the critic slightly more conservatively to stabilise value estimation. The discount factor is fixed to $\gamma = 0.95$, and target networks are updated using a soft update coefficient of $5 \times 10^{-3}$.

Experience replay is used throughout training, with a replay buffer capacity of $2 \times 10^5$ transi-

Table C.2: ADVICE hyperparameters used in the experimental evaluation.

| **Parameter** | **Value** |
|---|---|
| Network size | (512, 2, 512) |
| Optimizer | Adam |
| Learning rate | Reduce-on-plateau $1e-6$ |
| Batch size | 32 |
| Maximum epochs | 500 |
| Number of neighbours $K_{\max}$ | 5 |
| Number of safe neighbours $K$ | 4 |
| Loss functions | MSE, MSE, CL |
| Loss weights | (1, 1, 1.25) |
| Unshielded episodes $E_{max}$ | 1000 |
| Contrastive margin $m$ | 1.0 |

tions. Mini-batches of size 64 are sampled uniformly from the buffer, and gradient updates are performed at every environment step once sufficient experience has been collected. To stabilise training, gradients are clipped with a maximum norm of 2.5.

Exploration during training is induced by adding temporally correlated Ornstein–Uhlenbeck noise to the actor's actions, with noise scale 0.2. This encourages smoother exploration in continuous action spaces, particularly in the early phases of training.

#### C.3.1.3 ADVICE Hyperparameters

The hyperparameters reported in Table C.2 are fixed across all experiments and were selected to reflect stable and commonly used settings for latent representation learning and nearest-neighbour classification. Parameters were not tuned per environment and were chosen to ensure consistent behaviour across tasks.