Uncertainty Quantification in Vision Based Classification

Mahed Javed

A thesis submitted in partial fulfilment

of the requirements for the degree of

Doctor of Philosophy

Department of Automatic Control and Systems Engineering

University of Sheffield, United Kingdom

March 2022

Abstract

The past decade of artificial intelligence and deep learning has made tremendous progress in highly perceptive tasks such as image recognition. Deep learning algorithms map high dimensional complex representations to low dimensional array mappings. However, these mappings are generally blindly assumed to be correct, further justified with high accuracies on trending datasets. The challenge of creating a comprehensive, explainable and reasonable deep learning system is yet to be solved. One way to deal with this is by using uncertainty quantification, or uncertainty aware learning, with the help of Bayesian methods.

This thesis contributes to the field of uncertainty aware learning by demonstrating how uncertainty can be used to recover performance in case of a physical attack, how uncertainty can be used to improve sensitivity to noise and how it can be used to improve performance on dynamic datasets. The first contribution involves learning from model uncertainty in the application of deep learning-based semantic segmentation. The second contribution deals with robustness and sensitivity analysis in image classification and finally, the third contribution in continual learning by using variance to update the learning rate. The first contribution proposes the architecture AdvSegNet which aims to improve the performance of Bayesian SegNet. In the second contribution, a combined architecture of convolutional network feature extractor and a Gaussian process (CNN-GP) is made to classify images under uncertain conditions including noise, blurring and adversarial attacks. Finally, in the continual learning subject area, the architecture CNN-GP is trained on datasets presented sequentially. Results show an improvement in performance and sensitivity to adversarial attack and noisy conditions as well as an improvement in dynamic datasets with a small number of tasks.

Acknowledgements

I would first like to thank my PhD advisor **Prof. Lyudmila Mihaylova**. She has been a great inspiration to me throughout my journey as a budding researcher. This would not have been possible without her support. I would also like to thank my former PhD advisors **Dr. Shiyu Zhao** and **Prof. Robert Harrison** for initially introducing me to the research world and providing invaluable feedback from critical analysis exercises and confirmation review.

I must express my very profound gratitude to my parents, to my brothers and sisters for providing me with unfailing support and continuous encouragement throughout my PhD journey.

To my family ...

Table of Contents

List of Figures

List of Tables

Nomenclature

Acronyms / Abbreviations

AdvSegNet	Adversarial SegNet
AWGN	Additive White Gaussian Noise
AI	Artificial Intelligence
BDL	Bayesian Deep Learning
BLEU	Bilingual Evaluation Understudy
BNNs	Bayesian neural Networks
CIFAR	Canadian Institute for Advanced Research
CL	Continual Learning
CNN	Convolution Neural Network
Cov	Covariance Operator
CRFs	Conditional Random Fields
CWA	Carlini and Wegner Attacks
DCGAN	Deep Convolutional Generative Adversarial Networks
DNN	Deep Neural Network

ELBO	Evidence Lower Bound
EWC	Elastic Weight Consolidation
FGSM	Fast Gradient Sign Method
FN	False Negatives
FP	False Positives
FPR	False Positive Rate
GAN	Generative Adversarial Networks
GP	Gaussian Process
GPU	Graphical Processing Units
GTAE	Ground Truth Adversarial Example
HOG	Histogram of Oriented Gradients
HSI	Hue Saturation Intensity
i.i.d	Independently and Identically Distributed
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
JSD	Jensen Shanon Divergence
JSMA	Jacobian Saliency Map Attacks
KLD	Kullback-Leibler Divergence
LSQGAN	Least Squares Generator Adversarial Network
MAP	Maximum A Posteriori
MB	Motion Blur

MC	Maximum Correntropy
mIoU	Mean Intersection Over Union
MLE	Maximum Likelihood Estimation
MNIST	Modified National Institute of Standards and Technology
NN	Neural Network
PAC	Probably Approximately Correct
PCA	Principal Component Analysis
PGDA	Projected Gradient Descent Attacks
QC	Quality Critic
RBF	Radial Basis Function
RDF	Random Decision Forest
ReLU	Rectified Linear Unit
ResNet	Residual Convolutional Neural Network
RGB	Red, Green, Blue
SI	Synaptic Intelligence
SNR	Signal-to-Noise ratio
SqueezeNet	Squeezed Convolutional Neural Network
SVI	Stochastic Variational Inference
SVMs	Support Vector Machines
TN	True Negatives

TP	True Positives
TPR	True Positive Rate
UAL	Uncertainty Aware Learning
UAP	Universal Adversarial Perturbations
UC	Uncertainty Critic
UCB	Uncertainty Guided Continual Learning in Bayesian Networks
UQ	Uncertainty Quantification
VCL	Variance Guided Continual Learning
VGG	Visual Geometery Group
VI	Variational Inference

Chapter 1

Introduction

1.1 Research Questions

In the modern study of artificial intelligence (AI), deep learning methods involve passing input data to a sequential collection of linear functions followed by non-linear functions mapping to the output. Deep learning methods, as a result of this mapping, learn high-dimensional representations that best describe the input data. The majority of times, predictions from deep learning are usually taken as point estimates. These estimates tell very little on how the model perceives the problem, how confident it is in its predictions and more importantly whether it understands the problem or not. This thesis studies the application of uncertainty in the area of uncertainty aware learning in vision-based deep learning systems. In particular, there are three key research questions that this thesis aims to answer.

1. Can uncertainty be used to recover the performance of deep neural networks in segmentation in the presence of a physical attack that damages the input receptive field ?

2. Can regularisation be used to improve the performance of a deep neural network using a Gaussian process to quantify uncertainty. Furthermore, can the regularisation be used to improve the sensitivity of the uncertainty output to noise and adversarial attacks ?

3. Can uncertainty be used to improve performance on dynamically evolving datasets ?

Specifically, in the first research question a scenario where a neural network system encounters a physical attack where its sensory inputs are damaged such that it can only partially observe its inputs is considered. Then, using uncertainty in a loss function setting, it is then investigated whether the sudden drop in the performance can be recovered. This is also compared with training with the standard cross-entropy loss function. The second research question considers the case in which regularisation is used to improve the sensitivity of the variance information to adversarial attacks and noisy inputs. It also investigates whether the performance of the system is improved or not. Lastly, the third research question deals with investigating whether the uncertainty can be used to update the learning rate if the performance on various tasks within a dynamic dataset deteriorates. In Section 1.3 it is shown how each of the research questions is tied to the individual chapters of this thesis. In the next section, the motivation behind the topic of the thesis is discussed.

1.2 Research Motivation

This section aims to discuss the motivation behind the work presented in this thesis. Deep learning systems have been proven to have achieved performance in highly perceptive tasks such as image recognition to that of human-level perception. Then, there arise the questions that if deep learning systems perform so well on highly perceptive tasks, why should there be a need to model uncertainty in AI systems ? and why is modelling uncertainty important regardless ?.

Despite the success of deep learning in achieving near-human level performance in tasks such as image recognition, the majority of the research in deep learning is blindly assumed to be correct from high confidence predictions. Furthermore, slight modifications in inputs that are imperceptible to the human eye, are shown to be enough to fool deep learning methods to output erroneous predictions. This is vital in mission-critical systems that require fast and reliable decision-making.

Another issue with deep learning methods is that they are not transparent. Specifically, learning in deep learning methods is a black-box. It is impossible to understand what exactly the network has learned unless the network is interpretable. Some of the common properties of an interpretable AI system are: being able to account for uncertainty, being able to characterise the impact of inductive biases that arise from assumptions made during learning, being able to characterise model accuracy and transparency and finally, being able to characterise the impact of uncertainties in the presence of less-favourable working environments e.g. noisy inputs or adversarial attacks. This is vital in applications that involve rational decision making e.g. self-driving cars.

In summary, lack of interpretability and erroneous predictions are examples that motivate uncertainty quantification (UQ) in AI systems. The next section aims to link the research questions to the area of application of the proposed methods in the thesis, while also explaining the field of UQ.

1.3 Research Application Areas

This section aims to link the research questions to the individual research areas tackled within the methods proposed in the thesis. It also explains the motivation behind the choice of these particular topics, why these topics were specifically chosen, what are some of the research gaps in these topics and how uncertainty can be applied in these topics as part of the contributions of this thesis.

Semantic Segmentation and Generator Adversarial Learning

The first research question aims to deal with using uncertainty to recover performance in the area of semantic segmentation. As part of scene understanding, semantic segmentation methods aim at classifying the input images on the pixel level. This task requires the location of the class of each pixel in an image. The semantic segmentation differs from traditional image segmentation as the model not only learns to segment the images into feature maps but also logically place the categories together. For example, in an outdoor scene of a car, road and a pedestrian, the segmented outputs should place the car on top of the road and a pedestrian in a corner.

In the past, this was accomplished using traditional methods that required expert knowledge in the manual extraction of features. Ever since the rise of popularity of deep learning, the trend in segmentation has shifted to adopting more deep learning-based methods.

Semantic segmentation is important for studying uncertainty quantification since there is already a large body of work in the form of Bayesian deep learning based segmentation that uses Bayesian methods to quantify uncertainty in popular architectures such as the SegNet [7] and the U-Net [149]. However, still, there is the research gap of testing deep learning based semantic segmentation systems with attacks that can be physical, adversarial or simply noise based ones. Another gap is that papers that use Bayesian methods for deep learning based semantic segmentation simply output uncertainty in the form of variance information. Without ever utilising uncertainty as a learning tool or at least including it in the learning process. In the proposed methods in Chapter 3, generator adversarial learning is used to enable uncertainty as a loss function. This field is discussed next.

In generator adversarial learning, two networks compete in a min-max game. This is a two-player game. One player knows the move of the other, while the other player does not know. The player with the knowledge competes by maximising its utility, the player without knowledge competes by minimising the utility of its competitor. Though utility has many meanings in different games, it can simply be interpreted as a score or the success rate. In deep learning, this is exemplified by a generator (usually a decoder) that attempts to fool a discriminator (usually a classifier) into thinking that its outputs do not resemble the training distribution. The game is finished when the discriminator fails to differentiate between the output of the generator and the training distribution.

Robust Learning

The second research question aims to use regularisation to improve the sensitivity and the performance of deep learning systems in the presence of adversarial attacks, Gaussian white noise and blurring. The application is related to robust learning in image classification. Robust learning, also known as adversarial learning, is often confused with training in generator adversarial learning. The difference is that in adversarial, or robust learning, the learning system is under attack from an adversary that aims to fool the system by exploiting the weakness in the internal representation learned from set training samples. In a classifier, for example, this representation is the decision boundary.

Robust learning is important for studying uncertainty quantification as the literature in this field entail various attacks and defences that not only apply to image classification but also to cyber-physical systems as part of cybersecurity. A possible application of this can be from the point of sensor networks, where for example, a successful attack on the hub (the main sensor) will affect whatever sensor it communicates with.

The field of robust learning is suitable for the methods adopted in the thesis for reasons including a wealth of literature on adversarial attacks. Still, there is plenty of research gap in robust learning on developing attacks that can be easily generated utilising the feature space of the deep neural networks. The study from robust learning [191] shows that training on an individual type of adversarial attack will only guarantee robustness on that particular attack. There is a large gap in the research for attacks that can be easily generated and help in building a defense on numerous other attacks that can be either adversarial or noisy ones. The methods presented in Chapter 3 deal with proving robustness to noisy attacks, motion blurring and a white-box attack fast gradient sign method. These methods aim to fill the aforementioned gap in robust learning by using simply the learned feature space of the deep learning system. The final research question and its relation to its area of research are discussed next.

Continual Learning

The third research question aims to use the uncertainty, or the variance information to update the learning rate of the deployed system to improve performance in dynamic datasets. Dynamically evolving datasets change in terms of categories per task in continual learning, or life-long learning. In this type of learning, the main objective is to prevent forgetting knowledge learned from previous tasks. Task, in image classification based continual learning terminology, refers to a certain portion of the training set that contains a unique set of images and their labels that are used separately for training. In dynamic datasets, tasks are presented sequentially and the objective of continual learning is to perform accurately on both present and previously introduced tasks.

Continual learning is inspired by the learning mechanisms in the human brain that consolidate past knowledge and maintain their retention of previous tasks. During training, the learning agent is assigned to learn the current task or tasks without losing accuracy on previous ones. From the point of view of the thesis, using uncertainty quantification to study continual learning is important so that it can be investigated how modelling uncertainty can be used to improve performance in continual learning. There is a research gap in the application of uncertainty in continual learning, with works like [121] and [32] that all utilise a multi-classifier approach to continual learning. None of these methods applies a single classifier approach. The methods proposed in Chapter 3 aim to fill this gap using a single Gaussian processes classifier that performs on all tasks. This is a novel contribution to this field since there has not been a study that differentiates the performance of the single vs multi-classifier approach to continual learning problems. The next section studies the outline of the thesis while outlining the flow between the chapters.

1.4 Thesis Outline

This section details the content layout of this thesis. Specifically, this section discusses how are the chapters in the thesis organised, how are they related and what are the main contents of each chapter. The content detail regarding each of the chapters is arranged in separate paragraphs starting from Chapter 1 to Chapter 5. The link between Chapter 1 and Chapter 2 is that the latter provides the literature behind the key research questions highlighted in Chapter 1. The literature review from Chapter 2 is then used to motivate the methods proposed in Chapter 3 and the experiments in Chapter 4. This links Chapter 2 to Chapter 3 and 4. The final Chapter 5 provides the summary. It links to Chapter 3 as it provides the limitations of the methods proposed in this thesis. Furthermore, the final chapter also highlights the application of the proposed methods and concludes with future works and areas where the methods can be improved. The in-depth details regarding the content of the chapters are discussed next. Each paragraph discusses the individual chapters separately.

Chapter 1 introduces the research topic of this thesis. Firstly, it considers the three key research questions that are investigated in the thesis. Then, it proceeds to motivate the methodologies proposed in this thesis. It then describes the various fields tackled as well as provides the outline of the thesis and key contributions. Finally, this chapter ends by listing the publications as part of the contributions of this thesis.

Chapter 2 focuses on providing the literature behind the areas discussed in the research questions provided in Chapter 1. It also provides the background knowledge relevant to the subjects tackled in this thesis. Firstly, this chapter begins by providing a brief overview of the literature surrounding the methods. Here, the introduction to deep learning, Bayesian methods in deep learning and Gaussian processes are provided in Sections 2.1 and 2.1.13 respectively. Specifically, Section 2.1.1 outlines the difference between machine learning and deep learning, Section 2.1.3 outlines the vulnerabilities of deep learning networks and Section 2.1.4 discusses how the linearity in deep learning systems can be used to highlight the cause of their vulnerability to adversarial attacks and Section 2.1.5 introduces to the white-box adversarial attack the fast gradient sign method.

The focus of the chapter then changes to Bayesian methods in deep learning. Here, Section 2.1.6 begins by highlighting the importance of modelling uncertainty, the ethical perspective of AI and how Bayesian methods formalise uncertainty quantification and finally relating Bayesian methods to interpretability in AI. Then, Section 2.1.7 deals with the formal definition of Bayesian principles and in Section 2.1.8 the variational inference methods are introduced as an approximation to true posterior distribution as well as highlighting the advantages and disadvantages of variational inference. Further approximation methods in the form of reparameterisation trick and Monte Carlo integration that assist in simplifying variational inference are addressed in Section 2.1.9 and Section 2.1.10 respectively. Furthermore, Section 2.1.11 discusses how uncertainty can be modelled in Bayesian neural networks and the challenges of doing so are highlighted in Section 2.1.12. This is followed by Section 2.1.13 which details the introduction to Gaussian Processes.

Next, the focus of the chapter shifts to providing literature in the subject areas tackled in this thesis. This is organised into three essential key sections of this chapter. One discusses the literature behind semantic segmentation. This is in Section 2.2. The second discusses the literature behind robust learning. This is in Section 2.4. The third considers the literature behind continual learning in Section 2.5.

Section 2.2 begins by providing the background in the problem of scene understanding and segmentation. This is followed by Section 2.2.1 that reviews several deep learning based methods in segmentation. Also, this section goes in-depth in reviewing various architectures in deep learning based segmentation literature. The focus then slightly shifts to uncertainty based methods in segmentation, specifically in Bayesian SegNet discussed in Section 2.2.3. The use of generator adversarial neural networks is then discussed in Section 2.3.1. The improvements to basic generator networks in the form of convolutional architectures and least-squares generator networks are reviewed in Section 2.3.2 and Section 2.3.3 respectively. The semantic segmentation literature section ends with Section 2.3.4 which reviews the literature for adversarial learning methods in generator networks.

Considering the robust learning part of this chapter. First, Section 2.4 begins by providing a brief overview of the field of robustness, the motivations behind it as well as highlighting several issues in the applications of robust learning. The definition of robustness is provided in Section 2.4.1. Specifically, this section discusses both the informal definition and the formal definition of robustness. The formal definition is provided in Section 2.4.2 and the diagrammatical explanation of this is provided in Section 2.4.3. The next section (Section 2.4.4) addresses some of the challenges of robustness. The robust learning portion of Chapter 2 finishes with the literature survey on uncertainty and meta-learning based defences techniques in robust learning. This is presented in Section 2.4.5.

The final portion of this chapter concentrates on the area of continual learning. Firstly, Section 2.5 and 2.5.1 begin by providing a brief introduction and the motivation for continual learning. This is then followed by a discussion on Bayesian methods in continual learning. This is presented in Section 2.5.2. Finally, the summary of the entire reviewed literature in Chapter 2 is provided in the final section of this chapter, i.e. Section 2.6.

Chapter 3 focuses on providing the methods proposed by thesis on the subject of semantic segmentation, robust learning and continual learning. First, the proposed AdvSegNet is presented in Section 3.1.2. This section discusses the layer structures as well as the loss functions used in Section 3.1.3 and the optimisation and training in Section 3.1.4. Then, in Section 3.1.5, 3.1.7 and 3.1.8 the specifications regarding the outdoor dataset CamVid and the indoor dataset Sun RGB-D as well as the evaluation metrics used to measure the performance of the proposed architecture AdvSegNet are presented. The CamVid and Sun RGB-D datasets are mentioned in Section 3.1.5 and Section 3.1.7, and the evaluation metrics in Section 3.1.8.

The methods on the subject of robust learning are presented in Section 3.2. Specifically, the architecture used for the experiments is detailed in Section 3.2.2. This is a combination of a convolutional network and a Gaussian process. The proposed training procedure for the architecture is discussed in Section 3.2.3. Then, the proposed loss functions used as part of the training as regularisers are presented in Section 3.2.4. These are Kullback-Leibler, Wasserstein distance and maximum correntropy. This is followed by Section 3.2.5 that overviews the evaluation metrics used for the experiments in Chapter 4. It discusses further in-depth how precision-recall and receiver operator characteristics curves are computed from confusion matrices. The section ends with details on the datasets used for the experiments for robust learning. This includes details on datasets MNIST, Fashion-MNIST, CIFAR-10 and CIFAR-100 in Section 3.2.6.

Finally, the methods for the continual learning area of the thesis are discussed in Section 3.3. Then, Section 3.3.2 discusses the changes and modifications used for the combined architecture of a convolutional network and a Gaussian process. The learning rate update rule for the architecture is discussed in Section 3.3.3. In this section, the two settings adopted for the combined architecture of the convolutional network and Gaussian process: a deterministic and a Bayesian setting. In the prior, all the weights of the convolutional network are set to scalar and only the parameters of the Gaussian process are updated. In the Bayesian setting, the weights of the convolutional networks are probabilistically distributed and both the learning rate of the convolutional network and Gaussian process are updated. The Sections 3.3.4 and Section 3.3.6 discuss the datasets used, the algorithm for updating the learning rates as well as the detail regarding the methods used for adding noise to the datasets. Then, finally, the chapter ends with the summary in Section 3.4.

Chapter 4 focuses on the experimental work for the proposed methods in Chapter 3. It begins with Section 4.2 which presents results for the proposed methods in segmentation and the discussion on these results in Section 4.2. Then, the results for the methods proposed in robust learning are presented in Section 4.3 and discussed in Section 4.4. Lastly, Section 4.5 presents results for methods proposed in continual learning and Section 4.6 discusses these results. The summary for Chapter 4 is provided in Section 4.7.

Chapter 5 outlines the main contributions, the limitations of the contributions as well as possible future directions and applications of the methods proposed in the thesis. Firstly, the limitations of the contributions are presented in Section 5.1. Secondly, Section 5.2 discusses the possible applications of the thesis and lastly, Section 5.3 discusses the future works.

1.5 Key Contributions and Dissemination

The contributions of this thesis to the scientific research community include:

1. The thesis demonstrates the importance of learning from uncertainty in the application of semantic segmentation with the SegNet architecture. Furthermore, there is the problem of applying deep learning systems in small-scale datasets (< 100 samples) where model uncertainty is substantial.

    - This thesis overcomes the issue by using Monte Carlo sample variance and an adversarial trainer in order to reduce model uncertainty by adapting the Bayesian

SegNet for outputting uncertainty maps that are further from a perfect solution of a 2D blank white image.

- The thesis proposes the solution of training Bayesian SegNet to learn to reduce uncertainty without any addition of extra training samples. The purpose is to reduce model uncertainty without using large datasets with samples greater than 1000.
- The thesis investigates the performance of the proposed model AdvSegNet compared to the non-Bayesian counterpart (the SegNet) that is more uncertain in its predictions.
- The thesis further tests the proposed model AdvSegNet with a custom attack that reduces its receptive field from 360x480x3 to 128x128x3. This is done to simulate an attack on the sensor that damages its resolution. The aim is to investigate whether the model can recover from the drop in performance over time.
- The thesis also provides a comparison of performance between the proposed model AdvSegNet as well as with the state-of-the-art DeepLab segmentation network as well as other versions of both SegNet and Bayesian SegNet.

The paper written based on this contribution was published as a conference paper in IEEE Sensor Data Fusion and was awarded the Best Paper Award. The publication details are listed below.

- Javed, M. and Mihaylova, L. (2019). Leveraging Uncertainty in Adversarial Learning to Improve Deep Learning Based Segmentation. In *Proceedings of the 2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pages 1-8, IEEE.

2. The second contribution of this thesis is in the area of robust learning as well as sensitivity to attack strength. The thesis deals with the problem of operating machine

learning and artificial applications in uncertain conditions. These involve erroneous predictions and vulnerability to adversarial attacks.

- The thesis overcomes this issue by proposing a combined model of a convolutional network and a Gaussian process. The model performance is validated both with gradual and abrupt uncertainties and is compared to the state-of-the-art approach i.e. the Monte Carlo dropout.
- The thesis evaluates the model on four types of datasets with increasing complexity: MNIST, Fashion-MNIST, CIFAR-10 and CIFAR-100. Furthermore, the thesis demonstrates that backpropagation of maximum likelihood loss regularised with similarity losses is vital for developing CNNs and DNNs with strong robustness against noisy images and white-box adversarial attacks.
- The results presented from the performance evaluation are further justified in the thesis by analysing the uncertainty of the GP classifier as a bar graph. The thesis then demonstrates that backpropagating the regularised maximum likelihood loss affects the reduction of uncertainty on the dataset attacked with Gaussian noise and motion blurring.
- The accuracy of the proposed solution is further analysed using precision-recall and receiver operating characteristics curves
- The proposed solution provides reliable uncertainty estimates and has an increased computational efficiency compared to the state-of-art Monte Carlo dropout approach. The validation is performed with increasing strength of the noisy images and white-box attacks
- The paper shows that explainable AI is linked to robust AI, such that robustness in AI can be achieved by accounting for uncertainty measures in both the model and datasets

The paper written based on this contribution was published as a journal paper in the International Journal of Machine Learning (IJMLC). The publication details are listed below.

- Javed, M. and Mihaylova, L., Bouaynaya, N. (2021). Robustness Analysis of Gaussian Process Convolutional Neural Network with Uncertainty Quantification. *In Proceedings of the International Journal of Machine Learning and Computing (IJMLC), IACSIT Press.*

3. The third contribution of the thesis is in the area of continual learning. The problem tackled in this situation is that modern deep learning systems are trained on datasets presented in one go. When the same network is made to train with another dataset, the past learning representations are overwritten. The network forgets previously learned tasks.

   - The thesis overcomes this issue by proposing a modified extension of the CNN-GP framework that is not only capable of characterising the impact on uncertainties but is also able to use the uncertainty values to update the learning rate of the GP.
   - The thesis proposes a single-classifier (i.e. the GP) and compares its performance with the multi-classifier in the continual learning application known as UCB. The thesis evaluates this on two datasets: sequential-MNIST and permuted-MNIST.
   - The thesis further tests with two separate configurations of the CNN-GP framework: the deterministic and the Bayesian setting. In the deterministic setting, the weights of the CNN component in the CNN-GP framework are kept constant while only the weights of the GP are updated. In the Bayesian setting, the weights of the CNN are replaced with prior Gaussian mixture and both the weights of the CNN and the GP are updated. The thesis further investigates the efficiency of using either the deterministic or the Bayesian weights setting.

The paper written based on this contribution was published as a conference paper in proceedings of the International Conference of Information Fusion (FUSION'21). The publication details are listed below.

- Javed, M. and Mihaylova, L., Bouaynaya, N. (2021). Variance Guided Continual Learning in a Convolutional Neural Network Gaussian Process Single Classifier Approach for Multiple Tasks in Noisy Images. In Proceedings of the International Conference of Information Fusion, South Africa, pages 1-8, IEEE.

Chapter 2

Literature Review

Chapter Overview

- Provide the background knowledge in pixel-wise labelling in semantic segmentation, all as operating as part of scene understanding
- Introduce to the Bayesian SegNet architecture and investigate the use of uncertainty quantification in the Bayesian SegNet
- Introduce to the notion behind generator adversarial neural networks and discuss how adversarial learning can be used to leverage uncertainty quantification
- Discussion on the role of robustness in artificial intelligence.
- Discussing the field of adversarial attacks, including discussion on uncertainty and meta-learning based methods in adversarial defense.
- Relating learning from noisy labels to meta-learning.
- Introducing the field of continual, including discussion on catastrophic forgetting and sequential task incremental learning.
- Literature survey of different methods in variance guided continual learning and other regularisation-based approaches.

2.1 Background Knowledge

It has long been a common goal in scientific research to model natural systems. However, natural processes are very complex. Their complexity is attributed to the interdependencies within their interacting sub-systems. Modelling learning is an attribute of modelling natural processes. Consequently, it is a challenging task. The human brain is one example of a learning hub. Deep learning attempts to model the brain's learning capabilities to a certain extent. This section aims to introduce the field of deep learning which is a sub-topic in machine learning. It also discusses the difference between the two approaches, some of the recent architectures in deep learning research, the vulnerabilities of deep learning to adversarial attacks and introducing the popular white-box attack that is the fast gradient sign method.

2.1.1 Data Driven Algorithms: Machine and Deep Learning

This section aims to introduce the key concepts behind data-driven learning algorithms. Data-driven algorithms in statistical learning aim to understand the underlying correlations and statistical structures within the input data. These properties are often latent and interdependent [58], [153]. If the representation of the model is weak and/or not complex enough, then such features are missed during the learning process. This can result in an incorrect representation of the data. Deep neural networks (DNNs) are one example of data-driven algorithms [5], [12], [58], [153].

It is often assumed that the data can take any form or shape [5]. This also helps to differentiate the definitions of machine learning and statistics. Even though they are considered to be closely related, their definition digresses from the point of view of data. As the machine learning community is very little interested in modelling the data generation process, the field of statistics is. Nevertheless, the end goal of both fields remains the same. That goal is to ensure that the learning process efficiently captures the underlying features within the data. Furthermore, it should be capable of accurately predicting in the presence of uncertainty, noise and adversarial attacks.

When a learning algorithm is capable of changing its states to respond correctly to data, it is then said to have learned the correct representation of the data [5]. This is evaluated using a loss or an objective function. An objective function defines the cost required to optimize the model outputs (predicted labels) to the desired output (ground-truth labels) [5].

The process of learning is noisy and uncertain. This argument is well-supported in the recent findings in neuroscience where researchers have claimed that the animal brain adjusts learning strategies based on the level of uncertainty (see Introduction section of [109]). However, the field itself is diverse. There are many types of statistical learning methods. However, the focus of this thesis is specifically on deep learning methods. Next, the difference between deep learning and machine learning methods are discussed.

The main aim of artificial intelligence is to develop intelligent software. Specifically, AI applications tend to solve tasks that human beings are easily capable to do but cannot define such problems mathematically. Machine learning methods allow computers to learn from experience, gather concepts knowledge from them and in the process, be able to extract input from raw data [48]. Hence, learning from representation is a fundamental aspect of all machine learning algorithms, including deep learning.

What makes deep learning more attractive from the point of view of current AI applications as opposed to other machine learning methods is that these representations are rich. Traditional machine learning methods cannot learn these representations to the level of depth and flexibility as deep learning methods.

DNNs learn representations from inputs in a compositional manner. Specifically, high-level representations can be explained by simpler low-level representations. This hierarchical style of learning allows DNNs to build complex concepts out of simpler ones, allowing learning representations in a much more rich and flexible manner compared to other machine learning methods.

An example can be in the form of classifying images of cats and dogs. Here, DNNs can learn to join simpler features such as edges, blobs and corners to learn high-level features such as whiskers of cats. Equipped with this knowledge, in the next section it is seen how typical

DNN architectures are not suitable for image classification, but rather the convolutional family of architectures are suitable for the problem of image classification.

2.1.2 Review of Traditional Convolutional Network Architecture: LeNet

This section explores the concepts behind convolutional neural networks. These are specific types of neural networks that are used in image classification tasks. Classifying images into separate categories is a challenging task. Some of the challenges include viewpoint variation, scale variation, deformations, occlusion, high illumination, background clutter, intra-class variation, noise corruption and motion blurring [92].

Typical image classification datasets researched in the field of deep learning [84], [90] contain large number of pixels. These can amount to millions if the image dimensions are for example in high-definition (i.e. 1280x720). If one was to adopt a standard DNN approach with, for example, having an architecture with the first fully connected layer with 100 units. It will follow that these layers alone will contain approximately several, tens of thousands of weights. This is without considering the rest of the architecture.

As a result of this application, the capacity of the system would increase substantially and so will the demand for more training samples. Storing this information will also consume a lot of memory. Additionally, standard DNNs cannot encode translations of objects in an image and neither any local distortions. Hence most of the datasets used to train standard DNNs such as [90], [84] are pre-processed in the form that they are size-normalized, as well as forced to the centre of the image. In the author's opinion, this cannot always be possible for all datasets since the real-world scenes have objects that the majority of the time are varying in size, angle and position.

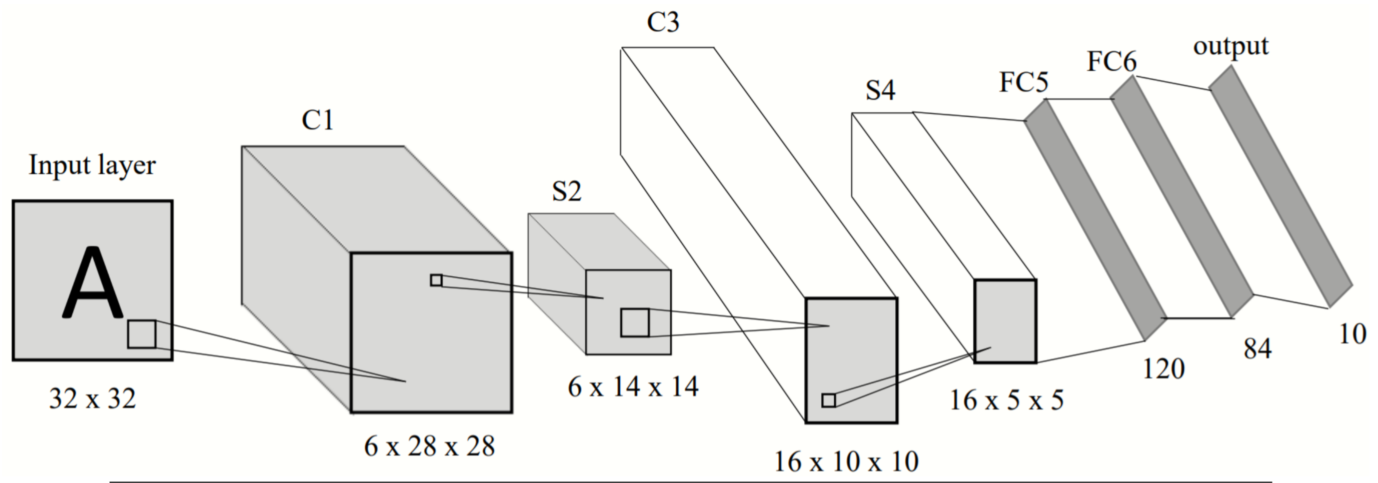
An important deficiency in standard DNN architectures also comes from the fully connected layers that are unable to capture the 2D local structures and local correlations of pixels. An image presented in a given order would be classified similarly if presented again, once the order is changed over time. This is contrasting to the structure of image datasets.

Pixels in an image dataset have a strong 2D local structure with a grid-like topology. Unfortunately, the fully connected layers ignore any form of topology in the input images [89].

Given the expensive computational cost and lack of local feature capturing in standard DNNs, these architectures are not ideally used for image classification. The research in deep learning for image classification inclines more towards the convolutional neural networks (CNNs). This particular family of DNNs are capable of capturing the aforementioned features much more easily. This is partly due to their architecture design which is functionally closer to how neurons operate in the visual cortex.

The most standard type of convolutional network is LeNet-5 [89], the very first convolutional neural network. This is studied in this section since the layer configuration adopted in this architecture is inspired by most modern convolutional architectures. The LeNet-5 architecture comprises six computational layers in total as shown in Figure 2.1. LeNet-5 is organized into an input layer, two convolutional layers, two fully connected layers and an output layer. The next paragraph will discuss each of these layers individually.

Fig. 2.1 The LeNet architecture consists of a total of six computational layers. S1, S2 and S3 layers represent the subsampling layers while C1, C2 and C3 represent the convolutional layers. F5 and F6 are the fully connected layers. Image inspired from [89].



The hidden layers in LeNet-5 are responsible for computing the convolution between the respective kernels and the image. Convolution is a mathematical operator that expresses the *amount of overlap* [47] of one function g as it is made to shift over an alternate function f.

The operator is shown in equation (2.1)

$$f(t) * g(t) = \int_0^{\tau} f(\tau)g(t-\tau)d\tau, \tag{2.1}$$

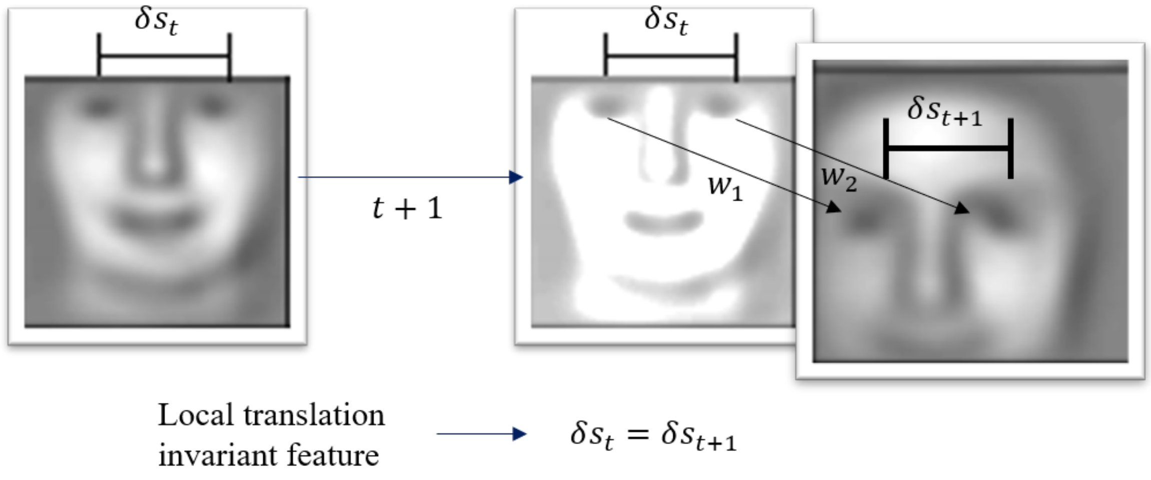
where, f and g are the two functions that are convolved, the symbol '*' represents the convolutional operator [47]. Both functions are time dependent. This is, however, not true in the case of networks since the input comes in the form of an image which is a multi-dimensional array (often termed *tensor*) [48]. These arrays capture the spatial correlations in the input.

The reason for the difference in terminology is that the expression given above is a standard version of the convolution operator. This is used more often in the field of signal processing. Here, the variable 'τ' represents the decay constant of a signal. An example of this can be a decaying voltage signal. For a more in-depth review, readers can refer to "Fast Convolution and Filtering" (Section 8) of [159] for a more signal processing perspective.

Conversely in the field of computer vision, the same operator is used to 'blur' and detect edges in an image [47]. It is defined in (2.2), where a discretized version of (2.1) is used. This is because a majority of the algorithms in deep learning are performed on digital computers that perform calculations on time steps. Convolutions are usually performed on 2D images. Hence the operator has two dimensions. The convolutional filter is also referred to as a kernel in deep learning research [48], [159]. The notation of the elements is as follows; if one is to compute the i^{th} row and the j^{th} column element of the feature matrix S, then the m^{th} row and n^{th} column element of the kernel would be multiplied by the $(i+m-1)^{th}$ row and the $(j+n-1)^{th}$ column element of the image matrix. This would be summed across the entire width (M) and height (N) of the kernel. The size of the feature matrix S would be equal to the size of the image matrix minus the size of the kernel matrix. This is then summed with one for both height and width dimensions respectively. The equation is shown in (2.2)

$$S(i,j) = (I * K)(i,j) = \sum_{m}^{M}\sum_{n}^{N} I(i+m-1, k+n-1)K(m,n). \tag{2.2}$$

Fig. 2.2 An example of local translation invariant feature in the form of distance between the eyes measurement. The distance between the eyes at timestep t is shown as $\delta s - t$. The distance at $t+1$ is shown as δs_{t+1}. Image reprinted from [93].


δs_t
δs_t
δs_{t+1}
$t+1$
w_1
w_2
Local translation invariant feature
$\delta s_t = \delta s_{t+1}$

Convolutional layers are responsible for the feature decomposition of the input images. Feature learning can also be misguided. An example of this can be when redundant features are learned during the training process. A local translation invariant feature is one such example [92]. These are types of features that do not alter in the feature description w.r.t their local axis. Even when translated ever so slightly. The below paragraph discusses this aspect in detail.

When considering, for example, the case of face recognition, the distance between the eyes is a type of a local translation invariant feature. The example is illustrated in Figure 2.2. The feature map represents a higher-order feature that consists of a human face presented at time step t. The distance between the eyes, represented as δs_t, is one of the local features in the map. As the time step increases, the image is translated such that the eyes are shifted by vectors w_1 and w_2. The distance feature, however, remains the same i.e. $\delta s_{t+1} = \delta s_t$. This occurs if the assumption that only the translation shift is applied. If the inputs were to be scaled, the assumption would break. Consequently, the features would not be local translation invariant. The figure does apply very little scaling as part of the augmentation process, but it is negligible. Hence, the assumption partially holds.

Pooling (or Subsampling) Layers

The key point for mentioning the aforementioned features is for discussing the concept behind subsampling layers in LeNet-5. The layers are also referred to as the pooling layers in DNN literature [48], [5]. The logic behind placing these layers after activated convolutional layers are to prevent the learning of redundant features. With this form of dimensionality reduction, one can drop feature size by aggregating features in a rectangular non-overlapping window. Then, the reduced features can be applied differently between max-pooling [140] and average pooling [98]. In one type, the aggregated elements are averaged within the window region. On the other, the maximum element from the window region is taken. The difference between the approaches is also illustrated in Figure 2.3.

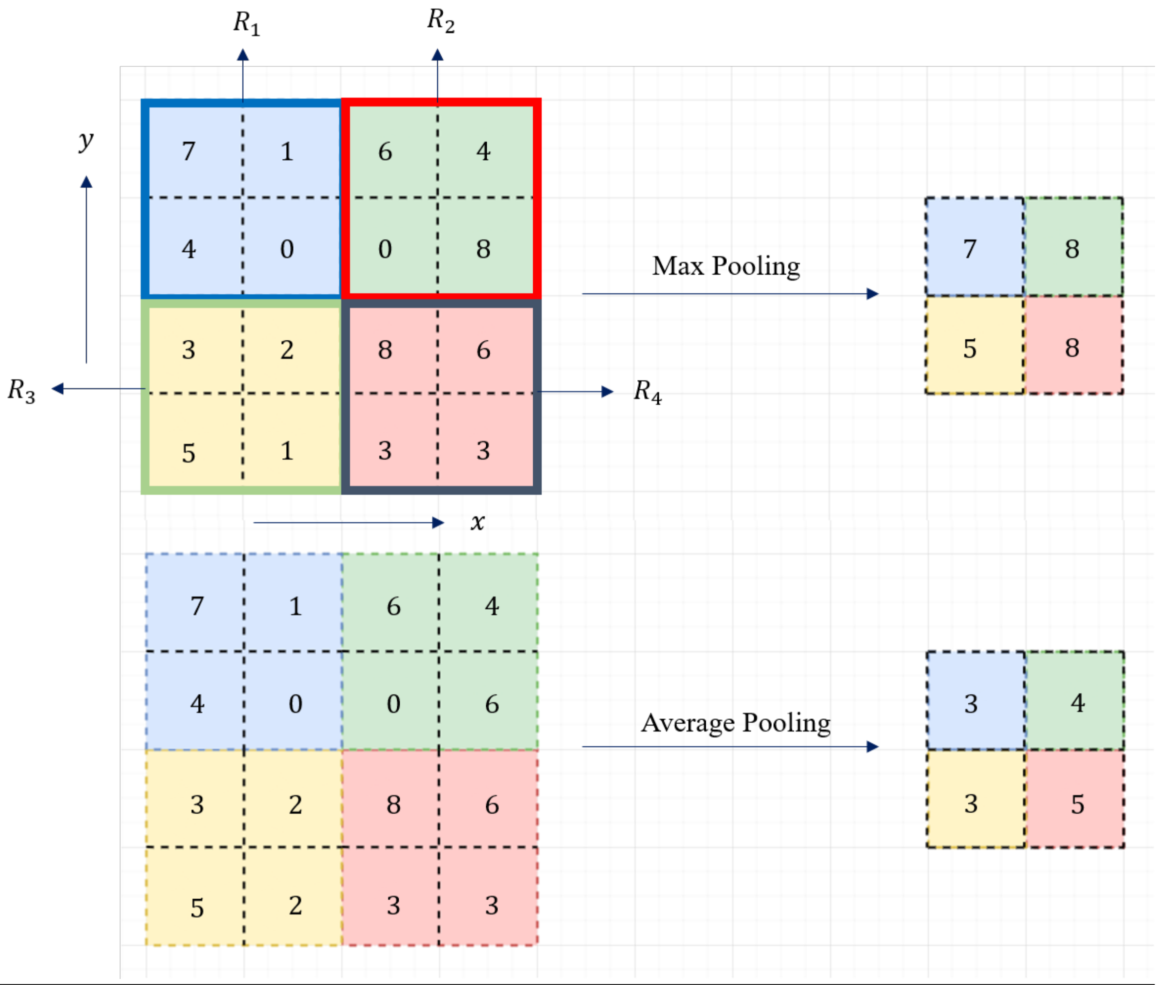
Max-Pooling Vs Average Layers

Consider an example of a feature layer l that is split into non-overlapping window regions. A single j^{th} window region is in the vector space R^l_j. Each window contains the pixel elements a^l_j which range from 1 to n. For example, in Figure 2.3, the examples have a total of $j = 4$ windows, shown as $R_1, \cdots, R_4$. Each window as four elements $a_1, \cdots, a_4$ (e.g. R_1 has 7,1,4 and 0). For brevity, the example does not index the windows according to the feature layer number l. This can change from one architecture to another. Then, the pooling operations can be formalized collectively as shown in equation (2.3)

$$\begin{gathered} a^{l+1}_j = pool(a^l_1, \cdots, a^l_i, \cdots, a^l_n) \text{ where } i \in R^l_j, \\ \text{if } pool = mean(.) \text{ then } a^{l+1}_j = \frac{1}{n}\sum_{i=1}^{n}(a^l_1, \cdots, a^l_n), \\ \text{if } pool = max(.) \text{ then } a^{l+1}_j = max(a^l_1, \cdots, a^l_i, \cdots, a^l_n). \end{gathered} \tag{2.3}$$

In the mean configuration, the elements of the region are averaged according to their total number n. In the maximum setting, the element with the highest pixel value is considered. Both pooling settings provide dimensionality reduction and are especially useful for large scale architectures, e.g. with more than sixteen convolutional layers. A further empirical study is considered in [154] where it is proven that max-pooling has overall better performance

Fig. 2.3 A simple example that considers two popular types of pooling schemes. The top half represents the 'max pooling' scheme. The bottom half for the 'average pooling' scheme. Four rectangular regions partition the feature matrix. These are represented as R_1, R_2 (first two) and R_3, R_4 (lower two).



advantages than average pooling. The study conducted experiments based on pooling configurations on the NORB dataset [91]. They found that max-pooling gives more robustness on local features translation than average pooling with about 4.57% on normalized NORB and about 5.6% on jittered cluster version of NORB. [154].

Batchnormalization & Dropout Layers

The final two layers that are frequently used in modern DNN architectures are batch normalisation [70] and dropout [172]. The purpose of batch normalisation layers is to improve the training process by standardising the feature layers batch-wise. The purpose of dropout is to regularise the training process and prevent overfitting by simply shutting down the activations of random nodes (along with their connections) during each forward pass.

Batch normalisation deals with the issue of *internal covariate shift* while dropout deals with *overfitting*. One way to imagine the internal covariate shift is to think of a target distribution that is continuously $shifting$ its *covariance* (how much it spreads). The target distribution describes the optimal configuration of weights. If this is evolving after every weight update step, then the weights will be continuously updating to meet the requirements posed by this distribution. This would be akin to tracking a continuously moving target. This will slow down the training process. Batch normalization aggregates the nodal activation outputs of previous layers (a.k.a batch statistics) and standardises it to zero mean and a unit standard deviation. As a result, the optimization process is considerably accelerated. Batch normalization can also be interpreted as a technique to reduce noisy updates of weights based on the gradient information.

Dropout layers, on the other hand, deal with overfitting. Overfitting occurs when the number of parameters is higher than what is required to explain the data [12]. However, it is mentioned in the work of [70] that using batch normalisation layers can also assist in regularisation. Avoiding the need to use dropout layers.

In summary, this section reviews the various types of layers used in a standard CNN architecture. These included convolutional layers, max and average pooling layers, batch normalisation and pooling layers. It is seen that convolutional layers are responsible for the hierarchical decomposition of features, pooling for dimensionality reduction, batch normalisation for standardising and improving training and dropout for regularisation. Two more layers have not been mentioned in this section, fully connected and softmax layers. These will be discussed in Chapter 2. In the next section, the vulnerabilities of DNNs and CNNs are explored. Specifically, how adversarial attacks can be used to fool DNNs and CNNs into outputting erroneous predictions.

2.1.3 Adversarial Attacks

Modern approaches in deep learning show that the performance of DNN and CNN architectures on image classification tasks is on par with human-level recognition [176]. However, it also has been proven in research [175], [49], [128], [115], [173] that if such architectures are presented with examples that perturb the inputs imperceptibly, i.e. very close to the dataset distribution, then these examples would be classified incorrectly with high confidence. The question of course remains how can these attacks be generated?

In adversarial attack literature, a portion of works exploit gradient information from the loss functions to update input images instead of network parameters [49], [175]. A popular example of this is the fast gradient sign method (FGSM) [49] discussed in Section 2.1.5. Another portion of work instead computes directly the mapping between the input and the output. Specifically, the work from [128] computes the forward derivatives of the input-output mapping with respect to the inputs. Using the input and the gradient information, the attacker can, to an extent, control the attack strength based on the type of perturbation. This method of generating adversarial attacks is defined in Definition 1, taken from [128].

Definition 1 *Suppose that a DNN model can be defined as a function that maps a raw input vector to an output vector, such that $F : X \rightarrow Y$. Then an adversarial example can be considered as a sample X^* that is obtained once X is perturbed by a perturbation vector δ_X. Then hence the problem of optimization this attack involves minimising its strength so that it is undetectable by the DNN*

$$\begin{gathered} argmin_{\delta_X} \|\delta_X\| \ s.t \ F(X+\delta_X) = Y^*, \\ where \ \ X^* = X + \delta_X. \end{gathered} \tag{2.4}$$

Here, X^* is referred to as the adversarial example and Y^* is the desired adversarial output.

However, there are disadvantages to using this approach to generate attacks. This is because solving this optimization problem is difficult as the above optimisation problem is non-convex (i.e. global optimal not guaranteed). Hence, there is a possibility that the attack may be detected by the DNN. However, this is not the only way to generate adversarial attacks. In the next sub-section, the reason behind the vulnerability of DNNs to adversarial attacks is discussed.

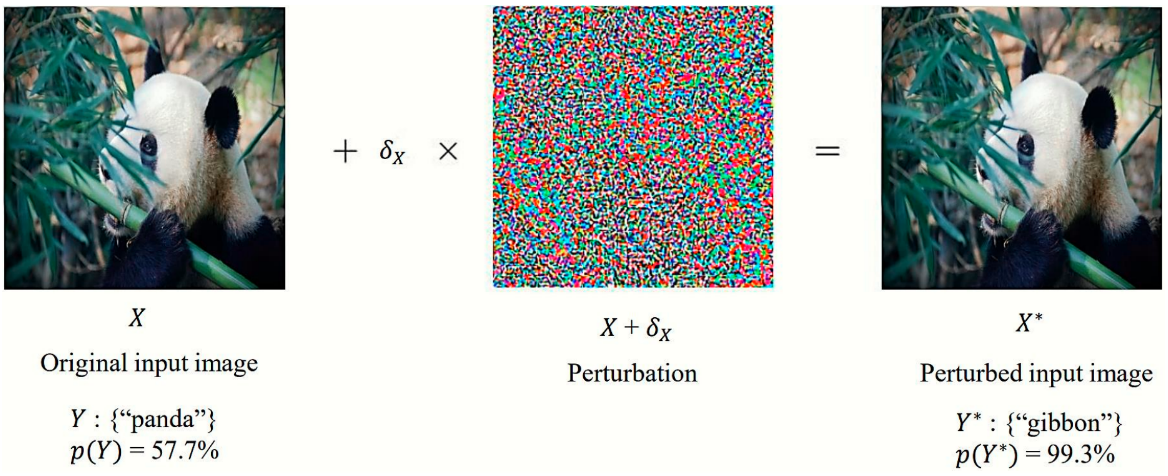
Vulnerability of Deep Neural Networks to Adversarial Attacks

In literature, there are two general explanations for why DNNs and CNNs are vulnerable to adversarial attacks: the non-linear view in the work of [175] and the linear view in the work of [49]. The linear view is more relevant to the proposed methods in this research. It is discussed separately in Section 2.1.4.

The non-linearity review states that the vulnerability of DNNs to adversarial attacks is a result of DNNs being too expressive. The more expressive the model, the better it can represent data. DNNs with deep architectures are more expressive. It is observed that the deeper the DNN architecture, the more expressive the DNN model [139].

Additionally, the non-linear view explores various counter-intuitive properties that arise during learning in DNNs. One example is that the semantic meaning of the activation (or features) of an arbitrary layer is unchanged whether projected from its natural domain or a random neighbouring domain. This suggests that, given an input image, DNNs do not learn the features entirely, but rather a subset of the input distribution. They are unable to differentiate features in the neighbouring vicinity of the natural domain of the feature vector. It is later seen in Chapter 3 how this property inspires the technique of swapping images with

Fig. 2.4 An example of an image X perturbed by fast gradient sign method attack with perturbation vector δ_X. The original image is classified with the category "panda" with confidence 57.7%. It is then perturbed by the vector δ_X. The resultant perturbed image X^* is classified incorrectly as 'gibbon' with high confidence of 99.3%. Figure inspired from [49]. The image of the panda is reprinted from [134]



nearest neighbours can be used to teach DNNs to learn the features space fully. Visualising an example of an adversarial perturbation is discussed next.

Adversarial Perturbation Example

An example of an adversarial perturbation of an input image can be illustrated as shown in Figure 2.4, taken from [49]. Here the original input image X can be seen being classified into the correct category of Y : "*panda*". Furthermore, the output is given with a less confident prediction of $p(Y) = 57.7\%$. Then, the image is perturbed by $X + \delta_X$ via the perturbation vector δ_X shown in Figure 2.4. This vector carries the values based on the sign of the gradient of the loss function w.r.t the inputs. The *sign* function resembles the hyperbolic tangent, except it is more constricted in the center region while the hyperbolic tangent is more curve-like in the center region. Then, it can be seen that the resulting image X^* is almost imperceptible to the human eye. The *sign* function is shown as

$$sign(x) = \begin{cases} -1, & \text{if } x < 0 \\ 0, & \text{if } x = 0 \\ 1, & \text{if } x > 0 \end{cases} \cdot \tag{2.5}$$

Despite the perturbations, human perception would still classify the adversarial example as a panda. However, it is seen that the network prediction has been completely altered into believing that the adversarial example consists of a gibbon i.e Y^* : "*gibbon*". On top of that it is classified with a high confidence of $p(Y^*) = 99.3\%$. This is prediction is erroneous.

Though the above definition points to the non-linearity property of the DNNs that is the cause of the highly-confident, erroneous predictions, research has shown that even the linear property of DNNs has a contributing factor to this problem [49]. Section 2.1.4 considers this linear explanation in-depth. The next section formalizes the strength of adversarial attacks.

2.1.4 Linear Growth of Adversarially Perturbed Activations

The linear nature of learning in DNNs and CNNs is considered to be one of the many explanations behind their vulnerability to adversarial inputs. Section 2.1.3 highlighted the weaknesses of DNNs and CNNs. It is seen that examples that are imperceptibly different from the original input images are enough to cause DNNs and CNNs to be fooled. Making them erroneously predict with high confidence. This section explores in-depth the possible explanations as to why such behaviours in DNNs and CNNs exist. In particular, discussing the works of [49] that interprets this problem from the perspective of the linear properties of DNN and CNN architectures.

Definition 2 *Suppose that a DNN model is defined to have weights in the form of a vector w holding weight information on all nodes. This model is then presented by an input image or a feature matrix X. Then, the linear growth of the change in activation $w^T\delta_X$ perturbed by δ_X is related to the adversarial output X^* activated by w^T via the sum relationship of the dot products of the weight vector w^T with the input matrix X and the perturbation vector δ_X. The dimensions can be arbitrarily defined in this definition*

$$w^T X^* = w^T X + w^T \delta_X. \tag{2.6}$$

Given the above definition of this growth, it is then possible to acknowledge the important role of linearity in the vulnerability of DNN and CNN architectures. Given that the norm constraint $\|\delta_X\|$ does not grow with the dimensionality of the problem, it follows that even a small change in activations perturbed by δ_X can grow linearly with the dimensions of the vector w. This would mean that infinitesimal changes to the input image matrix X will amplify the linear growth of the activations. In doing so, the output predictions will be incorrect and with a high confidence value ($\approx> 90\%$).

The advantage of this linear view of the vulnerability of DNNs is that it allows easy and cheap generation of adversarial examples that can be quickly be generated for testing the desired CNN architecture. Additionally, they can be used to train architectures on adversarial attacks. One such example is the FGSM [49], which forms the subject of the next section.

2.1.5 Fast Gradient Sign Method

In the previous section, it was discussed that the linearity view of the vulnerability of DNNs to adversarial examples allows simple, infinitesimally changes in activations to build up with the weight configurations and lead to erroneous, highly confident predictions. In this section, the previous definition is extended to accommodate techniques like FGSM, as means of generating adversarial attacks.

Relating Linear View of Vulnerability in Deep Neural Networks and Fast Gradient Sign Method

The linear view of adversarial examples helps in constructing a fast, cheap and simple method of generating such instances. This is important since the deployment of artificial intelligence systems require that they are reliable in their predictions, especially in mission-critical systems such as self-driving cars. Generating adversarial examples is a good way of assessing models as well as assisting them to tackle malicious inputs. One such example of an attack is the FGSM.

Considering a DNN or a CNN model that receives the input image matrix X with arbitrary dimensions and outputs the prediction as a vector Y. The desired labels take the form of the vector $\bar{Y}$. Given the parameters of this DNN model are considered as θ, the loss function used for training this model can be denoted as the function $J(\theta, X, Y)$ in equation (2.7)

$$\delta_X = \varepsilon \; sign(\nabla_X J(\theta, X, \bar{Y})). \tag{2.7}$$

The FGSM can be then used to output an adversarial noise by maximising the gradient of the loss function. Here, δ_X refers to the perturbation vector. The adversarial input X^* can then be computed by simply adding the original input with this perturbation vector i.e. $X^* = X + \delta_X$. The illustration in Figure 2.4 in Section 2.1.3 is an example of an FGSM attack.

In summary, the past section has so far introduced the field of deep learning for computer vision. It is seen how DNNs have advanced the field of computer vision. However, DNNs have limitations in dealing with adversarial attacks. The literature considers the explanation behind this in both a linear view and a non-linear view. The non-linear view studies the vulnerability of DNNs to adversarial attacks from the point of view of expressiveness and the number of non-linear layers in DNN architectures. The linear view studies the growth of adversarial perturbations as a result of minute changes to the activation layers in DNN architectures. Apart from adversarial attacks, DNNs are also unable to operate in uncertain environments and leave no trace of whether they are uncertain in their predictions or confident. One way to tackle this is to have DNNs account for their predictions using confidence measures (or error bars). The next section considers the Bayesian treatment of DNNs that allows DNNs to account for uncertainty in their predictions.

2.1.6 Review of Bayesian Methods in Deep Learning

Deep learning as a field has become widely popular in majority of real-life applications such as medical imaging [174], [186] and autonomous driving [141]. Despite the widespread adoption of CNN and DNN architectures, the performance of deep learning frameworks tends to drop when operating in highly uncertain environments [79]. This section discusses the recent surge of Bayesian methods being adopted in deep learning systems as well as providing an introduction to the field of Bayesian deep learning.

Limitations of Deep Neural Networks

The majority of times in literature [85], [59], [192], the high performance of deep neural networks in tasks such as object detection and image classification is blindly taken to be exact and accurate. To highlight a few examples, there have been two specific incidents that have exposed the weakness of DNNs in an uncertain environment. The first is a fatality that occurred in an accident involving a self-driving car, as reported in [179]. This occurred as a result of a direct influence from the false perception in the classifier algorithm within the self-driving car which classified the white side of the incoming trailer to the white sky category. Furthermore, [72] reports another incident involving a classifier that considered a human couple as animals, leading to a racial discrimination concern.

Importance of Modelling Uncertainty

From the aforementioned examples, it is evident that a DNN system needs to do more than simply achieve high accuracy in its predictions. Ideally, it should also be capable of explaining its predictions. It should be able to explain questions like why it has failed ? which category does it struggle to recognise the most ?. In literature, one way to accomplish this is to model uncertainty.

Modelling uncertainty allows AI systems to express confidence in their predictions. This is important since AI systems will be ubiquitous in the future. The number of AI-powered systems will increase as AI technology beings to be predominant in major sectors such as education, military, vehicle systems, transport and medical applications.

Ethical Perspective of Decision Making in Artificial Intelligence Applications

The question in the use of AI systems will not only depend on the accuracy but also the trust in the system, especially when considering the ethics in medical areas where patients might question or even doubt whether a human should perform their operation or a robot. In these scenarios, it would be more useful for AI systems to report on circumstances where they have doubts or difficulty in making decisions so that the human observer may take control in this situation or the system may try to generate several possibilities and decide the best course of action.

Bayesian Methods and Uncertainty Quantification

Bayesian methods formalize the quantification of uncertainty. It is a probabilistic paradigm that has been studied for many years in the area of statistics as a primary method behind many predictive stochastic models for applications such as weather forecasting, stock market exchange, banking business and recommendation systems. What differentiates these applications from other examples in image classification or face recognition is that these make intensive use of uncertainty quantification. Bayesian methods quantify uncertainty via the variance information during inference. Here, inference refers to the process of using a test sample from a data to *query* a statistical model by inferring the properties of its posterior distribution conditioned on the test sample [13].

Relating Bayesian Methods to Interpretable Artificial Intelligence

Using this value, one can infer the model's learning criteria and reveal whether the task has been learned or not. More importantly how it has been learned can be revealed. This is impossible in the traditional deep learning setting where the models studied are usually black-box. This means that they do not transparently express how they have learned the problem.

On a side note, exposing these black-box models is studied separately in the area of deep learning known as *interpretable AI* [31], [108], [18]. Bayesian methods, on the other hand, can express the certainty of a model's prediction of a solution to a problem. In doing so, they expose the underlying learned features in the model. However, such methods require formal definitions of probabilistic quantities. Though this section introduces these informally, the next sections go in-depth in discussing the formal half of uncertainty quantification in Bayesian methods in deep learning. Interpretability of AI applications is discussed further in Chapter 3.

2.1.7 Bayesian Methods for Probabilistic Modelling

This section formalizes the discussion on uncertainty quantification in Bayesian deep learning. It relies heavily on probabilistic models in Bayesian theory that endows the field of deep learning to develop tools for expressing uncertainty in predictions. The section begins by formalizing the Bayesian theorem, defining the inference stage, then later discussing the challenges in posterior computation in Bayesian methods.

Given a dataset of size N that consists of a set of training inputs $X = \{x_1, \cdots, x_N\}$ and a set of ground truth labels $Y = \{y_1, \cdots, y_N\}$, the function defining the relationship between the inputs and outputs can be shown as $y = \mathbf{f}^{\theta}(x_i)$ where $i \in \{1, \cdots, N\}$. This is a common regression case. Here, the parameters of the function are denoted as θ. For example if a DNN has L layers then $\theta = \{\theta_1, \cdots, \theta_L\}$, assuming there is one node in each layer. The Bayesian theorem can be formalized as shown in equation (2.8)

$$p(\theta|X,Y) = \frac{p(Y|X,\theta)p(\theta)}{p(Y|X) = \int_{\theta} p(Y|X,\theta)p(\theta)d\theta}. \tag{2.8}$$

The term $p(\theta|X,Y)$ refers to the posterior distribution, $p(Y|X,\theta)$ denotes the observation likelihood, $p(\theta)$ is the prior distribution and $p(Y|X)$ is the marginal (a.k.a the normalizer or the model evidence [13]). The prior distribution $p(\theta)$ captures beliefs regarding the parameter configurations that have generated the dataset before the observation of the data takes place. At the point of observing new data, the prior information is updated according to the likelihood of how the parameters are distributed given this observation (i.e. $p(\theta|X,Y)$). This is modelled by the observation likelihood which is chosen by the user according to the task. For example, for regression a Gaussian likelihood is used, for classification, the outputs of the Gaussian likelihood is passed through a softmax function [143]). The posterior distribution describes the most probable parameters given the observed data. This notion is opposite to observation likelihood. The above can also be simplified informally to equation (2.9)

$$posterior = \frac{observation\ likelihood \text{ x } prior}{marginaliser}. \tag{2.9}$$

During inference, a new observation point x^* is introduced and the inference stages aim to evaluate the posterior accordingly. The posterior update involves the integration of the new observation likelihood $p(y^*|x^*,\theta)$ and the former posterior $p(\theta|X,Y)$ with respect to the parameters. This is shown in the equation (2.10)

$$\underbrace{p(y^*|x^*,X,Y)}_{prediction} = \int \underbrace{p(y^*|x^*,\theta)}_{new\ observation} \underbrace{p(\theta|X,Y)p(\theta)}_{posterior\ times\ prior} \overbrace{d\theta}^{integrated\ over\ entire\ hypothesis\ space} . \quad (2.10)$$

In Bayesian statistics literature [148], the above integration is formally referred to as the marginalisation. The above expression does have an analytical solution but only for simple models. For more complex models, such as large neural network architectures [59], [85], the above computation is intractable. This means that closed-form solutions for posterior distribution from such models are difficult to compute.

However, research has shown [52] that even for simple models such as radial basis networks, DNNs that use squared exponential kernels (see Section 2.1.15) as activation functions [9], the posterior computation is intractable.

2.1.8 Variational Inference: Casting the Problem of Marginalisation for Posterior Approximation as a Simpler Optimisation Problem

The previous section mentioned that evaluation of the true posterior distribution for complex models is intractable. As a result approximation methods are adopted to provide a close estimate to the posterior. One of the popular approximation methods is known as variational inference (VI) [52].

Variational inference methods approximate the posterior distribution $p(\theta|X,Y)$ with a simpler parameterized distribution. The variational parameters can be optimised by minimising the Kullback-Leibler divergence between the approximate and the true posterior. This way, variational inference casts the approximation problem in the form of optimization [52]. As a result, Bayesian methods using variational inference are easier to optimize and scale well with both training set and architecture size. Furthermore, this approach can be easily applied in popular deep machine learning libraries like Tensorflow [1] and Torch [131].

In variational inference methods, a separate approximating distribution is considered. This distribution is termed variational distribution (a.k.a variational posterior) $q_\psi(\theta)$ and is parameterised by ψ, also known as the variational parameters [143]. This distribution is selected so that it is easier to evaluate than the true posterior and is tractable.

Tractable means that during inference, when a new data point x^* is introduced, computing the predicted variational posterior $q_\psi(y^*|x^*,X,Y)$ by integrating the new observation $q_\psi(y^*|x^*,\theta)$ with variational posterior $q_\psi(\theta|X,Y)$ has an analytical (exact) solution.

During the optimisation part in variational inference, the parameters of the variational posterior ψ are altered to make the variational distribution match closely to the true posterior $p(\theta|X,Y)$ by reducing the difference between the true posterior and the variational distribution. This *difference* quantity refers to as the Kullack-Leibler divergence (KLD) [87]. The equation is shown in (2.11)

$$KL(q_\psi(\theta)\|p(\theta|X,Y)) = \int q_\psi(\theta)\log\frac{q_\psi(\theta)}{p(\theta|X,Y)}d\theta. \tag{2.11}$$

Here, the symbol for KLD is KL. Hence, minimizing the Kullback-Leibler divergence brings the variational posterior closer and closer to the true posterior. The inference stage of variational approximation is shown in equation (2.12)

$$\underbrace{p(y^*|x^*,X,Y)}_{real\ prediction} \approx \int \underbrace{p(y^*|x^*,\theta)}_{new\ observation} \overbrace{q_\psi(\theta)}^{variational\ posterior} d\theta =: \underbrace{q_\psi(y^*|x^*,X,Y)}_{approximate\ prediction}. \tag{2.12}$$

This term is often re-written [148] in the format of a loss function as shown in equation (2.13). Here, the idea is that the minimization of the KLD between the true posterior and the variational posterior is mathematically equivalent to maximising the model evidence (observation likelihood) with respect to the change in the parameters $d\theta$. This form is also known as the evidence lower bound, or ELBO [13], as the difference between the variational posterior and the KLD between the variational posterior and the prior is strictly bounded by the log of observation likelihood $\log p(Y|X)$. This formal representation is commonly used to describe variational inference loss or ELBO in (2.13). It comprises two terms: the integral of the product of the maximum likelihood and the variational posterior and the KLD between the variational and the true posterior

$$\mathscr{L}_{ELBO} := \int q_{\psi}(\theta) \underbrace{\log p(Y|X,\theta)}_{log-likelihood} d\theta - \underbrace{KL(q_{\psi}(\theta)\|p(\theta|X,Y))}_{KLD\ term} \leq \log p(Y|X), \tag{2.13}$$

$$\psi^{*} = argmin \underbrace{KL[q_{\psi}(\theta)\|p(\theta|X,Y)]}_{regularisation} - argmax \underbrace{\mathbb{E}_{q_{\psi}(\theta)}[\log p(Y|X,\theta)]}_{expected\ observation\ likelihood} . \tag{2.14}$$

The first term of the equation is referred to as the log-likelihood loss or the maximum likelihood loss. This is given as $\log p(Y|X,\theta)$ [39]. This term optimises the ELBO so that it can best describe the data. The second term of the equation aims to minimize the KLD between the variational posterior and the true posterior distribution. This term also has been claimed to simplify the hypothesis space of the variational posterior [52], [39]. In doing so, models that best differentiate between the true and the variational distribution are favoured, as opposed to ones too complex to compute. Such behaviour in statistics is widely considered to be the Occam Razor [52] a model selection criteria that favour the simplicity of the models as opposed to their complexity.

To recap, the optimisation of the variational parameters involves minimising the KLD term or maximising the log observation likelihood term to obtain the optimal variational parameters ψ^{*}. This is shown in equation (2.14). The KLD term can be considered as a regularisation of the maximum likelihood term. Variational inference replaces computing the intractable integral of the marginal during marginalization in Bayesian inference with a simpler optimization problem. Additionally, the minimization step makes the process of finding the simplest solution faster than marginalising, as part of Occam's Razor.

Advantages and Disadvantages of Variational Inference

To consider the advantages of variational inference, literature shows that variational inference is useful in defining a "balance between complex models and models that explain the data well" [39]. The disadvantage of this approach is that it is difficult to scale with large datasets (i.e. > 1000 samples) and to adapt to complex models. The former is because the first term (i.e. log-likelihood) requires the calculation with respect to the entire dataset X. This can become computationally expensive with the increase in the size of the dataset. For the latter, the reason is that for a complex model there may not exist an analytical solution to the integral. Furthermore, differentiating the KLD term is difficult as using a sampling-based estimation (e.g. Monte Carlo sampling) will result in gradients with high variance. There have been many alternative approaches that attempt to circumvent these issues in variational inference. One particularly popular method in Bayesian methods in deep learning is the re-parameterisation trick [82]. The next section considers how the re-parameterisation trick helps with the circumventing of this issue.

2.1.9 Simplifying Variational Inference with Re-parameterisation Trick

This section aims to introduce the re-parameterisation trick. This is adapted from the derivation in the work of [39], [82] and of [83]. The motivation behind this technique is that gradients of the variational loss or the ELBO (specifically the KLD term $KL(q_{\psi}(\theta)\|p(\theta|X,Y))$ do not have closed-form solutions. Furthermore, if one was to use an estimator for differentiating the KLD term (i.e. a differential estimator), the gradients would be noisy if the estimator had high variance. For this reason, architectures that involve image generation, e.g. autoencoders [83], cannot backpropagate through stochastic nodes because of high variance. This can affect the learning in DNN architectures that use variational inference. One way to deal with high variance gradients is to use the re-parameterisation trick. This forms the subject of this section.

Re-parameterisation Using Differential Estimator

To begin with the formalisation, first consider the equation of ELBO loss function in (2.13). To recap, the term $\log p(Y|X,\theta)$ represents the log of observation likelihood and the term $q_\psi(\theta)$ represents the variational posterior. Note that $\log p(Y|X,\theta)$ has to be real and differentiable in order to compute the gradients. Then, assuming that there exists a finite integral for which $\log p(Y|X,\theta)$ is independent of θ, the derivative of $\log p(Y|X,\theta)$ with respect to the input can be written as $f^{'}$, for simplicity.

It is also assumed that $q_\psi(\theta)$ is distributed in a Gaussian fashion, such that $q_\psi(\theta) = \mathcal{N}(\mu,\sigma^2)$ where the parameters of this distribution can be defined as $\theta = \{\mu,\sigma\}$. The mean is given as μ and the standard deviation of the distribution is σ.

It is further assumed that $q_\psi(\theta)$ can be re-parameterised by a parameter-free distribution, also referred in literature as the auxiliary distribution [82], [39], $p(\varepsilon)$ where ε represents auxiliary variables that are related to the parameters θ via a differentiable, deterministic transformation function $g(.)$ that is bivariate (depends on two variables), such that $\theta = g(\psi,\varepsilon)$.

Given the above simple case, consider the example of a Gaussian distribution where $q_\psi(\theta) = \mathcal{N}(\mu,\sigma^2)$ and the parameter-free distribution takes the form $p(\varepsilon) = \mathcal{N}(0,I)$. Then, the deterministic function is $g(\psi,\varepsilon) = \mu + \sigma\varepsilon$. With this definition, one can estimate the integral $I(\theta)$ with the differential term as shown in equation (2.15), taken from [39]. In brief, the technique uses the differential $\hat{I}(\theta)$ to estimate the integral $I(\theta)$ where $I(\theta) = \int q_\psi(\theta)\log p(Y|X,\theta)d\theta$, the equation is shown in (2.15)

$$\hat{I}(\theta) = \frac{\partial}{\partial\varepsilon} p(\varepsilon) f^{'}(g(\psi,\varepsilon)). \tag{2.15}$$

Here, the term $\hat{I}(\theta)$ refers to as the differential estimator. The estimation considers the average of all the realisations $\mathbb{E}_{p_{(\varepsilon)}}[\hat{I}(\theta)]$ of the estimator to be equal to the differential integral $I(\theta)$, i.e. $\mathbb{E}_{p_{(\varepsilon)}}[\hat{I}(\theta)] = I(\theta)$. It is worth noting that the above formulation is for a continuous variable case. In DNN training, this is often discretized to be dependent on the mini-batch sample. This is considered next.

Substituting the Differential Estimator In the Variational Loss Equation

Following from equation (2.13), each of the integrals is re-parameterised to be integrated with respect to the auxiliary random variable ε. The variational posterior is replaced with the auxiliary distribution and θ with the deterministic function g. This step is shown in the equations below, taken from [39]. It is important to note that ε_i is the discrete variable that is sampled from the parameter-free distribution $p(\varepsilon)$. After the substitution, the equation takes the form

$$\begin{aligned}\mathscr{L}_{ELBO}(\psi) &= -\frac{1}{M}\sum_{i=1}^{M}\int q_{\psi}(\theta)\log p(y_i|\mathbf{f}^{\theta}(x_i))\,d\theta + KL(q_{\psi}(\theta)\|p(\theta|X,Y)),\\ &= -\frac{1}{M}\sum_{i=1}^{M}\int p(\varepsilon)\log p(y_i|\mathbf{f}^{g(\psi_i,\varepsilon_i)}(x_i))\,d\varepsilon + KL(q_{\psi}(\theta)\|p(\theta|X,Y)).\end{aligned} \tag{2.16}$$

The discrete versions of the variational parameters ψ_i are indexed by indices i of the mini-batch. The variational loss $\mathscr{L}_{ELBO}$ with the re-parameterised observation likelihood contains the sum over the inputs in the mini-batch, for a total of M inputs (i.e. $i \in \{1,\cdots,M\}$). A single mini-batch is indexed by a subset of $\{1,\cdots,N\}$, where N denotes the total number of samples in the dataset.

When considering solving the integral in (2.16), it is seen that solving for the entire dataset N is computationally expensive. Instead using a mini-batch is easier. However, still, there is the issue of solving the intractable integral over the likelihood which is taken with respect to the entire hypothesis. Regardless of using the re-parameterisation trick which can make the integral term differentiable and participatory in backpropagating, the problem is partially solved. In the next section, it is seen how Monte Carlo methods can help approximate this integral.

2.1.10 Approximating the Integral Term in the Variational Loss with Monte Carlo Integration

This section aims to discuss in detail the use of Monte Carlo methods to approximate the integral term in the variational loss. The advantages of using Monte Carlo methods for approximating the posterior is that it offers the simplicity of approximating the log-likelihood term in variational inference $\int q_{\psi}(\theta) \log p(y_i|\mathbf{f}^{\theta}(x_i))d\theta$. It is also useful for large and complex datasets, which can result in a multi-dimensional integral. Monte Carlo integration uses random sampling to estimate the integral. Later it is seen in Section 2.1.11 how this can be used in the context of dropout to estimate the mean and the variance in BNNs. This section begins with defining Monte Carlo estimation techniques.

Definition of Monte Carlo Methods

Monte Carlo methods are an example of stochastic sampling techniques that generate a sequence of random samples. Their application is mostly in the area of numerical approximation. Especially, when analytical solutions are not available. Their application can be for numerical integration, estimation or approximation and optimisation.

In Monte Carlo methods, the generated samples have Markovian property [24], meaning that the successive samples probabilistically depend on the previous samples. Once sampling is performed for a set sampling rate ($\approx$ 100 runs), the sequence of samples eventually distributes accordingly to the shape of the desired distribution specified by the user.

One disadvantage of sampling-based approximation methods is that they require a certain amount of time during deployment for the samples to converge to the desired distribution [73]. This can lead to another problem with sampling-based approaches where samples start producing a repeating pattern. To be specific, samples produced will be auto-correlated [73], i.e. there is no statistical difference between them. This can limit the ability of the sampling process to accurately capture the underlying distribution. Using sampling-based techniques also requires many iterations for generating a large number of samples and is computationally expensive with large architectures. There is a trade-off between accuracy and computational expense. Also, sampling methods are very sensitive to large perturbations in input. This can result in uncalibrated uncertainty outputs. This attribute is also used to explain some of the results in Chapter 4.

Approximating the Variational Loss with Monte Carlo Approximation

Monte Carlo methods can be used as a numerical approximation to the integral of the maximum likelihood term in the variational loss. However, in the interest of optimizing the variational inference process, the derivatives of log-likelihood with respect to the re-parameterised discrete variational parameters w_i are required. To recap, the loss can be formalised as shown in equation (2.17).

To recap, variational inference helps circumvent the problem of optimising for ψ which is impossible since $p(\theta|X,Y)$ is unknown. Instead, Monte Carlo sampling is used to optimise for the variational parameters ψ. Using the re-parameterisation trick in the previous section, re-parameterising each of the variational parameters of the distribution $q_\psi(\theta)$ with $\theta = g(\psi, \varepsilon)$, it is then possible to approximate the variational loss with a Monte Carlo integration. The term ψ_i is the discrete version of ψ since training is done batch wise w.r.t each inputs samples in a mini-batch. These range from $i = \{1, \cdots, M\}$ for mini-batch M, a random subset of the entire dataset. Approximating VI with Monte Carlo integration helps to avoid integration of the entire hypothesis space with respect to ε. In other words, the computation of $\int p(\varepsilon) \log p(y_i|\mathbf{f}^{g(\psi_i,\varepsilon_i)}(x_i))d\varepsilon$ is avoided. The final discrete version of the variational loss (or ELBO) takes the form in equation (2.17)

$$\hat{\mathscr{L}_{MC}}(\psi) = -\frac{1}{M}\sum_{i=1}^{M} \log p(y_i|\mathbf{f}^{g(\psi_i,\varepsilon_i)}) + KL(q_\psi(\theta)\|p(\theta|X,Y)). \tag{2.17}$$

It is assumed that Monte Carlo estimates are simulating the posterior distribution over the parameters. According to [39], the Monte Carlo estimate of variational loss $\hat{\mathscr{L}_{MC}}$ will closely approximate the true variational loss $\mathscr{L}_{VI}$ for all realisations of the estimate, i.e. $\mathbb{E}(\hat{\mathscr{L}_{MC}}) = \mathscr{L}_{VI}$, as the number of sampling runs is increased. The equation for weight update can be shown as

$$\Delta\hat{\psi}_i \leftarrow -\frac{1}{M}\sum_{i=1}^{M} \frac{\partial}{\partial \psi_i} logp(y_i|\mathbf{f}^{g(\psi_i,\varepsilon_i)})(x_i) + \frac{\partial}{\partial \psi} KL(q_\psi(\theta)\|p(\theta|X,Y)). \tag{2.18}$$

The rest of the optimisation in VI involves repeatedly sampling from the auxiliary distribution $p(\varepsilon)$ to produce ε_i. Followed by differentiating both the maximum likelihood and the KLD term with respect to the variational parameters ψ in order to obtain the change in estimate parameters $\Delta\hat{\psi}_i$. This is shown in equation (2.18) taken from [39]. The weights are updated according to $\Delta\hat{\psi}_i$. The model outputs $\mathbf{f}^{g(\psi_i,\varepsilon_i)}$ are also re-parameterised with the deterministic function g. This is done so that the gradients with respect to the model outputs can be obtained. The terms $\mathbf{f}^{g(\psi_i,\varepsilon_i)}$ and $\mathbf{f}^{w_i}$ are equivalent. Given that variational inference can be used to approximate the ELBO, the next section discusses how uncertainty can be obtained from the variational Bayes (i.e Bayesian methods that use variational inference to approximate the integral of the log observation likelihood).

2.1.11 Modelling Uncertainty Using Monte Carlo Dropout in Bayesian Neural Networks

The previous sections considered that Bayesian methods offer an alternative to the regular DNN architectures. It was seen how Bayesian methods can be difficult to infer from. This is because inferring from such methods requires calculating integrals of intractable distributions. It was later seen that VI can offer an alternative solution by casting the problem of inference to optimisation with respect to parameters of a pseudo-distribution that is easy to manipulate. Once this variational distribution is optimised, approximations of posterior distribution can be obtained.

It has still not been discussed how Bayesian methods can be used to characterise uncertainty. This section focuses on this aspect in-depth. Specifically, this section looks at how variational Bayes can be used to model uncertainty in a classification setting by using dropout layers as an approximation of Monte Carlo integration, this is also termed Monte Carlo dropout (MC dropout) [41]. Before getting to the discussion, it is worth mentioning to the reader that neural networks trained by Bayes by Backprop [15] are an example of variational Bayes. Hence, the characterisation of uncertainty using variational Bayes also applies to Bayesian neural networks.

Deep neural network architectures have the limitations that they output erroneous predictions with high confidence and are vulnerable to imperceptible attacks [49].In order to account for uncertainty, DNN architectures must be able to model both the posterior and the observation likelihood. This is lacking in traditional DNN architectures where the weights are deterministic and are limited to the scalar representation only. Bayesian Neural Networks (BNN) framework, on the other hand, focuses on shifting the deterministic setting to a stochastic one [118], [117]. In this setting, the DNN architectures can be used to model uncertainty. In BNNs, the scalar DNN weights are replaced with prior distributions. Though previous sections dealt with the inference part, this section will particularly focus on the uncertainty modelling in BNNs.

Defining Weights in Bayesian Neural Networks

The difference between DNNs and BNNs is that the former employs a scalar value-based representation of weights, whereas the latter replaces this with a prior distribution over the network's parameters [117]. This time, the parameters θ introduced in the previous sections can be defined as the weights of the BNN $\theta = \{W_j\}_{j=1}^{L}$. Here, W_j represents the j^{th} BNN layer, with L being the total number of layers.

In a classic example of placing a Gaussian likelihood over the parameters, it is possible to define the weights of a BNN as $p(W_j) \sim \mathcal{N}(0,I)$, where the matrix W_j, is associated with the j^{th} layer of the BNN architecture. Here, the $\mathcal{N}$ is a symbolic representation of a Gaussian distribution [143] and I refers to the identity matrix. It is worth mentioning that the bias b_j associated is considered a point estimate in both DNN and BNN architectures. The choice of placing a likelihood over the parameters is user-dependent.

Monte Carlo Estimation of First and Second Moments

To recall, the predictive variational distribution $q_\psi(y^*|x^*,X,Y)$ takes the form as shown in equation (2.19) and the parameters for a BNN architecture with a total of L layers can be denoted as $\theta = \{W_j\}_{j=1}^{L}$. The model output is denoted as $f^\theta(x^*)$ and the variational posterior as $q_\psi(\theta)$. The uncertainty from the posterior variational distribution can be derived via estimation of both the first and the second moments. Here, the first moments correspond to the mean and the second moments to the variance. The uncertainty is characterised by the variance value as shown in

$$q_{\psi}(y^*|x^*,X,Y) = \int p(y^*|f^{\theta}(x^*))\ q_{\psi}(\theta)\ d\theta. \tag{2.19}$$

First, consider the estimation of the first moments. Taking example of a traditional regression setting with a Gaussian likelihood, the observation likelihood takes the form $p(y^*|f^{\theta}(x^*)) = \mathcal{N}(y^*; f^{\theta}(x^*), \tau^{-1}I)$ for some $\tau > 0$. Since the posterior $q_{\psi}(\theta)$ does not depend on y^* it can be isolated and taken outside of the second (inner) integral term. The inner integral $\int y^* \mathcal{N}(y^*; f^{\theta}(x^*), \tau^{-1}I)\ dy^*$ can be integrated using parts. This leaves the term y^* which is defined as $y^* = f^{\theta}(x^*)$. Then, using the Monte Carlo integration leads to the last line in equation (2.20), taken from [39]. Increasing the sampling rate T will improve the estimation of the mean, this is shown in

$$\begin{aligned}
\mathbb{E}_{q_{\psi}(y^*|x^*)}[y^*] &= \int y^* q_{\psi}(y^*|x^*)\ dy^* \\
&= \int\int y^* \mathcal{N}(y^*; f^{\theta}(x^*), \tau^{-1}I)\ q_{\psi}(\theta)\ d\theta\ dy^* \\
&= \int \Big(\int y^* \mathcal{N}(y^*; f^{\theta}(x^*), \tau^{-1}I)\ dy^*\Big)\ q_{\psi}(\theta)\ d\theta \\
&= \int f^{\theta}(x^*)\ q_{\psi}(\theta)\ d\theta, \\
\tilde{\mathbb{E}}[y^*] := \frac{1}{T}\sum_{t=1}^{T} f^{\hat{\theta}_t}(x^*) &\xrightarrow[\mathrm{T}\to\infty]{} \int f^{\theta}(x^*)\ q_{\psi}(\theta)\ d\theta.
\end{aligned} \tag{2.20}$$

Finally, consider the estimation of the variance, the steps to follow will be similar to estimation of mean. In here, the quantity of interest is $\mathbb{E}_{q_{\psi}(y^*|x^*)}[(y^*)^T(y^*)]$ and it is now known from previous derivation that $y^*\mathcal{N}(y^*; f^{\theta}(x^*)$ integrates to yields $\mathbb{E}_{p(y^*|x^*,\theta)}[y^*]$. Then using integration by parts, the term $(y^*)^T\mathcal{N}(y^*; f^{\theta}(x^*)$ will in the same fashion get integrated to yield $\mathbb{E}_{p(y^*|x^*,\theta)}[y^*]^T$. The added term in the third line of equation (2.21) $Cov_{p(y^*|x^*,\theta)}[y^*]$ represents the covariance of the prediction y^*. The terms in the brackets in the third line of equation (2.21) correspond to the mean and standard deviation of a Gaussian likelihood example which takes the form $\tau^{-1}I + f^{\theta}(x^*)^T f^{\theta}(x^*)$, this is shown in

$$\begin{aligned} \mathbb{E}_{q_\psi(y^*|x^*)}\Big[(y^*)^T(y^*)\Big] &= \int (y^*)^T(y^*)\ q_\psi(y^*|x^*)dy^* \\ &= \int \Big(\int (y^*)^T(y^*)\ \mathcal{N}(y^*; f^{\theta}(x^*), \tau^{-1}I)\ dy^*\Big)\ q_\psi(\theta)d\theta \\ &= \int \Big(Cov_{p(y^*|x^*,\theta)}[y^*] + \mathbb{E}_{p(y^*|x^*,\theta)}[y^*]^T\mathbb{E}_{p(y^*|x^*,\theta)}[y^*]\Big)q_\psi(\theta)d\theta \\ &= \int \Big(\tau^{-1}I + f^{\theta}(x^*)^T f^{\theta}(x^*)\Big)q_\psi(\theta)d\theta. \end{aligned} \tag{2.21}$$

From here, to yield the estimator in the correct form, an additional step is taken for the MC integration with respect to T samples. Here, $\hat{\theta}_t$ is the Monte Carlo estimator of θ_t. It is also important to note that the above estimate of the uncertainty is the MC approximate of the posterior distribution. Finally, the model's predictive mean can be obtained as shown in equation (2.22), and the posterior variance as shown in (2.23)

$$\tilde{\mathbb{E}}\Big[(y^*)^T(y^*)\Big] := \tau^{-1}I + \frac{1}{T}\sum_{t=1}^{T} f^{\hat{\theta}_t}(x^*)^T f^{\hat{\theta}_t}(x^*), \tag{2.22}$$

$$\tilde{\text{Var}}[y^*] := \tau^{-1}I + \frac{1}{T}\sum_{t=1}^{T} f^{\hat{\theta}_t}(x^*)^T f^{\hat{\theta}_t}(x^*) - \tilde{\mathbb{E}}[y^*]^T\tilde{\mathbb{E}}[y^*]. \tag{2.23}$$

All derivations in this section are adapted from [39]. This section introduced the modelling of first and second moments from the variational posterior distribution in a BNN network. In particular, it was seen how the second moment estimates can be used to model uncertainty. The important point that the reader should note is that the Monte Carlo integration step is dependent on the sampling rate T. If the sample rate is small, the estimates will be inaccurate. This will later be used to explain results from Chapter 3 and 4. Additionally, there are many other disadvantages of using MC dropout based uncertainty estimation using dropout layers. The next subsection discusses in-depth all of these disadvantages.

2.1.12 Challenges in Modelling Uncertainty in Bayesian Neural Network with Monte Carlo Dropout and Variational Inference

So far it has been considered that modelling uncertainty in BNN architectures can be approximated using Monte Carlo estimation and dropout layers. There are many caveats of using this type of approximation. Firstly, it is worth noting that the test time scales with the sampling rate T. This is because MC dropout techniques require several stochastic forward passes through the architecture. Consequently, the testing time is scaled with the sampling rate T. In the experiments in Chapter 4, T is set to 100.

Another issue with Monte Carlo dropout methods is that the uncertainty estimate obtained is not calibrated [40]. A model's uncertainty is calibrated if a given uncertainty prediction from the model is reliable [27].

Compactness and Tightness Limitations in Variational Inference

For considering the last few disadvantages, it is important to recall that Monte Carlo methods are used to approximate integral term in variation inference $\int q_{\psi}(\theta) \log p(y_i| f^{\theta}(x_i))d\theta$. Hence, the limitations of the variational inference apply to Monte Carlo approximation as well. There are two that are heavily discussed in literature [182], i.e. the compactness and the tightness problems with variational inference. The prior is straightforward to describe, the latter has been mostly found through empirical means in Bayesian methods literature [182], [105].

To consider the first case, it is worth first recalling the formulation for variational inference from (2.11). It can be seen that the variational form $KL(q_{\psi}(\theta)\|p(\theta|X,Y))$ is compact. This means that wherever in the density approximation process if a region occurs such that it has zero density, it follows that the KLD divergence term will be infinite in that region. Though this sounds rather trivial and to some extent may be ignored for cases with the uni-modal distribution. The flaw of this limitation becomes more apparent when extending to let's say approximation of a bi-modal distribution. In this example, if a bi-modal distribution was distributed in a way that the two modes are separated by a region of zero density. The variational inference scheme will approximate only with a uni-modal distribution. Hence, it will distribute mass only on the regions that are distant from the zero density region. This would lead to a less accurate approximation of the true observation likelihood when compared to an estimate that would approximate a density by averaging the two modes.

Biasness Problem in Variational Inference

Secondly, to consider the bias issue in variational inference. This particular problem has been observed empirically in literature [182], [105]. In brief, the issue arises from the nature of the objective function that is the KLD term $KL(q_{\psi}(\theta)\|p(\theta|X,Y))$. To recap, this term forces the variational posterior to be as close as possible to log-likelihood for all possible observation likelihood parameters θ.

Studies have reported that the influence of the bounds in the KLD can lead to *pruning* (or removing) components from a range of chosen models and can lead to underestimation [182], [105]. Particularly, [182] found that when given a known true observation likelihood and a parameter to estimate (in their example the parameter is termed λ as the time decay constant for their time-series problem), increasing the number of parameters to be estimated leads to an increase in bias in the parameters. They also observed that this increase in bias was rejecting models from hyperspace that was comparatively more useful in terms of information.

A similar observation was considered in the works of [105], where a mixture of Gaussian distributions was made to fit a 1D dataset and it was found that as the model estimated more parameters, it simultaneously began to prune out the mixture models. Both the works have agreed that the consequences of this *spontaneous* Occam's razor [52]. In brief, this is not beneficial in terms of modelling and can damage the propagation of uncertainty in models that leverage variational inference schemes.

This section reviews the challenges involved in using sampling-based methods, such as MC dropout, in the modelling of uncertainty. It is seen that sampling methods are affected by the rate. The higher the rate, the better the approximation. However, the higher the computational time. Furthermore, there is also the issue of lack of calibration in MC dropout methods. The next section considers the use of Gaussian processes as an alternative approach to Bayesian methods in deep learning.

2.1.13 Gaussian Processes in Machine Learning

In the previous sections, a thorough review of Bayesian techniques that use approximation schemes like variational inference and Monte Carlo methods were given. This was followed by a discussion on how such models approximate uncertainty through the variance information. However, the models discussed in the previous sections are parametric. This is because they can be defined by a finite set of parameters. There exist a probabilistic family of models that are not restricted by parameters. These are Bayesian non-parametric models [143].

Bayesian non-parametric models are important to investigate for the work in this thesis since such models are capable of representing the input dataset much more accurately than parametric ones, as a result of their vast hypothesis space. One particular Bayesian non-parametric model that is of interest in this section is the Gaussian process (GP) model [143].

This section begins by giving an overview of the importance of Bayesian non-parametric models from a modeller perspective. It then provides a mathematical description of Gaussian Processes including what are the major components of a typical GP model. It then follows up with the section on kernel choices in GP and then finally ends with the section on the challenges in GP modelling.

2.1.14 Reviewing the Importance of Bayesian Non-Parametric Modelling

Before reviewing the drive towards non-parametric models it is of essential value that such techniques are discussed from the point of view of a modeller rather than a statistician. This is because a statistician merely looks for the intricacies in the data, perhaps not giving too much attention to the modelling aspect. It is important to remind oneself that though learning from data is the utmost goal of machine learning, the modelling side of things should also be given prominence. Hence, the tone in this subsection is slightly tilted towards modelling. This is because the author firmly believes that understanding the basics of modelling will act as a perfect foundation for justifying the importance of Bayesian non-parametric modelling.

Definition of a Model

Firstly, how can one define a model ? A model can be considered as a representation of data [44]. The more expressive the model the better its representation of data, however, at the cost of deeper architectures that can introduce non-linearity [139]. The purpose of modelling is to predict events. Two properties allow probabilistic models to differ from other types of models; learning from data and stating the confidence in predictions. The latter is used to account for uncertainty.

Properties of Bayesian Models

Bayesian methods differ from other types of models in it that they seldom suffer from the problem of overfitting. However, this only applies to fully Bayesian methods. Monte Carlo dropout methods with deterministic DNN weight setting, e.g. Bayesian SegNet (see Chapter 3), are an exception since these methods use approximation schemes. They do not model their parameter space probabilistically but instead use point samples representation. This representation can be termed as the deterministic weight setting for brevity.

Occam's Razor

Fully Bayesian methods model a probability mass function over their parameter space, e.g. BNNs (see Section 2.1.7) instead of using point-estimates. Additionally, Bayesian methods benefit from Occam's Razor principle [142] whilst marginalising over the parameter space (see Section 2.1.7). Such principle allows Bayesian methods to pick models that are simple enough to describe the data, but are not too complex or not too simple altogether.

Bayesian methods inherit this property from the laws of probability that they are derived from. Specifically, one of the rules stated by these laws is that probability distribution must always sum to one [114]. This is one example of Occam's Razor behaviour present inherently in Bayesian methods.

Neural Architecture Search

In the author's opinion, this can give the impression to the reader that standard deterministic DNN architectures may not benefit from this. However, there has been research on this topic on enforcing Occam's principles on DNNs using search algorithms that compute the complexity of the model selection process in DNNs. Mainly analyzing weight matrices. This study is popularly known as the neural architecture search. Though this is outside the scope of the thesis, the readers are encouraged to consider the survey paper on neural architecture search [33].

Functional Space View of Non-Parameteric Models

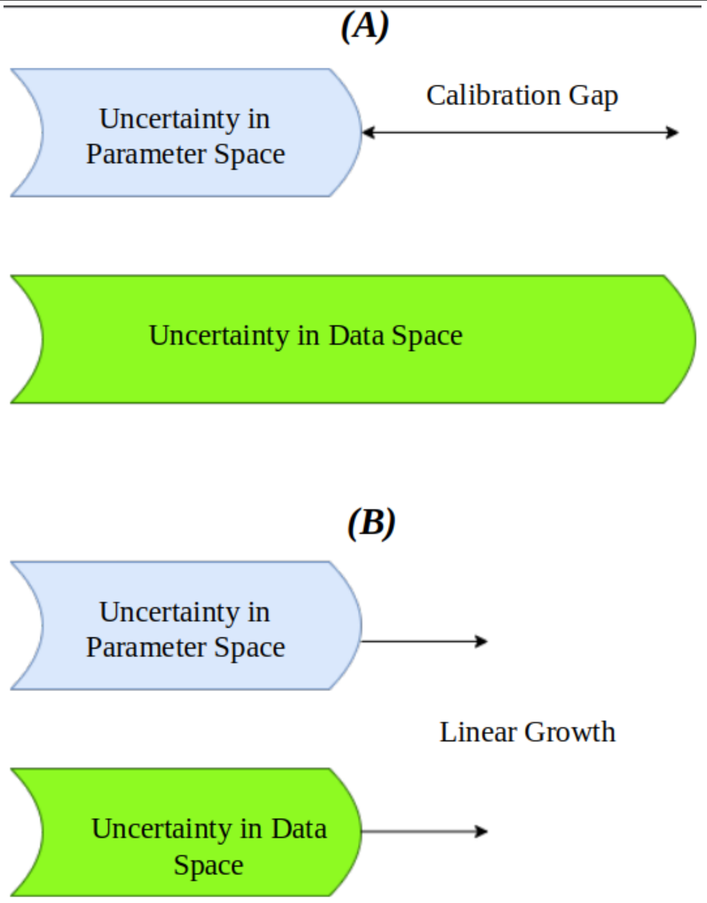
To compare the two approaches further, according to to [46] and [67], Bayesian non-parametric models differ from their parametric counterpart in it that it is assumed that the data distribution can no longer be described by a finite set of parameters alone. It is often the case that in Bayesian non-parametric modelling, the weights would take the form of a function. This can cause ambiguity in defining non-parametric models. Though these models are referred to as not having any parameters, it is more correct to define them as having infinitely many parameters [39]. This will be later considered in the next section.

This freedom in parameter choice means that Bayesian non-parametric are more flexible than their parametric counterparts. As a result, Bayesian non-parametric models have more expressive power than parametric models. In statistics, the word *expressivity* refers to the ability to model high-level concepts as well as being able to efficiently represent the data distribution [119].

Uncertainty Quantification Perspective of Non-Parametric Models

Having the ability to choose from an infinite space of parameters is very important to consider from the point of view of uncertainty quantification. This is because the more expressive the

Fig. 2.5 Comparison of non-linear as opposed to linear growth of both uncertainty in model space and uncertainty in data space. Prior is shown in (A), latter in (B)


(A)
Uncertainty in Parameter Space
Calibration Gap
Uncertainty in Data Space
(B)
Uncertainty in Parameter Space
Linear Growth
Uncertainty in Data Space

model, the more accurately it will approximate uncertainty. The reason behind this is that it will have a better representation of the data. This type of uncertainty is also referred to sometimes as the aleatoric uncertainty [79].

Linear Growth of Uncertainty

Another advantage of having an infinite space of parameters is that the amount of information the parameters can capture will grow with the size of the data. This is also important once again from the point of view of uncertainty quantification because it is easier to calibrate uncertainty in a model that grows with data. In the sense that one can place the calibration of uncertainty in a linear framework, which makes calibration very easy to do. In doing so, the uncertainty can be calibrated synchronously with data growth, as seen in Figure 2.5.

Works from [80] and [138] discuss the advantages of having uncertainty in models and parameters grow linearly. Here [80] shows that it is possible to place calibration in a linear model that allows easy calibration of uncertainty in GP. On the other hand, [138] accomplishes the calibration through averaging predictions from a variety of models that they employ in their system. Both works agree in their approach of placing calibration in a linear framework. However, the underlying difference in their approach is that one uses model averaging and the other uses the GP model.

Calibration of Uncertainty

In statistics literature, other calibration techniques make use of Bayesian non-parametric models apart from GP e.g. in [10] that uses sampling procedures to calibrate uncertainty in a mixture model of latent distributions to which they refer to as *beta mixtures*. However, from the author's point of view, calibration is outside the scope of this thesis since all models presented in this work assume perfect calibration. For an in-depth view, one may review the aforementioned works that give a more thorough review of calibration in Bayesian statistics.

This section focused on reviewing Bayesian non-parametric systems. The approach was taken from the point of view of modelling. Firstly, the issues with Bayesian parametric models were discussed. Then moving to Bayesian non-parametric methods. In particular, the advantages of using non-parametric models were taken into account. It was seen that one of the advantages was that Bayesian non-parametric can choose from an infinite parameter space. This is shown to be vital for uncertainty quantification. Though in literature, other advantages of using Bayesian non-parametric models are mentioned in-depth (e.g. exchangeability and projectivity), not all of them are useful in the context of uncertainty quantification. Interested readers are encouraged to consider Chapter 1 from [46] and [67] for an in-depth analysis. For a quick review of Bayesian non-parametric approaches, readers are encouraged to review the work from [44]. The next section will consider discussing an example of a Bayesian non-parametric relevant to this thesis, i.e. the Gaussian Process.

2.1.15 Review of Basic Components of a Gaussian Process with a Squared Exponential Kernel: The Mean and the Covariance Function

This section aims to formalise the Gaussian process learning framework. From the previous section, it was seen that Bayesian non-parametric methods extend their parameter space on the entirety of the dataset. In other words, by spreading their parameter space, the finite space of their parameters is transformed into infinite space. This is also known as the functional space view of non-parametric models. This functional space view of parameters has not only been discussed in the study of Bayesian non-parametric, but also in deep learning.

Examples of Studies on Functional Space View of Non-Parametric Models and Deep Neural Network Models

An example of functional form study on DNNs can be seen from the works of [126] and [117]. Both the works agree on the phenomenon that DNNs with a large network (large in terms of their magnitude rather than the amount) have the capability of fitting any function. Hence, such architectures with large depth start exhibiting behaviour that is more akin to non-parametric methods. In particular, [117] claims that if one was to increase the depth of any BNN architecture, with the caveat that each weight takes the Gaussian likelihood form, then the network would behave as if it is a Gaussian process. So along with this argument, it can be supposed that a Gaussian process model is an extension of its finite form, that is the Gaussian distribution.

Defining a Gaussian Distribution

A Gaussian distribution can be defined with it's mean and variance for say for example a distribution $p(x) \sim \mathcal{N}(\mu, \sigma)$. A GP model, on the other hand, is defined by its mean function and a covariance function. It is a form of a stochastic process, specifically, a class of models that place distribution in place of a function [143].

These distributions model a finite space of functions, say $\mathbf{f} = \{f(x_n)\}_{n=1}^{N}$. Here, x_N is considered to be an input sample from a sequence, which comes from a sub-sample X of size N such that $\mathbf{X} = \{x_n\}_{n=1}^{N}$. The parameters of the distribution $p(\mathbf{f})$ are also referred to as latent variables in literature [63].

Defining a Gaussian Process

The Gaussian process can then be defined by the posterior probability as shown in equation (2.24)

$$p(\mathbf{f}|\mathbf{X},\theta) = GP(\mu, k(x_n, x_n')). \tag{2.24}$$

Here, the symbol GP is used to define the Gaussian process. Furthermore, x_n and x_n' are two consecutive data points. The term $k(x_n, x_n')$ is the covariance function, sometimes also referred to as the kernel function [143]. Also, θ refers to the parameters of the kernel $k(x_n, x_n')$. For example, the parameters of a squared exponential kernel [143] are the amplitude and lengthscale. These can be defined as

$$\mathbb{E}[f(x_n)] = \mu(x_n), \tag{2.25}$$

$$Cov[f(x_n), f(x_n')] = \mathbb{E}[(f(x_n) - \mu(x_n))(f(x_n') - \mu(x_n'))] = k(x_n, x_n'). \tag{2.26}$$

In the mean function of the Gaussian process (in equation (2.25)), $\mathbb{E}$ is defined as the expected value. The x_n and x_n' are any two consecutive datapoint samples from X e.g (x_1, x_2) or (x_3, x_4). Before the GP is conditioned upon the dataset, the mean function is the only parameter that can be used to define the GP. Once the dataset is introduced, the kernel function is taken into the account.

Properties of the Kernel Function

Majority of times in research, the prior mean will be set to zero [63], [143]. This would result in all of the structure of the input data being determined by the covariance function of the GP model only. The kernel decides how a GP model generalises and extrapolates to new data. Moreover, the kernel decides the hypothesis space of the GP model. This affects the model selection behaviour of the GP as well (see Section 2.1.14). There is a limit to the type of functions that can be chosen. This is set to a condition so that the kernel function chosen must return the covariance matrix as a positive definite matrix. A positive definite matrix is defined to be a matrix that satisfies the condition $z^T Mx > 0$. Where, z refers to each vector entry, M being the positive definite matrix and z^T the transpose of that vector.

Choosing a Kernel Function

Though many kernel functions can be used for a GP model, there is a simple rationale that can be adopted when choosing one. Since the kernel function is used to define the similarity between two consecutive data points (x_n, x_n'), then observing the nature of the data would guide the choice of the kernel. For example, if the data is structured (e.g. in decision trees), then it makes sense to use a tree-structured kernel [26].

Uncertainty Quantification Perspective of a Squared Exponential Kernel

From the point of view of uncertainty quantification, the choice of kernel should be motivated by the way uncertainty should be modelled. For example, if one was to model uncertainty such that that the uncertainty should increase away from the training data samples, then a squared exponential kernel should be used [39], [143]. A squared exponential kernel is also referred to as the radial-basis function (RBF) kernel in literature [143]. This takes the form

$$k_{RBF}(x_n, x_n') = \sigma_{RBF}^2 exp\Big(-\frac{x_n - x_n'}{2L^2}\Big). \tag{2.27}$$

Here, the parameters σ_{RBF}^2 and L are referred to as the amplitude and the lengthscale respectively. The prior can be considered as a scaling factor that determines the average distance between the function and the mean. The latter describes the smoothness of the function. This means that the smaller the lengthscale, the faster the function value will change. Furthermore, it also determines how far it is possible to extrapolate from the new training input.

In this section, the basic components of a GP model were introduced. Firstly, it was discussed how GP is a Bayesian non-parametric model that extends itself from its finite form, that is the Gaussian distribution, to an infinite space of parameters. Then a basic GP model was considered along with a mean and a covariance function. Different properties of these parameters were discussed. The next section considers some of the challenges of using Gaussian process models, specifically from the point of view of computational expense.

2.1.16 Challenges in Gaussian Process Inference

This section reviews the challenges of inferring from Gaussian processes. First, it must be brought to the attention of the reader that a GP, like BNN architectures, is a Bayesian model. The rules of marginalisation apply to GP as well as BNNs. The quantity of importance are both the posterior $p(\mathbf{f}|\mathbf{X},\theta)$ and the predictive distribution $p(\mathbf{y}_*|\mathbf{x}_*,\mathbf{X},\mathbf{y},\theta)$ based on a new testing point $\mathbf{x}_*$ and output $\mathbf{y}_*$ and the observations $\mathbf{y} = \{y_n\}_{n=1}^N$.

The problems that were addressed to BNNs in the previous sections, also apply to the GP models. Hence, to recap, these include; the complexity of computing intractable integrals of observation likelihood multiplied by prior, scalability issues, long inference times and calibration of uncertainty.

The difference now is that in GP, the parameters to optimise belong to the kernel function $k(x_n, x_n')$. Whereas, in BNNs, these would be the weights of the neural network. To recap, for an RBF kernel, the parameters are defined as $\theta = (\sigma_{RBF}^2, L^2)$.

Though the scope of this thesis is around uncertainty quantification, this section focuses more on the problems relating to inference in GP. This is because the calibration of uncertainty is not as important as problems from inference in non-parametric models, as discussed before. Calibration becomes vital when data is scarce (e.g less than 100 samples) [39].

To discuss this issue, this section considers inference in GP from the point of view of both regression and classification. In regression case, the observations/labels are related to the inputs in the way that $\mathbf{y} = \mathbf{f}(\mathbf{X})$. Then, the predictive distribution based on a sample test point $f(x_*)$ takes the form $p(f(x_*)|\mathbf{y},\mathbf{X},\mu(x_*),k(x_*,x_*'))$.

For regression, if the conditional likelihood $p(\mathbf{y}|\mathbf{f})$ and the prior $p(\mathbf{f})$ are Gaussian distributed, then the predictive posterior will have a closed-form solution. However, two particular challenges arise when dealing with both the inference and when using non-Gaussian likelihood. The latter is the case with classification.

Firstly, the challenge in inference comes from the computation of the predictive posterior $p(f(x_*)|\mathbf{y},\mathbf{X},\mu(x_*),k(x_*,x'_*))$. This involves computing the inverse of the determinants of the kernel matrix $k(x_*,x'_*)$. Such operations have a complexity of $\mathscr{O}(N^3)$ flops, where N is the total number of training data points. This limits the scalability of GP to large datasets as well as the speed of forward pass.

Secondly, another challenge comes from scenarios where the conditional likelihood $p(\mathbf{y}|\mathbf{f})$ is non-Gaussian. In such cases, a tractable analytical solution to the integral of the conditional and the prior $p(\mathbf{y}) = \int_{\mathbf{f}} p(\mathbf{y}|\mathbf{f})p(\mathbf{f})d\mathbf{f}$ which is a crucial part in marginalisation. Though many approximations are provided in the literature, this thesis specifically uses a sparse variant of the variational inference method [63]. This method is also ideally preferred since it tackles both these challenges collectively. The next section discusses the literature regarding semantic segmentation.

2.2 Review of Methods in Semantic Segmentation

Semantic segmentation, as part of scene understanding [55] in computer vision, is an extension of classification. Classification involves placing objects or tasks into categories [12]. Segmentation, on the other hand, requires more of the *semantic* meaning from the classes. An example of a semantic feature would be the location of each of the categories relative to each other. For example, the car category should be present on top of the road category [113], [183]. Some of the common applications of semantic segmentation are in the area of medical imaging [75], [186] autonomous driving [34], [43], [25], machine translation [123] and weather forecasting [2]. What is common among these applications is that they all require reliable decision making. This has driven the field of deep learning-based segmentation to consider architectures that not only predict but also state the confidence in their predictions. The Bayesian SegNet [78] is one example of this movement.

Now, looking at scene understanding in depth. Firstly, this problem can be understood from the point of view of a simple classification example. If one was to classify an image that contained a digit '1', then the predictions from the classifier would not change if the position, angle and orientation of the digit '1' are changed. This is because simple classifiers, like LeNet-5 [89] or AlexNet [85], cannot capture the semantic meaning behind labels. Additionally, problems of scene understanding require that predictions must change accordingly to the change of input.

The question yet remains, how can one define scene understanding ?. Scene understanding involves establishing the contextual relationship between the object and the environment [55]. A scene can be considered as a combination of objects, semantics and the environment. The task of scene understanding is to make a useful interpretation and an abstraction of visual content. The task should extend recognising of objects to include the spatial distribution and the topology of the multiple objects present in the scene [55].

To summarize, semantic segmentation is an example of scene understanding. One example of semantic segmentation is pixel-wise or pixel-level labelling [55], [77], [180], [183]. Pixel-wise labelling problems are segmentation problems defined on a pixel level. The task builds on the classification of images along with localising the objects at original image pixel resolution. The segmented outputs contain pixel-level maps that localise different objects in the input image. The next section reviews the most common deep learning based methods in segmentation.

2.2.1 Review of Deep Learning Based Methods in Segmentation

Since the rise of the popularity of deep learning, the field of segmentation has gradually adopted methods in the study of deep learning. As reported in [113], there have been more than 100 research works in deep learning-based segmentation from 2019 to 2020. This section aims to review the techniques used in segmentation from the point of view of DNN and CNN based architectures. In particular, this sections reviews the fully connected architecture, the idea of skip connections [101], combined methods of both CNN and traditional CRF techniques, the global context concatenation method in ParseNet [100] and finally the encoder-decoder architectures like SegNet [7], U-Net [149] and the DeconvNet [122].

Fully Connected Architectures

Firstly, to consider the fully-connected architectures (FCNs) [101]. Unlike AlexNet [85], this type of architecture contains only convolutional layers. The later fully-connected layers for classification are replaced with convolutional layers. As reported by [101], this type of architecture overcomes the problem of non-fixed sized inputs and outputs which can be problematic for fully connected layers. This problem is common in architectures such as AlexNet [85] and LeNet [89] as these models have fully-connected layers that have fixed dimensions and also discard spatial information. This is not ideal for segmentation since it was discussed previously that spatial information between categories is a semantic feature. Using convolutional layers, the FCN model outputs spatial segmentation maps as opposed to classification scores in standard CNN architectures. Convolutional layers average and then sum all the inputs present in the *receptive field* (filter window) of the filter, this preserves spatial information as weights are shared across the field/window.

Review of Skip Connection Architectures in Segmentation

Apart from architectural contribution, the work of [101] also introduced the notion of skip-connections. The purpose of such techniques is to combine the semantic information and the appearance information from the convolutional layers. The semantic information comes from the deep and coarse layers while the appearance information comes from the shallow and fine layers [180].

In simple classification, most objects are ignored and only the targeted ones are considered for learning. For example, a network trained to classify cats and dogs will not consider input samples containing say e.g. birds in the distant background. In segmentation, each pixel needs to be localised with respect to the objects in its surrounding [113], [183] and [180]. This motivates the need to capture the differences between local and global features. Having an exchange of information from fine layers and coarse layers helps make accurate local and global predictions. This means that a small object would be segmented in the correct size and shape with respect to large objects [113], [183]. The same applies in the relation of distance (i.e. objects in distance).

Limitations of Fully Connected Architectures

Despite these features introduced from the works of [101], it has been shown by [100] and [21] that FCNs are limited in terms of localisation as well as preserving global context features. Specifically, [21] highlights that FCN “ignores useful scene-level semantic context” [21], while [100] highlights that FCN architectures are slower compared to traditional CNN methods. They also do not take into account the global context information in segmentation.

Global Vs Local Contextual Information

Briefly, contextual information encodes *contextual relationships*. These relationships formalise the contextual information as a measure of *semantic compatibility* relationships between a given object and its neighbouring object (global context) and the relationship between features within an object (local context). An example of global context was seen previously as spatial correlations like a car is more likely to be semantically compatible to a road as a pedestrian is more compatible to a side-walk category. Consequently, spatial correlation is an example of global contextual information.

According to [97] global context information involves changes in spatial information while local context information deals with the local multi-scale information [96], i.e. the information between features within an object category. However the key idea remains the same, that is the objects to be segmented by a classifier must establish a semantic relationship between themselves and neighbouring objects.

Loss of Global Contextual Problem and Vanishing Gradient Problem in Fully-Connected Architectures

Going back to the limitations of FCNs, the reason behind the loss of global context information in FCNs, as highlighted in [100], is that the architecture utilises consecutive pooling operations, this reduces the resolution of features over time. The idea is similar to the problem of vanishing gradients in training large scale neural networks like ResNet [59] since that also involves loss of information (in this case gradients) in layers further from the output layer. The difference is that one causes degradation of feature information from consecutive pooling while the latter causes degradation of gradient information from consecutive non-linearities.

Combining Convolutional Architectures with Conditional Random Fields for Post-Processing and Refinement

Given the aforementioned limitations of the FCN, several methods in the segmentation literature have provided solutions in the light of better localisation accuracy and global feature preservation. Particularly, the works of [156] and [196] aim to improve the CNN responses from the final layer by training traditional methods from CRFs place after CNN based architectures.

Such methods have been proven to give accurate localisation results for object segmentation by integrating more context into the architectures. The difference in their approaches is that [156] uses a VGG based architecture while [196] uses a recurrent type neural network architecture [110]. These types of DNN architectures can incorporate memory components that are mainly used for time-series data. They both show improved performance when compared to FCN, however, they have been reported to be difficult to train [100]. In the author's opinion, a much better solution is proved by the work of [97] where patch-based global and local information is incorporated separately in their CNN architecture. This simplifies training since the learning is done on a small subset of the input image. The obvious downside is the difficulty of preserving the contextual relationships while observing the input partially.

ParseNets

A different approach to using a mixture of CNN and the traditional CRF technique is the idea of adding global context using averaging features. This is specifically explored in the works of [100] that term their architecture as ParseNet. This architecture explores one way to overcome the issue arising from the limitations of FCN. ParseNet aims to use global concatenation methods for efficient parsing of global features into local features. It improves the work of FCN by adding global context information to the feature maps in each of the convolutional layers.

As detailed in [100], ParseNet works by using the average feature for a layer and using this to augment the features at each layer location. A context vector is obtained by pooling the features over the entire image and is later unpooling (opposite operation to pooling) it produces feature maps of sizes similar to previous features. In this way, the global information in terms of size is preserved.

Mixture of Experts

The mixture of experts is another popular deep learning based method that is used in segmentation methods. This technique is a type of ensemble based learning technique. In ensemble based learning, multiple models with different sets of hyperparameters are trained on an input task [8]. The predictions from the respective models are combined to give one reliable prediction.

In the mixture of experts, first proposed in [71], the inputs are decomposed into separate sub-tasks. Each expert model is trained on a separate input task. A separate gating model is trained separately for which the inputs are fed directly. The task of the gating network is to learn which experts to trust based on a given input. For example in [71], gating networks can assign probabilities to each expert. The higher the probability the more reliable the expert on an input task.

The challenge in using the mixture of experts is to partition the input domains. This requires some domain knowledge. Specifically, in the application of segmentation, the works of [132] and [184] both use a maximum of two expert architectures and both divide their inputs by different semantic modalities. This means that a subset of input may contain classes that represent more of the category sky or pedestrian (in the case of an outdoor dataset like CamVid [16]). Therefore, the experts in either of the works get separate input tasks.

The main difference in their approach is that [184] also uses segmented images as inputs to their second expert network and [132] subdivides the inputs into their difficulty and different scenarios e.g. windy, rainy or stormy etc. Additionally, the work of [132] use ResNet [59] as backbone architecture for both its expert networks while [132] uses a 13-layered variant of the U-Net [149] architecture they term UpNet. The difference in their architecture is that in [132] each expert network comprises two streams: pooling and residual. The pooling stream is kept for the encoding and decoding architectures. The residual units are present in the residual stream. Furthermore, in the work of [132], each expert network is capable of the whole input image while the experts in [184] partially observe the input. Furthermore, [132] performed a separate experiment to judge which layer from the expert network to extract the features. They concluded on choosing layers closer to the encoding end.

In terms of gating network architecture, both the works of [132] and [184] feed the entire input to the expert architecture and in both, the outputs of the experts are weighted by the gating network. The difference in their approaches is that [184] only uses a single-gated network architecture and the combined feature maps from the experts are passed through an additional convolutional layer which they claim to improve the performance of the gating network. This is further empirically proven in [132] which supports the rationale behind using a separate convolutional layer after the outputs of each of the experts are combined. However, they also pointed out that using a convolutional layer is only useful when the gating network predicts separately for each class. Lastly, both approaches achieve better performance than the state-of-art. However, [132] extend their experiments further to incorporate challenging environments with different lighting and weather effects.

SegNet, U-Net and DeconvNet

Finally, the last methods to consider are the encoder-decoder based architectures. Encoder-decoder methods employ both the use of a series of hidden layers that compress the input information down to a latent dimension (often a 1D vector). These can be either FC or convolutional layers followed by pooling operations for dimensionality reduction. Followed, by a series of layers that decompress these features back to the original input dimensions (commonly a 4D input tensor). These can be either deconvolutional layers, similar to those used in ZFNet [192], or upsampling layers [7].

The architectures SegNet [7], U-Net [149] and DeconvNet [122] are popular examples of encoder-decoder architectures. They all utilise a set of hidden layers for encoding information, e.g. [149] uses this information to determine the location of objects to segment. Furthermore, they all use the decoding half to determine the pixel position. Lastly, they are all derivatives of the original VGG [164] networks.

The difference between them is that the DeconvNet, compared to both [7] and [149], uses fully-connected layers for both encoding and decoding hidden layers. This increases the model parameters greatly. The work from [7] reports that DeconvNet is more substantial than both the SegNet and the U-Net, requires plenty of computational resources and is hard to train via end-to-end means.

Comparing State-of-Art U-Net with SegNet Architecture

The SegNet and the U-Net, on the other hand, use only convolutional layers. The architectures are roughly the same, with U-Net only lacking the fifth convolutional and max-pooling layer compared to the SegNet. The major difference between them is the SegNet reuses information from pooling layers (specifically the pooling indices) from the encoder half, and feeds them to the upsampling layers in the decoder half. U-Net, on the other hand, simply feeds the entire feature map from the encoder to the decoder. Hence, as reported in [7], U-Net uses more memory than the SegNet

This section aims to review the modern, widely adopted methods in semantic segmentation. Architectures such as FCN, ParseNet, SegNet, U-Net and DeconvNet are considered. These architectures, despite showing high performance in segmentation tasks, cannot model uncertainty in their predictions. The next section reviews methods in segmentation that make use of uncertainty quantification.

2.2.2 Review of Uncertainty Quantification Methods in Semantic Segmentation

Previously, it was seen that segmentation methods can capture feature information from each input image and then later segment images on a pixel level. However, these methods do not account for the uncertainty in their predictions. Awareness of prediction is a vital task in mission-critical and decision-making systems e.g. autonomous vehicle navigation and medical systems. Uncertainty is an innate and natural characteristic that every DNN system should model. This is important for both developing trust and decision making in the semantic segmentation system.

DNNs are capable of mapping from data in high-dimensions to a low-dimensional output vector. However, the majority of the applications consider these mappings blindly. High accuracies on test sets can confirm good performance, but can also be misleading, as reported in [78]. This leads to disastrous consequences. Two vital ones are outlined below.

There have been real-life examples that have motivated the drive towards uncertainty modelling. For example, applications that have used segmentation in real-life applications have reported erroneous predictions and fatal decision making in the case of [179]. In this example, a perception system placed on a vehicle confused the white side of a trailer for the category bright sky. In the author's opinion, this is a typical example of class intersection and self-occlusion. This is also a major motivation for the technique outlined in this chapter which aims to use uncertainty to deal with class-intersection regions.

Another important example is in the case of [72]. Here, an image classification system in zoo-based security erroneously classified two African-American citizens as 'gorillas'. This leads to racial concerns in the social community, leading to the public questioning the use of AI.

Given the aforementioned incidents, AI systems must express uncertainty in their predictions. This is vital for reliable predictions and good decision-making skills. This is even more in demand in mission-critical systems. The question then of course remains, how can this be implemented ?. The approaches to modelling uncertainty in deep learning-based segmentation that aim to answer the aforementioned questions are reviewed next.

Firstly, the works of [14] and [60] of considered. These methods utilise the modelling of uncertainty in a non-deep learning setting. The difference in their aim is that [60] studies on the specific problem of automatic labelling of input images while [14] uses tracking of segmented contours. The work from [60] applies an algorithm that looks for contrasting information in the images and aims to generalise the conditional random field [21] approach to image labelling. It aims to model uncertainty to improve feature information so that the localisation of segmented boundaries is more precise. Work from [14], on the other hand, deals with uncertainty in a more systematic way using a sequential Monte Carlo sampling method. The aim is to apply a tracking of on a single segmented contour for gesture tracking.

In comparison, both works mention a performance improvement when modelling uncertainty. Particularly, [60] can obtain a consistent measurement of confidence that outperforms methods that explicitly model generative distributions between inputs and labels while [14] shows that modelling uncertainty allows their approach to account for distortions in input images from noisy labels. The limitations from both works are that [14] it is high computational complexity from matrix multiplication of observation matrices (these carry the segmented contour pose information) and [60] struggles with the requirement of contextual understanding within the input images.

On the other hand, deep learning-based methods for modelling uncertainty focus on two techniques that are heavily adapted in this field; Bayes By Backdrop (BBB) [15] and Monte Carlo dropout [41]. These techniques are popular in uncertainty aware segmentation literature as they provide a quick and easy solution to modelling uncertainty without having to change the architecture in operation. They both build on the idea of BNNs (see Section 2.1.11). BNNs consider distributions in place of weights. The other approach is to use Monte Carlo dropout (see Section 2.1.11).

In comparison, BBB focuses on placing prior distribution on each weight and updating the posterior of weights using backpropagation. Both Monte Carlo dropout and BBB scale with the number of layers. However, some works like [133] have tested that BBB has better robustness to adversarial attacks like FGSM (see Section 2.1.5) when compared with Monte Carlo dropout.

Although there are plenty of works in segmentation literature that leverage uncertainty, the most relevant ones that use BBB and/or Monte Carlo dropout are from the works of [68], [188] and [120]. Particularly, [68] uses temporal information from video-based input to model the Monte Carlo dropout simulations. They use the average output of a set number of consecutive frames to model the sample runs in Monte Carlo dropout. The movement of objects on consecutive frames is measured from the shift of pixels.

The work of [188], on the other hand, focuses on using uncertainty to model interpretability so that the model can understand the regions where inaccurate segmentation has taken place. Their work modifies both the architectures of FCN and SegNet by placing batch normalisation layers after each convolutional layer.

Similar in terms of motivation, the work of [120] also aims to use uncertainty for the detection of inaccurate segmentation regions. However, they employ Monte Carlo dropout and BBB at different initialisation. For example, they use multiple dropout rates (from 0.5, 0.3 and 0.1) and BBB with different variance priors (from 0.1, 1.0, 10 and 30). When modelling uncertainty, the aforementioned works agree on observing improvement in performance in segmentation. Additionally, [68] improves the speed of Monte Carlo dropout sampling by ten times (compared with 5 sample runs) and only requires to sample once to generate uncertainty. Furthermore, they both criticise BBB and Monte Carlo dropout in their works differently.

Some of the examples along this line include the work of [68]. It claims that both BBB and Monte Carlo dropout are impractical for real-time applications while [120] mentions that adding BBB to U-Net and the SegNet architecture increases the number of parameters twice as many times as adding Monte Carlo dropout.

In summary, though traditional methods are capable of stating uncertainty in predictions, it is challenging to model the same phenomenon in deep learning-based classification methods. This is because such methods only output vector based scores for each of the categories. One way to overcome this is to use Bayesian methods. On the other hand, modelling uncertainty has been proven to be more computationally expensive in deep learning-based methods [77] from the point of view of scalability. In the author's opinion, though traditional methods have the advantage of explicit modelling of uncertainty, such methods do not suffer from scalability issues as deep learning approaches do. This is especially a difficult scenario with BBB methods since they model uncertainty over each network weight. The next section specifically explores how to sample variance can be used to model uncertainty in weights in segmentation architectures. Specifically, discussing how this is achieved in the probabilistic extension of the SegNet architecture, the Bayesian SegNet [78].

2.2.3 Review of Modelling of Epistemic Uncertainty

Modelling epistemic uncertainty gives insight into how the model selection is performed in neural networks [79]. It captures the uncertainty in the parameters and can be explained away given more training samples [78]. However, it is also used in literature to explain inputs that are outside the training distribution [79]. This is ideal for detecting attacks or anomalies in the training set. Hence, it is a great tool in understanding why neural networks output erroneous predictions with high confidence. This section aims to discuss how epistemic uncertainty is utilised in the literature of segmentation.

Review of Modelling of Epistemic Uncertainty in Segmentation Literature

Modelling epistemic uncertainty has been accomplished through various methods in segmentation literature. The papers that are more relevant to this research are those that specifically use predictive variance to model epistemic uncertainty (Monte Carlo dropout (see Section 2.1.11)). These can be considered from the works of [161], [135], [158] and [88]. All of these works consider the Monte Carlo dropout technique or an improved variant of it.

Firstly, considering the examples of [161] and [135]. These works improve the Monte Carlo dropout technique by removing the need to sample as well as removing the need to generate an auxiliary dataset. They all use the predictive variance to model epistemic uncertainty. They also confirm that using epistemic uncertainty is ideal for detecting out-of-training distribution samples.

Another portion of works, [161], [88] and [135], observe an improvement in their results when modelling epistemic uncertainty. The main difference in their approach is that both [161] and [88] aims to model epistemic and aleatoric uncertainty using the predictive variance. To recap, epistemic uncertainty captures the uncertainty of model parameters. Aleatoric uncertainty, on the other hand, captures the uncertainty in the data [79]. In doing so, [161] is capable of accounting for both types of uncertainties. However, the work from [161] differs from the aforementioned methods, in it that they employ the use of Generator Adversarial Networks (GANs) (see Section 2.3.1).

The rest of the works apart from [161] model epistemic only. Specifically, [135] uses error propagation [178] technique to account for uncertainty. This involves holding the classes at test time and using noise injection and forward propagating the model to obtain the variance information.

On the other hand, [158] uses a post-processing scheme to relate labelling to epistemic uncertainty, essentially providing a solution to weakly supervised segmentation. This is because the work involves using a surrogate segmentation network to automatically label the out-of-training distribution samples.

In the case of [158], this network is a pretrained Bayesian U-Net. The Bayesian version of U-Net that uses Monte Carlo dropout to output predictive variance, very akin to the methods in the Bayesian SegNet [78], the Bayesian version of the SegNet. Both works use auxiliary training samples to reduce their epistemic uncertainty but the prior aims to automate this process while the latter uses a well-crafted noise injection scheme.

The advantages from each of the aforementioned works are that [161] and [88] prove that modelling both epistemic and aleatoric is more advantageous since each can be dealt with individually to highlight a different property of the model. This is beneficial in the application of medical imaging fields where model interpretability is in high demand [174]. This is also the case for both the works [161] and [88], although [88] is more focused on segmentation of medical images for stroke detection.

Furthermore, [161] benefits from the use of GANs for automatic selection of auxiliary datasets which allows their model to reduce the epistemic uncertainty. Reducing epistemic uncertainty has been argued by both the works in segmentation literature to improve the prediction from the segmentation network in operation (e.g. the SegNet or the U-Net) on samples that are out-of-training distribution. On the other hand, [88] claims the benefits from avoiding estimation of extra-parameters for the variance by accounting for both epistemic and aleatoric uncertainty. Then, [135] benefits from free sampling methods while [158] can generate training data with more relevance and with appearance availability by reducing uncertainty in their model.

However, despite the advantages in their approach, they have their shortcomings as well. Particularly, [161] states difficulty in training while [135] is unable to achieve performance greater than the state-of-the-art. Additionally, [158] is limited in performance from the assumptions that epistemic uncertainty will correlate from auxiliary training set similar to the original set and finally [88] suffers from limitations in uncertainty estimate which they reason to be because of using a restricted group of variational distributions. In the author's opinion, all of these works agree that modelling epistemic uncertainty improves performance in segmentation. Also, modelling predictive variance via Monte Carlo dropout is the easiest and the simplest method. Despite that, they also together agree that Monte Carlo dropout methods are not suitable for real-time applications. This is a little different in the case of [161] that leverages a GAN to simulate the training behaviour of an automatic out-of-distribution sampler. Which although may not serve well for real-time applications but does offer great improvements in the correct representation of uncertainty, as proved by the results of [161]. This rather brings the focus of this chapter towards an adversarial style of learning that has been recently adopted in the segmentation literature. They also are relevant to the methods proposed in this chapter. The purpose of this section is to discuss the importance of using the adversarial learning method in segmentation to the reader. It also forms the subject of the next section.

2.3 Review of Methods in Adversarial Learning

This section aims to introduce the field of adversarial learning in the context of segmentation applications. The section begins by discussing GAN based learning, the major underlying components, the training methods and literature that builds upon the standard GAN architecture. Then, the section moves to review various methods that use GANs in the application of high-resolution images, followed by a further discussion on the use of optimal transport in improving learning in GANs. This is continued with a review of various adversarial learning methods in segmentation, the pros and cons of each method. Finally, ending with the summary.

2.3.1 Generator Adversarial Networks

Earlier, it was mentioned that much of the success in deep learning has originated from achieving human-level accuracy in supervised learning-based tasks, such as classification. In particular, the success of discriminator models such as CNNs. However, DNNs have witnessed very little success in generative modelling.

The task of deep generative modelling involves replication of the input data distribution as well as learning representations from unlabelled data [48]. This task is more challenging than discriminative modelling. It entails that the model can approximate several distributions that are intractable. Many approaches [54], [82] have attempted at modelling distributions using probabilistic backpropagation with maximum likelihood methods.

GAN training circumvents the problem of learning for image generation using backpropagation, by adopting a mini-max game-based training approach. Mini-max game [23] is a strategy that aims at minimizing the maximum possible loss obtained from the player's move. In the analogy of GANs, this is accomplished using the loss function in equation (2.28) (taken from [48])

$$\underset{G}{min}\,\underset{D}{max}\,V(D,G) = \underbrace{\mathbb{E}_{x\sim p_{data}(x)}[\log D(x;\theta_d)]}_{to\ maximize} + \underbrace{\mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z;\theta_g)))]}_{to\ minimize}. \tag{2.28}$$

The aim is to train the generator function G to minimize the term $\log(1-D(G(z;\theta_g)))$ and the discriminator function D to maximize $\log D(x;\theta_d)$. The discriminator takes the input vector x and outputs a scalar which can be either '0' or '1'. The output '0' is obtained if the discriminator considers $x \sim p_{data}$ and '1' if $x \sim p_g$. Here, the terms p_{data} and p_g refer to the data and the generator distribution respectively. The terms θ_d and θ_g are the parameters of the discriminator and the generator respectively.

The generator, on the other hand, takes input from noise variables z and outputs an input vector $x \sim p_g$ of an image. The distribution $p_z(z)$ is used to define a prior on the input noise variables. The generator is then learns to define a mapping from noise to distribution p_g i.e. $G : z \sim p_z \to x \sim p_g$. The generator is said to have learned if outputs $x \sim p_{data}$ mimic samples from the data distribution.

The training is performed in alternating steps. The reason provided by [48], is so as to avoid overfitting if D is allowed to maximize in a separate inner loop. Instead, training is done jointly (both D and G), in a step-wise fashion. Firstly, the discriminator is optimised for k steps on m mini-batches of $\{x^{(i)}\}_{i=1}^{m}$ inputs and $\{z^{(i)}\}_{i=1}^{m}$ noise samples. This is shown in equation (2.29)

$$\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\Big[\log D\big(x^{(i)}\big) + \log\big(1-D\big(G(z^{(i)})\big)\big)\Big]. \tag{2.29}$$

The parameters of the discriminator and the generator are updated via backpropagation. However, gradients of the min-max loss function are ascended for the discriminator and descended for the generator. Additionally, the loss penalty for the generator's outputs only consists of the second half of the min-max loss. This is shown in equation (2.30)

$$\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m}\log\big(1-D\big(G(z^{(i)})\big)\big)\big]. \tag{2.30}$$

In summary, adversarial learning is an example of unsupervised learning. In this type of learning the system being trained is presented with a challenge by an adversary (called the discriminator). It penalizes the system for producing fake or undesired outcomes (labelled ‘0’ by the discriminator) as opposed to real or desired outcomes (labelled ‘1’). Learning is accomplished when the learning system successfully manages to confuse the discriminator so that it always predicts 0.5. However, the standard GAN architecture has limited resolution on the output window. The large resolution requires capturing of rich feature information. Especially for applications in semantic segmentation. The next section discussed the solution for this problem using the convolutional style approach.

2.3.2 Deep Convolutional Generator Adversarial Networks For Higher Resolution Images

This section aims to discuss the convolutional based architectures in GANs for outputting high-resolution images. Standard GANs are limited in their ability to output images of high resolution. Standard GANs [51] can maximum output 64x64 window size. The problem of generating high-resolution images is difficult. It requires learning complex representations from unlabeled samples. Additionally, as pointed in [76], it adds a bias in the training of GANs which is in the favour of the discriminator. The reason given is that high-resolution images are easy to differentiate from training images than low-resolution ones.

Furthermore, [124] reports that the problem of generating high resolution outputs “drastically amplifies the gradient problem” [124]. Tipping the min-max game in favour of the discriminator. Other minor problems are highlighted in [76], which states that large resolution training would constrain training to small mini-batches as such images would have high memory consumption.

Per-pixel segmentation requires understanding the semantics behind class labels for accurate results (see Section 2.2). High-resolution images are vital for preserving these features. Deep convolutional generator adversarial networks (DCGANs) [137] circumvent this issue using several architectural modifications on the standard GAN architecture. Some of these changes are discussed next.

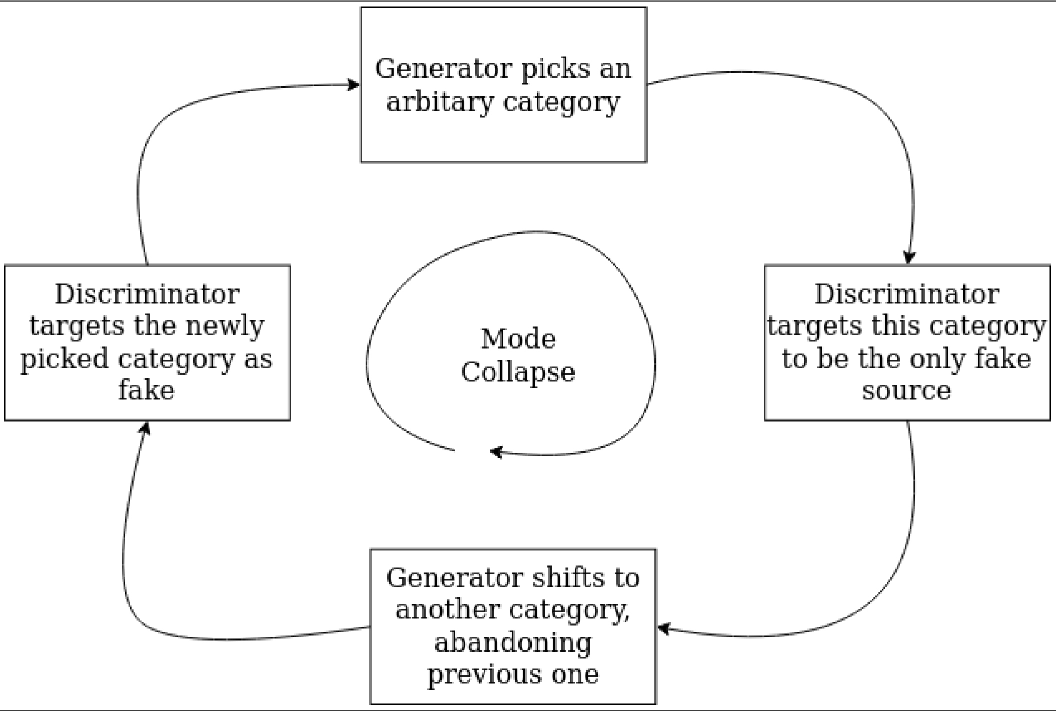
DCGANs focus on an all-convolutional approach. This trend has seen great improvement in CNN based image classifications as well, for example in the work of [171]. The only exceptions are the input layer of the generator and the output of the discriminator. These are strictly kept FC and sigmoidal respectively. The reason for prior is because of simple matrix multiplication in the generator input, the reason for latter is for binary classification.

In terms of pooling layers, DCGANs avoid having any form of spatial pooling functions. These include both average and max pooling. These pooling features are replaced with convolutional layers with a stride value set to an arbitrary value greater than 0. The purpose behind, as stated in [137], is to teach the GAN network to downsample correctly, to prevent loss of spatial information.

Another modification in the case of the DCGANs approach is the addition of batch normalisation layers. These are placed after every *convolutional block*. Here, the term *block* refers to a collection of layers that begin with a feature decomposition layer, a dimensionality reduction layer and then an activation layer. For example, in AlexNet [85], this block would be convolutional-batch normalisation-ReLU. In DCGANs, the purpose behind using batch normalisation, as stated in [137], is for remedying problems that can arise from poor initialisation of weights as well as assisting in gradient flow (i.e. continuous descent/ascent).

It also stated, in [137] that using batch normalisation prevents *mode collapse* [137] in GANs. Mode collapse refers to the phenomenon that occurs when the generator collapses all the samples to a single point. It is summarized in Figure (2.6). It occurs when the generator, at the start of learning, learns to fool the discriminator by outputting a single category of images. The discriminator responds by considering these as the only fake set. Then generator shifts to another category until the discriminator considers these as fake. This keeps repeating in a loop. This results in the lack of diversity of samples generated. The solution to modal collapse is ensuring that the generator does not stick to a single solution in the early stages of learning. According to [137], the batch normalisation layer ensures this by stabilizing the learning process via normalising each input to a block that has a zero mean and unit variance.

Fig. 2.6 The process of mode collapse during training generator adversarial neural networks


Generator picks an arbitary category
Discriminator targets this category to be the only fake source
Generator shifts to another category, abandoning previous one
Discriminator targets the newly picked category as fake
Mode Collapse

However, [137] warns that if the batch normalisation layer is kept at the generator's output layer and discriminator's input layer, it can result in model instability and sabotage learning.

This section reviews the architectural improvements proposed in DCGANs to deal with high-resolution images. First, the importance of dealing with high-resolution images is discussed. Then, some of the challenges involved in using GANs for this application. Various methods proposed by DCGANs to improve GANs is considered, as well as considering issues related to this, e.g. mode collapse. In the next section, further improvements in GAN training is considered from the point of view of loss functions and stability. It also introduces the GAN architecture that heavily inspires the proposed methods in Chapter 3, i.e. the least-squares GAN.

2.3.3 Least Square Generator Adversarial Neural Networks

This section aims to highlight the limitations of GANs. The section also aims to discuss the use of least squares GANs and how it builds improves the two major limitations of standard GANs: vanishing gradients and poor resolution of output images.

Although GANs have seen wide success in image generation [51] as opposed to other methods such as autoencoder [82] which require approximation of the real sample distribution, e.g. variational autoencoders use VI to approximate the distribution of real samples. GANs, on the other hand, can be trained end-to-end using gradient descent. Other methods would require the reparameterisation trick to overcome the non-differentiable stochastic nodes as opposed to differentiable deterministic nodes in standard DNN or CNN architectures.

Despite this, GANs are difficult to train. Several works have highlighted this issue from the point of view of the loss function [6], [107]. The standard GAN training involves minimising of the Jenson-Shannon divergence (JSD) loss function $C(G)$ (in equation (2.31)) between the real sample distribution p_{data} and the distribution of generated samples p_g. The equations for the loss function for both discriminator (in equation (2.32)) and generator (in equation (2.33)) are shown as

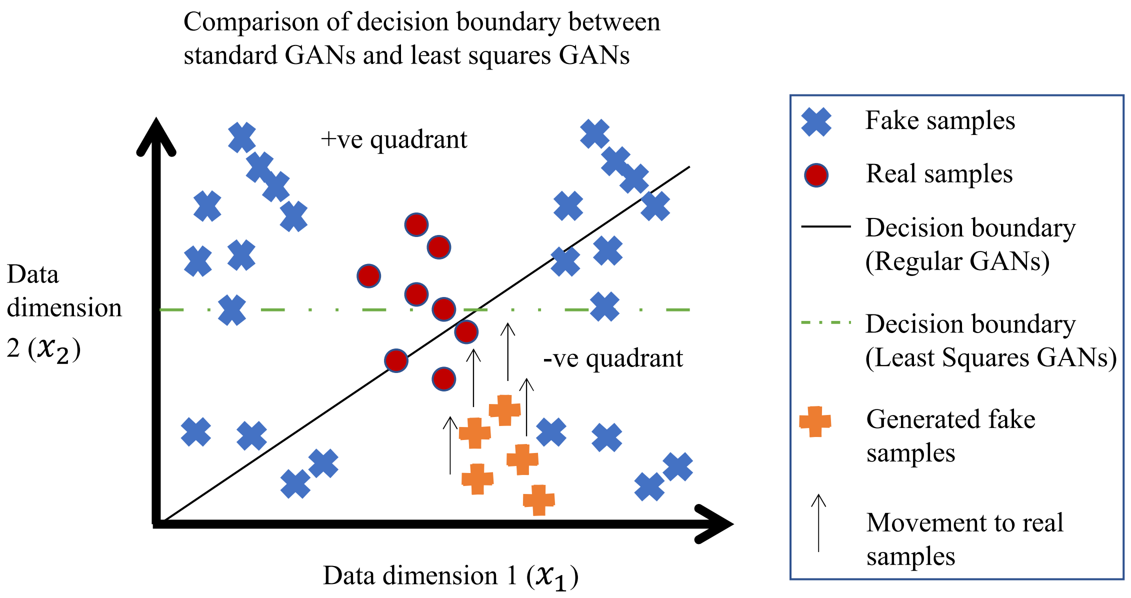
$$C(G) = KL\big(p_{data}||\frac{p_{data}+p_g}{2}\big) + KL\big(p_g||\frac{p_{data}+p_g}{2}\big) - \log 4, \tag{2.31}$$

$$\underset{D}{min} V_{LSQGAN}(D) = \frac{1}{2}\mathbb{E}_{x\sim p_{data}(x)}\Big[\big(D(x)-b\big)^2\Big] + \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}\Big[\big(D(G(z))-a\big)^2\Big], \tag{2.32}$$

$$\underset{G}{min} V_{LSQGAN}(G) = \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}\Big[\big(D(G(z))-c\big)^2\Big]. \tag{2.33}$$

KLD divergence has desirable properties such as being symmetric and always finite, as pointed in [6]. However, the problem in using this for training GANs is that it is not continuous and therefore will not provide usable gradients everywhere. Specifically, [6] shows that if one was to vary p_g from -1 to 1 for JSD $(p_{data}||p_g)$, then the graph lands (0,0) when $p_g = 0$. At this point, gradient computation is impossible and therefore learning in GAN will fail. This is also elaborated further in the work of [107] which shows that the loss function of JSD saturates for a large number of data samples. This is termed as the vanishing gradient problem in [6] and [107].

Fig. 2.7 Comparison of behaviours obtained from the loss function of both standard GANs and least-squares GANs. The fake samples are classified into the positive quadrant and the real samples in the negative quadrant. The solid line represents the decision boundary of the discriminator of the regular GAN and the dashed line for the least-squares GAN. The least-squares GAN penalises generated fake samples to move closer to real samples. Image inspired from [107]


Comparison of decision boundary between standard GANs and least squares GANs
+ve quadrant
-ve quadrant
Data dimension 2 (x_2)
Data dimension 1 (x_1)
Fake samples
Real samples
Decision boundary (Regular GANs)
Decision boundary (Least Squares GANs)
Generated fake samples
Movement to real samples

Another issue with GAN training, as highlighted in [107], is the lack of quality, from the point of view of both resolution and diversity, of outputs samples, generated. As the generator learns to produce fake samples that can fool the discriminator, the generated output samples may all look similar, which is in the case of mode collapse (see Section 2.3.2), or the samples may look completely different from the real samples. The reason behind this, as pointed in [107], is that during generator training, the fake samples are not penalised by the JSD loss function to be close to the real samples. This can be exemplified when comparing the decision boundary of the regular GAN and the solution proposed by [107], known as the least-squares GAN (LSQGAN) in Figure 2.7. The fake samples are categorised in the positive quadrant and the real samples in the negative quadrant. As the generator updates, it produces fake samples that are classified as real by the discriminator (i.e. in the negative quadrant). However, these lie far away from the decision boundary, far from the real sample distribution. The LSQGAN critic penalty function in equation (2.32) regresses the fake samples closer to the boundary decision line, i.e. closer to real samples. Hence, the quality of output images from the generator trained using LSQGAN is claimed by [107] to be superior to regular GANs [51]. The next section considers the use of GANs in segmentation literature.

2.3.4 Adversarial Learning in Semantic Segmentation

This section aims to review the literature on the use of GANs in segmentation applications. Previous sections have discussed the concept of GANs, how they can be utilised in problems of generative modelling. A convolutional extension of the GAN architecture has also been studied in the form of DCGANs for dealing with images with large resolutions. Then, improvements to GAN training in the form of Wasserstein distance have also been considered. The focus of this chapter now shifts to the more application side of things.

In the field of semantic segmentation, GANs have led to the development of supervised and semi-supervised (or weakly supervised) learning algorithms that either improve the segmentation performance or assist in producing labelled examples.

This section specifically focuses on a few notable examples that are close to the proposed architecture in the methodology section (see Chapter 3, Section 3.1). These works include [69] in weakly-supervised and [102] in a supervised setting. Both the works treat the segmentation network as a generator network and propose a coupling of adversarial loss with standard cross-entropy.

Furthermore, they propose a fully convolutional discriminator that learns ground truth label maps from probability maps of segmentation predictions. Similar to DCGANs, in terms of architecture. The only difference between their approaches is that [102] focuses more on the label quality and uses the adversarial framework as a supervisor for improving the accuracy of the segmentation network, while [69] proposes a semi-supervised setting where the prior framework adds additional input images without labels, thus increasing segmentation accuracy. This also avoids the manual construction of dataset samples.

The work adopted in this chapter follows a similar approach to [102]. The difference is that Bayesian methods to output uncertainty are utilized and then used as part of adversarial learning to teach the network to learn to deal with uncertainty. Specifically, two discriminators are used; one penalizes the segmentation network for output labels that differ from ground truth (QC) and the other penalizes if there is uncertainty in the prediction (UC). The next section focuses on literature relating to the robust learning area tackled in this thesis.

2.4 Review of Robust Learning in Artificial Intelligence

This section aims to introduce the idea behind robust artificial intelligence systems. Specifically, robustness against noise and adversarial attacks. The past decade of AI research has witnessed a substantial leap in the progress of algorithms that can achieve near-human level performance. This is witnessed, for example, in the field of computer vision where architectures like AlexNet [85], VGG [164] and other CNN architectures such as ResNet [59] have achieved performance closer to human vision. Furthermore, a trend observed by [151] shows a drop of top-5 error rate from 28.2% in 2010 to 6.7% in 2014. A similar trend has also been observed in the application of speech and language processing. A popular example of this is the evaluation of the Bilingual Evaluation Understudy (BLEU) [130], scores used to evaluate machine-translated texts, which has been reported by [160] to have risen from 14.6 in 2007 to 23.6-24.7 by 2014.

Despite the achievements, the works of [31] and [108] in AI research have reported a new design requirement for the building of future AI systems. This requirement is robustness and interpretability. The requirement has risen after the applications from the previous decade have reported outright flaws in procedures and the transparency of their thinking abilities (as seen in Chapter 2, Section 2.2.2). Specifically, invoking questions like how does the AI system learns the problem ? Why does it give erroneous predictions ? Why does it fail with small changes in input despite showing near-human performance on the evaluated datasets ?.

A major criticism has been remarked from areas of situational awareness [181], mission-critical [181] and/or medical surgery [162]. A common point linking the aforementioned applications is their similar design requirement (or desideratum). The requirements are that the system on-site must be able to carry out swift and rational decisions in a short period. These applications also have high-stakes at hand, with consequences deciding between life and death. Take for example the case of robot-assisted medical surgery in the work of [162] and military coordination in [181]. In such cases, a wrong decision or delayed response can have fatal outcomes, e.g. a faulty stitch from the robot surgeon or an incorrect command issued by the drone responsible for reconnaissance in military outposts.

Further addressing this issue are the works from the AI research in [31], [181], [4], [108] and [56] who have all acknowledged that though the performance of AI has seen substantial improvement, the quality is not on par with rational human decision making. Such systems lack the capability of correct perception, reasoning and plan execution. This is notified critically by the remarks from [31] who have advocated the regulation of autonomous weaponry systems to prevent fatalistic effects in the coming future of robotics technology.

Apart from the design requirement from high-stakes applications, there is also exists the issue of adversarial attacks [128], [115], [173], [49]. These attacks also motivate the development of robust AI systems. This is because deep learning has been proven in literature [129] to be sensitive to small changes in the input that may be imperceptible to human beings but is sufficient to confuse the system.

Given the aforementioned reason, the research in AI has adopted a new trend in developing AI systems that are of broader intelligence as opposed to narrow intelligence, termed in [108]. The motivation now is to force AI systems to be capable of extrapolating their knowledge to samples outside the training distribution, Furthermore, systems should be robust to minute and imperceptible changes in the input. Given the key concepts of robustness, the next section considers the definition of robustness in both a formal and informal manner. This forms the subject of the next section.

2.4.1 Defining Robustness

This section aims to formally define robustness in AI as stated in the literature [31], [4], [61] and [187]. The section begins with reviewing three different definitions of robustness in the AI literature: a generic and a specific view-based definition and a formal version that uses the Lipschitz metric [62]. Then, a visualisation of the definition of robustness from the work of both [61] and [187] is briefly reviewed.

The previous section discussed the key concepts behind robustness in AI. In literature, there are several definitions for robustness. For example, [108] considers robustness to be a quality of a model that considers intelligence from a wide perspective, as opposed to a narrow one. According to [108], a narrow system is capable of performing a single goal, vice versa for the wider case. The wider intelligent systems do not require extensive retraining.

When considering robustness, the work by [108] considers the narrow intelligent system to be less robust to wider systems. They also lack the capability of transferring knowledge to other models and output erroneous predictions even with modest changes in inputs [108]. From this definition, it can be seen that a vital component of robustness is consistency in predictions.

This form of definition is also known as the globalist definition of robustness [56]. It gives a generic view of robustness. Alternatively, a more specific definition of robustness is defined in the works of [61] and [187]. This defines robustness as the "variation of prediction with respect to the changes in the input leading to that prediction" [56]. However, this definition does not specify the meaning of the variation information. A formal definition of robustness studies the meaning of variation in-depth and is discussed next.

2.4.2 Formal Definition of Robustness

The formal definition is proposed in [4]. This definition is based on the variation of the predictions of the classifier. According to this definition, a robust classifier must be able to predict correctly despite modest changes being made to the input. These modifications slightly perturb the input to the classifier. According to this definition, the outputs of a robust classifier should not be altered in the presence of such small changes to input.

Furthermore, to test the stability of the classifier function, this definition also studies the continuity of the classifier function. These works define robustness based on functional stability. Specifically, in the works of [4], [61] and [187] use the Lipschitz criterion [62] to assess the stability of the classifier function. The Lipchitz metric originates from the study of uniform calculus and real analysis [62]. It is also studied in fields outside of AI, e.g. for readers with a background in fluid mechanics, the Lipschitz continuity is used to study airfoils in [45]. However, in the context of robustness, it is used to define the formal definition of robustness as shown below.

Definition 3 *Given a function, f that maps an input space in real dimension n to m dimensions, such that $f : X \subseteq \mathbb{R}^n \to \mathbb{R}^m$. This function is locally Lipschitz if for a vector x_0 there exist a constant $\delta > 0$ and a constant $L \in \mathbb{R}$ such that the norm of the difference between an arbitrary input point x and x_0 is bounded by δ, i.e. $\|x - x_0\| < \delta$, and that the norm of the difference of the neural network output functions $\|f(x) - f(x_0)\| \geq L\|x - x_0\|$ must not exceed the Lipschitz bound L based on the norm of the difference between x and x_0.*

Here, x_0 is the focal or the anchor point from which the changes in the input space are measured. The quantity δ denotes the amount of perturbation to the input x while L defines the Lipschitz bound. The term δ decides how much the adversarial image should resemble an arbitrary input image x_0.

Remark An important point to note is that both the quantities x_0 and L are pre-determined. Since x_0 is seldom a known prior (i.e. initial value is unknown), it needs to be estimated along with L separately.

Given the above definition, it can be seen that defining the robustness of f can be quantified in terms of the bound L on the inputs. Adversarial examples are only classified incorrectly if they lie inside this bound. Outside this bound, they are classified as anomalies. The problem of course is then how to define these bounds ?, as well as what distance measure to use to differentiate both outputs $f(x)$ and inputs x from adversarial samples ?. The requirements can be listed in the following way.

Robustness Requirements

- Defining the bounds on input L.
- Defining how much the adversarial images should resemble the input images by δ.
- Defining the anchor x_0, a representative set of input samples from which the difference between the adversarial images are measured. The x_0 must lie within the decision boundary line to ensure it is a correctly classified example.

This is essential for generating adversarial attacks as well as for deciding robustness either through accuracy measures or optimisation of bounds, as in [4]. The methods in this chapter use accuracy measures to define robustness.

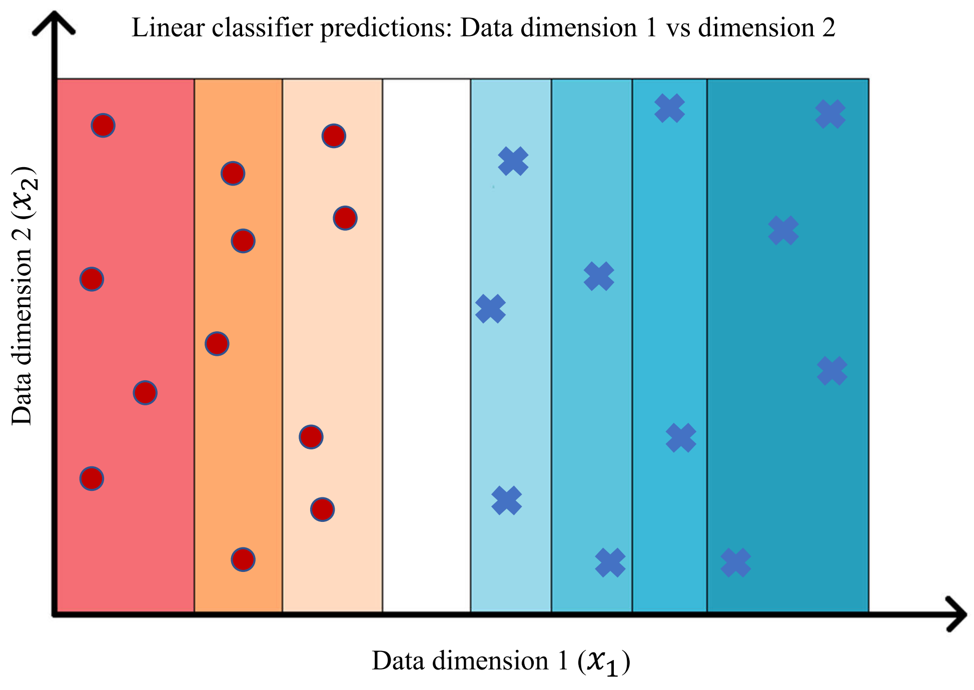
Though it is difficult to estimate L at first, several papers adopt different approaches. One approach from the work of [4] involves taking the quotient of the l_2 norm of the deviation between the outputs and the l_2 norm of the deviations of inputs. The problem with this approach is that it involves computing distance for each of the data samples. This can be computationally expensive for large datasets, e.g. Sun RGB-D [169]. Other distances include l_1 or l_∞ norms.

Specifically, the proposed framework in this chapter generates adversarial examples by selecting and randomly ranking samples from the dataset. Then, using the Euclidean distance between the feature outputs based on the original mini-batch and the feature outputs from the selected ranked batch, the top 10 closest neighbours with respect to each sample are randomly selected and replaced with the samples. This is done for the entire mini-batch until a new mini-batch is made up of neighbouring samples. This technique is inspired from [94], except they used this to generate noisy labels, the work in this chapter focuses on input images. The evident advantage of this method is that the bound L and the anchor point x_0 need not be defined explicitly. Each mini-batch sample acts as x_0 and L is not needed since nearest neighbours are used. In brief, the l_2 norm between the neighbourhood and original features implicitly defines L and the randomly selected samples define the term x_0. It will be later seen how this method of automatically learning to be robust by replacing samples of mini-batch with nearest neighbours is convenient. It avoids the difficulty of having to manually invent an attack, as this is outside the scope of this thesis.

2.4.3 Figurative Explanation of Local Robustness in Linear and Non-Linear Classifier

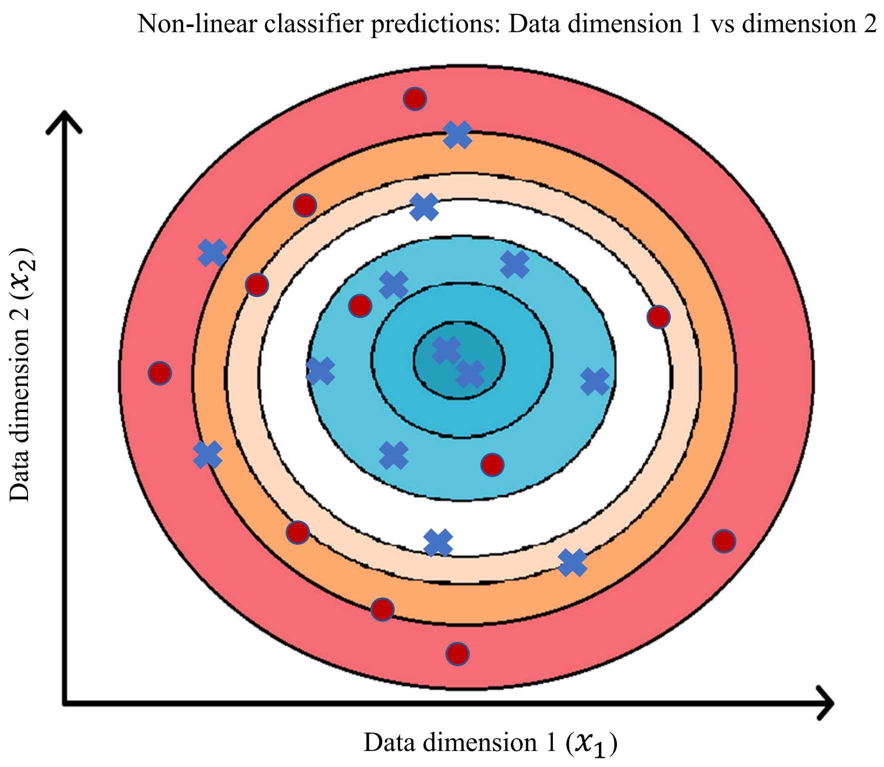
The definition of robustness can also be diagrammatically explained, as studied in the work from [145] and [103]. This work provides a qualitative description of how robustness can be locally defined using the predictions of a linear classifier in Figure 2.8 and a non-linear classifier in Figure 2.9. Both systems are trained on a 2D dataset of two classes. Class 1 is represented as red circles while class 2 is represented as blue crosses.

Fig. 2.8 Predictions from the linear classifier provided in the works of [103] and [145]. Image re-printed from [4]


Linear classifier predictions: Data dimension 1 vs dimension 2
Data dimension 2 (x_2)
Data dimension 1 (x_1)

From the figures, it can be seen that the darker shades of the colour bands define how distant the predicted category is from the decision boundary represented as the blank white region. Given that the dataset is 2D, the x-axis plots the first dimension and the y-axis plots the second dimension of the data. Observing the figures, the linear correctly classifies more samples than the non-linear classifier. Having the correct representation of data and correct decision boundaries can affect the robustness of the classifier. Additionally, considering the case of the non-linear classifier, if one was to consider any point in the boundary line as the anchor, then as the distance away from the boundary increases, the predictions begin to vary and are more inconsistent with each other. On the other, the predictions from the linear classifier are very stable away from the decision boundary.

Fig. 2.9 Predictions from the non-linear classifier provided in the works of [103] and [145]. Image re-printed from [4]



In summary, the key messages that can be taken from this section are that robustness can be defined in both a global and local sense. Robustness can be defined for a minute and imperceptible changes in the input. This is an important property in a robust AI system. Furthermore, the figurative explanation of robustness gives the message that for a classifier to be robust, it must have an accurate and well-defined decision boundary that *distinctly separates* dataset points. However, it is difficult to design robust systems. This is discussed further in the next section.

2.4.4 Challenges in Developing Robust Artificial Intelligence Applications

This section aims to introduce various challenges involved in the development of robust AI systems. The previous section discussed the definition of robustness both in the global and local settings. It was discussed that being consistent in predictions while dealing with minute changes in the input is an essential requirement for robustness.

Designing robust systems entails plenty of challenges. Some of these are highlighted in [31] and are briefly reviewed in this section. The highlighted challenges include tackling human user error, adversarial attacks, misspecified goals, incorrect models and unmodeled phenomena.

Considering first the human-user error, an example for this is provided in [31] for the case of a robotic surgery and weapon automated system. Here, the *human in the loop* system may commit an error e.g. out of oversight or inconsistency which can result in a wrong surgical action or false weapon deployment in the case of the weapon automated system. Therefore, in this case, it should be vital for the system to be robust to human-user error.

Next, considering misspecified goals, these can arise from conflicting requirements between the original intended AI system design goals and the end-user goal. An example of this is provided in [31] as commanding a self-driving car to “get to the airport as soon as possible”. One possible fatal outcome of such command can be that if the system such as the self-driving car interprets it without reasoning the outcome, then carrying the command may result in breaking of laws, injuring of pedestrians and/or the driver. To avoid this scenario, the system must be able to account for aleatoric uncertainty for rational decision making. In doing so, the system must take into account both the human laws and the laws set for the autonomous system. On the other hand, it is worth noting that if the laws do not take into account aleatoric reasoning, the result can be catastrophic and may lead to the case of misspecified goals.

Next to consider is incorrect modelling. Such examples are the exact opposite of misspecified goals. These can occur if the epistemic reasoning (reasoning based on knowledge) of the system is incorrect. This is caused when the AI system is not robust to errors within the model it builds of the real world. Consequently, the system can generate a prediction that is outside its actions. For example, the case of uncertainty in the positional data in mobile robots can lead to errors in precision [31].

Lastly, unmodeled phenomenon challenges arise if the system is not able to account for the knowledge needed to solve a problem. This is different to incorrect modelling since in that case the knowledge is present but incorrectly models the world. On the other hand, error because of the unmodeled phenomenon is more difficult to deal with since not all AI systems are modelled incorporating knowledge of everything in the world. If this was true then the model would be too complex for the problem it needs to solve. Additionally, it will require large amounts of the training set. This balance of model complexity and training dataset decides the error in decisions [31]. It is formalised in [31] in the equation (2.34)

$$error\ rate \propto \frac{model\ complexity}{training\ datasize}. \tag{2.34}$$

From the equation it can be seen modelling every aspect of the world would increase the complexity of the model and this, in turn, would take an extremely large amount of dataset to compensate. This of course is limited by computational expenses as well.

This section explores the various challenges highlighted in the AI literature [108], [31] in designing robust AI systems. These include adversarial attacks, human user error, unmodeled phenomena, misspecified goals and incorrect modelling. This section does not include detail for adversarial attacks, however, Section 2.1.5 discussed a common type of white-box attack, i.e. the FGSM. In the next section, it is seen how uncertainty and meta-learning can be used to achieve defense against such attacks.

2.4.5 Uncertainty and Meta-Learning Based Methods For Adversarial Robustness

This section aims to discuss methods in uncertainty and meta-learning for adversarial robustness. Uncertainty is a valuable tool in highlighting confidence in predictions. The future trends in AI technologies aim towards endowing robustness and interpretability to DNN architectures [108]. Thus, having an interplay of robustness and interpretability would be advantageous from the standpoint of AI safety and establishing trust in AI technology.

The fields of uncertainty based learning and meta-learning can be used to explain the link between interpretability and robustness. Briefly, uncertainty based learning focuses on the interpretability of AI systems and meta-learning allows multiple learning objectives to be learned at the same time. In this thesis, the additional objective is robustness and the proposed solution in this chapter uses regularisation to enable this objective as a separate training step (see Section 3.2). This section begins with a brief introduction to both fields. It then provides a survey of the various techniques, the pros and cons of each and finally end with a summary section. This section also highlights ways in which robustness can be linked to the interpretability of DNNs. This can be summarized as follows.

Linking Interpretability to Robustness

- Uncertainty allows models to be interpretable
- Uncertainty can be used as a tool to highlight the impact of adversarial attacks and/or noisy images
- Uncertainty can be used to explain the robustness against attacks with different strengths

Learning from uncertainty is an actively developing field in Bayesian deep learning. It is practised in many forms under different fields. The most widely adopted ones are Bayesian deep learning and meta-learning [155]. Meta-learning treatment of uncertainty-based learning consists of recognizing that learning from uncertainty is a *meta* step (an additional objective) that operates along with the standard gradient descent step. Meta-learning also forms part of continual learning.

In literature, meta-learning is applied to semi-supervised tasks, carried out in noisy, uncertain conditions. In this type of learning, the dataset comes with limited labels. This can be seen in the works of [177] and [94]. The difference in their approach is that [177] adopts a global averaging scheme on DNN weights as a means of modelling noise in the labels, while [94] generates an external noise model and a student-teacher learning scheme (as in [195]) to teach their network to be consistent in predictions. Other methods along these lines [195] adopt a similar strategy by using consistency as a means of teaching reliability. However, these methods involve implicit noise generation which requires alteration of DNN architectures. They are also difficult to scale in terms of the number of layers, whether it is convolutional or fully connected.

In uncertainty based learning, the emphasis is given to the correct use of the variance information. This variance information is then utilised for learning in the presence of adversarial attacks. A few notable works in literature include [181], [144] and [106]. Considering the work from [181], it emphasises the relationship between interpretability and robustness to adversarial attacks. Their approach is to use saliency methods to determine whether uncertainty is a good measure of attack strength. They test their results on CWA attacks. Their results show that saliency maps do not provide good measurements of attacks. The author claims that they are not sensitive enough to detect the attacks. However, the author concludes by foretelling that other more robust Bayesian methods in DNNs may provide a stronger relationship between explainable AI and robust AI.

An improvement on the approach by [181] is carried out in the work of [144]. The method likens adversarial perturbations to random Gaussian perturbation (or Gaussian noise) and provides evidence of a clear relationship between increased attack strength and the increase of model uncertainty (or epistemic uncertainty).

The work of [144] also compares detection methods from both MC dropout, standard DNNs and prior networks. For a quick recap, MC dropout is a Bayesian deep learning technique that uses dropout layers to obtain uncertainty. Prior networks, on the other hand, generate distribution implicitly as learning representations [106]. Results show that MC Dropout and standard DNNs behave similarly to both black-box and white-box attacks. Both are good at detecting FGSM attacks but are ineffective against black-box. Prior networks are found to be robust against FGSM attacks but perform worse on black-box ones. Furthermore, it is proven that standard DNNs perform best under adversarial learning objectives.

On another side of the application, a further study in the use of uncertainty to mitigate the effects of adversarial attacks is carried out in the works of [152], [165] and [57]. These works aim to reduce the effects of adversarial attacks. Major differences between their approaches are that [152] uses a GAN to train their main network to resist attacks while [165] and [57] uses Bayesian methods. Specifically, [165] uses softmax variance to account for uncertainty while [57] uses MC dropout. GAN methods don't discriminate between black-box or white-box attacks. Therefore such methods are flexible and applicable to any form of classifier. MC dropout, on the other hand, can scale well with network architecture.

In [165] the softmax variance is an approximation to the measure of mutual information [104]. Mutual information measures the KLD between the joint and the marginal distributions of two random variables. A random variable is a function that maps an event to the real space. Comparing this with sample variance obtained from MC dropout, it is shown that mutual information accurately represents uncertainty in predictions in the presence of attacks.

The drawbacks of these approaches are that GAN based methods are difficult to train since they involve optimizing two separate DNN models. These are the discriminator and the generator. MC dropout is very slow at uncertainty computation due to continuous sampling of variance and the quality of uncertainty measure is also dependent on the sampling rate. Another important factor is the issue of calibration. Both GAN and MC dropout methods poorly calibrate uncertainty as opposed to the slightly better softmax variance.

There also exist a line of papers that focus on placing distribution on each layer and propagate the probabilities across all the layers of their respective DNN architectures [150], [189], [64], [42], [29], [17]. For brevity, these can be termed as density propagation methods [29]. They differ from the aforementioned ones as those only consider placing distributions on outputs or using sampling-based methods. These methods are not reliable and cannot generate well-calibrated uncertainties, especially those that involve the use of softmax probabilities [41]. They also have the issue of scalability. It can take longer to run large architectures (e.g. > 10 layers). The advantage of using density propagation methods is that they inherently compute expected predictions along with the uncertainty estimates. These are computed at the outputs for every layer. This is beneficial from the standpoint of computation since standard Monte Carlo based VI methods require continuous sampling.

Some methods use Bayesian methods for propagating density functions. To recap, the goal of Bayesian inference in BNNs is to infer the posterior distribution over the weights of the neural network. There is also the need to compute the expectations w.r.t posterior. In the earlier sections, it was seen that VI methods can be used to approximate posterior distribution in BNNs. Despite it has been shown in the literature that this can be scaled to large architectures [52] such as ResNet [59], certain limitations render the practicality of VI unfeasible. Although these have been discussed in the previous sections, the below paragraph provides a quick recap.

First, the variational loss in equation (2.13) (see Chapter 1) is intractable. It does not have a closed-form analytical solution. Solutions often rely on approximations such as the reparameterization trick (see Chapter 1, Section 2.1.9). However, these results in approximations of gradients that are stochastic (i.e. high variance). Secondly, the KLD term in the variational loss does not have a close form analytical solution except for Gaussian priors [150]. This limits the choice of the prior distribution and also adds on a heavy computational budget [189]. Thirdly, the VI approach is very sensitive to the choice of prior [117].

The limitations of methods in density propagation [150], [189], [64], [42], [29], [17] can be discussed as follows. Firstly, [150] focuses on obtaining a closed-form solution to the KLD term while [189] aims to develop methods for automatic selection of prior variance via empirical means. From the point of view of the author, both [189] and [150] attempt at approximating the expected log-likelihood (observation likelihood), but the performance of [189] is much better since it is capable of completely ridding VI from stochastic gradients. They both obtain closed-form solutions by propagating first and second moments (i.e. mean and variance). However, [189] extends its propagation scheme to both ReLU and Heaviside activation functions while [150] limits to ReLU. Then, moving to consider the works of [64] and [42]. The difference between their approaches is that [64] considers both the forward and backward propagation of probabilities while [42] only considers the forward propagation. The advantage of each of the approaches is that [64] is capable of producing reliable uncertainty measures but [42] is faster. This is because they avoid adopting fully Bayesian principles and use restricted distributions to parametric families to place probabilities on all layers. In doing so, they can achieve results relatively faster and their system is also proven to be robust to adversarial attacks. Specifically, they test their algorithm on FGSM attacks. Finally, there are the works from [29] and [17] which use a tensor normal distribution to propagate their probabilities from one layer to another. They extend upon the methods by previous solutions to convolutional layers and can achieve robustness in both Gaussian noise and FGSM attacks. They further improve upon their work in [17] and use a robust Kalman filtering technique to generate ensembles of their moments through all the layers of a general CNN architecture.

This section considered various methods in uncertainty based solutions for adversarial robustness. The section was divided into three parts; the first portion considered the introduction to the field of meta-learning, the second portion discussed techniques that build upon variational methods while considering disadvantages of these methods, the final portion discussed techniques that use density propagation methods and how these improve upon variational techniques. The next section will introduce the readers to the literature relating to the third subject area tackled in this thesis, i.e. continual learning.

2.5 Overview of Continual Learning

This section aims to review the field of continual learning. Continual learning aims to provide solutions in key areas of machine learning where tasks are presented consecutively. The motivation behind this is that real-world tasks continually evolve and the size of datasets often prohibits frequent batch updating. This is a difficult objective in machine learning since many learning algorithms rely on batch-wise learning of the entire dataset. Real-life applications often involve solving a set of related tasks that need to be jointly handled, e.g. in learning multiple tasks consecutively. This is an issue that can become problematic as AI technologies become more and more prominent in applications worldwide.

Neural networks can be vulnerable to catastrophic forgetting [38]. A phenomenon in which a substantial drop in performance is observed when a network is presented with a new task. This is true whether the new task resumes a portion of the current operating dataset or switches to a completely new one. Continual learning (CL) targets solutions to deal with forgetting.

The past two decades of AI witnessed a rise in algorithms that can learn from stationary datasets and perform with near, or in some cases beyond, human-level accuracy [85], [59]. A further study on the vulnerabilities of DNNs, in the form of erroneous predictions and sensitivity to imperceptible input changes, has led to several works in research advocating the importance of robustness and interpretability [108], [31].

In the real world, data comes in a dynamic, stream-like format. This is a huge issue for DNNs because, apart from vulnerability to imperceptible attacks, DNNs have been reported to be only capable of learning static or stationary datasets [38]. Consequently, they may only be able to operate on data streams temporarily, becoming obsolete as soon as the data stream is changed. From the standpoint of cybersecurity, continuous hacks and attacks can contribute to the dynamic changes in data streams. This can worsen the situation.

When solving for continual learning, it can be helpful to take inspiration from learning dynamic methods. Learning in the human brain is one example of continual learning. In literature, [38], a study on dynamic learning in the brain shows that learning concepts sequentially is an inherent phenomenon in the human brain. However, literature also shows that even the brain can suffer from catastrophic forgetting. For example, a study by [127] shows that a group of adults of Korean descent underwent a shift in the environment once adopted by French families. The analysis of their brain's linguistic capabilities showed that their behavioural pattern did not discriminate regardless of the language they were presented with. Thereby, approaching the conclusion that there is an absence of residual knowledge as a result of catastrophic forgetting.

Despite this, studies have shown that though the human brain gradually forgets old information, a complete loss of all of the previous knowledge is rarely observed [38]. This, on the other hand, is not the case with DNNs. DNNs cannot learn in this manner. As they are presented with the new data (knowledge), learning by backpropagation completely overrides knowledge of the previous data. One may even argue that this can be the reason why some fine-tuning techniques require the freezing of earlier layers [85], [59]. To study this in detail, CL aims to investigate ways to circumvent this problem. Hence, the next section aims to give a brief overview of this field and the types of techniques adopted.

2.5.1 Defining Continual Learning

This section aims to give a brief introduction to the field of continual learning, the underlying problem, how it is approached, the concept of stability-plasticity phenomenon and some examples of the types of approaches in CL. Particularly, those dealing with regularisation. Continual learning is defined as the type of learning in which the learner acquires knowledge from an infinite stream of data/tasks [28], [193]. Continual learning is also known with various monikers such as lifelong learning and continuous learning [22]. Meta-learning is also considered as a type of continual learning [155] (see Chapter 4, Section 2.4.5).

The main challenge of CL is that the learner must be able to acquire knowledge gradually. It also must be capable of extending this knowledge to tasks incoming in future. Additionally, it should not suffer from catastrophic forgetting [38]. This requires that the learning system retains knowledge from previous tasks. Catastrophic forgetting is observed when there is an abrupt drop in performance on previously learned tasks at the onset of receiving a new task.

The vulnerability of neural networks to dynamically changing datasets can also be explained as a result of the stability-plasticity dilemma [53]. The term *plasticity* refers to the ability to integrate a vast amount of knowledge and the *stability* of the acquired knowledge refers to its conservation [53], [111]. This phenomenon explains forgetting in DNNs in two folds; a) it establishes as a constraint for distributed learning systems, b) highlights the weakness of overlapping internal representations as a direct consequence of the excess of stability. It is challenging to achieve a balance between adapting to recent data and retaining knowledge from old data. Too much plasticity leads to catastrophic forgetting and too much stability leads to an inability to adapt to new knowledge. Consequently, the ability of DNNs to train on a continuous stream of data can be constrained by their plasticity and stability. Furthermore, [37] suggests that DNNs have highly distributed internal representation, resulting in overlapping patterns of activation at hidden layers. This causes override of previously attained knowledge at the onset of receiving new information.

Several research works have aimed at circumventing the issue of forgetting in DNNs. Some target the overlapping internal representation problem using dynamically shifting representations, e.g. in the work of [37]. Also referred to as parameter isolation methods [28]. Others retain portions of training set whilst training new tasks [163], [147]. This is also referred to as replay methods [28].

Finally, some methods make use of regularisation techniques. Such methods focus on constraining the weight update rule. The underlying idea is to encourage learning new concepts while learning on the current task. At the same time, retaining and consolidating prior knowledge and memory. The advantages of these methods are that they do not require storage of raw inputs. This alleviates the problem of memory consumption to a certain extent. Regularisation methods are also flexible since restrictions can be placed on inputs or loss functions. In literature, regularisation methods can be sorted into two types; data-focused and prior focused methods.

Data focused methods work on refining knowledge gained from previous models trained on former tasks. In literature, this is also known as knowledge distillation [65]. The first solution in CL literature to use distillation is from the work of [163]. This involves using previous model outputs to consolidate knowledge of the model acting on new inputs. This work is further improved by [95] which aims at using outputs of the previous model trained on the prior task as labels. This requires that network outputs are stored separately.

The disadvantage of this approach is that storing knowledge from previous model configurations can consume large amounts of memory depending on the number of tasks. Further disadvantages of this approach include; a large overhead for forwarding combination of new tasks and gradual build-up of errors to the prior tasks if the relationship between the new and prior task is not strong.

An improvement in the direction of better distillation and consolidation can be seen from the works of [74] and [194]. The difference in their approaches is that [74] works with using all of the datasets and maintains the original feature space of the source domain (i.e. the domain of the dataset). It is more applicable to popular gradient descent techniques. On the other hand, [194] works similarly to [74]. The difference in their approaches is that [194] performs training in three steps; first, they deploy a separate model that trains only on the new classes, secondly, they combine their two individual models (one trained on previous tasks and the other on the current task) by training them jointly using a double distillation training objective, then finally they consolidate both their models using stored auxiliary dataset. Samples in the auxiliary dataset have rich and robust feature representation which best describes what is required to learn the task.

The main idea behind the double distillation approach by [194] is that this type of distillation allows their model to learn from two separate pretrained ones (termed teachers). This has two major benefits over other methods like [74]; one is that it helps get rid of intrinsic bias caused by over-regularisation and secondly, it improves the generalisation of the learning system. Regardless, both the works [194] and [74] are capable of using their algorithm in the presence of the entire training set.

This section considered the definition of continual learning while exploring the challenges and various methods in combating catastrophic forgetting. In particular, the regularisation methods were emphasised. These methods focus on additional constraints on weight updates as well as inputs. In the author's opinion, these methods generally aim at using regularisation based on the inputs and either use distillation or penalise inputs directly. However, regularisation methods are promising since they provide solutions for overfitting. It is difficult to achieve the same with other methods that require memory storage consumption. In the next section, a more Bayesian approach to regularisation methods is explored.

2.5.2 Bayesian Methods for Continual Learning

This section aims to focus on methods that use Bayesian approaches to CL. The aforementioned techniques in the previous sections focused on weight parameter based loss or input regularisation. Though promising, such techniques often require careful crafting of loss function penalties. This is done so to avoid overfitting [19].

An alternative approach to regularisation is to adopt Bayesian methods. This section will begin by discussing the choice of Bayesian approaches in CL. It then proceeds to discuss how the Bayesian methods can act as regularisation based solutions to continual learning. It also briefly recaps Bayesian learning. Finally, it reviews two of the most popular techniques used in Bayesian CL; variance guided continual learning (VCL) [121] and the Uncertainty Guided Continual Bayesian Networks (UCB) [32].

The main idea behind CL is that the model should keep itself up to date with the changing data. The concept of updating weights while observing data changes is similar to learning in Bayesian methods. Bayesian learning is an exemplary candidate for continual learning. To recap, Bayesian models, update using the distribution over model parameters (i.e. the posterior). When the new data is encountered, the combination of what is learned from the previous data (based on the old posterior) and what is learned from the current data (based on the observation likelihood) is used for updating the posterior. The downside of Bayesian approaches is that the posterior inference is intractable for large datasets and complex distributions. In literature, several approximations can be used. These include VI [52] and Laplace approximation [166]. VI has been briefly reviewed in earlier chapters and Laplace approximation is outside the scope of the thesis. The next subsection will review two of the most recent techniques in continual learning that inspire from both VI and Laplace approximation, i.e. the VCL and UCB. First, the use of VI approximation as a regularisation based solution in continual learning is discussed.

2.5.3 Variational Inference as a Regularisation Based Solution for Continual Learning

UCB and VCL were inspired by Bayesian methods in continual learning. Bayesian methods act as regularisation based solutions for continual learning. The objective function used for training can be seen in equation (2.35). The main aim is to optimise the objective function in equation (2.35) (taken from [121]). The posterior update is performed using the prior and the posterior w.r.t the former task (2.36). On a side note on notations, previous chapters dealt with examples of stationary datasets. When considering the equations in this chapter, data samples x or labels y are indexed with t, since the difference now is that these quantities are task dependent. Therefore, θ_t represents the parameters estimated based on task t and θ_{t-1} represents those estimated on the previous task. The equations for the loss function for Bayesian methods for continual learning and the posterior update is shown in equations

$$\mathscr{L}^t(\theta_t) = \underbrace{\sum_{n=1}^{N_t} \log p(y_t^{(n)}|\theta_t, x_t^{(n)})}_{\text{maximum likelihood}} - \underbrace{\frac{1}{2}\lambda_t(\theta_t - \theta_{t-1})^T S_{t-1}^{-1}(\theta_t - \theta_{t-1})}_{\text{regularisation term}}, \tag{2.35}$$

$$\begin{gathered}\underbrace{p(\theta|\mathscr{D}_{1:T})}_{\text{new posterior}} = \underbrace{p(\theta)}_{\text{prior}} \prod_{t=1}^{T} p(\mathscr{D}_t|\theta)\, \underbrace{p(\theta|\mathscr{D}_{1:T-1})}_{\text{old posterior}} \\ \text{where } \prod_{t=1}^{T} p(\mathscr{D}_t|\theta) \propto p(\mathscr{D}_T|\theta).\end{gathered} \tag{2.36}$$

The maximum likelihood, as discussed in Chapter 1, Section 2.1.8, involves optimizing the parameters θ of the Bayesian model so that the output distribution matches closely to the training set. In other words, maximising the probability of observing the data. The additional regularisation term is added as part of CL training to prevent catastrophic forgetting [194]. Briefly, this term adds a loss term that directs the change of parameter configuration based on the current task t towards those estimated in the previous task $t-1$. This is also referred to in the literature as knowledge consolidation [194] since it encourages the learning system to retain knowledge from previous tasks.

The hyperparameter λ_t is used to control the level of contribution from the previous data and the diagonal matrix S_{t-1} controls the strength of the regularisation, e.g. a diagonal of twos will result in twice the regularisation strength as opposed to unit diagonals of the matrix S_{t-1}. Ideally, it would be good to have high values of λ to enforce consolidation, however, one must be reminded about the stability-plasticity dilemma (see Section 2.5.1).

Variance Guided Continual Learning and Uncertainty Guided Continual Bayesian Networks

The VI and the Laplace approximation has inspired two of the most popular works in Bayesian methods for continual learning; VCL [121] and UCB [32]. In comparison, VCL differs from UCB in it that it uses both VI and Monte Carlo sampling-based techniques for both discriminative and generative setting. UCB, on the other hand, focuses on discriminative and weight pruning tasks. Though Bayesian in setting, VCL also exploits replays which they term core-sets [121] in their paper. UCB, on the other hand, does not use any form of representative data from previous tasks. The objective function of both the approaches is given in equations (2.37) and (2.38) taken from [121] and [32]

$$\mathscr{L}^t_{VCL}(q_t(\theta)) = -\sum_{n=1}^{N_t} \mathbb{E}_{\theta\sim q_t(\theta)}\Big[\log p\big(y_t^{(n)}|\theta, x_t^{(n)}\big)\Big] + \mathrm{KL}\big(q_t(\theta)\|q_{t-1}(\theta)\big), \tag{2.37}$$

$$\mathscr{L}_{UCB}(\theta,\mathscr{D}) = -\mathbb{E}_{q(\mathbf{w}|\theta)}\Big[\log(p(\mathscr{D}|\mathbf{w}))\Big] + \mathrm{KL}\big(q(\mathbf{w}|\theta)\|p(\mathbf{w})\big). \tag{2.38}$$

Here, the terms θ and $\mathbf{w}$ correspond to the parameters of the variational posterior and the weights of the BNN respectively. Though they both exploit VI as their approximation scheme. However, the UCB approach uses the BBB algorithm for training their BNN based architecture. VCL, on the other hand, updates the posterior of the new task by multiplying the prior with their posterior on the previous task. This is similar to the method mentioned in equation (2.36).

Furthermore, VCL also compares their update scheme using Laplace propagation [166] as opposed to VI. Another important difference in the approaches of VCL and UCB is the use of sufficient statistics. Particularly, UCB uses the variance information from the probability distribution on weights to update the learning rates for each of the layers of their BNN architecture. In particular, given that the weights of the variational posterior $q(\mathbf{w}|\theta)$ are represented by a Gaussian pdf. Then, using the reparameterisation trick (see Chapter 2, Section 2.1.9), the weights can be obtained using the function $g(\varepsilon)$. This is shown as

$$\begin{gathered}\theta = (\mu, \sigma^2) \;\text{ where }\; \sigma = \log(1 + \exp(\rho)),\\ g(\varepsilon) = \mathbf{w} = \mu + \sigma \circ \varepsilon.\end{gathered} \tag{2.39}$$

Here, ε is a random variable sampled from a Gaussian distribution of unit variance and zero-mean. Additionally, θ defines the parameters of the Gaussian pdf. This represents the variational posterior. The sufficient statistics are denoted by μ and σ^2 and the term ε is a noise drawn from a unit Gaussian. The Gaussian distribution has diagonal covariance where ρ represents the diagonal elements. The noise is multiplied point-wise (i.e. $\circ$) with the standard deviation σ. The learning update rule is defined using the standard deviation which represents the uncertainty. In particular, the UCB approach adopts two separate approaches to learning rate updates. In one, the learning rate is updated by dividing with $\frac{1}{\sigma}$ and in the other by $\frac{|\mu|}{\sigma}$. The second quantity is also known as the Signal-to-Noise ratio (SNR). In signal processing, SNR can be used to outline the desired part of a signal as opposed to the noisy part [47]. The higher the SNR value of a signal, the less noisy and more informative the signal. The opposite for a noisy signal when considering the vice versa.

When considering the disadvantages for each of the approaches, it follows that VCL has the disadvantage that it is not flexible. VCL works only for limited types of variational distributions [121]. If the user does not exploit the suggested type, then the repeated iterations for each task can accumulate errors. According to [121], this alongside the approximation of the KLD step can cause catastrophic forgetting. For this reason, the VCL approach uses episodic memory. The concept is similar to auxiliary datasets (see Section 2.5.1). For each task, a core-set is computed. This set can be chosen at random, however, VCL uses a clustering method that centers the set according to its importance. Furthermore, there a couple of caveats with the VCL approach; a) specification of posterior (Gaussian posterior) is needed, b) it is required that posterior remains close to prior and explanation of why this is important in variational approximation growth, c) the KLD is available for closed-form but the likelihood is not, hence why MC with reparameterisation is utilised which can suffer from biased gradients [39]. On the other hand, though UCB outperforms VCL, it utilises a multi-classifier approach to continual learning. This means that for each task a separate classifier is used. According to [28] and [116], techniques that employ a multi-classifier approach do not fully solve the problem of continual learning. A much better investigation would be so that what will be the performance difference on a single classifier. This is studied in the methods proposed in this Chapter, they also form the subject of the next section.

This section reviews the Bayesian methods in continual learning. It is seen how VI approximation in Bayesian methods can act as a regularisation based solution to continual learning. In particular, two different approaches adopted from Bayesian methods are reviewed: UCB and VCL. Additionally, the advantages and disadvantages of each of the approaches are considered. It is seen that VCL uses additional memory to store auxiliary datasets while UCB uses different classifiers for different tasks. The next section discusses the summary of this chapter and the proposed methodologies in each of the three areas reviewed in this chapter are provided in Chapter 3.

2.6 Summary

The chapter provided background knowledge on deep learning. This included the difference between machine learning and deep learning. Then, the convolutional architecture LeNet was reviewed. This was followed by the limitations of standard neural networks in computer vision applications. Then, some of the common layer configurations in convolutional networks were also covered. The focus of the chapter then shifted to adversarial attacks. Here, it was mentioned that despite the success of deep neural networks in image recognition, CNNs and DNNs are known to be easily tricked to output erroneous predictions despite minute changes in the input that are imperceptible to the human eye. Furthermore, both the non-linear and the linear views of the vulnerability of CNN and DNNs were reviewed, followed by the review of the fast-gradient sign method. The chapter then focused on providing background information on Bayesian methods. These methods allow the impact of uncertainties which can be used to deal with both erroneous predictions and vulnerability to adversarial attacks. In particular, approximation methods for the Bayesian approach such as variational inference and Monte Carlo methods were reviewed. Finally, the non-parametric Bayesian methods, in particular, the Gaussian process was discussed. The chapter then shifted to the background literature for the methods proposed in this thesis for the area of semantic segmentation, robust learning and continual learning. Firstly, it was seen that segmentation is an extension to classification. In the author's opinion, it can be seen as a more category region-based localisation extension of classification. Then, some of the most popular architectures in deep learning were considered. It was seen that the deep learning approach learns features automatically. However, like most deep neural networks, it is vulnerable to erroneous predictions and adversarial attacks. This motivated the section on uncertainty-aware segmentation methods. Here, the deep learning based approaches to uncertainty modelling were reviewed. Then, the focus shifted to the use of generative adversarial learning and GANs. In particular, the architectures DCGAN and LSQGANs were reviewed, as well as reviewing how LSQGANs can be used to improve learning in generative networks and deal with the problem of mode collapse. Secondly, the chapter then focused on literature involving the robust learning area tackled in this thesis. This began with the introduction to the field of adversarial robustness. Firstly, it was seen that the definition of robustness varies for different fields. In particular, both the informal and formal view of robustness was given as well as using examples of a linear and a non-linear classifier to explain the formal definition of robustness. Then, the challenges of developing robust artificial intelligence were considered. In particular, the effects of the unmodeled phenomenon, the human-user error, misspecified goals and incorrect models. The challenges from adversarial attacks were then considered in a separate section. This was followed by the review of literature in uncertainty and meta-learning based adversarial defence methods. It was reviewed that these methods establish a link between interpretability and robustness. Lastly, the focus of the chapter shifts to the final area tackled in the thesis, i.e. continual learning. Firstly, it was seen how DNNs are unable to learn dynamic datasets that are introduced sequentially in separate tasks. Furthermore, the definition of continual learning was introduced, including challenges in continual learning in the form of catastrophic forgetting and the stability-plasticity phenomenon that makes learning on continuous data. Then, it was also discussed that Bayesian methods can offer solutions to continual learning problems, specifically from variational inference in VCL and adaptive learning rate in UCB.

Chapter 3

Methodology

Chapter Overview

- Provide the background knowledge in pixel-wise labelling in semantic segmentation, all as operating as part of scene understanding
- Introduce to the Bayesian SegNet architecture and investigate the use of uncertainty quantification in the Bayesian SegNet
- Introduce to the notion behind generator adversarial neural networks and discuss how adversarial learning can be used to leverage uncertainty quantification
- Discuss the uncertainty based solutions as part of the methodologies proposed in this section
- Discussion on the role of robustness in artificial intelligence.
- Discussing the field of adversarial attacks, including discussion on both black box and white box.
- Relating learning from noisy labels to meta-learning.
- Role of Gaussian processes in adversarial robustness. Introducing the field of continual, including discussion on catastrophic forgetting and sequential task incremental learning.
- Literature survey of different methods in variance guided continual learning and other regularisation-based approaches.

3.1 Methodology for Semantic Segmentation

This section discusses the methods proposed for improving the performance of deep learning-based segmentation using adversarial learning, with a specific focus on the Bayesian SegNet [78]. The Bayesian SegNet is a popular architecture in segmentation that builds on the original SegNet [7]. The proposed architecture leverages model uncertainty obtained from the SegNet architecture at train time and uses adversarial based learning to obtain high-quality segmented maps. The novel contributions of this chapter in segmentation are discussed next.

3.1.1 Novel Contributions in the Area of Semantic Segmentation

The novelties of the proposed methods in the area of semantic segmentation are that for the first time, an add-on architecture that improves the performance of the popular architecture SegNet in the presence of a physical attack which reduces the receptive field size of the network. This architecture takes the form of two discriminator networks that are trained separately. These are termed uncertainty critic (UC) and quality critic (QC). One discriminates between segmentation maps coming either from the SegNet or the ground truth. The other discriminates between the model uncertainty obtained from the SegNet and an ideal solution that represents no uncertainty. The hypothesis regarding the proposed architecture approach states that uncertainty is a measure of confidence, learning from uncertainty can help improve the performance of neural networks. Furthermore, dealing with uncertainty is beneficial for decision making in neural networks for applications that pose as highly uncertain environments. Then, the second novel contribution is that the results, in Chapter 4, show that higher accuracies compared to the Bayesian SegNet are obtained. Training is performed on a small-sized dataset called CamVid [16] and a large-sized dataset Sun RGB-D [169]. The results show that dealing with uncertainties is beneficial for decision making in neural networks, especially in applications with highly uncertain environments. The next section goes further in-depth in studying the proposed architecture from the point of view of layer configuration and layout of the layers.

3.1.2 The Proposed Architecture: AdvSegNet

Network Layers

The proposed architecture (the AdvSegNet) inspires greatly from the popular architecture SegNet architecture. As shown in Figure 3.1, this is a simplified version of the original the Bayesian SegNet [78]. The differences can be listed as follows. The input receptive field is reduced to 128 x 128 x 3 (height, width, depth). This is done to simulate an attack where the receptive field of the system is damaged from noise, adversarial attacks or even physical ones.

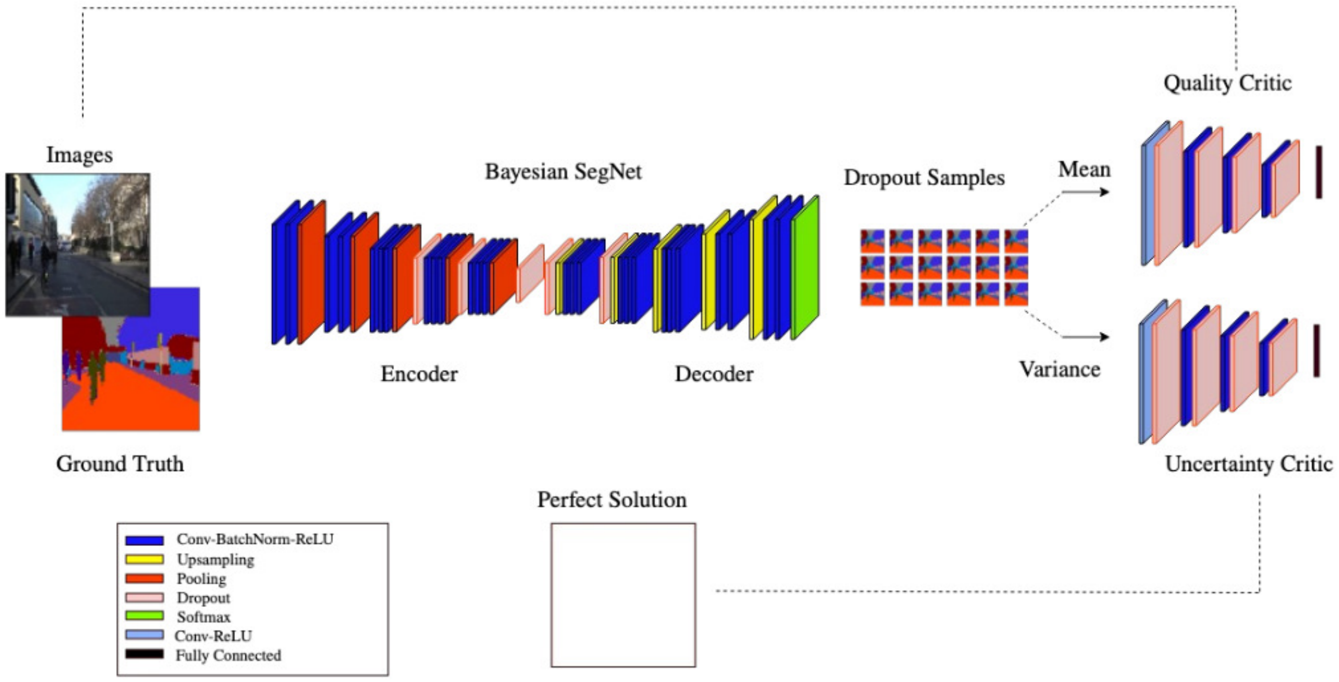
The encoder consists of a series of convolutional layers and 2 x 2 sized pooling layers inserted after every block of convolutional layers. The initial kernel size of the convolutional layers is set to 3 with 64 filters. The number of filters is doubled after every block.

Dropout layers are inserted after the 3^{rd}, 4^{th} and the 5^{th} pooling layer. The probability of dropout is set to 0.5 for each dropout layer. Convolutional layers decompose the features of an image and hierarchically learn them, starting from simple features in the earlier layers and more complex ones in the later layers. Pooling layers downsample these features.

Batch normalization layers are added to the SegNet after every convolutional layer. These layers scale and adjust the activations of network layers and help in the stabilizing of the training process. These are followed by rectified linear units (ReLU) [85] that introduce the non-linearities in the network.

The decoder architecture follows a similar style. However, it uses upsampling layers that increase the window size back to 128 x 128. Dropout layers are added before the start of the decoder and before the 1^{st} and the 2^{nd} upsampling layer. This is then passed through a softmax layer. The softmax layer reduces the dimensions of the logits of the AdvSegNet to probabilities of class predictions. Then, the architecture is run several times to output dropout samples.

Fig. 3.1 The AdvSegNet architecture. The dropout layers are placed after the third, fourth and fifth convolutional block in the encoder half and before the first and second upsampling layers in the decoder half. For the two critics UC and QC, dropout layers are placed after each convolutional block.



Uncertainty and Quality Critic

Once the dropout samples are obtained from the decoder, the sample mean and the sample variance is calculated as shown in the equations below. The output of the decoder is denoted to be $g_{n,t}$ for the t^{th} run on the n^{th} input image x_n. Here, $n = 1,\ldots,N$ and N is the number of training images. Then the mean of the dropout samples is defined as μ_n. The model uncertainty (variance) is defined as σ_n. These are shown in equation (3.1) and (3.2) respectively. The mean is fed to QC and the model uncertainty to UC. The mean and the variance are shown obtained as

$$\mu_n \approx \frac{1}{T}\sum_{t=1}^{T} g_{n,t}, \tag{3.1}$$

$$\sigma_n \approx \frac{1}{T}\sum_{t=1}^{T} g_{n,t}^T g_{n,t} - \mu_n^T \mu_n. \tag{3.2}$$

The QC is a discriminator that outputs the logits denoted by the term d_n^{QC}. QC learns to map the ground truth sample y_n to ‘1’ (real) and dropout samples μ_n to ‘0’ (fake), this is so that

$D_{QC} : \{y_n, \mu_n\} \rightarrow \{0,1\}$. UC, on the other hand, is defined as a discriminator that learns to map the perfect solution (no uncertainty), σ_p, to '1' (real) and the uncertainty coming from the SegNet, σ_n, to '0' (fake). We represent its logits as d_n^{UC}. This can be shown as $D_{UC} : \{\sigma_p, \sigma_n\} \rightarrow \{0,1\}$. The perfect solution is considered to be a blank white image of dimensions 128 x 128. Given the architecture, the loss functions are discussed in the next section.

3.1.3 Loss Functions for the Proposed Architecture

This section aims to discuss the loss functions used in the proposed architecture AdvSegNet. Conceptually, the losses in the training of the AdvSegNet follow a similar approach to [102]. There is a total of three loss functions used. Each with a different purpose. These can be listed as a) cross-entropy, b) adversarial loss on QC, c) adversarial loss on UC.

The cross-entropy loss takes the logits from the SegNet architecture. The adversarial losses take the logits from both the outputs of UC and QC. The cross-entropy loss is denoted as $\mathscr{L}_{SEG}$, while the critic losses as $\mathscr{L}_{UC}$ and $\mathscr{L}_{QC}$. This is shown as

$$\mathscr{L}_{SEG} = -x_n \log(\mu_n) + \tfrac{1}{2}\mathbb{E}[d_n^{QC} - 1]^2 + \tfrac{1}{2}\mathbb{E}[d_n^{UC} - 1]^2. \tag{3.3}$$

The cross-entropy term encourages AdvSegNet to produce labels that match the ground truth observations. The first term in the square brackets in (3.3) defines the adversarial loss for QC and the term in the second square brackets for UC. These losses both encourage the AdvSegNet to a) improve the quality of segmented label outputs, b) learn to deal with uncertainty. Furthermore, the individual loss functions of the critics that ensure that both QC and UC discriminate effectively are calculated in (3.4) and (3.5) as shown

$$\mathscr{L}_{QC} = \tfrac{1}{2}\mathbb{E}[d_n^{QC} - 1]^2 + \tfrac{1}{2}\mathbb{E}[d_n^{QC}]^2, \tag{3.4}$$

$$\mathscr{L}_{UC} = \tfrac{1}{2}\mathbb{E}[d_n^{UC} - 1]^2 + \tfrac{1}{2}\mathbb{E}[d_n^{UC}]^2. \tag{3.5}$$

The adversarial losses associated to the AdvSegNet in (3.3) and the adversarial loss functions in (3.4) and (3.5) are trained separately. They differ from the standard GAN loss in (2.28) and are specifically used in Wasserstein GANs and Least Squares (LSQ) GANs [107]. LSQ GAN based losses are chosen in our experiment over the traditional loss because they provide stability in terms of training. The traditional GAN loss suffers from vanishing gradient problem [6] and is proven to output images of low-quality [137]. The next section considers the optimisation and the training half of the proposed architecture.

3.1.4 Optimization & Training of the Proposed Architecture

This section aims to discuss the optimisation and training half of training the proposed AdvSegNet architecture. Firstly, considering the type of optimisers used in the experiments, Adam optimizers [81] are utilised to train both the SegNet and the two discriminators: UC and QC. Ideally, the training of the discriminators is kept at a lower learning rate, compared with when training the layers in the segmentation network body (see Algorithm 4).

The choice of having a low learning rate for discriminator is taken for stabilising the training process [112]. The training regime adopted for training the discriminators is inspired from [6]. This entails that, for each episode, the number of optimization steps taken by both discriminators is denoted by t_{critic}. After that, the SegNet takes an optimization step. The default value of t_{critic} is set to 5. However, for episodes less than 25 and on the 500^{th} episode, t_{critic} is set to 25.

Furthermore, after the discriminators take an optimization step, their weights (i.e. θ_{QC} and θ_{UC}) are clipped to values in between -0.01 and 0.01. This is done to prevent the vanishing gradient problem in the training process [6]. The weights of the SegNet, represented as θ_g, are not clipped in this experiment as part of the training requirements. The entire training algorithm is shown in Algorithm 4. The next section discusses the datasets used for the training of the proposed architecture.

3.1.5 CamVid: Outdoor Scene Labels

This section aims to introduce the specifications of the two datasets used in the training of the proposed AdvSegNet architecture. These are the datasets: CamVid [16] and Sun RGB-D [169]. Firstly, this section deals with the specification of CamVid, followed by the second section on Sun RGB-D.

CamVid is an outdoor road scene understanding dataset consisting of a collection of videos in both day and evening settings taken from a camera-rigged automobile. It has a total of 367 training images and 233 validation images. The segmented classes amount to 12 and consist of common outdoor objects such as roads, cars, buildings, signs and poles. CamVid has been widely adopted across various segmentation based techniques as benchmark for studying motion-based segmentation [16], [7], [149], [122]. Though this dataset consists of 12 object categories, each object varying in size and shape, the dataset only contains 367 training images and is considered simple to validate on different architectures. The next subsection considers a much harder segmentation dataset, the Sun RGB-D.

3.1.6 Sun RGB-D: Indoor Scene Labels

Road scenes like in CamVid tend to have a limited variety of images and classes. Sun RGB-D, on the other hand, is a very challenging dataset. It is a large dataset, comprising 5825 training images and 5050 testing images of indoor scenes taken from different sensors with varying resolutions. The total number of categories is 37. They include categories such as floor, chair, wall, ceiling, table and sofa. Examples in Sun RBG-D entail various challenges including a variety of shapes, sizes and various poses. Indoor scenes also tend to be cluttered and as a result, can be severely occluded by objects that are larger in size and shape. This dataset also comes with depth information for 3D applications. However, since this work focuses more on segmentation in 2D images, depth information is not utilised during training.

3.1.7 Pascal Visual Object Classes

Pascal Visual Object Classes (VOC) [35] is a dataset containing various objects that can be captured in day to day life. The dataset consists of a total of 20 categories/classes that vary from vehicles such as aeroplanes, bicycles, birds, boats, buses and cars to animals such as cats, cows to various objects such as chairs, beds and sofas. The Pascal VOC dataset comprises a total of 1464 images for training and 1449 images for validation. Each of the images in this dataset is accompanied by annotations that are separate for various object detection tasks. This includes object classes for classification, segmented label maps for segmentation and bounding boxes for detection. The diversity of the images in the Pascal VOC dataset makes it a challenging and attractive dataset for deep learning/AI researchers to benchmark their deep learning algorithms using various metrics for evaluation. Some of these metrics are discussed in detail in the next section.

3.1.8 Evaluation Metrics

This subsection considers the evaluation metrics used in the methods proposed in this chapter. To measure the performance of the proposed architecture, two accuracy measures and the mean intersection over union metrics ($mIoU$). The two accuracy measures are the global average accuracy and the class average accuracy [113], [183], [170].

The accuracy is measured by computing the frequency of predictions that match the labels. In general speaking, it highlights the correctness of the segmentation. There are two ways to compute the accuracy of segmentation: global average and class average.

The global average accuracy, also known as the per-pixel accuracy [113], [183], [170], calculates the ratio between the amount of correctly classified pixels and the total number of them. The class average accuracy, on the other hand, computes the ratio of the correctly classified pixels based on per-class value. This is then averaged over all the classes. Class average accuracy is also known as the mean pixel accuracy [113], [183]. The respective equations for both are shown in equation (3.6) and (3.7), taken from [170]. Here, the term N_{ij} denotes the number of pixels that category i that is predicted as category j, vice versa for N_{ji} and n_c represents the total number of different categories. The global, class average accuracies and the $mIOU$ can be shown as

$$Global\ Average\ Accuracy\ = \frac{\sum_i N_{ii}}{\sum_i \sum_j N_{ij}}, \tag{3.6}$$

$$Class\ Average\ Accuracy\ = \frac{1}{n_c} \text{x} \sum_i \frac{N_{ii}}{\sum_j N_{ij}}, \tag{3.7}$$

$$mIOU = \frac{1}{n_c} \text{x} \sum_i \frac{N_{ii}}{\sum_j N_{ij} + \sum_j N_{ji} - N_{ii}}. \tag{3.8}$$

The accuracy measurement alone does not provide any useful information regarding how much of the segmented ground truth map has been classified correctly in terms of shape. In other words, it is useful to know if the shape of the output segmented map matches the shape of the ground truth segment map. Hence, an additional evaluation metric in the form of intersection over union is considered [113], [183]. This metric computes the area between the predicted segmentation map and the ground truth. This is then divided by the area of the union between the predicted and the ground truth map. The mIOU is the average intersection over union over all of the classes and is shown in equation (3.8). The next section discusses the proposed methodology for robust learning in the CNN-GP portion of this chapter.

3.2 Methodology for Robust Learning

This section presents an architecture for image classification which comprises a convolutional neural network (CNN) feature map extractor combined with a Gaussian process (GP) classifier. It is further used to quantify the impact of gradual and abrupt uncertainties in the presence of adversarial attacks, Gaussian noise and blurred images. Uncertainty quantification is achieved by combining a CNN with a GP classifier algorithm. The variance of the post-softmax samples obtained from GP output characterises the uncertainty. The novel contributions of this section are discussed next.

3.2.1 Novel Contributions in the Area of Robust Learning

The novelty of this contribution is two-fold. First, a novel regularisation method that comprises of three separate loss functions: Kullback-Leibler, Wasserstein distance and maximum correntropy is used to improve the performance of the CNN-GP classifier to noisy attacks and blurring. This also assists in improving the sensitivity of the uncertainty information to both noisy and adversarial attacks.

Secondly, a novel adversarial training method is used to train the CNN-GP network firstly with clean images and then with adversarial samples generated by replacing original images in the mini-batch with their nearest neighbours. The training is carried out in the following steps. First, the discrepancy between the GP classifier outputs and the ground truth labels is minimized. This is accomplished by updating the weights of the CNN-GP based on the maximum likelihood loss (see equation (2.13)) between the GP classifier outputs and ground truth labels. Then, images are replaced with their nearest neighbours in the mini-batch. This is done by computing the Euclidean distance (or l_2 norm) between the features obtained by forward propagating the mini-batch and a randomly ranked mini-batch. Based on the distances, a sample is picked randomly from the nearest 10 neighbours and is used to replace the input image. This is repeated until the entire mini-batch is replaced with their neighbours. This technique is inspired from [94]. The difference is that their method focuses on noisy labels, this method focuses on the neighbour replacement of images. The choice of the number of samples to replace with neighbours is dependent on the mini-batch. For example, in the case of the experiments, the mini-batch size is 16. The chosen value should not be too high so that the system is unable to detect the clean samples from perturbed and not too low so that it is unable to detect perturbations. Hence, the value 10 is reasonably in between the two extreme cases.

The samples from the new mini-batch are forward propagated to obtain the new predictions from the CNN-GP. These predictions are compared with ground truth labels with one of the three similarity losses: Kullback-Leibler divergence, Wasserstein distance and maximum correntropy. These losses regularise the maximum likelihood loss. The proposed model is tested separately on the datasets: MNIST [90], Fashion-MNIST [190], CIFAR-10 and CIFAR-100 [84].

Additionally, tests on robustness to noisy, blurred images and white-box attack FGSM are also carried out for the MNIST dataset case. The results show that the test accuracy improves the performance of the proposed model, compared to using methods that do not quantify the impact of uncertainties. A comparison with a state-of-art Monte Carlo dropout method is made. Furthermore, it is shown that the CNN-GP model outperforms the version of the network without regularisation, in terms of reliability and computational efficiency. In particular, accuracy measures, precision-recall and receiver operating characteristic curve (ROC) are used for performance evaluation. The system is tested under three settings: Gaussian noise, motion blurring and adversarial attacks (specifically FGSM [49]).

3.2.2 Architecture Detail: Convolutional Network Gaussian Process

The architecture used in the methods proposed in this section consists of a CNN feature extractor, the outputs of which are fed to a GP classifier (see Figure 3.2). Combining CNNs with GP is not a new idea. It has been previously been explored for example in the work of convolutional Gaussian processes in [185].

The main difference between their approach and the one implemented in this thesis is that in [185], the convolution operation is performed as part of the kernel function and in this thesis, this is externally applied via the deterministic convolutional layers (the CNN component). As a result, the advantage of the approach by [185] is that the inputs are directly fed to the kernel and the feature decomposition (as part of convolution operation) is performed directly on the inputs as part of computing the covariance matrices of the GP. Another advantage of this is that it does not require training of the CNN component separately. This is unavoidable in the CNN-GP architecture used in the methodology of this chapter. Furthermore, this alleviates the need to tune a large number of parameters that as is the case whenever training deep neural networks [89].

To consider the CNN-GP architecture used in the methods of this chapter in more detail. Firstly, the CNN has two convolution layers of 32 and 64 filters of 3x3 kernel size.

The padding size of convolutional layers vary. This is because MNIST [90] and Fashion-MNIST [190] datasets share the same input size of 28x28x1 as opposed to CIFAR-10 and CIFAR-100 [84] i.e. 32x32x3. For MNIST and Fashion-MNIST padding size is set to 2 and 1 for CIFAR-10 and CIFAR-100. Secondly, a max-pooling layer is introduced between the second layer and two dropout layers each with probability 0.5 and 0.25, respectively. Pooling layers downsample the features and dropout layers are used as a regularizer. Lastly, the fully connected layer, on the other hand, converts the flattened outputs to a 16x128 feature vector, as the batch size is set to 16 and the number of feature units is set to 128 as defaults.

The CNN architecture can be seen in Figure 3.2. The output from the GP is a categorical distribution, from which an N x 1 vector (where N is the batch size) is then approximated with softmax function to obtain a vector of size 16x10 containing a prediction for each of the 10 classes in MNIST and FMNIST dataset, 10 and 100 classes for CIFAR-10 and CIFAR-100 respectively.

The architecture is presented in two levels: the high-level and the low-level. The high-level description is presented in Figure 3.2, and the low-level description in Figure 3.3. The high-level description displays only the layer inputs and outputs with a brief overview of the system. The low-level description, on the other hand, goes further in-depth. It explains each input and output in detail, highlighting the connections and providing a description of the post-softmax sampling performed towards the end of the GP classifier component.

High-Level View Description

- Highlights connection between layers
- Provides brief description of each layer
- Defining the total number of layers
- Does not provide description of the sampling procedure

Fig. 3.3 The CNN-GP model at test time. It consists of a CNN base feature extractor with a GP after it. This diagram represents the deterministic version i.e. scalar weights. In the Bayesian setting, the weights of the convolutional layers are replaced with prior distributions. The input has size 16x1x28x28 for its batch size, depth, height and width respectively. The output dimensions are shown for each output.

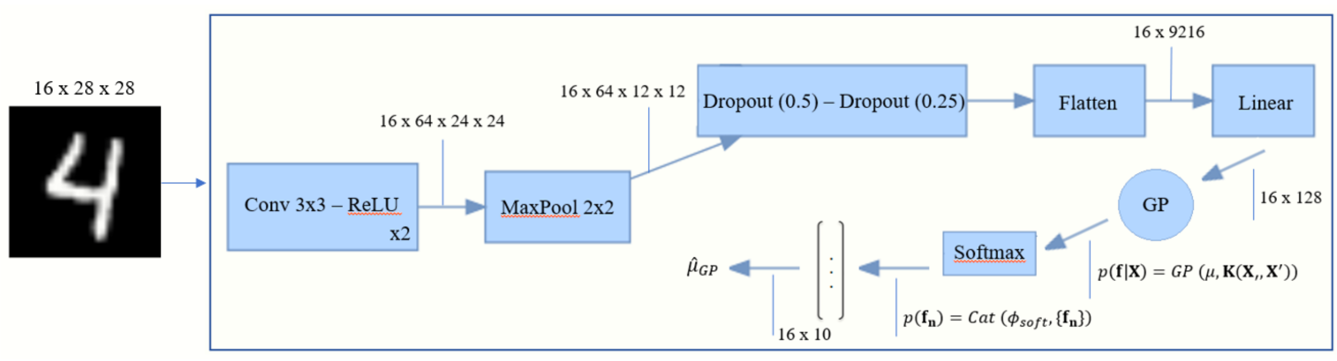
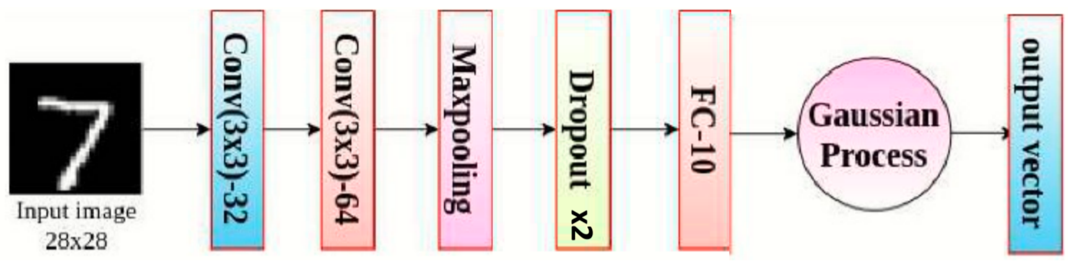

16 x 28 x 28
Conv 3x3 – ReLU x2
16 x 64 x 24 x 24
MaxPool 2x2
16 x 64 x 12 x 12
Dropout (0.5) – Dropout (0.25)
Flatten
16 x 9216
Linear
16 x 128
GP
$p(\mathbf{f}|\mathbf{X}) = GP\ (\mu, \mathbf{K}(\mathbf{X}, \mathbf{X}'))$
Softmax
$p(\mathbf{f_n}) = Cat\ (\phi_{soft}, \{\mathbf{f_n}\})$
16 x 10
$\hat{\mu}_{GP}$

Fig. 3.2 A high-level view of the proposed model, the CNN-GP



Low-Level View Description

- Highlights the outputs and in the inputs in detail
- Includes feature sizes of inputs and outputs
- Displays kernel sizes for convolutional layers
- Specifies the sampling procedure towards the end of the classifier layers

3.2.3 The Training Algorithm for CNN-GP

The training set is divided into mini-batches. Each mini-batches consist of 16 samples, as batch size is set to 16. In the first step, the input images from the mini-batch are forward propagated to obtain the CNN-GP labels. The maximum likelihood of the GP function is used to update the weights of both the CNN and the GP components. Then, backpropagation of the regularised maximum likelihood is performed. Before this step, the input images in the mini-batch are replaced with their nearest neighbours. This is done by taking a sample after randomly ranking the mini-batch and computing the Euclidean distance between its corresponding feature and the features from the entire mini-batch. Given the distance measures, the nearest 10 neighbours are selected. The size of neighbours cannot exceed the batch size and cannot be less than 5 since the effect will be too small, hence this is a sensible choice for the number. It can be set to 16 which is the size of the mini-batch but ideally, it is good to leave some clean samples so the algorithm can differentiate between the perturbed and the clean samples while still learning to classify the clean samples correctly. Then, from these neighbours, a single sample is randomly chosen to replace the original sample in the mini-batch. This is repeated until the entire mini-batch is changed.

The new mini-batch is forward propagated to obtain the new CNN-GP labels. These labels, predicted by the GP classifier, are compared with the actual labels using a maximum likelihood error. The maximum likelihood loss is regularised with one of the similarity losses (i.e. KLD, Wasserstein and maximum correntropy). These are used to encourage CNN-GP to remain consistent in its predictions. The error obtained is then backpropagated and the parameters of the GP (the lengthscale and amplitude and the variational parameters) and weights of the CNN (convolutional and fully connected layers) are updated.

An approximated variational inference is used since categorical likelihood is needed for classification. A softmax function converts the functional vector from the GP into a vector of probabilities for each of the classes. A full description of the algorithm is provided in Algorithm 1 along with notation definitions in Table 3.1. The next section considers these losses in depth.

Table 3.1 Notations and Definitions

Notation	**Meaning**
M	Total number of episodes
γ	Learning rate of the CNN-GP architecture
K	Number of nearest neighbours to pick a new sample from
N	Batch size
β	Kullback-Leibler divergence scaling factor
m	Episode number
λ	Lengthscale parameters of the GP classifier
A	The amplitude for the squared exponential kernel
u_i	Variational free parameters for the i^{th} batch of data
$q(u_i)$	Variational likelihood based on the i^{th} batch of data
$p(u_i)$	Expected real likelihood based on the i^{th} batch of data
θ^m_{CNN}	Weights of the CNN feature extractor for the m^{th} episode
θ^m_{GP}	Parameters of the GP classifier for the m^{th} episode
x_i	Data sample from the i^{th} batch of data
y_i	Label sample from the i^{th} batch of data
X	4D-data tensor holding the data samples
Y	4D-data tensor holding the labels samples
D	Dataset ordered pair holding X and Y
Z	Number of units passed from CNN
δx_i	The difference between the i^{th} data point and the CNN-GP
f^{GP}	The Gaussian process function
f^{CNN}	The CNN function
$\hat{y}^{CNN}_i$	Softmax prediction from the CNN base
$\hat{y}^{CNN}_z$	Prediction from the z^{th} node from the CNN base
$\mathscr{L}^{MLE}$	Maximum likelihood loss
$\mathscr{L}^{SIM}$	Similarity loss

Algorithm 1 CNN-GP Training algorithm for robustness analysis

```
1:  Require $M$: episodes, $\gamma$: learning rate (GP), $k$: number of nearest neighbours for choice of new sample, $N$: batch size, $\beta$: KLD scaling factor
2:  Do initialization of weights: $\theta^m_{CNN}$, $\theta^m_{GP}$
3:  for $m = 0, \cdots, M$ do
4:      Sample mini batch ($x_i$,$y_i$), of length $N$ from dataset $D = X, Y$ where $X$ and $Y$ are 4-D tensors holding images and labels from the entire dataset, where $x_i \in \mathbb{R}^{hxwxc}$ (image height, width and channel) and $y_i \in \mathbb{R}^{1xC}$ ($C$ is total number of classes).
5:      BEGIN Updating the weights of the CNN-GP based on the maximum likelihood loss $\mathscr{L}^{MLE}$
6:      do $\rightarrow$ forward pass of CNN base $f^{CNN} : x_i \rightarrow z_i$, where $z_i \in \mathbb{R}^{ZxC}$ and $Z$ is the number of hidden units' feature outputs passed from final fully connected layer of CNN base feature extractor
7:      do $\rightarrow$ forward pass of GP $f^{GP}(z_i)$ to obtain the posterior likelihood $p\big(y_i|f^{GP}(z_i);\mu_i,\sigma_i^2\big) = \mathscr{N}\big(\mu_i, K_i\big)$ where $\mu_i$ represents the mean of the GP and $K_i$ is the kernel (i.e. squared exponential $K_i = A\exp\Big[-\frac{1}{2}\big(\frac{\delta x_i}{\lambda}\big)^2\Big]$ and $\mathscr{N}$ represents the Gaussian distribution
8:      do $\rightarrow$ Compute the maximum likelihood loss: $\mathscr{L}^{MLE} \approx \sum_{i=1}^N \mathbb{E}_q\Big[\log\big(p(y_i|f^{GP}(z_i);\mu_i,\sigma_i^2\big) - \beta D_{KL}\big(q(u_i)\|p(u_i)\big)\big)\Big]$
9:      do $\rightarrow$ Compute gradients of loss w.r.t weights of CNN base feature extractor and GP : $\frac{\partial \mathscr{L}^{MLE}}{\partial \theta_{GP}}, \frac{\partial \mathscr{L}^{MLE}}{\partial \theta_{CNN}}$
10:     do $\rightarrow$ Update the parameters of GP and CNN feature extractor for $m^{th}$ episode: $\theta^{m+1}_{CNN} \leftarrow \theta^m_{CNN} - \gamma\frac{\partial \mathscr{L}^{MLE}}{\partial \theta^m_{CNN}}.\mathscr{L}^{MLE}$, $\theta^{m+1}_{GP} \leftarrow \theta^m_{GP} - \gamma\frac{\partial \mathscr{L}^{MLE}}{\partial \theta^m_{GP}}.\mathscr{L}^{MLE}$
11:     BEGIN backpropagation of the regularized loss $\mathscr{L}^{MLE} + \mathscr{L}^{SIM}$
12:     do Replace input samples with top-10 neighbours $\hat{x}_i$
13:     do $\rightarrow$ forward pass of the CNN base feature extractor $f^{CNN} : \hat{x}_i \rightarrow \hat{z}_i$
14:     do $\rightarrow$ forward pass of the GP $f^{GP} : \hat{z}_i \rightarrow p\big(\hat{y}_i|f^{GP}(\hat{z}_i);\hat{\mu}_i,\hat{\sigma}_i^2\big) = \mathscr{N}\big(\hat{\mu}_i, K_{\hat{x}_i}\big)$
15:     do $\rightarrow$ Calculate $\mathscr{L}^{MLE} + \mathscr{L}^{SIM}$ between the labels $y_i$ and the GP classifier posterior mean $\hat{\mu}_i$ using the KLD
16:     do $\rightarrow$ Update the new parameters of $\hat{\theta}^m_{GP}, GP : \hat{\theta}^m_{GP} \leftarrow \theta^m_{GP} - \gamma\frac{\partial \mathscr{L}^{MLE} + \partial \mathscr{L}^{SIM}}{\partial \theta^m_{GP}}.\mathscr{L}^{MLE} + \mathscr{L}^{SIM}$
17:     do $\rightarrow$ Update parameters of CNN feature extractor : $\hat{\theta}^m_{CNN} \leftarrow \theta^m_{CNN} - \gamma\frac{\partial \mathscr{L}^{MLE} + \partial \mathscr{L}^{SIM}}{\partial \theta^m_{CNN}}.\mathscr{L}^{MLE} + \mathscr{L}^{SIM}$
18: end for
```

3.2.4 Proposed Loss Functions: Kullback-Leibler, Wasserstein Distance and Maximum Correntropy

Consider two sets of probability mass functions $p(x)$ and $q(x)$ that take a data point x. Finding the shift of mass from one set to the other requires calculating the discrepancy between the two. The Kullback-Leibler divergence [87] KL, shown in equation (3.9), represents this discrepancy as a measure of entropy difference, where entropy is defined as the sum of the log of the possible outcomes for a random variable. It quantifies the shift of probability mass by taking the difference of entropy across the distributions.

The Wasserstein distance, on the other hand, [125], solves the problem from the point of view of optimal transport. These problems are divided into two parts: assignment and cost. The assignment strategy determines how much mass is moved across the supports of the distributions. The cost measures the effort required for the assignment strategy. Two versions of the Wasserstein metric are used. One is an approximation in equation (3.10), where matrices P and C represent assignment and cost respectively. The total cost can be obtained by taking the Frobenius (i.e component-wise) inner product of the two (i.e. $\langle\, C, P\, \rangle$). The transport plan is to obtain the minimum of the product. This is subtracted from the entropy of $p(x)$ $\big($i.e. $\sum_x p(x)\, \log p(x)\big)$ in equation (3.10). Here, η is a scalar multiple whose default value is set to $\eta = 0.1$ in the experiments. Additionally, a quadratic distance-based cost function is used as an approximation to the full Wasserstein distance formulation. The full form (in equation (3.11)) takes the infimum of the absolute difference between the masses where γ denotes the transport plan. This work uses the differences between the pair of successive values across two masses $p(x)$ and $q(x)$ as the transport plan.

Finally, the maximum correntropy loss function [136] has also been implemented in the backpropagation of secondary losses step . The maximum correntropy loss function uses a kernel to compute the difference across two variables instead of using entropy-based methods such as in KLD and Wasserstein functions. The formulation can be seen in (3). The Gaussian kernel is a popular one: $k_\sigma(p(x)-q(x))^2 = \frac{1}{\sqrt{2\pi\sigma}} \exp\big(-\frac{(p(x)-q(x))^2}{(2\sigma^2)}\big)$ where σ^2 represents the variance of the distribution. The aforementioned cost functions are given as

$$\mathscr{L}^{KLD} = KL(p(x)\|q(x)) = -\sum_x q(x)\log q(x) + \sum_x p(x)\log p(x), \tag{3.9}$$

$$\mathscr{L}^{WASS} = min\langle C, P\rangle - \eta \sum_x p(x)\log p(x), \tag{3.10}$$

$$\mathscr{L}^{WASS-FIRST} = \inf \int |p(x) - p(y)|\, \gamma\,(p(x) - p(y)), \tag{3.11}$$

$$\mathscr{L}^{MC} = V_\sigma\big(p(x), q(x)\big) = \mathbb{E}\Big[k_\sigma\big(p(x) - q(x)\big)\Big] = \frac{1}{N}\sum_{x=1}^{N} k_\sigma(p(x) - q(x)). \tag{3.12}$$

The term V_σ refers to the MC and $\mathbb{E}$ refers to the expected value. This measure has been proven to be less sensitive to outliers [136]. This is found in many second-order statistics measures such as cross-entropy. It is heavily studied in outlier suppression [136]. The next section considers the performance evaluation metrics.

3.2.5 Evaluation Metrics: Precision-Recall and ROC Curves

This section aims to review the evaluation metrics used for the performance validation of the proposed model CNN-GP. It begins with motivating the problem of imbalance classification. Then, it gives details on how the confusion matrix can be used to extend the usual accuracy measure on classification performance evaluation. Finally, it overviews how precision-recall and ROC curves can be calculated.

The Importance of Class Imbalance in Classification Study

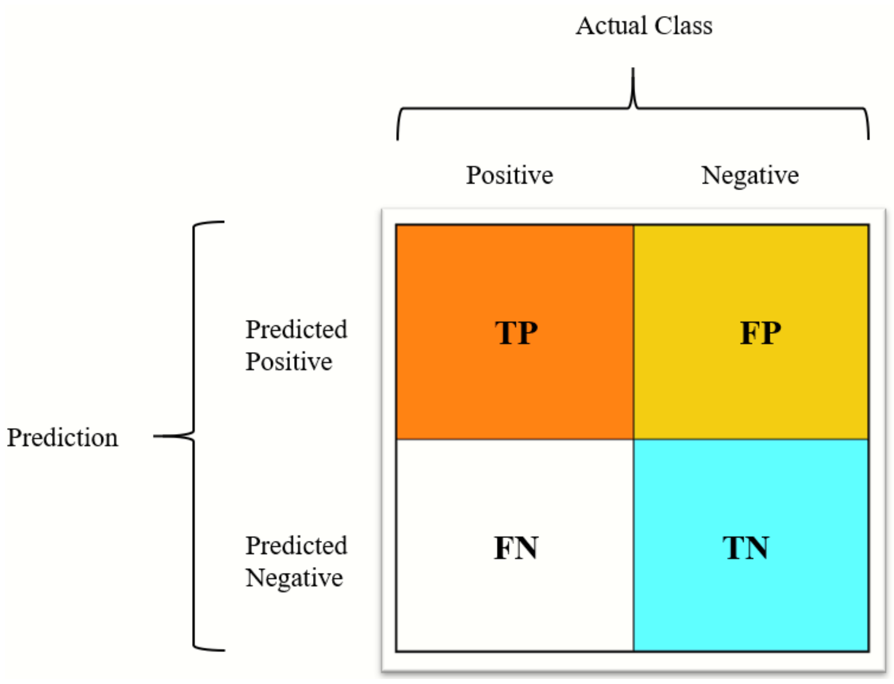
The majority of DNN algorithms are designed assuming that they will be trained on a training set with an equal number of samples for all categories. In reality, this is not the case. The majority of the datasets in AI research [16], [169] vary from a slight bias, leading to a skewed number of categories.

Class imbalance can be caused by the introduction of bias in the data capturing process, this is also known as biased sampling [86]. In other cases, external noises can also play an influential role. Take for instance the CamVid dataset (see Section 3.1.5) as an example for motivating the explanation of class imbalance and influence of environmental factors. Since this dataset is based on outdoor scenes, the majority of the samples have the category ‘sky’. The presence of the category of say for example ‘pole’ and ‘sign symbol might not be as frequent as the ‘sky’ category. The same can be said about the ‘pedestrian’ category. For example, in instances when the images are shot at a specific time of day, there can be examples where the ‘pedestrian’ category might not be visually clear e.g. in shadows. A similar case can be observed in the time of day when the roads are overcrowded. In this case, there would be more number of ‘pedestrians’ category samples than those of ‘vehicles’ (e.g. car or truck). Hence, in brief, it can be said that a dataset with a perfect balance of classes is extremely rare to capture, especially in image classification applications.

The Confusion Matrix

In terms of predictive classification performance, this can cause misinterpretation of categories. In particular, those categories that are in the minority. This type of bias then forms an inherent defect in the classification system. In these instances, it is often not sufficient to simply quote the ‘accuracy’ on the dataset. Since claiming a classifier is 80% accurate does not reflect this performance on all categories. In the previous example of methods in segmentation, i.e. the AdvSegNet (see Section 3.1.2), this was dealt with using per-class accuracy. In this section, the use of confusion matrix is adopted [86]. This is shown in Figure 3.4.

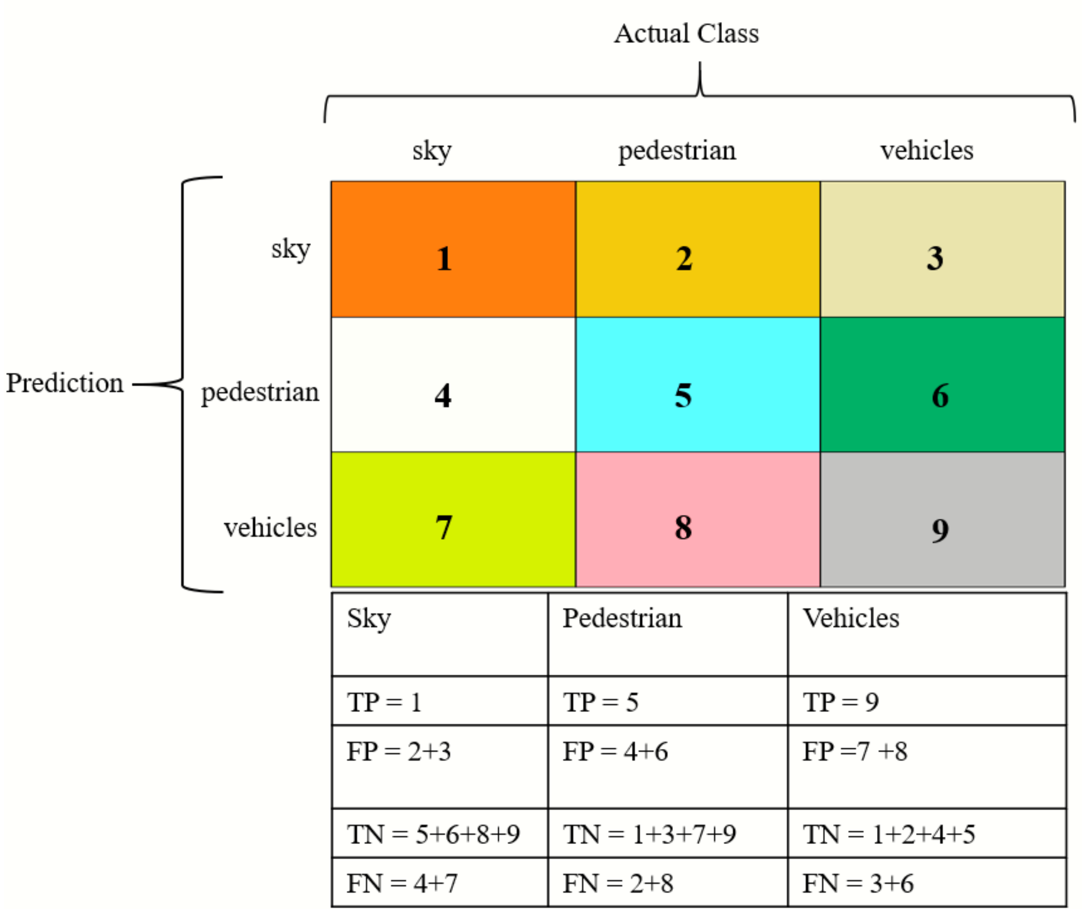
Fig. 3.4 An example of a confusion matrix. The categories are placed on the x-axis and the predictions on these classes are placed on the y-axis. The true positive and the false-positive samples are in the top half red and green quadrants respectively. The false-negative and the true negatives are presented in the lower half white and blue quadrants respectively. Image reprinted from [3]



.

The above example shows a confusion matrix applied to binary classification. This, of course, can also extend to multi-class examples. The process of computing a confusion matrix involves first organising predictions on the test set into two categories: the number of correct predictions and the number of incorrect predictions. These need to be obtained for all the categories. Based on this, the predictions are sorted w.r.t the true labels on two separate axes.

Fig. 3.5 An example of a confusion matrix for a multi-class setting. The categories are placed on the x-axis and the predictions on these classes are placed on the y-axis. The table below the figure tabulates the TP, FP, TN and FN for each of the categories. Image re-printed from [3]



Prediction \ Actual Class	sky	pedestrian	vehicles
sky	**1**	**2**	**3**
pedestrian	**4**	**5**	**6**
vehicles	**7**	**8**	**9**

Sky	Pedestrian	Vehicles
TP = 1	TP = 5	TP = 9
FP = 2+3	FP = 4+6	FP =7 +8
TN = 5+6+8+9	TN = 1+3+7+9	TN = 1+2+4+5
FN = 4+7	FN = 2+8	FN = 3+6

.

The true positive (TP) stands for instances where the classifier predicts the correct labels for the positive class (i.e. the class being inspected). The true negative (TN) instances are where the predictor correctly classified categories that are different from those that are positive. In false positive (FP) instances the classifier misclassifies negative classes as positives and vice versa for false negative (FN) instances. The extension to multi-class confusion matrix can be seen in the Figure 3.5.

Receiver Operating Characteristic Curves

To further evaluate the predictive performance of the classification system, the evaluation metric precision-recall and ROC curves can be used. The ROC curves are plots that can be used to summarise the results of classification for positive classes only. Specifically, the ROC curves are obtained by first computing the true positive rate (TPR) and false-positive rate (FPR) [3]. The TPR is defined as the total number of false positives divided by the sum of true positive and false negatives. On the other hand, the FPR is defined as the total number of false positives divided by the sum of false positives and true negatives. These can be seen in equations (3.13) and (3.14). The ROC curve is then plotted as FPR (on the x-axis) versus TPR (on the y-axis). These are computed as shown

$$TPR = \frac{TP}{TP+FN}, \tag{3.13}$$

$$FPR = \frac{FP}{FP+TN}. \tag{3.14}$$

The TPR terms are also referred to as the sensitivity or the recall of the classification system. Ideally, the classifier should have a TPR value of 1, starting at the top of the ROC curves, and the FPR to be 0, at the left-hand side of the ROC curve. On the other hand, a classifier with no judgement of the categories will appear as a straight line on the ROC curve.

Precision-Recall

One more diagnostic tool for studying imbalanced classes, which is also used in the experiments in this chapter, is the precision-recall curve [86]. The principles behind are similar to ROC, they are often used interchangeably, depending on the application and the field. The precision of a classification system is a metric that defines the number of correctly predicted positive categories. It is calculated by taking the total number of TPs divided by the sum of all TPs and FPs. The recall, on the other hand, is used to define the number of correct positive predictions that are achievable from all of the positive predictions. These can be seen in equations (3.15) and (3.16) as shown

$$Precision = \frac{TP}{TP+FP}, \tag{3.15}$$

$$Recall = \frac{TP}{TP+FN}. \tag{3.16}$$

This section reviews the performance evaluation metrics used in the experiments performed in this chapter. It begins with an explanation of why the problem of imbalanced categories is important in classification. It then gives an overview of the concept behind the confusion matrix as well as provides an example of computing the matrix for multi-class labels. Finally, the section ends with explaining precision-recall and ROC curves. The next section briefly reviews the datasets used in the experiments.

3.2.6 Datasets

This section entails the description of the datasets used in the evaluation of the proposed model CNN-GP. These include the MNIST [90], Fashion-MNIST [190], CIFAR-10 and CIFAR-1000 [84]. The section reviews the three respectively. In particular, this section includes details regarding the sizes of the images in the set, whether the images are coloured or grayscale, the pre-processing techniques used benchmark status in AI and DNN literature.

MNIST

The MNIST dataset [90] consists of images of secluded handwritten digits. Each of the images has a size of 28x28 pixels. The total number of images is 70,000, of which 60,000 are used for training and 10,000 for testing. All of the images in MNIST are size-normalized and centered to a fixed size of 20x20. In doing so, the aspect ratio of the images is preserved. Original images were black and white (i.e. binary or bi-level images). However, applying anti-aliasing techniques via the normalization process results in images being in greyscale. The 28x28 size images are computed using the center of mass of the pixels and later translating images so that the position of the center matches the 28x28 field. MNIST is an ideal go-to dataset for beginners to try learning methods and pattern recognition techniques on a real-world dataset since users need not spend time and effort on the pre-processing and formatting of the data. All of the images in MNIST are labelled according to the digits that vary from 0-9, therefore, there are a total of 10 classes. As of the writing date of this thesis, the current state-of-art performance accuracy on MNIST is 99.75% by Capsule Convolutional Neural Networks [66].

Fashion-MNIST

Fashion-MNIST [190] is a dataset of images taken from the article images of fashion garment brand Zolando[1]. Each shot of the product is made to demonstrate various visual aspects of the product, e.g. taking shots at various angles, front or back. The original images are photographed in a light-grey background and shot as coloured images. These are then stored in a 762x1000 JPEG format, saved in different resolutions ranging from 51 x 73 (large size) to 11x28 (thumbnail size). For preparing Fashion-MNIST images, however, only the front angle of the thumbnail-sized version of images is taken, which are later resized to 28 x 28 images in the post-processing stage. There are a total of 60,000 training images and 10,000 testing images. There are a total of 10 classes. The labels range as follows : 0; T-shirt, 1; Trouser, 2; Pullover, 3; Dress, 4; Coat, 5; Sandal, 6; Shirt, 7; Sneaker, 8; Bag, 9; Ankle boot. Fashion-MNIST shares similarities with MNIST in terms of size and structure. Hence, it can be used to evaluate machine learning algorithms since MNIST is an easy dataset to work with. The work from [99] achieves the state-of-art performance on Fashion-MNIST with the test accuracy of 96.91%.

CIFAR-10 and CIFAR-1000

Roughly similar to MNIST [90] and Fashion-MNIST [190], CIFAR-10 [84] consists of 50,000 images and 10,000 test images. The difference is that the size of each image is 32x32x3 instead of 28x28x1 in MNIST and Fashion-MNIST. The extra number in the channel suggests that the images from CIFAR-10 are coloured unlike greyscale ones from

[1]https://corporate.zalando.com/en/

MNIST and Fashion-MNIST. Furthermore, there are a total of 10 classes in CIFAR-10, ranging from aeroplanes, frogs, cars, birds, cats, dogs, ships, deer, horses and trucks. To prevent an imbalance of classes, there are a total of 6,000 images for each class. The classes are mutually exclusive which means there is no overlap between them for a particular image. CIFAR-100, on the other hand, is similar to CIFAR-10, except that the number of classes is 100, each class having 600 images each. There are a total of 500 training images and 100 test images for each of the classes. Currently, the work from [36] has the state-of-art performance on CIFAR-10 with a test accuracy of 99.70%

3.3 Methodology for Continual Learning

Previous sections discussed the definition of continual learning as well as various techniques used in regularisation methods in CL [28]. In particular, the Bayesian approaches to regularisation were explored. Here, two of the popular methods VCL [121] and UCB [32] were reviewed. Then, it was considered that these approaches adopt a multi-classifier approach. A setting in which there exists a separate classifier for each of the categories. The next section details the novel contributions in the area of continual learning.

3.3.1 Novel Contributions in the Area of Continual Learning

The method proposed in this section aims to provide a single-classifier solution to continual learning by using a Gaussian process with a base convolutional network architecture (CNN-GP) (see Chapter 4). Post softmax samples are used to estimate the variance. The variance is used to update the learning rate parameters. Here, two settings are adopted. In one, the weights of the CNN are deterministic and only the GP learning rate is updated. In the second setting, prior distributions are used to replace the standard scalar weights of the CNN and both the learning rates of the CNN and the GP classifier are updated. The algorithm is trained on two variants of the MNIST dataset [90], i.e. split-MNIST and permuted-MNIST [193], as well as on sequential CIFAR-10 and CIFAR-100 [84]. Results are compared with UCB's [32] multi-classifier approach. These show that the proposed algorithm in the Bayesian setting outperforms the UCB in some tasks with a comparable accuracy alongside robustness to images perturbed with Gaussian noise.

3.3.2 Architecture Detail: Convolutional Network Gaussian Process

Compared to the model presented in the previous chapter, this version of the model is slightly modified in terms of weight settings, but overall it follows the same architecture format. For a quick recap, this is a base model CNN with GP placed after. The CNN has two convolution layers of 32 and 64 filters of 3 x 3 kernel size and one fully connected layer. Convolution of images results in the decomposition of features. These features are learned hierarchically, starting from simple features (e.g. edges) in earlier layers and more complex at the end [48].

A max-pooling layer is introduced between the second layer and two dropout layers. The dropout layers are assigned the probability 0.5 and 0.25 respectively. Pooling layers downsample the features and dropout is used as a regularizer. The fully connected layer, on the other hand, converts the flattened outputs to a 16 x 128 feature vector. The softmax function outputs a vector of size 16 x 10 containing a prediction for each of the 10 classes in the permuted-MNIST [50] dataset, a 16 x 2 is made to output for split-MNIST [193].

The training algorithm consists of two components a) updating of the weights of CNN and parameters of GP based on the maximum likelihood loss and b) backpropagating the maximum likelihood loss regularised with KLD. The loss compares the ground truth labels and the predictions from the GP classifier. In the first step, the prediction from the GP classifier is compared with the ground truth labels using maximum likelihood error. The error obtained is then used to update the parameters of the GP (the lengthscale L, amplitude σ^2_{RBF} and the variational parameters ψ from $q_\psi(\theta)$). It is also used to update the weights of the CNN, specifically the weights of the convolutional and the fully connected layers. For inferring from the GP, an approximated variational inference is used since categorical likelihood is needed for classification. This is followed by a softmax function that converts the features into a vector of probabilities for each of the classes.

In the second half, the images of the mini-batch are replaced with their nearest neighbours [48] but the labels are kept the same. This step is inspired by the work of [168].

The difference is that they focus on noisy labels while the methods adopted in this section focus on replacing images in the mini-batch. Firstly, the randomly sampled mini-batches are ranked. This is followed by the random selection stage where the top 10 nearest neighbours of the mini-batch samples are selected to replace the original samples. The noisy artificial samples are then fed to both the CNN and GP. The KLD loss is used to encourage both the GP and the CNN to remain consistent in their predictions. These losses are used to train both models to be less influenced by the noisy images. In doing so, they learn to be robust. A full description of the algorithm is provided in Algorithm section 3.3.6 along with notation definitions in Table 3.1. Details on the learning rate updates based on the uncertainty measures are provided in the section.

3.3.3 Learning Rate Update Rule in CNN-GP

This section discusses in detail how the CNN-GP adapts to the change in the dataset by updating the learning rates using the uncertainty outputs. To recall, there are two versions of the CNN-GP architecture used in this chapter. These are the deterministic and the Bayesian setting.

In the deterministic setting, the weights of the hidden layers in the CNN component of the CNN-GP model are scalar. When the dataset is made to change, the learning rate updates are only performed on the weights of the GP classifier. To recall, these are the lengthscale, the amplitude and the variational parameters. For a further recap, readers are encouraged to read Chapter 2 and Section 2.1.15. Collectively, these can be denoted as γ_{GP}.

In the Bayesian setting, the weights of the CNN are sampled from a prior distribution. This distribution is specified as a scaled mixture of two Gaussian distributions with variance 0.0 and 6., respectively. The scaling coefficient of this mixture is set to 0.25. The above specification is inspired by the approach in UCB [32]. The full algorithm for the learning rate update for both sets is given in Algorithm 2.

The key thing to note here is that both the learning rates of the CNN and the GP components are updated separately, using their uncertainty measures. To be more specific, the GP classifier uses the post-softmax sample variance σ_{GP}. The CNN feature extractor

Algorithm 2 CNN-GP Learning Update Rule

```
1: for episode = 1, $\cdots$, 200 do
2:     train the CNN-GP model                                  ▷ see Algorithm 3
3:     validation $\rightarrow$ outputs the tuple (validation loss, $\sigma_{GP}$)
4:     if (validation loss < previous loss) then
5:         set $\rho \rightarrow \rho_{default}$                ▷ Default patience $\rho_{default} = 5$
6:         save weights as checkpoint files
7:     else
8:         set $\rho \rightarrow \rho - 1$                      ▷ Reduce patience by 1
9:         if $\rho$ < 0 then
10:            set $\mathscr{P} \rightarrow max(\sigma_{GP})/mean(\sigma_{GP})$   ▷ Set the proportionality factor
11:            set $\gamma_{GP} \rightarrow \gamma_{GP} \cdot \mathscr{P}$;       ▷ Update learning rate of GP
12:            if setting == ‘Bayesian’ then
13:                for all CNN layers do        ▷ Only convolutional and fully-connected
14:                    Compute $\sigma_{CNN} = \log(1 + \exp(\kappa_{CNN}))$
15:                    set $\gamma_{CNN} \rightarrow \gamma_{CNN} \cdot \sigma_{CNN}$
```

uses the variance from the scaled mixture of Gaussian distributions to model uncertainty. This is captured in all of the CNN layers with weights: convolutional and fully connected. For brevity, this can be collectively termed κ_{CNN}.

The problem with using the variance κ_{CNN} to update the learning rate is that it can output a negative value. One solution is to use a soft-plus normalisation, as adopted in UCB [32]. Therefore, the normalised uncertainty in CNN is represented as $\sigma_{CNN} = \log(1 + \exp(\kappa_{CNN}))$.

Then finally, updating the learning rates of both the GP (γ_{GP}) and the CNN (γ_{CNN}) involves multiplying the uncertainties with the learning rate. This approach is also adopted in UCB [32]. However, for the GP, the proposed learning rate update algorithm follows computing the proportionality factor $\mathscr{P}$. This is calculated as computing the fraction of the max and mean of the post-softmax sample variance. To recap, the samples from the GP posterior distribution are made to pass through the softmax function. Only the variance of the correctly classified samples is taken (see Section 4.3.2). This is then multiplied by the proportionality factor.

During the learning rate update process, care is taken not to update the learning rate on each step of validation. This is done to avoid drastic changes in the learning rate. Instead, a separate variable, denoted as ρ, is used to monitor the learning update as changes in the validation loss are observed. If the validation loss fails to lessen for five episodes, then the value of ρ falls to 0. After that, the learning rate, depending on the setting, is updated for both the CNN and GP components. Once updated, the default value of $\rho = 5$ is recovered. The next section discusses the datasets used in this chapter of the thesis.

3.3.4 Datasets: Split-MNIST and Permuted-MNIST, sequential CIFAR-10 and CIFAR-100

The CNN-GP architecture is further tested on the datasets split-MNIST and permuted-MNIST [193]. These are modified versions of the original MNIST dataset [90]. The idea is to split the MNIST dataset into a separate collection of categories. These are then supplied in a sequence. The architecture is also tested on sequential CIFAR-10 and CIFAR-100 [84]. The former consists of 10 categories while the latter consists of 100 categories.

The split-MNIST introduced by [193], divides the full MNIST training set into five tasks of consecutive classes. Each task contains two classes. With split-MNSIT, there is a total of five tasks each with categories: 0-1, 2-3, 4-5, 6-7, 8-9. Each task is further divided into training and validation set by 80-20% ratio. Training is performed by first introducing task 1 with categories 0-1. Once the CNN-GP model is trained on task 1, the weights are saved and the next task is introduced. This is repeated until the final task 8-9 is learned. To further test the proposed model the system is introduced to the permuted-MNIST [193]. In permuted-MNIST, all of the pixels in the MNIST dataset are randomly permuted. This is done separately for ten tasks. Permuted-MNSIT is more challenging than split-MINIST.

This work follows the configuration adopted in UCB [32]. Here, the default setting contains ten permutations set from a seed number of 100 for the random number generator used in the Torch library. Furthermore, for both split-MNIST and permuted-MNIST, the averaged accuracies are computed by dividing the correctly classified samples by the total number of samples. This is done separately for each task in per-task accuracy evaluation and collectively on all tasks when testing on noisy images. The next section discusses the method used for generating noisy images.

The sequential CIFAR-10 and CIFAR-100 are divided into five different tasks. Each task contains $10,000$ examples. Each task is split further into training and validation examples. The ratio between training and validation sets are $80-20\%$, like with split and permuted-MNIST. Then, the training is only performed on the training set. After the CNN-GP architecture is tested on each task, it is then validated based on the validation set. During validation, multiple forward passes (100 times) of the GP are used to obtain the post-softmax sample variance. The variance is used to update the learning rate. However, this is only performed if the patience value approaches zero. The patience value is dropped by 1 after every validation step, provided the validation loss is less than that lowest error stored from previous episodes. Performance validation on each task is obtained through accuracy measures on each task averaged across mini-batches. For robustness analysis, each task is perturbed by Gaussian noise, this is discussed further in the next section.

3.3.5 Robustness Analysis with Gaussian Noise

This section aims to discuss in detail the experiments that test the performance of both the UCB and the CNN-GP solution in the presence of noisy images. Gaussian white noise is a popular example that is widely adopted across DNN and robust learning literature [29], [17]. This is also used as an extension to the performance validation experiments used in this chapter.

A most common way to add Gaussian noise to training set samples is by computing the standard deviation of a unit Gaussian distribution and then adding it to the training images. The strength of the noise can be changed by varying the standard deviation. However, using SNR to measure the strength of the noise can be more useful. The SNR has to be computed for each mini-batch of the testing set. This can be calculated as shown in equation (3.17) (taken from [47])

$$SNR = 10 \cdot \log_{10}\left[\frac{\sum_0^{n_x-1}\sum_0^{n_y-1}[r(x,\,y)]^2}{\sum_0^{n_x-1}\sum_0^{n_y-1}[r(x,\,y)-t(x,\,y)]^2}\right]. \tag{3.17}$$

Here, $r(x,\ y)$ and $t(x,\ y)$ refer to the training and test images respectively. Then, n_x and n_y refers to respective sizes of training and test images. The terms $\sum_0^{n_x-1}, \sum_0^{n_y-1}$ refer to the summation of both numerator and denominator elements w.r.t the sizes values.

3.3.6 Algorithm Description

This sub-section aims to briefly describe the training algorithm used in this chapter. The steps followed here are similar to those adopted in the previous chapter. The main difference is that instead of using the three secondary losses: KLD, Wasserstein distance and maximum correntropy, only the KLD loss is backpropagated along with the maximum likelihood. All experiments in this version of the training algorithm use the following default arguments; batch size=16, episodes=200, the initial learning rate of the GP classifier=0.1, number of neighbours to replace with images in the mini-batch=10, KLD scaling factor = 1, $\rho = 5$.

3.4 Summary

This chapter provided the proposed methods for areas tackled in this thesis: semantic segmentation, robust learning and continual learning. Firstly, this chapter focused on presenting the proposed architecture AdvSegNet. It was seen how AdvSegNet utilises generative adversarial learning and uncertainty-aware learning to aim to improve the performance of the Bayesian SegNet, one of the popular architectures used in semantic segmentation, in the presence of a physical attack. This attack damages the input resolution (i.e. receptive field) of the SegNet architecture. Then, the focus of the chapter focused on introducing the CNN-GP architecture. The chapter considered various layers configurations within the architecture. Secondly, the proposed training algorithm for the CNN-GP was discussed, followed by the discussion on the three proposed loss functions used for regularisation, i.e. KLD, Wasserstein distance and maximum correntropy. The final portion of the chapter discussed the proposed methods for continual learning. In particular, proposing methods for comparing the training of a single-classifier in continual learning to the state-of-the-art multi-classifier methods UCB.

Algorithm 3 CNN-GP Training algorithm for continual learning problem

```
1:  **Require** $M$: episodes, $\gamma$: learning rate (GP), $k$: neighbors sampling number, $N$: batch size, $\beta$: KLD scaling factor, $\rho$: patience
2:  **Do** initialization of weights: $\theta^{m}_{CNN}$, $\theta^{m}_{GP}$
3:  **for** $m = 0, \cdots, M$ **do**
4:      Sample mini batch ($x_i$,$y_i$), of length $N$ from dataset $D = X, Y$ where $X$ and $Y$ are 4-D tensors holding images and labels from the entire dataset, where $x_i \in \mathbb{R}^{hxwxc}$ (image height, width and channel) and $y_i \in \mathbb{R}^{1xC}$ ($C$ is total number of classes).
5:      **BEGIN** Update the parameters of the GP and the weights of the CNN using the maximum likelihood loss $\mathscr{L}_{MLE}$
6:      **do** $\rightarrow$ forward pass of CNN base $f^{CNN} : x_i \rightarrow z_i$, where $z_i \in \mathbb{R}^{ZxC}$ and $Z$ is the number of hidden units' feature outputs passed from final fully connected layer of CNN base feature extractor
7:      **do** $\rightarrow$ forward pass of GP $f^{GP}(z_i)$ to obtain the posterior likelihood $p\big(y_i|f^{GP}(z_i); \mu_i, \sigma_i^2\big) = \mathscr{N}\big(\mu_i, K_i\big)$ where $\mu_i$ represents the mean of the GP and $K_i$ is the kernel (i.e. squared exponential $K_i = A\exp\Big[-\frac{1}{2}\big(\frac{\delta x_i}{\lambda}\big)\Big]$ and $\mathscr{N}$ represents the Gaussian distribution
8:      **do** $\rightarrow$ Compute the expected log likelihood to obtain max likelihood loss: $\mathscr{L}_{max} \approx \sum_{i=1}^{N} \mathbb{E}_q\Big[\log\big(p(y_i \| f^{GP}(x_i); \mu_i, \sigma_i^2\big) - \beta D_{KL}\big(q(u_i)\|p(u_i)\big)\big)\Big]$
9:      **do** $\rightarrow$ Compute gradients of loss w.r.t weights of CNN base feature extractor and GP : $\frac{\partial \mathscr{L}_{max}}{\partial \theta_{GP}}, \frac{\partial \mathscr{L}_{max}}{\partial \theta_{CNN}}$
10:     **do** $\rightarrow$ Update the parameters of GP and CNN feature extractor for $m^{th}$ episode: $\theta^{m+1}_{CNN} \leftarrow \theta^{m}_{CNN} - \gamma\frac{\partial \mathscr{L}_{max}}{\partial \theta^{m}_{CNN}}.\mathscr{L}_{max}$, $\theta^{m+1}_{GP} \leftarrow \theta^{m}_{GP} - \gamma\frac{\partial \mathscr{L}_{max}}{\partial \theta^{m}_{GP}}.\mathscr{L}_{max}$
11:     **BEGIN** backpropagation of KLD regularised with maximum likelihood
12:     **do** $\rightarrow$ Replace input samples with top-10 neighbours $\hat{x}_i$
13:     **do** $\rightarrow$ forward pass of the CNN base feature extractor $f^{CNN} : \hat{x}_i \rightarrow \hat{z}_i$
14:     **do** $\rightarrow$ forward pass of the GP $f^{GP} : \hat{z}_i \rightarrow p\big(\hat{y}_i|f^{GP}(\hat{z}_i); \hat{\mu}_i, \hat{\sigma}_i^2\big) = \mathscr{N}\big(\hat{\mu}_i, K_{\hat{x}_i}\big)$
15:     **do** $\rightarrow$ Calculate similarity loss $\mathscr{L}^{GP}$ between the labels $y_i$ and the GP classifier posterior mean $\hat{\mu}_i$ using the KLD
16:     **do** $\rightarrow$ Update the new parameters of $\hat{\theta}^{m}_{GP}, GP : \hat{\theta}^{m}_{GP} \leftarrow \theta^{m}_{GP} - \gamma\frac{\partial \mathscr{L}^{GP}}{\partial \theta^{m}_{GP}}\mathscr{L}^{GP}$
17:     **do** $\rightarrow$ Update parameters of CNN feature extractor : $\hat{\theta}^{m}_{CNN} \leftarrow \theta^{m}_{CNN} - \gamma\frac{\partial \mathscr{L}^{GP}}{\partial \theta^{m}_{CNN}}.\mathscr{L}^{GP}$
18: **end for**=0
```

Chapter 4

Experimental Results

Chapter Overview

- Detail the experiments carried out and benchmark the performance evaluation for AdvSegNet
- Introducing and testing the proposed model CNN-GP in presence of adversarial attacks and noise.
- Introducing and testing continual learning with the CNN-GP model.
- Discuss results relating to robustness analysis between CNN-GP and UCB.

4.1 Results on Semantic Segmentation

The previous sections discussed the methodology adopted in the proposed framework AdvSegNet as well as the evaluation metrics. These considered perspectives from the architecture details, the individual loss functions, the training regime and various training specifications e.t.c. This section, however, will consider the results. Specifically, the proposed framework is tested on a small-sized dataset CamVid [16]. An extensive experiment is also performed on the dataset Sun RGB-D [169].

4.1.1 Performance Validation of Bayesian and the AdvSegNet on CamVid, Sun RGB-D and Pascal VOC

First, the weights of a pretrained Bayesian SegNet are fed to the encoder and decoder layers of the AdvSegNet. In particular, the convolutional ones. Then, the network is simulated with an attack. The involves reducing the input receptive field of the SegNet. Though there are many forms of such attacks in the literature of machine learning security [173], [49], the exploited method uses a simple reduction of the receptive field. Particularly, the input size of the Bayesian SegNet is reduced from 360x480 to 128x128. This is introduced in the first episode (Epoch = 0).

The network is then trained to 2000 episodes. There are two configurations used within the experiments. The first one, termed the classical training method, involves training the AdvSegNet with simple cross-entropy loss, without considering the discriminators. Considering the second setting, termed the adversarial training method, this involves training the AdvSegNet in the presence of the discriminators UC and QC. Both configurations are trained on CamVid [16] (see Figure 4.1A). A separate experiment is performed on Sun RGB-D [169] and Pascal VOC [35] (see Figure 4.1B). Each of the experiments was carried out 100 times, the points represent the average values obtained at each episode, while the vertical bars represent the standard deviation ($\approx 3-5\%$). Experiments carried out in Figure 4.1B are trained from scratch rather than using pretrained weights. It is only performed with a single configuration that involves the use of discriminators. The result for the change in validation accuracy w.r.t the episode number is presented in Figure 4.1 A and B. The dashed line in Figure 4.1A sets the Bayesian SegNet performance benchmark on CamVid.

4.1.2 Evolution of Loss Functions in the AdvSegNet on CamVid

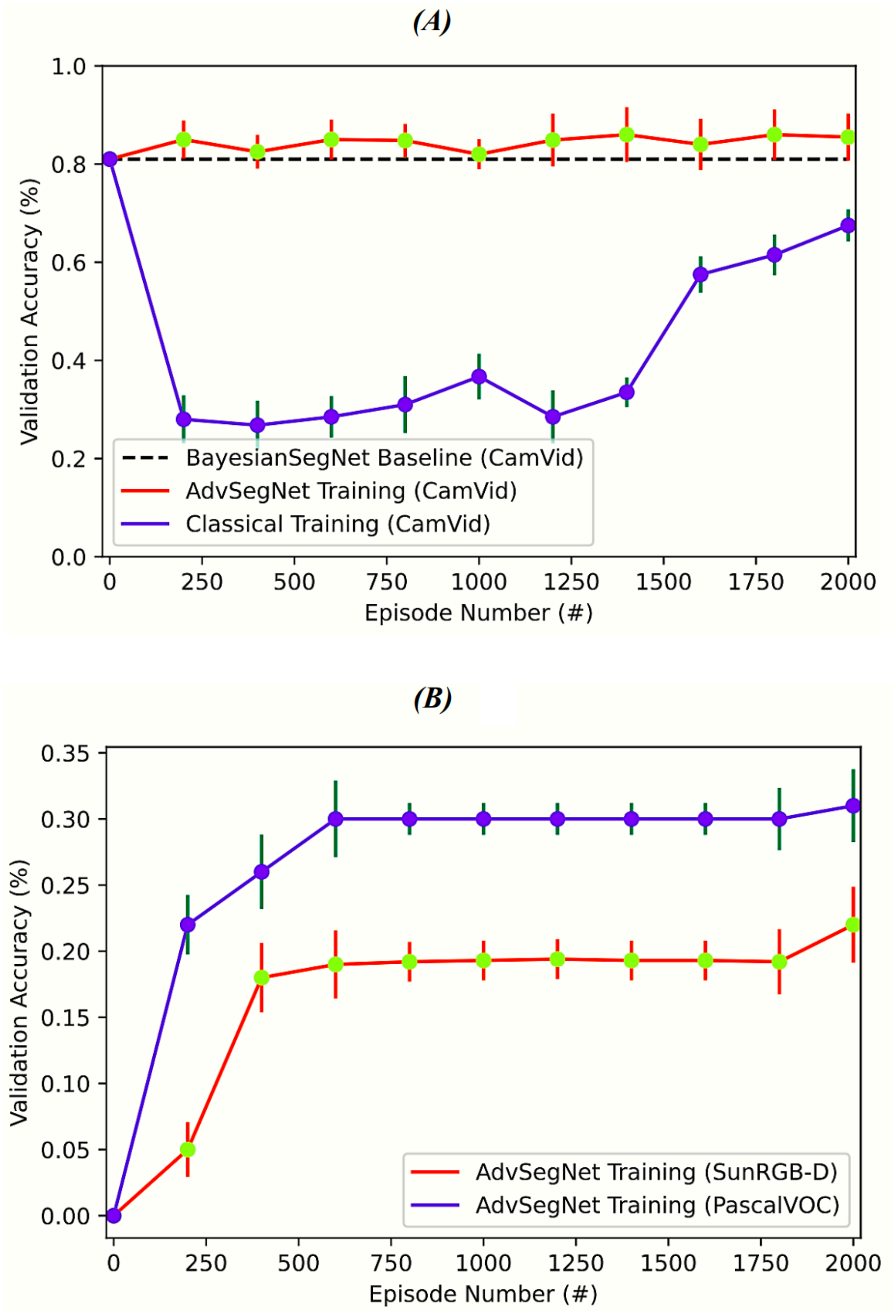
This experiment aims to evaluate the evolution of the adversarial losses for the critics UC and QC. The result is plotted in Figure 4.2. The training procedure involved here follows similar steps to the performance validation experiment. However, in this experiment, the values of both the critic loss functions for UC (3.5) and QC (3.4) are monitored after each episode for 2000 total episodes. Additionally, the cross-entropy loss function for training the AdvSegNet is also monitored. This is plotted as the blue line in Figure 4.2. The loss value of QC is plotted as the red line and the loss value of UC is plotted as the dashed green line.

4.1.3 Effect of Changing the Learning Rate on the Performance on Sun RGB-D

In the previous sections, it was reviewed that GAN training is very sensitive to learning rate changes. This is especially difficult with complex datasets. To train effectively on the Sun RGB-D dataset, choosing the optimum learning rate is crucial. Hence, a separate experiment is performed that observes the sensitivity of validation accuracy on various learning rate configurations. This is shown in Figure 4.3. The configurations of the learning rate are split into three parts in this experiment. At first, the learning rate of the segmentation network half of the AdvSegNet is fixed at 1 and the learning rate of the critics UC and QC is fixed at 0.2. Then, the validation accuracy is observed as it evolves with the increasing number of episodes. This is plotted as the crossed green line in Figure 4.3.

In the second configuration, the learning of the segmentation half of the AdvSegNet is decreased to 0.1 and the learning rates of the critics UC and QC are decreased to 0.02. In the third, this is further decreased to 0.01 and 0.002 respectively. These are plotted as the crossed blue line and the crossed red line respectively. The experiment is carried out for 2000 episodes.

Fig. 4.1 Results for training the pretrained Bayesian SegNet for 2000 episodes in both scenarios of using the AdvSegNet and without discriminators UC and QC in classical training. Here, (A) shows results for experimenting with datasetc CamVid [16] and (B) shows results for Sun RGB-D [169] and Pascal VOC [35]



.

Fig. 4.2 The evolution of loss functions for the AdvSegNet versus the number of episodes. The light blue line shows loss values for the cross-entropy loss, the loss values of the critics QC and UC are shown as red line and dashed green line respectively

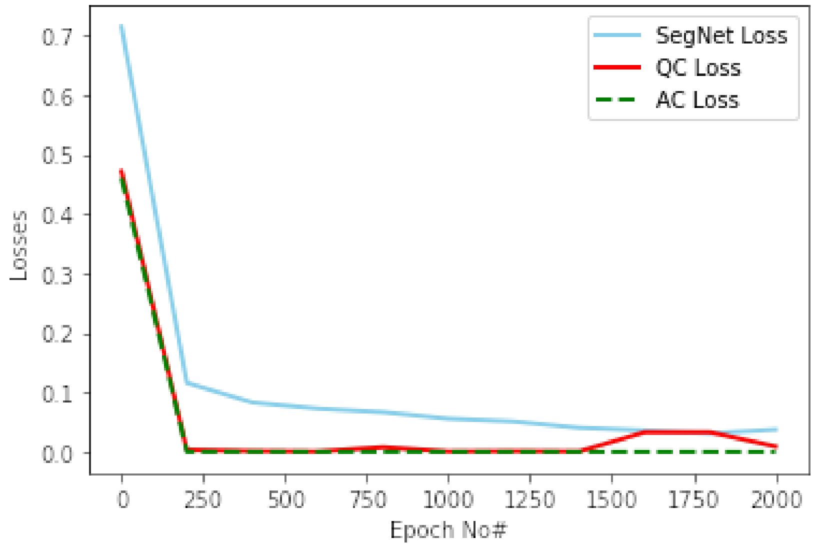
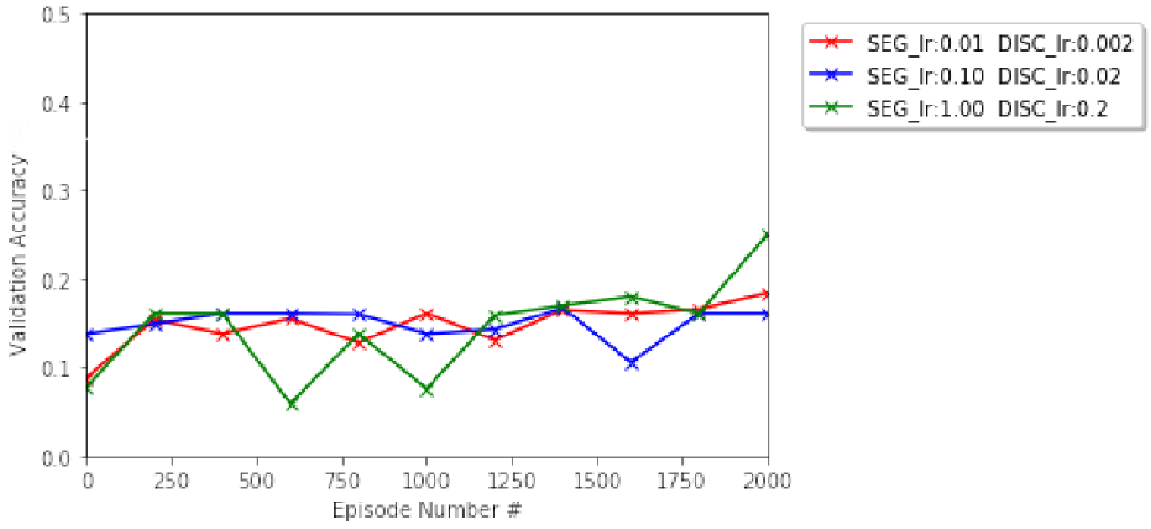


Fig. 4.3 Validation accuracy comparison for various learning rate configurations of the AdvSegNet on the Sun RGB-D dataset. The term SEG lr corresponds to learning rate associated with the AdvSegNet and DISC lr for the learning rate of the discriminators



4.1.4 Comparison of Performance with State-of-the-Art and Other SegNet Architectures

To recap, the main objective of this research is to improve the performance of the Bayesian SegNet by utilising the variance information from the dropout samples. Hence, a separate experiment is performed to compare the performance of the AdvSegNet with various variants of the Bayesian SegNet as mentioned in [7]. In this experiment, the Bayesian SegNet variants are trained using the same hyperparameters using the AdvSegNet training. The list of these along with their default values is provided in Algorithm 4.

Furthermore, the proposed framework is then compared with the state-of-the-art segmentation architectures including DeepLab [20] and the original SegNet [7] in Table 4.1. The SegNet-Basic version is a smaller sized version of the original SegNet [7]. It consists of 4 encoder layers and 4 decoder layers. Similar to the original SegNet, max-pooling and subsampling is performed after every layer in the encoder of SegNet-Basic. Furthermore, the decoder layers of SegNet-Basic also use pooling indices from the encoder to upsample in the decoder network. Batch normalisation layers are also placed after every convolutional layer in the encoder and the decoder network. In both the networks, the SegNet and SegNet-Basic, no biases are added after the convolutional layers and no ReLU activations are used in the decoder network.

Algorithm 4 AdvSegNet, our proposed algorithm. All experiments in our paper use the following default arguments. batch size $= 1$, $t_{critic} = 5$ or 25, $\gamma_{seg} = 0.005$, $\gamma_{disc} = 0.0005$, $c = 0.01$, $epochs = 2000$

```
Require: $t_{critic}$ : critic episodes, $\gamma_{seg}$: AdvSegNet learning rate, $\gamma_{disc}$: critic learning rate, $clip$: clip parameter for critic weights, $epochs$: training episodes for AdvSegNet
Do initialization

for $epoch = 0, ..., epochs$ do
    if $epoch < 25$ or $epoch = 500$ then $t_{critic} = 25$
else $t_{critic} = 5$
Sample an image batch $x_n$ from the dataset of $N$ training samples

        for $epoch = 0, ..., t_{critic}$ do Obtain mean prediction: $\mu_n$ from (3.1)
Compute loss $\mathscr{L}_{QC}$ from (3.4)
Update loss: $\theta_{QC} \leftarrow \theta_{QC} + \gamma_{disc} \cdot Adam\ (\theta_{QC},\ \mathscr{L}_{QC})$
Clip weights: $\theta_{QC} \leftarrow clip\ (\theta_{QC}, -c, c)$
end
            for $t = 0, ..., t_{critic}$ do Obtain uncertainty: $\sigma_n$ from (3.2)
Compute loss $\mathscr{L}_{UC}$ from (3.5)
$\theta_{UC} \leftarrow \theta_{UC} + \gamma_{disc} \cdot Adam\ (\theta_{QC},\ \mathscr{L}_{UC})$
$\theta_{UC} \leftarrow clip\ (\theta_{UC}, -c, c)$
end Compute SegNet loss $\mathscr{L}_{SEG}$ from (3.3)
Update loss: $\theta_g \leftarrow \theta_g + \gamma_{seg} \cdot Adam\ (\theta_g,\ \mathscr{L}_{SEG})$
end
```

4.1.5 Comparison of Predicted Labels and Variance on CamVid Example

The purpose of this experiment is to compare both the prediction and the variance maps (model uncertainty) obtained from two different configurations of the AdvSegNet. These

Fig. 4.4 Segmented output labels and uncertainty map results from both the Bayesian SegNet trained with classical training (A, C, E, G) and the Bayesian SegNet trained with adversarial training (AdvSegNet) (B, D, F, H). The figures on the left column (A, C, E, G) represent results for the Bayesian SegNet trained classical training and the column for the AdvSegNet (B, D, F, H). The first row represents input images (A, B), the second row ground truths (C, D), the third row segmented output labels (E, F) and the final row model uncertainty outputs (G, H)

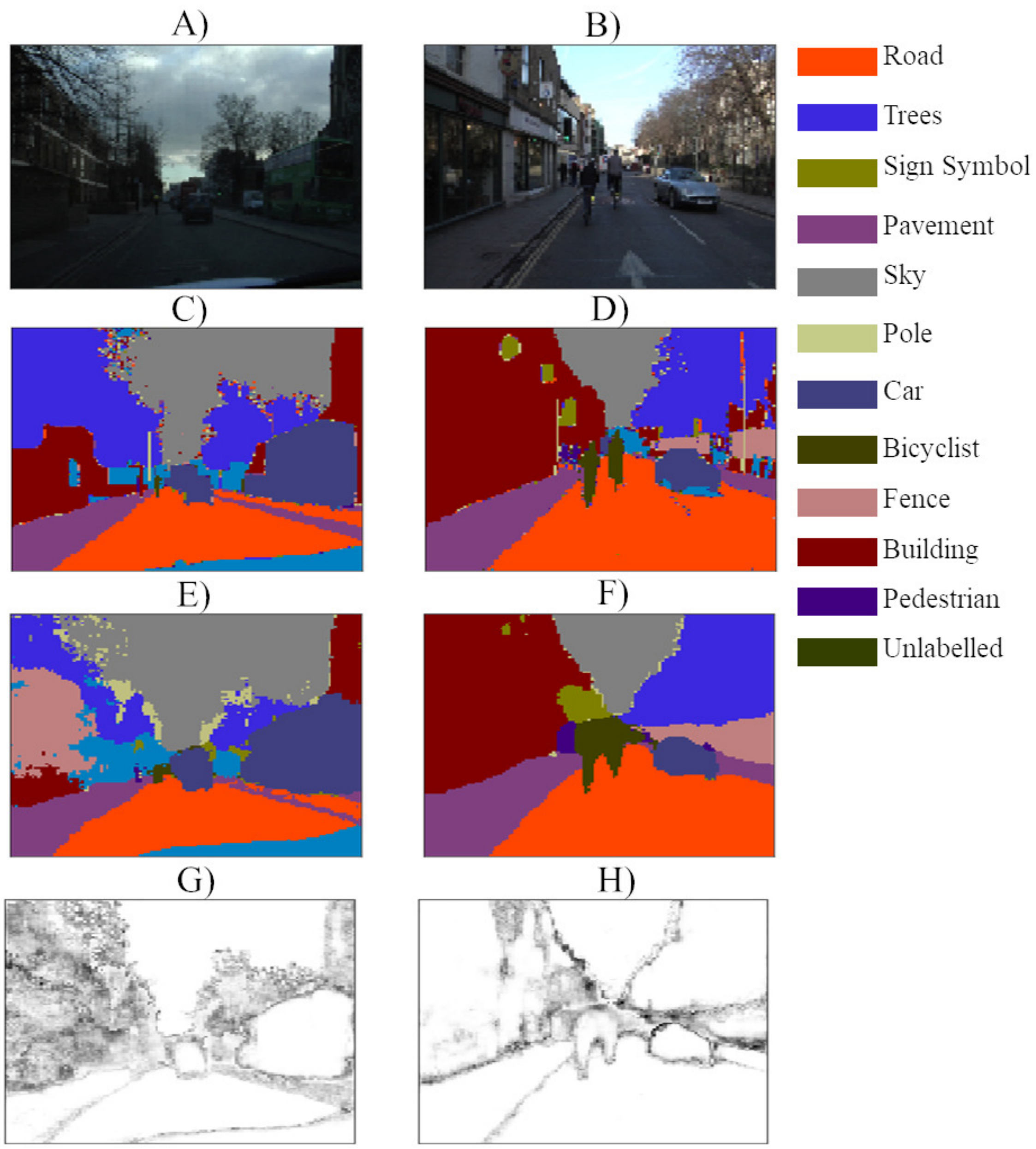


Table 4.1 Comparison of accuracies of state-of-art and SegNet based architectures included those made in this experiment highlighted as bold

Method	Building	Tree	Sky	Car	Sign-Symbol	Road	Pedestrian	Fence	Column-Pole	Side-Walk	Bicyclist	ClassAvg	GlobalAvg	MeanIU
SegNet-Basic [7]	80.6	72.0	93.0	78.5	21.0	94.0	62.5	31.4	36.6	74.0	42.5	62.3	82.8	46.3
SegNet [7]	88.0	87.3	92.3	80.0	29.5	97.6	57.2	49.4	27.8	84.8	30.7	65.9	88.6	50.2
BayesianSegNet-Basic [78]	75.1	68.8	91.4	77.7	52.0	92.5	71.5	44.9	52.9	79.1	69.6	70.5	81.6	55.8
BayesianSegNet [78]	80.4	85.5	90.1	86.4	67.9	93.8	73.8	64.5	50.8	91.7	54.6	76.3	86.9	63.1
DeepLab [20]	81.5	74.6	89.0	82.2	42.3	92.2	48.4	27.2	14.3	75.4	50.1	60.7	89.7	54.7
BayesianSegNet-128x128	**11.4**	**93.8**	**88.0**	**81.7**	**47.1**	**90.8**	**22.8**	**5.5**	**3.3**	**86.0**	**38.0**	**55.7**	**63.4**	**30.0**
AdvSegNet	**77.9**	**95.8**	**98.1**	**75.5**	**36.6**	**95.8**	**52.5**	**77.6**	**6.2**	**92.6**	**84.6**	**72.1**	**87.1**	**52.8**

configurations are similar to those adopted in the performance validation experiment, i.e. one where the AdvSegNet is trained without the critics and one with. Then, a sample from the test set is chosen as input for each configuration of the AdvSegNet separately. These samples share a majority of the categories. In the case of Figure 4.4, these include road, trees, sky, pole, car, bicycles, buildings and pedestrians. In the example, Figure 4.4A is fed to the classical training example and Figure 4.4B is fed to the adversarial training example. The respective ground truth labels are shown in Figure 4.4C and Figure 4.4D. The predictive labels are presented in Figure 4.4E for classical training example and 4.4F for adversarial training example. Finally, the respective variance (uncertainty) maps are shown in Figure 4.4G and Figure 4.4H. These maps are obtained by taking the variance of the dropout samples and averaging the channel number dimension to get a 2D greyscale image of size 128 x 128 (height and width).

4.2 Discussion for Results on Segmentation

Considering the results, from Figure 4.1, it is seen that the validation accuracy of the Bayesian SegNet trained under classical training method, shown as the crossed blue-line, approximately drops from 82% to 28% for the first 250 episodes. It takes up to 1000^{th} episode for the Bayesian SegNet trained under classical training to adapt to the sudden change in the receptive field. In contrast, the AdvSegNet adapts much faster and performs substantially better than the Bayesian SegNet trained with classical training. The validation accuracy of the AdvSegNet, shown as the crossed red line, also remains higher (approximately $\geq$ 80%) than the baseline the Bayesian SegNet shown as the dashed green line. On the other hand, training on Sun RGB-D is more challenging than CamVid. The AdvSegNet does not improve in validation accuracy from approximately 375^{th} to 1800^{th} episode.

When considering the test for the sensitivity of validation accuracy on various learning rate configurations for the Sun RGB-D dataset, Figure 4.3 shows that increasing the learning rate from 0.1 to 1.0 for both the SegNet and the discriminators leads to unsteady performance, with accuracies shifting from 10% to 15% after every 250^{th} episode. The accuracies shift more for large learning rates. For example for the learning rate value of 1, the accuracies shift from 5% to 30%. The shift is less when compared with low learning rates. For example for the learning rate of 0.01 the accuracies shift from 14% to 18% approximately.

Then, observing the change in loss functions for the discriminators UC and QC in Figure 4.2, it is observed that both the losses for training QC and UC, shown as the dashed-green and the solid orange line respectively, follow a similar trend. They decrease at a steeper rate than the SegNet loss, shown as the solid blue line. This observation strengthens the hypothesis which states that highlighting the impact of uncertainties improves the performance of neural networks.

Considering the comparison of the AdvSegNet with the state-of-the-art network the DeepLab and the Bayesian SegNet (128x128) in Table 4.1, it is seen that the performance in terms of global average accuracy of the AdvSegNet is similar to the Bayesian SegNet (with receptive field 128x128) in the majority of the class such as tree, road, column-pole and side-walk. In some classes, the validation accuracy of the AdvSegNet is higher than the

Bayesian SegNet (with receptive field 128x128) for example in building, pedestrians, fences and bicyclists. In brief, the AdvSegNet can outperform the Bayesian SegNet versions but loses performance in classes sign-symbol and column-pole.

The results from Figure 4.4 outline the impact of uncertainty on the performance of the Bayesian SegNet and the AdvSegNet. The uncertainty analysis in Figure 4.4F also supports the tabulated results that compare with the state-of-art. Considering the uncertainty maps for the Bayesian SegNet in Figure 4.4G and the AdvSegNet in Figure 4.4H, these results show that the amount of the uncertainty (grey levels of the uncertainty map) around the group of classes building, fence and trees is much higher in classical training of the Bayesian SegNet as opposed to the AdvSegNet. One possible conjecture is that this is the result of erroneous prediction of the fence category that is labelled building in the ground truth labels. A similar case appears for the Figure 4.4H. This is the uncertainty map for the AdvSegNet while Figure 4.4F is the segmented output label map. Here, it is evident that the AdvSegNet fails to detect the two poles present in the ground truth label in Figure 4.4D.

From the above observation, it can be conjectured that since the uncertainty is less in the AdvSegNet, it is more successful in isolating the tree class from the fence. Another key point to note is that the performance of the AdvSegNet is achieved with the receptive field less than half of those employed for the Bayesian SegNet and the Deep Lab, making it much easier and faster to train. On the system of GPU cluster (NVIDIA Tesla K80) provided by ShARC in the University of Sheffield which is used for training the AdvSegNet, the wall clock time to train AdvSegNet is approximately three hours. This is fast compared to training the Bayesian SegNet with a full receptive field of 256 x 256. Training the Bayesian SegNet at half the receptive field 128 x 128 is faster than the AdvSegNet but the performance of the AdvSegNet is superior. The AdvSegNet also learns to adapt to the drop in receptive field. This type of attack is black-box (see Chapter 4). These attacks do not have access to the entire training system but can harm the inputs of the system. Hence, the AdvSegNet is ideal for applications that require accurate results while in the presence of black-box attacks. In the authors' opinion, this is perhaps a very important test result obtained from the experiments since it shows how uncertainty is linked to performance and also how the impact of uncertainties changes the performance of neural networks but more importantly it agrees with the hypothesis. In the coming chapters, this relationship of reduced uncertainty and less erroneous prediction is studied furthermore, as well as exploring further ways to obtain uncertainty and experiment with different Bayesian architectures such as BNNs and Gaussian processes.

4.3 Results for Robust Learning

This section aims to consider the results based on the methods provided in the previous section. In particular, the proposed model CNN-GP is evaluated with respect to performance, variance sensitivity, the evolution of sensitivity to attack strength and computational time. Each of the experiments follows a certain aim. For example, the performance validation experiment aims to investigate whether adding a GP unit after a CNN improves its performance in the presence of noise (specifically AWGN) and motion blurring effects. The purpose of the uncertainty analysis experiment is to justify the results of the performance validation test by observing the uncertainty values from the post-softmax samples. Then, the purpose of the sensitivity test to attack strength is in two folds: one is to see if the proposed model is sensitive to increasing strength of adversarial attacks and two is to observe if this change is abrupt so that it can be used as an early-warning detection system for applications that can benefit from the last-minute manual intervention. Finally, the computational time experiment is carried out to test the speed of the system with the state-of-art MC dropout techniques widely adopted in standard DNN architectures.

4.3.1 Performance Validation: Accuracy Measures, Precision-Recall and Receiver Operating Characteristics Curves

This section aims to discuss the performance validation of the CNN-GP algorithm, the precision-recall curve and the ROC curve for evaluation on three datasets: MNIST [90], Fashion-MNIST [190] and CIFAR-10 [84]. This section first begins by describing how the above three evaluation procedures are carried on the proposed model, then moving to provide details about the results. This includes the presented plots and tables.

Before the experiments, the CNN-GP classifier is trained with the three different secondary losses. The purpose is to observe accuracy as a means of performance evaluation. The average results are calculated by dividing the averaged correct samples by the total number of samples. Experiments are run ten times and accuracy values are averaged. The standard deviation is ± 2. Then, the system is disrupted using the following two methods: a) an additive white Gaussian noise (AWGN) and b) motion blur (MB). The results are compared with the system version where no secondary losses are used (i.e. without regularization). These results are presented in Table 4.2. Next, the precision-recall and the ROC results characterize the accuracy of the proposed CNN-GP model. The precision-recall curves are plotted for each dataset in Figures 4.5, 4.7 and 4.9. The ROC curves are plotted in Figures 4.6, 4.8 and 4.10. The average precision (AP) and ROC area are two quantities that are obtained by averaging the individual curve entities. In the figures, the AP and the ROC area are placed in bracketed terms after the 'loss type (e.g KLD)' and the 'attack type (e.g MB)'. They help in grouping entities that give similar results and make it easy to read the curves individually.

Table 4.2 Performance Validation Based on Test Accuracy for Each Attack Type on Four Datasets and Three Conditions

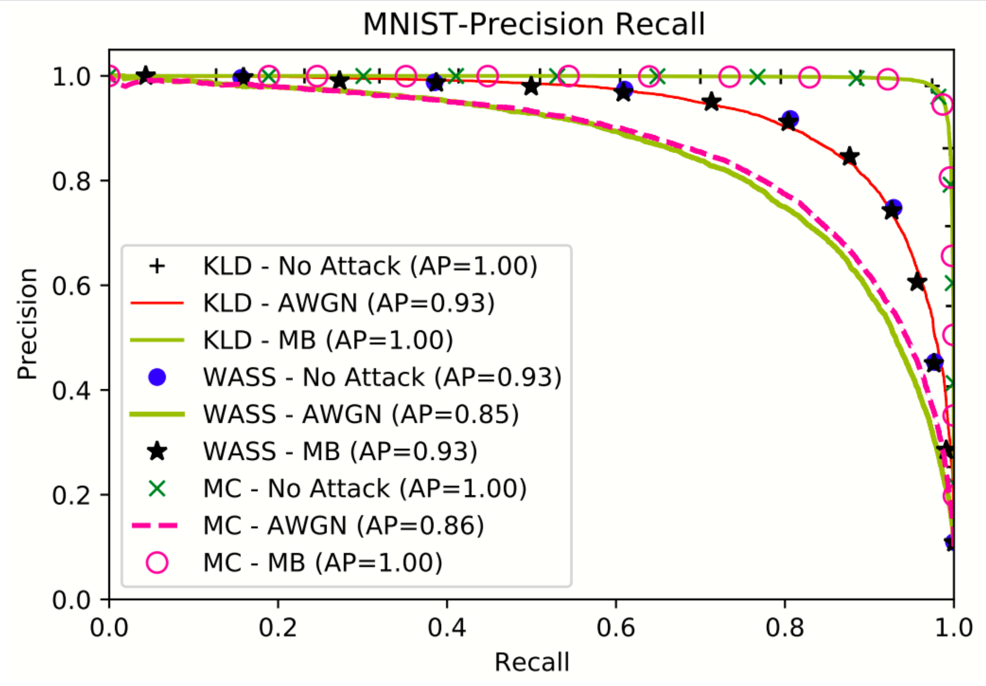
MNIST			
Loss Type	No Attack(%)	AWGN(%)	Blur(%)
No Regularisation	88	51	65
KLD	97	89	72
WASS	86	77	70
MC	97	78	75
WASS-FIRST	97	85	88

Fashion-MNIST			
Loss Type	No Attack(%)	AWGN(%)	Blur(%)
No Regularisation	85	32	12
KLD	88	53	76
WASS	81	56	72
MC	89	35	80
WASS-FIRST	89	54	77

CIFAR-10			
Loss Type	No Attack(%)	AWGN(%)	Blur(%)
No Regularisation	67	10	11
KLD	73	26	38
WASS	40	28	28
MC	65	25	38
WASS-FIRST	68	26	40

CIFAR-100			
Loss Type	No Attack(%)	AWGN(%)	Blur(%)
No Regularisation	24	10	8
KLD	30	10	12
WASS	27	10	10
MC	28	8	8
WASS-FIRST	27	10	10

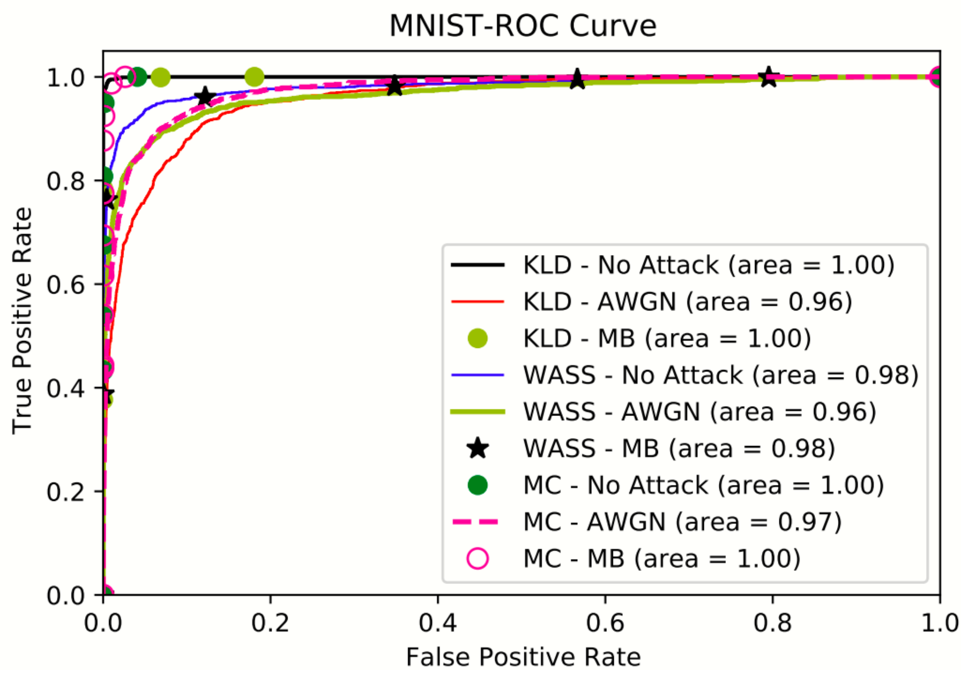
Fig. 4.5 Precision-recall for CNN-GP trained on the MNIST dataset. The plots are arranged according to their average precision. The plots are for each of the three losses: KLD, Wasserstein and maximum correntropy and for the three configurations: no attacks, Gaussian noise (shown as AWGN) and motion-blurring (shown as MB)



w

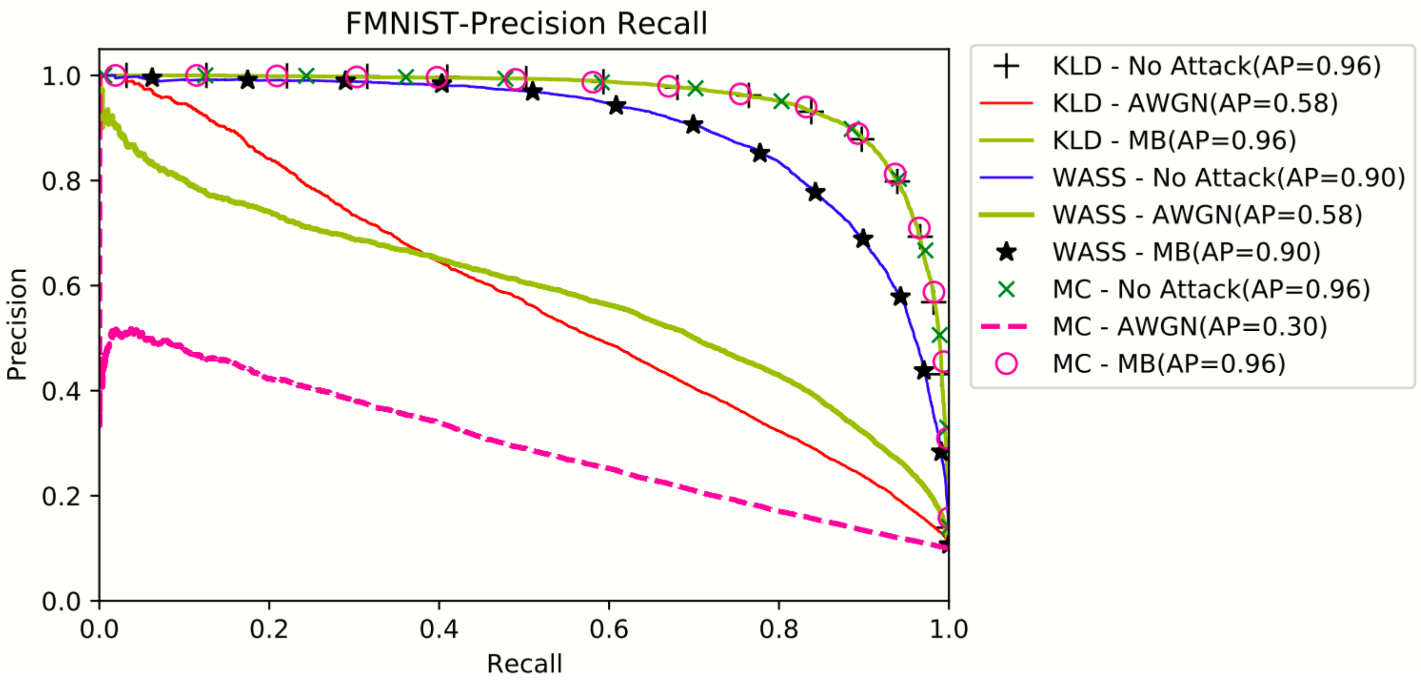
.

Fig. 4.6 ROC curve for CNN-GP trained on the MNIST dataset. The plots are arranged their ROC area. The plots are for each of the three losses: KLD, Wasserstein and maximum correntropy and for the three configurations: no attacks, Gaussian noise (shown as AWGN) and motion-blurring (shown as MB)



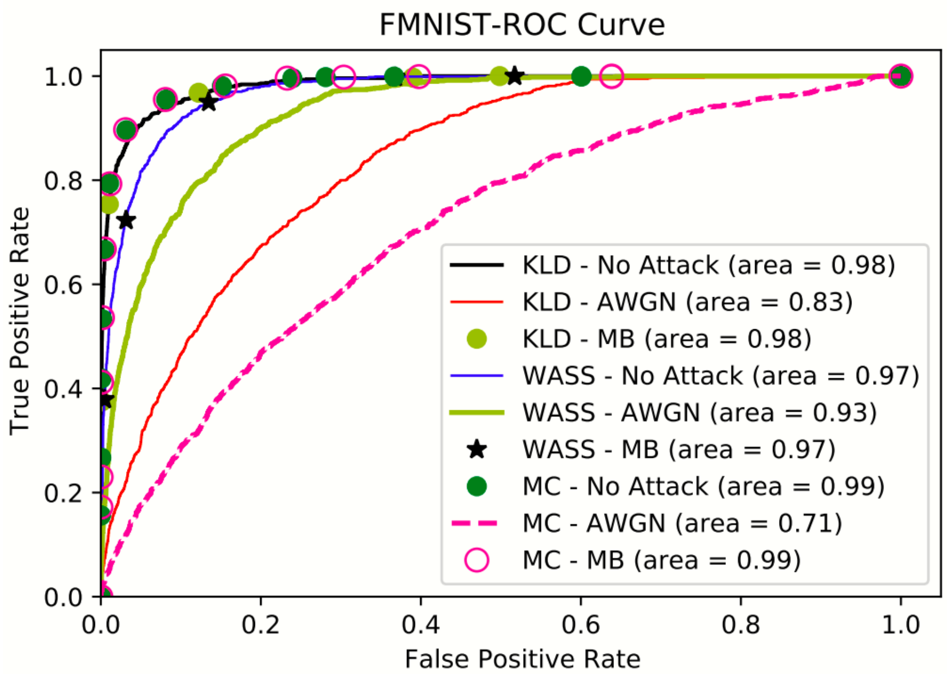
.

Fig. 4.7 Precision-recall for CNN-GP trained on the Fashion-MNIST dataset. The plots are arranged according to their average precision. The plots are for each of the three losses: KLD, Wasserstein and maximum correntropy and for the three configurations: no attacks, Gaussian noise (shown as AWGN) and motion-blurring (shown as MB)



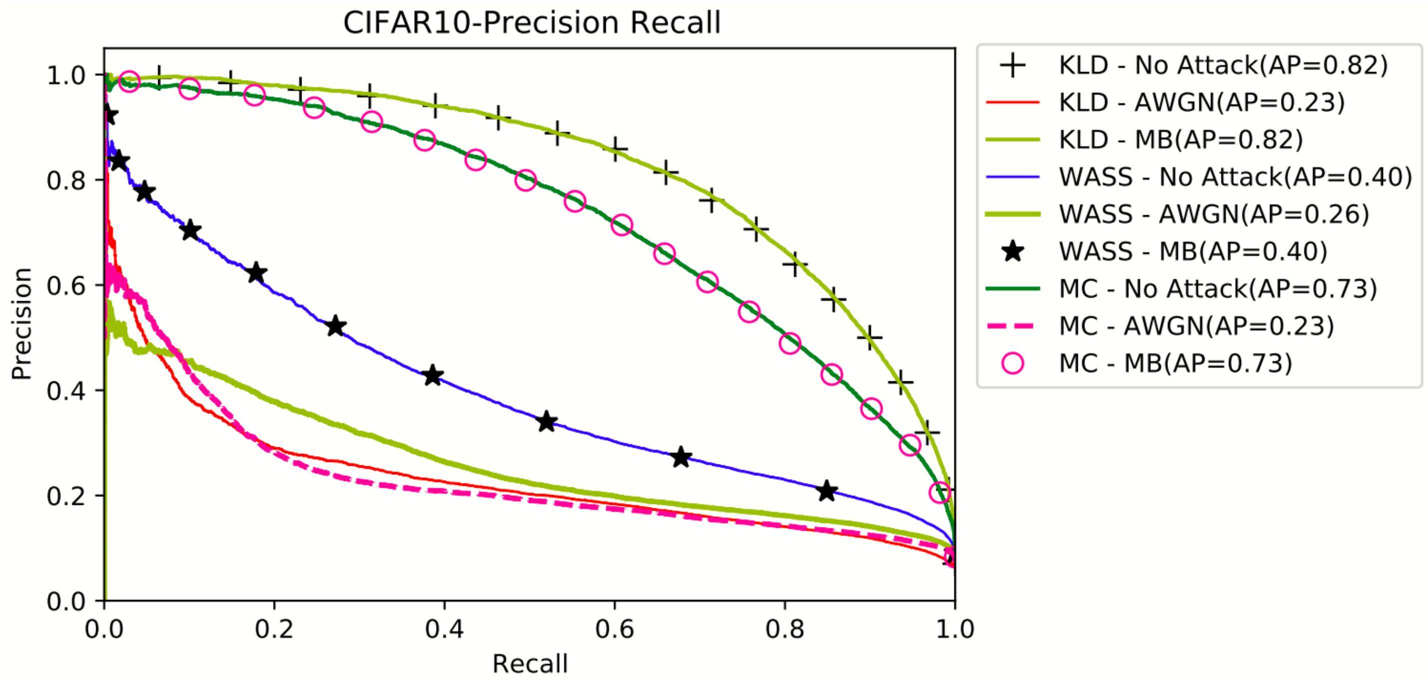
.

Fig. 4.8 ROC curve for CNN-GP trained on the Fashion-MNIST dataset. The plots are arranged their ROC area. The plots are for each of the three losses: KLD, Wasserstein and maximum correntropy and for the three configurations: no attacks, Gaussian noise (shown as AWGN) and motion-blurring (shown as MB)



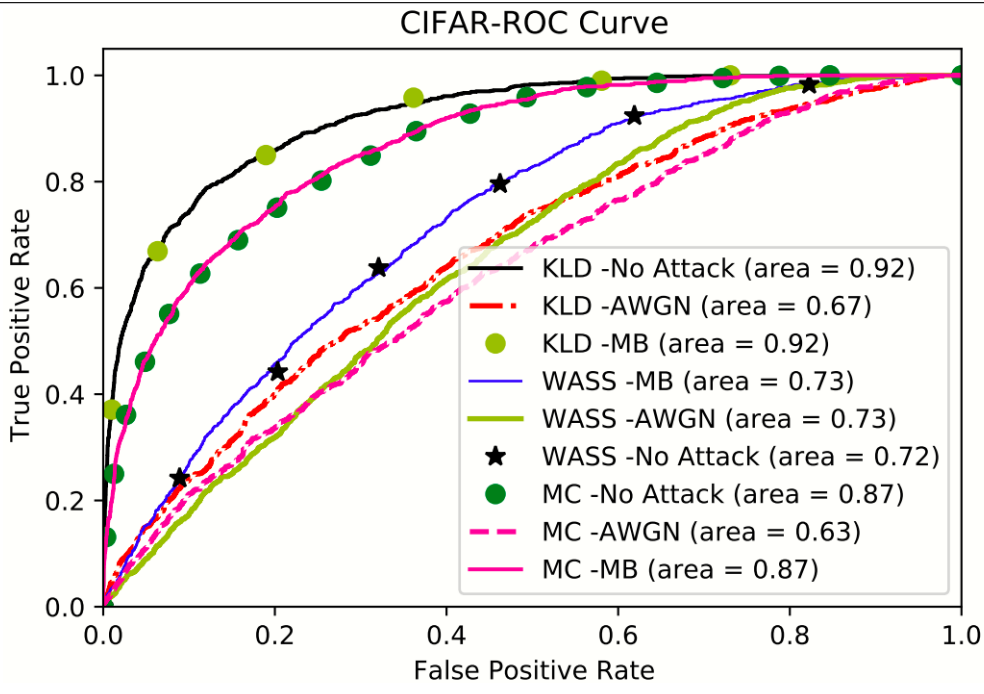
.

Fig. 4.9 Precision-recall for CNN-GP trained on the CIFAR-10 dataset. The plots are arranged according to their average precision. The plots are for each of the three losses: KLD, Wasserstein and maximum correntropy and for the three configurations: no attacks, Gaussian noise (shown as AWGN) and motion-blurring (shown as MB)



.

Fig. 4.10 ROC curve for CNN-GP trained on the CIFAR-10 dataset. The plots are arranged their ROC area. The plots are for each of the three losses: KLD, Wasserstein and maximum correntropy and for the three configurations: no attacks, Gaussian noise (shown as AWGN) and motion-blurring (shown as MB)

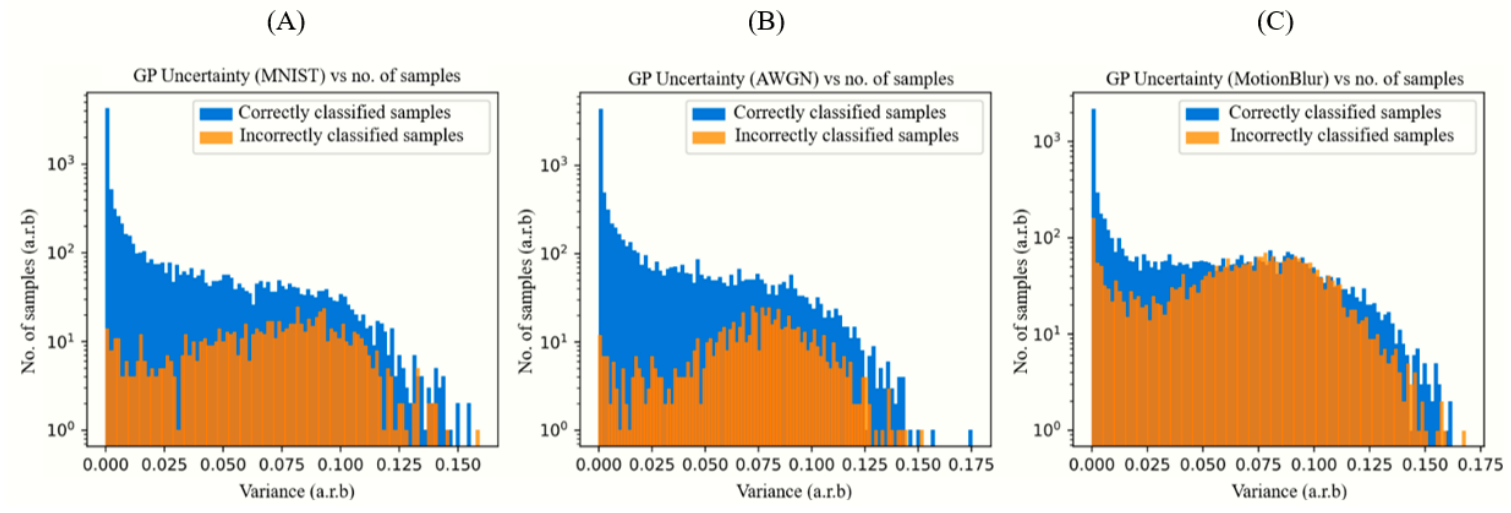


.

4.3.2 Uncertainty Analysis

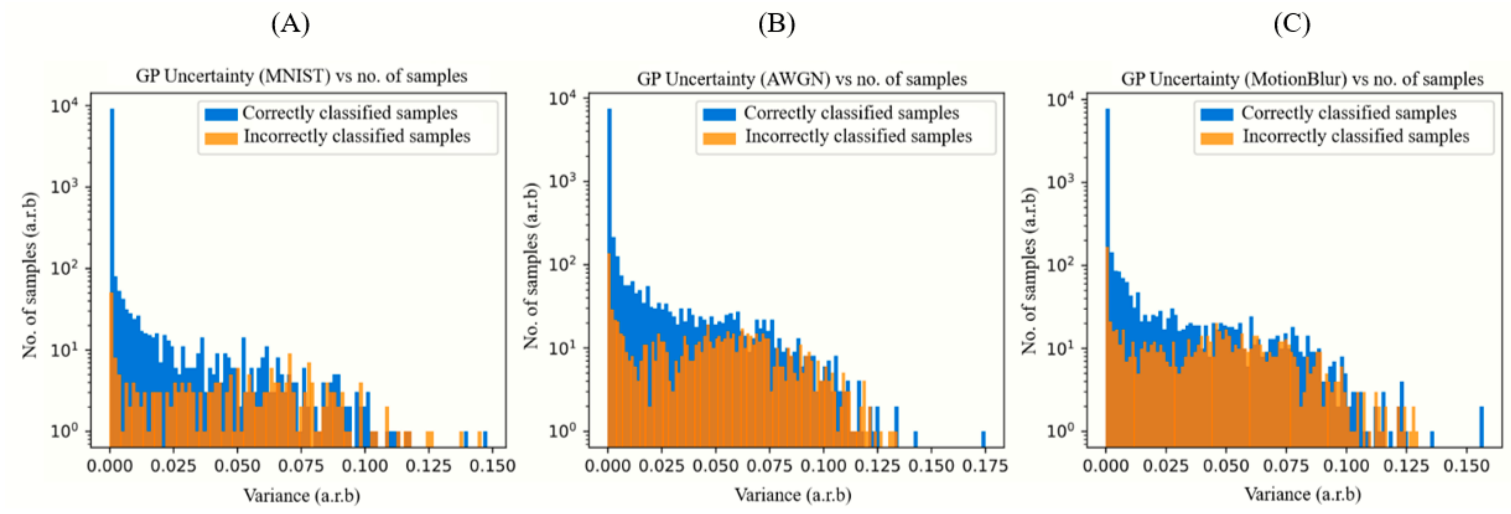
The following experiment aims to further support the performance evaluation results in the previous subsection by investigating the individual uncertainty values of the different experiment settings adopted in Table 4.2. The purpose of this experiment is to justify the improved performance with the reduced amount of uncertainty on Gaussian noise and motion-blurring. These experiments consider the MNIST datasets only. The mean output of the post softmax samples from the GP classifier and the variance is plotted as bar graphs. Given the predictions, for each time the label is correct, the appropriate variance is computed from the likelihood. Blue bars represent the variance of correct samples and orange for the incorrect. This is carried out for each of the samples in the test set (10000 MNIST images). The proposed model is tested with three examples. One on clean MNIST images (column 1), the other two on MNIST corrupted by AWGN (column 2) and MB (column 3). This is true for all the Figures 4.11, 4.12, 4.13 and 4.14. The instruction in which these graphs can be read is that the more the number of blue bars, the more accurate the model is, the lower the height of the blue or orange bars, the more reliable the model is. If the predictions have higher orange bars than blue, it is more susceptible to Gaussian noise and motion blurring. The term ‘a.r.b’ stand for arbitrary units.

Fig. 4.11 The uncertainty output bar graphs for the CNN-GP model with secondary loss type set to no regularisation setting. The Figures A, B and C correspond to the three attack settings: no attack, Gaussian noise and motion blurring. The blue bars represent variance on correctly classified samples, the orange for the incorrect ones



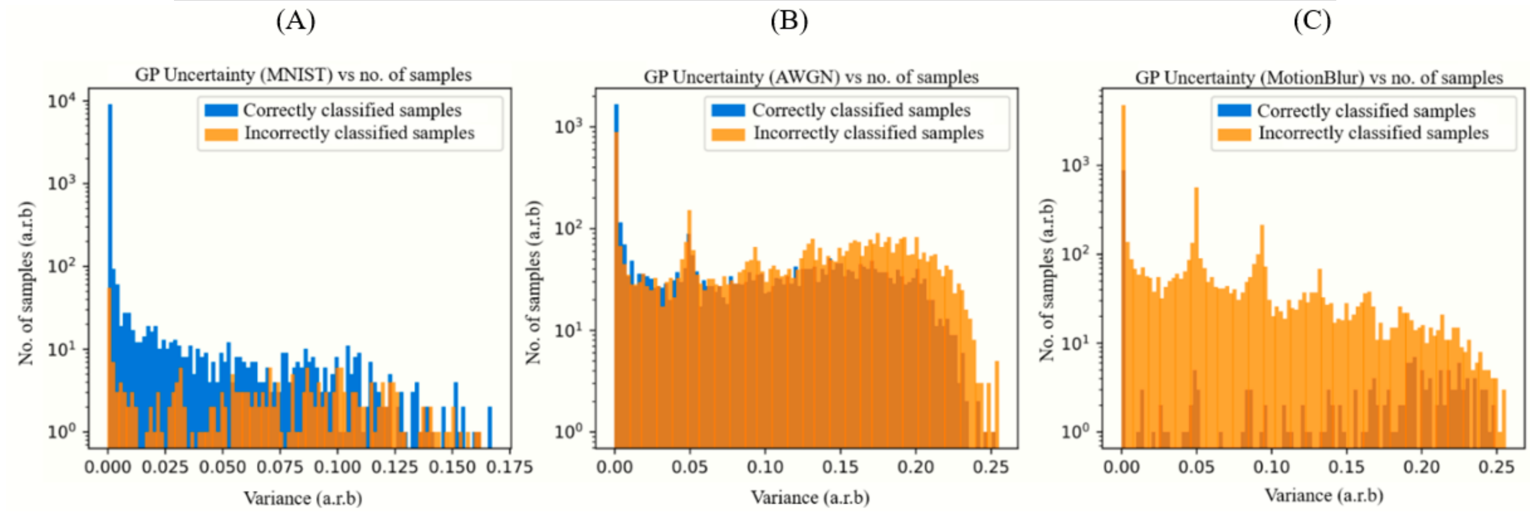
.

Fig. 4.12 The uncertainty output bar graphs for the CNN-GP model with secondary loss type set to KLD setting. The Figures A, B and C correspond to the three attack settings: no attack, Gaussian noise and motion blurring. The blue bars represent variance on correctly classified samples, the orange for the incorrect ones



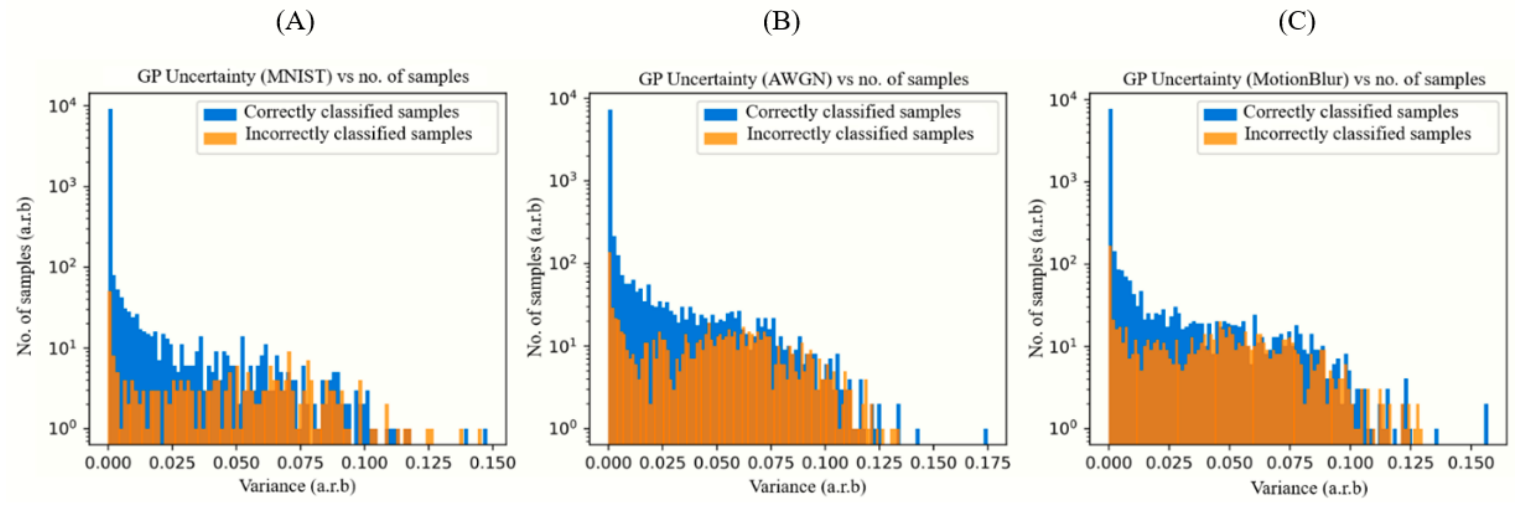
.

Fig. 4.13 The uncertainty output bar graphs for the CNN-GP model with secondary loss type set to WASS setting. The Figures A, B and C correspond to the three attack settings: no attack, Gaussian noise and motion blurring. The blue bars represent variance on correctly classified samples, the orange for the incorrect ones



.

Fig. 4.14 The uncertainty output bar graphs for the CNN-GP model with secondary loss type set to MC setting. The Figures A, B and C correspond to the three attack settings: no attack, Gaussian noise and motion blurring. The blue bars represent variance on correctly classified samples, the orange for the incorrect ones



.

4.3.3 Sensitivity of Variance to Attack Strength

This experiment aims to test the sensitivity of the variance information from the GP classifier of the CNN-GP model to both Gaussian noise and FGSM attack. This is performed on the MNIST dataset only. The plots for the case of Gaussian noise are shown in Figure 4.15 and Figure 4.16. The plots for the FGSM attacks are shown in Figure 4.17.

Considering the case of the Gaussian noise, the testing input mini-batch X_{TEST} are perturbed and fed to the GP classifier. Then, using the post-softmax samples, the variance information of the samples is plotted. The experiment is then repeated for the Gaussian noise attacks with varying standard deviation σ_{AWGN} from 0.0 to 2.0. The inputs are perturbed as shown in equation (4.1)

$$X^*_{TEST} = X_{TEST} \ . \ \sigma_{\text{AWGN}} + \mu_{\text{AWGN}}. \tag{4.1}$$

Here, σ_{AWGN} and μ_{AWGN} represent the standard deviation and the mean of the Gaussian noise respectively and X^*_{TEST} is the perturbed test mini-batch. Then, in a separate experiment, the system is attacked with the FGSM attacks. The FGSM attacks are made by using the gradients of only the CNN component of the CNN-GP model. The negative log of the output vector of size 16x128 (as shown in Figure 3.3) from the CNN feature extractor, i.e $\hat{y}_{CNN}$, is then computed. The gradients of this w.r.t. input test sample are then used to produce the perturbed data. The value of this is clamped between the range $[0,1]$. This is also shown in equation (4.2)

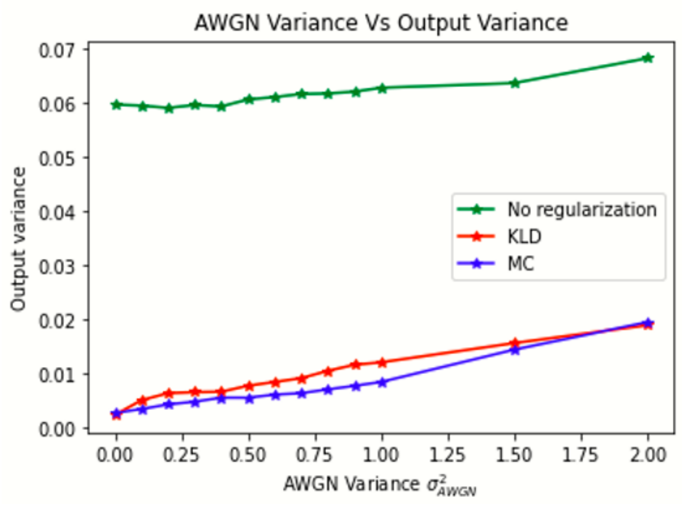
$$X^*_{TEST} = X_{TEST} + \varepsilon \ . \ sign\big(-\nabla_{X_{TEST}} \ log \ (\hat{y}_{CNN})\big). \tag{4.2}$$

Here, the term $-\nabla_{X_{TEST}} log \ (\hat{y}_{CNN})$ represents the gradients for computing the negative log-likelihood loss from the CNN-GP output based on the input test mini-batch X_{TEST}. Specifically, the nabla term $\nabla_{X_{TEST}}(y)$ is shorthand for $\frac{\partial y}{\partial X_{TEST}}$. In FGSM, computing the gradients of the output from the CNN feature extractor with respect to the mini-batch images through a sign function $sign(.)$, allows one to generate a new mini-batch of images X_{TEST} that is imperceptible to the human eye. However, can easily mislead the system's representation. Additionally, the term ε denotes the strength of the FGSM attack. In this experiment, this variable is changed increased within the range 0.0 and 3.0 in steps of 0.5. This is plotted in Figure 4.17.

The highlighted region in Figure 4.17 denotes the vital change of state in the system that can alert the system of the attack. This serves as a region where a high variance can lead to early detection of the attack before its intensity builds over time. Beyond this region, any change in variance would not be beneficial for a safety-critical system.

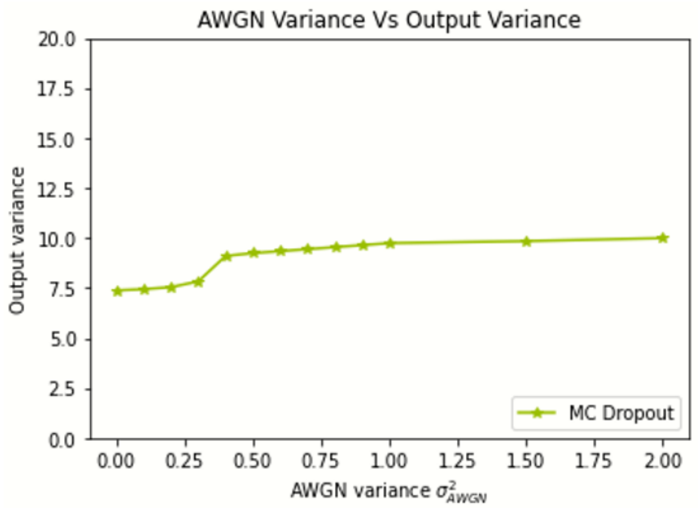
The proposed CNN-GP approach is also compared with the standard MC dropout method [41] and results are presented in Figure 4.16. The MC dropout results are obtained by isolating the pretrained CNN feature extractor and running forward passes 100 times. From this, the variance is computed and later averaged across the samples.

Fig. 4.15 Output variance computed from the GP classifier compared with the strength additive white Gaussian noise attack for three cases: training CNN-GP with no regularisation, training with KLD loss function and MC as regularisers



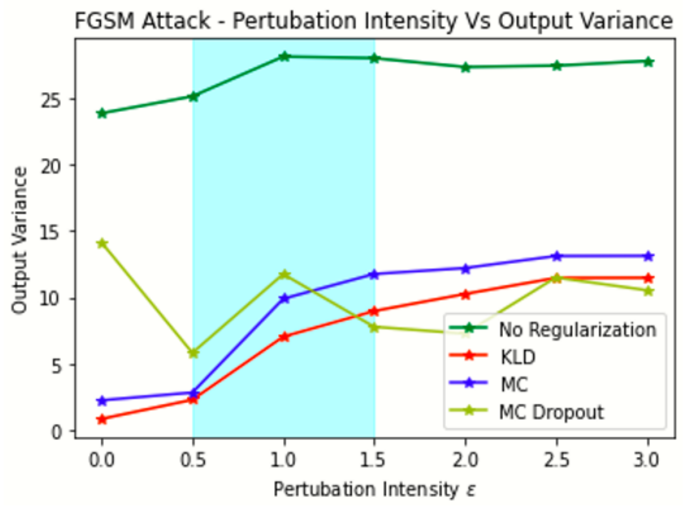
.

Fig. 4.16 Output variance computed from the GP classifier using MC dropout compared with the strength of additive white Gaussian noise attack



.

Fig. 4.17 Output variance computed from the GP classifier compared with the strength of white-box FGSM attack for four cases: training CNN-GP with no regularisation, training with KLD loss function and MC as regularisers, using MC dropout for uncertainty output



.

4.3.4 Computational Time

The purpose of this experiment is to compare the computational run times of CNN-GP with MC dropout methods [41]. This experiment is important since uncertainty aware learning applications in situational awareness and decision making require fast computation of uncertainty. In such applications, rational decisions made under fractions of seconds can have a substantial impact on the task. In some applications, such as medical robotics, decisions can result in life or death. Furthermore, the experiment compares the computational time with the MC dropout method. Comparison with MC dropout methods is important since MC dropout are applied frequently in standard DNN architectures [41], [78], [30] in order to account for uncertainty in results. To recap, MC dropout methods involve continuous sampling from several runs. The mean of the samples is taken as the prediction and the variance as uncertainty.

The experiment is carried out using the same architecture proposed in this chapter, i.e. the CNN-GP and then with the CNN component only with Monte Carlo sampling. Both models are made to output variance information on simple MNIST input images. The sampling rate for the MC dropout method is set to 100. The respective run-time for each is then computed on the University of Sheffield provided GPU cluster (NVIDIA K80). The experimentation process is carried out in the following way.

Firstly, the testing image set is divided into separate mini-batches. The initial timer based on the CPU clock is initialised to zero. Then, the test batches are forward propagated through both the CNN and GP components of the proposed algorithm and the timer is updated on each step. This is continued until the last batch-test from the test set is forward propagated. The results are averaged and the testing time is measured in minutes. The results are tabulated in Table 4.3.

Secondly, the CNN-GP architecture is taken again and the GP component is removed. The dropout values on both the dropout layers in the CNN component are set to 0.5 to prevent bias from either of the dropout layers. At test time, the MC dropout version of the CNN network is done so that the architecture is kept the same. In doing so, the test is kept fair between CNN-GP and UCB. In the next subsection, the convergence of each of the three similarity loss functions is considered.

Table 4.3 Run Time Analysis For Proposed Model CNN-GP and MC Dropout

Model Type	Run Time (min)
CNN-GP	1.18
MC dropout	7.27

4.3.5 Convergence of Similarity Losses Applied on CNN Only

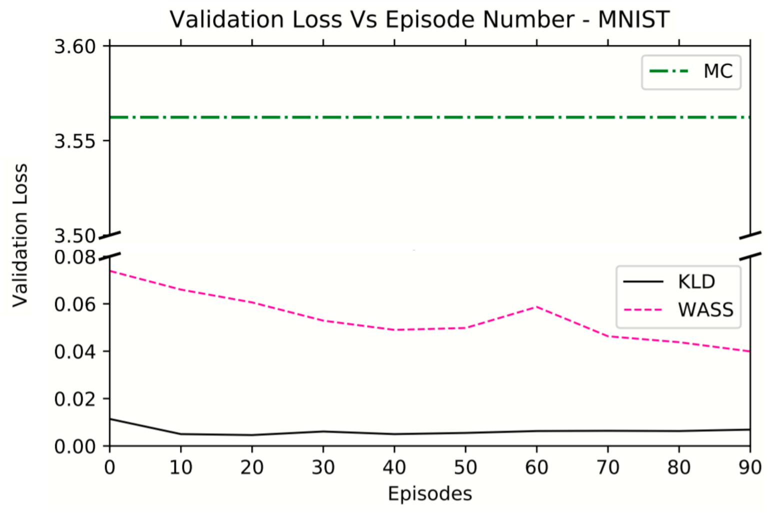
The purpose of this experiment is two-fold: a) one is to study the convergence of each of the three similarity functions and b) evaluate the performance of the CNN only component of CNN-GP trained on the three losses separately. The GP component is removed since it is only trainable through maximum likelihood and not the similarity loss functions.

In this experiment, the three similarity loss functions: KLD, Wasserstein and maximum correntropy, are not used as regularizers, as in previous experiments, but are now used for training on maximising classification performance on each of the four datasets: MNIST, Fashion-MNIST, CIFAR-10 and CIFAR-100. It is important to note that the Wasserstein distance used in this experiment is the entropy regularised version labelled WASS in previous experiments, instead of the full Wasserstein distance, labelled WASS-FIRST in previous experiments.

This experiment is important since it investigates which of the loss function is more suitable for stable training in CNN-GP for the classification task. Additionally, it is important to understand how these loss functions behave as the classification task becomes more difficult, for example classifying from MNIST to Fashion-MNIST or CIFAR-10.

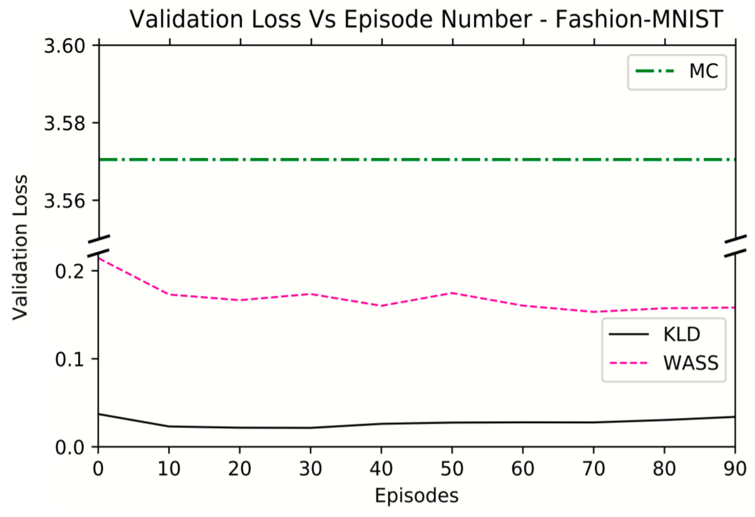
All of the experiments are carried out on the CNN only component of the proposed framework CNN-GP (see Section 3.2.2). In this case, the GP layer is removed and a softmax layer is added for converting the predicted feature vector into a vector of probabilities. This is considered for all three loss functions except the Wasserstein distance where instead a hyperbolic tangent activation is used.

Fig. 4.18 Validation losses observed during training CNN on MNIST with the three similarity losses for 100 episodes.



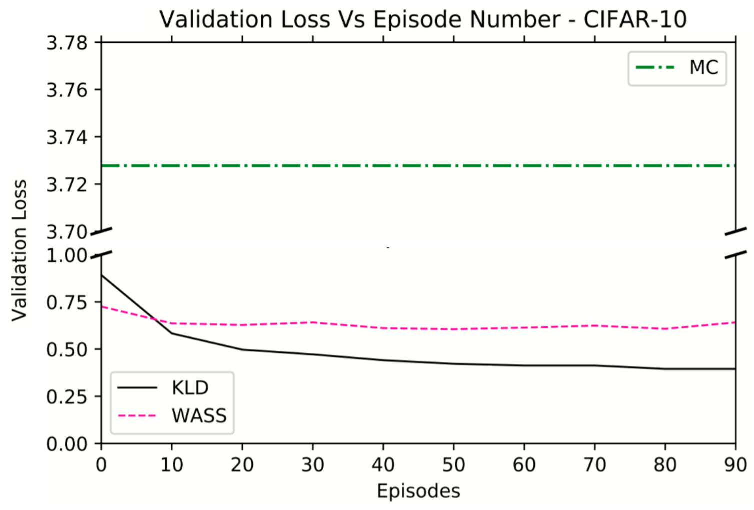
.

Fig. 4.19 Validation losses observed during training CNN on Fashion-MNIST with the three similarity losses for 100 episodes.



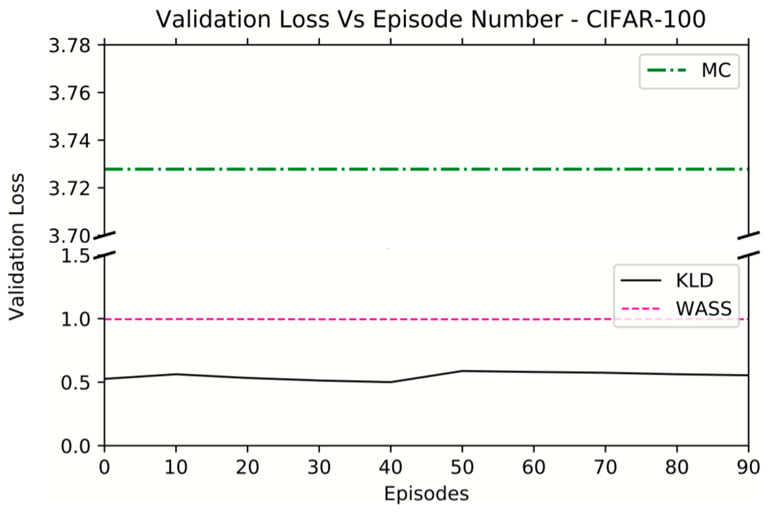
.

Fig. 4.20 Validation losses observed during training CNN on CIFAR-10 with the three similarity losses for 100 episodes



.

Fig. 4.21 Validation losses observed during training CNN on CIFAR-100 with the three similarity losses for 100 episodes



.

Each of the experiments on the three datasets is carried out separately. Within each of the experiments, the input training set is divided into mini-batches. These are then forward propagated through the CNN-GP to obtain the predictive vector of probabilities. The classified predictive outputs are compared with the labels using either of the loss functions

Table 4.4 Test Accuracy of the CNN Component Trained on KLD, WASS and MC on MNIST, Fashion-MNIST, CIFAR-10 and CIFAR-100.

Dataset Type	KLD	WASS	MC
MNIST [90]	98.75	96.48	8.92
Fashion-MNIST [190]	90.69	84.92	10.01
CIFAR-10 [84]	73.38	27.00	5.84
CIFAR-100 [84]	28.57	1.00	1.08

separately. Both the loss value and the accuracy are averaged for all mini-batches and recorded per episode. The experiments are carried out for a total of 100 episodes. The graphs in Figures 4.18 - 4.21 show the change of loss function w.r.t episodes. The test accuracy based on each of the loss functions for all datasets is tabulated in Table 4.4.

4.4 Discussion for Robust Learning

This section deals with the discussion on the results obtained on the robust learning methods in the thesis. The section is organised according to the results obtained in the previous sections. First, the discussion focuses on results from Table 4.2, then the focus shifts towards the precision-recall and ROC curves in Figures 4.5 - 4.10, then moving to the uncertainty analysis charts from Figures 4.11 - 4.14. Then, the results on sensitivity to adversarial and noisy attacks are discussed, these include the Figure 4.15, 4.16 and 4.17. This is followed by the discussion on the computational time taken for the CNN-GP and the MC dropout method in Table 4.3 and the results for the convergence of KLD, Wasserstein and maximum correntropy for the CNN component only in the Figures 4.18 - 4.21 and Table 4.4.

4.4.1 Discussion on Performance Validation

First to consider are the performance validation results in Table 4.2. Here, it is seen that dealing with a noisy AWGN attack is more difficult than dealing with images with motion blurring. This is evident with all of the dataset types, i.e. MNIST, Fashion-MNIST, CIFAR-

10 and CIFAR-100. Regardless of the dataset, experiments with or without regularisation perform better on motion-blurred images than AWGN. For example, considering the no regularisation case on the MNIST dataset, the performance on blurred images is 65% as opposed to 51% on images attacked by AWGN.

A second observation to note is that with regularisation, regardless of the type, the performance of CNN-GP is better than without regularisation. For example, in the MNIST case, results with no regularisation with AWGN attack and blurred images, the accuracies are 51% and 65% respectively, and with regularisation, it is 89% and 72% for KLD, 77% and 70% for Wasserstein distance, 78% and 75% for maximum correntropy.

The best performance, however, is obtained when the regularisation for the Wasserstein distance in the full form is used. Here, for example for the MNIST dataset, the regularisation with the full Wasserstein distance achieves 85% and 88%, 54% and 77% on Fashion-MNIST, 26% and 40% for CIFAR-10, This is the highest accuracy recorded compared to other forms of regularisation. Compared to maximum correntropy, regularisation with the full Wasserstein distance is much better at dealing with AWGN. For example on the MNIST dataset, regularisation with MC achieves 78% while with the full Wasserstein distance it achieves 85%. Additionally, on Fashion-MNIST, regularisation with MC achieves 35% while with the full Wasserstein distance it achieves 45%.

Another observation from the performance evaluation in Table 4.2 is that the complexity of the dataset is more influential than the difficulty of the inputs. For simple datasets e.g. MNIST and Fashion-MNIST, CNN-GP achieves better performance than on difficult datasets e.g. CIFAR-10 and CIFAR-100. Consider for example in the case training with no regularisation, the performance drops from 88% on MNIST to 85% on Fashion-MNIST, and then from 67% on CIFAR-10 to 24% on CIFAR-100. This is without considering AWGN attacks and blurring of images.

4.4.2 Discussion on Precision-Recall and ROC Curves

To characterize the robustness of the approaches, the recall function is calculated. Precision is heavily affected by uncertainties and impacts the results of all methods. However, the approaches with the maximum correntropy and KLD maintain a good level of precision despite having poor recalls (e.g. in AWGN attacks for MC and KLD). Hence, it is possible to diagnose the recall aspect as a measure of sensitivity to the attack. Furthermore, the precision-recall and ROC curves explain and support the results from the performance evaluation in Table 4.2. Firstly, it can be seen that regardless of the dataset type configuration, the performance of CNN-GP on images attacked by AWGN is less than motion blurring. This can be seen in for example for the case of MNIST in Figure 4.5, the average precision for the maximum correntropy regularisation with AWGN attack is 0.86 and 0.85 for Wasserstein distance regularisation with AWGN attack. This is less compared to the average precision when either configuration in the case of motion-blurred images (1.00 and 0.93 respectively).

The second important observation, which also supports and explains the results in performance evaluation in Table 4.2 is that when the images are perturbed by motion blurring, the performance of CNN-GP when using regularisation with MC and KLD is similar in terms of precision and recall but superior to regularisation with Wasserstein distance. This is evident in for example in the case of MNIST and Fashion-MNIST dataset, the average precision of CNN-GP regularised with both maximum correntropy and KLD is 0.07 units more than regularisation with Wasserstein distance (see Figure 4.5) and 0.06 units more in the Fashion-MNIST case (see Figure 4.7).

Another point to note is that, in the case of images perturbed by AWGN, the performance of CNN-GP with Wasserstein distance based regularisation is similar, in some cases, and sometimes better than regularisation with MC and KLD. This can be seen for example in Figure 4.5 where the average precision for using Wasserstein distance based regularisation is on par with maximum correntropy (i.e. 0.85 and 0.86) but is 0.28 average precision units less than using KLD regularisation. However, in Figure 4.7, the average precision of CNN-GP regularised with Wasserstein distance is on par with KLD (i.e. 0.58) but is also 0.28 average precision units more than with maximum correntropy based regularisation. In the case of the CIFAR-10 dataset in Figure 4.9, the average precision of CNN-GP with Wasserstein based regularisation is 3.00 units more than KLD and maximum correntropy based regularisation.

Lastly, the precision-recall results also show that the complexity of the dataset is more influential than the strength and type of the attacks. This also supports the results for performance evaluation in Table 4.2 which demonstrate the same concept. Additionally, this is also supported in precision-recall and ROC curves for the case of CIFAR-10 dataset in Figure 4.9 and 4.10. Here, it is seen that AWGN attacks affect the performance of CNN-GP regardless of the regularisation type ($\approx$ 0.26 average precision units).

4.4.3 Discussion on Uncertainty and Sensitivity Analysis

The results for uncertainty analysis are presented in Figures 4.17 - 4.14 and the sensitivity analysis in Figures 4.15 - 4.17. These results do not emphasize performance as the results from Table 4.2 and precision-recall. However, these results focus on the sensitivity of the uncertainty (variance information) to the presence of attacks. Ideally, these results should highlight less number of samples in the high variance range (e.g. 0.10 - 0.25 in Figures 4.11 - 4.14). This would mean the network is confident in its predictions and is less likely to output erroneous predictions. Despite this, some of these values should be present so that the system can demonstrate that it can highlight when the attack is too severe to make any predictions at all. Some of these example cases are discussed next.

Firstly, focusing on uncertainty analysis in Figures 4.11 - 4.14, it is observed that CNN-GP shows sensitivity to noisy attacks and motion blurring. This is evident in all of the figures in uncertainty analysis results from Figure 4.11 - 4.14 where the number of incorrect samples increase (from 10 to 100) for all ranges of variances (0.00 - 0.25). This is true regardless of regularisation type. However, the lowest number of incorrect samples is observed in KLD and the maximum correntropy regularisation type. The issue is that the uncertainty results with the Wasserstein metric (in Figure 4.13) display samples classified with high uncertainty ($\approx$ 100 in the range 0.10 - 0.25), regardless of the attack. It performs rather poorly than expected, unable to cope with the attacks. This agrees with the hypothesis of [11] which claims that the Wasserstein metric yields biased gradients that have a higher chance of leading to a false local minimum than the KLD during optimization.

Then, considering the maximum correntropy and KLD based regularisation results, it is evident that using these losses results in high accuracies in motion blurring when compared with the Wasserstein metric results. The performances of the MC and KLD are similar, as evident in Figure 4.11 - 4.14 where uncertainty charts for both KLD and MC based regularisation have a greater number (≥ 100) of correct sample variance (blue) as compared to those for the Wasserstein metric (in Figure 4.13). For MC based regularisation, this is expected since this type of loss is ideal for robust algorithm design. This is further supported in Figure 4.5 - 4.10 where the precision-recall for both KLD and MC for motion blurring (MB) remain the highest (solid blue, green and yellow lines) as the dataset complexity increases (MNIST to CIFAR-10).

One possible reason for the results can be that an approximated version of the Wasserstein metric is implemented. An approximate implementation is performed to avoid the complexity and intractability of computing the infimum of double integrals in the full version [125]. This is further supported by the precision-recall diagram for the Wasserstein metric for all attacks which shows that the precision for these methods slowly drop when the dataset complexity is increased (from MNIST to CIFAR-10). The downward shift of blue, black and yellow dashed lines in Figure 4.9 - 4.10 visualizes these drops.

Regarding the variance sensitivity to attack strength, we can see from Figure 4.15 - 4.17 that CNN-GP trained on the MC similarity loss is more responsive than both KLD as well as the no regularization configuration. This also demonstrates that the MC is suitable for robust algorithm design. The graphs show that both the MC and KLD functions, start with higher confidence in predictions (i.e. low variance) before the attack strength is increased when compared to the case without regularization. This confirms both our hypothesis and our results in Figure 4.11 - 4.14 that backpropagation in the CNN-GP framework reduces the impact of uncertainties and attacks on the classification results and characterize the model's confidence. For the MC dropout method, it is seen from both Figures 4.16 and 4.17 that this model is not representing the uncertainty estimates well when compared with the CNN model. Hence, it is less reliable for uncertainty quantification applications.

4.4.4 Discussion on Computational Run Time

The results for the computational time for a single forward pass of the CNN-GP architecture and the state-of-art MC dropout is presented in Table 4.3. One important observation is that for the equal batch size of 16 MNIST samples, the time taken for a single forward pass of CNN-GP is approximately 6x faster than a single forward pass of MC dropout. This is an important observation from the point of view of mission-critical systems that require fast computation of uncertainty. However, neither of the models output uncertainty faster than a minute and therefore are not ideal for mission-critical scenarios where decisions are made in the order of seconds. One example of such systems include robotic surgeon [167], [162].

4.4.5 Discussion on Convergence of Kullback-Leibler, Wasserstein Distance and Maximum Correntropy Loss for the Convolutional Network Component of CNN-GP

The convergence of the three aforementioned similarity losses on the CNN component of the CNN-GP is studied in Figures 4.18 - 4.21. The purpose of these experiments is to investigate how the three similarity losses are to behave if they were applied separately instead of being used as part of the regularisation of the maximum likelihood loss. Since in these experiments the GP component is removed, regularisation of maximum likelihood is avoided. Then, looking at the results, it can be observed that for the three datasets MNIST [90], Fashion-MNIST [190], CIFAR-10 and CIFAR-100 [84], both the loss functions KLD and Wasserstein distance converge while the maximum correntropy does not regardless of the dataset type. However, it is seen that even in the case of the difficult dataset (e.g. CIFAR-100 in Figure 4.21) none of the losses converge. This supports both the performance validation (see Table 4.2) and the precision-recall (see Figures 4.5, 4.7 and 4.9) which highlight that the complexity of the data influences the performance of CNN-GP more than the impact of noise and motion blurring, regardless of the type of regularisation used.

4.5 Results for Continual Learning

This section aims to introduce the results for the proposed solution CNN-GP applied to the continual learning scenario. Several different experiments are performed. Some focus on performance-based tasks, while others investigate the computational run time and some focus on robustness analysis in the presence of noisy images. Experiments are performed on datasets split-MNIST, permuted-MNIST, sequential CIFAR-10 and CIFAR-100. In terms of performance evaluation, both the setting deterministic and Bayesian setting are compared separately. Additionally, the proposed solution, in the Bayesian setting, is also compared with the state-of-art UCB method. These experiments also analyse the number of episodes required for convergence and analysis of computational run-time between the deterministic and the Bayesian setting. A further test on the robustness to noisy images is also carried out for both UCB and the CNN-GP in the Bayesian setting.

4.5.1 Accuracy Comparison Per Task - Deterministic vs Bayesian Setting

This experiment aims to evaluate the performance of the CNN-GP framework. The experiment is carried out for both split and permuted-MNIST datasets. Furthermore, it compares performance for both the two settings of the CNN-GP, the deterministic and the Bayesian. It also compares with a version that is trained on the entire MNIST dataset (i.e. no continual learning setting). The experiments are carried out by training the framework on each of the tasks for a total of 200 episodes. This is followed by the validation step. The validation is carried out on each of the tasks a total of 100 times and this is done for each episode. Later, these values are averaged and stored.

The averaged accuracies are plotted for the no continual learning setting in Figures 4.22-4.26 with heading (A), the deterministic setting in Figures 4.22-4.26 with heading (B) and for the Bayesian setting in 4.22-4.26 with heading (C). The experiments on split-MNIST are plotted in Figures 4.22-4.25 and the experiments on permuted-MNIST are shown in

Figure 4.26. The x-axis represents the task number ranging from 0-5 for the split-MNIST dataset and 1-10 for permuted-MNIST dataset.

Fig. 4.22 Comparison of accuracy for no continual learning (A), deterministic (B) and the Bayesian setting (C) for the proposed framework CNN-GP validated on split-MNIST. The results are shown for the network trained on task 23 and validated on tasks 01 and 23.

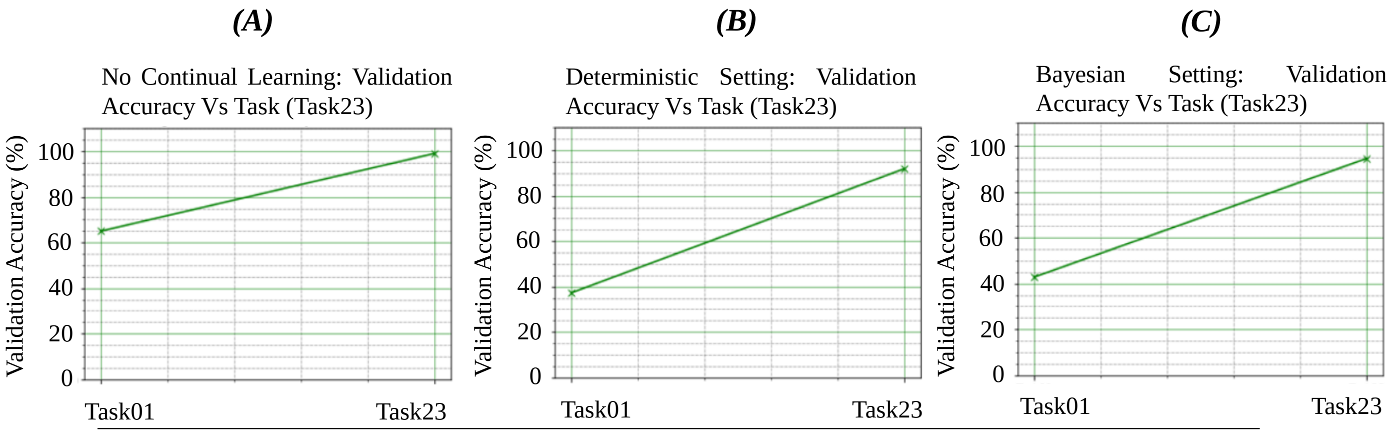


Fig. 4.23 Comparison of accuracy for no continual learning (A), deterministic (B) and the Bayesian setting (C) for the proposed framework CNN-GP validated on split-MNIST. The results are shown for the network trained on task 45 and validated on tasks 01, 23 and 45.

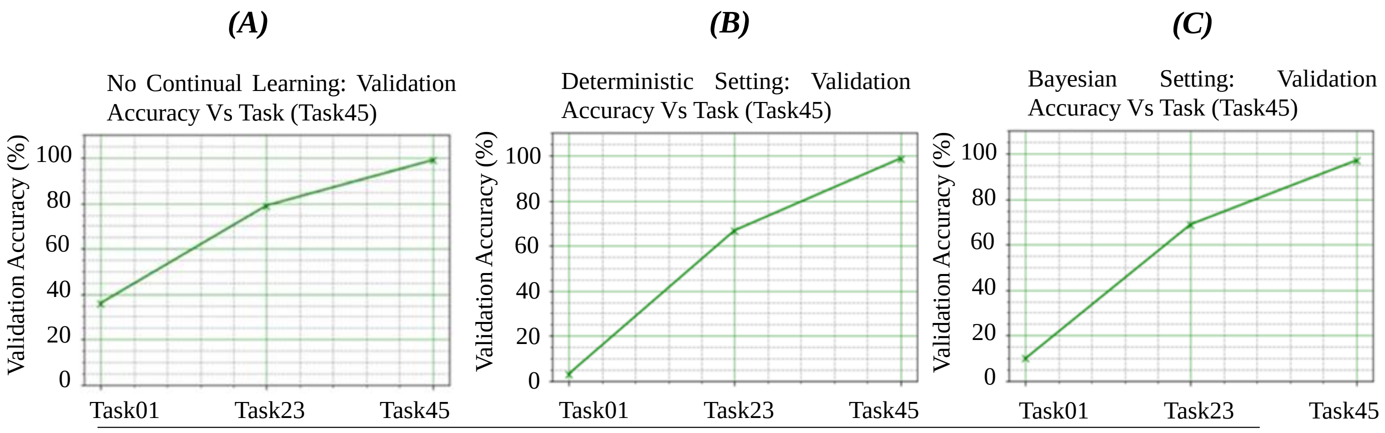


Fig. 4.24 Comparison of accuracy for no continual learning (A), deterministic (B) and the Bayesian setting (C) of the proposed framework CNN-GP validated on split-MNIST. The results are shown for the network trained on task 67 and validated on tasks 01, 23, 45 and 67.

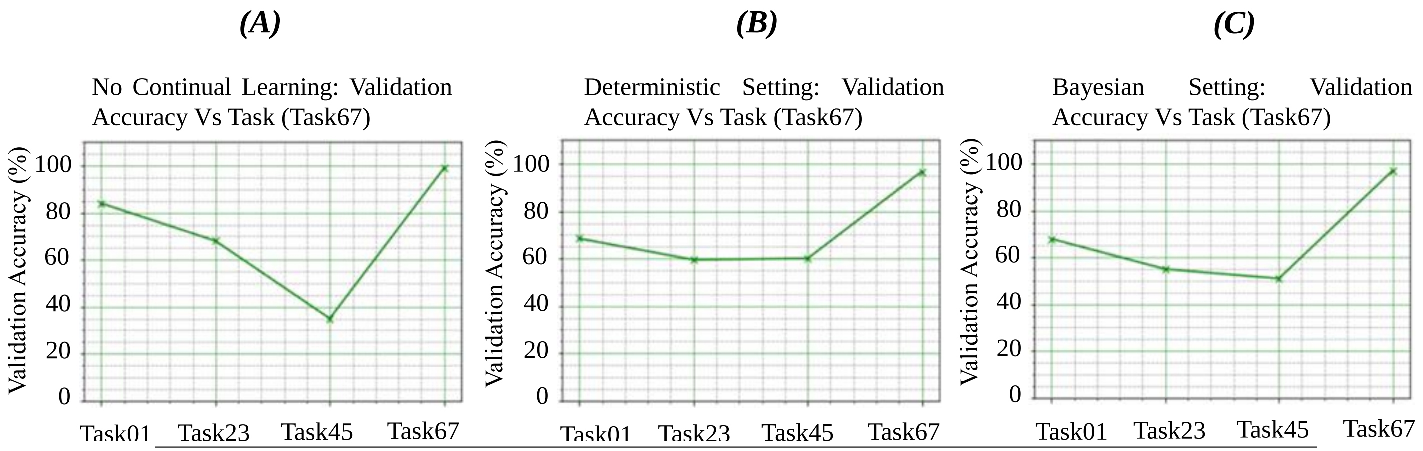


Fig. 4.25 Comparison of accuracy for no continual learning (A), deterministic (B) and the Bayesian setting (C) of the proposed framework CNN-GP validated on split-MNIST. The results are shown for the network trained on task 89 and validated on tasks 01, 23, 45, 67 and 89.

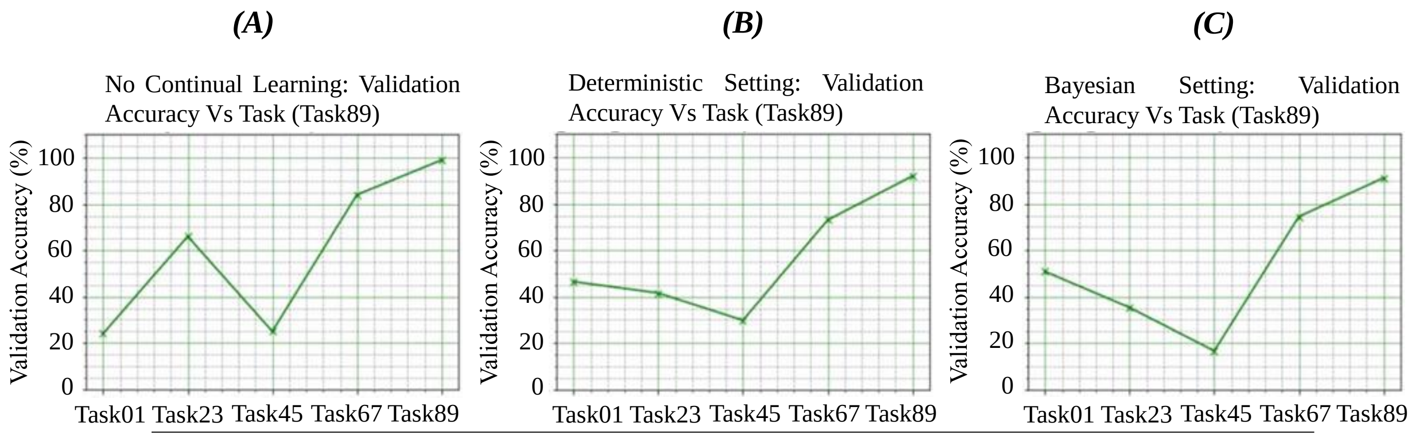
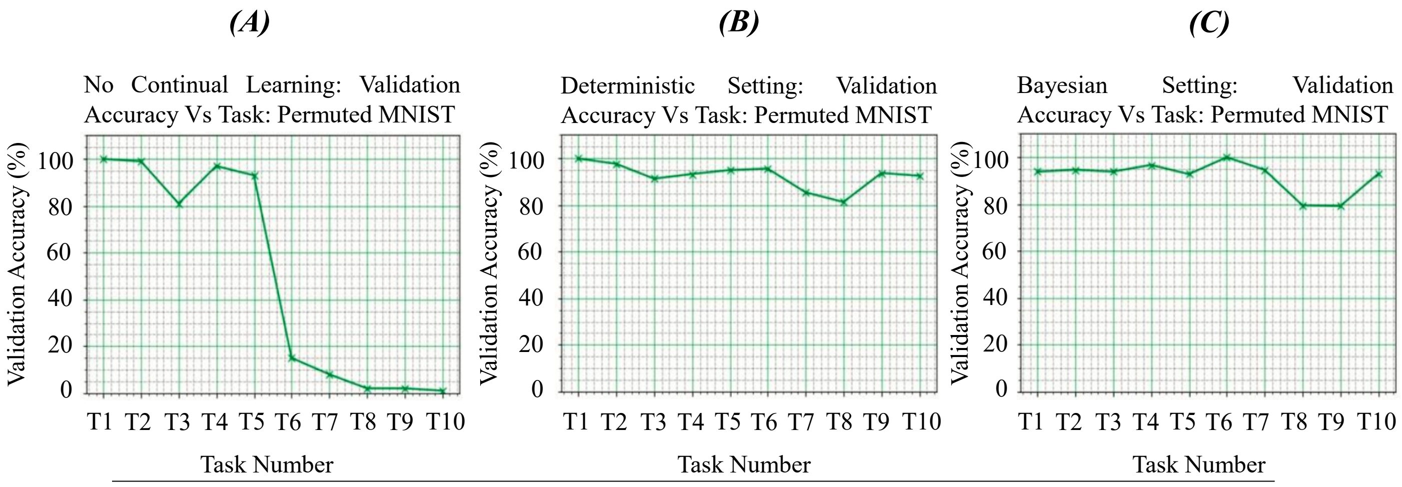


Fig. 4.26 Comparison of accuracy for no continual learning (A), deterministic (B) and the Bayesian setting (C) of the proposed framework CNN-GP validated on permuted-MNIST. The results shown validation accuracy on each of the ten tasks. Here, 'T' refers to task on the x-axis



4.5.2 Comparison of Episodes Required for Convergence of Training Loss on Permuted-MNIST - Deterministic Vs Bayesian

The purpose of this experiment is to compare the number of episodes required for both model configurations to converge on permuted-MNIST. This is carried out by training each task repeatedly 80 times. Then, the number of episodes required for each is averaged and tabulated for both the configurations in Table 4.5. Comparison for split-MNIST was not required since the results were closely similar ($\pm$ 2 episodes).

Table 4.5 Comparison of Episodes Required for Training on Permuted-MNIST for Both Deterministic and Bayesian Setting

Tasks	**Deterministic**	**Bayesian**
0	26	45
1	33	8
2	112	22
5	74	9
6	40	10
7	54	14
8	95	29
9	18	4

Table 4.6 Comparison for Average Time Taken for Running a Single Episode for Both Deterministic and Bayesian Setting

	Deterministic	**Bayesian**
Average time for a single episode (mins)	5.02	11.65

4.5.3 Comparison of Computational Time Taken on Single Episode on Permuted-MNIST - Deterministic Vs Bayesian

The focus of this experiment is to compare the computational time taken to train on a single episode of the permuted-MNIST testing set. Once again both the deterministic and Bayesian setting is compared. The testing time is measured in minutes and averaged across 50 repeats for each of the 10 tasks. The results are presented in Table 4.6.

Table 4.7 Per-Task Accuracy on Split-MNIST for Both UCB and Bayesian Setting

Model Type	**Task 0**	**Task 1**	**Task 2**	**Task 3**	**Task 4**
UCB	99.52	93.23	93.23	95.59	97.87
BCNNGP	96.17	95.22	94.69	97.36	92.86

Table 4.8 Per-Task Accuracy on Permuted-MNIST for Both UCB and Bayesian Setting

Model Type	**Task 0**	**Task 1**	**Task 2**	**Task 3**	**Task 4**	**Task 5**	**Task 6**	**Task 7**	**Task 8**	**Task 9**
UCB	76.29	60.91	61.30	61.55	60.60	60.41	60.25	61.32	60.60	61.30
BCNNGP	40.78	43.65	34.25	35.77	37.20	34.89	32.31	33.74	36.50	39.11
BCNNGP-256	86.33	83.08	67.89	67.55	89.23	68.42	63.36	84.35	85.41	87.62

Table 4.9 Per-SNR Accuracy on Split-MNIST for Both UCB and Bayesian Setting

Signal-to-Noise Ratio									
Model Type	20	12	6	2	0	-2	-4	-5	-6
UCB	99.67	99.50	99.39	98.32	97.28	95.50	93.35	91.28	88.70
BCNNGP	94.95	93.35	95.59	95.22	96.40	96.67	95.91	91.33	97.91

4.5.4 Robustness Analysis - UCB Vs Bayesian Setting

The objective of this experiment is two-fold: a) to compare per-class accuracy on split-MNIST, permuted-MNIST and CIFAR-10 b) test robustness against Gaussian noise attacks, all for cases of Bayesian setting and UCB. The experiments are carried out by first training UCB on both split, permuted-MNIST and CIFAR-10 using the hyperparameters similar to those provided in [32] and also in Algorithm 3.3.6. In all experiments where CNN-GP is compared with UCB, the CNN component of CNN-GP is reduced to two fully connected layers. This is done since UCB architecture also uses only two fully connected layers. Furthermore, in the per-task accuracy experiments on permuted-MNIST, two versions of CNN-GP are tested. One where the input features are set to 128 and grid size to 16, termed BCNNGP. The other version uses 256 input features and a grid size of 64, termed BCNNGP-256.

Table 4.10 Per-SNR Accuracy on Permuted-MNIST for Both UCB and Bayesian Setting

Signal-to-Noise Ratio									
Model Type	20	12	6	2	0	-2	-4	-5	-6
UCB	87.90	86.97	83.07	75.06	63.63	52.80	43.60	37.26	31.77
BCNNGP	81.00	73.16	47.32	35.56	26.30	21.85	18.64	14.41	13.15

Once trained, the system is perturbed with Gaussian noise. This noise is added by varying the value of the standard deviation from 0-2 in steps of 0.25. The images are perturbed with the noise and the respective SNR values are calculated using equation (3.17) for each batch per task. This is averaged across all batches and tasks to obtain the final SNR value.

Table 4.11 Per-Task Accuracy on CIFAR-10 for Both UCB and Bayesian Setting

Model Type	**Task 0**	**Task 1**	**Task 2**	**Task 3**	**Task 4**
UCB	80.63	86.41	69.92	70.66	70.68
BCNNGP	61.20	93.55	67.24	61.29	67.14

Table 4.12 Per-Task Accuracy on CIFAR-100 for Both UCB and Bayesian Setting

Model Type	**Task 0**	**Task 1**	**Task 2**	**Task 3**	**Task 4**
UCB	44.51	38.30	39.61	38.02	37.61
BCNNGP	45.87	54.44	11.29	13.71	8.14

As the results are obtained, the respective per-task and per-SNR accuracies are calculated. Per-task is obtained by averaging the accuracies across all SNR values for each task. Per-SNR accuracies are calculated by averaging across all tasks for each SNR value. The results are presented for split-MNIST in Tables 4.7 and 4.9, for permuted-MNIST in Table 4.8 and for CIFAR-10 and CIFAR-100 in Table 4.11 and 4.12. The next discusses the results.

4.6 Discussion on Experiments of Continual Learning

The results show that the performance of the deterministic and Bayesian settings is similar. The difference is approximately 5% in terms of accuracy. However, on the majority of the tasks, the deterministic setting takes more iterations to converge than the Bayesian setting. However, the computational time for the deterministic setting (5.02mins) is nearly half that of the Bayesian setting (11.65mins). When compared with the UCB approach, the CNN-GP can perform better than UCB in some tasks (Task 1 and 3) in split-MNIST. However, it is unable to perform in permuted-MNIST. In terms of robustness, for high SNR values (0-20) UCB is more robust to Gaussian attacks than CNN-GP, for low SNR values (0 to -6) CNN-GP is more robust. However, UCB is more robust than CNN-GP in the permuted-MNIST dataset. Therefore, the main takeaway message of this study is that for relatively simple datasets and few tasks, a single-classifier can perform as good as a multi-classifier for continual learning, while being robust to noise. However, if the dataset is difficult to train and has more than, e.g. ten tasks, then a multi-classifier setting is better suited. For the interest of the readers, this study can be improved further with multiple GP classifiers or by testing with other forms of learning rate update methods such as momentum-based learning rate schedules [146]. The contributions of these experiments are as follows. For the first time, a study comparing the performance of a single-classifier approach with the state-of-art multi-classifier approach UCB [32]. Then, it is shown that the single-classifier approach achieves performance on par with the multi-classifier approach in some tasks (e.g. split-MNIST). However, it fails to achieve higher performance on complex tasks (e.g. permuted-MNIST).

4.7 Summary

This chapter detailed the experiments presented in the three areas tackled in this thesis: semantic segmentation, robust and continual learning. Firstly, the chapter focused on the experiments on the proposed AdvSegNet architecture for semantic segmentation application. The experiments were performed for both CamVid [16] and Sun RGB-D [169] datasets. Then, the performance of AdvSegNet was compared with state-of-art architectures as well as qualitative results and uncertainty maps were used to explain the impact of uncertainties on the performance of AdvSegNet.

The goal of the experiments was to develop an add-on architecture that would aim to improve the performance of the popular segmentation architecture SegNet. The purpose was to explore methods to use uncertainty to recover performance lost from a physical attack on the sensors. Additional objectives included achieving performance close to or better than the state-of-art. Based on the experiments, only some of these objectives were achieved. Firstly, it was shown that using AdvSegNet improved performance on classes that take a large area as opposed to those that take up smaller space. For example, a sky category is larger than a column-pole category in terms of size. Secondly, using AdvSegNet does not improve performance on large and challenging datasets such as Sun RGB-D and Pascal VOC, however, AdvSegNet manages to achieve performance close to the original Bayesian SegNet despite having more than half the receptive field size of the Bayesian SegNet. Thirdly, the limitations of AdvSegNet included the slow speed of uncertainty inference which is not ideal for mission-critical applications.

The chapter then shifted the focus to detailing experiments for robust learning. In particular, CNN-GP architecture was evaluated on datasets MNIST [90], Fashion-MNIST [190], CIFAR-10 and CIFAR-100 [84]. Experiments were performed for performance evaluation, sensitivity to noisy and motion blurred images, sensitivity to white-box attack FGSM and computational run-time analysis. This was followed by the discussion section which showed how using KLD and MC regularisation helps in improving sensitivity to FGSM, noisy and motion-blurred images.

The aims and objectives of the experiments on robust learning were to: a) prove that regularisation can help improve performance of a DNN classifier and the sensitivity to noisy, blurred images and adversarial attacks, b) to investigate whether generating adversarial images using the learned feature space of the DNN classifier can be used to improve both robustness and sensitivity of the system too noisy, blurred images and adversarial attacks, and finally c) to prove that using regularisation along with feature space based adversarial learning can be used to drop uncertainty.

These objectives were accomplished as follows. Firstly, it was shown that regularisation from both KLD and MC improves the sensitivity to noisy, blurred images and white-box attack FGSM. Secondly, in terms of accuracy, the results from performance evaluation showed that using the proposed training methods and regularisation achieves better accuracies than without. However, this is partially accomplished since it works on simpler datasets such as MNIST, Fashion-MNIST, but not on complex datasets such as CIFAR-10 and CIFAR-100. Thirdly, the uncertainty analysis results do show that using the proposed loss functions and training does reduce uncertainty which supports the performance evaluation results.

Some of the limitations of the CNN-GP include, for example, it was shown that even in no regularisation scenario the sensitivity to noisy images works as good as with CNN-GP trained with the proposed loss function and training.

Additionally, some of the objectives were partially accomplished. For example, the results for performance evaluation could have been improved with a better CNN backbone architecture. Finally, it was not explained why simply using the MC loss function did not lead to convergence of the loss on the CNN component of the CNN-GP.

Finally, the focus of the chapter shifted to providing experiments in the area of continual learning. The experiments involved investigating the performance of a single-classifier (CNN-GP) in continual learning compared to the state-of-the-art multi-classifier method UCB. The learning rate updates were performed by scaling according to the post softmax sample variance obtained after GP output. Experiments were carried out on both split and permuted-MNSIT. Two settings were adopted: deterministic and Bayesian.

The purpose of the experiments was to compare the performance of a single-classifier approach with a multi-classifier, state-of-art approach UCB. The results showed that the performance of both settings is on par. However, the Bayesian setting took fewer amount of episodes to converge. On split-MNIST, the performance of CNN-GP and UCB was on par but UCB outperformed CNN-GP in permuted-MNIST. However, some of the objectives were partially accomplished. For example, it could not be explained why for some tasks such as Task 1 for split CIFAR-10, the BCNNGP performed better than UCB. This was the same observation that existed for split-MNIST as well. Furthermore, it could also have been proven that BCNNGP may have performed better than UCB if larger grid-size were to be used. Lastly, some of the limitations of the work were that the Bayesian setting was slower than deterministic in terms of episodes required to converge. The methods only worked for simple datasets e.g. split-MNIST, with limited performance on complex datasets such as split CIFAR-10.

Chapter 5

Conclusion and Future Works

This chapter presents a summary of the main contributions of the thesis. It outlines new directions that can be considered for future research. It also highlights limitations in the proposed methods in the thesis. The chapter is divided into three sections: conclusions and limitations, application and future works.

The first section provides a summary of the main contributions of the thesis, reviewing the contributions of each of the chapters. Then, for each method proposed in the chapters, the limitations of the methods are provided. This outlines possible flaws in the proposed methods, including areas or conditions where the methods can or cannot be applied as well as highlighting how the methods can be improved upon in the future works section.

5.1 Findings and Limitations

This thesis aims to provide solutions in the area of uncertainty aware deep learning. The applications cover the four key areas in deep learning and Bayesian deep learning: semantic segmentation, generator adversarial learning, robust learning and continual learning. A brief introduction of the field of uncertainty aware deep learning is discussed in Chapter 1. Then, a further detailed review of both deep learning and Bayesian methods, as well as Gaussian processes, is provided in Chapter 1. This chapter mainly deals with the background information, it gives a generic overview of deep learning, Bayesian methods, adversarial attacks and Gaussian processes. It is later further explored in later chapters how these fields can be applied to segmentation, robust learning and continual learning.

Deep Learning-Based Semantic Segmentation

The application of the proposed methods of this thesis in the area of semantic segmentation is discussed in Chapter 3. In particular, the main contribution from this chapter is the add-on architecture AdvSegNet. This improves upon the Bayesian SegNet, one of the many popular architectures in deep learning-based semantic segmentation. It provides a generator adversarial learning-based approach to improving the segmentation accuracy of the Bayesian SegNet as well as training it to recover from attacks that damage the input resolution of the network. The AdvSegNet builds on the Bayesian SegNet by utilising both the mean and the variance prediction. The variance prediction is a 2D greyscale image map.

In Chapter 3 it is seen that AdvSegNet trains two separate discriminators (binary classifiers) that penalise both the mean and the variance from the Bayesian SegNet. The former is termed quality critic and the latter the uncertainty critic. The quality critic loss function measures the discrepancy between the predicted mean labels from the Bayesian SegNet and the ground truth labels. The uncertainty critic measures the discrepancy between the variance image map and a perfect solution, a blank white image. The novelty of this approach is that it can use uncertainty information to update its weights. Previous approaches in segmentation merely used uncertainty information to highlight the confidence of their predictions. Furthermore, the approaches in the past dealt with model uncertainty in the presence of a small dataset by either increasing the size of the dataset or the network architecture. However, with AdvSegNet, it is possible to improve segmentation accuracy on a small-sized dataset (i.e. $\approx< 1000$ samples) without needing to modify the dataset or the architecture. Another contribution of AdvSegNet is that it can outperform architectures that do not account for nor learn for uncertainty in their predictions. However, a major limitation of this approach comes from continuous sampling to obtain both mean and variance information. Hence, it can only be applied in areas that are not mission-critical since such applications require an instantaneous measure of confidence in predictions for quick decision making e.g. in surgery in medical robotics.

Robust Learning

The application of the proposed methods of this thesis in the area of robust learning is discussed in Chapter 3. The methods used in this chapter make use of the combined architecture of a convolutional neural network and a Gaussian process, i.e. CNN-GP, where the convolutional network acts as a feature extractor and the GP component as the classifier. The main contribution is in the use of regularisation loss that improves robustness to noise and adversarial attacks. The second contribution is the proposed method for generating attacks easily and cheaply by using learned feature space.

The motivation behind the use of the architecture CNN-GP is that it allows rich feature extraction that makes use of the expressiveness of DNNs, along with the accounting of uncertainty from the GP classifier. Furthermore, the loss function for training (i.e. the maximum likelihood loss) is regularised separately with three functions: Kullback-Leibler divergence, Wasserstein distance and maximum correntropy. The results show that using the combined architecture of CNN-GP along with the regularisation improves the robustness to noisy, blurred images and white-box adversarial attacks such as the fast-gradient sign method.

This chapter also contributes to a new way of generating perturbations by using the learned feature space and swapping with images that are neighbours in the feature space. This is a novel contribution since previous approaches in adversarial training have involved training the network on a particular type of attack that is introduced in the training sample during training. However, this limits the performance of the DNN system as it learns to be robust to only the type of attack it is trained on. The proposed method of generating perturbations contributes a faster and simpler way of adversarial training DNN networks by replacing input samples with their nearest neighbours in feature space, without changing labels. This is an easier way to simulate adversarial attacks. Finally, unlike previous approaches, the proposed method of generating attacks is model agnostic.

Additionally, in terms of reliability of uncertainty outputs, previous approaches in adversarial defence utilised uncertainty as a measure of confidence but hardly considered well-calibrated and reliable uncertainties, e.g. in the MC dropout examples. In this architecture, a more reliable way of quickly obtaining uncertainty values is contributed, as results show that obtaining uncertainty from CNN-GP is more reliable and quicker than the state-of-art MC dropout. It is also tested on datasets: MNIST, Fashion-MNIST, CIFAR-10 and CIFAR-100. The limitation of this approach is that it does not perform well on a large and complex dataset. The performance of the CNN-GP is limited by the size of the CNN architecture. A larger number of layers can contribute to the performance, however, it needs to be cautiously picked since a large number of layers can impact the speed at which uncertainty is obtained.

Continual Learning

The application of the proposed methods of this thesis in the area of continual learning is discussed in Chapter 3. The main contribution of this chapter is the extension of the application of the CNN-GP architecture to the problem of continual learning. Continual learning differs from traditional deep learning methods that tend to supply the entire dataset at once or in mini-batches. With continual learning, the dataset is split into multiple tasks with each task containing a sub-set of the ground truth labels. The contribution of the proposed method is that a single classifier is used to train sequentially on all tasks, where each task is a multi-classification problem. This is a different step in the field of continual learning as previous approaches have focused on using a multi-classifier approach to continual learning. In the multi-classifier approach, for each task, a separate classifier is trained. The technique, inspired from UCB, one of the many popular approaches in continual learning, involves updating the learning rate of the architecture proportionally to the learning rate. Specifically, the learning rate is updated by multiplying it with the max by mean of the post-softmax sample variance.

Additionally, the contributions of the CNN-GP approach to continual learning also investigates the efficiency of both the deterministic and the Bayesian settings. In the deterministic setting, the weights of the CNN component are set to scalar and only the learning rate parameters of the GP are updated according to the post-softmax sample variance. In the Bayesian setting, the weights of the CNN are distributed according to a scale Gaussian mixture prior. The variance of the distribution is used to update the learning rate of the CNN while the learning rate of the GP classifier is updated based on the post-softmax sample variance, similar to the deterministic setting. Additionally, another contribution of this method is that the performance of the CNN-GP is evaluated for datasets: split-MNIST, permuted-MNIST, sequential CIFAR-10 and CIFAR-100. Finally, additional tests based on the robustness to noisy images are carried out for split-MNIST and permuted-MNIST. The major limitations of this approach are that it can improve the results of the state-of-art UCB but in a few tasks only. Another limitation of the problem is that the robustness of the architecture to noisy images decreases as the number of tasks increases.

5.2 Application of Proposed Methods in Thesis

The methods proposed in this thesis can be applied to a wide range of purposes. This section aims to list some of the key areas based on the applied methods.

5.2.1 Robotics and Autonomous Vehicle Systems

The field of robotics and autonomous vehicle systems is evolving rapidly while adapting to changes in data, architectures and training algorithms. A common theme underlying applications in robotics and autonomous systems is the importance of such systems to make rational decisions. In doing so, autonomous systems must be able to understand their environment. This is also partly the objective in scene understanding. Scene understanding is also vital for navigation in autonomous systems. This includes the ability of the autonomous systems to avoid obstacles and be able to interact with the environment.

The proposed methods in Chapter 3 deal with scene understanding as part of the application in semantic segmentation. Here, methods adapted from deep learning based computer vision and segmentation can be used to enable robotic systems the ability to interact with their environment. Segmented label maps allow correct classification and localisation of various objects in the scene. It can also help in the decision making of, for example, an autonomous vehicle. Furthermore, the proposed methods also allow both accounting and learning from uncertainty which can play a crucial role in decision making in navigating autonomous systems.

The challenge of applying scene understanding and semantic segmentation to robotics and autonomous systems is that the uncertainty values need to be reliable so that the autonomous system can make a correct sense and judgement of its surroundings. Additionally, the uncertainty outputs also need to be instantaneous or at least quick enough to allow plenty of time for the autonomous system to decide what to do next. This can also be the application for methods in Chapter 3 in robust learning since these builds up from MC dropout methods proposed in Chapter 3 in segmentation learning, from the perspective of both reliability and computational speed.

5.2.2 Medical Imaging

Deep learning based computer vision is currently leading as the most popular approach in medical imaging [174]. Some of the most popular architectures in medical imaging based segmentation are based on convolutional neural networks. The problem with the application of medical imaging is that the image acquisition stage is difficult as the field suffers from bias in the training set which favours one condition over the other. An example of this is that the dataset of patience with lung cancer would be higher in number for example individuals that are smokers. Hence, learning systems made to train on medical imaging are challenged with training on a limited small-sized dataset with high bias. The methods proposed in Chapter 3 in the area of segmentation can provide a solution to this problem as they can be used to improve DNN performance without additional datasets by reducing epistemic uncertainty. Furthermore, applications in medical imaging also require a reliable measure of uncertainty for correct decision making. This is for example in the case of robot surgeons like Da Vinci. In this case, methods proposed in Chapter 3 in segmentation can be used to provide reliable uncertainty measures that are also quicker than sampling-based uncertainty which is popularly used in segmentation, e.g. MC dropout [30].

5.2.3 Security Systems

Having secure artificial intelligence systems is an ongoing challenge. As AI and deep learning technologies become more and more ubiquitous, the security of such systems would be evaluated by both the general public and the scientific research community. The trust of an AI system would be questionable in many applications that involve decisions that determine life and death circumstances such as autonomous driving vehicles and medical surgery. Therefore, the AI systems operating need to be robust to both systematic and physical adversarial attacks. This is in large deal in the field of cybersecurity. The methods proposed in Chapter 3 as part of robust learning can be applied to these application areas. Specifically, dealing with adversarial attacks of white-type, e.g. FGSM. These are more lethal than black-box types which do not have all of the information regarding the AI system.

5.2.4 Recommender Systems

As AI and deep learning technologies grow, the potential applications for such technologies in dynamic dataset based learning also increases. The dynamic style of learning allows models to learn incrementally. In doing so, they learn to change and adapt their behaviour with respect to changes in their inputs, while making sure that the past knowledge and information is not lost or forgotten. This is followed by the ever-changing and growing data in real-life example applications, bringing new challenges. A popular example of this is in recommendation systems. In this popular example such as Amazon and Netflix continuously change their inputs according to the user's input feedback. For example, keeping updated with the kind of movies that the user is interested in. This type of example can also involve capturing various amounts and different types of data that varies every minute of the day. A typical example is theatrical releases of films that are available on the IMDb database.

Hence, methods proposed in Chapter 3 in continual learning can be used for applications in this scenario.

5.3 Future Works

The final section of this thesis concludes with the future works section. This section highlights various directions the thesis can take to further adapt the techniques presented. The majority of these areas are currently subjected to cutting-edge research that dominates the scientific research community's interests. Moreover, the challenges posed by the field of uncertainty aware deep learning and AI remain unexplored and carry a large scope for research. Some of the extensions that can be applied to the methods proposed in the thesis are as follows:

5.3.1 Similarity Guidance in Few-Shot Learning

Few-shot learning is a learning paradigm that involves learning from a few data samples. The majority of the deep learning techniques involve learning from a wide range of training samples from 1000 - 10 million sized datasets. In practice, large datasets are not always available for some of the applications. Adding to this, inspiring from human intelligence, it follows that learning to recognize objects in human beings does not involve having to observe a million images. Usually, one or two images are sufficient for image recognition in human intelligence. The methods from Chapter 3 in segmentation and continual learning can be further improved for few-shot applications. This is because few-shot applications require similarity measures to differentiate the training set from the query set. The query set is the set of images that the network has to learn to differentiate the training samples from. Specifically, the proposed three loss functions: KLD, Wasserstein distance and maximum correntropy can be used to obtain the similarity measure. Another important point is that CNN-GP is a Bayesian model and is ideal for few-shot learning since Bayesian methods are known to be superior to traditional DNN approaches when it comes to training with few samples.

5.3.2 Improving Architectures

The methods proposed from this thesis can be improved in terms of architecture. One example is the use of capsule networks [66] instead of convolutional based architectures. Convolutional architectures, although are a popular choice for deep learning based computer vision, they have a certain number of limitations. One limitation is that they are not rotationally invariant, i.e. they categorize objects differently despite the same objects being made to rotate without changing other parameters such as position or lighting conditions. Additionally, the max-pooling operation (see Section 2.1.1) used in CNNs works by preventing neurons that have low activation values to pass from layer to layer. Briefly, neurons with the highest amount of activation are passed. Although this concept is inspired by the winner-take-all dynamics in the brain and it serves as a widely adopted method for feature dimensionality reduction, it is disadvantageous from the point of view of transferring useful spatial information [66].

In capsules networks, each weight component is a vector that captures two properties: the probability of an object present in an image and the instantiating properties (i.e. physical and geometrical properties such as shape, size and lighting condition). Additionally, the learning by maximising from softmax likelihood is replaced with a routing-by-agreement mechanism [66]. In CNNs, the activations from a higher layer do not communicate with the lower layer because of the max-pooling, this is different in capsule networks. In capsule networks, the higher layer passes its activation to the lower layer whose dot product of prediction vectors is the highest, when compared with the dot product of predictions with other lower layers. Capsule networks overall perform better than convolutional networks but the limitation is that they are difficult to train with. Improving training in capsule networks and its firm application in different learning paradigms is an open research question that can be a good direction for this thesis.

5.3.3 Further Tests on Adversarial Attacks

Despite deep learning emerging as a suitable and efficient candidate for solving complex learning problems, it still suffers from the vulnerability to adversarial attacks. Well-crafted adversarial examples appear imperceptible to the human eye but are sufficient enough to compromise deep neural networks. The methods proposed in Chapter 4 can be tested to various adversarial attacks, of either black-box or white-box attacks.

5.3.4 Weakly Semi-Supervised Segmentation in Medical Imaging

Weakly semi-supervised learning deals with the problem of supervised learning where the labels are partially available or are labelled incorrectly. This is a common issue in data acquisition in medical imaging. The reason behind this is that pixel-wise annotation in medical imaging is laborious and expensive. Annotating medical images requires experienced physicians who take long hours (even days), to investigate the medical images for accurate segmentation. However, even this is not guaranteed to be perfect as the labels can contain defects. Defects in labels can have a substantial effect on the training of the DNN network being used. A solution to this is in the application of weakly semi-supervised segmentation. Methods in this field involve generating segmentation maps of unlabeled data using uncertainty measures. For example, [157] uses MC dropout to generate uncertainty maps, the confidence levels are used in a cross-entropy loss function to train the network to generate labels for the unlabelled examples. This is a good direction for the methods proposed in Chapter 3 which focus on leveraging uncertainty maps for improving segmentation performance.

References

[1] Abadi, M. et al. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.

[2] Abhishek, K., Singh, M., Ghosh, S., and Anand, A. (2012). Weather Forecasting Model Using Artificial Neural Network. *Procedia Technology*, 4:311–318. 2nd International Conference on Computer, Communication, Control and Information Technology.

[3] Ahmad Hijazi, M. H. (2012). *Image Classification: A Study in Age-Related Macular Degeneration Screening*. PhD thesis, University of Liverpool.

[4] Alvarez-Melis, D. and Jaakkola, T. S. (2018). On the Robustness of Interpretability Methods. Presented in 2018 ICML Workshop on Human and Interpretability in Machine Learning (WHI 2018), Stockholm, Sweden.

[5] Anthony, M. and Bartlett, P. L. (2009). *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, New York, NY, USA, 1st edition.

[6] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein Generative Adversarial Networks. In Precup, D. and Teh, Y. W., editors, *In Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 214–223. PMLR.

[7] Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 39(12):pages 2481–2495.

[8] Barber, D. and Bishop, C. M. (1998). Ensemble learning in bayesian neural networks. *NATO ASI SERIES F COMPUTER AND SYSTEMS SCIENCES*, 168:215–238.

[9] Barber, D. and Schottky, B. (1997). Radial Basis Functions: A Bayesian Treatment. *Advances in Neural Information Processing Systems*, volume 10:pages 402–408.

[10] Bassetti, F., Casarin, R., and Ravazzolo, F. (2018). Bayesian Non-Parametric Calibration and Combination of Predictive Distributions. *Journal of the American Statistical Association*, volume 113(522):pages 675–685.

[11] Bellemare, M. G., Danihelka, I., Dabney, W., Mohamed, S., Lakshminarayanan, B., Hoyer, S., and Munos, R. (2017). The Cramer Distance as a Solution to Biased Wasserstein Gradients. *arXiv preprint arXiv:1705.10743*.

[12] Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA.

[13] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

[14] Blake, A., Curwen, R., and Zisserman, A. (1995). *A Framework for Spatio-Temporal Control in the Tracking of Visual Contours*, page 3–33. Cambridge University Press, USA.

[15] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2018). Weight Uncertainty in Neural Networks. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 520–535, Lille, France.

[16] Brostow, G. J., Fauqueur, J., and Cipolla, R. (2009). Semantic Object Classes in Video: A High-Definition Ground Truth Database. *Pattern Recognition Letters*, volume 30(2):pages 88–97.

[17] Carannante, G., Dera, D., Rasool, G., Bouaynaya, N. C., and Mihaylova, L. (2020). Robust Learning via Ensemble Density Propagation in Deep Neural Networks. In *Proceedings of the 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE.

[18] Chakraborty, S., Tomsett, R., Raghavendra, R., Harborne, D., Alzantot, M., Cerutti, F., Srivastava, M., Preece, A., Julier, S., Rao, R. M., et al. (2017). Interpretability of deep learning models: A survey of results. In *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation*, pages 1–6. IEEE.

[19] Chaudhry, A., Dokania, P. K., Ajanthan, T., and Torr, P. H. (2018). Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547.

[20] Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2018). DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 40(4):pages 834–848.

[21] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2015). Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. In *Proceedings of the 2015 Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA.

[22] Chen, Z. and Liu, B. (2018). Lifelong Machine Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, volume 12(3):pages 1–207.

[23] Chen Janet, L. S.-I. and Dan, V. (1998). Stratergies of Play. Available at: https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/game-theory/Minimax.html. (Accessed Feburary 6th, 2021).

[24] Christian, R. and George, C. (2005). *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg.

[25] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223.

[26] Da San Martino, G. (2009). *Kernel Methods for Tree Structured Data*. PhD thesis, University of Bologna, Padova.

[27] Dawid, A. P. (1982). The Well-Calibrated Bayesian. *Journal of the American Statistical Association*, volume 77(379):pages 605–610.

[28] Delange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. (2021). A Continual Learning Survey: Defying Forgetting in Classification Tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1.

[29] Dera, D., Rasool, G., and Bouaynaya, N. (2019). Extended Variational Inference for Propagating Uncertainty in Convolutional Neural Networks. In *In 29th IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE.

[30] DeVries, T. and Taylor, G. W. (2018). Leveraging Uncertainty Estimates for Predicting Segmentation Quality. In *1st Conference on Medical Imaging with Deep Learning (MIDL)*, Amsterdam, The Netherlands.

[31] Dietterich, T. G. (2017). Steps Toward Robust Artificial Intelligence. *AI Magazine*, volume 38(3):pages 3–24.

[32] Ebrahimi, S., Elhoseiny, M., Darrell, T., and Rohrbach, M. (2018). Uncertainty-Guided Continual Learning with Bayesian Neural Networks. In *8th International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia.

[33] Elsken, T., Metzen, J. H., and Hutter, F. (2019). Neural Architecture Search: A Survey. *Journal of Machine Learning Research*, volume 20:pages 55:1–55:21.

[34] Ess, A., Mueller, T., Grabner, H., and Gool, L. V. (2009). Segmentation Based Urban Traffic Scene Understanding. In *Proceedings of the 2009 Conference on British Machine Vision (BMVC)*, volume 1, page 2.

[35] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, volume 88(2):pages 303–338.

[36] Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. (2021). Sharpness-Aware Minimization for Efficiently Improving Generalization. In *9th International Conference on Learning Representations (ICLR)*, Virtual Conference.

[37] French, R. M. (1994). Dynamically Constraining Connectionist Networks to Produce Distributed, Orthogonal Representations to Reduce Catastrophic Interference. In *Proceedings of the 16th Annual Cognitive Science Society Conference*, pages 335–340, Atlanta, USA.

[38] French, R. M. (1999). Catastrophic Forgetting in Connectionist Networks. *Trends in Cognitive Sciences*, volume 3(4):pages 128–135.

[39] Gal, Y. (2016). Uncertainty In Deep Learning. *University of Cambridge*, volume 1(3).

[40] Gal, Y. and Ghahramani, Z. (2015). Dropout as a Bayesian Approximation: Insights and Applications. In *Deep Learning Workshop, ICML.*

[41] Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *International Conference on Machine Learning (ICML)*, pages 1050–1059.

[42] Gast, J. and Roth, S. (2018). Lightweight Probabilistic Deep Networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3369–3378, Salt Lake City, Utah, USA.

[43] Geiger, A. (2012). Are We Ready for Autonomous Driving ? The Kitti Vision Benchmark Suite. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR'12)*, pages 3354–3361. IEEE Computer Society.

[44] Ghahramani, Z. (2013). Bayesian Non-Parametrics and the Probabilistic Approach to Modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, volume 371(1984):pages 20110553–20110553.

[45] Ghassemi, P., Lulekar, S. S., and Chowdhury, S. (2019). Adaptive Model Refinement with Batch Bayesian Sampling for Optimization of Bio-Inspired Flow Tailoring. In *AIAA Aviation 2019 Forum*, page 2983.

[46] Ghosh, J. K. and Ramamoorthi, R. (2003). *Bayesian Non-Parametrics*. Springer Science & Business Media.

[47] Gonzalez, R. C. and Woods, R. E. (2008). *Digital Image Processing*. Prentice Hall.

[48] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[49] Goodfellow, I., Shlens, J., and Szegedy, C. (2015). Explaining and Harnessing Adversarial Examples. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

[50] Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. (2013). An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. *arXiv preprint arXiv:1312.6211*.

[51] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Networks. In *2014 Advances in Neural Information Processing Systems (NIPS)*, Palais des Congrès de Montréal, Montréal, Canada.

[52] Graves, A. (2011). Practical Variational Inference for Neural Networks. In *Advances In Neural Information Processing Systems*, pages 2348–2356.

[53] Grossberg, S. T. (2012). *Studies of Mind and Brain: Neural Principles of Learning, Perception, Development, Cognition, and Motor Control*, volume 70. Springer Science & Business Media.

[54] Gu, S. and Rigazio, L. (2018). Towards Deep Neural Network Architectures Robust to Adversarial Examples. In *6th International Conference on Learning Representations (ICLR)*, Vancouver Convention Center, Vancouver, Canada.

[55] Gu, Y., Wang, Y., and Li, Y. (2019). A Survey on Deep Learning-Driven Remote Sensing Image Scene Understanding: Scene Classification, Scene Retrieval and Scene-guided Object Detection. *Applied Sciences*, volume 9(10):pages 2110.

[56] Hancox-Li, L. (2020). Robustness in Machine Learning Explanations: Does it Matter? In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 640–647, Barcelona, Spain.

[57] Harakeh, A. (2017). Adversarial Robustness of Uncertainty Aware Deep Neural Networks. Technical report, University of Toronto, Faculty of Applied Science & Engineering.

[58] Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition.

[59] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, USA.

[60] He, X., Zemel, R. S., and Carreira-Perpinán, M. A. (2004). Multiscale Conditional Random Fields for Image Labeling. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1–2. IEEE.

[61] Hein, M. and Andriushchenko, M. (2017). Formal Guarantees on the Robustness of a Classifier Against Adversarial Manipulation. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*, Long Beach Convention Center, Long Beach, USA.

[62] Heinonen, J. (2005). *Lectures on Lipschitz Analysis*. Number 100. University of Jyväskylä.

[63] Hensman, J., Matthews, A. G., and Ghahramani, Z. (2015). Scalable Variational Gaussian Process Classification. *Proceedings of Machine Learning Research*, volume 38:pages 351–360.

[64] Hernández-Lobato, J. M. and Adams, R. P. (2015). Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML)*, page 1861–1869, Lille, France.

[65] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*, Palais des Congrès de Montréal, Montréal, Canada.

[66] Hinton, G. E., Sabour, S., and Frosst, N. (2018). Matrix Capsules with EM Routing. In *6th International Conference on Learning Representations (ICLR)*, Vancouver Convention Center, Vancouver, Canada.

[67] Hjort, N. L., Holmes, C., Müller, P., and Walker, S. G. (2010). *Bayesian Nonparametrics*, volume 28. Cambridge University Press.

[68] Huang, P.-Y., Hsu, W.-T., Chiu, C.-Y., Wu, T.-F., and Sun, M. (2018). Efficient Uncertainty Estimation for Semantic Segmentation in Videos. In *Proceedings of the 2018 European Conference on Computer Vision (ECCV)*, pages 520–535.

[69] Hung, W., Tsai, Y., Liou, Y., Lin, Y., and Yang, M. (2018). Adversarial Learning for Semi-Supervised Semantic Segmentation. *CoRR*, abs/1802.07934.

[70] Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML)*, pages 448–456.

[71] Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive Mixtures of Local experts. *Neural Computation*, 3(1):79–87.

[72] Jessica, G. (2016). Google Photos Labeled Black People 'Gorillas'. Avaialble at: https://eu.usatoday.com/story/tech/2015/07/01/google-apologizes-after-photos-identify-black-people-as-gorillas/29567465/. USA Today, 2015. (Accessed October 30, 2019).

[73] Jospin, L. V., Buntine, W., Boussaid, F., Laga, H., and Bennamoun, M. (2020). Hands-On Bayesian Neural Networks–A Tutorial for Deep Learning Users. *ACM Computer Survey*, volume 1(1).

[74] Jung, H., Ju, J., Jung, M., and Kim, J. (2016). Less-Forgetting Learning in Deep Neural Networks. *arXiv preprint arXiv:1607.00122*.

[75] Kälviäinen, R. and Uusitalo, H. (2007). DIARETDB1 Diabetic Retinopathy Database and Evaluation Protocol. In *Proceedings of the 2007 Conference on British Machine Vision Association (BMVC)*, volume 1, pages 1–10.

[76] Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2018). Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *6th International Conference on Learning Representations (ICLR)*, Vancouver Convention Center, Vancouver, Canada.

[77] Kendall, A. (2018). *Geometry and Uncertainty in Deep Learning for Computer Vision*. PhD thesis, University of Cambridge.

[78] Kendall, A., Badrinarayanan, V., and Cipolla, R. (2017). Bayesian SegNet: Model Uncertainty in Deep Convolutional Eencoder-Decoder Architectures for Scene Understanding. In *28th British Machine Vision Conference (BMVC)*, London, UK.

[79] Kendall, A. and Gal, Y. (2017). What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision ? In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, pages 5574–5584.

[80] Kennedy, M. C. and O'Hagan, A. (2001). Bayesian Calibration of Computer Models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, volume 63(3):pages 425–464.

[81] Kingma, D. P. and Ba, J. (2015). Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA.*

[82] Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational Dropout and the Local Reparameterization Trick. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS)*, pages 2575–2583.

[83] Kingma, D. P. and Welling, M. (2014). Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations (ICLR)*, Banff, AB, Canada.

[84] Krizhevsky, A. and Hinton, G. (2010). Convolutional Deep Belief Networks on CIFAR-10. Technical report, University of Toronto.

[85] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems (NIPS)*, volume 25:pages 1097–1105.

[86] Kuhn, M., Johnson, K., and Others (2013). *Applied Predictive Modeling*, volume 26. Springer.

[87] Kullback, S. and Leibler, R. A. (1951). On Information and Sufficiency. *The Annals of Mathematical Statistics*, volume 22(1):pages 79–86.

[88] Kwon, Y., Won, J.-H., Kim, B. J., and Paik, M. C. (2018). Uncertainty Quantification Using Bayesian Neural Networks in Classification: Application to Ischemic Stroke Lesion Segmentation. In *1st Conference on Medical Imaging with Deep Learning (MIDL)*, Amsterdam, The Netherlands.

[89] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, volume 86(11):pages 2278–2324.

[90] LeCun, Y., Cortes, C., and Burges, C. (2010). *MNIST Handwritten Digit Database*. Available at: http://yann.lecun.com/exdb/mnist. (Accessed November 2, 2020).

[91] LeCun, Y., Huang, F. J., and Bottou, L. (2004). Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages II–104, Washington, DC, USA. IEEE.

[92] Lee, F. and Andrej, K. (2010). *CS231n Convolutional Neural Networks for Visual Recognition*. Available at: http://engineering.purdue.edu/~mark/puthesis. (Accessed July 25, 2018).

[93] Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2009). Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616.

[94] Li, J., Wong, Y., Zhao, Q., and Kankanhalli, M. S. (2019). Learning to Learn From Noisy Labeled Data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5051–5059.

[95] Li, Z. and Hoiem, D. (2017). Learning Without Forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 40(12):pages 2935–2947.

[96] Lin, C.-Y., Chiu, Y.-C., Ng, H.-F., Shih, T. K., and Lin, K.-H. (2020). Global-and-Local Context Network for Semantic Segmentation of Street View Images. *Sensors*, volume 20(10):pages 2907.

[97] Lin, G., Shen, C., Van Den Hengel, A., and Reid, I. (2016). Efficient Piecewise Training of Deep Structured Models for Semantic Segmentation. In *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition*, pages 3194–3203.

[98] Lin, M., Chen, Q., and Yan, S. (2014). Network in Network. *CoRR*, volume abs/1312.4400.

[99] Liu, H., Simonyan, K., and Yang, Y. (2018). Darts: Differentiable Architecture Search. In *7th International Conference on Learning Representations (ICLR)*, Ernest N. Morial Convention Center, New Orleans, USA.

[100] Liu, W., Rabinovich, A., and Berg, A. C. (2015). ParseNet: Looking Wider to See Better. *CoRR*.

[101] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully Convolutional Networks for Semantic Segmentation. In *Proceedings of the 28th IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440.

[102] Luc, P., Couprie, C., Chintala, S., and Verbeek, J. (2016). Semantic Segmentation Using Adversarial Networks. *CoRR*, abs/1611.08408.

[103] Lundberg, S. M. and Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, page 4768–4777, Red Hook, NY, USA. Curran Associates Inc.

[104] MacKay, D. and David, J. (2003). *Information theory, inference and learning algorithms*. Cambridge university press.

[105] MacKay, D. and JC, D. (2001). A Problem with Variational Free Energy Minimization. Technical report, Department of Physics, University of Cambridge.

[106] Malinin, A. and Gales, M. (2018). Predictive Uncertainty Estimation via Prior Networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS)*, pages 7047–7058, Montréal, Canada.

[107] Mao, X., Li, Q., Xie, H., Lau, R. Y. K., and Wang, Z. (2016). Multi-class Generative Adversarial Networks with the L2 Loss Function. *CoRR*, abs/1611.04076.

[108] Marcus, G. (2020). The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence. *arXiv preprint arXiv:2002.06177*.

[109] Massi, B., Donahue, C. H., and Lee, D. (2018). Volatility Facilitates Value Updating in the Prefrontal Cortex. *Neuron*, volume 99(3):pages 598–608.

[110] Medsker, L. R. and Jain, L. (2001). Recurrent Neural Networks. *Design and Applications*, volume 5:pages 64–67.

[111] Mermillod, M., Bugaiska, A., and Bonin, P. (2013). The Stability-Plasticity Dilemma: Investigating the Continuum From Catastrophic Forgetting to Age-Limited Learning Effects. *Frontiers in Psychology*, volume 4:pages 504.

[112] Mescheder, L., Nowozin, S., and Geiger, A. (2017). The Numerics of GANs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, page 1823–1833, Red Hook, NY, USA.

[113] Minaee, S., Boykov, Y. Y., Porikli, F., Plaza, A. J., Kehtarnavaz, N., and Terzopoulos, D. (2021). Image Segmentation Using Deep Learning: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. DOI: 10.1109/TPAMI.2021.3059968, PMID:33596172.

[114] Mlodinow, L. (2009). *The Drunkard's Walk: How Randomness Rules our Lives*. Vintage Books/Random House.

[115] Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2016). DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582.

[116] Mundt, M., Pliushch, I., and Ramesh, V. (2021). Neural Architecture Search of Deep Priors: Towards Continual Learning Without Catastrophic Interference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3523–3532.

[117] Neal, R. M. (2012). *Bayesian Learning for Neural Networks*, volume 118. Springer Science & Business Media.

[118] Neal, R. M. and Hinton, G. E. (1998). A View of the EM Algorithm that Justifies Incremental, Sparse, and Other Variants. In *Learning in Graphical Models*, pages 355–368. Springer.

[119] Nedelkoski, S., Bogojeski, M., and Kao, O. (2020). Learning More Expressive Joint Distributions in Multimodal Variational Methods. pages 137–149.

[120] Ng, M., Guo, F., Biswas, L., and Wright, G. A. (2018). Estimating Uncertainty in Neural Networks for Segmentation Quality Control. In *Proceedings of 32nd Conference on Neural Information Processing Systems (NIPS)*, pages 1–4, Montreal, Canada.

[121] Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. (2018). Variational Continual Learning. In *6th International Conference on Learning Representations (ICLR)*, Vancouver Convention Center, Vancouver, Canada.

[122] Noh, H., Hong, S., and Han, B. (2015). Learning Deconvolution Network for Semantic Segmentation. In *Proceedings of the 15th IEEE International Conference on Computer Vision (ICCV)*, pages 1520–1528.

[123] Oberweger, M., Wohlhart, P., and Lepetit, V. (2015). Hands Deep in Deep Learning for Hand Pose Estimation. *CoRR*, abs/1502.06807.

[124] Odena, A., Olah, C., and Shlens, J. (2017). Conditional Image Synthesis with Auxiliary Classifier GANs. In *34th International Conference on Machine Learning (ICML)*, pages 2642–2651. PMLR.

[125] Olkin, I. and Pukelsheim, F. (1982). The Distance Between Two Random Vectors with Given Dispersion Matrices. *Linear Algebra and its Applications*, volume 48:pages 257–263.

[126] Ongie, G., Willett, R., Soudry, D., and Srebro, N. (2020). A Function Space View of Bounded Norm Infinite Width ReLU Nets: The Multivariate Case. In *8th International Conference on Learning Representations, (ICLR)*, Addis Ababa, Ethiopia.

[127] Pallier, C., Dehaene, S., Poline, J.-B., LeBihan, D., Argenti, A.-M., Dupoux, E., and Mehler, J. (2003). Brain Imaging of Language Plasticity in Adopted Adults: Can a Second Language Replace the First ? *Cerebral Cortex*, volume 13(2):pages 155–161.

[128] Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. (2016a). The Limitations of Deep Learning in Adversarial Settings. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. IEEE.

[129] Papernot, N., McDaniel, P. D., Sinha, A., and Wellman, M. P. (2016b). Towards the Science of Security and Privacy in Machine Learning. *CoRR*, abs/1611.03814.

[130] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

[131] Paszke, A. et al. (2019). Pytorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32 (NIPS)*, pages 8024–8035.

[132] Pavlitskaya, S., Hubschneider, C., Weber, M., Moritz, R., Hüger, F., Schlicht, P., and Zöllner, J. M. (2020). Using Mixture of Expert Models to Gain Insights into Semantic Segmentation. pages 1399–1406. Computer Vision Foundation, IEEE.

[133] Pawlowski, N., Brock, A., Lee, M. C., Rajchl, M., and Glocker, B. (2017). Implicit Weight Uncertainty in Neural Networks. In *2nd Workshop on Bayesian Deep Learning in Neural Information Processing Systems (NIPS)*, Long Beach, CA, USA.

[134] Pixabay (2019). Pixabay Panda Picture by qgadrian. URL https://cdn.pixabay.com/photo/2019/09/08/19/54/panda-4461766_1280.jpg [Online; Accessed April 20, 2022].

[135] Postels, J., Ferroni, F., Coskun, H., Navab, N., and Tombari, F. (2019). Sampling-Free Epistemic Uncertainty Estimation Using Approximated Variance Propagation. In *Proceedings of the 32nd IEEE/CVF International Conference on Computer Vision*, pages 2931–2940.

[136] Qi, Y., Wang, Y., Zhang, J., Zhu, J., and Zheng, X. (2014). Robust Deep Network with Maximum Correntropy Criterion for Seizure Detection. *BioMed Research International*, 2014.

[137] Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico.

[138] Raftery, A. E., Gneiting, T., Balabdaoui, F., and Polakowski, M. (2005). Using Bayesian Model Averaging to Calibrate Forecast Ensembles. *Monthly Weather Review*, volume 133(5):pages 1155–1174.

[139] Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Sohl-Dickstein, J. (2017). On the Expressive Power of Deep Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 2847–2854.

[140] Ranzato, M., Huang, F. J., Boureau, Y.-L., and LeCun, Y. (2007). Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE.

[141] Rao, Q. and Frtunikj, J. (2018). Deep Learning For Self-Driving Cars: Chances and Challenges. In *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*, pages 35–38.

[142] Rasmussen, C. and Ghahramani, Z. (2001). Occam's Razor. In *Advances in Neural Information Processing Systems (NIPS)*, volume 13. MIT Press.

[143] Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

[144] Rawat, A., Wistuba, M., and Nicolae, M.-I. (2017). Harnessing Model Uncertainty for Detecting Adversarial Examples. In *NIPS Workshop on Bayesian Deep Learning*.

[145] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "Why Should I Trust You ?" Explaining the Predictions of any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144.

[146] Robbins, H. (2007). A Stochastic Approximation Method. *Annals of Mathematical Statistics*, volume 22:pages 400–407.

[147] Robins, A. (1995). Catastrophic Forgetting, Rehearsal and Pseudorehearsal. *Connection Science*, volume 7(2):pages 123–146.

[148] Rogers, S. and Girolami, M. (2011). *A First Course in Machine Learning*. Chapman Hall/CRC, 1st edition.

[149] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Proceedings of the 18th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241. Springer.

[150] Roth, W. and Pernkopf, F. (2016). Variational Inference in Neural Networks using an Approximate Closed-form Objective. In *Proceedings of the NIPS 2016 Workshop on Bayesian Deep Learning*.

[151] Russakovsky, O. et al. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, volume 115(3):pages 211–252.

[152] Samangouei, P., Kabkab, M., and Chellappa, R. (2018). Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, Vancouver Convention Center, Vancouver, Canada.

[153] Samarasinghe, S. (2006). *Neural Networks for Applied Sciences and Engineering*. Auerbach Publications, Boston, MA, USA.

[154] Scherer, D., Müller, A., and Behnke, S. (2010). Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In *International Conference on Artificial Neural Networks*, pages 92–101. Springer.

[155] Schmidhuber, J. (1987). *Evolutionary Principles in Self-Referential Learning, or on Learning How to Learn: The Meta-Meta-... Hook.* PhD thesis, Technische Universität München.

[156] Schwing, A. G. and Urtasun, R. (2015). Fully Connected Deep Structured Networks. *CoRR*, abs/1503.02351.

[157] Sedai, S., Antony, B., Rai, R., Jones, K., Ishikawa, H., Schuman, J., Gadi, W., and Garnavi, R. (2019). Uncertainty Guided Semi-Supervised Segmentation of Retinal Layers in OCT Images. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 282–290. Springer.

[158] Seeböck, P., Orlando, J. I., Schlegl, T., Waldstein, S. M., Bogunović, H., Klimscha, S., Langs, G., and Schmidt-Erfurth, U. (2019). Exploiting Epistemic Uncertainty of Anatomy Segmentation for Anomaly Detection in Retinal OCT. *IEEE Transactions on Medical Imaging*, volume 39(1):pages 87–98.

[159] Selesnick, I. W. and Burrus, C. S. (1998). Fast Convolution and Filtering.

[160] Sennrich, R. (2018). Neural Machine Translation: Breaking the Performance Plateau. http://www.meta-net.eu/events/meta-forum-2016/slides/09_sennrich.pdf. (Accessed November 6th, 2020).

[161] Sensoy, M., Kaplan, L., Cerutti, F., and Saleki, M. (2020). Uncertainty-Aware Deep Classifiers Using Generative Models. In *Proceedings of the 34th International Conference on Artificial Intelligence (AAAI)*, volume 34, pages 5620–5627.

[162] Shademan, A., Decker, R. S., Opfermann, J. D., Leonard, S., Krieger, A., and Kim, P. C. (2016). Supervised Autonomous Robotic Soft Tissue Surgery. *Science Translational Medicine*, volume 8(337):pages 337ra64–337ra64.

[163] Silver, D. L. and Mercer, R. E. (2002). The Task Rehearsal Method of Life-Long Learning: Overcoming Impoverished Data. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 90–101. Springer.

[164] Simonyan, K. and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, The Hilton San Diego Resort Spa, San Diego, USA.

[165] Smith, L. and Gal, Y. (2018). Understanding Measures of Uncertainty for Adversarial Example Detection. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 560–569.

[166] Smola, A. J., Vishwanathan, S., and Eskin, E. (2004). Laplace Propagation. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16, pages 441–448. MIT Press.

[167] Solis, M. (2016). New frontiers in robotic surgery: The latest high-tech surgical tools allow for superhuman sensing and more. *IEEE pulse*, 7(6):51–55.

[168] Song, H., Kim, M., Park, D., and Lee, J.-G. (2020). Learning From Noisy Labels with Deep Neural Networks: A Survey. *arXiv preprint arXiv:2007.08199.*

[169] Song, S., Lichtenberg, S. P., and Xiao, J. (2015). Sun RGB-D: A RGB-D Scene Understanding Benchmark Suite. In *Proceedings of the 28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576.

[170] Song, W., Zheng, N., Zheng, R., Zhao, X., and Wang, A. (2019). Digital Image Semantic Segmentation Algorithms: A Survey. *J. Inf. Hiding Multim. Signal Process.*, volume 10(1):pages 196–211.

[171] Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving For Simplicity: The All Convolutional Net. In *3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA.

[172] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, volume 15(1):pages 1929–1958.

[173] Su, J., Vargas, D. V., and Sakurai, K. (2019). One Pixel Attack for Fooling Deep Neural Networks. *IEEE Transactions on Evolutionary Computation*, volume 23(5):pages 828–841.

[174] Suzuki, K. (2017). Overview of Deep Learning in Medical Imaging. *Radiological Physics and Technology*, volume 10(3):pages 257–273.

[175] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing Properties of Neural Networks. In *2nd International Conference on Learning Representations (ICLR)*, Banff, AB, Canada.

[176] Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1701–1708.

[177] Tarvainen, A. and Valpola, H. (2017). Mean Teachers Are Better Role Models: Weight-Averaged Consistency Targets Improve Semi-Supervised Deep Learning Results. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, pages 1195–1204.

[178] Taylor, J. (1997). *Introduction to Error Analysis, the Study of Uncertainties in Physical Measurements*.

[179] Tesla (2016). A Tragic Loss. Available at: https://www.tesla.com/en_GB/blog/tragic-loss. (Accessed October 30, 2019).

[180] Thoma, M. (2016). A Survey of Semantic Segmentation. *CoRR*, volume abs/1602.06541.

[181] Tomsett, R., Widdicombe, A., Xing, T., Chakraborty, S., Julier, S., Gurram, P., Rao, R., and Srivastava, M. (2018). Why the Failure? How Adversarial Examples Can

Provide Insights for Interpretable Machine Learning. In *21st International Conference on Information Fusion (FUSION)*, pages 838–845. IEEE.

[182] Turner, R. E. and Sahani, M. (2011). Two Problems With Variational Expectation Maximisation for Time-Series Models. In Barber, D., Cemgil, T., and Chiappa, S., editors, *Bayesian Time Series Models*, chapter 5, pages 109–130. Cambridge University Press.

[183] Ulku, I. and Akagunduz, E. (2019). A Survey on Deep Learning-Based Architectures for Semantic Segmentation on 2D Images. *CoRR*, volume abs/1912.10230.

[184] Valada, A., Dhall, A., and Burgard, W. (2016). Convoluted Mixture of Deep Experts for Robust Semantic Segmentation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Workshop, State Estimation and Terrain Perception for All Terrain Mobile Robots*, volume 2.

[185] van der Wilk, M., Rasmussen, C. E., and Hensman, J. (2017). Convolutional Gaussian Processes. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA*, pages 2849–2858.

[186] Wei, G.-Q., Arbter, K., and Hirzinger, G. (1997). Automatic tracking of laparoscopic instruments by color coding. In *Proceedings of the 1st Joint Conference on Computer Vision, Virtual Reality and Robotics in Medicine and Medial Robotics and Computer-Assisted Surgery*, page 357–366, Berlin, Heidelberg. Springer-Verlag.

[187] Weng, T.-W., Zhang, H., Chen, P.-Y., Yi, J., Su, D., Gao, Y., Hsieh, C.-J., and Daniel, L. (2018). Evaluating the Robustness of Neural Networks: An Extreme Value Theory Approach. In *6th International Conference on Learning Representations (ICLR)*, Vancouver Convention Center, Vancouver, Canada.

[188] Wickstrøm, K., Kampffmeyer, M., and Jenssen, R. (2018). Uncertainty Modeling and Interpretability in Convolutional Neural Networks for Polyp Segmentation. In *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE.

[189] Wu, A., Nowozin, S., Meeds, E., Turner, R. E., Hernandez-Lobato, J. M., and Gaunt, A. L. (2019). Deterministic Variational Inference for Robust Bayesian Neural Networks. In *7th International Conference on Learning Representations (ICLR).*

[190] Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *CoRR*, volume abs/1708.07747.

[191] Yan, Z., Guo, Y., and Zhang, C. (2018). Deep Defense: Training DNNs With Improved Adversarial Robustness. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS)*, Red Hook, NY, USA.

[192] Zeiler, M. D. and Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. In *European Conference on Computer Vision*, pages 818–833. Springer.

[193] Zenke, F., Poole, B., and Ganguli, S. (2017). Continual Learning Through Synaptic Intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR.

[194] Zhang, J., Zhang, J., Ghosh, S., Li, D., Tasci, S., Heck, L., Zhang, H., and Kuo, C.-C. J. (2020). Class-Incremental Learning Via Deep Model Consolidation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1131–1140.

[195] Zhang, Y., Xiang, T., Hospedales, T. M., and Lu, H. (2018). Deep Mutual Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4320–4328.

[196] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. (2015). Conditional Random Fields as Recurrent Neural Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537.