Risk-Aware Neural Network Ensembles

Misael Alpizar Santana

PhD

University of York

Computer Science

December 2022

Abstract

Autonomous systems with safety-critical concerns such as self-driving vehicles must be able to mitigate risk by dependably detecting entities that represent factors of risk in their environment (e.g., humans and obstacles). Nevertheless, the machine learning (ML) techniques that these systems use for image classification and real-time object detection disregard risk factors in their training and verification. As such, they produce ML models that place equal emphasis on the correct detection of all classes of objects of interest—including, for instance, buses, pedestrians and birds in a self-driving scenario.

To address this limitation of existing solutions, this thesis proposes an approach for the development of *risk-aware ML ensembles* applied to image classification. The new approach (i) allows the risk of misclassification between different pairs of classes to be quantified individually, (ii) guides the training of deep neural network classifiers towards mitigating the risks that require treatment, and (iii) synthesises risk-aware ensembles with the aid of multi-objective genetic algorithms that seek to optimise the ensemble performance metrics while also mitigating risks.

Additionally, the thesis extends the applicability of this approach to real-time object detection (RTOD) deep neural networks. RTOD involves detecting objects of interest and their positions within an image using bounding boxes, and the RTOD extension of the approach employs a suite of new algorithms to combine the bounding box predictions of the models from the risk-aware RTOD ensemble.

Last but not least, the thesis introduces a self-adaptation approach that leverages risk-aware RTOD ensembles to improve the safety of an autonomous system. To that end, the new approach switches dynamically between ensembles with different risk-aware profiles as the system moves between *regions* of its operational design domain. This dynamic RTOD selection approach is shown to reduce the number of crashes and to increase the number of correct actions for a simulated autonomous vehicle.

Contents

To my grandparents Gregorio and Juanita and my siblings Isra and Irma Selene. For being with me when the world is inhospitable.

Acknowledgements

I would like to express my deepest respect, admiration and gratitude to my supervisors Prof. Radu Calinescu and Dr. Colin Paterson who guided my research and gave me advice and support through all the stages of this project. Without their constant supervision and suggestions this endeavor would not have been possible.

I would like to express my gratitude and appreciation for Dr. Simos Gerasimou my internal assessor whose support and encouragement has been invaluable throughout this project.

I would like to thank Consejo Nacional de Ciencia y Tecnología (CONACYT) for providing the founding necessary to complete this research project.

I am especially grateful to the members of the Assuring Autonomy International Programme for the interesting and engaging seminars and discussions. Special thanks to my colleagues Dr. Ioannis Stefanakos, Dr. Saud Yonbawi, Dr. Naif Alasmari, Brendan Devlin-Hill, Gricel Vazquez, Qi Zhang, Dr. Xinwei Fang and Dr. Calum Imrie.

Thanks should also go to Dr. Marcelo Romero Huertas, MCompSci. Sara Vera Noguez, and Dr. José Raymundo Marcial Romero for their support and advice on the early stages of this project.

I would be remiss in not mentioning Sergio Estrada, Hugo Celis, family Chavez Vera and Erika Jaimes for their support, their encouragement and for always believing in me.

Finally, I would like to acknowledge the endless efforts and support of all my family. My deepest gratitude for their love and patience during this journey.

Declaration

I declare that this thesis is a presentation of original work and that I am its sole author. This work has not previously been presented for an award at this, or any other, university. All sources are acknowledged as references.

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

- Misael Alpizar Santana, Radu Calinescu, and Colin Paterson. "Mitigating Risk in Neural Network Classifiers." in 48th Euromicro Conference Series on Software Engineering and Advanced Applications (SEAA). IEEE, 2022.

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

- Misael Alpizar Santana, Radu Calinescu, and Colin Paterson. "Risk-aware Real-time Object Detection." in 18th European Dependable Computing Conference (EDCC). IEEE, 2022. pp. 105-108.

Chapter 1

Introduction

1.1 Motivation

Machine learning (ML) is envisaged to enable autonomous systems to make safety-critical decisions previously reserved for humans. ML techniques such as deep learning [80] have been used to train deep neural network (DNN) classifiers. A DNN is a structure comprising an input layer, an output layer and several hidden layers in between. Each layer contains a set of neurons which apply a non-linear function to a weighted sum of inputs from the previous layer. A DNN learns to perform particular tasks through training—examples of such tasks include image classification and object detection. During training, the strength of the connections between neurons is learned. Afterwards, the trained DNN is used to perform the same task on novel inputs [28]. DNNs have been shown to perform remarkably well on a wide range of problems. These include, for instance, speech recognition [9, 27, 105, 135], detection and recognition of traffic signs [10, 145], autonomous driving [24, 64, 83, 127], diagnosis of medical conditions [34, 102, 165], identification of risk factors [85, 113, 150] in medical diagnosis, and biomedical imaging leading to a better understanding and the early diagnosis of severe diseases [92, 93, 130].

This thesis focuses on the dependable use of ML-based image classification (Chapter 3) and real-time object detection (RTOD) (Chapter 4) in safety-critical applications such as autonomous driving, and on the dynamic switching between synthesised ensembles as the autonomous system moves between Operational Design Domain (ODD) regions (Chapter 5). Image classification is the process of allocating a single label, taken from a set of possible labels, to an image. This is a fundamental problem in computer vision, where it forms the basis of localisation, detection, and segmentation [24, 116]. On the other hand, object detection is a complex task that deals not only with classification but also with the location of objects in a scene—a task of significant importance to many real-world applications where we need to differentiate between many objects to make sense of our environment [22, 112, 170]. When applied online, e.g., to the successive frames of a video stream, the task is termed *real-time object detection* [120, 117].

Despite significant advances, specifically in the area of deep learning for image classification and object detection, DNNs cannot be 100% accurate due to challenges ranging from insufficient training data [87, 99], class imbalance [88, 172] and imperfect performance evaluation measures [43, 136] to inherent localisation and identification errors [117, 163, 172]. As such, the use of DNNs in safety-critical ap-

plications introduces risks that need to be systematically assessed and mitigated. Nevertheless, the error function minimised by the training of DNN classifiers, and the assessment [148] of their performance are typically oblivious to the risks associated with the intended use of these classifiers. A DNN classifier makes no difference between misclassifying a 30 km/h speed limit sign on a residential road as 130 km/h (an error with potentially fatal consequences) and misclassifying the same 30 km/h speed limit sign as a 20 km/h speed limit sign (an error much less likely to lead to serious incidents). Similarly, they place equal emphasis on the detection accuracy of other classes of objects of interest—including, for example, buses, cats, bikes and birds in an autonomous driving scenario. All these errors in classification are as eagerly avoided as each other during training. all are as keenly counted when assessing classifier performance, with the result that a classifier which avoids a misclassification that poses a significant safety concern is deemed no better than one which avoids a misclassification associated with no risk.

By disregarding the likelihood, consequences, and impact of different misclassifications, DNNs introduce unknown risks that limit their adoption in safety-critical autonomous systems. Even when these risks can be mitigated (e.g., by using special monitors to detect unsafe DNN inputs [14, 96] and traditionally developed and verified software to replace the DNN outputs with fail-safe, suboptimal values when using the DNN is deemed unsafe [23]), this mitigation comes at a high engineering and operational cost and leads to increased system complexity.

The risk factors associated with the use of DNN components need to be mitigated if the adoption of this technology is to find use in safety-critical applications, or otherwise such adoption may lead to endangerment or loss of life. An unfortunate example of this is the Uber self-driving car which failed to properly identify a pedestrian walking alongside a bicycle [143]. As the vehicle and pedestrian paths converged, the self-driving system software classified the pedestrian as an unknown object, as a vehicle, and then as a bicycle with varying expectations on its future travel path. As a result of the crash, the pedestrian unfortunately died. In another relevant accident [147], a Tesla vehicle crashed into an overturned truck and apparently fails to correctly identify a stationary pedestrian. For such systems to be trustworthy and trusted, their developers and operators should demonstrate that the risk factors associated with neural network classification have been appropriately considered, and appropriate risk mitigation has been employed.

This thesis presents a suite of machine learning approaches that overcome this major drawback of current DNN classifiers and real-time object detection DNNs by considering the risks associated with their intended use throughout the ML model training and verification. The risk is defined as the possibility of something bad happening [115]. Risk is normally described in terms of risk sources, potential events, their consequences and their likelihoods [44]. We follow the risk management process recommended by the ISO 31010 standard [65], which provides a framework for selecting and applying risk assessment techniques that are appropriate for the specific context and objectives of an organisation. We chose this standard because it expresses an international consensus on best practices and risk assessment techniques. Additionally, this standard provides a systematic and structured approach to risk assessment that can help organisations identify and assess risks more effectively, and it describes a range of risk assessment techniques, from brainstorming to expert judgment, that can be tailored to the specific needs of the problem at hand.

Last but not least, adopting ISO/IEC 31010 enhances the credibility of a solution by demonstrating that it follows well established risk assessment practices [65].

ISO/IEC 31010 also suits our approach because it provides specific risk assessment techniques such as those that use a consequence/likelihood matrix. In contrast, ML-specific approaches like the 'Assurance of Machine Learning for use in Autonomous Systems' (AMLAS) [59] provide a methodology for systematically integrating safety assurance into the development of machine learnt components. Such approaches propose the identification of safety requirements to control the risk of the identified contributions of the ML component to system hazards, without proposing concrete quantitative/qualitative techniques for assessing the likelihood and severity of identified concerns, as done by the ISO/IEC 31010 standard.

As such, in addition to the usually labelled data used to train and verify DNNs, our approach requires standard risk information—specific for the intended DNN use, and provided by domain experts. This information captures:

(i) The likelihood of encountering each of the classes identified by the DNN. This refers to the likelihood of an event happening and can be described as an expected probability or frequency of encountering an instance of a given class during a determined period of time.

(ii) The level of impact of the possible misclassifications, which considers how the event could influence cost, schedule, or technical performance objectives.

(iii) The maximum risk level that is acceptable.

Given this information, we identify the misclassification risks that require treatment, we train DNNs capable of mitigating individual (or small sets of) such risks, and we combine these DNNs into neural network or RTOD ensembles [58, 82]. Unique to our approach, the risk-aware training involves the use of loss functions tailored to reduce high-risk misclassifications, and the risk-aware ensemble synthesis employs multi-objective genetic algorithms [103] to generate ML model ensembles that are Pareto optimal with respect to both traditional ML performance metrics and risk mitigation capabilities.

1.2 Research contributions

The work presented in this thesis is underpinned by the hypothesis that *the training of DNNs and the synthesis of DNN ensembles can consider and mitigate the risks associated with their intended operating domain, so as to enable autonomous systems to operate with fewer incidents and higher effectiveness.* To validate this hypothesis, the thesis makes the research contributions summarised below.

1. **An approach for risk mitigation in neural network classifiers.** We present in Chapter 3 an approach for effectively identifying and mitigating risks associated with relevant misclassifications for DNN image classifiers. Our approach (i) uses a risk management process recommended by the ISO 31010 standard to identify high-risk misclassifications, (ii) guides the training of DNN classifiers towards mitigating the risks that require treatment, and (iii) synthesises risk-aware ensembles with the aid of multi-objective genetic algorithms

that seek to optimise DNN performance metrics while also mitigating risks. The effectiveness of the approach is assessed through applying it to two widely used data sets, CIFAR-10 [77] and a subset of the German Traffic Sign Recognition Benchmark (GTSRB) [141]. The obtained results from the evaluation indicate that better image classifiers can be constructed by synthesising Pareto-optimal ensembles that include risk-oblivious and risk-aware models.

2. **An approach to synthesising risk-aware ensembles for real-time object detection.** In Chapter 4 we extend the approach from the first contribution for the development of risk-aware ML ensembles for real-time object detection. The extended version supports the dependable use of ML-based RTOD in safety-critical applications such as autonomous driving. The approach was extended to deal with images coming from video streams and containing multiple objects. The key challenge overcome by our new approach is the effective combination of the knowledge from each of the base learners, since each RTOD model is predicting a different set of multiple objects with different locations. To overcome this challenge, we propose three algorithms applied in the ensemble synthesis stage, and used for combining the bounding box predictions of the models in the ensemble. To evaluate our approach, we performed experiments using the PASCAL Visual Object Classes Challenge (VOC) 2007 object detection data set [42] which contains 9,963 images for the training, validation and test of object detector models with 24,640 annotated objects. The data set contains 20 classes from the categories person, animal, vehicle and indoor. The results suggest that our approach can effectively mitigate risk, supporting the development of dependable RTOD-based systems for safety-critical applications.

3. **A method for integrating risk-aware object detection ensembles into autonomous systems.** In Chapter 5 we propose a method that uses several risk-aware ML ensembles with dynamic switching between ensembles as the system moves between different regions of its operational design domain. To evaluate our method we ran multiple episodes of the simulator described in the fourth contribution. This allowed us to assess the effectiveness of ensemble switch as the ODD region changes. We assessed effectiveness by measuring safety as the number of crashes and number of correct actions taken, i.e., slow and fast navigation for an autonomous vehicle, during journeys of the vehicle. The evaluation of our approach shows that the dynamic switching between ensembles as the autonomous vehicle enters a new ODD region has the benefit of improving safety by decreasing the number of crashes and by increasing the number of correct actions taken.

4. **A reusable simulator from the autonomous mobile robot navigation domain.** The open-source 3D robotics simulation platform Gazebo [74] was used to implement a reusable autonomous mobile robot simulator for the evaluation of the method from the previous contribution. The simulator, which includes a simple track-circuit, allows the deployment of mobile robots and cubes with images which represent objects in the space. The simulator assesses the effectiveness of object detection algorithms such as YOLO real-time object detection system [119] by measuring the number of crashes and number

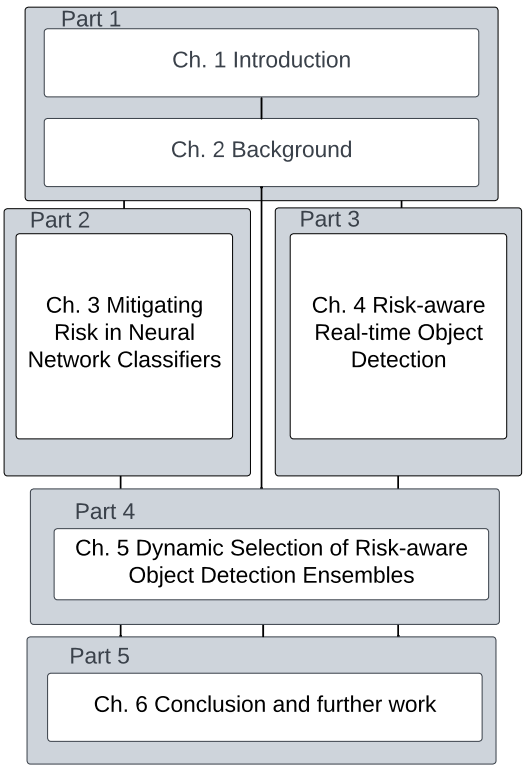


Figure 1.1: The thesis is structured in five parts and contains six chapters, with the main contributions covered in Chapters 3, 4 and 5.

of correct actions taken by the mobile robot. The simulator offers the advantage of testing RTOD methods in a challenging environment due to varying lighting conditions, with the possibility to adjust the speed of the robot, and its angles of object detection. Additionally, mapping the images onto the cubes deforms them by changing their original size, creating multiple types of disturbances.

5. A systematic evaluation of the real-time object detection ensembles from the second contribution and of the method from the third contribution using the simulator from the fourth contribution.

1.3 Thesis structure

As shown in Figure 1.1, the remainder of this thesis is structured as follows.

Chapter 2 introduces background terminology and concepts used throughout the rest of the thesis, and therefore required to understand the proposed methods. It starts with a brief introduction to machine learning in Section 2.1, with an emphasis on supervised learning and in particular deep neural networks applied to image classification and the more complex task of object detection. Next, the chapter provides an introduction to risk assessment in Section 2.2; this is paramount to understanding the core ideas in Chapters 3, 4 and 5. The chapter continues with a succinct presentation of genetic algorithms in Section 2.3, where key concepts and

definitions are addressed. Finally, in Section 2.4, the chapter introduces ensemble learning, providing a description of the most representative ensemble methods.

Chapter 3 presents a four-step approach that considers the risk factors associated with neural network classification and places mitigation strategies to deal with the risk value associated to relevant misclassifications. The proposed method (i) allows the risk misclassification between classes to be quantified, (ii) guides the training of DNN classifiers towards mitigating the risks that require treatment, and (iii) synthesises risk-aware ensembles with the aid of multi-objective genetic algorithms that seek to optimise DNN performance metrics while also mitigating risk. The chapter starts by describing the stages of the mentioned approach in Section 3.1. Next, Section 3.3 presents the evaluation of the method in two different data sets, CIFAR-10 [77] and a subset of the GTSRB [141]. Lastly, it presents the related work in Section 3.4.

Chapter 4 introduces a method for the development of *risk-aware ML ensembles* for real-time object detection. The reported method in this chapter supports the dependable use of real-time object detection by (i) identifying the risks that require treatment, (ii) training a set of ML models that mitigate these risks, and (iii) using multi-objective genetic algorithms to combine the ML models into risk-aware ML ensembles. The chapter begins with a description of the stages of the method in Section 4.1. Following, in Section 4.2 the evaluation of the method in the PASCAL VOC 2007 [42] dataset is presented. Lastly, the related work is developed in Section 4.3.

Chapter 5 describes an approach that advocates the use of risk-aware ML ensembles with dynamic switching between models as a real-time adaptation system (RAS) moves from one ODD region to another. The chapter starts with an introduction in Section 5.1; then the five components of the approach are described in Section 5.2. Next, the evaluation is presented in Section 5.3. Finally, the related work is shown in Section 5.4.

In the last part of this thesis, Chapter 6 presents the conclusions of the thesis by summarising its findings and contributions, and lists a range of areas for further research which build on the approaches introduced in the earlier parts of our work.

Chapter 2

Background

This chapter introduces background terminology and concepts used throughout the rest of the thesis, and therefore required to understand the proposed methods. We start with a brief introduction to machine learning in Section 2.1, emphasising supervised learning and in particular deep neural networks applied to image classification in Section 2.1.1 and then extended to the more complex task of object detection in Section 2.1.2. Next, we introduce the reader to risk assessment in Section 2.2, which is required to understand the core ideas from Chapters 3 and 4. We then succinctly present genetic algorithms in Section 2.3, where key concepts and definitions are addressed. Finally, in Section 2.4, ensemble learning is introduced, and the most representative ensemble methods are described.

2.1 Machine learning

ML is a subfield of artificial intelligence (AI) and represents the scientific study of algorithms and statistical models that computer systems use to perform a specific task without being explicitly programmed [13, 97]. Normally, an ML model is presented with many examples relevant to a task and it finds statistical structure in these examples that eventually allows the system to come up with rules for automating tasks. On the basis of algorithm procedure, ML techniques can be classified into four types: (i) supervised ML techniques, (ii) unsupervised ML techniques, (iii) semi-supervised ML techniques and (iv) reinforcement ML techniques [38, 66].

Supervised ML techniques build on knowledge gained from labelled data samples in order to forecast future events. These techniques employ a training process that exploits the labelled dataset, in order to infer a function that produces output values for new data samples. To that end, the training algorithm compares the results produced by the ML model under development to the actual (i.e., expected) results in order to identify errors and to change the model based on these results. After training on the labelled data sets, the ML models are expected to generalise to unseen data from the real world [126].

Unsupervised ML techniques are employed when the training data is non-classified and not labelled. They analyse how the system can deduce a function to explain the hidden patterns from the unlabelled data. The system does not identify the proper output, but it discovers the data and writes observations from the dataset to find hidden patterns from unlabelled data [11].

Semi-supervised ML techniques lie between supervised and unsupervised ML

techniques. This type of ML uses labelled and unlabelled data during the training process. Generally, it considers a smaller quantity of labelled data and a larger quantity of unlabelled data. These techniques can adjust themselves to attain higher accuracy and are preferable in cases where acquiring labelled data requires skilful and appropriate resources to train or learn from them. In contrast, obtaining unlabelled data does not need the extra resources [126].

Finally, reinforcement learning techniques interact with the environment by actions, and adapt their learning based on errors or rewards received as a result of this interaction. Trial and error search and delayed rewards are some of the common features of the reinforcement method. Reinforcement learning enables systems and software programs to identify the ideal behaviour in a specific context, e.g. in order to increase the performance of a process comprising multiple activities or of a navigation path comprising multiple segments [84].

In this thesis, we focus on supervised ML, in particular on neural networks, which will be described in detail below. Supervised ML requires three key ingredients [67]:

1. Input data samples, for instance, pictures.

2. Examples of the expected output for the input data samples. In an image classification task, the expected output could be labels such as *dog* or *cat*.

3. A way to evaluate the model produced by the ML algorithm. This is necessary to determine the "distance" between the model's current output and its expected output, e.g. the *loss function*.

2.1.1 Image classification

Image classification is the process of allocating a single label, from a set of possible labels, to an image. This is a fundamental problem in computer vision, where it forms the basis of localisation, detection, and segmentation [24, 116].

Recently, DNNs have proven to be particularly effective at image classification tasks. In addition to the performance influenced by the network structure itself, the data set is also one of the influencing factors that cannot be ignored. The data provided to the algorithm is crucial in image classification, especially for supervised classification. The picture dataset feeds the DNN, and the better the quality of the data, the more accurate the model [94].

Examples of commonly used benchmark data sets for evaluating the performance of DNN image classifiers are IMageNet data set, GTSRB, CIFAR-10, CIFAR-100, and MNIST [131]. The data used in supervised ML are normally split into training, validation, and test sets. According to [121], these data sets have the following roles:

- *the training set* is a set of examples used for learning, that is to fit the parameters of the classifier or the data used to fit the model;

- *the validation set* contains a set of examples used to tune the parameters of a classifier, for example, to choose the number of hidden units in a neural network; in conclusion, it provides an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters;

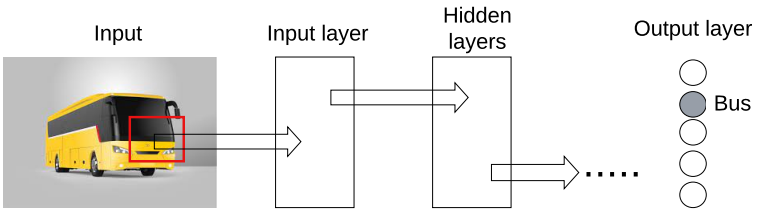


Figure 2.1: DNN architecture. The DNN has an input layer, an output layer and several hidden layers in between. The input image is decomposed into pixels and normalised to obtain values between 0 and 1 representing each such pixel, and all pixel representations are forwarded to the neural network's input layer.

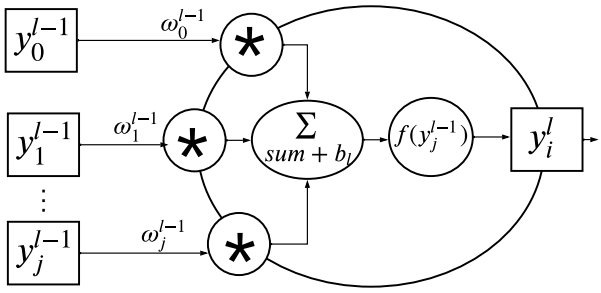


Figure 2.2: Representation of a neuron depicting its inputs, computation and output

- *the test set* is a set of examples used only to assess the performance of a fully-trained classifier, it provides an unbiased evaluation of a final model fit on the training dataset.

As shown in Figure 2.1, a DNN is a structure constructed as layers stacked on top of each other. The input image is decomposed into pixels and normalised to obtain a single number between 0 and 1 for each such pixel, and the pixel representations are presented to the neural network's input layer. The DNN has an input layer, output layer and various hidden layers in between (the term deep refers to how many hidden layers contribute to the network). Each layer contains a set of neurons which apply an activation function to a weighted sum of outputs from the previous layer. A DNN learns to perform particular tasks through training, during which the strength of connections or weights between units is learned. Subsequently, the trained DNN is expected to correctly classify samples which were not used during the training, i.e., the model is expected to generalise to samples from the real world [28].

The output y_i^l of a neuron i in layer l is a function of the outputs of the earlier layer such that

$$y_i^l = f\left(\sum_{j=1}^{n_{l-1}} w_j^{l-1} y_j^{l-1} + b_l\right) \tag{2.1}$$

where n_{l-1} is the number of neurons in layer $l-1$, y_j^{l-1} is the output of the j^{th} neuron in layer $l-1$, w_j^{l-1} is a weight associated with the output, b_l is a bias term for layer l, and f is the activation function. An example of an artificial neuron is shown in Figure 2.2, the artificial neuron takes several inputs (these inputs are for instance pixels from an image represented in values ranging from 0 to 1), sums them

together and finally applies the activation function to obtain the output signal. The summation of the inputs is done in a weighted way. The specification of what a layer does to its input data is stored in the layer's *weights*. Initially, the *weights* are randomly assigned to the network. Another parameter also trained and used in the summation process is the neuron's bias b. Its value is added to the weighted sum as an offset and makes the model more flexible to better model the given data.

An activation function could be for instance a binary function such that if y is above a threshold the next neuron will be activated, that is $y = 1$ or 0 in other cases. Formally, this activation function can be expressed as:

$$y = f(x) = \begin{cases} 0, & \text{if } x < t \\ 1, & \text{otherwise} \end{cases} \tag{2.2}$$

The step function is a simple example of an activation function; however, advanced functions which contain properties such as non-linearity, to model complex behaviours, have been introduced [112]. Examples of commonly used activation functions include the *sigmoid* function

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{2.3}$$

the *hyperbolic tangent*

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}, \tag{2.4}$$

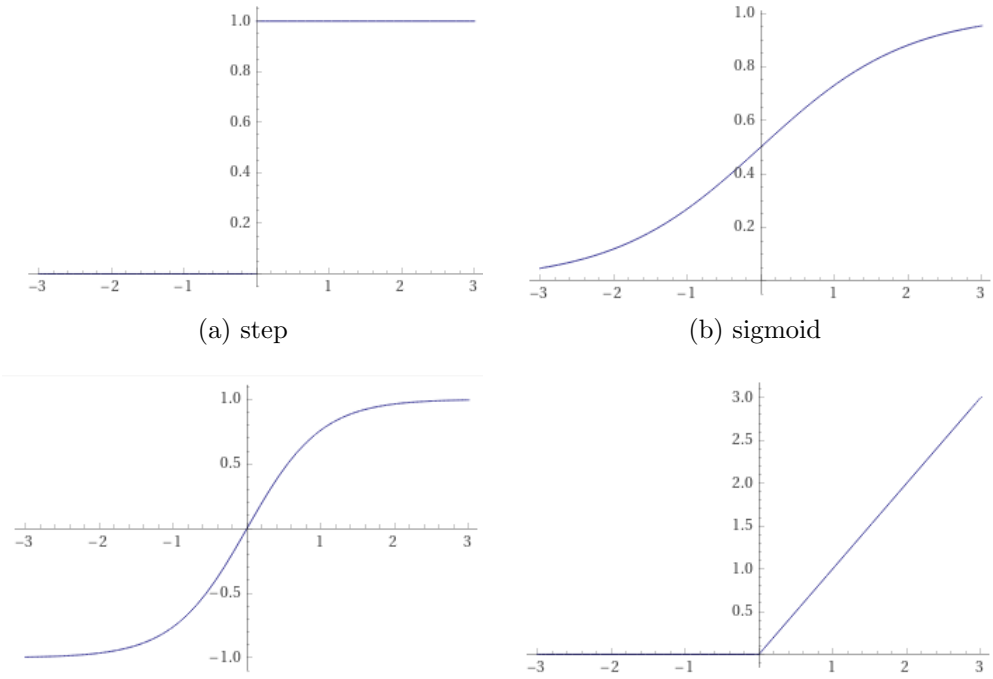
and the *Rectified Linear Unit (ReLU)*

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{cases} \tag{2.5}$$

A graphic representation of these activation functions is shown in Figure 2.3. As mentioned before, different activation functions can be applied to different problems, with the non-linearity in the activation being useful to model complex problems where the data can not be modelled as a linear function. Each of the listed functions has its own advantages and disadvantages; however, given its computational simplicity, the ReLu function is the most popular choice of activation function for hidden layers [112].

In order to adjust the *weights* so that the DNN correctly makes predictions, a *loss function* or *objective function* is used to compute a distance score between the true target and the prediction of the network, as shown in Figure 2.4. This score (error) is used as a feedback signal, with the weights adjusted in a direction that will lower the loss score. The adjustment is done by an *optimiser*, the mechanism that allows the network to update itself based on its loss function. This process is repeated many times for each of the instances of the data.

According to [156] common loss functions in classification are (i) the perceptron loss function, (ii) binary cross-entropy (logarithmic loss), (iii) Sigmoid cross-entropy loss, (iv) softmax cross-entropy loss, (v) hinge loss, and (vi) ramp loss. Below we briefly describe each of these loss functions, and Table 2.1 provides their definitions.



(a) step (b) sigmoid

(c) tanh (d) ReLU

Figure 2.3: Graphic representation of commonly used activation functions.

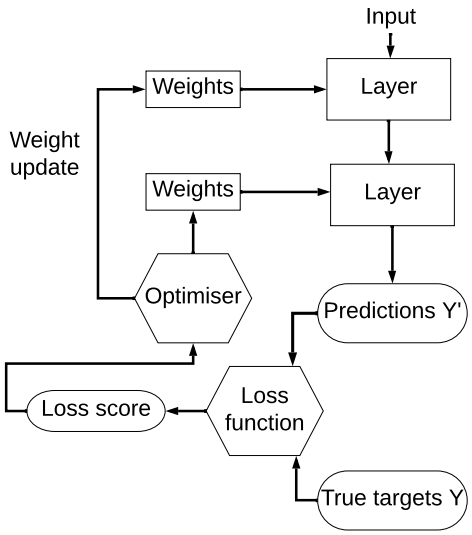


Figure 2.4: The loss score is the measure of the distance between the current prediction and the true target and is then used as a feedback signal to adjust the weights of the DNN (taken from [27]).

(i) Perception loss function. This loss function is a piecewise function. When the predicted value of the sample has the same sign as the real label, the loss value is 0; otherwise, the loss value is the absolute value of the predicted value. From the perspective of geometric meaning, the former means that there is no loss for correctly classified samples, while the latter measures the distance from the predicted samples to the decision boundary $f(x) = 0$, and the larger

Table 2.1: Commonly used loss functions in classification (adapted from [156]).

Name	Formula	Explanation	Algorithm
Perceptron loss	$\mathcal{L}(y, f(x)) = max\{0, -yf(x)\}$	$max(a,b) = \begin{cases} a \; if \; a \geq b \\ b \; if \; a < b \end{cases}$	Perceptron. The obj func is $\mathcal{L} = -\sum_{x_i \in M} y_i(w^T x_i + b)$, where, M is the misclassified sample set.
Binary Cross-Entropy (Logarithmic loss)	$\mathcal{L}(y, \tilde{p}) = -log\, \tilde{p}$, where $\tilde{p} = \begin{cases} p, \; if \; y = 1 \\ 1-p, \; if \; y \neq 1 \end{cases}$	Conditional prob dist: $p = P(y=1\|x) = \frac{1}{1+e^{-f(x)}}$, $1-p = P(y=-1\|x) = \frac{1}{1+e^{f(x)}}$	Logistic regression (LR). The obj func is $\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} log\frac{1}{1+e^{-y_i(w^T x_i+b)}}$
Sigmoid cross entropy loss	$\mathcal{L}(y, \tilde{p}) = -log\, \tilde{p}$, where $\tilde{p} = \begin{cases} p, \; if \; y = 1 \\ 1-p, \; if \; y \neq 1 \end{cases}$	$p = \sigma(f(x)) = \frac{1}{1+e^{-f(x)}}$, where $\sigma(\cdot)$ is the Sigmoid function	Neural network (NN). The obj func is the same as LR.
Softmax cross entropy loss	$\mathcal{L}(y, P(y\|x)) = -logP(y\|x)$	$P(y\|x) = \frac{e^{f_y(x)}}{\sum_k e^{f_k(x)}}$, where $f_y(x)$ and $f_k(x)$ are the decision funcs corresponding to the y and k class respectively.	Neural network (NN). The objective function is $\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{w_{y_i}^T x_i + b_{y_i}}}{\sum_k e^{(w_k^T x_i + b_k)}}$
Hinge loss	$H_s(z) = max\{0, s-z\}$, where, s is a constant	Specially, in SVM $\mathcal{L}(y, f(x)) = H_1(yf(x)) = max\{0, 1-yf(x)\}$	Support vector machine (SVM) The objective function is $\mathcal{L} = \frac{1}{2} \parallel w \parallel^2 + C\sum_{i=1}^{N} max\{0, 1-y_i(w^T x_i + b)\}$
Ramp loss	$R_s(z) = H_1(z) - H_s(z)$, where, $s < 1$ is a constant	Specially, in Ramp loss SVM $\mathcal{L}_r(y, f(x)) = H_1(yf(x) - H_s(yf(x)))$	Ramp loss SVM. The objective function is $\mathcal{L} = \frac{1}{2} \parallel w \parallel^2 + C\sum_{i=1}^{N} \mathcal{L}_r(y_i, f(x_i))$

the distance, the greater the error. It is a continuous and differentiable loss function with respect to variables so that it is easy to optimize. However, its goal is only to determine the sample category correctly, that is, the sample meets the requirement when it is located in the decision boundary. In this way, the model obtained has poor generalisation performance and is not robust to noise data [101].

(ii) Binary cross-entropy (logarithmic loss). This loss function is a function of the sample prediction probability value, where the prediction probability value is obtained through the conditional probability distribution (detailed in Table 2.1). Specifically, the greater the probability of the sample being predicted as its label, the smaller the corresponding loss value; otherwise, the greater the loss value. In the actual calculation, the probability of being predicted as a positive class is expressed as p, while the other probability is $1-p$ [156].

(iii) Sigmoid cross-entropy loss. The Sigmoid cross-entropy loss and logarithmic loss are the same. The reason why the same losses are described by two names is that they are defined from different sources. The Sigmoid cross-entropy loss is obtained by seeing the probability value converted from the predicted value through the Sigmoid activation function as the actual output of cross-entropy, and the real label of the sample as the expected output of cross-entropy. Cross-entropy describes the distance between the actual output and the expected output. The smaller the value of cross-entropy is, the closer the two probability distributions are.

(iv) Softmax cross-entropy loss. The only difference between this loss function and the Sigmoid cross-entropy loss function is that this loss function replaces the Sigmoid function with the softmax function to solve the multi-classification

problem [73].

(v) Hinge loss. In the binary classification problem, the hinge loss function defines the margin (represented by the parameter s) near the decision boundary, and the correctly classified samples in the middle of the two margin boundaries and all misclassified samples have the cost. The hinge loss function does not punish the correct classification samples of $|f(x)| \geq s$. It is considered that such samples have been learned well enough so that the model is more focused on the overall classification error. It should be noted that s is generally set to 1 in applications and improvements of hinge loss function [37].

(vi) Ramp loss. This is an improved version of the hinge loss function. The hinge loss value of outlier is very large and outliers play a leading role in determining the decision boundary so that the model will reduce the accuracy of normal samples to reduce such loss, and finally reduce the overall classification accuracy, resulting in low generalization ability of the model. However, the ramp loss function limits the maximum loss value, which limits the influence of outliers to some extent, and the model is more robust to outliers. In addition, when the ramp loss function is applied to SVM, the number of support vectors can be reduced and the training efficiency can be improved [156].

Before describing how the performance of an image classifier is evaluated, is important to mention that during the training of a DNN model *underfitting* and *overfitting* should be considered. For instance, if a DNN model has too few layers, or the layers are too small, the accuracy of the model may stagnate. This is, the DNN is *underfitting*, meaning that it does not have enough parameters for the complexity of the task. The only solution to adopt, should this case happen, is to adopt a new architecture that is better suited for the intended application. On the other hand, if the DNN architecture is excessively complex or the training data set is overly small, the network may start *overfitting* the training data. This means that the network will learn to fit very well the training distribution data; nevertheless, it will not generalise to new samples. One possible solution to this problem might be gathering a larger, more diverse training set, although this is not always possible in practice (e.g., due to limited access to the target objects). Another solution is to adapt the network or its training in order to constrain how much detail the DNN learns [112].

Now, consider an n-sample data set $D = \{d_1, d_2, ..., d_n\}$ and a set of classes $C = \{c_1, c_2, ...c_s\}$. Image classification involves assigning a class $c_i \in C$ to each data sample $d_k \in D$. Let $Y = \{y_1, y_2, ..., y_n\}$ be the set of actual classes (ground truth) corresponding to the data set D, where y_k is the actual class of d_k. And let $Y' = \{y'_1, y'_2, ..., y'_n\}$ be the set of predictions made by a DNN model M for each element in D where y'_k is the prediction for the element d_k. The performance of M can be assessed using a measuring function Φ, which assigns a metric $\phi \in \mathbb{R}$ to the pair (Y, Y'), that is, $(Y, Y') \xrightarrow{\Phi} \phi$ [95].

There are various ways of assessing the effectiveness of an ML model such as a DNN. The first information that can be obtained once a model has been tested is the confusion matrix, a performance measurement for machine learning classification based on the testing data set. When the number of classes is $s = 2$, and one of the classes is called *positive* and the other *negative*, the confusion matrix can be written

as

$$CM = \begin{bmatrix} TruePositives\ (TP) & FalseNegatives\ (FN) \\ FalsePositives\ (FP) & TrueNegatives\ (TN) \end{bmatrix} \tag{2.6}$$

This can be extended to multi-class in the following way. Think about a data set $D = \{d_1, d_2, ..., d_n\}$ that contains n samples, where d_k is the $k-th$ element in D. Consider $C = \{c_1, c_2, ...c_s\}$ a set of classes where c_i represents the $i-th$ class. A DNN model M that is given the d_k sample predicts the label c_j when in reality it belongs to the c_i class thereby causing a misclassification. Let $Y = \{y_1, y_2, ..., y_n\}$ be the set of actual classes (ground truth) corresponding to the data set D, where y_k is the actual class of d_k. And let $Y' = \{y'_1, y'_2, ..., y'_n\}$ be the set of predictions made by M for each element in D where y'_k is the prediction for the d_k element. According to [95], the performance of M can be assessed using a measuring function Φ, which assigns a metric $\phi \in \mathbb{R}$ to the pair (Y, Y'), that is, $(Y, Y') \xrightarrow{\Phi} \phi$. A multi-class confusion matrix is defined as:

$$CM = \begin{bmatrix} m_{11} & m_{12} & ... & m_{1s} \\ m_{21} & m_{22} & ... & m_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ m_{s1} & m_{s2} & ... & m_{ss} \end{bmatrix} \tag{2.7}$$

where m_{ij} represents the number of elements actually belonging to the i-th class (c_i) but that are classified as members of the j-th class (c_j) (except for the values on the diagonal).

Based on the binary confusion matrix (2.6), numerous performance metrics have been proposed [56, 69, 100, 114, 137] and are summarised in Table 2.2. All these metrics take values between 0 and 1, except for Matthews Correlation Coefficient (MCC), Bookmaker Informedness (BM), and Markedness (MK), whose ranges lie between -1 and 1. From all these metrics, the precision, recall (sensitivity) and F1 score are commonly used for classification [137].

Recall or sensitivity is the proportion of real positive cases that are correctly predicted. This measures the coverage of the real positive cases. Concisely, recall measures the effectiveness of a classifier to identify positive labels. Its desirable feature is that it reflects how many of the relevant cases were picked up. It tends to be neglected when the focus is on how confident one can be in the rule or classifier. However, in a computational linguistics/machine translation context and in a medical context, recall is regarded as primary, as the aim is to identify all real positive cases [47, 114].

Precision denotes the proportion of predicted positive cases that are correctly real positives. This is what ML, data mining and information retrieval focus on, it can also be seen as a measure of the accuracy of predicted positives or class agreement of the data labels with the positive labels given by the classifier [114].

The F1 measure is a combination of the above (precision and recall) and is widely used in most application areas of machine learning, not only in the binary scenario but also in multi-class cases. Is defined as the harmonic mean of precision and recall [137, 25]. In multi-class cases, the F1 micro/macro averaging can be employed.

Table 2.2: Performance metrics for a DNN based on the binary confusion matrix (taken from [95]).

Symbol	Metric	Defined as
SNS	Sensitivity or Recall	$\frac{TP}{TP+FN}$
SPC	Specificity	$\frac{TN}{TN+FP}$
PRC	Precision	$\frac{TP}{TP+FP}$
NPV	Negative Predictive Value	$\frac{TN}{TN+FN}$
ACC	Accuracy	$\frac{TP+TN}{TP+FN+TN+FP}$
F1	F1 score	$2\frac{PRC \cdot SNS}{PRC+SNS}$
GM	Geometric Mean	$\sqrt{SNC \cdot SPC}$
MCC	Matthews Correlation Coefficient	$\frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$
BM	Bookmaker Informedness	$SNS + SPC - 1$
MK	Markedness	$PPV + NPV - 1$

The macro-averaged F1 is computed on a set of instances; it is defined as the average of the single label F1 measures computed for each label and gives the same weight to each label:

$$F_\beta^M = \sum_{i=1}^{m} \frac{(1+\beta^2)TP_i}{(1+\beta^2)TP_i + \beta^2 FN_i + FP_i} \tag{2.8}$$

The micro-averaged F1 is computed after pooling the labels of all instances of a given set, and gives equal weight to each labelling decision:

$$F_\beta^m = \frac{\sum_{i=1}^{m}(1+\beta^2)TP_i}{\sum_{i-1}^{m}[(1+\beta^2)TP_i + \beta^2 FN_i + FP_i]} \tag{2.9}$$

where $\beta \in [0, +\infty)$ controls the trade-off between precision and recall and m is the number of labels [81, 111, 151].

2.1.2 Object detection

In the previous section, we provided an introduction to image classification, whose main objective is to simply label a given image. In this section, we review object detection, a much more complex task that deals not only with classification but also with the location of objects in a scene, which is closer to real-world applications where normally, we need to differentiate between many objects to make sense of our environment. Formally, object detection [22, 112, 170] is defined as a function that maps an image to a list of objects $O = (o_1, o_2, \ldots, o_n)$, where the i-th detected object $o_i = (c_i, box_i)$ specifies:

- (estimate) probabilities $c_i = (c_{i1}, c_{i2}, \ldots, c_{iN})$ that object i belongs to each of N classes of interest, $\sum_{j=1}^{N} c_{ij} = 1$;

- a *bounding box* (box_i) comprising either the coordinates of the top-left and bottom-right corners of the image region where object i is located or the coordinates of the centre of the image in addition to its width and height measures.

The state-of-the-art ML models for object detection can be divided into two categories: two-stage and one-stage detectors. In general terms, two-stage detectors tend to obtain higher accuracy, but with a higher computational cost than one-stage detectors. However, this fact highly depends on the selected convolutional backbone network and the hyperparameter configuration, which is a complex procedure [21].

Two-stage frameworks divide the detection process into the region proposal and the classification stage. These models first propose several object candidates, known as regions of interest (RoI), using reference boxes (anchors). In the second step, the proposals are classified and their localisation is refined [170]. The region proposal-based methods mainly include R-CNN [53], spatial pyramid pooling (SPP)-net [61], Fast R-CNN [52], Faster R-CNN [120], region-based fully convolutional network (R-FCN) [31], feature pyramid networks (FPN) [86], and Mask R-CNN [60].

On the other hand, one-stage detectors contain a single feed-forward fully convolutional network that directly provides the bounding boxes and the object classification. One-stage detectors regard object detection as a regression or classification problem, adopting a unified framework to achieve final results (categories and locations) directly. The one-stage methods mainly include MultiBox [40], AttentionNet [164], G-CNN [104], YOLO [117], Single Shot MultiBox Detector (SSD) [91], YOLOv2 [118], YOLOv3 [119], deconvolutional single shot detector (DSSD) [49], RetinaNet [88], and deeply supervised object detectors (DSOD) [132].

The one and two stage algorithms owe their name to the sub steps taken in the detection process. Two-stage frameworks divide the detection process into the region proposal and the classification stage and the one-stage algorithms perform both tasks simultaneously. Regardless of the stages in the detection step the pipeline of object detection models can be mainly divided into three steps [170]. The pipeline is depicted in Figure 2.5 and each of the steps is detailed below:

1. In a first step, a feature extraction neural network (NN) is used to detect potential objects in the input image. Feature extraction is an important component of every image classification and object recognition system. It consists on mapping the image pixels into the feature space. For automatic identification of the objects from remote sensing data, they are to be associated with certain attributes which characterise them and differentiate them from each other. The similarity between images can be determined through features which are represented as a vector. Feature extraction is concerned with the extraction of various attributes of an object and thus associates that object with a feature vector that characterises it. It is the first step to classifying an image and identifying the objects. The characteristics of an image such as colour, texture, shape etc. are used to represent and index an image or an object [149].

2. In a second, *detection* step, *anchor bounding boxes* (i.e., approximate bounding boxes drawn from a set of predefined box sizes) are fitted around these objects, and then adjusted appropriately, this adjustment is represented in Figure 2.6,

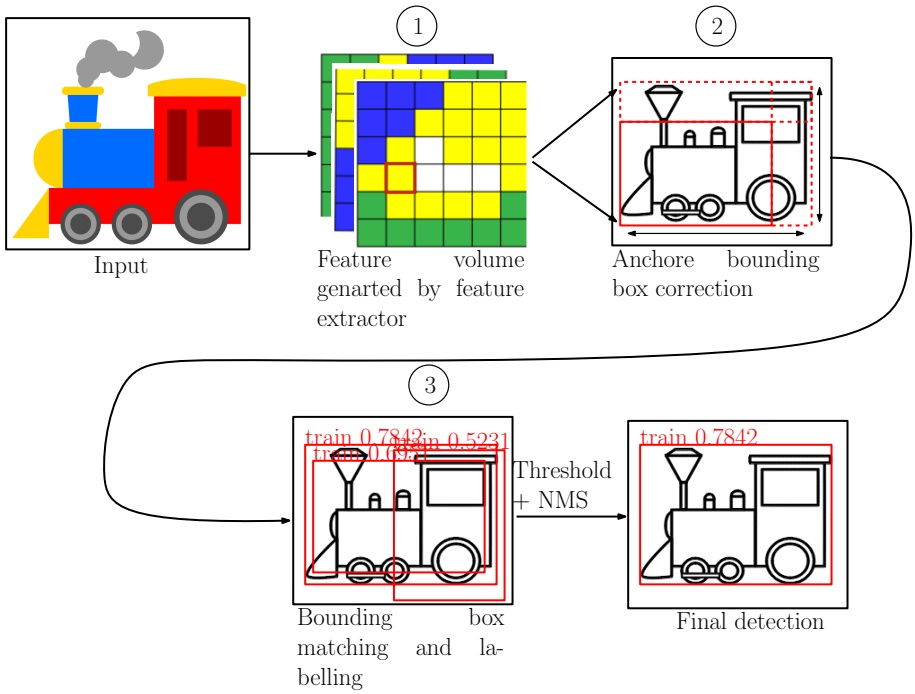


Figure 2.5: The object detection pipeline can be summarised in three stages: 1) Feature extraction; 2) Detection and anchor bounding box correction; 3) Bounding box matching and labelling.

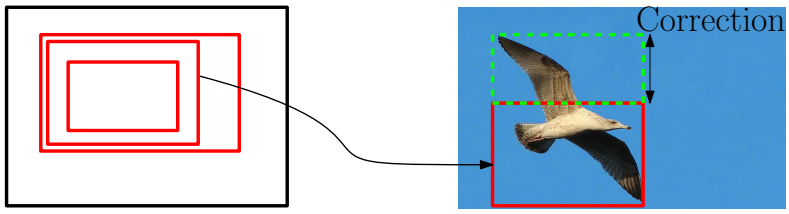


Figure 2.6: On the left is the set of predefined anchor bounding boxes with different sizes picked to detect an object. On the right is the adjustment of an anchor box to match the detection(the picture was taken from [41]).

where from a set of predefined anchor boxes the one that better fits the current detection is picked to latter adjust it accordingly, the correction is computed by the neural network. Next, an NN classifier is used to estimate the probabilities that the object within each bounding box belongs to the N classes of interest, as well as a confidence measure that an object is actually present in each of these boxes.

3. In the final step, *bounding box matching and labelling* is performed. To start with, irrelevant and duplicate boxes are eliminated. A box is deemed irrelevant if the product between its confidence measure and maximum class probability is below a predefined "relevance" threshold. To identify duplicates, a *non-maximum suppression (NMS)* algorithm processes the remaining boxes in decreasing probability order, i.e., starting with the box i with the highest class probability $c_{i,j}$, and eliminating all unprocessed boxes that overlap significantly with box i. The overlap between two boxes is deemed significant if their *intersection over union* (IoU) measure (i.e., the ratio between their

Figure 2.7: True positive in the object detection context. The detection meets two conditions: detection and location (the picture was taken from [41]).

intersection and their union) exceeds a predefined threshold. Each remaining box is then labelled with the name corresponding to its maximum class probability.

The NNs employed by ML-based RTOD solutions are trained and tested using large sets of image samples annotated with both the bounding boxes and the class labels for all relevant objects in each image. For detailed descriptions of ML-based RTOD, we refer the reader to [117, 119, 91, 88].

In order to evaluate the performance of object detection algorithms, some metrics have been proposed. We start by defining the components of the confusion matrix which is described by the true positives (TP), the false negatives (FN), and the false positives (FP), they were already described in Section 2.1.1. However, the definition of these concepts in the detection context is different:

- A true positive is a prediction that matches a ground-truth bounding box of the same class. It also can be defined as the elements correctly classified. The detection must satisfy two conditions: (i) the confidence score of the predicted bounding box should be greater than the confidence threshold and (ii) the IoU (described below) between the predicted bounding box and the ground-truth bounding box should be greater than the IoU threshold. Figure 2.7 shows an example of a TP detection with an IoU of 0.81 and a Confidence of 0.7.

- A false positive represents a prediction for which: (i) an incorrect detection of a non-existent object is made with high confidence; (ii) the IoU is below the threshold; or (iii) the bounding box correctly fits the predicted object, but the class label of the prediction is incorrect. Examples of false positives are depicted (as yellow boxes) in Figure 2.8.

- A false negative is a prediction for which there is an object in the ground truth, yet the model was unable to detect it; an example is shown in Figure 2.9, where only the ground-truth box is present.

Note that, in the object detection context, the true negative result does not apply, as there is an infinite number of bounding boxes that should not be detected within any given image [107].

Then we have the IoU, which is a measurement based on the Jaccard Index, a coefficient of similarity for two sets of data. In the object detection field, the

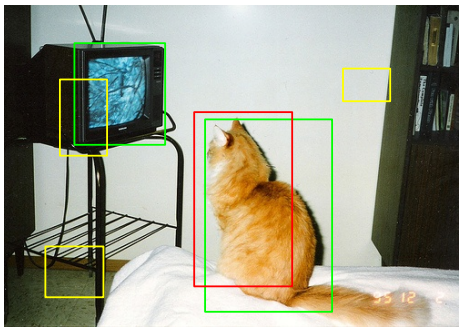


Figure 2.8: Example of true positive (red box), false positives (yellow boxes) and ground truth (green box). (the picture was taken from [41]).

Figure 2.9: Examples of a false negative detection where only the ground truth is present (the picture was taken from [41]).

IoU measures the overlapping area between the predicted bounding box B_p and the ground-truth bounding box B_{gt} divided by the area of union between them (see also Figure 2.10):

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{2.10}$$

The IoU is used to determine if a detection is considered correct or incorrect by comparing it with a threshold, for example, if $IoU \geq t$ then detection is correct. If $IoU < t$ then the detection is incorrect.

The precision $\times$ recall curve, which is shown in Figure 2.11, can be seen as a trade-off between precision and recall for different confidence values associated with the bounding boxes generated by a detector. If the confidence is close to 1,

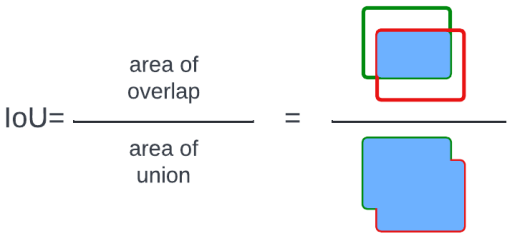


Figure 2.10: Calculation of the intersection over union (IoU) (taken from [107]).

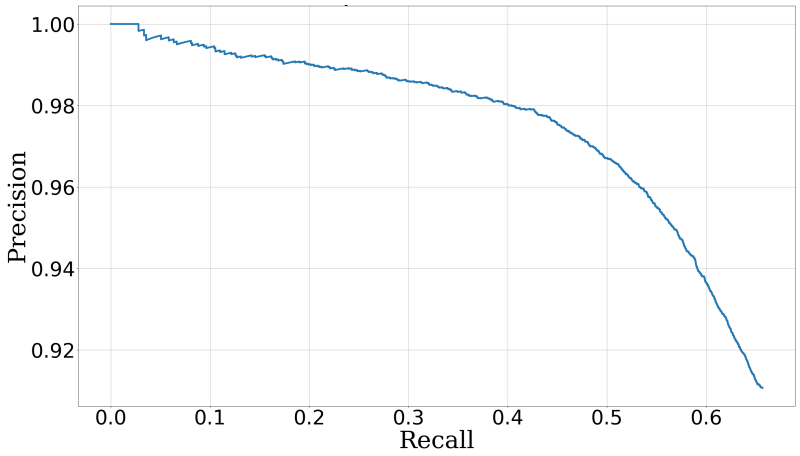


Figure 2.11: Precision-Recall curve plot.

the precision will be high, but the recall will be low. This is undesirable because although the detections made will be mostly correct, most of the objects will be missed. As only confident predictions are kept, the recall shrinks and the precision raises. On the other hand, when the confidence is close to 0, the precision will be low, but the recall will be high. Most of the predictions are kept, which rises the recall; however, because the model is less confident in its predictions, the precision shrinks [112]. An object detector is considered good if its precision stays high as its recall increases, which means that if the confidence threshold varies, the precision and recall will still be high. Therefore, a high area under the precision × recall curve (AUC) tends to indicate both high precision and high recall. As described in [107], in practical cases, the precision × recall plot is often a zigzag-like curve and this difficult the accurate measure of the AUC. In order to remove the zigzag behaviour [42] proposes the use of the 11-point interpolation where the shape of the precision × recall curve is summarized by averaging the maximum precision values at a set of 11 equally spaced recall levels [0,0.1, 0.2, ... , 1] and then averages them out. Mathematically, this can be expressed as:

$$AP_{11} = \frac{1}{11} \sum_{R \in \{0,0.1,0.2,\ldots,1\}} P_{interp}(R), \tag{2.11}$$

where

$$P_{interp}(R) = \max_{\widetilde{R}:\widetilde{R} \geq R} P(\widetilde{R}) \tag{2.12}$$

and $P(\widetilde{R})$ represents the measured precision at recall $\widetilde{R}$.

The mean average precision (mAP) is the most common performance metric for object detection and is widely accepted by the research community. Many object detection algorithms, such as Faster R-CNN [120], MobileNet SSD [91] and YOLO [117, 119] use mAP to evaluate their models. The mAP is also used across several benchmark challenges such as the Pascal VOC [42], Microsoft COCO: Common Objects in Context [87] and ImageNet Object Detection Challenge [123]. Average precision (AP) is calculated by estimating the area under the curve of the precision × recall relationship, mAP is the average of AP of each class. The mAP formula requires the calculation of other sub-metrics such as confusion matrix, IoU, precision and recall [107].

The mAP measures the accuracy of object detectors over all classes in a specific dataset. The mAP[1] is simply the average AP over all classes [91], [120], that is

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i, \tag{2.13}$$

with AP_i being the AP in the ith class and N the total number of classes.

2.2 Risk assessment

Risk is defined as the possibility of something bad happening [115]. Risk is normally described in terms of risk sources, potential events leading to the risk occuring, their consequences and their likelihoods (i.e., the chances of the events happening). An event can have multiple causes and lead to multiple consequences. The seriousness of consequences can be estimated using metrics that can take a number of discrete values, be continuous variables or be unknown. Consequences may not be discernible or measurable at first, but may accumulate over time. Sources of risk can include inherent variability or uncertainties related to factors including human behaviour and organizational structures or societal influences for which it can be difficult to predict any particular event that might occur [44].

The ISO 31000 international standard [44] proposes a risk assessment process that involves identifying risks, analysing them, and using the understanding gained from the analysis to evaluate risk. The ultimate aim of this process is to establish: whether the risk needs to be addressed; the priorities for treatment of risks; and the actions needed to treat the risks.

For recording and reporting information about the magnitude of risk, various techniques have been proposed in the ISO 31010 standard. The purposes of recording the risk include communicating information about risk to decision makers and regulators; providing a record and justification of the motive for decisions made; preserving the results of assessment for future use and reference; tracking performance and trends; provide confidence that risks are understood and are being managed appropriately; enable verification of the assessment and to provide an audit trail.

Among other methods for risk assessment, [44] suggests the use of a consequence/likelihood matrix (also referred to as a risk matrix or heat map) for recording and reporting information about the magnitude of a risk. An example of a such a risk matrix is shown in Table 2.3. The approaches to developing risk-aware ML ensembles proposed in this thesis adopt this risk-assessment method.

The risk matrix allows to report information about the likelihood, consequence or impact and level of risk, indicated by the position in the matrix. Additionally, it displays a rating for the significance of risk. The axes of the matrix feature the scales for consequence (impact) and likelihood and the scales can have any number of points, five-point scales being the most common. For instance, in Table 2.3 the consequence is represented on a five-points scale ranging from a–e with a being the highest consequence and e being the least severe consequence. The consequence scale can depict positive or negative consequences and the scales should extend from the maximum credible consequence to the lowest consequence of interest. For instance,

[1]For a practical example of the mAP calculation the reader is referred to [107]

Table 2.3: Example of a consequence/likelihood matrix. The consequence rating is given on a five-point scale of a–e with a being the most significant consequence and e the less significant. The likelihood is similarly represented on a five-point scale of 1–5, where 5 corresponds to 'very likely' and 1 to 'remotely likely' events. The colours from the cells fade from red (I) to goldenrod (V), indicating the priority of the risk (adapted from [44]).

		Likelihood rating				
		1	2	3	4	5
Consequence rating	a	III	III	II	I	I
	b	IV	III	III	II	I
	c	V	IV	III	II	I
	d	V	V	IV	III	II
	e	V	V	IV	III	II

ranging from multiple fatalities to first aid only required, for a health and safety context. The rating can be given in words, numbers or letters.

The likelihood scale should span the range relevant to data for the risks to be rated. For example, ranging from very likely to remotely possible and can be given as words, numerals or letters, if we refer to our example Table 2.3, the likelihood rating is given on a 1–5 scale with 5 being the highest likelihood and 1 the lowest.

As already mentioned, a risk matrix is assembled with the consequence on one axis and the likelihood on the other. A rating for risk *level* is often associated wtih each cell. Typically, boxes are coloured to indicate the magnitude of risk and decision rules can be linked to them. If we look again at Table 2.3, five colours are used to identify the different risk levels. The colours fade from red to goldenrod indicating the priority of the risk. The highest priority is for the risk cells shaded in red and marked with an 'I' and the risks with the lowest priority are shaded in goldenrod and marked with an 'V'. The matrix can be set up to give extra weight to consequences or to likelihood, or it can be symmetrical, depending on each specific application. A consequence/likelihood matrix is used to evaluate and communicate the relative magnitude of risks on the basis of a consequence/likelihood pair that is typically associated with a focal event. To rate a risk, the user finds the consequence (impact) and then the likelihood at which it is believed to occur. A point is placed in the cell that combines these values and the level of risk is read off from the matrix.

The consequence/likelihood matrix has several limitations. In particular, significant expertise is required to design a valid matrix, as proposing suitable likelihood and consequence ratings needs a deep understanding of all aspects relevant to the assessed risk context; if the proposed ratings are based on biased interpretations, the obtained risks will mislead the decision-takers instead of providing useful insight. In addition, even when experts are dealing with the ratings, it can be difficult to define common scales that apply across a range of circumstances relevant to an organisation. This might limit the generalisation of a derived matrix. When talking about the scales, it is difficult to define them unambiguously to enable users to

weight consequence and likelihood consistently, and this too represents a limitation as there will always be the possibility that the proposed scales are not the most accurate, which can directly affect the validity of risk ratings because they depend on how well the scales were developed and calibrated.

Despite the drawbacks mentioned in the previous paragraph, the risk matrix has the advantage of being relatively easy to use and provides a rapid ranking of risks into different significance levels. This provides a clear visual display of the relevant significance of risks by consequence, likelihood, and level of risk. Furthermore, this allows an easy understanding of risks to stakeholders, with clear indication of which risks should be considered with urgency and which are less concerning. It also can help to decide which risks can be alleviated with the current mitigation strategies in place and which ones need further consideration. Finally, the risk matrix can be used to compare risks with different types of consequence and how the likelihood influence the risk level. For instance, there can be a risk with a high level of consequences but because its likelihood is very low, its significance remains low as well; nevertheless, if the conditions of the context change and the likelihood increases, what once was an insignificant risk could become a high priority one.

2.3 Genetic algorithms

Genetic algorithms (GAs) are a family of heuristic search algorithms inspired by the principles of evolution in nature. Genetic algorithms are useful both as search methods for solving problems and for modelling evolutionary systems. Due to the probabilistic development of the solution, the use of a GA does not guarantee optimality even when it may be reached. However, GAs are likely to yield solutions close to the global optimum. This probabilistic nature of the solution is also the reason they can overcome being "stuck" at local optima. GAs are particularly effective at tackling large (potentially huge) search spaces and navigating them, looking for optimal combinations of parameter values, architecture options, etc. [45, 161].

GAs are founded on the principles of the Darwinian evolution theory and their operation is underpinned by several key principles:

1. the principle of variation: the traits of individual specimens belonging to a population may vary;

2. the principle of inheritance: the traits are consistently passed on from specimens to their offspring;

3. the principle of selection: the specimens possessing traits that are better adapted to the environment will be more successful at surviving, and will also contribute more offspring to the next generation.

Genetic algorithms use a series of key concepts [161] that can be described as follows. First, a GA *genotype* is a collection of genes that are grouped into chromosomes. In GA, each individual is represented by a chromosome representing a collection of genes. As shown in Figure 2.12, a chromosome can be expressed as a binary string, where each bit represents a single gene.

A collection of individuals that represent candidate solutions for the problem to be solved, is termed a *population*. The population represents the current generation

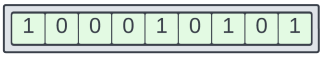


Figure 2.12: Example of a binary-coded chromosome.

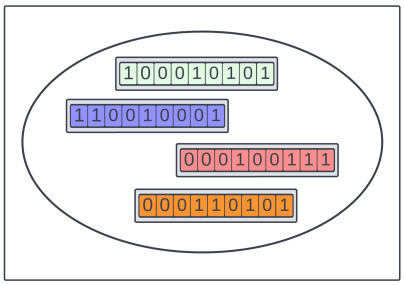


Figure 2.13: Population of binary-coded individuals.

of chromosomes, and evolves over time when the current generation is replaced by a new one. An example of a GA population is illustrated in Figure 2.13, where each individual is binary-coded and shaded with different colours to represent the various individuals of the population.

Genetic algorithms require a way to evaluate the quality of the individuals within a population. A *fitness function* (also known as the "target function") is used for this purpose. This is the function to be optimised or the problem to be solved by the GA. The fitness function evaluates individuals, where the higher the score the better the solution encoded by and individual, and the more likely for the individual to be chosen to reproduce and to be represented in the next generation. As generations pass by, the quality of the solution is expected to improve, and the fitness to increase, When an acceptable fitness function value is reached, after a predetermined number of generations, or when the solution is no longer improving after a predefined number of generations, the GA process stops.

A genetic algorithm consists of three main stages after the evaluation of an individual, i.e., a set of operations performed by the algorithm for each generation. The first stage is *selection*: this stage determines which of the individuals from the current population will be used to reproduce and create offspring for the next generation. The selection is based on the fitness score, with priority given to individuals with higher scores; however, low-scored individuals can still be chosen with lower probabilities, which helps to maintain diversity in future generations.

In the next stage, *crossover* is applied to create new individuals. Two parents are usually chosen from the current generation, and parts of their chromosomes are interchanged to create two new chromosomes representing the offspring. This process is illustrated in Figure 2.14

Lastly, *mutation* is applied in the last stage. The purpose of the mutation GA operator is to periodically and randomly refresh the population, introducing new patterns into the chromosomes, and encouraging search in uncharted areas of the solution space. As illustrated in Figure 2.15, mutation may appear as a random change in a gene, in this case flipping a bit in a binary string. However, when the solution space is continuous, i.e. the individuals are constructed of real (floating-point) numbers, this type of mutation is not longer valid. Specialised algorithms to deal with the crossover and mutation of real-coded chromosomes (i.e., blend

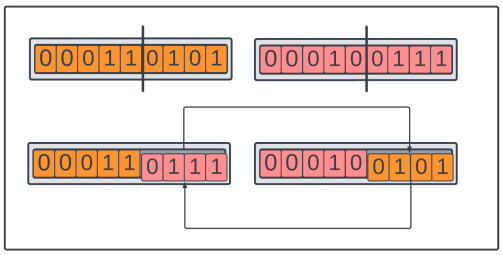


Figure 2.14: Crossover operation between two chromosomes.

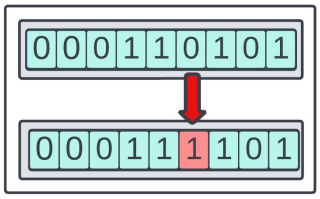


Figure 2.15: Mutation in an individual presented at randomly flipping a bit.

crossover, simulated binary crossover, and real mutation) are presented below:

- *Blend crossover* randomly selects each offspring from the following interval created by its parents $parent_1$ and $parent_2$:

$$[parent_1 - \alpha(parent_2 - parent_1),\ parent_2 + \alpha(parent_2 - parent_1)] \tag{2.14}$$

  where, α is a constant whose value lies between 0 and 1.

- *Simulated binary crossover* imitates the properties of the single-point crossover that is commonly used with binary-coded chromosomes. One of these properties is that the average of the parent's values is equal to that of the offspring's values. This technique uses the following formula:

$$\begin{aligned} offspring_1 &= \frac{1}{2}[(1+\beta)parent_1 \ + \ (1-\beta)parent_2] \\ offspring_2 &= \frac{1}{2}[(1-\beta)parent_1 \ + \ (1+\beta)parent_2] \end{aligned} \tag{2.15}$$

  where β is a random number known as the spread factor.

- *Real mutation* generates a random real number that resides in the vicinity of the original individual, e.g. by using normally distributed (or Gaussian) mutation where a random number is generated using a normal distribution with a mean value of zero and some predetermined standard deviation.

The stopping criteria for GAs can combine multiple conditions. The two most commonly used such conditions are:

1. A maximum number of generations has been reached;

2. Over the last generations of the GA, no noticeable improvement has been observed. This can be achieved by keeping the best fitness value obtained at each generation and comparing the current best value with a predefined one previously obtained, if the difference is below a given threshold the GA process can be finalised.

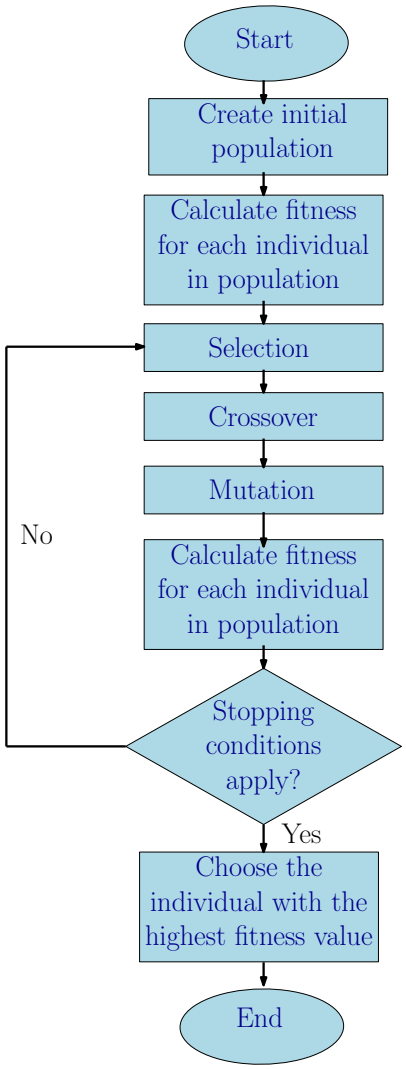


Figure 2.16: The main stages of the basic GA workflow.

Other stopping conditions can be: a predetermined amount of time has elapsed since the process began; a certain cost or budget has been consumed, such as CPU time and/or memory; the best solution has taken over a portion of the population that is larger than a preset threshold [110, 161].

To summarise, the genetic algorithm workflow, illustrated in Figure 2.16, starts with a random population generated using candidate solutions or individuals, which need to be evaluated using the fitness function. The core of the flow is a loop where the GA successively performs selection, crossover, and mutation, followed by a re-evaluation of the individuals. The loop continues until one of the stopping conditions applies, at this point, the best individual of the existing population is retrieved, and this is the final solution of the algorithm.

As mentioned in [161], genetic algorithms present the advantage of having global optimisation capability. In contrast, more naive search and optimisation algorithms are prone to get stuck in a local maximum rather than finding the global one—because, in the vicinity of a local maximum, any small change will degrade the score. An example of this situation is depicted in Figure 2.17. Genetic algorithms, on the other hand, are less sensitive to this phenomenon and are more likely to find (or approximate) the global maximum. This is due to the use of a population of

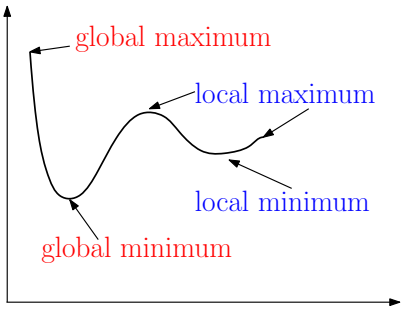


Figure 2.17: Optimisation problems have local maxima and minima points; these represent solutions that are better than those around them, but not the best overall.

candidate solutions rather than a single one. GAs are also good at handling problems with a complex mathematical representation, since genetic algorithms only require the outcome of the fitness function for each individual and are not concerned with other aspects of the fitness function such as derivatives. As such, they can be used for problems with complex mathematical representations or functions that are hard or impossible to differentiate. Other complex cases where genetic algorithms excel include problems with a large number of parameters and problems with a mix of parameter types; for example, a combination of continuous and discrete parameters.

Another benefit of GA's is that they can handle problems that does not have a clear mathematical representation provided that the problem can run on a computer and the problem's objective functions are encoded using a mathematical representation (to enable evolution). One such case of particular interest is when the fitness score is based on human opinion. Likewise, GAs can deal with cases where the score of each individual cannot be obtained, as long as there is a way to compare two individuals and determine which is better. For instance, an ML algorithm that drives a car in a simulated race can be improved using GAs. A GA-based search can optimise and tune the ML algorithm by having different versions of it compete against each other to determine which version is better. Additionally, GAs are generally resilient to noise, thanks to the repetitive operation of reassembling and reevaluating the individuals. They also support parallelism and distributed processing, given that the fitness calculation, as well as the operations of selection, crossover, and mutation, can each be performed concurrently on individuals and pairs of individuals in the population, respectively. This makes GAs suitable for distributed processing. Finally, GAs are suitable for continuous learning because they can operate continuously in an ever-changing environment, and at any point in time, the best current solution can be fetched and used as required.

On the other hand, GAs also have several limitations. These include, for instance, the need for special definitions: each problem needs a suitable representation, i.e. to define the fitness function and the chromosome structure, as well as the selection, crossover, and mutation operators that will work for this problem. This can be challenging and time-consuming, and requires significant domain expertise. GAs also need hyperparameter tuning, as their operation relies on hyperparameters such as population size and mutation rate. There are no exact rules for making these choices. Because a GAs operate on large populations and given its repetitive nature, it can be computationally intensive, as well as time-consuming before a good result is reached. Genetic algorithms also have the risk of premature convergence. If

the fitness of one individual is much higher than the rest of the population, this individual may be duplicated enough that it takes over the entire population. This can lead to the GA getting prematurely stuck in a local maximum, instead of finding the global one. Finally, there is no guaranteed solution. The use of GAs does not guarantee that the global maximum for the problem at hand will be found [161].

Despite these limitations, GAs generally remain an appropriate option for solving complex non-linear models where the location of the global optimum is a difficult task—not least because it may be possible to use GA techniques to tackle problems that cannot be modelled easily or accurately using other approaches.

2.4 Ensemble learning

Ensemble learning represents a machine learning approach that allows multiple machine learning models, called base learners or weak learners, to consolidate their predictions and output a single, optimal prediction, given their respective inputs and outputs [167]. Ensemble learning is inspired by *the wisdom of crowds* phenomenon described in social science, which suggests criteria applicable to groups of people making decisions together. It is claimed that, if these criteria are satisfied, then the aggregate decisions made by the group will often be better than those of its individual members [4].

Ideally, the individual models used in ensemble learning should be diverse to make the ensemble rich in solutions, with some models classified as generalists and others as specialists. For an ensemble used in a classification problem, a generalist model can maintain a decent performance for a variety of classes without excelling in any of them, whereas a specialist is an excellent model for making predictions about one class but may perform poorly for the rest of the classes. The combination of both model types is expected to show great improvement in terms of performance. To achieve this, the ensemble is expected to learn when to rely on specialist models and when to disregard their opinion and consider the output of a generalist model.

Ensemble methods try to solve the problems of bias and variance in ML models. Bias refers to the inability of a method to correctly estimate the target. In ML, bias refers to the difference between the expected prediction and its target. Biased models cannot properly fit the training data, resulting in poor performance during training and testing, as illustrated by the example on the left of Figure 2.18. On the other hand, variance refers to how much individuals vary within a group. In the ML context variance refers to the model's variability or sensitivity to data changes. High-variance models can generally fit the training data well; however, they perform poorly in the test data set. This is termed overfitting, an example of which is shown on the right of Figure 2.18. Achieving a good trade-off between bias and variance can be an optimal point of complexity, where the error of the model is minimised and it performs best both during training and testing [79].

Ensemble methods rely on the fact that the same algorithm can produce different models, due to the initial conditions and hyperparameters or through the use of a mix of different model architectures. By combining different, diverse models, it is typically possible to reduce the expected error of the group, while each individual model remains unchanged.

The benefits of using ensemble learning over single models encompasses performance and robustness. The performance is increased by combining the predictions

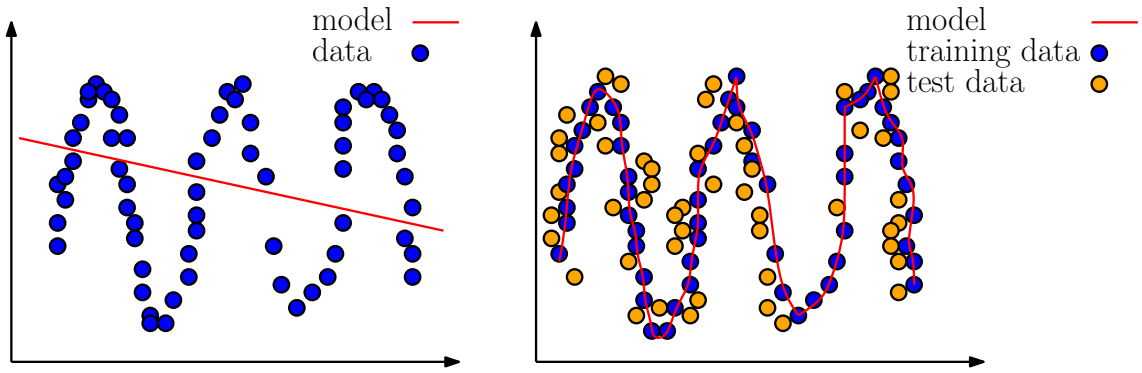


Figure 2.18: A biased linear regression model for data obtained from a sine function data (left), and a high variance model that fits the training data perfectly but has a poor performance on the test data (right).

of base learners. In general, the more base learners added to the ensemble, the more accurate the ensemble will be, under the condition that each learner is uncorrelated to the others. This is increasingly difficult to achieve as the set of base learners grows larger. Furthermore, the law of diminishing returns applies [33].

Additionally, ensemble methods try to construct a set of hypotheses and combine them for joint use. As such the output and forecasts of ensembles tend to remain consistently accurate even when one or more of the input variables or assumptions are changed due to unforeseen circumstances. As a result, the robustness of an ensemble is superior compared to that of a base learner [169].

Ensemble learning also presents several difficulties and drawbacks. If the data used for training and testing the ensemble models are noisy or incomplete, no single machine learning technique will generate a highly performant model. For example, in a study of a population of cars, if data about the colour, shape, and manufacturer are gathered, it is difficult to generate an accurate model for any variable, as many cars have the same colour and shape but are built by a different manufacturer. In this example, adding more features to the dataset can greatly improve the model's performance. In the same scenario, adding more models to an ensemble cannot provide improved performance.

There is also the interpretability problem, as by employing a large number of models, interpretability could be reduced. Sometimes, it is difficult to explain why a single model predicted one class. In an ensemble, there are multiple models affecting the final prediction and may be more things to explain than the prediction process itself, such as why the ensemble choose to train these specific models and not others and also the number of selected models. Coming next is the high computational cost of training an ensemble. Training a single neural network is already computationally expensive, and training 100 models requires 100 times more computational resources. Computational costs do not only hinder the ensemble training, but also the use of the ensemble in production, as now multiple predictions need to be obtained and combined to produce the output. In real-time settings, millisecond execution times are sometimes expected, and therefore even a few microseconds of added latency can make a huge difference.

Finally, there is not a clear way of choosing the right models for inclusion into an ensemble. Ideally, the models included in the ensemble should possess different characteristics. Their diversity could depend on a number of factors, such as the

size and quality of the training data set, and the learning algorithm itself. However, many different models could be developed and there is not a clear way of deciding which to include in an ensemble [79].

2.4.1 Ensemble methods

Ensemble methods are divided into two major taxonomies [79]: generative and non-generative methods. Non-generative methods are focused on combining the predictions of a set of pre-trained models usually trained independently of one another, and the ensemble algorithm dictates how their predictions will be combined. Base classifiers are not affected by the fact that they exist in an ensemble. Generative methods, on the other hand, are able to generate and affect the base learners that they use. They can either tune their learning algorithm or the dataset used to train them.

2.4.1.1 Non-generative methods

Voting. This type of non-generative ensemble method refers to techniques that allow models to vote in order to produce a single answer, similar to how individuals vote in elections. The most popular (most voted for) answer is then selected as the winner. Two main voting techniques can be used, hard and soft voting. Hard voting combines a number of predictions by assuming that the most voted class is the winner. Soft voting takes into account the probability of the predicted classes. In order to combine the predictions, this technique calculates the average probability of each class and assumes that the winner is the class with the highest average probability [79, 122, 152].

Stacking. Stacking, on the other hand, refers to methods that utilise a model which learns how to best combine the base learners' predictions. Stacking is a form of meta-learning. The main idea is that use base learners in order to generate metadata for the problem's dataset and then utilise another learner called a meta-learner, in order to process the metadata. Base learners are considered to be level 0 learners, while the meta-learner is considered a level 1 learner [79, 122, 152].

2.4.1.2 Generative methods

Bagging. Bagging ensemble methods aim to reduce variance. A bagging algorithm resamples instances of the training dataset, creating many individual and diverse datasets, originating from the same dataset. Afterwards, a separate model is trained on each sampled dataset, forcing diversity between the models included in the ensemble [8, 79].

Boosting. Boosting is a technique mainly targeting biased models. Its main idea is to sequentially generate models, such that each new model addresses biases inherent in the previous models. Thus, by iteratively correcting previous errors, the final ensemble has a significantly lower bias [16, 79].

Random forests. This type of generative ensemble method is similar to bagging, in that it resamples from the training dataset. However, instead of sampling data instances, it samples features, thus creating even more diverse trees since features strongly correlated to the target may be absent in many trees [8, 79].

2.5 Summary

This chapter presented concepts, definitions and terminology necessary to understand the proposed approaches and contributions of this thesis. We started by describing machine learning and narrowed it down to supervised learning and in particular deep neural networks and how they are applied to image classification and object detection. Next, we defined risk assessment following the recommendations provided by the ISO 31000 international standard, and discussed the use of the consequence/likelihood matrix for recording and reporting risk information. We then presented genetic algorithms, focusing on key concepts such as chromosome, population and fitness function, each of which was briefly explained. We also presented the main operations performed by a GA, namely selection, crossover and mutation, as well as the basic GA workflow. Finally, we concluded the chapter by describing ensemble learning and how it aims to tackle the problem of bias and variance in ML, improving the performance and robustness of machine learning. Two main classes of ensemble methods, i.e. non-generative and generative methods, were discussed briefly.

Chapter 3

Mitigating Risk in Neural Network Classifiers

This chapter presents a four-step approach that considers the risk factors associated with supervised learning models, reducing the misclassifications associated with high risks in the intended use of a ML classifier. The proposed method (i) allows the risk misclassification between classes to be quantified, (ii) guides the training of DNN classifiers towards mitigating the risks that require treatment, and (iii) synthesises risk-aware ensembles with the aid of multi-objective genetic algorithms that seek to optimise ML model performance metrics while also mitigating risk. We start by describing the general approach in Section 3.1. Then, in Section 3.2 we describe the application of our approach to the context of image classification using DNN classifiers. Next, in Section 3.3, we evaluate the the approach on two data sets, CIFAR-10 [77] and a subset of the GTSRB [141]. We then present related work in Section 3.4. Finally, Section 3.5 concludes the chapter with a brief summary.

3.1 General approach for the synthesis of risk-aware supervised-learning model ensembles

The steps of our method for taking risk into account in ML models are depicted in Figure 3.1. In the first stage, *risk-oblivious training*, we obtain a set of traditionally trained ML models. The weights of these models are initialised with random values prior to training such that each of the trained models, although having the same structure, is different. Then, in stage two, *risk-aware verification*, a set of *risk concerns*, which are risk values associated with class pairs whose misclassification results in risks that require treatment, is obtained. We follow the ISO/IEC 31010 risk management standard [65], which provides guidance on how information supplied by a domain expert can be included in a risk assessment process by considering risk information such as likelihood of encounter, impact, and likelihood of misclassification. In the third stage, called *risk-aware training*, the risk concerns identified in the previous stage are used to produce sets of risk-aware ML models that mitigate the risk for the concern they were created for. Finally, in the fourth stage, *risk-aware ensemble synthesis and verification*, our approach combines subsets of the risk-oblivious and the risk-aware models, where these subsets are selected based on their performance and risk values. The models in these subsets are fed to a

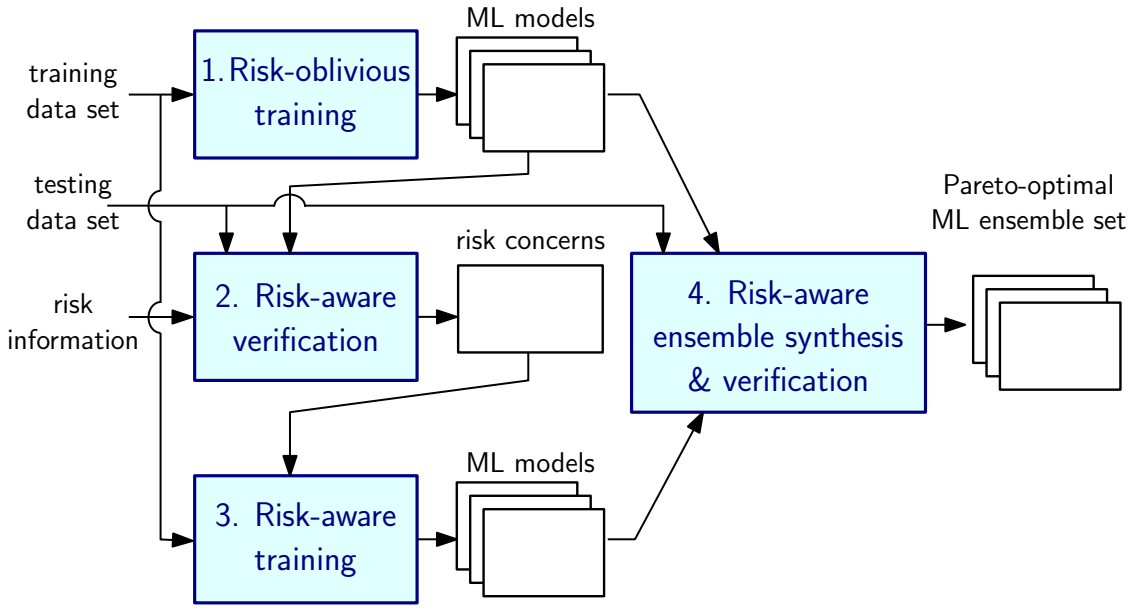


Figure 3.1: Four-step method for the synthesis of risk-aware supervised learning models.

multi-objective genetic algorithm whose fitness function seeks to create a set of ML ensembles that achieve optimal trade-offs between maximising a performance metric (which depends on the type of supervised learning models involved) and minimising the residual risk.

3.2 Implementing the approach for deep neural network classifiers

In this section, we instantiate our generic approach from Figure 3.1 for DNN classifiers. We provide a comprehensive breakdown of each stage of the instantiated approach in the following sections, offering detailed descriptions and insights into its implementation process.

3.2.1 Risk-oblivious training

In this stage, we obtain a set of traditionally trained DNN models. The fact that initially the weights in the DNN are randomly assigned results in different final models, even when the same DNN structure is used for all models. Nevertheless, our approach also supports the use of DNN models with different structure. By generating a set of models, we are able to identify those risk concerns that are common to all models generated from the training set. We term these models *risk-oblivious* because no risk information has been considered during the training of these models, and because we use the traditional loss function, which gives the same importance to all misclassifications.

We note that this first stage is required in order to obtain a baseline that allows the identification (in the next stage of the approach) of class pairs whose misclassification as one another corresponds to unacceptable levels of risks. As such, risk-aware training needs to be employed (in a third stage of our approach) to mitigate these

risks.

3.2.2 Risk-aware verification

The objective of this step is to obtain a set of *risk concerns*, which are risk values associated with class pairs deemed to require treatment. We follow the ISO/IEC 31010 risk management standard [65], which provides guidance on how information from a domain expert can be included in a risk assessment process by using the following parameters:

1. the likelihood LoE_i of encountering each class i, provided on a five-point ordinal scale—very low (VL), low (L), medium (M), high (H) and very high (VH), and reflecting the likelihood of encountering an instance of a class in a given context;

2. the impact of misclassifying an instance of class i as class j when $i \neq j$, which is defined on the same five-point scale, i.e., $impact_{(i,j)} \in \{VL, L, M, H, VH\}$;

3. the likelihood of misclassification (LoM) thresholds

$$LoM^0, LoM^{\mathsf{VL}}, LoM^{\mathsf{L}}, LoM^{\mathsf{M}}, LoM^{\mathsf{H}}, LoM^{\mathsf{VH}}, LoM^1 \tag{3.1}$$

   where

$$0 = LoM^0 < LoM^{\mathsf{VL}} < LoM^{\mathsf{L}} < LoM^{\mathsf{M}} < LoM^{\mathsf{H}} < LoM^{\mathsf{VH}} < LoM^1 = 1; \tag{3.2}$$

4. the risk threshold τ which specifies the maximum risk level that can be tolerated; the risk associated with misclassifying a data sample of class i as class j needs to be mitigated if its risk level $r_{(i,j)}$ is greater than τ.

Given these parameters, and the fraction $p_{(i,j)}$ of data samples of class i from a test data set that are misclassified as class $j \neq i$ by the DNNs from step 1, we first establish the likelihood of misclassification $LoM_{(i,j)}$ for the class pair (i, j) as the unique element from $\{VL, L, M, H, VH\}$ that satisfies $LoM^{pred(LoM_{(i,j)})} < p_{(i,j)} \leq LoM^{LoM_{(i,j)}}$, where the predecessor function $pred$ is defined by $pred(\mathsf{VL}) = 0$, $pred(\mathsf{L}) = \mathsf{VL}$, etc. Next, we compute the overall likelihood $OL_{(i,j)}$ of misclassifying class i as class j by combining LoE_i and $LoM_{(i,j)}$ by using the mapping from Table 3.1. Finally, we use the mapping from Table 3.2 to derive the risk level $r_{(i,j)}$ associated with misclassifying class i as class j from $OL_{(i,j)}$ and $impact_{(i,j)}$, and we consider the class pair (i, j) as a *risk concern* if and only if $r_{(i,j)} > \tau$.

Example 3.2.1 *Assume that $LoE_i = M$, $LoM_{(i,j)} = VH$ and $impact_{(i,j)} = H$ for a class pair (i, j). Using the mapping from Table 3.1, we obtain $OL_{(i,j)} = H$; then, we combine $OL_{(i,j)}$ and $impact_{(i,j)}$ by using the mapping from Table 3.1, obtaining $r_{(i,j)} = H$. If we further consider $\tau = M$, then all the class pairs with risk level above M, i.e., those corresponding to the highlighted cells from Table 3.2 represent risk concerns. This includes the class (i, j) from our example.*

Table 3.1: Overall likelihood $OL_{(i,j)}$

	$LoM_{(i,j)}$				
LoE_i	VL	L	M	H	VH
VH	L	M	H	VH	VH
H	L	M	M	H	VH
M	L	L	M	M	H
L	VL	L	L	M	M
VL	VL	VL	L	L	L

Table 3.2: Risk $r_{(i,j)}$

	$impact_{(i,j)}$				
$OL_{(i,j)}$	VL	L	M	H	VH
VH	L	M	H	VH	VH
H	L	L	M	H	H
M	VL	L	M	M	M
L	VL	L	L	L	L
VL	VL	VL	VL	VL	VL

Once the thresholds (3.1) are specified by domain experts for the intended application for the DNN classifier under development, we can define the misclassification intervals as follows: VL=[LoM^{0}, LoM^{VL}); L=[LoM^{VL}, LoM^{L}); M=[LoM^{L}, LoM^{M}); H=[LoM^{M}, LoM^{H}); VH=[LoM^{H}, LoM^{VH}); P=[LoM^{VH}, LoM^{1}].

We can think of them as buckets were each of the misclassification pairs, will be placed. We added an extra interval, 'punish' (P), that will include all misclassifications that go higher than the maximum value of misclassification LoM^{VH}; this is useful to guide the ensemble synthesis, because the GA will instantly realise that certain combination of models and weights incur such a high amount of risk and will penalise the fitness function of that individual. A detailed description of how the risk is encoded is provided in Section 3.2.4.

3.2.3 Risk-aware training

In this step, the risk concerns identified in the previous step are used to produce sets of risk-aware DNN models that aim to mitigate the risk for the concern they were created for. To achieve that, we modify the cross-entropy loss function by multiplying the result of the classification loss by a weights matrix ω that encodes a penalty value in the class-pair whose misclassification risk we need to mitigate. This amplifies the contribution of misclassifications that need mitigation to the overall loss function. Consequently, the network is *forced* to tweak its weights in order to reduce the error, and eventually achieve a smaller misclassification value for the concern. The cross-entropy function traditionally assumes a uniform weighting independent of the predicted or actual class. In contrast, our a class-weighted cross entropy is constructed as:

$$\mathcal{L}(\theta) = -\frac{1}{M}\sum_{i=1}^{M}\sum_{n=1}^{N}\omega_n y_n^i \log\left(\hat{p}_n^i\right) \tag{3.3}$$

where θ represents the DNN parameters to be learnt, M is the number of instances in the data set, y_n^i is the target probability that the i^{th} instance belongs to class n and $\hat{p}_n^i$ is the output of the soft-max for instance i belonging to class n. Note that generally y_n^i is either 1 or 0 depending on whether the instance belongs to the class or not. Finally, ω_n is a weight associated with misclassifying class n. Weighting the class in this way is commonly used to tackle class imbalance in training sets. There is not a specific rule for the selection of the ω_n values. In the experiments carried out in Section 3.3 the values were selected through trial and error, with the observation that, as the weight value increases for a given class pair, the associated number of misclassifications for that class pair is reduced (see Figure 3.9). However,

the value also has an effect on the misclassifications for the rest of the class pairs, in the sense that if it is set too high, then the overall accuracy of the DNN model is adversely affected.

The resulting *risk-aware models* may be viewed as "experts" in avoiding one form of misclassification. In the next step, we show how these experts are combined with the "generalists" created in the first step of our approach.

3.2.4 Risk-aware ensemble synthesis and verification

The final step of our approach combines subsets of the risk-oblivious and the risk-mitigating models obtained in steps 1 (Risk-oblivious training) and 3 (Risk-aware training), respectively, where these subsets are selected based on their performance and risk values. The models in the subsets are fed to a multi-objective genetic algorithm (GA) that synthesises the ensemble described in the following section by searching for the best set of weights to combine the risk-oblivious and risk-aware models into an ensemble that maximises performance and reduces the risk-relevant misclassification errors as much as possible. From among the existing search algorithms, we preferred to use GAs because they can efficiently search through large and complex solution spaces such as the one presented in this section. Moreover, GAs are designed to explore multiple solutions simultaneously and can quickly converge on promising solutions through the use of crossover and mutation operators. This makes them particularly effective in situations where (like in the case of our ensemble synthesis) there are multiple possible solutions to a problem, and it is not feasible to explore all possible solutions exhaustively.

Another method for synthesising the ensemble would have been to train a NN that would combine the results of individual NNs in the ensemble into the final result. We did not opt for this method because it requires to first identify the best set of individual NN to be combined in the output-layer of the ensemble. Therefore, using this method would require an extra step before the ensemble creation, whilst the GA architecture that we adopted performs both tasks (i.e. model selection and ensemble training) simultaneously. Nevertheless, using a NN to combine the results of the risk-oblivious and risk-mitigating NNs from the ensemble remains a valid option that is worth exploring in future research.

3.2.4.1 Ensemble construction

We use the risk-aware ensemble architecture depicted in Figure 3.2, where each of n models allowed in the ensemble is presented with an input to be classified into one of m classes. Each model then produces a distribution of class probabilities using a $soft\_max$ function, such that p_i^j is the probability of class i predicted by model j. Each of these predictions is then weighted by a value w_i^j, the intuition being that if model j is good at predicting class i then it will end up being more highly weighted in the ensemble. For each class, we then sum the weighted predictions and apply an arg_max function to obtain a class prediction from the ensemble.

To decide which models and weights should be used in the ensemble and to ensure that the appropriate trade off is achieved between performance and risk we used the GA strategy described below.

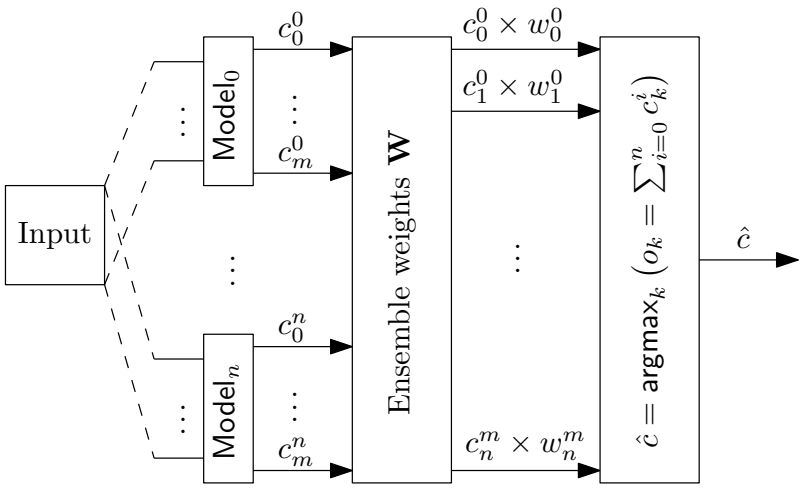


Figure 3.2: Risk-Aware Ensemble architecture. We combine the predictions of n models for the same input with a set of weights. The weights are values between 0 and 1 and help to determine how much confidence the ensemble has on the model's prediction. Finally, all the predictions are added up and we apply an argmax function to obtain the ultimate ensemble prediction.

3.2.4.2 The GA strategy

The GA strategy requires (i) the definition of a chromosome which encodes features of the problem space; (ii) the optimisation criteria and constraints against which chromosome fitness will be assessed; and (iii) the specification of an evolution strategy for each generation.

Chromosome encoding. The chromosome is encoded as a single binary indicator value and N floating point values per model in the candidate set. A value of 1 in the binary field indicates that the model is included in the ensemble, while a value of zero indicates it is excluded. Each of the floating point values then represents the weights associated with the output of that model.

Fitness function. Our GA strategy has two fitness functions and one constraint. The first fitness function seeks to maximise a performance metric for the models such as the F1 score for image classifiers or (see the next chapter) mAP for object detectors, and the second to minimise the residual risk defined as:

$$residual\_risk = \sum_{r_{(i,j)} > \tau} res\_risk(LoE_i, impact_{(i,j)}, p_{(i,j)}, \tau) \tag{3.4}$$

where $res\_risk(LoE_i, impact_{(i,j)}, p_{(i,j)}, \tau)$ for the class pair (i, j) is a positive value that reflects by how much the risk of misclassifying class i as class j exceeds τ if $r_{(i,j)} > \tau$, and zero otherwise. The constraint is that the number of models included in the ensemble (Ens) must satisfy:

$$Ens_{min} \leq Ens \leq Ens_{max} \tag{3.5}$$

where the bounds Ens_{min}, Ens_{max} are pre-specified GA hyperparameters.

Evolution strategy. For the evolution strategy we use the DEAP evolutionary framework [46] that implements the NSGA-II [36] algorithm for selection and then apply cross-over and mutation as described next.

First, let us consider the binary indicator fields. Using a single cross over point applied to the model selection genes, we generate two children. Next we examine each candidate chromosome. If the number of models *turned on* is outside the constraint bounds (3.5), we choose models, at random, to be turned on or off as appropriate.

Example 3.2.2 *For example consider the following model selection genes for two parents describing a problem with 10 candidate models:*

$$\begin{aligned} g1 =& [1,1,1,1,0,0,1,0,0,0] \\ g2 =& [0,0,0,0,1,0,1,0,1,1] \end{aligned} \tag{3.6}$$

Assume that we have a constraint stating that the number of active models must be between 3 and 5 inclusive, and that a cross over point of two is selected at random, generating two child genes as:

$$\begin{aligned} c1 =& [1,1,|0,0,1,0,1,0,1,1] \\ c2 =& [0,0,|1,1,0,0,1,0,0,0] \end{aligned} \tag{3.7}$$

where | denotes our crossover point. The resulting child $c2$ is valid since 3 of its models are active; $c1$, however, has 6 models active and is hence invalid. From those models turned on in $c1$ we select one at random (in this case model 2) and set its value to zero to obtain two valid genes:

$$\begin{aligned} c1 =& [1,0,0,0,1,0,1,0,1,1] \\ c2 =& [0,0,1,1,0,0,1,0,0,0] \end{aligned} \tag{3.8}$$

Mutation is applied to a chromosome as a bit swap. First, two genes of the chromosome are selected at random and their binary indicators are swapped. After the swap we check that the total number of active models remains unchanged to meet the constraint (3.5). For example, to mutate chromosome $c2$ models 1 and 3 are swapped giving a mutated chromosome $c3$:

$$c3 = [1,0,0,1,0,0,1,0,0,0] \tag{3.9}$$

For the floating point values, we applied simulated binary crossover [35] (which applies if and only if the model is selected for inclusion in one of the chromosomes). For this reason it is necessary to first apply crossover and mutation to the binary indicator fields.

3.2.4.3 Encoding the risk with fractional parts

In this section we explain that the risk value for each concern needs to have a fractional part, and how this value is obtained.

In order to guide the search of the GA it was necessary to implement a strategy that could help to identify when a selection of models and weights during the ensemble synthesis had a noticeable impact on the amount of the residual risk. In this

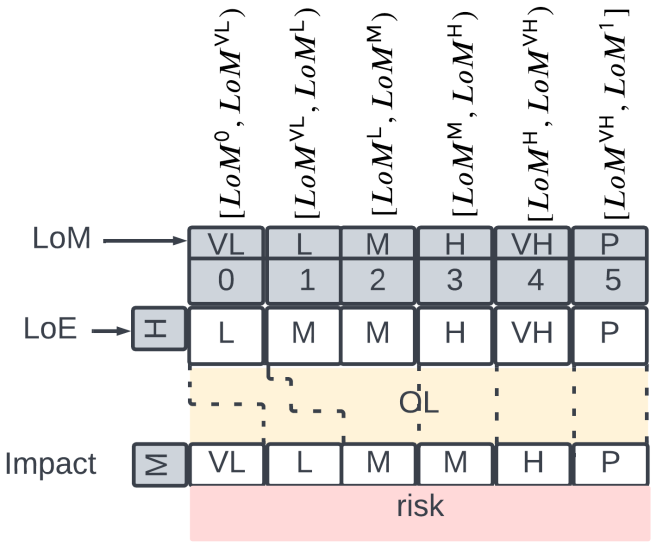


Figure 3.3: The relationship between the mapping tables and the thresholds of misclassification. Observe, how the LoM combines with the LoE to obtain the overall likelihood (OL). Note how the columns are not equally distributed, e.g. the mapped OL has two cells for medium (M); however, we still want to differentiate how further into the segment a mapped value is, otherwise, a LoM with a low (L) value will not be different from a medium (M) LoM in the OL. Finally, the OL combines with the impact to obtain the final risk.

way, the GA can find in which direction to continue the search for Pareto-optimal solutions.

The strategy that we propose is to add a fractional part to the risk value and even when the risk for a concern could not be brought below the current risk level from one GA generation to the next, the GA will still be able to notice a change in the risk value although it may be subtle. For example, if the current risk is 4.8 and a set of weights selected by the GA yields to a risk value of 4.5, we can see how the risk remains in the same risk level 4; however, there is an indication of risk reduction and by continuing the search in that direction the GA is likely to find a better solution.

The relationship between the mapping Tables 3.1 and 3.2 and the likelihood of misclassification thresholds is depicted in Figure 3.3. The risk mapping comprises the following steps:

1) Obtain the *fractional* likelihood of misclassitication $fLoM$, which transforms the mean fraction of misclassification $(p(i,j))$ of class i as class j over the risk-oblivious models to a number within the interval $[0,5]$. We do this by using the intervals defined in Section 3.2.2 and the following formula:

$$fLoM = mapped\_value(p(i,j)) + \frac{p(i,j) - left(p(i,j))}{right(p(i,j)) - left(p(i,j))} \tag{3.10}$$

where

$$mapped\_value(x) = \begin{cases} 0, & \text{if } x \in [LoM^{0}, LoM^{\mathsf{VL}}) \\ 1, & \text{if } x \in [LoM^{\mathsf{VL}}, LoM^{\mathsf{L}}) \\ 2, & \text{if } x \in [LoM^{\mathsf{L}}, LoM^{\mathsf{M}}) \\ 3, & \text{if } x \in [LoM^{\mathsf{M}}, LoM^{\mathsf{H}}) \\ 4, & \text{if } x \in [LoM^{\mathsf{H}}, LoM^{\mathsf{VH}}) \\ 5, & \text{if } x \in [LoM^{\mathsf{VH}}, LoM^{1}] \end{cases}$$

is the index of the LoM interval that $p(i,j)$ belongs to, and

$$(left(x), right(x)) = \begin{cases} (LoM^{0}, LoM^{\mathsf{VL}}), & \text{if } x \in [LoM^{0}, LoM^{\mathsf{VL}}) \\ (LoM^{\mathsf{VL}}, LoM^{\mathsf{L}}), & \text{if } x \in [LoM^{\mathsf{VL}}, LoM^{\mathsf{L}}) \\ (LoM^{\mathsf{L}}, LoM^{\mathsf{M}}), & \text{if } x \in [LoM^{\mathsf{L}}, LoM^{\mathsf{M}}) \\ (LoM^{\mathsf{M}}, LoM^{\mathsf{H}}), & \text{if } x \in [LoM^{\mathsf{M}}, M^{\mathsf{H}}) \\ (LoM^{\mathsf{H}}, LoM^{\mathsf{VH}}), & \text{if } x \in [LoM^{\mathsf{H}}, LoM^{\mathsf{VH}}) \\ (LoM^{\mathsf{VH}}, LoM^{1}), & \text{if } x \in [LoM^{\mathsf{VH}}, LoM^{1}] \end{cases}$$

represent the smallest misclassification fraction values for a concern with the same LoM tier as the concern under analysis and for a concern with the next LoM tier, respectively. For example, if $p(i,j) \in [LoM^{\mathsf{VL}}, LoM^{\mathsf{L}})$, we have $mapped\_value(p(i,j)) = 1$, $left(p(i,j)) = LoM^{\mathsf{VL}}$ and $right(p(i,j)) = LoM^{\mathsf{L}}$, so we obtain

$$fLoM = 1 + \frac{p(i,j) - LoM^{\mathsf{VL}}}{LoM^{\mathsf{L}} - LoM^{\mathsf{VL}}}.$$

2) Obtain the *fractional* overall likelihood fOL, which combines the likelihood of encounter LoE and $fLoM$ using Table 3.1. We need to look up the row of Table 3.1 where LoE is located, e.g. if $LoE = 3(H)$ then we use the $row = [L(1)\ M(2)\ M(2)\ H(3)\ VH(4)]$. As we can see, the table is not evenly divided, for instance in row we have M located in positions 1 and 2, and because we want to capture this variations, i.e. how far into M a concern is located, we calculate the size sz of the segment of row elements with the same (non-fractional) OL value as the concern being examined:

$$sz = last - first + 1 \tag{3.11}$$

where *first* is the index of first element in row with the same OL as our concern, and *last* is the index of last element in row with the same OL as our concern. With this notation, we calculate fOL as:

$$fOL = OL + \frac{fLoM - first}{sz} \tag{3.12}$$

We illustrate this step in Figure 3.3, where we assume that LoE=H. It is possible to appreciate how the columns are not equally distributed, e.g. the mapped OL has two cells for medium (M). However, we still want to differentiate how further

into the segment a mapped value is, in order to guide the risk reduction in the GA. Otherwise, a LoM with a low (L) value will not be different from a medium (M) LoM in the OL value.

3) Finally, we calculate the fractional risk *frisk* in a similar way to fOL, but now mapping OL and *impact* in Table 3.2. In this case, we require the relevant *column* of Table 3.2 because the impact levels are stored column-wise. For example, if $impact = 2(M)$, then $column = [VL(0)\ L(1)\ M(2)\ M(2)\ H(3)]$ needs to be used. Again, we calculate sz using (3.11) and the final fractional risk is obtained using the expression:

$$frisk = risk + \frac{fOL - first}{sz} \tag{3.13}$$

where $risk$ is the mapped OL and *impact* using Table 3.2, and sz, *first* were already defined above. An example of applying the whole process is presented later in Section 3.3.2.

In Figure 3.3, we can visualise how OL combined with different impact values yields different risk levels. We highlight the fact that the mapping is not one to one, e.g. the minimum risk level that can be obtained when OL=L and impact=M is L instead of VL, and the maximum risk can never be higher than H (except when a penalisation happens).

3.3 Evaluation

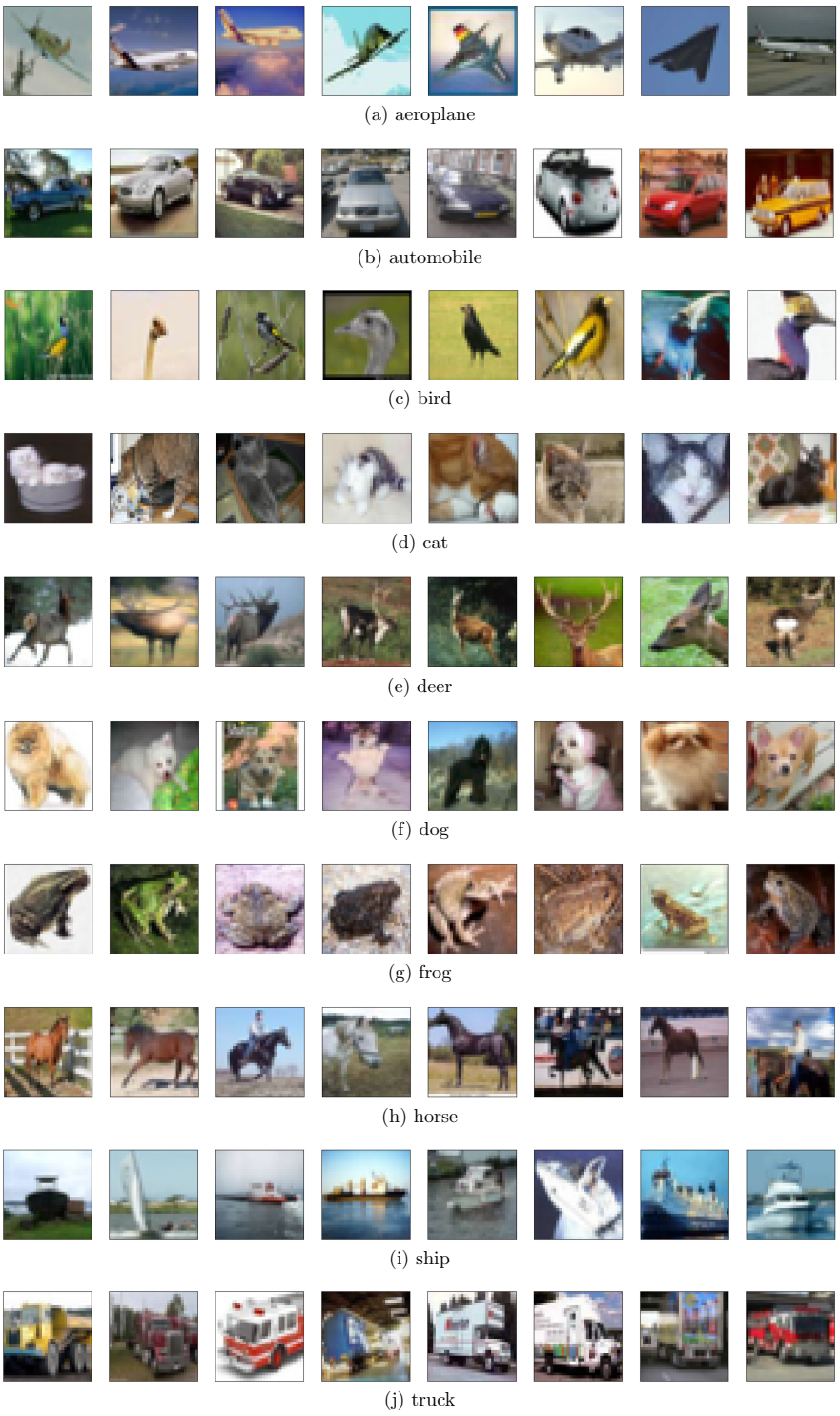
3.3.1 Evaluation methodology

The effectiveness of the approach was assessed through applying it in two case studies. For the first case study, we took the widely used CIFAR-10 dataset [77] which contains 60,000 images of transportation vehicles e.g. truck, car, etc., ships, and animals and whose classes are depicted in Figure 3.4. The individual models were trained using the training set of CIFAR-10 images. For testing, we split the CIFAR-10 test image set into two subsets of the same size, and we used one subset for testing the individual models, and the other subset for the testing of the ensemble.

For the second case study we used the well-known GTSRB benchmark [141], which contains 50,000 labelled images from 43 classes. The data set contains images of speed limit signs, prohibitory signs, mandatory signs, and danger signs. Examples are provided in Figure 3.5. Due to its highly imbalanced nature, a subset of the benchmark was created by selecting only samples from the speed limit signs and an extra class composed of random samples from the non-speed limit signs in the dataset. From the original test set, we randomly took 100 samples per class in the subset, and then augmented them to obtain a test set for the ensemble. The remaining test images were used to test the individual models. For augmentation, we used width shift range, height shift range and rotation range—all of which were carried out using functions already available in the Keras ImageDataGenerator API.[1]

All the experiments were performed on a server with 64GB of RAM with 8 Intel Core i7-9700K 3.60GHz CPUs, running Ubuntu 18.04.5 with a 64-bit architecture and 6.9TB of hard disk.

[1]more details about this API can be found at `https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator`



(a) aeroplane

(b) automobile

(c) bird

(d) cat

(e) deer

(f) dog

(g) frog

(h) horse

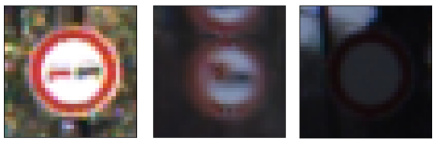
(i) ship

(j) truck

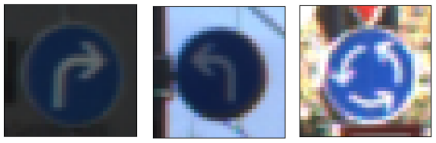
Figure 3.4: Sample images for each of the 10 classes in the CIFAR-10 data set.

The experiments carried out and their results are presented and discussed in Section 3.3.2 and 3.3.3, respectively.

(a) Speed limit signs



(b) Prohibitory signs



(c) Mandatory signs

(d) Danger signs

Figure 3.5: Example images from the GTSRB benchmark which includes images from speed limit signs, prohibitory signs, mandatory signs and danger signs. The data set contains 43 classes in total.

3.3.2 Evaluation on the CIFAR-10 data set

3.3.2.1 Inputs

The data set For our experiments, we used the CIFAR-10 dataset [77] which consists of 60,000 32x32 colour images with $N = 10$ classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images. The risk-oblivious and risk-aware models were trained using the training data. We split the 10,000 test images in two, 5,000 images for testing the individual models and 5,000 for testing the final ensemble.

Table 3.3: *Impact* for the misclassifications of classes in the CIFAR-10 dataset provided on a 0-4 scale of VL-VH. The cells with a dash are misclassifications considered irrelevant for the context or that does not make sense.

		Predicted Class									
		0	1	2	3	4	5	6	7	8	9
Actual Class	0	-	3	-	-	-	-	-	-	2	2
	1	4	-	-	-	-	-	-	-	1	0
	2	-	-	-	2	0	1	0	1	-	-
	3	-	-	0	-	1	0	1	1	-	-
	4	-	-	2	3	-	2	2	4	-	-
	5	-	-	3	4	3	-	3	2	-	-
	6	-	-	2	1	1	0	-	2	-	-
	7	-	-	2	1	2	1	0	-	-	-
	8	2	2	-	-	-	-	-	-	-	1
	9	3	4	-	-	-	-	-	-	2	-

Risk information

Impact level $impact(i, j)$**.** Typically, the impact assessment considers how the undesirable event being assessed could influence cost, schedule, or technical performance objectives, i.e., how the context will be affected if such an event is to happen. In Table 3.3 we present the impact values used in our evaluation for class misclassifications. We present the actual classes on the rows of the table and the predicted classes in the columns. The impact is being measured in a 0–4 scale (corresponding to VL–VH), and we marked with a dash the cells for misclassifications that are considered irrelevant for the context or that are not applicable because the actual class is the same as the predicted class. For example, if we look at Table 3.3 again, we can see that class aeroplane (0) predicted as class aeroplane (0) is dashed because this is not a misclassification and will not make sense to rate it with an impact value. Likewise, class aeroplane (0) predicted as either bird (2), cat (3), deer (4), dog (5), etc. have been dashed, which indicates that these misclassifications are not of interest for our context.

The dash should not be confused with a zero impact, when the impact is 0 (VL), this means that the misclassification is of interest in our context, nevertheless, when it happens, the effect that it will have is very low and its contribution to risk will be minimum; for example, we assessed that misclassifying a cat (3) as a bird (2) will have a very low impact in a self-drive scenario. In contrast, misclassifying a dog (5) as a cat (3) in a risky manoeuvre could lead to legal implications in the UK if the dog is injured as a result, which is why we assigned a 4 (i.e., VH) impact to this misclassification. One more example is misclassifying a truck (9) as an automobile (1): in an overtaking situation, wrong assumptions made about the size

Table 3.4: Likelihood of encounter (LoE) for the 10 classes of the CIFAR-10 dataset used to perform the evaluation of the proposed approach, it is given on a 0-4 scale of VL-VH.

Class	0	1	2	3	4	5	6	7	8	9
LoE	3	4	4	2	4	4	2	1	0	4

of the vehicle, speed and time could lead to a potential accident, hence the very high impact in such misclassification.

All the proposed impact values in Table 3.3 could be arguably different depending on perspectives and particular experiences. For the purpose of our evaluation, estimating these values is fine because we are just providing an example of what this table could look like. In a real-world scenario, expert domain input is required in order to fine-tune the impact values accordingly, to accurately reflect the actual impact values in the particular context of interest.

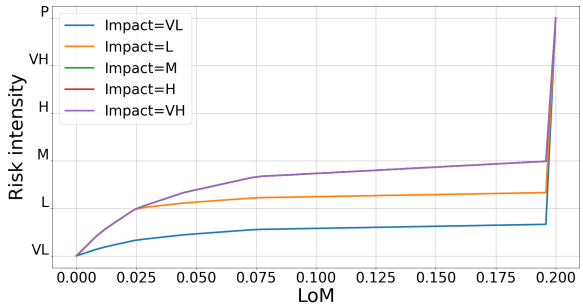
Likelihood of encounter $LoE(i)$. The LoE parameter refers to the likelihood of an event happening. LoE can be described as an expected probability or frequency of encountering an instance of a given class during a determined period of time. The likelihood of encounter we used for the evaluation on this data set is provided in Table 3.4. The LoE for each class is given again on a 0–4 scale (corresponding to VL–VH). For instance, we assessed that the classes automobile (1), bird (2), deer (4), dog (5) and truck (9) are very likely to be encountered during an hour-long road trip; on the other hand, the class ship (8) will very rarely be observed during an hour-long road trip. This table is again just an example; for any real-world application, it is recommended that this table is proposed by an expert domain.

Likelihood of misclassification thresholds. As described in Section 3.2.2, the LoM thresholds facilitate the mapping of the fractions of misclassification to a 5-point scale of VL–VH.

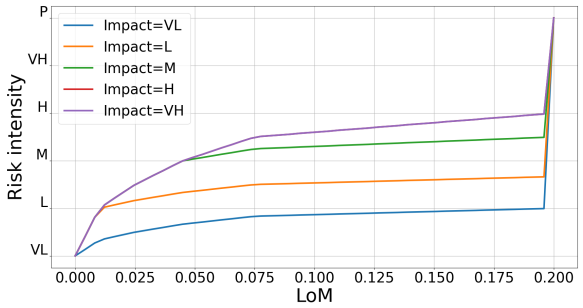
The thresholds that we used for this experiment are $0 < 0.01 < 0.025 < 0.045 < 0.075 < 0.2 < 1$. With the thresholds in place we can define the intervals for the risk levels as follows: VL=[0, 0.01); L=[0.01, 0.025); M=[0.025, 0.045); H=[0.045, 0.075); VH=[0.075, 0.2); P=[0.2, 1].

Figures 3.6 and 3.7 show how the calculated risk looks like when the LoE and impact, respectively, varies using the defined intervals. As we can see in both figures, the risk grows steadily as the LoM value is increasing. Each chart shows that the risk is a piece-linear function of the LoM between the intervals for the risk described above at different LoE (Figure 3.6) and impact levels (Figure 3.7). Our graphs were thought in this way because our synthesised ensemble can only mitigate risk by decreasing the LoM (risk mitigation by reducing likelihood of encounter and/or impact is possible by other means, which are out of scope for our work). These charts also allow to anticipate, given the LoE and impact, how much the LoM needs to be decreased for a concern to be mitigated e.g. to be moved down from VH to H, etc. which is directly linked to the width of each interval, the LoE and the impact.

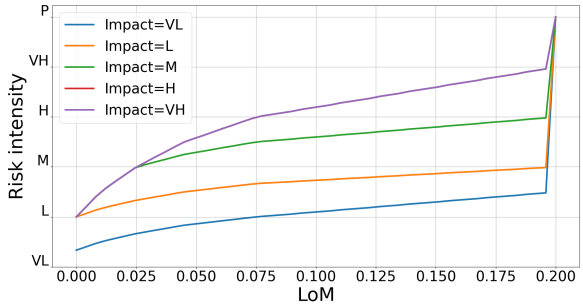
Figure 3.6, shows the effect of varying the LoE while the *impact* remains the



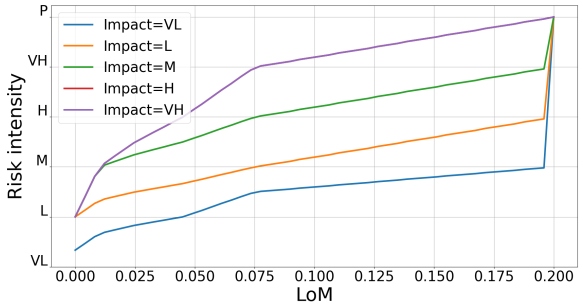
(a) LoE=VL



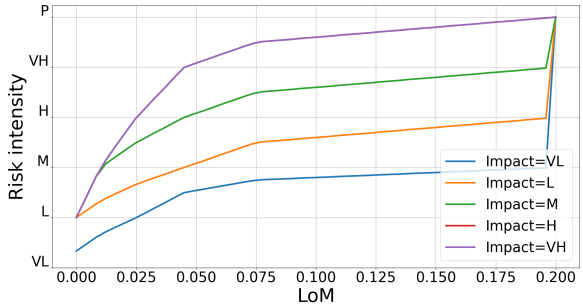
(b) LoE=L



(c) LoE=M



(d) LoE=H



(e) LoE=VH

Figure 3.6: The effect of varying the LoE at different impact levels.

same, there are five levels for the impact in each of the graphs, from VL-VH and each of them is depicted with a different colour to compare the risk behaviour at each impact level and how it changes in relation to the LoE. The charts are connected with the mapping tables 3.1 and 3.2. For instance when $LoE = VL$ if we look at Table 3.1 it does not matter how high the LoM goes, the highest overall likelihood (OL) will be Low (see the row for LoE=VL in Table 3.1). When we combine the OL with the $impact$ in Table 3.2 we realise that at a fixed $impact = VL$ the final $risk$ will never be above VL and this is reflected in chart 3.6a, see how the risk for the blue line ($impact = VL$) never hits the Low risk threshold. Basically, the charts allow the visualisation of the mapping tables for a better understanding of the complex risk mapping we are carrying out, and help us to spot where the risk thresholds leave the concerns and by how much the LoM should be reduced to actually mitigate our risk concerns. Some of the impact levels are overlapped for instance in graphs 3.6b to 3.6e the risk for $impact = H$ (red line) is never visible, this is because it overlaps with the $impact = VH$, and this can be confirmed when looking at Table 3.2 it can be seen how the columns of the table for levels of impact

high and very high have the same values.

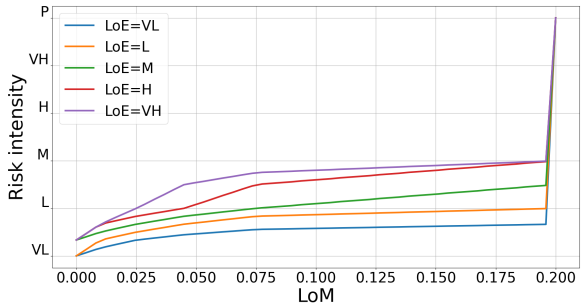
Figures 3.6 and 3.7 present a steep jump to P (the penalise value) when the LoM ≥ 0.2. This comes from the way in which the likelihood of misclassification thresholds are defined. In particular, when LoM $\in [\geq]0.075, 0.2)$, it is mapped to a VH likelihood of misclassification value, while LoM values of 0.2 or above are mapped to P. If the LoM maps to P then we automatically assign a P value to the risk as well. The objective of mapping to the P value is to guide the GA search and quickly notice when a selection of models and weights yields a risk value that is unacceptable, so that the search can continue in a different direction.

Now consider Figure 3.6a with an impact of VH (the purple line), moving a concern from the low risk to the very low risk requires the LoM to be decreased to a value smaller than 0.025 which depending on the risk-oblivious models, might be difficult to achieve, especially if the LoM is greater than 0.1; graphically, we can see how the risk for this interval seems to be mostly flat from the LoM values between 0.075 and 0.175. We are careful to avoid this behaviour when doing the mappings as in these cases, our method has little to do to mitigate the risk.

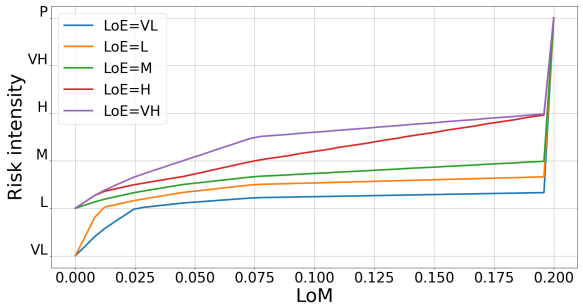
Conversely, in Figure 3.6e when the impact is VH (purple line), if we have a risk of H that we would like to mitigate to M, we can see how the slope of the risk in that area is sharp which indicates that mitigating the concern will be quick as even a small reduction of the LoM will have a significant influence in the amount of risk. If we discuss about mitigating a risk from VH to H (again with an impact of VH) we observe that the risk reduction needs to be significant for this concern to be alleviated, especially, if the LoM is greater than 0.1.

Figure 3.7 shows the risk behaviour at each of the five impact levels varying at each of them the LoE, which is represented with different colours. In Figure 3.7a we can observe the risk at each LoE value when $impact = VL$. Note that even when the $LoE = VH$, the worst risk value will be low; nevertheless, as the *impact* increases, so does the risk, e.g. in Figures 3.7d and 3.7e when the LoE has the same value VH, the risk gets as high as VH. Note how in Figure 3.7a, when the LoE is M, H and VH, the corresponding risk values (shown by lines in green, red, and purple) do not start at VL like the blue and yellow lines. This is because we split VL into three different segments, which is directly related to the mapping tables: if you look at Table 3.2, you will see how when the impact is VL (first column) the value very low is repeated three times, for $OL \in \{VL, L, M\}$, so when the $LoE \in \{VL, L\}$ (see Table 3.1), we have $OL = VL$ and mapping $OL = VL$ and $impact = VL$ gives a risk value of VL (first segment). However, when the $LoE \in \{M, H, VH\}$ (see Table 3.1), the mapped OL begins at level L (rather that VL), and when $OL = L$ and $impact = VL$ in Table 3.2 the mapped risk is VL (second segment), which explains why the risk values for $LoE = M$, H and VH do not start at VL. For further details about how we calculate the segment sizes, see Section 3.2.4.3.

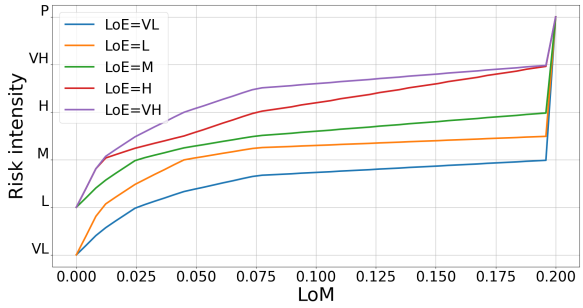
Figures 3.6 and 3.7 also give us an idea of the risk values for the set of concerns, and how by adjusting either the LoE or the *impact* values, our concerns could end up with different risk values. In other words, in the real world, there are different contexts where our systems could be deployed, and for each of these contexts the LoE or *impact* values could be different. For example, in a rural area it may be more likely to encounter animals such as horses, deer, etc. than in the city, whereas in the city, pedestrians and bicycles could be more commonly found. The analysis of these graphs gives us an idea of the risk values when our system will be deployed



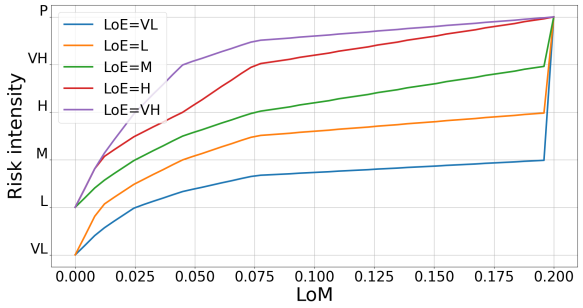
(a) Impact=VL



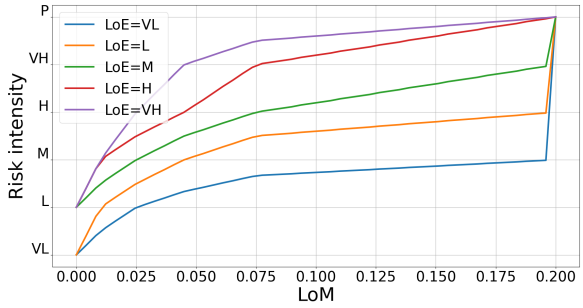
(b) Impact=L



(c) Impact=M



(d) Impact=H



(e) Impact=VH

Figure 3.7: The effect of varying the impact at different LoE levels.

in different contexts and whether our approach will be sufficient to mitigate the risk, or further mitigation strategies should be considered, e.g., to reduce impact or likelihood of encounter.

When our method is constrained because the LoM can not be decreased at least one risk level below the current one, other mitigation strategies can be put in place, e.g. to reduce the LoE of deer by placing fences around the road or banning pedestrians from highways to avoid the risk of them being misclassified as any other class and lead to an accident. As we already mentioned these charts allow to have this conversation, make an analysis and make decisions based on the likelihood of misclassification thresholds, LoM values, and how the impact and LoE combine together and determine the behaviour of the risk which is required to be understood, particularly, because of the sophisticated risk mapping that we are using.

Maximum acceptable risk level We evaluated our approach in this data set using two acceptable risk levels: $\tau = M$ and $\tau = H$ (i.e. mitigate all risks $> M$ and $> H$ respectively).

Table 3.5: Mean fraction of misclassification $p(i, j)$ for the evaluation of the 30 risk-oblivious models on the CIFAR-10 dataset. The values on the diagonal are 0 because we excluded the cases in which the model correctly identified the classes.

		Predicted Class									
		0	1	2	3	4	5	6	7	8	9
Actual Class	0	0.0	0.0104	0.0378	0.0174	0.0164	0.0058	0.0109	0.0093	0.0469	0.0245
	1	0.0102	0.0	0.0023	0.0057	0.002	0.003	0.0121	0.0018	0.0179	0.0602
	2	0.055	0.0025	0.0	0.0476	0.0847	0.0475	0.0631	0.0151	0.0088	0.0065
	3	0.0187	0.0027	0.0489	0.0	0.0543	0.1373	0.0805	0.0271	0.0091	0.0093
	4	0.0097	0.0021	0.041	0.0459	0.0	0.0227	0.0461	0.0325	0.0072	0.0019
	5	0.0051	0.0012	0.0285	0.1324	0.0393	0.0	0.0273	0.0339	0.0015	0.0034
	6	0.0068	0.0011	0.0255	0.044	0.0245	0.015	0.0	0.0027	0.002	0.002
	7	0.0112	0.0005	0.0212	0.0415	0.0526	0.0419	0.0107	0.0	0.0037	0.0089
	8	0.0449	0.0147	0.0073	0.0162	0.0057	0.0043	0.0081	0.0034	0.0	0.016
	9	0.0223	0.0391	0.0069	0.0098	0.0035	0.0023	0.0066	0.0038	0.0239	0.0

3.3.2.2 Step 1: Risk-oblivious training

To create the models for our experiments we used a convolution neural network with the structure based on those developed by the Keras team [26]. We trained 30 models with an average F1 score of 0.7907 and average residual risk of 17.5758, for 100 epochs. The time required to train one model was around 4.52 hours.

3.3.2.3 Step 2: Risk-aware verification

The mean *fLoM* and *fOL* values for the 30 risk-oblivious models are shown in Table 3.6 and in Table 3.7, respectively. The values from these tables indicate the respective parameter values on a 0-4 scale corresponding to VL-VH *plus* a fractional part that will be explained below. The *LoM* value is the mapped fraction of misclassification (see Table 3.5) for each of the classes; note that we require it to be scaled because the mapping table *OL* (Table 3.1) expects both the *LoM* and the *LoE* in the 0-4 scale of VL-VH. The *fOL* (Table 3.7) is the result of combining the *fLoM* and the *LoE* in the mapping Table 3.1. The dashes indicate class pairs, e.g. the actual class airplane (0) predicted as either, class bird (2), cat (3), deer (4) or dog (5), that are not of interest for our context.

Table 3.8 shows the final mapped risk, which is the result of combining the *fOL* and the *impact* as indicated earlier in Table 3.2. The shaded cells correspond to class pairs with a risk value greater than or equal to $\tau = 2$ (note that for the experiments for $\tau = 3$, a subset of these shaded cells indicate the risks that need to be mitigated). In total, there are 21 risk concerns for $\tau = 2$, and their corresponding risk values are: 1) frog (6) predicted as bird (2) with risk 2.005; 2) plane (0) predicted as car (1) with risk 2.01; 3) car (1) predicted as plane (0) with risk 2.01; 4) bird (2)

Table 3.6: Mean *fLoM* from the evaluation of the 30 risk-oblivious models on the CIFAR-10 data set, the values are provided on a 0-4 scale of VL-VH. The cells with a dash are misclassifications considered irrelevant for the context.

		Predicted Class									
		0	1	2	3	4	5	6	7	8	9
Actual Class	0	-	1.02	-	-	-	-	-	-	3.06	1.96
	1	1.01	-	-	-	-	-	-	-	1.52	3.5
	2	-	-	-	3.08	4.07	3.08	3.6	1.34	-	-
	3	-	-	3.13	-	3.31	4.49	4.04	2.1	-	-
	4	-	-	2.8	3.03	-	1.84	3.03	2.37	-	-
	5	-	-	2.17	4.45	2.71	-	2.11	2.44	-	-
	6	-	-	2.02	2.95	1.96	1.33	-	0.27	-	-
	7	-	-	1.74	2.82	3.25	2.84	1.04	-	-	-
	8	2.99	1.31	-	-	-	-	-	-	-	1.4
	9	1.81	2.7	-	-	-	-	-	-	1.92	-

Table 3.7: Mean *fOL* for the evaluation of the 30 risk-oblivious models on the CIFAR-10 data set, the values are provided on a 0-4 scale of VL-VH. The cells with a dash are misclassifications considered irrelevant for the context.

		Predicted Class									
		0	1	2	3	4	5	6	7	8	9
Actual Class	0	-	2.01	-	-	-	-	-	-	3.06	2.48
	1	2.01	-	-	-	-	-	-	-	2.52	4.25
	2	-	-	-	4.04	4.54	4.04	4.3	2.34	-	-
	3	-	-	2.56	-	2.66	3.49	3.04	2.05	-	-
	4	-	-	3.8	4.01	-	2.84	4.01	3.37	-	-
	5	-	-	3.17	4.72	3.71	-	3.11	3.44	-	-
	6	-	-	2.01	2.48	1.98	1.66	-	1.14	-	-
	7	-	-	1.37	1.91	2.12	1.92	1.02	-	-	-
	8	1.33	0.66	-	-	-	-	-	-	-	0.7
	9	2.81	3.7	-	-	-	-	-	-	2.92	-

predicted as dog (5) with risk 2.04; 5) horse (7) predicted as deer (4) with risk 2.06; 6) plane (0) predicted as truck (9) with risk 2.24; 7) deer (4) predicted as dog (5) with risk 2.42; 8) truck (9) predicted as ship (8) with risk 2.46; 9) plane (0)

Table 3.8: *frisk* for the class-pairs on the CIFAR-10 data set, the values are provided on a 0-4 scale of VL-VH. The cells with a dash are misclassifications considered irrelevant for the context. The shaded cells indicate the concerns with risk > 2 (M).

		Predicted Class									
		0	1	2	3	4	5	6	7	8	9
Actual Class	0	-	2.01	-	-	-	-	-	-	2.53	2.24
	1	2.01	-	-	-	-	-	-	-	1.51	1.62
	2	-	-	-	3.04	1.77	2.04	1.65	1.45	-	-
	3	-	-	0.86	-	1.55	1.25	1.68	1.35	-	-
	4	-	-	2.9	4.01	-	2.42	3.01	3.37	-	-
	5	-	-	3.17	4.72	3.71	-	3.11	2.72	-	-
	6	-	-	2.0	1.49	1.33	0.55	-	1.14	-	-
	7	-	-	1.37	1.3	2.06	1.31	0.34	-	-	-
	8	1.33	0.66	-	-	-	-	-	-	-	0.7
	9	2.81	3.7	-	-	-	-	-	-	2.46	-

predicted as ship (8) with risk 2.53; 10) dog (5) predicted as horse (7) with risk 2.71; 11) truck (9) predicted as plane (0) with risk 2.81; 12) deer (4) predicted as bird (2) with risk 2.9; 13) deer (4) predicted as frog (6) with risk 3.01; 14) bird (2) predicted as cat (3) with risk 3.04; 15) dog (5) predicted as frog (6) with risk 3.11; 16) dog (5) predicted as bird (2) with risk 3.17; 17) deer (4) predicted as horse (7) with risk 3.37; 18) truck (9) predicted as car (1) with risk 3.7; 19) dog (5) predicted as deer (4) with risk 3.71; 20) deer (4) predicted as cat (3) with risk 4.01; and 21) dog (5) predicted as cat (3) with risk 4.72.

Example 3.3.1 *We present an example of the risk calculation for the concern 'dog (class 5) misclassified as cat (class 3)' using the intervals previously defined and the formulas introduced in Section 3.2.4.3.*

1. *Obtain fLoM. The concern dog(5) as cat(3) has a* $p(i, j) = 0.1324$ *(see Table 3.5), which belongs to the interval* $VH : [0.075, 0.2)$*; substituting in* (3.10)*, we compute:*

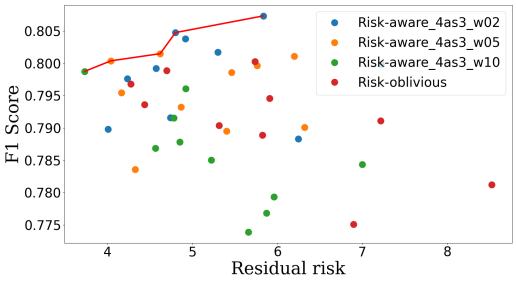
$$fLoM = 4 + \frac{0.1324 - 0.075}{0.2 - 0.075}$$

and we obtain $fLoM = 4.45$.

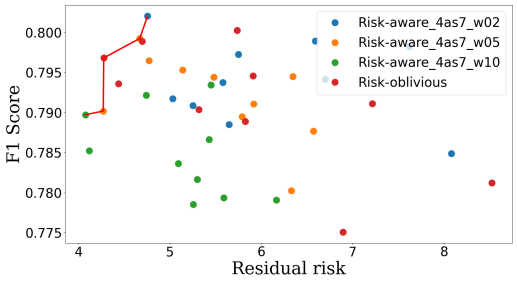
2. *Obtain fOL. Continuing with our example, we have* $LoM = 4$, $LoE = 4$, *and* $row = [L(1)\ M(2)\ H(3)\ VH(4)\ VH(4)]$. *so* $first = 3$ *and* $last = 4$, *and substituting in* (3.11) *we obtain* $sz = 2$. *Using* (3.12) *and substituting, we compute:*

$$fOL = 4 + \frac{4.45 - 3}{2}$$

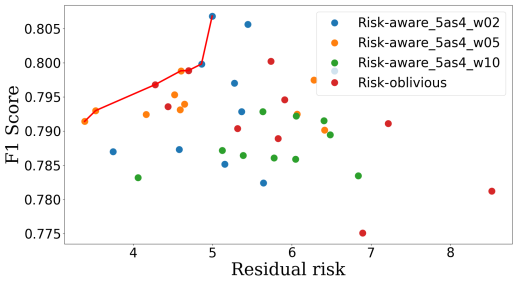
and get $fOL = 4.725$ *(see Table 3.7).*



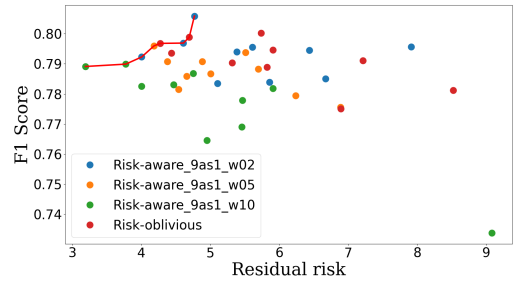
(a) Models for the concern deer (4) as cat (3)



(b) Models for the concern deer (4) as horse (7)



(c) Models for the concern dog (5) as deer (4)



(d) Models for the concern truck (9) as car (1)

Figure 3.8: Risk-oblivious vs risk-aware models for four of the concerns for the evaluation on the CIFAR-10 data set.

3. *Finally, we calculate the frisk. As impact* $= 4(VH)$ *and* $fOL = 4.725(VH)$*, then column* $= [VL(0)\ L(1)\ M(2)\ H(3)\ VH(4)]$*. We calculate* sz *using* (3.11)*, where* $first = 4$ *and* $last = 4$ *(for the specific fOL and impact):*

$$sz = 4 - 4 + 1 = 1.$$

Using (3.13) *and substituting, we calculate:*

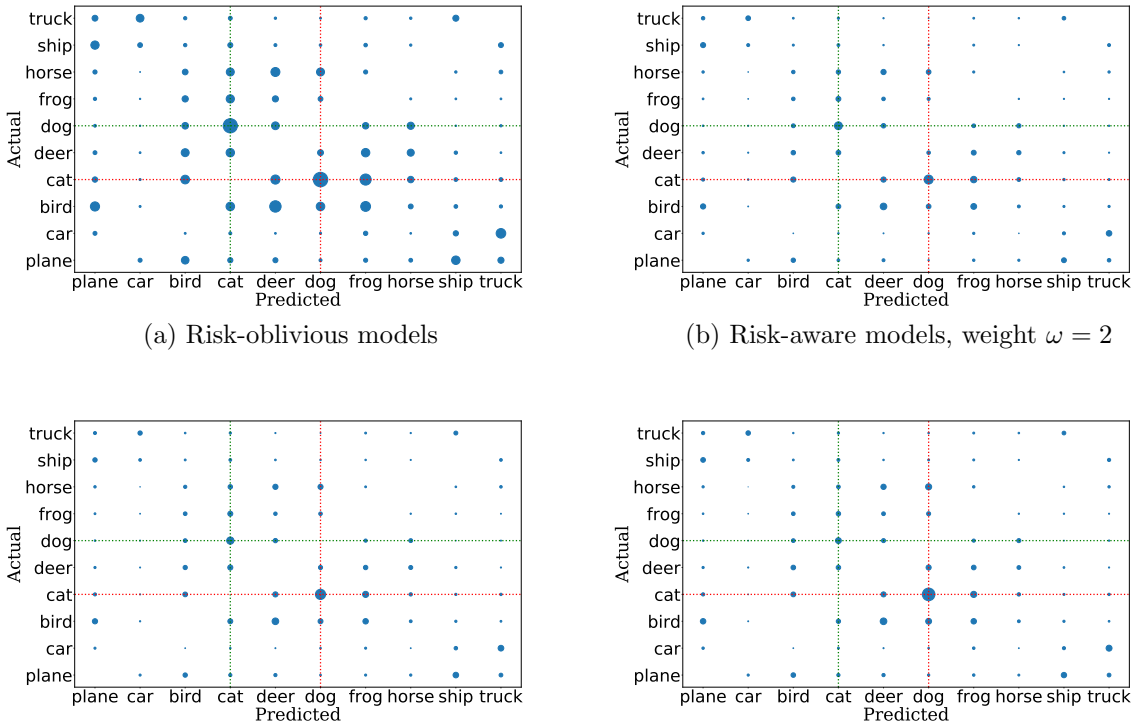
$$frisk = 4 + \frac{4.725 - 4}{1}$$

and obtain the fractional frisk $= 4.725$*. This can be corroborated in Table 3.8 for the concern dog(5) as cat(3).*

For each τ value we obtained a different number of concerns. For $\tau = M$ see Table 3.9, and for $\tau = H$ see Table 3.13 for the list of concerns and their corresponding risk values.

3.3.2.4 Step 3: Risk-aware training

We trained 30 models per concern identified in the previous step, with 10 such models built for each $\omega \in \{2, 5, 10\}$ (giving a total of 630 risk-aware models for $\tau = M$, and 270 risk-aware models for $\tau = H$) with an average F1 score of 0.7946 and average residual risk of 18.0412. If we compare these values with those from the risk-oblivious models, we can see that on average the risk-aware models have more residual risk. However, when we carefully compare the risk-aware models for each specific concern to the risk-oblivious models, the risk-aware ones have lower



(a) Risk-oblivious models

(b) Risk-aware models, weight $\omega = 2$

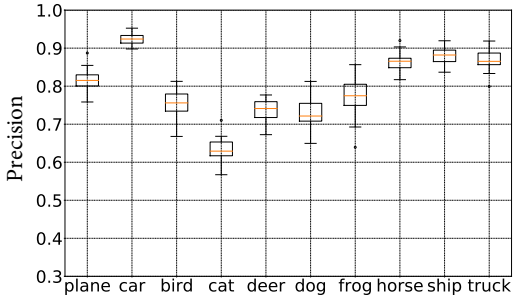
(c) Risk-aware models, weight $\omega = 5$

(d) Risk-aware models, weight $\omega = 10$

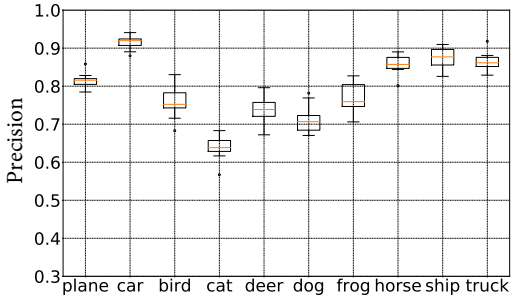
Figure 3.9: The effect of applying weight to the concern dog predicted as cat on misclassification. The size of the marker is proportional to the number of misclassifications observed, the green lines intersect the concern e.g. class dog predicted as cat and the red lines the inverted classes of the concern e.g. cat predicted as dog to visualise the effect of weight in both situations.

risk levels for the concern they are "aware" of than the risk-oblivious. This is the expected outcome, i.e. the risk-aware models are expected to dominate in terms of F1 and residual risk. This result is depicted in Figure 3.8, which shows the F1 and residual risk for the generated models, distinguishing between $\omega = \{2, 5, 10\}$ and risk-oblivious for four randomly chosen concerns. For example, in Figure 3.8a the risk-aware model with $\omega = 10$ (green dot in the Pareto-front) is the best performing model in terms of risk with a residual risk below 4 and F1 close to 0.8. We also have in the Pareto-front two risk-aware models for $\omega = 5$ (yellow dots) and two risk-aware models for $\omega = 2$ (blue dots). We also notice in Figures 3.8b, 3.8c and 3.8d that at least one of the risk-oblivious models (red dots) made it to the Pareto-front; nonetheless, most of the Pareto-dominating models belong to the three sets of risk-aware models. Furthermore, having good risk-oblivious models is of benefit for the ensemble, as we will be combining the specialist (risk-aware) models with good-performing generalist models to account for the misclassifications without risk-aware models.

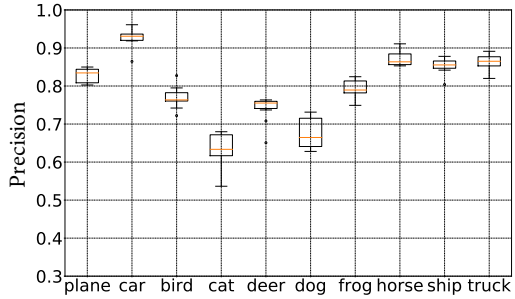
For each model created we examined the confusion matrix and the number of misclassifications for the test set were recorded. The results are plotted for the concern dog as cat in Figure 3.9 where we compare the effect of the weighting ω used to train the models. As shown in this figure, the tendency is for misclassification in the risk-aware models to be reduced as the weight is increased. This reduction



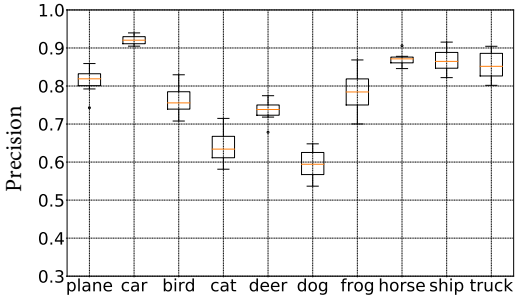
(a) Risk-oblivious models



(b) Risk-aware models, weight=2



(c) Risk-aware models, weight=5



(d) Risk-aware models, weight=10

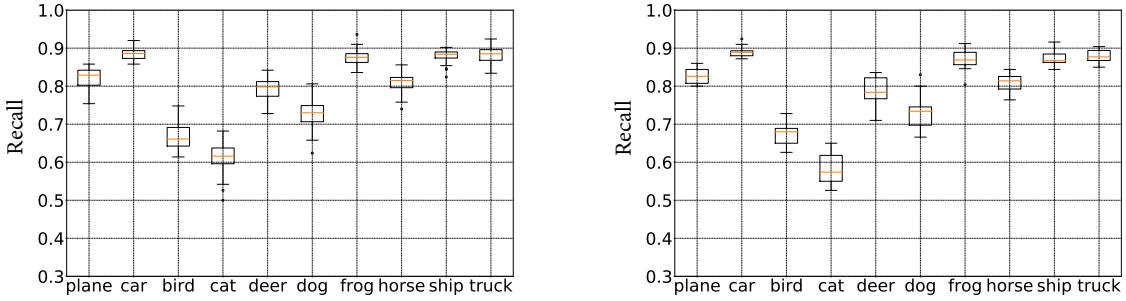
Figure 3.10: The effect of applying weight to the concern dog predicted as cat on precision.

in misclassification rate is not *cost-free* however, and an examination of the whole model reveals that a trade off is necessary between controlling the risk associated with this mode of misclassification and performance metrics. The size of the marker is proportional to the number of misclassifications observed. For clarity we omit predictions correctly made. Here we see that, the size of the marker associated with misclassifying a dog as a cat reduces as the weight increases. It is clear however that we now more frequently misclassify a cat as a dog.

Figure 3.10 shows an examination of the precision associated with each class for the model sets. We can see from this plot that as the weight is increased the precision of the dog class drops. This means that we are obtaining a higher number of false positives since we are more willing to accept a cat classified as a dog. When we examine recall, as shown in Figure 3.11, we note that recall for the cat class drops as the weight increases. This is due to an increase in the false negative rate, i.e. the number of cats which are not correctly identified by the model. Indeed, with a weight of 10 the recall for cats has dropped to approximately 0.47.

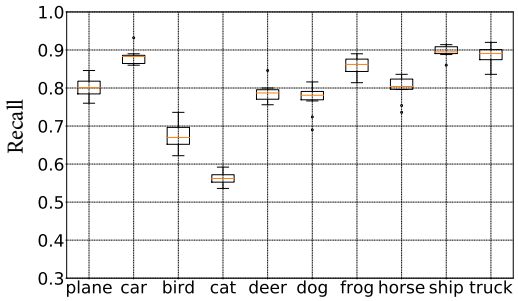
3.3.2.5 Step 4: Risk-aware ensemble synthesis and verification

As mentioned in Section 3.3.2.1, we used two different values of τ ($\tau = M$ and $\tau = H$), to assess whether our approach is able to mitigate risk at different thresholds. In this section we show the results that we obtained for each of the defined thresholds of acceptable risk values.

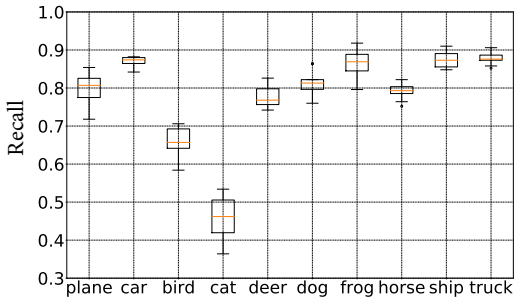


(a) Risk-oblivious models

(b) Risk-aware models, weight=2



(c) Risk-aware models, weight=5



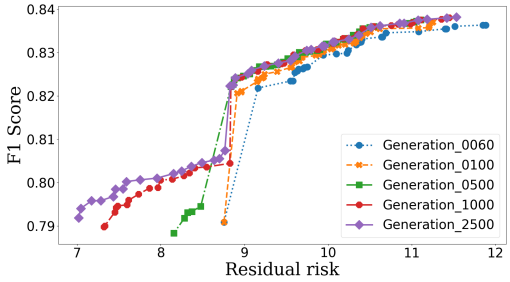
(d) Risk-aware models, weight=10

Figure 3.11: The effect of applying weight to the concern dog predicted as cat on recall.

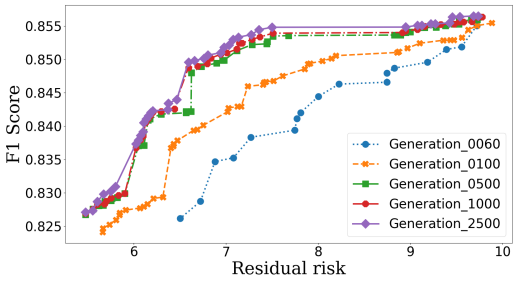
Ensemble synthesis for $\tau = M$. We synthesised three different risk-aware ensembles, each comprising a different number of allowed models l. We first randomly selected eight models for each of the 21 concerns obtained at this τ value (see Table 3.9) and eight of the risk-oblivious ones from the F1/residual-risk Pareto front for each type of model, obtaining a set of $21 \times 8 + 8 = 176$ models. These models were used as input for the GA to synthesise a) an ensemble allowing $l = 2$ models; b) an ensemble allowing $l = 5$ models; and c) and ensemble allowing $l = 10$ models. We did it in this way to assess whether increasing the size of the ensemble leads to ensembles with higher F1 score capable of mitigating the risk concerns.

Figure 3.12 summarises the results obtained in this experiment. First, Figures 3.12a, 3.12b and 3.12c show the Pareto fronts of the ensemble learning at the different sizes of the ensemble, i.e. 2, 5 and 10 models. Each of these three graphs displays in different colours the results obtained at 60 (blue), 100 (orange), 500 (green), 1000 (red) and 2500 (purple) generations. The F1 score of the ensemble classifiers increases and the residual risk is reduced as more GA generations are used for the ensemble synthesis. For example, in Figure 3.12b we can observe how at 60 generations (blue Pareto front) the minimum residual risk achieved is around 6.5 and the F1 score just above 0.825, while at 100 generations (orange Pareto front) the residual risk has been reduced to roughly 5.5 with nearly the same F1 score. We also note that this improvement is less and less noticeable (the principle of diminishing returns), which suggests that at 1000 generations we approximate the best solution that the GA can produce.

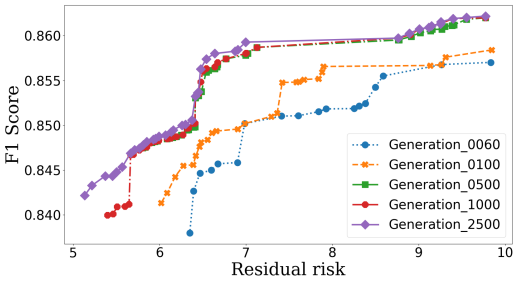
Figure 3.12d compares the three Pareto fronts obtained from the different sizes



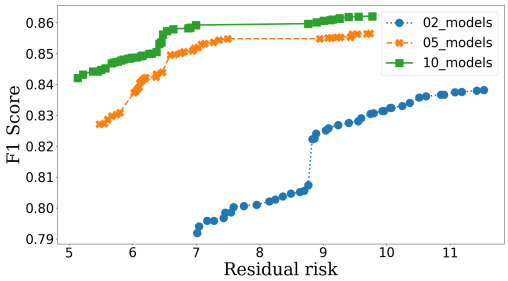
(a) Ensemble 2 models



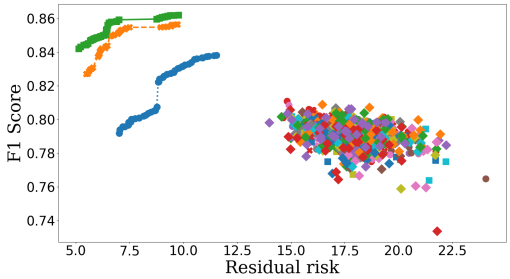
(b) Ensemble 5 models



(c) Ensemble 10 models



(d) Comparing the 3 ensemble sizes at 2500 generations



(e) Comparing single models(risk-oblivious and risk-aware) vs the 3 ensemble sizes

Figure 3.12: Ensemble learning history for different sizes of the ensemble when $\tau = M$ on the CIFAR-10 dataset.

of the ensemble at 2500 generations i.e. the best set of solutions found by the GA. The Pareto front in blue for the ensemble of 2 models, in orange the ensemble of 5 models and in green the ensemble of 10 models. It can be appreciated how as more models are allowed in the ensemble the solutions move towards the top left of the graph, meaning that the residual risk is reduced and the F1 score augmented. However, we acknowledge that as the number of models grows, so does the required time to synthesise the ensemble, which is shown in Table 3.10; the ensemble of 2 models required 2.68 hours, then the ensemble of 5 models doubled the time to 5.47 hours and finally, the ensemble of 10 models demanded 14.5 hours. As we can see the quality of the solution is a quid pro quo for the time required to synthesise the ensemble.

Finally, in Figure 3.12e we compare the Pareto fronts for the three ensemble sizes with the individual risk-oblivious and risk-aware models. Clearly, all ensemble

Table 3.9: Results for the CIFAR-10 data set at the different sizes of the ensemble when $\tau = M$ during training of the ensemble. The risk values correspond to the solution in the Pareto front with the smallest amount of risk when 10 models are allowed in the ensemble after 2500 generations. The risk values in red show the concerns that were not mitigated below the maximum acceptable risk $\tau = M$ (2).

Id	Concern		Initial risk	Risk ensemble 2 models	Risk ensemble 5 models	Risk ensemble 10 models
	Actual	Predicted				
1	Frog (6)	Bird (2)	2.005	1.5	1.63	1.96
2	Plane (0)	Car (1)	2.01	1.8	1.4	1.4
3	Car (1)	Plane (0)	2.01	1.8	1.6	1.8
4	Bird (2)	Dog (5)	2.04	2.38	1.98	2.05
5	Horse (7)	Deer (4)	2.06	2.32	2.19	2.12
6	Plane (0)	Truck (9)	2.24	2.38	2.31	2.16
7	Deer (4)	Dog (5)	2.42	2.06	2	2.13
8	Truck (9)	Ship (8)	2.46	2.06	1.4	2.2
9	Plane (0)	Ship (8)	2.53	2.29	2.2	2.31
10	Dog (5)	Horse (7)	2.71	2	2.33	1.6
11	Truck (9)	Plane (0)	2.81	2.13	2.13	2.13
12	Deer (4)	Bird (2)	2.9	2.26	2.13	2.2
13	Deer (4)	Frog (6)	3.01	2.26	2.33	1.8
14	Bird (2)	Cat (3)	3.04	2	2.13	2.4
15	Dog (5)	Frog (6)	3.11	2.93	2	2
16	Dog (5)	Bird (2)	3.17	2	2	2
17	Deer (4)	Horse (7)	3.37	1.8	2.13	2
18	Truck (9)	Car (1)	3.7	1.8	2.13	2.26
19	Dog (5)	Deer (4)	3.71	4.48	4.31	3.15
20	Deer (4)	Cat (3)	4.01	1.6	1.8	2
21	Dog (5)	Cat (3)	4.72	3.45	3.15	4.01
Residual risk			18.035	7.018	5.475	5.13

sizes achieved much lower risk and typically higher F1 score. The exception is the Pareto front for the ensembles allowing two models, which includes ensembles whose F1 score is lower than that of some of the individual models. However, even for this small ensemble size, there is a trade-off between the F1 score and the risk, i.e., what amount of F1 score is given up in exchange for risk mitigation.

A detailed summary of the results is presented in Table 3.9, which compares the initial amount of risk for each of the 21 concerns obtained at this τ value with the amount of risk obtained at each size of the ensemble during its training. The risk values correspond to the solution in the Pareto front with the smallest amount of

Table 3.10: F1 score achieved and time required to train the ensemble using the GA for 2500 generations when $\tau = M$ at the smallest residual risk.

Ensemble 2 models		**Ensemble 5 models**		**Ensemble 10 models**	
F1 score	**Required Time (in hours)**	**F1 score**	**Required Time (in hours)**	**F1 score**	**Required Time (in hours)**
0.7919	2.68	0.8271	5.47	0.8422	14.5

risk achieved when 10 models are allowed in the ensemble after 2500 generations. We coloured in red the risk values with the concerns where the risk is above the maximum accepted $\tau \geq M$ (2). At the bottom of the table we compare the residual risk values for the different sizes of the ensemble and we can see how it decreased as more models were allowed in the ensemble. We also notice that despite a significant reduction in the residual risk, from 18.035 of the initial concerns to 7.0175 for the ensemble of 2 models, 5.475 for the ensemble of 5 models and 5.13 for the ensemble of 10 models, not all concerns were mitigated below the acceptable risk level τ. In fact, at all sizes of the ensemble the GA failed to bring down the risk for 12 concerns out of 21; however, most of these concerns were mitigated to some extent. For instance, the majority of the concerns with a risk of H (3) were reduced to at least a risk of M (2), we are referring specifically to the concerns 13) deer predicted as frog, 14) bird predicted as cat, 15) dog predicted as frog, 16) dog predicted as bird, 17) deer predicted as horse, and 18) truck predicted as car.

To understand this behaviour, remember that the objective of the genetic algorithm is to minimise the amount of residual risk as much as possible and not only to mitigate the risk below the proposed threshold τ i.e. the GA will choose the appropriate set of DNN models and weights to satisfy this objective and it will not stop when the risk for individual concerns has been mitigated because the algorithm is dealing with all the concerns at the same time. Our method only alleviates risk by lowering the LoM, and despite a great improvement in terms of risk mitigation, getting the very low misclassification rates require for the complete mitigation of all risk concerns, which is unfeasible. These limitations are not specific to our approach but are affecting all machine learning approaches. These problems include: the complexity of the data, limited training data, overfitting and generalisation problems, incomplete feature representation, and inherent localisation and identification. Being aware of these problems helps to understand that the chosen models and weights by the GA will in some cases achieve the objective τ but in other cases (despite a reduction in the LoM) the risk level will not be lowered below τ.

If we examine these results carefully, we can also notice that some concerns got worse. For instance the concern 5) horse predicted as deer has an initial risk value of 2.06, and for the 3 different sizes of the ensemble a risk of 2.32, 2.19 and 2.12. Similarly, the concern 4) bird predicted as dog initially has a risk value of 2.04 and for the ensemble of 2 and 10 models the risk is elevated to 2.38 and 2.05, respectively; also the concern 19) dog predicted as deer has an initial risk of 3.71 and it is raised to 4.48 and 4.31 in the ensemble of 2 and 5 models correspondingly; nevertheless, we should not forget the residual risk, in order for the GA to accomplish decreasing the risk for some concerns others will rise, and the GA will affect some of them in

this way for the benefit of decreasing the residual risk, which is considering not a single concern but all the concerns with risk values above the acceptable threshold e.g. among others, increasing the risk from 2.06 to 2.32 for the concern 5) horse predicted as deer when 2 models are allowed in exchange of mitigating the risk from 4.01 (VH) to 1.6 (L) for the concern 20) deer predicted as cat, and also the concern 18) truck predicted as car where the risk was mitigated from 3.7 (H) to 1.8 (L). In general we can think of the concerns as a mattress for which whenever some of the springs are pressed down, others will come up. This analogy, we think, helps to understand how the GA combined the models and weights to obtain the overall best solution.

We can also notice in Table 3.10 that all ensemble sizes achieved a better F1 score (at the minimum residual risk) than the risk-oblivious models; 0.7919, 0.8271, and 0.8422 vs 0.7907, which is the average F1 of the risk-oblivious models. For the risk-aware models, the F1 score was slightly higher than that of the ensemble allowing 2 models, 0.7946 for the risk-aware models vs 0.7919 for the ensemble allowing 2 models; however, the residual risk is where the benefit can be seen as it was mitigated from 18.035 to 7.018. Regarding the time, it is evident that as more models are allowed in the ensemble, more time is required to synthesise the solution.

As we mentioned at the beginning of this section, we kept half of the test set to test our final ensemble. The results of this testing are presented in Table 3.11, which shows a similar behaviour to that obtained during training, i.e. the tendency for the residual risk is to decrease as more models are allowed in the ensemble. We notice how the residual risk values obtained in the test set are higher to those obtained during training; however, there is a significant improvement when we compare the initial residual risk with the residual risk at the different sizes of the ensemble e.g. the residual risk was lowered from 18.035 to 8.4424 for the ensemble of two models, 7.8949 for the ensemble of five models and 7.6724 for the ensemble of 10 models, which demonstrates the benefit of our approach. We also observe a variation in the level of risk for each concern when using ensembles of different capacity. For example, the concern with id 8 (i.e. truck misclassified as ship) in Table 3.11 has risk values of 2.065, 1.4 and 2.065 for the ensembles allowing two, five and 10 models, respectively. This behaviour is also observed in Table 3.15, for example for the concerns with ids 5 and 8. This variation in risk values for the same concern in different ensemble sizes is expected because for each size of the ensemble different individual risk-oblivious and risk-aware models are proposed and selected, and also the sets of weights the GA will choose is different each time. Rather than focusing on individual risk concerns, the GA aims to minimise the residual risk and at each iteration of the GA a new set of models and weights is proposed. Finally, since the GA performs a stochastic search, it is not guaranteed that the same solution will be found in different runs of the ensemble synthesis even when considering the same number of individual models.

Likewise, we present Table 3.12, which shows the F1 score achieved at each size of the ensemble when it was evaluated on the test data set. As expected, the F1 score increases with the size of the ensemble, from 0.7862 for the ensemble of 2 models to 0.8236 for the ensemble of 5 models, and to 0.8376 for the ensemble of 10 models.

This concludes the analysis of the results obtained for $\tau = M$ on the CIFAR-10 data set, the following fragment summarises the outcome for the experiments using

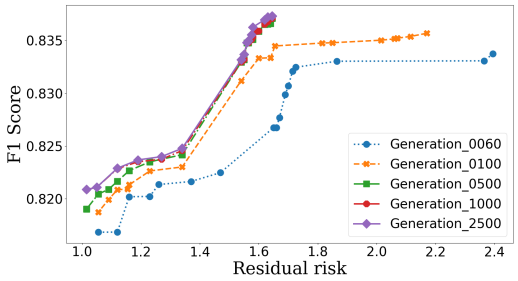
Table 3.11: Results for the CIFAR-10 data set at the different sizes of the ensemble when $\tau = M$ in the test dataset. The risk values correspond to the solution in the Pareto front with the smallest amount of risk when 10 models are allowed in the ensemble after 2500 generations. The risk values in red show the concerns that were not mitigated below the maximum acceptable risk $\tau = M$ (2).

Id	Concern		Initial risk	Risk ensemble 2 models	Risk ensemble 5 models	Risk ensemble 10 models
	Actual	Predicted				
1	Frog (6)	Bird (2)	2.005	1.565	1.7	1.565
2	Plane (0)	Car (1)	2.01	1.8	1.4	1.6
3	Car (1)	Plane (0)	2.01	1.8	1.6	2.13
4	Bird (2)	Dog (5)	2.04	2.5	2.215	2.415
5	Horse (7)	Deer (4)	2.06	2.28	2.175	2.1575
6	Plane (0)	Truck (9)	2.24	2.3625	2.3625	2.2325
7	Deer (4)	Dog (5)	2.42	2.13	2.13	2.13
8	Truck (9)	Ship (8)	2.46	2.065	1.4	2.065
9	Plane (0)	Ship (8)	2.53	2.385	2.1975	2.2875
10	Dog (5)	Horse (7)	2.71	2	2	2
11	Truck (9)	Plane (0)	2.81	1.4	1.6	1.8
12	Deer (4)	Bird (2)	2.9	1.8	2.13	2.265
13	Deer (4)	Frog (6)	3.01	2.465	2.395	2.13
14	Bird (2)	Cat (3)	3.04	2	2.065	2.2
15	Dog (5)	Frog (6)	3.11	3.35	2	1.6
16	Dog (5)	Bird (2)	3.17	2.13	3.04	2.26
17	Deer (4)	Horse (7)	3.37	1.8	2.13	1.8
18	Truck (9)	Car (1)	3.7	2.26	2.66	3.04
19	Dog (5)	Deer (4)	3.71	4.5	4.315	4.08
20	Deer (4)	Cat (3)	4.01	2	1.8	1.8
21	Dog (5)	Cat (3)	4.72	4.015	4.08	4.28
Residual risk			18.035	8.4424	7.8949	7.6724

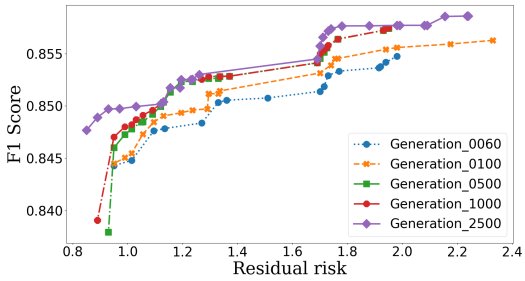
Table 3.12: F1 score at different sizes of the ensemble obtained when the ensemble was evaluated in the test dataset using $\tau = M$.

F1 ensemble 2 models	F1 ensemble 5 models	F1 ensemble 10 models
0.7862	0.8236	0.8376

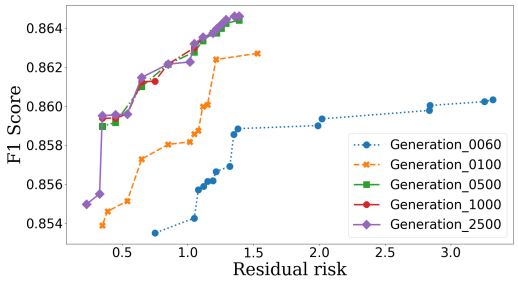
the threshold of acceptable risk $\tau = H$ on the same data set.



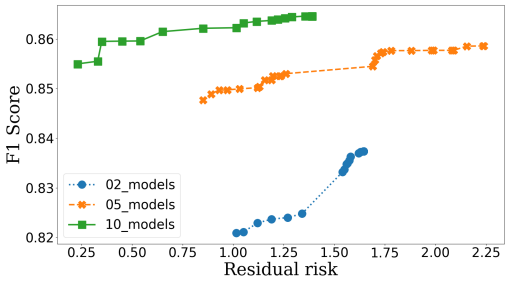
(a) Ensemble 2 models



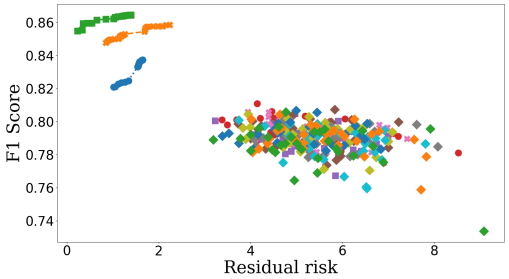
(b) Ensemble 5 models



(c) Ensemble 10 models



(d) Comparison of the 3 ensemble sizes at 2500 generations



(e) Comparison of single models (risk-oblivious and risk-aware) to the 3 ensemble sizes (at 2500 generations)

Figure 3.13: Ensemble learning history for the different sizes of the ensemble when $\tau = H$ on the CIFAR-10 dataset.

Ensemble synthesis for $\tau = H$. In this part of the section we describe the results obtained for $\tau = H$. The list of the nine concerns obtained is presented in Table 3.13. As can be seen, this is a subset of the concerns obtained for $\tau = M$. Again, we synthesised three different risk-aware ensembles varying the number of allowed models l in each of them. We first randomly selected eight models for each of the concerns obtained at this τ value (see Table 3.13) and eight of the risk-oblivious ones from the F1/residual-risk Pareto front for each type of model, obtaining a set of 80 models. These 80 models were given to the GA to synthesise ensembles allowing two, five and ten models.

In Figure 3.13 we present the obtained results. Figures 3.13a, 3.13b and 3.13c show the Pareto-optimal ensembles found for the ensemble allowing 2, 5 and 10 models, respectively. For each of them, the Pareto front was plotted at 60, 100, 500,

1000 and 2500 generations. In all three plots, the same pattern is evident: as more generations of the GA pass by, the Pareto front improves, finding new solutions with lower residual risk and higher F1 score. For example, this improvement is clearly distinguishable in Figure 3.13c, where the 60-generation (blue) Pareto front has a residual risk of around 5.5, the 100-generation (yellow) Pareto front has a residual risk of approximately 4.8, and the 2500-generation (purple) Pareto front has a residual risk of roughly 4.1. We also witness an increase in the F1 score from around 0.853 at 60 generations to almost 0.856 at 2500 generations. Additionally, we spot how, in all three graphs (Figures 3.13a, 3.13b and 3.13c), the improvement is increasingly difficult to achieve from generations 500 to 2500: the law of diminishing returns applies.

Similarly to our previous results for $\tau = M$, in Figure 3.13d we compare the three sizes of the ensemble at 2500 generations of the GA. There are no new insights to report, yet these results confirm our previous finding that, as more models are allowed in the ensemble the quality of the solution found is superior in terms of both residual risk and F1 score.

Finally, in Figure 3.13e we compare the individual risk-aware and risk-oblivious models to the 3 ensemble sizes we synthesised. Again, it is possible to appreciate how the residual risk of the individual models is comparably high to that of the ensembles. The residual risk of the risk-oblivious and risk-aware models ranges between around 2.5 and 8, whereas the residual risk for the Pareto fronts is between 0 and 2, which is a significant risk reduction. In addition, all Pareto-front ensembles achieved higher F1 scores than the individual models, the better individual model obtained an F1 of almost 0.82 while the ensemble of 10 models is very close to 0.86—again, a considerable improvement.

In Table 3.13, we present the risk values for each concern and compare it to the one obtained at each ensemble size. We chose the ensemble from the Pareto front with the lowest residual risk during the training of the ensemble. We also compare in the last row of the table the total residual risk. The red-coloured values are the concerns with a risk value $\tau \geq 3$ (H). Examples of these are: concern 9) dog predicted as cat with risk value of 4.01 for the ensemble allowing 2 models, 3.85 for the ensemble of 5 models, and 3.15 for the ensemble of 10 models; concern 6) truck predicted as car with a risk value of 3.04 for the ensemble of 10 models; and concern 7) dog predicted as deer also with a risk value of 3.4 for the ensemble of 10 models. In contrast to what we found for our previous τ value, in this case none of the concerns ended up with a risk value higher than the initial value; on the contrary, at all sizes of the ensemble the risk values obtained were reduced, including for the concern 9) dog misclassified as cat, for which the risk was mitigated from 4.72 to 4.01, 3.85 and 3.15 for sizes 2, 5 and 10 of the ensemble, respectively.

Unexpectedly, when 10 models are allowed in the ensemble we ended up with 3 concerns with residual risk $\tau \geq H(3)$. Looking carefully, we also realise how the GA found that by slightly increasing the risk in concerns 6) truck misclassified as car and 7) dog misclassified as deer (compared to sizes 2 and 5 of the ensemble), it was able to significantly decrease the risk for the concern 9) dog misclassified as cat, from 4.72 to 3.15, which is the lowest risk level recorded for that concern. We already discussed this behaviour of the GA in the previous evaluation for $\tau = M$, explaining that the GA always aims to improve the total residual risk rather than the number of concerns over τ; this is confirmed by how the residual risk was reduced from 4.84

Table 3.13: Results for the CIFAR-10 data set at the different sizes of the ensemble when $\tau = H$ during training of the ensemble. The risk values correspond to the solution in the Pareto front with the smallest amount of risk when 10 models are allowed in the ensemble after 2500 generations. The risk values in red show the concerns that were not mitigated below the maximum acceptable risk $\tau = H$ (3).

Id	Concern		Initial risk	Risk ensemble 2 models	Risk ensemble 5 models	Risk ensemble 10 models
	Actual	Predicted				
1	Deer (4)	Frog (6)	3.01	2.4	2.13	2.26
2	Bird (2)	Cat (3)	3.04	2.13	2.52	2.46
3	Dog (5)	Frog (6)	3.11	2.79	1.4	2
4	Dog (5)	Bird (2)	3.17	2.93	2.93	2.26
5	Deer (4)	Horse (7)	3.37	2.93	2.79	2.93
6	Truck (9)	Car (1)	3.7	2.66	2.66	3.04
7	Dog (5)	Deer (4)	3.71	2.93	2.66	3.04
8	Deer (4)	Cat (3)	4.01	2.66	2.93	2.4
9	Dog (5)	Cat (3)	4.72	4.01	3.85	3.15
Residual risk			4.84	1.015	0.85	0.23

to 0.23 when 10 models were allowed in the ensemble.

Table 3.14 shows the F1 score and the time required to synthesise the ensemble at sizes 2, 5 and 10. We appreciate how as more models are allowed in the ensemble the F1 score raises but also does the ensemble synthesis time, e.g. from 2.28 hours for the ensemble allowing 2 models to 13.18 hours for the ensemble of 10 models.

In Table 3.15 we present the results obtained by evaluating the ensembles on the test set. The results are as expected, with the residual risk shrinking as more models join the ensemble. We notice how in the test set we obtained a higher number of concerns with the risk above $\tau = 3$, for example in size 2 of the ensemble the concerns 3) dog predicted as frog, 4) dog predicted as bird, 5) deer predicted as horse, 6) truck predicted as car, 7) dog predicted as deer, and 9) dog predicted as cat have a risk above 3. All of them however, except for concerns 6) truck predicted as car and 7) dog predicted as deer, were mitigated with respect to the initial risk value, which is the expected behaviour. As for concerns 6) truck predicted as car and 7) dog predicted as deer have worsen their risk values from 3.7 and 3.71 to 3.85. Again when comparing the residual risk from initially 4.84 to 3.61 obtained in the ensemble of 2 models, we notice the overall risk reduction, which is the aim of our method and shows its benefit of significantly improving the residual risk. To conclude, we present Table 3.16, which shows the F1 score for the test data set increasing from 0.8102 to 0.8417 as the ensemble size grows.

Finally, in Figure 3.14 we present an example of the predictions made by the ensemble for 32 randomly selected images from the test set used to evaluate the ensemble of 10 models with $\tau = H$. The figure shows the predicted class, in brackets the actual class, and the image. We coloured in green the correct predictions and in red the misclassifications. This concludes the evaluation of our approach on the

Table 3.14: F1 score achieved and time required to train the ensemble using the GA for 2500 generations when $\tau = H$ at the smallest residual risk.

Ensemble 2 models		**Ensemble 5 models**		**Ensemble 10 models**	
F1 score	**Required Time (in hours)**	**F1 score**	**Required Time (in hours)**	**F1 score**	**Required Time (in hours)**
0.82088	2.28	0.8476	4.2	0.8549	13.18

Table 3.15: Results for the CIFAR-10 data set at the different sizes of the ensemble when $\tau = H$ in the test dataset. The risk values correspond to the solution in the Pareto front with the smallest amount of risk when 10 models are allowed in the ensemble after 2500 generations. The risk values in red show the concerns that were not mitigated below the maximum acceptable risk $\tau = H$ (3).

Id	**Concern**		**Initial risk**	**Risk ensemble 2 models**	**Risk ensemble 5 models**	**Risk ensemble 10 models**
	Actual	**Predicted**				
1	Deer (4)	Frog (6)	3.01	2.52	2.52	2.465
2	Bird (2)	Cat (3)	3.04	2.13	2.265	2.13
3	Dog (5)	Frog (6)	3.11	3.04	1.6	2
4	Dog (5)	Bird (2)	3.17	3.15	3.04	2.79
5	Deer (4)	Horse (7)	3.37	3.15	2.79	3.25
6	Truck (9)	Car (1)	3.7	3.85	3.15	3.08
7	Dog (5)	Deer (4)	3.71	3.85	3.85	3.75
8	Deer (4)	Cat (3)	4.01	2.26	3.25	2.26
9	Dog (5)	Cat (3)	4.72	4.52	4.215	4.215
Residual risk			4.84	3.61	2.505	2.295

CIFAR-10 data set. In the next section we present the evaluation of the approach for a subset of the German Traffic Sign Recognition Benchmark, a well-known data set used to perform image classification.

Table 3.16: F1 score at different sizes of the ensemble obtained when the ensemble was evaluated in the test dataset using $\tau = H$.

F1 ensemble 2 models	**F1 ensemble 5 models**	**F1 ensemble 10 models**
0.8102	0.8326	0.8417

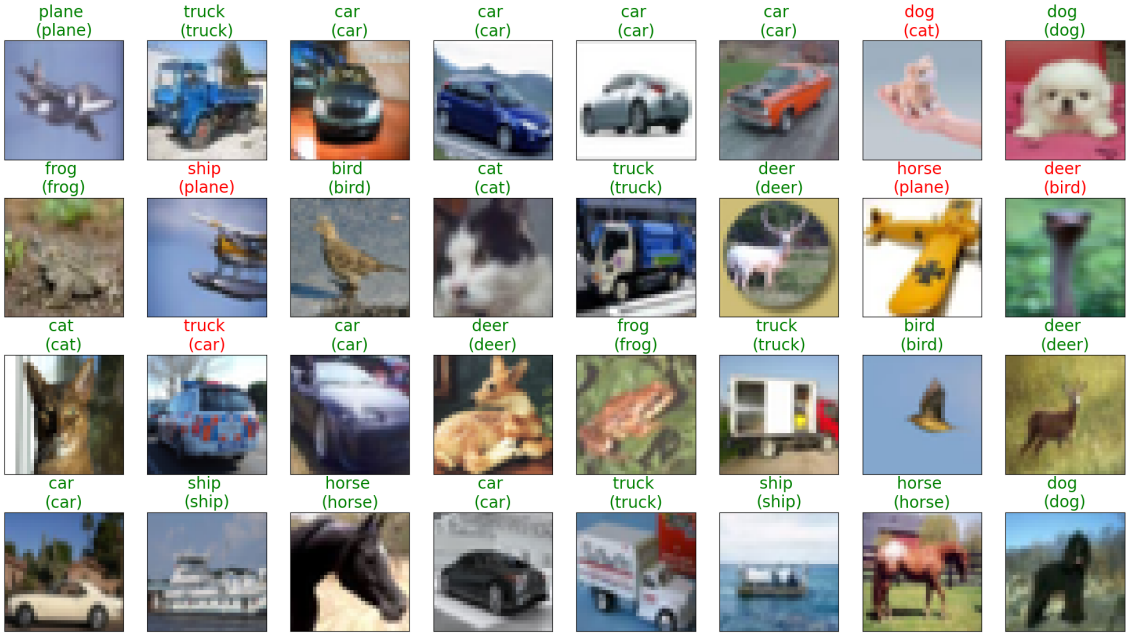


Figure 3.14: Predictions of the ensemble of 10 models with $\tau = H$ in the CIFAR-10 test dataset for the ensemble. In green the correct predictions and in red the incorrect ones, the actual class is in brackets.

3.3.3 Evaluation on the GTSRB dataset

3.3.3.1 Inputs

The data set For these experiments we used a subset of the GTSRB [141] dataset by selecting seven classes related to speed limits, plus a class *other* composed of random samples selected from the remaining 36 classes in the dataset. The resulting training set comprises 12920 samples and the test set has 4756 samples. These were then used to train and test the individual risk-oblivious and risk-aware models. We created a secondary test set for the ensemble by randomly selecting 100 samples for each class to obtain a total of 800 samples that were augmented with image pertaurbation to create a complete test set of 4756 samples. The augmentation applied a width shift, height shift and rotation using the Keras ImageDataGenerator API[2]. We defined the following class IDs: 0) speed limit 30km/h; 1) speed limit 50km/h; 2) speed limit 60km/h; 3) speed limit 70km/h; 4) speed limit 80km/h; 5) speed limit 100km/h; 6) speed limit 120km/h; 7) Other.

Risk information

Impact level $impact(i, j)$ Our impact proposal for class misclassifications, for use in the evaluation of our approach on the GTSRB, is shown in Table 3.17. We remind the reader that the impact considers how an event, in this case, a speed sign misclassification, could influence system behaviour in terms of cost, schedule, safety or performance. The impact is presented in the same way as previously shown with

[2]Details about this API can be found at `https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator`

Table 3.17: *Impact* for the misclassifications on the subset of the GTSRB dataset provided on a 0-4 scale of VL-VH. The cells with a dash are misclassifications considered irrelevant for the context or that do not make sense.

		Predicted Class							
		0	1	2	3	4	5	6	7
Actual Class	0	-	3	4	4	4	4	4	4
	1	1	-	2	3	4	4	4	4
	2	3	0	-	2	3	4	4	4
	3	4	2	0	-	2	3	4	4
	4	4	2	1	0	-	3	4	4
	5	4	4	3	2	1	-	3	4
	6	4	4	4	4	2	1	-	4
	7	4	4	4	4	4	4	4	-

Table 3.18: Likelihood of encounter (LoE) for the 8 classes of the subset of the GTSRB dataset used to perform the evaluation of the proposed approach, it is given on a 0-4 scale of VL-VH.

Class	0	1	2	3	4	5	6	7
LoE	2	4	3	0	3	1	4	4

a 0-4 scale of VL-VH and dashes indicating misclassifications that are not of interest for our context.

We assigned an impact value with a consideration of potential accidents arising from class misclassifications, for example, we propose that misclassifying a speed limit of 30 km/h (0) as any of 60 km/h (2), 70 km/h (3), 80 km/h (4), 100 km/h (5), and 120 km/h (6) could lead to a very high impact accident since the speed limit is not being respected, and we therefore assign a VH (4) impact to this misclassifications (see Table 3.17).

Likelihood of encounter $LoE(i)$ The likelihood of encounter for this evaluation is shown in Table 3.18 and reflects how likely is to encounter an instance of each class in an hour long road trip. Once more we classify these on a 0-4 scale of VL-VH.

Likelihood of misclassification thresholds As previously described we used thresholds to map the fractions of misclassification to a 5 point scale 0f VL-VH. The thresholds used for this experiment are $0 < 0.002 < 0.005 < 0.01 < 0.0152 < 0.03 < 1$. With the thresholds in place we can define the intervals for the risk levels as follows: VL =[0, 0.002); L=[0.002, 0.005); M =[0.005, 0.01); H = [0.01, 0.0152); VH=[0.0152, 0.03); P = [0.03, 1].

Table 3.19: Mean fraction of misclassification $p(i, j)$ for the evaluation of the 30 risk-oblivious models for the subset of the GTSRB.

		Predicted Class							
		0	1	2	3	4	5	6	7
Actual Class	0	0.0	0.0005	0.0	0.0006	0.0015	0.0005	0.0	0.0029
	1	0.0022	0.0	0.0009	0.0003	0.0008	0.0	0.0	0.0
	2	0.0001	0.0019	0.0	0.0	0.0262	0.0006	0.0001	0.0022
	3	0.0013	0.0004	0.0004	0.0	0.0019	0.0012	0.0025	0.0164
	4	0.0124	0.0153	0.004	0.0001	0.0	0.0025	0.0009	0.0
	5	0.0002	0.0	0.0	0.0	0.0006	0.0	0.0027	0.0
	6	0.0	0.0008	0.0053	0.0027	0.0039	0.007	0.0	0.0135
	7	0.0008	0.0003	0.0022	0.0014	0.0023	0.0006	0.0008	0.0

Maximum acceptable risk level. Similarly to the evaluation on CIFAR-10 we evaluated this data set using two acceptable risk levels: $\tau = M$ and $\tau = H$ (i.e. mitigate all risks $> M$ and $> H$ respectively).

3.3.3.2 Step 1: Risk-oblivious training

To create the models for our experiments we used a convolution neural network with the structure based on those developed by the Keras team [26]. We trained 30 models with an average F1 score of 0.9825 and average residual risk of 10.028, for 100 epochs, the time required to train one model was around 1.55 hours.

3.3.3.3 Step 2: Risk-aware verification

As we detailed in Section 3.1, the aim of this step of our approach is to obtain the set of concerns to be mitigated by the risk-aware ensemble. To achieve this we first combine the *fLoM* with the *impact* and *LoE* to obtain the fractional overall likelihood, *fOL*. Table 3.19 show the mean fraction of misclassification for the GTSRB dataset and Table 3.20 shows the associated *fLoM* on a scale of 0-4 (VL-VH) with the dashes assigned to correct classifications.

Table 3.21 shows the *fOL* for all class-pairs in the data set. The *fOL* is also provided on a 0-4 scale of VL-VH and the dashes are for the correct classifications. Lastly, Table 3.22 shows the *frisk* associated with each of the class-pairs in the data set, with the 11 concerns acquired for $\tau = 2(M)$ highlighted in red (note this also includes the concerns for $\tau = 3$ (H)). The list of concerns highlighted, and their corresponding risk values, are as follows: 1) 60 km/h (2) predicted as class Other (7) with a risk of 2.03; 2) Other (7) predicted as class 60 km/h (2) with a risk of 2.06; 3) 80 km/h (4) predicted as class 100 km/h (5) with a risk of 2.08; 4) Other (7) predicted as class 80 km/h (4) with a risk of 2.1; 5) 120 km/h (6) predicted as class 70 km/h (3) with a risk of 2.23; 6) 120 km/h (6) predicted as class 80km/h (4) with a risk of 2.315; 7) 80 km/h (4) predicted as class 50 km/h (1) with a risk of 3;

Table 3.20: Mean *fLoM* for the evaluation of the 30 risk-oblivious models on the GTSRB data set, the values are provided on a 0-4 scale of VL-VH. The cells with a dash are misclassifications considered irrelevant for the context.

		Predicted Class							
		0	1	2	3	4	5	6	7
Actual Class	0	-	0.25	0.0	0.3	0.75	0.25	0.0	1.29
	1	1.06	-	0.44	0.15	0.4	0.0	0.0	0.0
	2	0.05	0.95	-	0.0	4.74	0.3	0.05	1.06
	3	0.64	0.2	0.2	-	0.95	0.6	1.16	4.08
	4	3.46	4.0	1.66	0.05	-	1.16	0.44	0.0
	5	0.1	0.0	0.0	0.0	0.3	-	1.23	0.0
	6	0.0	0.4	2.06	1.23	1.63	2.4	-	3.67
	7	0.4	0.15	1.06	0.7	1.1	0.3	0.4	-

Table 3.21: Mean *fOL* for the evaluation of the 30 risk-oblivious models on the GTSRB data set, the values are provided on a 0-4 scale of VL-VH. The cells with a dash are misclassifications considered irrelevant for the context.

		Predicted Class							
		0	1	2	3	4	5	6	7
Actual Class	0	-	1.12	1.0	1.15	1.38	1.12	1.0	1.64
	1	2.06	-	1.44	1.15	1.4	1.0	1.0	1.0
	2	1.05	1.95	-	1.0	4.74	1.3	1.05	2.03
	3	0.32	0.1	0.1	-	0.48	0.3	0.58	1.69
	4	3.46	4.0	2.33	1.05	-	2.08	1.44	1.0
	5	0.1	0.0	0.0	0.0	0.3	-	1.12	0.0
	6	1.0	1.4	3.06	2.23	2.63	3.4	-	4.34
	7	1.4	1.15	2.06	1.7	2.1	1.3	1.4	-

8) 120 km/h (6) predicted as class 60 km/h (2) with a risk of 3.06; 9) 80 km/h (4) predicted as class 30 km/h (0) with a risk of 3.46; 10) 120 km/h (6) predicted as class Other (7) with a risk of 4.33; 11) 60 km/h (2) predicted as class 80 km/h (4) with a risk of 4.74.

3.3.3.4 Step 3: Risk-aware training

We trained 30 models per concern identified in the previous step, 10 of such models were built for each $\omega \in \{2, 5, 10\}$ (a total of 330 risk-aware models for $\tau = M$,

Table 3.22: *frisk* values for the class-pairs in the GTSRB data set, the values are provided on a 0-4 scale of VL-VH. The cells with a dash are misclassifications considered irrelevant for the context. Highlighted in red the concerns with risk value > 2 (M).

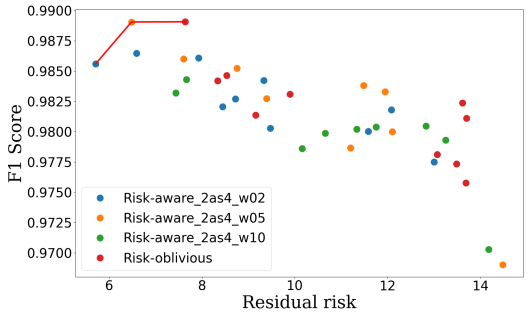
		Predicted Class							
		0	1	2	3	4	5	6	7
Actual Class	0	-	1.12	1.0	1.15	1.38	1.12	1.0	1.64
	1	1.35	-	1.44	1.15	1.4	1.0	1.0	1.0
	2	1.05	0.65	-	1.0	4.74	1.3	1.05	2.03
	3	0.32	0.1	0.03	-	0.48	0.3	0.58	1.69
	4	3.46	3.0	1.44	0.35	-	2.08	1.44	1.0
	5	0.1	0.0	0.0	0.0	0.3	-	1.12	0.0
	6	1.0	1.4	3.06	2.23	2.32	1.8	-	4.34
	7	1.4	1.15	2.06	1.7	2.1	1.3	1.4	-

and 120 risk-aware models for $\tau = H$) with an average F1 score of 0.9813 and average residual risk of 10.2770. Figure 3.15 compares the risk-oblivious and risk-aware models and we would expect to see the risk-aware models dominating the risk-oblivious ones. However, in this experiment we note that there is always one risk-oblivious model present on the Pareto front. The general tendency, however, is for the risk-aware models to dominate the risk-oblivious in terms of risk. For instance in Figure 3.15d out of 4 models in the Pareto front three belong to the risk-aware models, two weighted as 2 (blue dots) and one weighted as 10 (green dots) Where a risk-oblivious model is included it tends to be on the basis of improved F1 score. We believe that including high performing risk-oblivious models is of benefit for the performance of the ensemble as these models will compensate for the class-pairs that do not have specialist models, i.e. all misclassifications with risk values smaller than τ.

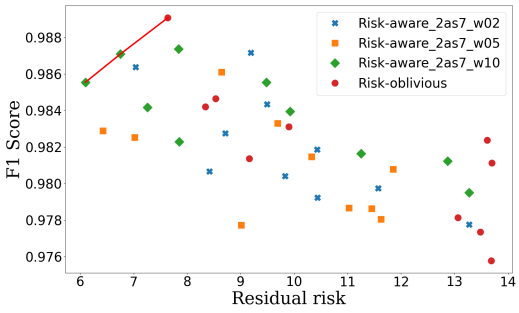
Figures 3.16, 3.17, and 3.18 show the effect of applying a weight to the loss function for the risk-aware models associated with the 60km/h predicted as 80km/h concern. We observe a similar pattern to that observed for the previous dataset. In Figure 3.16 the tendency is for the number of misclassifications to reduce as the value of the weight is increased. The marker at the intersection of the green lines represents the number of misclassifications for an actual class of 60 km/h predicted as 80 km/h. This marker is smaller on Figure 3.16b for $\omega = 2$ than for the risk-oblivious models indicating that the number of misclassifications has dropped.

There is not a great difference between the sizes of marker in figures 3.16b and 3.16c, although the reduction becomes clear as the weight increases to 10, this is shown in Figure 3.16d where we obtain the smallest number of misclassifications.

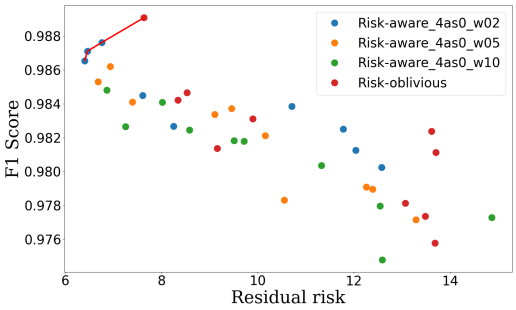
The weight also has an effect on the rest of class pairs, observe how in Figure 3.16a the markers for the class pairs 80 km/h predicted as 30 km/h, 80 km/h predicted as 50 km/h, and a misclassifcation of 70 km/h predicted as other are larger than



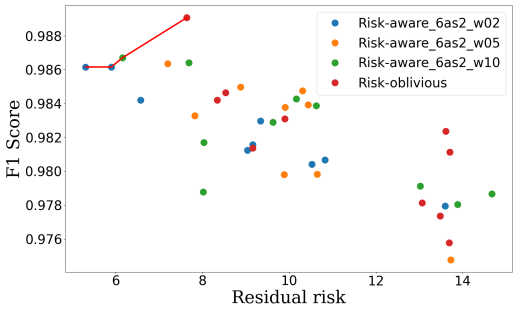
(a) Models for the concern 60km/h(2) as 80km/h(4)



(b) Models for the concern 60km/h(2) as other(7)



(c) Models for the concern 80km/h(4) as 30km/h(0)



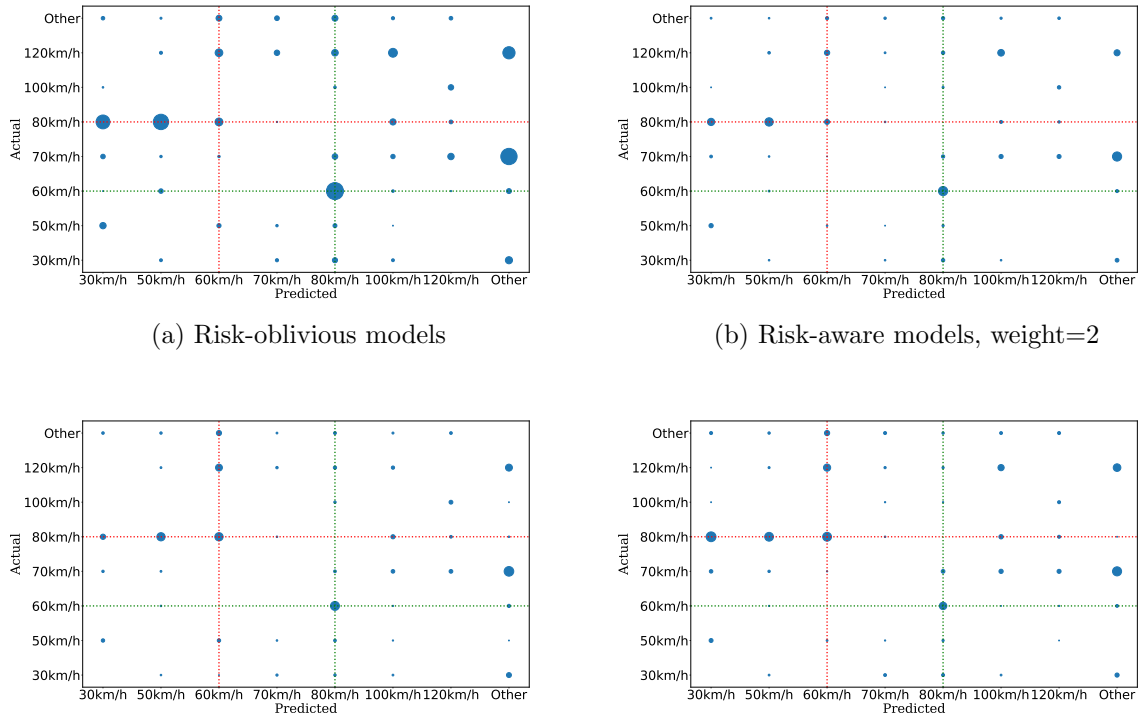
(d) Models for the concern 120km/h(6) as 60km/h(2)

Figure 3.15: Risk-oblivious vsrisk-aware models for the GTSRB dataset.

those observed in figures 3.16b, 3.16c and 3.16d. This indicates that the number of misclassifications for those concerns has also been reduced when the risk-aware models were created for the concern 60 km/h predicted as 80 km/h. This is not always the case, however, and sometimes we observe that whilst the concern is controlled other missclassification rates are increased. For example, if we compare the size of the marker for the class-pair 80 km/h misclassified as 60 km/h (the intersection of the red lines) in Figure 3.16b with that on Figure 3.16c. The marker is bigger in Figure 3.16c, which indicates how increasing the weight to fix one concern affected the other.

In Figure 3.17 we show the effect of applying weight to the concern 60 km/h predicted as 80 km/h on precision. We first plot the precision for the eight classes on the risk oblivious models in Figure 3.17a, and then we show the precision for each class at the different weight values in figures 3.17b, 3.17c, and 3.17d. The precision of the class 60 km/h drops from a mean value of approximately 0.99 in Figure 3.17a, to a mean value of around 0.97 in Figure 3.17d. This drop in precision occurs because a higher number of false positives is obtained as a result of being more willing to accept an 80 km/h classified as 60 km/h.

In Figure 3.18 we show the effect on recall of applying a weight to the concern associated with the speed limit 60 km/h being predicted as 80 km/h. In this case we see that the recall for the 80 km/h class drops from a maximum value of roughly 0.99 in Figure 3.18a to below 0.97 in Figure 3.18d. Again this occurs due to a rise in the number of false negatives and is a consequence of having a higher number



(a) Risk-oblivious models

(b) Risk-aware models, weight=2

(c) Risk-aware models, weight=5

(d) Risk-aware models, weight=10

Figure 3.16: The effect of applying weight to the concern speed limit 60km/h predicted as 80km/h on misclassification. The size of the marker is proportional to the number of misclassifications observed, the green lines intersect the concern e.g. class 60 km/h predicted as 80 km/h whereas the red lines the inverted classes of the concern e.g. 80 km/h predicted as 60 km/h to visualise the effect of weight in both situations.

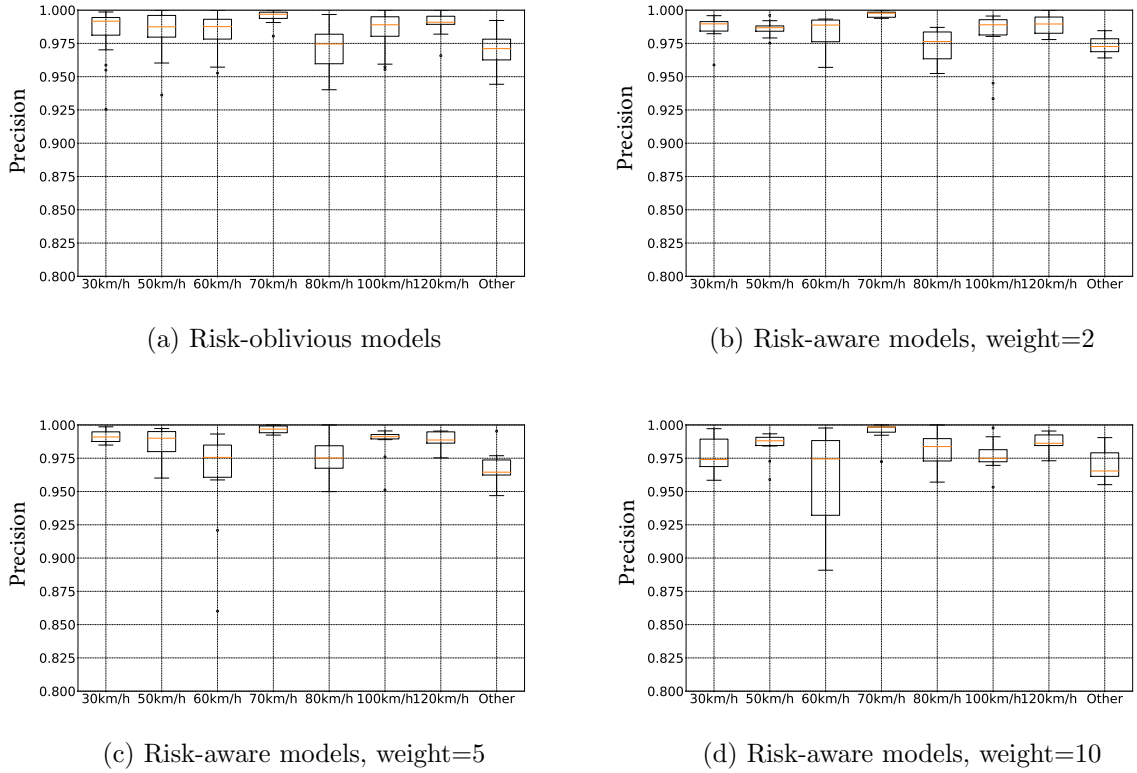
of 80km/h not correctly identified by the model. By contrast, the recall associated with the 60km/h class sees an increase from around 0.96 in Figure 3.18a to 0.98 in Figure 3.18d. This is the result of having more 60km/h correctly classified, which is the aim for this set of models.

3.3.3.5 Step 4: Risk-aware ensemble synthesis and verification

As described previously, we synthesised ensembles for two values of τ; $\tau = M$ and $\tau = H$, to show that our approach is able to mitigate risk at different thresholds of risk acceptance. The obtained results are summarised below for each τ.

Ensemble synthesis for $\tau = M$ We first randomly selected eight models for each of the 11 concerns identified for this value of τ (the set of concerns is listed in Table 3.23). To this model set we added eight randomly selected risk-oblivious models from the F1/residual-risk Pareto front. This resulted in a set of 96 models from which the GA could synthesise three ensembles with the number of models set to $l = 2$, $l = 5$ and $l = 10$.

The results for this experiment are reported in Figure 3.19. The first three figures 3.19a, 3.19b and 3.19c show the F1/residual risk Pareto-optimal ensembles



(a) Risk-oblivious models

(b) Risk-aware models, weight=2

(c) Risk-aware models, weight=5

(d) Risk-aware models, weight=10

Figure 3.17: The effect of applying weight to the concern speed limit 60km/h predicted as 80km/h on precision.

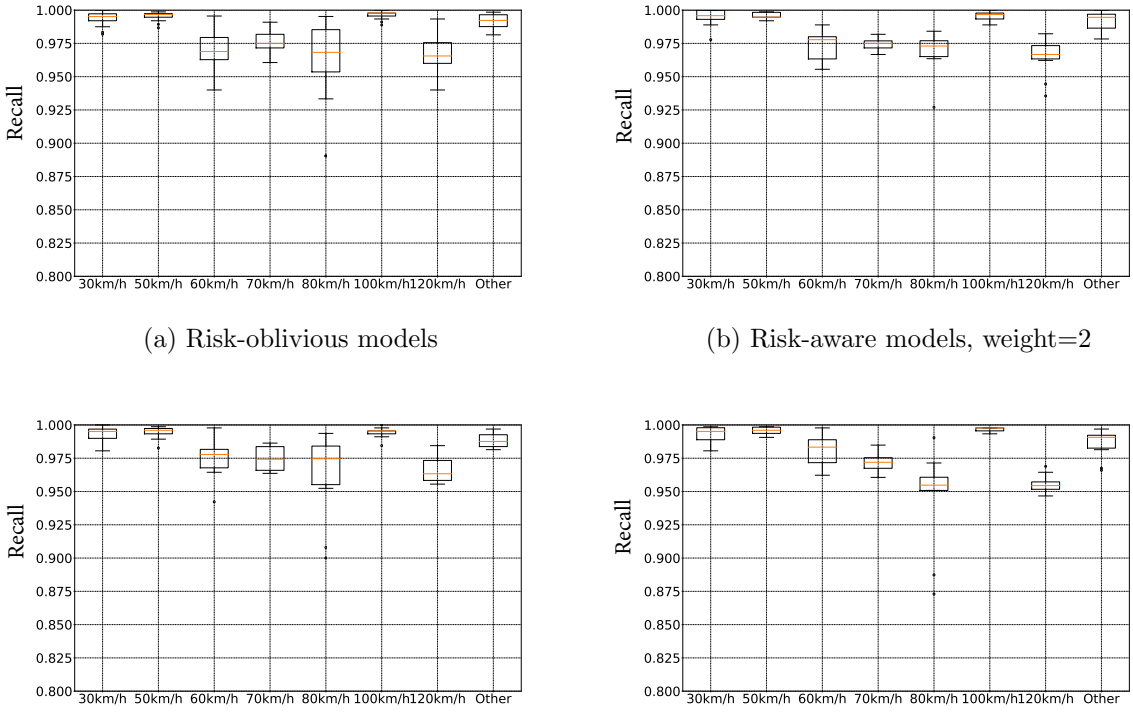
obtained for 2, 5 and 10 models. We observe the same pattern as found in the previous evaluation for the CIFAR-10 data set, i.e. as the number of generations used to create ensembles increases so does the F1 score whilst the residual risk is reduced. For example if we look at Figure 3.19b we see that the minimum residual risk found by the ensemble at 20 generations (blue Pareto front) is 1.2, and later, at 40 generations (yellow Pareto front), it drops to approximately 0.9. Finally, it is reduced to 0.8 after 2500 generations (red Pareto front). At the same time the F1 score, is increased from approximately 0.9930 after 20 generations to 0.9950 after 2500 generations.

In Figure 3.19d we compare the 3 different sized ensembles after 2500 generations of the GA. As the number of models is increased from 2 to 10, the performance of the synthesised ensembles improves with the residual risk falling from 1.5 to roughly 0.6 for the ensemble allowing 10 models.

Finally, in Figure 3.19e the performance of the risk-oblivious and the risks-aware models is compared to the Pareto fronts for the 3 synthesised ensembles. We observe that the ensembles clearly dominate the best individual models, both in terms of risk and F1 score.

Table 3.23 shows a summary of the risk values obtained for each of the ensembles during the training stage compared to the initial risk for each of the identified concerns. The values shown in red indicate where the risk values remain above the maximum acceptable risk $\tau = M$ (2).

We observe that, for all sizes of ensemble, all the concerns were mitigated to some extent, with the exception of concern 1) 60km/h predicted as the class other



(a) Risk-oblivious models

(b) Risk-aware models, weight=2

(c) Risk-aware models, weight=5

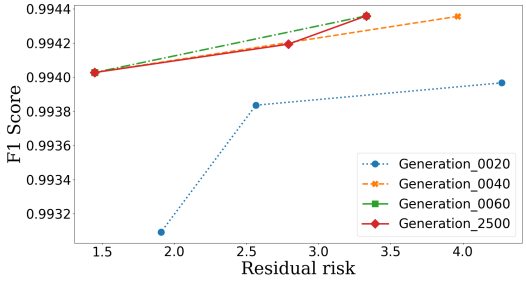
(d) Risk-aware models, weight=10

Figure 3.18: The effect of applying weight to the concern speed limit 60km/h predicted as 80km/h on recall.

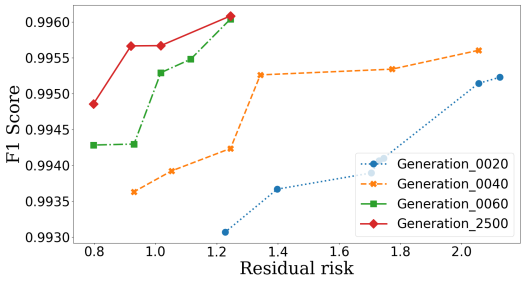
whose risk value was slightly raised from 2.03 to 2.04. This behaviour mirrors that observed in the evaluation of the approach on the CIFAR-10 data set in which some concerns seem to get worst in terms of risk. However, we do observe that the residual risk has been significantly mitigated at all sizes of the ensemble. Whilst the most significant risk reduction can be seen in the largest, 10 model, ensemble we note that all ensembles are able to reduced many concerns with even the smallest ensemble reducing the risk associated with concern 10 from very high (4) risk to moderate (2).

Table 3.24 shows the F1 score and the time for ensemble construction with both increasing as the size of the ensemble is increased.

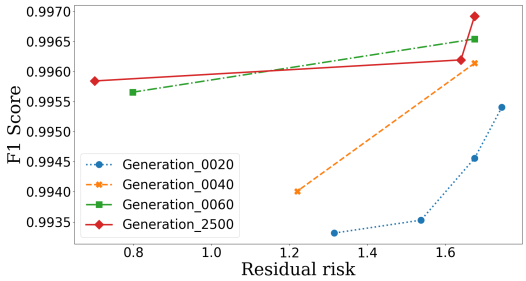
Table 3.25 presents the results obtained for the test set reserved for the ensemble. In general, we observe the same pattern as described above, with the residual risk decreasing as more models are allowed in the ensemble. From an initial residual risk of 9.405 we observe a fall to 2.71 for the ensemble of size 2, 1.47 for the ensemble of size 5 and finally 1.1 for the ensemble of size 10. We note that while the residual risk obtained in the test set is higher than that observed during the training of our approach it is still able to successfully mitigate the risk associated with misclassifications in this dataset. Lastly, Table 3.26 shows the F1 score obtained for the different sizes of the ensemble which shows an increasing trend from 0.9935 (2 models) to 0.9938 (5 models) and to 0.9947 (10 models).



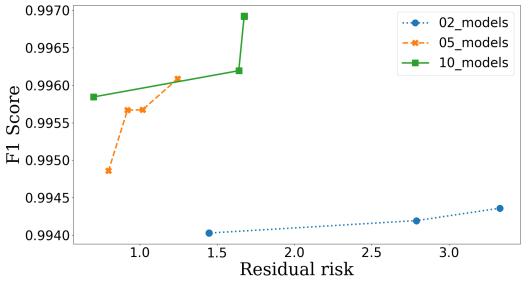
(a) Ensemble 2 models



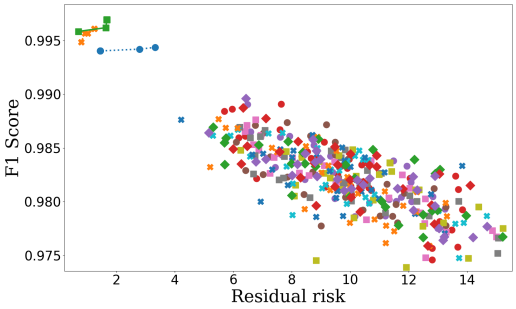
(b) Ensemble 5 models



(c) Ensemble 10 models



(d) Comparing the 3 ensemble sizes at 2500 generations



(e) Comparing single models(risk-oblivious and risk-aware) vs the 3 ensemble sizes

Figure 3.19: Ensemble learning history for the different sizes of the ensemble when $\tau = M$ on the subset of the GTSRB dataset.

Ensemble synthesis for $\tau = H$ The list of concerns obtained at this risk threshold can be seen in Table 3.27. For consistency with our previous experiments we selected eight models for each of the four identified concerns from the F1/residual-risk Pareto front. Again we added eight models from the risk-oblivious set to obtain a total of 40 models. The GA once more synthesised 3 ensembles of size 2, 5 and 10 models. The result are summarised in Figure 3.20 where we show the: Pareto front as it evolves over the GA generations; the effect of ensemble size on performance; and a comparison with single models.

Table 3.27 shows the risk per concern at different sizes of ensemble during training. We observe that all concerns were mitigated below $\tau = H$ and the residual risk achieved is 0, which is a great improvement. This however, does not mean that we are no longer making misclassifications, they are still happening but the number of

Table 3.23: Results for the GTSRB dataset at the different sizes of the ensemble when $\tau = M$ during training of the ensemble. The risk values correspond to the solution in the Pareto front with the smallest amount of risk when 10 models are allowed in the ensemble after 2500 generations. The risk values in red show the concerns that were not mitigated below the maximum acceptable risk $\tau = M$ (2).

Id	Concern		Initial risk	Risk ensemble 2 models	Risk ensemble 5 models	Risk ensemble 10 models
	Actual	Predicted				
1	60 km/h (2)	Other (7)	2.03	2.04	2.04	2.04
2	Other (7)	60 km/h (2)	2.06	1	1.77	1
3	80 km/h (4)	100 km/h (5)	2.08	1	1	1
4	Other (7)	80 km/h (4)	2.1	1.77	1	1
5	120 km/h (6)	70 km/h (3)	2.23	1	1	1
6	120 km/h (6)	80km/h (4)	2.315	2.04	1	1
7	80 km/h (4)	50 km/h (1)	3	2.1	2.1	1.79
8	120 km/h (6)	60 km/h (2)	3.06	1	1	1
9	80 km/h (4)	30 km/h (0)	3.46	1.79	1.79	1
10	120 km/h (6)	Other (7)	4.33	2.07	1	1
11	60 km/h (2)	80 km/h (4)	4.74	3.21	2.66	2.66
Residual risk			9.405	1.4475	0.7975	0.7

Table 3.24: F1 score achieved and time required to train the ensemble on the GTSRB data set using the GA after 2500 generations when $\tau = M$ at the smallest residual risk.

Ensemble 2 models		Ensemble 5 models		Ensemble 10 models	
F1 score	Required Time (in hours)	F1 score	Required Time (in hours)	F1 score	Required Time (in hours)
0.994028	0.9	0.994859	1.58	0.995843	4.13

misclassifications are tolerated, i.e. below $\tau = H$. Table 3.28 shows the F1 score and the time required to synthesise the ensemble for the evaluation on the subset of the GTSRB and again we observe the same pattern as for previous evaluations, i.e. the F1 score and the time increases as more models are allowed in the ensemble.

Tables 3.29 and 3.30 show the risk values and F1 score corresponding to the evaluation of our approach on the test data set for the ensemble. We note that two risk values, coloured in red, for the concern 4) 60 km/h predicted as 80 km/h, remain above the maximum acceptable risk for the smaller ensembles; nevertheless, they were reduced from an initial risk value of 4.74 to 3.21. We also note that the overall residual risk was reduced considerable from 3.59 to 0.21 for both the smaller ensembles and completely mitigated by the larger, 10 model, ensemble.

Finally, in Figure 3.21 we show an example of predictions made by the ensemble of size 10 for $\tau = M$, the 32 images were randomly taken from the test data set. The figure shows the image, predicted class and ground truth label in brackets, the

Table 3.25: Results for the GTSRB dataset at the different sizes of the ensemble when $\tau = M$ in the test dataset. The risk values in red show the concerns that were not mitigated below the maximum acceptable risk $\tau = M$ (2).

Id	Concern		Initial risk	Risk ensemble 2 models	Risk ensemble 5 models	Risk ensemble 10 models
	Actual	Predicted				
1	60 km/h (2)	Other (7)	2.03	2.035	2.035	2.035
2	Other (7)	60 km/h (2)	2.06	1	1.77	1
3	80 km/h (4)	100 km/h (5)	2.08	1	1	1
4	Other (7)	80 km/h (4)	2.1	1.77	1	1
5	120 km/h (6)	70 km/h (3)	2.23	1	1	1
6	120 km/h (6)	80km/h (4)	2.315	2.035	1	1
7	80 km/h (4)	50 km/h (1)	3	2.0975	2.23	1.79
8	120 km/h (6)	60 km/h (2)	3.06	1	1	1
9	80 km/h (4)	30 km/h (0)	3.46	1.79	1.79	1.79
10	120 km/h (6)	Other (7)	4.33	3.33	1	2.18
11	60 km/h (2)	80 km/h (4)	4.74	3.21	3.21	2.885
Residual risk			9.405	2.7075	1.475	1.1

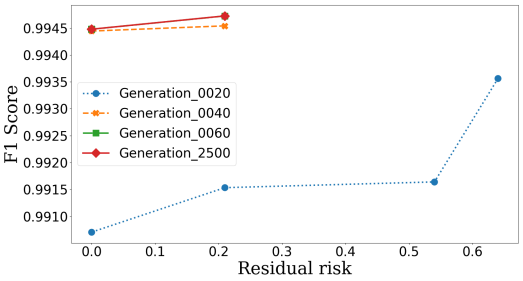
Table 3.26: F1 score at different sizes of the ensemble obtained when the ensemble was evaluated in the GTSRB test dataset using $\tau = M$.

F1 ensemble 2 models	F1 ensemble 5 models	F1 10 ensemble models
0.9935	0.9938	0.9947

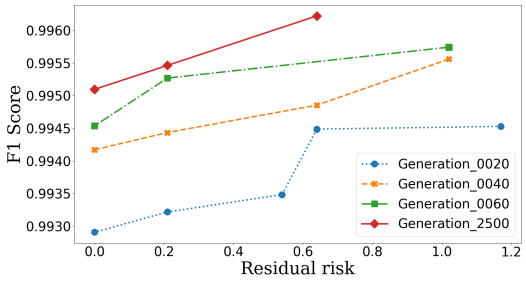
correct predictions are coloured in green and the incorrect ones in red.

Table 3.27: Results for the GTSRB dataset at the different sizes of the ensemble when $\tau = H$ during training of the ensemble. The risk values correspond to the solution in the Pareto front with the smallest amount of risk when 10 models are allowed in the ensemble after 2500 generations.

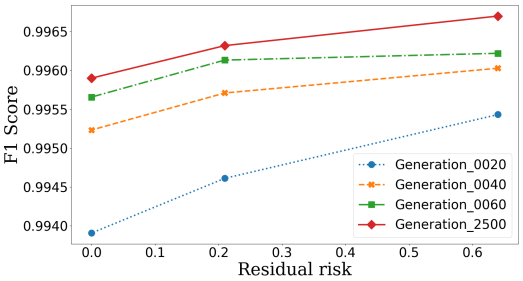
Id	Concern		Initial risk	Risk ensemble 2 models	Risk ensemble 5 models	Risk ensemble 10 models
	Actual	Predicted				
1	120 km/h (6)	60 km/h (2)	3.06	1	1	1
2	80 km/h (4)	30 km/h (0)	3.46	1	2.19	1
3	120 km/h (6)	Other (7)	4.33	2.07	2.07	1
4	60 km/h (2)	80 km/h (4)	4.74	2.88	2.88	2.88
Residual risk			3.59	0	0	0



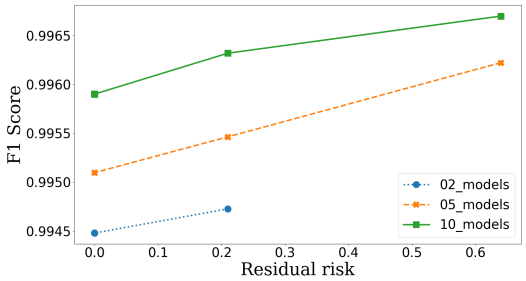
(a) Ensemble 2 models



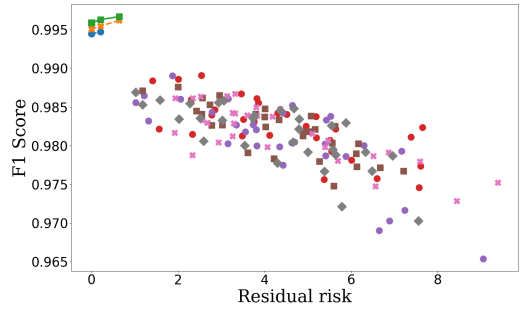
(b) Ensemble 5 models



(c) Ensemble 10 models



(d) Comparing the 3 ensemble sizes at 2500 generations



(e) Comparing single models(risk-oblivious and risk-aware) vs the 3 ensemble sizes

Figure 3.20: Ensemble learning history for different sizes of the ensemble when $\tau = H$ on the subset traffic sign dataset.

Table 3.28: F1 score achieved and time required to train the ensemble on the GTSRB data set using the GA for 2500 generations when $\tau = H$ at the smallest residual risk.

Ensemble 2 models		**Ensemble 5 models**		**Ensemble 10 models**	
F1 score	**Required Time (in hours)**	**F1 score**	**Required Time (in hours)**	**F1 score**	**Required Time (in hours)**
0.994479	0.85	0.995096	1.5	0.9959	3.05

3.3.4 Discussion

We presented the evaluation of our approach using two different data sets, CIFAR-10 and a subset of the GTSRB. Both data sets were evaluated using the same

Table 3.29: Results for the GTSRB dataset at the different sizes of the ensemble when $\tau = H$ in the test dataset. The risk values correspond to the solution in the Pareto front with the smallest amount of risk when 10 models are allowed in the ensemble after 2500 generations. The risk values in red show the concerns that were not mitigated below the maximum acceptable risk $\tau = H$ (3).

Id	Concern		Initial risk	Risk ensemble 2 models	Risk ensemble 5 models	Risk ensemble 10 models
	Actual	Predicted				
1	120 km/h (6)	60 km/h (2)	3.06	2.07	1	1
2	80 km/h (4)	30 km/h (0)	3.46	1	2.195	1
3	120 km/h (6)	Other (7)	4.33	2.81	2.81	2.07
4	60 km/h (2)	80 km/h (4)	4.74	3.21	3.21	2.885
Residual risk			3.59	0.21	0.21	0

Table 3.30: F1 score at different sizes of the ensemble obtained when the ensemble was evaluated in the GTSRB test dataset using $\tau = H$.

F1 ensemble 2 models	F1 ensemble 5 models	F1 ensemble 10 models
0.9939	0.9942	0.995

hyperparameters, i.e. number of models allowed in the ensemble, maximum risk acceptable thresholds, number of generations in the GA, etc. The results obtained for both data sets indicate that our method can effectively identify and mitigate the risk associated with relevant misclassifications for DNN image classifiers. We

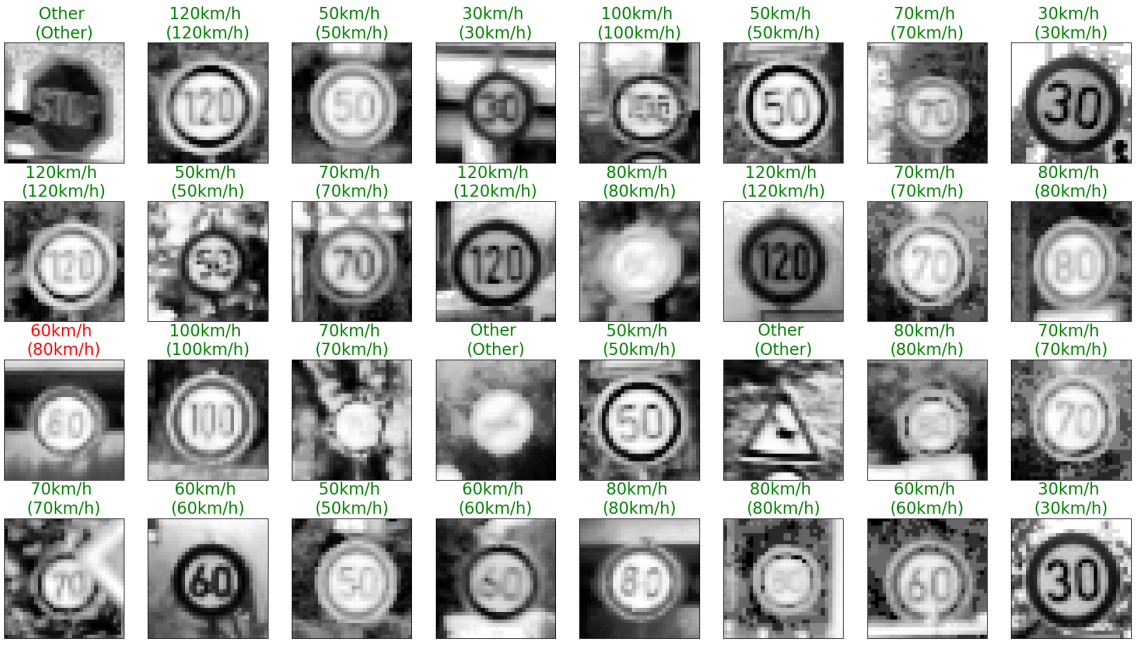


Figure 3.21: Predictions of the ensemble of 10 models with $\tau = M$ in the GTSRB test dataset for the ensemble. In green the correct predictions and in red the incorrect ones, the actual class is in brackets.

observed how, in both experiments, during training and testing of the ensembles the tendency is for the residual risk to decrease and the F1 score to increase as more models were allowed in the ensemble. We also notice how the time to synthesise the ensembles is higher as more models are allowed in the GA; additionally, the time to synthesise an ensemble grows when the GA has more models to choose from, for instance, in the CIFAR-10 experiments 176 models were available to the GA and it took 14.5 hours to complete 2500 generations whereas in the subset of the GTSRB 96 models were available and it took 4.13 hours to complete the 2500 generations.

We could further observe how in both data sets, some concerns appeared to have worsen their risk values (i.e. increased the initial risk value) after the synthesised ensembles; however, is the residual risk what was minimised and not the number of concerns, even when the risk for an individual concern rose, the residual risk was always minimised, we found an analogy with a mattress, whenever some of the springs are pressed down, others will come up. We understand that our method can certainly deal and mitigate the risk for the most of the concerns as our results indicate; nevertheless, we acknowledge that our proposed method has little space for maneuver when the data sets are highly imbalanced or classes are underrepresented. Our method can effectively decrease the amount of risk by lowering the likelihood of misclassification; however, that does not prevent the use of additional mitigation strategies (when our approach is not sufficient) such as lowering the likelihood of encounter of a given class by putting other measures in place, e.g. fences can be built in the surroundings of a road to prevent deer from crossing the road. In conclusion, the results from these experiments show that better image classifiers can be constructed by synthesising Pareto-optimal ensembles that include risk-oblivious and risk-aware models.

3.3.5 Threats to validity

Construct validity. Potential threats to construct validity may be due to the procedure employed during the design of the case study for evaluating the proposed approach, and due to the assumptions or simplifications made, such as determining values for the *LoE* and the *Impact* of misclassification of class pairs. To mitigate this concern, we based our approximations of these values based on real-world consequences of misclassifications and LoE. Specifically, we relied on the Road Accidents and Safety Statistics reports [3] published on the UK Government's web site as a reliable source to inform our approximations. This approach allowed us to increase the credibility of our findings. To further reduced these threats, we recommend the use of input from domain experts in order to fine-tune the values accordingly to reflect the context of interest.

Internal validity. Internal validity threats may first arise from biases introduced during the acquisition of results, specifically related to the data used for training the DNN models and performing ensemble synthesis. Additional threats may arise due the possibility of sub-optimal solutions as a result of the GA getting trapped in local minima.

[3]The information is available at `https://www.gov.uk/government/collections/road-accidents-and-safety-statistics`

To address these concerns, we implemented specific measures. First, we carefully segmented the data into separate sets for validation, training and testing, for both the individual models and the ensemble synthesis. This ensured that the evaluation process was conducted on independent datasets, reducing the risk of bias.

Moreover, we conducted experiments using two distinct datasets, which allowed us to create diverse scenarios in terms of length and complexity. This approach facilitated the exploration of different ensemble configurations, including varying the number of models involved, and enabled us to train the GA for different numbers of generations. By incorporating these variations, we aimed to minimise the potential biases and limitations associated with a single dataset or fixed experimental parameters. Overall, these measures were taken to address the aforementioned threats, and thus to increase the reliability and robustness of our findings.

External validity. These threats can originate from concerns related to the generalisability of our method, i.e. whether it will yield similar results in terms of risk minimisation and improving performance metrics when applied to other datasets. To alleviate this threat, we have developed our approach in a context-independent way, allowing its usage regardless of the DNN model structure, dataset, risk thresholds, *LoE* values, *Impact* values, and performance metrics used to guide the ensemble synthesis stage.

These considerations allow the necessary flexibility for its implementation in different contexts, as demonstrated in the evaluation section where we applied the approach to two different datasets. However, to claim complete general applicability of the approach, further experiments on emerging datasets are needed.

3.4 Related Work

Various research efforts focus on the importance of weighting for DNNs [12, 17, 90] and the problem of unbalanced data in the training process [76, 139, 168, 173]. These studies mention risk minimisation but refer only to the loss function; a proper risk profile is never used in the training and verification stages.

In [12], the authors highlight a problem that emerges when machine learning models are employed *in the wild*, i.e. how the distribution of the data of interest can be fairly shifted compared to the distribution of the data on which the model was trained. According to this source, there are cases where the publicly available datasets with which the models are trained do not represent and reflect the statistics of a particular dataset of interest. To deal with this problem they present a principled and a practical domain-adaptation algorithm to correct for shifts in the label distribution between a source and a target domain named regularised learning under label shifts (RLLS). The algorithm estimates importance weights using labeled source data and unlabeled target data, and then trains a classifier on the weighted source sample. The method was evaluated on the CIFAR-10 and MNIST datasets and the results show that RLLS improves classification accuracy. This method is comparable to ours, because the importance weighting works by altering the relative contribution of mistakes on various training points of the loss function, our method applies a similar idea to create the sets of risk-aware models for a given concern with a high likelihood of misclassification; nonetheless, their approach only concentrates on classification accuracy and disregards risk factors.

Similarly [17] investigates the effects of importance weighting in deep learning across a variety of architectures, tasks, and data sets. Their experiments focus on classification problems in which they apply class-conditioned weights of various strengths and evaluate the impact of the weights on the learned decision boundaries. To investigate the effects of importance weighting they down-weight the loss contributions of examples from a particular class. The objective of this research is to analyse how the weighting in the loss function affects the contributions of specific targeted classes. We also show the effect of weight in the number of misclassifications, and how different weight values affect the number of misclassifications, again the value of our approach resides in the risk information we include and how we combine the different models to mitigate risk.

The use of cost matrices to assign a cost to each misclassification based on the distribution of each class is proposed in [139]. These costs were then utilised during the training process to increase the final classification accuracy. While this is similar to our approach, in the sense that they target particular classes, this solutions focuses on modifying the output of a classifier. In contrast, we target the loss function. To deal with data with imbalanced class distribution [173], proposes a weighted method which is able to generalise to balanced data by assigning different weights for each example according to users' needs. To alleviate the bias in performance caused by imbalanced class distribution, an extra weight is assigned to each sample to strengthen the impact of minority class while weaken the relative impact of majority class; different to our approach, we add a weight not to each sample, but rather to each relevant misclassification.

Distinct to the previously mentioned approaches [168], proposes an evolutionary cost-sensitive deep belief network (ECS-DBN) for imbalanced classification. Their method uses adaptive differential evolution to optimise the misclassification costs based on the training data that presents an effective approach to incorporating the evaluation measure into the objective function by first optimising the misclassification costs, and then apply them to DBN. Another, similar approach is presented by [109], which formulates a misclassification cost minimising genetic algorithm; they propose three different fitness functions that aim to minimise the total misclassification cost. Although, these methods propose an evolutionary algorithm to deal with the imbalanced classification and total misclassification cost, they are not aiming to decrease the misclassification for relevant class pairs as they solely attempt to improve the overall accuracy of the models disregarding the risk factors.

Studies such as [50] use ensemble models to handle concept drift and oversampling, and for dealing with class imbalance; they briefly mention the importance of class misclassification reduction. However, their main focus is not on tackling class misclassification, and their ensembles use simple averaging to combine probability outputs from a set of models.

A model of lifelong learning is introduced in [5], based on a network of experts. New tasks/experts are learned and added to the model sequentially, building on what was learned before. Their main line of reasoning is that different models are normally created for different tasks trained on different datasets, each of which is an expert on its own domain, but not on others. They explain how learning a new task, requires adapting the model to the new set of classes and fine-tuning it with the new data. The authors of this work emphasise the need for having different specialist or expert models for different tasks and build a network of experts, where a new

expert model is added whenever a new task arrives and knowledge is transferred from previous models. This work is relevant to us because it considers individual models as experts, analogous to our risk-aware models and propose the creation of an ensemble to improve the performance of single classifiers. By contrast, however, they do not consider generalist models (or risk-oblivious models) and their intention is not to reduce the number of misclassifications but rather to focus on how to select the proper specialist for the right environment. The example they provide is an autonomous vehicle that uses video prediction and needs to be able to load the correct model for the current environment. It might not have all the data from the beginning, and so it becomes important to learn specialists for each type of environment, without the need for storing all the training data. Yet another important difference to our approach is that, despite the fact that they show interesting confusion cases or examples of misclassifications such as flowers as birds, cars as aircrafts, aircrafts as cars, etc. their approach do not aim to mitigate them per se, likewise they do not combine the knowledge from different models, but instead they provide the mechanism to retrieve the most appropriate individual model.

A similar approach to that previously presented is [63] which introduces a method for distilling the knowledge in an ensemble of models into a single model with the argument that making predictions using a whole ensemble of models is cumbersome and may be too computationally expensive to allow deployment to a large number of users, especially if the individual models are large neural networks. They also explain how their distilling of knowledge from a large model into a small one, allows to train the small model to generalise in the same way as the large model. If a cumbersome model generalises well because, for example, it is the average of a large ensemble of different models, a small model trained to generalise in the same way will typically do much better on test data than a small model that is trained in the normal way on the same training set as was used to train the ensemble. Interestingly, they also introduce a new type of ensemble composed of one or more full models and many specialist models which learn to distinguish fine-grained classes that the full models confuse. The advantage they state is that unlike a mixture of experts, these specialist models can be trained rapidly and in parallel. This is similar to our approach because we also propose a combination of generalists and specialists in our ensemble with the intention of having a generalist model which compensates for the class-pairs that do not have a specialist model, otherwise, a class-pair which was initially not regarded as a concern could end up with a risk value above the maximum tolerated. We can also see that this work diverge from our approach when they do not focus their efforts on risk mitigation of relevant class-pair misclassifications as we do.

Authors in [3] construct classifier ensembles using multiple Pareto features as inputs extracted by a multi-objective evolutionary trace transform algorithm. By contrast, for traditional classifier ensembles using single input features, data randomisation techniques have been employed in training the ensemble members to explicitly promote the diversity of ensembles, in this work, majority voting is adopted to combine the output of the base classifiers. The method combines different models to obtain better classifiers; however, they do not consider specialist and generalist models, moreover, they do not consider weights when combining the output of the classifiers which helps to give weight to the models contribution based on their ability to correctly classify a given sample. Finally, [76] aims to deal with class imbalance by

presenting an ensemble of cost-sensitive decision trees for imbalanced classification. Base classifiers are constructed according to a given cost matrix, but are trained on random feature sub-spaces to ensure sufficient diversity of the ensemble members. They also employ an evolutionary algorithm for simultaneous classifier selection and assignment of weights for the fusion process. Nonetheless, this ensemble synthesis approach does not considers risk; its goal is simply to tackle imbalance classification.

As we can see, none of the presented related works consider misclassification of relevant class pairs, nor do they consider risk during the training or verification stages. Our approach is unique in using the risk-aware training of DNNs using loss functions tailored to reduce high-risk misclassifications as well as risk-aware ensemble synthesis employing multi-objective genetic algorithms to generate DNN ensemble classifiers that are Pareto optimal with respect to both, traditional ML performance metrics and risk mitigation capabilities. To the best of our knowledge, no existing approaches mitigate the risk associated to different DNN misclassifications as we do.

3.5 Summary

In this chapter we presented an approach to synthesise risk-aware DNN classifiers and we discussed in detail each of the stages of our method in Section 3.1. We presented the evaluation of the proposed method in two different data sets, namely CIFAR-10 in Section 3.3.2 and a subset of the GTSRB in Section 3.3.3, for which we created risk-oblivious and risk-aware models. We identified the relevant concerns for each data set and synthesised Pareto-optimal ensembles that were tested and compared at different thresholds of risk acceptance to show that our proposed approach can successfully identify and mitigate the risk associated with relevant misclassifications for DNN image classifiers. Next, we presented a discussion of the findings in Section 3.3.4. Finally, we presented the related work in Section 3.4. In the next chapter (Chapter 4 Risk-aware Real-time Object Detection) we present a method for the development of risk-aware ML ensembles for real-time object detection.

Chapter 4

Risk-aware Real-time Object Detection

This chapter presents the instantiation of our generic risk-aware ML ensemble synthesis method from Section 3.1 for the development of *risk-aware ensembles* for Real-time Object Detection (RTOD). The instantiated method supports the dependable use of RTOD by (i) identifying the risks that require treatment, (ii) training a set of ML models that mitigate these risks, and (iii) constructing risk-aware ML ensembles using multi-objective genetic algorithms. The chapter begins with a description of the stages of the proposed method in Section 4.1 before an evaluation, using the PASCAL VOC 2007 dataset, is presented in Section 4.2. A discussion of related work is given in Section 4.3 before we provide a summary of the chapter in Section 4.4.

4.1 Approach

The implementation of the method for risk-aware RTOD ensembles is illustrated in Figure 4.1. In **stage 1: Risk-oblivious training**, we use standard ML training to generate a set of RTOD models. Next, in **stage 2: Risk-aware verification**, a five-point semi-quantitative risk assessment technique from the ISO 31010 standard [65] is used to identify *risk concerns*, i.e., RTOD object misclassifications that induce

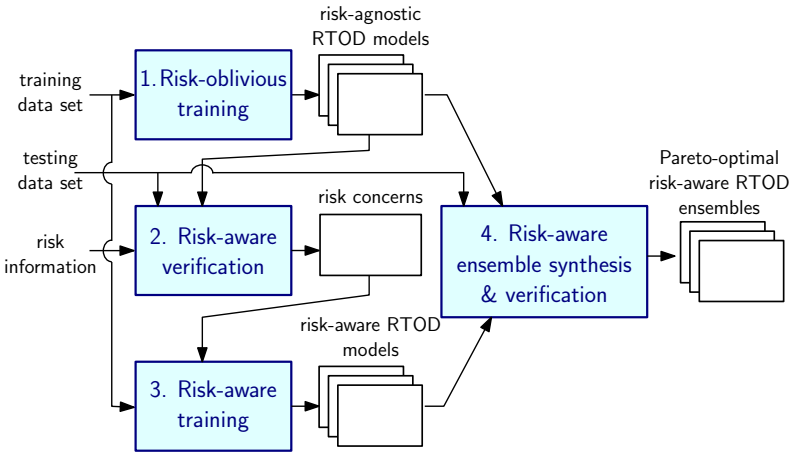


Figure 4.1: Risk-aware RTOD ensemble synthesis method.

unacceptably high risks. Next, in **stage 3: Risk-aware training**, a configurable number of *risk-aware RTOD models* are produced for each risk concern identified in the previous step. These are models whose neural networks are trained using a loss function $\mathcal{L}(\theta)$ that prioritises the minimisation of misclassifying class i as class j over that of all other misclassifications, as introduced in Chapter 3. Finally, in **stage 4: Risk-aware ensemble synthesis & verification**, we select a subset of risk-oblivious models from stage 1 and risk-aware models from stage 3, and use a multi-objective genetic algorithm to optimise a set of weights for combining the outputs from these models in an ensemble that achieves Pareto-optimal trade-offs between maximising the *mAP score* and minimising the *residual risk*.

Our presented method builds upon the approach introduced in Chapter 3, however, object detection is a much more complex task than image classification involving the classification of multiple objects, and bounding boxes to describe their locations, within a single image. The application of the method to object detection required significant extensions for the solution described in Chapter 3 as follows:

1. In the second stage of the approach (*Risk-aware verification*) the derivation of the likelihood of misclassification is qualitatively different because the object detector, unlike the classifier, is concerned with both the class label *and* the object position. To obtain the risk-concerns for the RTOD we apply the process described in Section 4.1.2;

2. In the fourth stage (*Risk-aware ensemble synthesis & verification*) we considerably extended the approach to support the effective combination of knowledge from each of the base learners, since each RTOD model is predicting a different set of multiple objects *with different locations*. This extension required the development of three algorithms for combining the bounding box predictions of the models in the ensemble;

3. The performance evaluation of the RTOD models is not the F1 score as for image classification. For object detection the commonly used performance metric is the mAP, a more complex metric as it involves an assessment of the algorithm's capacity to perform detection *and* location of *multiple objects* accurately, and of the precision-recall curve for each of the classes.

A detailed description of each of the stages of this approach can be found in the following sections.

4.1.1 Stage 1 : Risk-oblivious training

The first stage of the approach uses standard ML training to generate a set of RTOD models. Each model is obtained from the same training data set with different random initial weights for the RTOD neural network. As previously mentioned in Section 3.2.1 the importance of this step resides in the need to identify a baseline that allows the identification of class pairs whose misclassification corresponds to unacceptable levels of risks. We also remind the reader of the importance of having a large enough data set to ensure that the DNNs once trained truly reflect that the proposed model structure has issues misclassifying objects of class A as class B and are not noise or outlier values due to limited data which could be addressed with reasonable efforts to increase the number of samples available.

4.1.2 Stage 2: Risk-aware verification

In this stage a five-point semi-quantitative risk assessment technique from the ISO 31010 standard [65] is used to identify *risk concerns* for the RTOD. The likelihood of encounter loE, the impact level $impact(i, j)$ and likelihood of misclassification (LoM) thresholds are obtained as shown previously in Section 3.2.2.

The derivation of the likelihood of misclassification is different because the object detector, unlike the classifier, is concerned with both the class label and object position. This added complexity means that multiple bounding boxes, associated with multiple objects, may be incorrectly placed or missing altogether. Indeed a false detection will produce a bounding box for an object which does not exist in the scene. In this way matching predictions to the ground truth becomes more challenging.

To obtain the risk concerns for the RTOD we apply the following process:

1. For each object o_j, $j \in \{1, 2, \ldots, m\}$, in a data sample, let $c_j \in \{1, 2, \ldots, n\}$ be the true class of o_j, and $box_j = (((x_j, y_j)^{top}, (x_j, y_j)^{bottom}))$ the set that contains the coordinates of the top left and bottom right corners of each true bounding box. A pair (c_j, box_j) is an element in the ground truth set of information per object; therefore $y = \{(c_1, box_1), (c_2, box_2), \cdots, (c_m, box_m)\}$ is the set that contains the ground truth information of one sample with m objects in it.

   The data sample is then passed to a model M_i, $i \in \{1, 2, \ldots, s\}$, where s is the number of models to be assessed, whose prediction for object o_j is $c'_{i,j} = (P^1_{i,j}, P^2_{i,j}, ..., P^n_{i,j})$ that contains the probabilities of object o_j belonging to each of the n classes in the data set. And also predicts $box'_{i,j} = (((x_{i,j}, y_{i,j})^{top}, (x_{i,j}, y_{i,j})^{bottom}))$, similarly to the previous case, a tuple $(c'_{i,j}, box'_{i,j})$ will be added to the predictions set:

$$y'_i = \{(c'_{i,1}, box'_{i,1}), (c'_{i,2}, box'_{i,2}), ..., (c'_{i,j}, box'_{i,j})\} \tag{4.1}$$

   There can be three possible cases for the prediction:

   (a) All the objects in the sample where predicted by the model M_i.

   In this case we must match predicted objects to the list of objects known in the sample using the intersection over union. Note that each object may generate multiple bounding boxes:

$$\begin{aligned} Matches = \{(box_1, box_2) \in boxes(Y) \times boxes(Y'_i) \mid \\ IoU(box_1, box_2) \geq threshold \end{aligned} \tag{4.2}$$

   Next we remove any duplicates (ground-truth boxes that match with more than one predicted bounding box), in this case of duplicates the greater IoU will be selected — this generates a $Matches' \subseteq Matches$ set of box pairs.

   $Matches'$ will contain the ground truth and the prediction for the object, so there is (c_j, box_j) and $(c'_{i,j}, box'_{i,j})$ for each object (needed to add to y and to y'_i respectively).

(b) All the objects in the scene were missed. In this case, for each object in the ground-truth add $box'_{i,j} = ((0,0),(0,0))$ and $c'_{i,j} = (0_0, 0_1, .., 1_n)$ i.e. (set 0 for the probability of all classes, except for class n that is identified as *no detected*).

(c) The prediction of one or more objects in the sample is $\emptyset$ (the object was not predicted by the model M_i), but, not all objects were missed, for example in a scene with five objects two are detected and three are missed. For the detected objects do as in case (a) above, i.e match the predictions with their ground truths, and for all the ground truths without a match do as in case (b).

The ground-truth set y for each sample will be added to Y to obtain $Y = \{y_1, y_2, ...y_r\}$ that will contain the ground truth information of each of the r samples in the data set; likewise a set $Y'_i = \{y'_{i,1}, y'_{i,2}, ..., y'_{i,r}\}$ will contain the predictions for each sample in the data set and for each model M_i.

2. Once the set of ground truths Y and the set of predictions Y'_i are available we can obtain the confusion matrix.

   It is important to mention that, as in the established practice from RTOD (and therefore orthogonal to our approach), confusion matrices for object detection typically do not consider missed objects and duplicated predictions directly. Missed objects and duplicated predictions are indirectly considered in the evaluation process through IoU thresholds and precision-recall analysis. Missed objects are typically reflected in false negatives (objects that were not detected) in the precision-recall curve. Duplicated predictions may affect precision, as they contribute to false positives if they do not meet the IoU threshold or overlap with ground truth bounding boxes. In our approach we use the confusion matrix not as the performance metric when synthesising the ensemble, but as a mechanism to identify the misclassifications made by the object detector. The performance of the RTOD model is independently measured by the mAP.

3. The risk level threshold $\tau \in \{\mathsf{VL}, \mathsf{L}, \mathsf{M}, \mathsf{H}, \mathsf{VH}\}$ is defined for the application using the RTOD model which specifies the maximum risk level that can be tolerated, e.g. if $\tau = \mathsf{VH}$ we aim to mitigate all concerns with a risk value $\geq \mathsf{VH}$.

4. Given this risk information, we consider each pair of classes $i \neq j$, and we calculate $p(i,j)$ as the mean fraction of misclassifications of class i objects as class j objects over the risk-oblivious RTOD models from Section 4.1.1. Next, we use the thresholds $LoM^{\mathsf{VL}}, LoM^{\mathsf{L}}, LoM^{\mathsf{M}}$, LoM^{H}, and LoM^{VH} to establish the likelihood $LoM(i,j) \in \{\mathsf{VL}, \mathsf{L}, \mathsf{M}, \mathsf{H}, \mathsf{VH}\}$ of class i being misclassified as class j by a risk-oblivious RTOD model. Finally, we use likelihood and risk matrices [65] to establish an overall likelihood $OL(i,j)$ and a risk level $r(i,j)$ for the misclassification of class i as class j as shown in Figure 4.2. The shading of the 'VH' elements from the risk matrix indicates that the diagram assumes a maximum acceptable risk level $\tau = \mathsf{VH}$, and that any pair of classes (i,j) whose misclassification risk level resides in the shaded area represents a *risk*

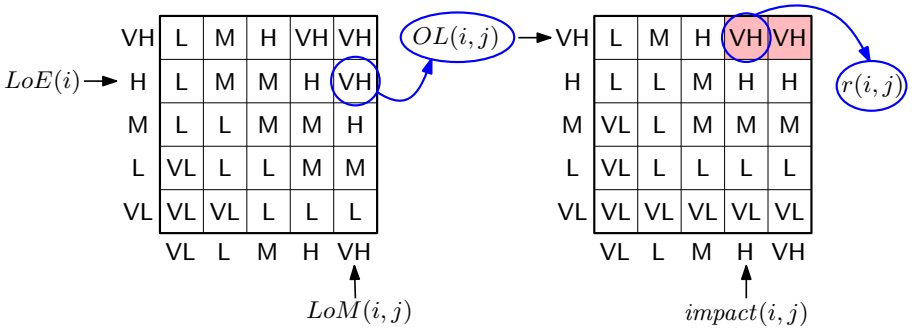


Figure 4.2: Calculation of the risk level $r(i,j)$ for misclassifying class i as class j given the likelihood of encounter $LoE(i)$ for class i and the impact $impact(i,j)$ of such a misclassification, where $LoM(i,j)$ is the calculated likelihood of misclassification.

concern, i.e., a risk that needs to be mitigated. The outcome of this stage of our method is a set of all such risk concerns for the RTOD models.

4.1.3 Stage 3 : Risk-aware training

If risk concerns were identified in the previous stage (Section 4.1.2), then this stage produces a configurable number of *risk-aware RTOD models* for each risk concern (i,j). These are models whose NNs are trained using a loss function $\mathcal{L}(\theta)$ that prioritises the minimisation of misclassifying class i as class j over that of all other misclassifications:

$$\mathcal{L}(\theta) = -\frac{1}{M}\sum_{k=1}^{M}\sum_{l=l}^{N} w(y_k,\hat{y}_k)\,y_{kl}\log p_{kl}, \tag{4.3}$$

where θ represents the NN weights to be learnt, M is the number of samples in the training data set, N is the number of classes, $y_{kl}=1$ if the true class for the k-th sample $y_k = l$ and $y_{kl}=0$ otherwise, p_{kl} is the value of the l-th NN output neuron for the k-th sample, and (unique to our method) $w(y_k,\hat{y}_k) > 0$ is a weight associated with classifying sample k as class $\text{argmax}_{l=1}^{N} p_{kl} = \hat{y}_k$; to obtain RTOD models that mitigate the risk concern (i,j), we use:

$$w(y_k,\hat{y}_k) = \begin{cases} \omega N^2/(N^2+\omega-1), & \text{if } y_k = i \wedge \hat{y}_k = j \\ N^2/(N^2+\omega-1), & \text{otherwise} \end{cases} \tag{4.4}$$

where $\omega > 1$ is a parameter of our risk-aware model training. Note that (i) setting $\omega = 1$ reduces (4.3) to the standard loss function used in training, and (ii) the use of $\omega > 1$ increases the contribution of the loss term $y_{kl}\log p_{kl}$ from (4.3) for samples k of class i misclassified as class j.

We show in Section 4.2 that using the loss function (4.3) can yield RTOD models with significantly reduced misclassification rates $p(i,j)$. Of course, this technique cannot guarantee that such models will be obtained under all circumstances. For instance, the technique may be unable to mitigate risks due to training data sets that are unbalanced or too small, or when poorly chosen NN architectures are used. In such cases, these issues may be addressed directly, or it may be possible to mitigate

the risk by reducing its impact (e.g., a self-driving car can drive slower) or likelihood of encounter (e.g., by banning bicycles on certain roads used by self-driving cars).

4.1.4 Stage 4: Risk-aware ensemble synthesis & verification

In this stage, we select a subset of risk-oblivious models from Stage 1 and risk-mitigating models from Stage 3, and use a multi-objective genetic algorithm to optimise a set of weights for combining the outputs from these models into a set of ensembles that achieve Pareto-optimal trade-offs between:

1. maximising the *mAP score*, an established performance measure for object detection models [42, 91, 107, 120];

2. minimising the residual risk (see Equation 3.4).

The ensemble construction, GA strategy, and the risk encoding mechanism is the same as that described in Section 3.2.4. Below, we provide the steps needed to form the ensemble for an object detector model. Since each model predicts multiple objects the challenge is to define how the outputs from each model should be combined. In order to address this challenge we adopt the following approach.

We first formally define an ensemble *Ens* as a set of l models such that $Ens = \{M_1, M_2, \ldots, M_l\}$ with $l \geq 2$. Each of the models may be risk-aware or risk-oblivious. The objective of the heuristic search is to obtain the set of ensembles that achieves Pareto optimality between maximising the mAP score and minimising the risk. The steps taken to form the ensemble are shown in Algorithms 1–3:

1. The function getParetoOptimalEnsembles from Algorithm 1 shows the main pseudocode for the ensemble creation. A *combination* of object predictions that belong to the same ground-truth is obtained by first identifying sets of objects with significantly overlapping bounding boxes (according to the IoU measure). In line 8 we call the function getNewElement from Algorithm 2, which adds a new element to the set *combination* if the IoU between the ground-truth box and the predicted box exceeds a predefined threshold. Once all the model's predictions are checked for an object, we call the function getClassif in line 10 to combine the predictions of all selected single models to obtain one prediction per object.

2. The combination of the individual predictions is generated by function getClassif from Algorithm 3. This function multiplies the set of predictions generated by each model by the associated weights learnt by the GA (line 6), and adds each model's weighted prediction to obtain the output of the ensemble as $O_c = (c'_1 * weights_1 + c'_2 * weights_2, ... + c'_l * weights_l)$. The weights $(w_{ji})_{1 \leq j \leq l, 1 \leq i \leq n}$, optimised by the GA, are those used to combine the lists of objects detected by $l > 1$ RTOD models into a single list *EnsembleO*. After this combination of individual model predictions, a single object is predicted by the ensemble. This object has:

   (a) bounding box coordinates computed as the mean coordinates of the bounding boxes for the objects;

(b) class given by $\text{argmax}_{i=1}^{n} \sum_{j \in J} w_{ji} c_i^j$, where $J \subseteq \{1, 2, \ldots, l\}$ is the subset of models that contribute objects to $EnsembleO$, $w_{ji} > 0$ is a weight that reflects the ability of j-th model to detect objects of class i, and c_i^j is model j's estimate probability that the object detected is of class i.

The inputs for these algorithms are:

1. The sets of predictions from l models ($setOfPredictions = \{Y'_1, Y'_2, ..., Y'_l\}$) such that set i has the form $Y'_i = \{(c'_{i1}, box'_{i1}), ..., (c'_{ili}, box'_{ili},)\}$, where $c'_{ij} = (p_1, p_2, \ldots, p_n)$ is a tuple of probabilities for the n classes, box'_{ij} contains the coordinates of the bounding boxes for the n classes, $j \in \{1, 2, ..., ni\}$; i.e. model i identifies $ni \geq 0$ objects in the image.

2. $Y = \{(c_1, box_1), ..., (c_{ni}, box_{ni})\}$ where (c_{ij}, box_{ij}) represents the ground-truth probabilities and boxes that correspond to an object.

3. $weights = \{W_1, ..., W_l\}$ contains one element per model in the ensemble, where $W_i = \{w_1, w_2, ..., w_n\}$, contains n values (one value per class); w_i represents a value between 0 and 1.

   The weights are used to multiply each predicted class probability of each model in the ensemble. By adjusting these weights the GA evaluates if the current selection of weights generated a better solution than those ecountered in previous generations.

Algorithm 1 Ensemble algorithm.

```
1: function getParetoOptimalEnsembles
2:     EnsembleO = {}
3:     while the GA is running do
4:         setOfPredictions = {Y'_1, Y'_2, ..., Y'_l}
5:         for each ((p_1, p_2, ..., p_n), box) ∈ Y do      ▷ for each element in ground-truth
6:             combination = {}
7:             for each Y' ∈ setOfPredictions do            ▷ for each set of predictions
8:                 combination = getNewElement(combination,box,Y')    ▷ see Alg. 2
9:             end for
10:            EnsembleO = EnsembleO ∪ {getClassif(combination)}    ▷ see Alg. 3
11:        end for
12:    end while
13:    return EnsembleO
14: end function
```

4.2 Evaluation

4.2.1 Evaluation methodology

To evaluate our method we applied it to the PASCAL VOC 2007 dataset [42], which is widely used as a benchmark for object detection. The goal of this task is to recognise objects from a number of visual object classes in realistic scenes which can contain multiple objects.

Algorithm 2 Function to check if an object was identified by a model.

function GETNEWELEMENT($combination$,box, Y')
  $bestMatch = nil$
  $bestScore = 0$
  $firstTime = 0$
  for each $((p_1, p_2, \ldots, p_n), box') \in Y'$ **do**
    $score = IoU(box, box')$
    if $firstTime == 0$ and $score \geq threshold$ **then**
      $bestScore = score$
      $bestMatch = ((p_1, p_2, \ldots, p_n), box')$
      $firstTime = 1$
    else if $firstTime == 1$ and $score \geq threshold$ and $score > bestScore$ **then**
      $bestMatch = ((p_1, p_2, \ldots, p_n), box')$
      $bestScore = score$
    end if
  end for
  if $bestMatch != nil$ **then**
    $combination = combination \cup \{((p_1, p_2, \ldots, p_n), box')\}$
    $Y' = Y' \setminus \{((p_1, p_2, \ldots, p_n), box')\}$
  end if
  return $combination$
end function

Algorithm 3 Function to combine the knowledge of all models in the ensemble and decide the final class.

function GETCLASSIF($combination$)
  $finalClass = \{\}$
  $finalBox = \{\}$
  $i = 0$
  for each $((p_1, p_2, \ldots, p_n), box) \in combination$ **do**
    $wPred = (p_1, p_2, \ldots, p_n) * weights[i]$
    $finalClass = finalClass \cup \{wPred\}$
    $finalBox = finalBox \cup \{box\}$
    $i+ = 1$
  end for
  $finalBox = finalBox/lenght(combination)$
  return $finalClass$, $finalBox$
end function

The VOC dataset contains significant variability in terms of object size, orientation, pose, illumination, position and occlusion. The distributions of images and objects by class are approximately equal across the training/validation and test sets. The VOC 2007 data set contains in total 9,963 images, 2,501 samples for training, 4,952 for testing and 2,501 for validation, containing 24,640 annotated objects [42] with $n = 20$ classes. The test data was augmented to 19,808 images using Gaussian noise. All individual models were trained and tested on the corresponding data splits. The validation split was not used during the training of the individual models and it was used to test the final synthesised ensemble. The twenty object classes are depicted in Figure 4.3 and listed as follows, with class ids included in brackets:

Figure 4.3: Example images from PASCAL VOC2007 dataset depicting all 20 classes.

- Person: person (14)
- Animal: bird(2), cat(7), cow(9), dog(11), horse(12), sheep(16)
- Vehicle: aeroplane(0), bicycle(1), boat(3), bus(5), car(6), motorbike(13), train(18)
- Indoor: bottle(4), chair(8), dining table(10), potted plant(15), sofa(17), tv/monitor(19)

For our experiments we used the real-time object detection pipeline of YoloV3 [119] implemented in TensorFlow 2.0 [1]. According to [112], this is a poof of concept implementation and therefore the performance of the obtained models is not guaranteed to be the best. We opted for this option given its ease of implementation and well documented code base.

All the experiments were performed on a server with 64GB of RAM with 8 processors Intel Core i7-9700K CPU @ 3.60GHz, running Ubuntu 18.04.5 with a 64-bit architecture and 6.9TB of hard disk.

[1]For a complete description of the implementation refer to `https://github.com/zzh8829/yolov3-tf2`

4.2.2 Evaluation on the PASCAL VOC 2007 data set

4.2.2.1 Risk information

In order to evaluate our approach we first defined a set of risk information for the VOC context as follows:

Impact level $impact(i, j)$ The *impact* of misclassification is shown in Table 4.1 where each of the (i, j) cells contains the impact for the relevant class pairs in a 0-4 point scale of VL to VH. It measures the interactions between misclassifications of classes and how they would affect the context. The dashes in the table refer to misclassifications that are not relevant for the context where the system will be deployed, for instance, misclassifying the class bottle (4) as class dining table (10) in a self-driving car operating in an urban setting is not relevant and therefore is dashed. This does not mean that the individual models are not confusing these classes, and in a different context this might well be a misclassification that needs to be dealt with. A value of 0 means that such misclassification has a very low (VL) impact when it occurs, e.g. misclassifying class dog (11) as class sheep (16) is deemed to have VL impact. We allocated a range of impact scores to illustrate our approach however, in practice, such values would be obtained from domain experts.

Likelihood of encounter $LoE(i)$ Is a measure of how likely is to encounter instances of a given class during a determined period of time. Table 4.2 shows the LoE for the $n = 20$ classes in the Pascal VOC 2007 data set. The values are in a 0-4 point scale of VL to VH. For example, classes bicycle (1), bus(5), car(6), etc. are expected to be encountered quite often in our self-driving scenario. This values are also expected to be provided by domain expert.

Likelihood of misclassification thresholds We defined the thresholds for this experiment as $0 < 0.027 < 0.03 < 0.08 < 0.1096 < 0.20 < 1$. The thresholds have been derived after making the analysis described in Section 4.1.2. To determine the values we need to first look at the confusion matrices and identify the minimum and maximum LoM, then we propose five different values between 0 and the maximum LoM so that we are able to map the LoM values to a 0-5 scale of VL-VH. Using the likelihood of misclassification thresholds we obtained the intervals for the risk levels as follows:

- VL:[0, 0.027)
- L:[0.027, 0.03)
- M:[0.03, 0.08)
- H:[0.08, 0.1096)
- VH:[0.1096, 0.20)
- P:[0.20, 1]

Maximum acceptable risk level For the experiment being described we proposed the acceptable risk level to be $\tau = H$, which indicates that we want to reduce the risk associated with any concerns identified as having a risk value $> H$.

Table 4.1: *Impact* for the risk-aware RTOD method evaluated on the Pascal VOC 2007 dataset provided on a 0-4 scale of VL-VH. The cells with a dash are misclassifications considered irrelevant for the context or that do not make sense.

Actual Class \ Predicted Class	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	-	-	-	-	-	2	2	-	-	-	-	-	-	2	4	-	-	-	-	-
2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
5	-	3	-	-	-	-	3	-	-	-	-	-	-	3	2	-	-	-	2	-
6	-	3	-	-	-	2	-	-	-	-	-	-	-	4	3	-	-	-	3	-
7	-	-	-	-	-	-	-	-	-	4	-	2	3	-	4	-	4	-	-	-
8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	4	-	-	-	-	2	-	-	-	1	-	-	-
10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
11	-	-	-	-	-	-	-	4	-	2	-	-	3	-	2	-	0	-	-	-
12	-	-	-	-	-	-	-	0	-	0	-	0	-	-	-	-	0	-	-	-
13	-	4	-	-	-	0	4	-	-	-	-	-	-	-	4	-	-	-	-	-
14	-	4	-	-	-	4	4	4	-	4	-	4	4	4	-	-	-	-	-	-
15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
16	-	-	-	-	-	-	-	4	-	4	-	4	4	-	4	-	-	-	-	-
17	-	-	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-
18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
19	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 4.2: Likelihood of encounter (*LoE*) for the 21 classes of the Pascal VOC dataset used to perform the evaluation of the proposed approach, it is given on a 0-4 scale of VL-VH.

Class	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
LoE	0	4	0	0	0	4	4	2	0	2	0	4	2	4	4	0	2	0	0	0

4.2.2.2 Risk-oblivious model training

In this step we trained 21 risk-oblivious RTOD models for 30 epochs with transfer learning [2] where each model took approximately 2 hours to be trained. The obtained

[2]For a detailed explanation of model training using YoloV3 implemented in TensorFlow 2.0 refer to `https://github.com/zzh8829/yolov3-tf2/blob/master/docs/training_voc.md`

Table 4.3: Mean fraction of misclassification $p(i, j)$, for the evaluation of the 21 risk-oblivious models risk-aware RTOD method evaluated on the VOC 2007 dataset.

Actual Class \ Predicted Class	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0.0	0.0	0.0313	0.031	0.0003	0.0007	0.0077	0.0007	0.0067	0.0005	0.0002	0.0001	0.0001	0.0003	0.0132	0.0019	0.009	0.0009	0.0033	0.0023
1	0.0004	0.0	0.0012	0.0012	0.0067	0.0004	0.0036	0.0	0.007	0.0001	0.0004	0.0005	0.0	0.0155	0.0363	0.0038	0.0012	0.0006	0.0007	0.0002
2	0.0071	0.0003	0.0	0.0115	0.0023	0.0	0.0002	0.0071	0.0037	0.0029	0.0005	0.0059	0.0012	0.0	0.0418	0.0058	0.0212	0.0002	0.0001	0.0025
3	0.0107	0.0001	0.0122	0.0	0.0037	0.0034	0.0195	0.0001	0.0052	0.0008	0.0004	0.0002	0.0002	0.0	0.018	0.0054	0.0107	0.0016	0.005	0.0028
4	0.0	0.0002	0.0021	0.0013	0.0	0.0003	0.0018	0.0	0.0084	0.0	0.0037	0.0	0.0	0.0	0.0196	0.0045	0.0004	0.0017	0.0001	0.0035
5	0.0	0.0	0.0005	0.0051	0.0006	0.0	0.143	0.0001	0.0005	0.0003	0.0	0.0	0.0001	0.0	0.0135	0.0005	0.0008	0.0009	0.0213	0.0016
6	0.0004	0.0006	0.0002	0.0019	0.0002	0.0029	0.0	0.0	0.0009	0.0	0.0002	0.0	0.0003	0.0037	0.014	0.0008	0.0005	0.0009	0.001	0.0006
7	0.0001	0.0	0.0148	0.0007	0.0017	0.0002	0.0013	0.0	0.0034	0.0002	0.0013	0.031	0.0003	0.0	0.0256	0.007	0.0045	0.0055	0.0001	0.004
8	0.001	0.0007	0.0005	0.0006	0.0064	0.0	0.0021	0.0006	0.0	0.0001	0.0114	0.0012	0.0	0.0001	0.0352	0.0162	0.0003	0.0449	0.0	0.0062
9	0.0031	0.0	0.0113	0.0101	0.0	0.0001	0.0025	0.0015	0.0041	0.0	0.0	0.019	0.0172	0.0	0.0075	0.0005	0.0598	0.0004	0.0007	0.0003
10	0.0	0.0004	0.0003	0.0005	0.0145	0.0001	0.0012	0.0	0.0949	0.0	0.0	0.0004	0.0	0.0	0.0234	0.0143	0.0002	0.0042	0.0	0.0006
11	0.0004	0.0003	0.0214	0.001	0.0007	0.0001	0.0004	0.0403	0.0074	0.0092	0.0001	0.0	0.012	0.0001	0.0493	0.0011	0.0156	0.0166	0.001	0.0004
12	0.0005	0.0004	0.0075	0.0023	0.0002	0.0002	0.0023	0.0059	0.0022	0.0375	0.0	0.0271	0.0	0.0007	0.0414	0.0	0.0126	0.0016	0.0004	0.0003
13	0.0001	0.0287	0.0003	0.0009	0.0029	0.0005	0.0308	0.0	0.0019	0.0004	0.0006	0.0001	0.0005	0.0	0.0989	0.0019	0.0011	0.0	0.0009	0.0015
14	0.0001	0.0006	0.0007	0.0012	0.0011	0.0002	0.0015	0.0004	0.0044	0.0	0.0007	0.0006	0.0003	0.001	0.0	0.001	0.0003	0.0021	0.0004	0.0006
15	0.0003	0.0006	0.0045	0.0035	0.0059	0.0018	0.0007	0.0001	0.0262	0.0	0.0045	0.0	0.0001	0.0002	0.0177	0.0	0.0005	0.0039	0.0001	0.0052
16	0.0012	0.0001	0.033	0.0039	0.0003	0.0	0.0023	0.0022	0.002	0.0462	0.0003	0.0352	0.0071	0.0	0.0084	0.0002	0.0	0.0002	0.0001	0.0005
17	0.0001	0.0	0.0006	0.0006	0.0022	0.0006	0.0041	0.0007	0.1271	0.0	0.0152	0.0037	0.0	0.0001	0.0407	0.0065	0.0004	0.0	0.0001	0.0045
18	0.001	0.0	0.0002	0.0084	0.001	0.0123	0.0117	0.0001	0.001	0.0004	0.0003	0.0003	0.0004	0.001	0.0084	0.0002	0.0005	0.0011	0.0	0.0025
19	0.0005	0.0	0.0004	0.0006	0.0036	0.0024	0.0054	0.0013	0.0294	0.0	0.0018	0.0	0.0	0.0	0.0095	0.0081	0.0	0.0032	0.0002	0.0

models have an average mAP score of 0.4310 at IoU=0.5 (i.e. a valid detection is considered when the IoU between the ground-truth and the prediction is ≥ 0.5) and an average residual risk of 3.54.

4.2.2.3 Risk assessment

Using the risk-oblivious models we calculate the mean fraction of misclassification, as described in Section 4.1.2, the results are shown in Table 4.3. Next, using the likelihood of misclassification thresholds and Table 4.3, we obtain the *fLoM* presented in Table 4.4. The *fLoM* is given on a scale of VL to VH with dashes signifying that the misclassifications are considered irrelevant for the context, e.g. the class sofa (17) predicted as potted plant (15), the class tv/monitor (19) predicted as chair (8).

To calculate the fractional overall likelihood *fOL* we then combine the *LoM* and the *LoE* using the look up Table 3.1 to obtain Table 4.5. Finally, the *OL* and the *impact* are mapped using Table 3.2 to obtain the *frisk* presented in Table 4.6, the shading of the cells indicate all the concerns with a risk value above the maximum acceptable risk level $\tau > H$ (3). The five concerns identified are shown in Table 4.7.

4.2.2.4 Risk-aware model training

For each of the concerns identified in the previous step we then created 21 models, consisting of 7 models for each value of $\omega \in \{2, 5, 10\}$. In this way we obtain a total of 105 risk-aware models with an average mAP score of 0.4317 at IoU=0.5, and average residual risk of 3.44. We compare the performance of the risk-aware and the risk-oblivious models in Figure 4.4a. We can see from this figure that some of the risk-aware models outperform the best risk-oblivious models in both residual risk and mAP score. We note that risk-awake models for: bus predicted as car (yellow marker); motorbike predicted as car (red marker); and bicycle predicted as person

Table 4.4: Mean *fLoM* for the evaluation of the 21 risk-oblivious models risk-aware RTOD method evaluated on the Pascal VOC 2007 dataset provided on a 0-4 scale of VL-VH. The cells with a dash are misclassifications considered irrelevant for the context or that do not make sense.

		Predicted Class																			
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Actual Class	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	1	-	-	-	-	-	0.01	0.13	-	-	-	-	-	-	0.57	2.12	-	-	-	-	-
	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	5	-	0.00	-	-	-	-	4.36	-	-	-	-	-	-	0.00	0.50	-	-	-	0.78	-
	6	-	0.02	-	-	-	0.10	-	-	-	-	-	-	-	0.13	0.51	-	-	-	0.03	-
	7	-	-	-	-	-	-	-	-	-	0.00	-	2.02	0.01	-	0.94	-	0.16	-	-	-
	8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	9	-	-	-	-	-	-	-	0.05	-	-	-	-	0.63	-	-	-	2.59	-	-	-
	10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	11	-	-	-	-	-	-	-	2.20	-	0.34	-	-	0.44	-	2.38	-	0.57	-	-	-
	12	-	-	-	-	-	-	-	0.21	-	2.15	-	1.03	-	-	-	-	0.46	-	-	-
	13	-	1.56	-	-	-	0.01	2.01	-	-	-	-	-	-	-	3.63	-	-	-	-	-
	14	-	0.02	-	-	-	0.00	0.05	0.01	-	0.00	-	0.02	0.01	0.03	-	-	-	-	-	-
	15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	16	-	-	-	-	-	-	-	0.08	-	2.32	-	2.1	0.26	-	0.31	-	-	-	-	-
	17	-	-	-	-	-	-	-	-	4.19	-	-	-	-	-	-	-	-	-	-	-
	18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	19	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

(blue marker); reduce risk when compared to risk-oblivious models (brown marker), with residual risk reduced from a minimum of 2.7, for the risk-oblivious models, to under 2.5 for risk-aware models. In the following section we describe the results of combining the individual models in an ensemble.

4.2.2.5 Ensemble synthesis

In this step, we first randomly selected 2 risk-aware models per concern and 2 of the risk-oblivious models from the mAP/residual-risk Pareto front for each type of model, obtaining a total of 12 models. From this set of models the GA was instructed to synthesise an ensemble allowing $l = 4$ models. We allowed the GA to run for 125 generations and recorded the time to synthesise the ensemble as well as the mAP scores for training an testing sets, the results are shown in Table 4.8. We see that the amount of time required to synthesise the ensemble is significant i.e. we require 28 hours to obtain a solution at 125 generations. This time is affected by the number of models in the ensemble, the number of classes and also the number

Table 4.5: Mean *fOL* for the evaluation of the 21 risk-oblivious models on risk-aware RTOD method evaluated on the Pascal VOC 2007 dataset provided on a 0-4 scale of VL-VH. The cells with a dash are misclassifications considered irrelevant for the context or that do not make sense.

		Predicted Class																			
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Actual Class	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	1	-	-	-	-	-	1.01	1.13	-	-	-	-	-	-	1.57	3.12	-	-	-	-	-
	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	5	-	1.00	-	-	-	-	4.68	-	-	-	-	-	-	1.00	1.50	-	-	-	1.78	-
	6	-	1.02	-	-	-	1.10	-	-	-	-	-	-	-	1.13	1.51	-	-	-	1.03	-
	7	-	-	-	-	-	-	-	-	-	1.00	-	2.01	1.0	-	1.47	-	1.08	-	-	-
	8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	9	-	-	-	-	-	-	-	1.02	-	-	-	-	1.32	-	-	-	2.3	-	-	-
	10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	11	-	-	-	-	-	-	-	3.20	-	1.34	-	-	1.44	-	3.38	-	1.57	-	-	-
	12	-	-	-	-	-	-	-	1.10	-	2.08	-	1.52	-	-	-	-	1.23	-	-	-
	13	-	2.56	-	-	-	1.01	3.01	-	-	-	-	-	-	-	4.31	-	-	-	-	-
	14	-	1.02	-	-	-	1.0	1.05	1.01	-	1.00	-	1.02	1.01	1.03	-	-	-	-	-	-
	15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	16	-	-	-	-	-	-	-	1.04	-	2.16	-	2.05	1.13	-	1.16	-	-	-	-	-
	17	-	-	-	-	-	-	-	-	1.73	-	-	-	-	-	-	-	-	-	-	-
	18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	19	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

of samples in the data set.

Figure 4.4b shows the set of Pareto-optimal ensembles found by the GA at 10 (blue Pareto front), 20 (yellow Pareto front), 25 (green Pareto front), 30 (red Pareto front), 40 (purple Pareto front) and 125 generations (brown Pareto front). As the number of generations increases, the residual risk decreases and the mAP score improves, with diminishing returns as more GA generations are produced. For example, after 10 generations the minimum residual risk is just below 2 with a mAP score below 0.502, after 30 generations the residual risk has been reduced to 1.5 and the mAP score reached 0.504, at the maximum, 125, generations the residual risk is approximately 1.2 and the mAP score is 0.506.

All of the ensembles achieved much better mAP scores than the risk-oblivious and risk-aware models and this is shown in Figure 4.4c. We can observe how the mAP for all the individual models is below 0.45 whereas the ensemble achieves a mAP score of 0.506 at the minimum risk. As expected, additional mAP performance can be traded against risk by selecting an ensemble from the Pareto front with a

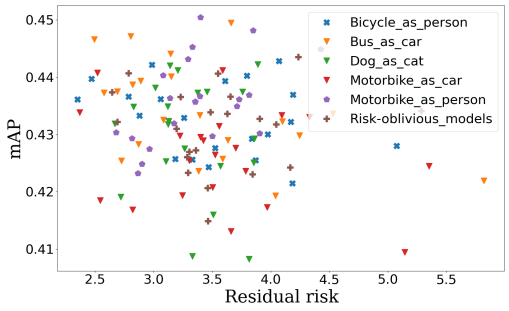
Table 4.6: *frisk* values for the class-pairs in the Pascal VOC 2007 data set, the values are provided on a 0-4 scale of VL-VH. The cells with a dash are misclassifications considered irrelevant for the context. Highlighted in red the concerns with risk value > 3 (H).

Actual Class \ Predicted Class	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	-	-	-	-	-	1.01	1.13	-	-	-	-	-	-	1.57	3.12	-	-	-	-	-
2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
5	-	1.0	-	-	-	-	4.68	-	-	-	-	-	-	1.0	1.5	-	-	-	1.78	-
6	-	1.02	-	-	-	1.1	-	-	-	-	-	-	-	1.13	1.51	-	-	-	1.03	-
7	-	-	-	-	-	-	-	-	-	1.0	-	2.0	1.0	-	1.47	-	1.08	-	-	-
8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	1.02	-	-	-	-	1.32	-	-	-	1.43	-	-	-
10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
11	-	-	-	-	-	-	-	3.2	-	1.34	-	-	1.44	-	2.69	-	0.52	-	-	-
12	-	-	-	-	-	-	-	0.37	-	0.69	-	0.5	-	-	-	-	0.41	-	-	-
13	-	2.56	-	-	-	0.34	3.01	-	-	-	-	-	-	-	4.31	-	-	-	-	-
14	-	1.02	-	-	-	1.0	1.05	1.01	-	1.0	-	1.02	1.01	1.03	-	-	-	-	-	-
15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
16	-	-	-	-	-	-	-	1.04	-	2.16	-	2.05	1.13	-	1.16	-	-	-	-	-
17	-	-	-	-	-	-	-	-	0.58	-	-	-	-	-	-	-	-	-	-	-
18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
19	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

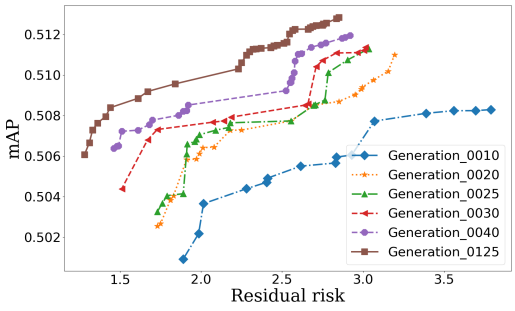
Table 4.7: Risk concerns identified for the RTOD with class ids shown in brackets.

ID	True Class	Predicted Class	Risk
1	Motorbike (13)	Car (6)	3.01
2	Bicycle (1)	Person (14)	3.12
3	Dog (11)	Cat (7)	3.2
4	Motorbike (13)	Person (14)	4.31
5	Bus (5)	Car (6)	4.68

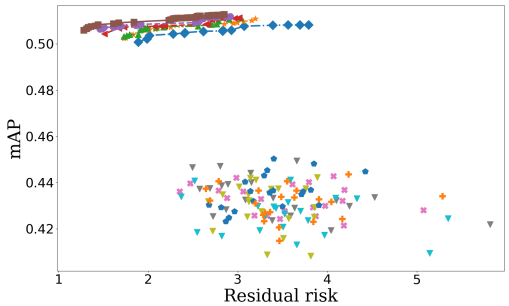
different set of trained weights, e.g. in the brown Pareto front obtained at 125 generations we can opt to select an ensemble that yields to a mAP score of 0.513, which is the highest possible, at the cost of assuming a residual risk of approximately



(a) Performance of single models



(b) Ensemble learning



(c) Performance of ensemble vs single models

Figure 4.4: Performance of single models and ensemble results of the proposed approach evaluated on the VOC 2007 data set with acceptable risk level $\tau = M$.

Table 4.8: Time to synthesise the ensemble and mAP score of it for the evaluation on the VOC2007 dataset

No of generations	Required time (in hours)	mAP (training) @IoU=0.5	mAP (testing) @IoU=0.5
125	28	0.5061	0.5366

2.8.

Table 4.9 presents the risk values per concern of the synthesised ensemble at 125 generations for the smallest risk obtained during training and testing of the ensemble. The last row of the table also shows the residual risk. We can observe how during the training of the ensemble the concerns 1) motorbike (13) predicted as car (6), 2) bicycle (1) predicted as person (14) and 3) dog (11) predicted as cat (7) were fully mitigated below the maximum acceptable risk threshold $\tau = H$ (3), with corresponding risk values of 1.36, 1.8 and 2.8. As for concerns 4) motorbike (13) predicted as person (14) and 5) bus (5) predicted as car (6) they could not be alleviated below the proposed $\tau = H$ (3); nevertheless, in both of them the risk was decreased from VH (4) to H (3).

When the ensemble was tested, we found that 3 concerns remained above the maximum acceptable threshold: 2) bicycle (1) predicted as person (14); 4) motorbike (13) predicted as person (14); and 5) bus (5) predicted as car (6). However, the risks were generally reduced, except for concern 2 whose risk value increased from

Table 4.9: Risk values per concern during training and testing of the ensemble evaluated on the Pascal VOC 2007 dataset. Highlighted in red the concerns with risk value > 3 (H).

Id	Concern		Initial risk	Ensemble risk-training	Ensemble risk-testing
	Actual	Predicted			
1	Motorbike(13)	Car(6)	3.01	1.36	1.00
2	Bicycle(1)	Person(14)	3.12	1.80	3.50
3	Dog(11)	Cat(7)	3.20	2.80	1.31
4	Motorbike(13)	Person(14)	4.31	3.97	3.52
5	Bus(5)	Car(6)	4.68	3.11	3.77
Residual risk			3.32	1.08	1.79

Table 4.10: Mean RTOD time (in seconds) to process a single image in a batch of 100 images

Ensemble Size	1	2	4	6	8	16
RTOD Time	0.137	0.267	0.527	0.776	1.036	2.073

3.12 to 3.5. We can also see how the initial risk of 3.32 was reduced to 1.79 for the test set, which indicates that despite the fact of one concern been exacerbated, it was for the benefit of lowering the overall residual risk. This occurs since the GA will trade off risks for individual concerns in order to minimise the overall risk. Similarly, we observe that the balance of objects present in the testing and training set can impact our training and evaluation. The number of objects for the class bicycle in the split for testing the ensemble has a small number of instances, and therefore, a single misclassification has a significant impact of the reported results. For example, if the total of objects for class A is 10 and 3 objects are misclassified the mean fraction of misclassification will be $p = 3/10$, from this we can see that each new misclassification will significantly increase the value of p, and hence the total risk, even when the number of misclassifications is relatively low. As previously mentioned the data quality has a significant impact on the quality of the results that our method provides; nonetheless, because all the other concerns were mitigated and the residual risk was significantly reduced we conclude that this establishes the benefit of our method.

To evaluate the impact of increasing complexity on object detection time, we built a series of ensembles of increasing size and recorded the time taken to process 100 samples. As shown in Table 4.10, the mean time to process a sample on a desktop workstation increases linearly with the number of models in the ensemble. These times, which can be reduced considerably by applying the optimisations described in [118], indicate that our method is of practical use.

Finally, Figure 4.5 illustrates the performance of a synthesised ensemble and its constituent models when applied to four images from the data set. A tick in the top

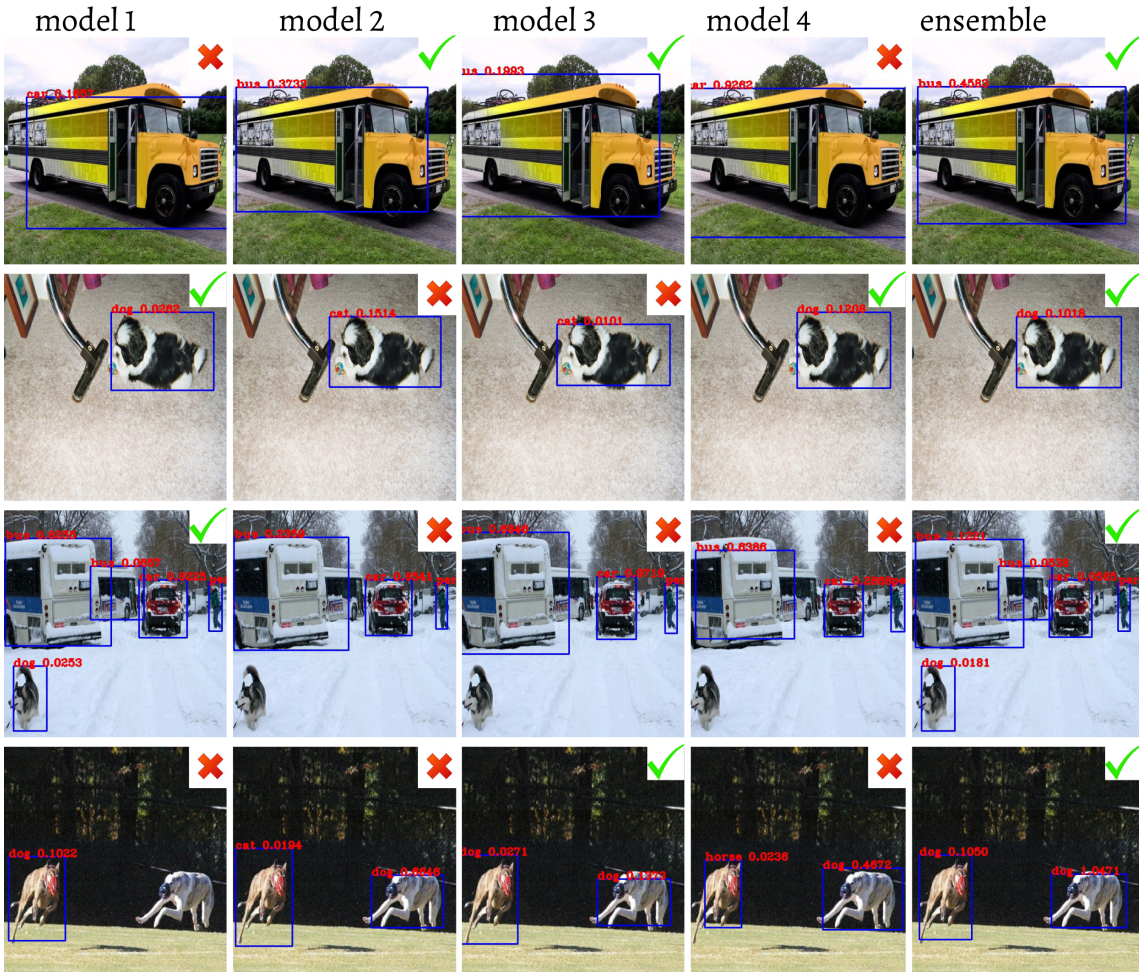


Figure 4.5: Object detection comparing four models to the risk-aware RTOD ensemble synthesised from them.

right corner of the image indicates the model correctly identified all objects, whilst a cross indicates that at least one object was not detected or was misclassified. We can see that the ensemble is able to successfully identify objects even when the majority of models are unable to do so.

4.2.3 Discussion

We introduced a new method for the synthesis of risk-aware ML ensembles for real-time object detection. The experimental work to evaluate our method was performed on the widely used Pascal VOC 2007 data set for the maximum acceptable risk level $\tau = H$ at which we encountered five concerns and we synthesised an ensemble allowing four models for 125 generations of the GA. The obtained results, presented in Section 4.2.2, suggest that our method can effectively mitigate risk, supporting the development of dependable RTOD-based systems for safety-critical applications.

We observed how the required time to synthesise the ensemble is large, we require around 14 minutes per generation of the GA, this time is affected by the number of individuals per generation (50 for our experiments), the number of models allowed in the ensemble, the number of classes in the data set, the number of objects per image and the number of images in the data set. In Section 4.1.4 we presented algo-

rithms 1, 2, and 3 which first groups sets of objects identified by the same individual models based on the IoU metric and then combines each model's prediction with sets of weights to determine the ultimate prediction of the ensemble. This process is repeated for each element in the ground truth and for each model allowed in the ensemble which explains the amount of time required to synthesise the ensemble.

When looking at the initial risk values per concern and how these values compare with that obtained using the ensemble we observe the gain of using our approach as the risk is always mitigated, and this is reflected in the residual risk, whose tendency is to be decreased as generations of the GA elapse while improving the mAP score. We also note that the residual risk obtained during the evaluation of the ensemble on the test data set is higher than that obtained during the training of the ensemble, which suggests that our model overfitted, i.e. it had a better performance during training. This is possibly because it was trained for longer than needed but it could also be related with the data distribution in the training and testing splits. For example, if a class is underrepresented in the test split and a lower number of total instances of such class is present compared to the training split, each new misclassification during testing leads to an increment in risk. However, the improvement of our approach is significant as the initial risk dropped from 3.32 to 1.79 which shows the advantage in risk mitigation of our approach.

4.2.4 Threats to Validity

Construct validity. These threats can arise from the methodology employed during the design of the experimental study and the underlying assumptions. This includes assumptions made during the risk assessment stage, which should consider RTOD, as well as proper inclusion of a risk profiling of the relevant class pairs.

To address this threat, we meticulously derived the likelihood of misclassification for object detection, which presented added complexity due to the association of multiple bounding boxes with multiple objects. We also considered the need to create a confusion matrix, to identify high likelihood of misclassification class-pairs, while simultaneously considering the mAP as a performance metric for the ensemble.

Regarding risk assessment, we adhered to the ISO/IEC 31010 risk standard, which provides guidance on implementing risk management principles and processes. This standard specifically offers guidance on the selection and application of risk assessment techniques.

By following these measures, we aimed to mitigate threats related to the experimental methodology, assumptions made during risk assessment, and the accurate evaluation of ensemble performance. The inclusion of ISO/IEC 31010 guidance ensured a robust approach to risk management and assessment throughout the project.

Internal validity. These threats may arise from the construction of the ensemble and the need for accurate integration of the predictions provided by each individual RTOD model. Additionally, there is a potential threat regarding the efficiency of the synthesised ensemble in merging the models' predictions for real-time object detection.

To address this concern, we proposed a suite of algorithms specifically designed for the ensemble synthesis stage to effectively combine the bounding box predictions from the individual models. Furthermore, to achieve real-time object detection,

we integrated our proposed algorithms into the YoloV3 object detection pipeline, ensuring an efficient knowledge combination process.

In order to assess the practical usability of our approach, we synthesised ensembles of different sizes and measured the time required to process a single image. The obtained values indicate that our method is indeed practical and can be effectively employed in real-time object detection scenarios.

External validity. While we acknowledge that one case study alone may not provide comprehensive validation of the effectiveness of our approach, we have chosen to evaluate our method using the widely used Pascal VOC 2007 dataset. This dataset is highly regarded in the field of computer vision, particularly for object detection tasks, due to its diverse range of object classes, large-scale nature, and real-world complexity.

The selection of the Pascal VOC 2007 dataset allows us to assess our approach's performance and capabilities within the context of this established benchmark. However, as part of our future work, we aim to further investigate the effectiveness of our approach by applying it to additional case studies. By doing so, we can expand our understanding of its applicability and robustness in diverse scenarios.

4.3 Related work

Several existing RTOD solutions use ensembles of ML models [22, 2, 140, 50, 146]. Unlike our method, most of these solutions [22, 50] combine the output of their component ML models with equal weights, disregarding the fact that each model may be better at predicting certain classes. [22] presents an ensemble algorithm that can be applied with any object detection model. The method has been employed to define a test-time augmentation procedure for object detection models. Similar to our approach, they threshold the IoU to group which predictions correspond to the same object; nonetheless, they add the restriction that all boxes in the group should be predicting the same class. Then in order to consider each group as a valid detection, they use 3 strategies: (i) affirmative, in this strategy, whenever one of the models that produced the initial predictions says that a region contains an object, such a detection is considered as valid; (ii) consensus, in this case, the majority of the initial models must agree to consider that a region contains an object. The consensus strategy is analogous to the majority voting strategy commonly applied in ensemble methods for image classification; (iii) unanimous, in the last strategy, all the models must agree to consider that a region contains an object. This is different to our approach because we do not consider the condition of all boxes for the same object predicting the same class. We recognise that some models make different predictions for the same object i.e. some predictions are correct and some are not, and this is the problem we are trying to solve, to decrease the number of such misclassifications by combining the weighted predictions of different models in the ensemble.

In [162] the authors propose the construction of a detector with higher detection performance and without lowering the detection speed to perform RTOD, their idea is to combine a one-stage framework, context modeling, and multi-scale representation. They apply dilated convolution to object detection and build a context detection module based on the fact that dilated convolution extends the receptive

field without increasing the number of computations. At the same time, they capture fine-grained details through multi-scale representation to enhance the representation capability of the model. In addition, they combine the idea of ensemble learning to further improve the performance of the detector. Their ensemble first concatenates the inference results of two models at inference time. They assume that a single model can generate N prediction bounding boxes; thus, they will get 2N bounding boxes after inference. Then they perform a non-maximum suppression algorithm on the 2N bounding boxes to obtain the final prediction bounding boxes. Finally, they apply NMS with IoU overlap of 0.45 per class and keep the top 200 detections per image (they call this feature ensembling). They divide their ensemble modes into three strategies but the most relevant is the ensemble of different models which combines a model based on VGG16 (a DNN 16 layers deep [134]), and the model trained by feature ensembling. The idea is interesting and already has in mind the overload that the ensemble may impose in the predictions, that is way they use one-stage detectors, but is limited in scalability as only two models can be included in the ensemble, besides their only interest resides in improving the mAP score, in contrast, our method supports the synthesis of ensembles of different sizes and uses GA that aim to improve the mAP score and to mitigate risk of misclassification for sets of concerns.

Another interesting ensemble method is weighted boxes fusion (WBF) [140] which proposes a method for fusing predictions of different object detection models. Unlike NMS and soft-NMS methods that remove some predictions, the proposed WBF method uses confidence scores of all proposed bounding boxes to construct average boxes. This method uses weighting in combining the outputs of their ML models, this weights are determined using a basic heuristic that is focused on improving accuracy by constructing averaged boxes. The proposed technique (WBF) is also implemented in SyNet [2] that combines a multi-stage RTOD with a single-stage one with the motivation of decreasing the high false negative rate of multi-stage detectors and increasing the quality of the single-stage detector. This approach considers a detection valid if the IoU of the ground truth and the prediction is over a threshold, which is the standard in object detection. As mentioned, for the combination of the different RTOD models they implement WBF which utilises all predictions in order to find bounding box clusters. Both WBF and SyNet, only focus on improving accuracy by constructing averaged boxes, without explicitly reducing the number of misclassifications. In contrast, our method uses a multi-objective genetic algorithm that yields RTOD ensembles with Pareto-optimal trade-offs between object detection accuracy and risk.

[57] proposes a method called the Wasserstein loss based model for object detection to assign different weights to one sample identified as different classes with different values. The distance metric is designed by combining the cross-entropy (CE) or binary cross-entropy (BCE) with Wasserstein distance to learn the detector considering both the discrimination and the seriousness of different misclassifications. The focus of this method is on avoiding unacceptable misclassifications caused by CE/BCE loss-based object detection methods. They apply the Wasserstein loss as an alternative to empirical risk minimisation to improve classification accuracy. Specifically, they calculate the Wasserstein distance between a softmax prediction histogram and its ground-truth label. By defining the ground metric based on the appearance similarity and misclassification severity (e.g., the distance between *bike*

and *car* is larger than *bike* and *motorbike*), classification performance for each object can be measured related to inter-class correlations. The Wasserstein metric is a distance function defined between probability distributions in a given metric space. The authors adapt this for object classification, using the ground-distance matrix defined by dividing the classes into different groups using prior knowledge and measuring the distance between different groups using a Gaussian filter. They divide the classes of the data set in groups, e.g. group 1: bus, truck and car, group 2: traffic light, traffic sign, etc. and make the assumption of risk free misclassification for objects in the same group; however, in the self-driving domain making a distinction between a traffic light and a traffic sign seem to be highly important, and the same occurs with the rest of the groups, therefore, we argue that making a distinction between individual misclassifications is essential. Although, this method cares about misclassification of groups of classes, they fail to introduce a proper risk metric as we do.

Finally, the RTOD ensemble generation solution devised by [146] uses a genetic algorithm to optimally combine the ML models from the ensemble. Their fitness function aims to satisfy two characteristics, (i) fewer classifiers and (ii) lower error rate. The authors in this work claim that is preferable that fewer classifiers give a correct prediction for a given object rather than more classifiers. This contradicts our approach that has demonstrated in our experiments that adding more models to the ensemble yields to a better performance. Besides, this solution focuses exclusively on optimising the ensemble accuracy, and therefore, does not considers the risks associated with different misclassifications like our method. To the best of our knowledge, no existing RTOD approach considers the risks corresponding to different misclassifications.

4.4 Summary

This chapter discussed a method for the development of *risk-aware ML ensembles* for real-time object detection. We evaluated our method on the widely used PASCAL VOC 2007 data set and we implemented a version of the object detector YoloV3 to train risk-oblivious models. We set a maximum acceptable risk level of $\tau = H$ and obtained a set of five concerns. We then trained a set of risk-aware models for each of the identified concerns and synthesised Pareto-optimal RTOD ensembles. The experimental results presented in this section suggest that our method can effectively mitigate risk, supporting the development of dependable RTOD-based systems for safety-critical applications.

Chapter 5

Dynamic Selection of Risk-aware Object Detection Ensembles

This chapter describes an approach that proposes the use of risk-aware ML ensembles with dynamic switching between models as a Real-time Adaptation System (RAS) moves between Operational Design Domain (ODD) regions. We assessed the effectiveness of the approach measuring safety by observing the number of crashes and number of correct actions taken during journeys of the autonomous vehicle. The chapter starts with an introduction in Section 5.1; we then describe the five components of the approach in Section 5.2. Next, we move to the evaluation presented in Section 5.3. Related work is presented in Section 5.4 and, finally, we present a summary of the chapter in Section 5.5.

5.1 Introduction

Ensemble Methods are inspired by the wisdom of the crowd phenomenon, in which many weaker learner predictions are combined in order to obtain a more accurate prediction [125]. The hypothesis is that by combining multiple base learners, the errors of a single model is likely to be compensated by other base learner models and, as a consequence, the overall performance of the ensemble would be optimal [4, 167]. Ensemble methods composed of DNN models have been successfully applied to object detection tasks such as scene recognition [32, 106], x-ray inspection [75], object-tracking [171] and safety critical systems such as autonomous vehicles [154, 15] and airborne collision avoidance systems [70, 68]. In all of these cases the ensemble methods have shown to perform remarkably well, in most of the cases surpassing humans or hand-crafted algorithms [70].

The ensembles using DNN models as base learners are usually trained in large data sets and expected to generalise to previously-unseen inputs. However, this is not always the case and it has been shown that even small perturbations to images produce a detection failure in the base learners [144, 55, 166] and, as a consequence, in the output of the ensemble. Such disturbances in the input of the ensembles could be produced by significant changes in the real world operating domain, or context perturbations. Such changes could occur in different weather conditions, such as haze or rain, but also through differences in lighting conditions, such as dawn, daylight and dusk [108, 154]. Because the initial ensembles could have been synthesised without considering all these possible scenarios the likelihood

of misclassifications (LoM) and the probability of missing objects in the detection stage may also increase. In addition, the likelihood of encountering different classes (LoE) at different times of day, or in different geographical areas also varies, e.g. at peak hours you are more likely to encounter pedestrians and cyclists in areas surrounding schools. These changes in LoE and LoM therefore affect the list of risk concerns as we move through operating domain region. This unexpected behaviour in the synthesised ensembles is likely to result in decisions that would compromise the safety of the system, restricting its usage in safety-critical applications.

We have previously shown how, by integrating information from domain experts concerning the risk profile of an operating domain we are able to synthesise risk-aware ensemble models that successfully mitigate the risk for each of their concerns. In this chapter we propose an approach that mitigates the uncertainty that affects real-world applications [1, 20, 62, 138] by using risk-aware ML ensembles with dynamic switching between models as a RAS moves from one ODD region to another. For this work we assume that such ODD regions can be identified in advance, for instance we may identify that the LoE and Impact of misclassification varies as we move from a motorway to a residential area, and that by identifying concerns which are appropriate for each region we can improve safety.

To demonstrate the effectiveness of our approach we make use of the Gazebo [74] simulator and the robot operating system (ROS) [142] to simulate a Turtlebot robot [1] acting as an autonomous-driving vehicle. The vehicle moves around a circuit completing laps, each of them representing different ODD regions. As the autonomous-driving vehicle moves around the circuit it encounters different objects and, once an object is detected, it is expected to choose between a set of actions to operate safely and avoid a collision. The vehicle is equipped with a camera and makes use of a risk-aware RTOD ensemble as described in Chapter 4. In addition the vehicle has the ability to switch between different risk-aware synthesised ensembles as it moves into a different region. The evaluation presented shows the benefit of dynamically switching between ensembles depending on the ODD region by successfully decreasing the number of unsafe actions taken and the number of crashes of the self-driving vehicle.

5.2 Approach

In this section we describe the main components of our proposed approach depicted in Figure 5.1 which includes (i) the environment where the system will be deployed composed of different ODD regions; (ii) The ODD detector which allows our system to identify in which region it is currently operating through analysis of sensors mounted on the vehicle; (iii) the controller whose main task is to seamlessly switch between the different available risk-aware ensembles; (iv) the decision framework that decides the speed of the vehicle based on the objects detected by the risk-aware RTOD ensembles; and (v) the system we intend to control for its safe-autonomous navigation in the environment. The components are described in detail in the following sections.

[1] `https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/`

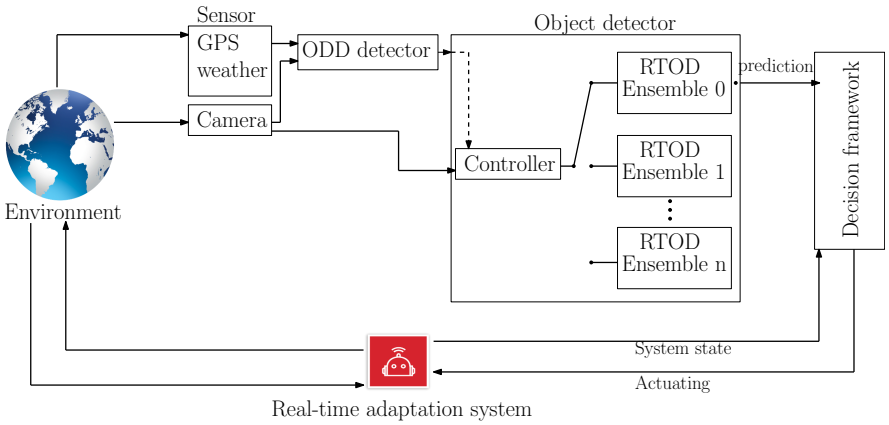


Figure 5.1: The components and interactions between components of the proposed approach for the dynamic selection of risk-aware object detection ensembles.

5.2.1 The environment

We consider the environment to be the external world in which the system is expected to operate. Formally it has been defined as a grouping of external and observable entities that the system can only observe and interact with but can not control [133]. If we intend to deploy a system in the real world, it is paramount to understand and delimit the regions in which it is expected to work to ensure its safe deployment. It is also essential that the uncertainties associated with the environment which impact system safety are understood and considered during system design [72].

In the work we present this means that we have consulted the appropriate domain experts to provide the relevant risk-information. This includes the LoE for all classes as well as the impact that specific misclassifications may have. Based on this information a suitable set of risk concerns is identified and mitigation strategies, which include the synthesis of Pareto-optimal risk-aware ensembles, employed. An effort has been made by [30] to formally define the real-world entities existing in the environment, including their attributes, relationships, and if applicable, behaviors. They propose an ontology which refers to elements that occur in road environments for specifying operational world models for self driving vehicles used to define the ODD. According to [124] an ODD alludes to the functional conditions under which a system has been specifically designed to operate and includes environmental, geographical, and time-of-day restrictions, and/or the requisite presence or absence of certain traffic or roadway characteristics. Following the previous definitions we can state that environment in this work refers to a collection of different ODDs for which the experts-domain are expected to provide their relevant risk-information and each ODD's sets of concerns have been identified as well as the corresponding mitigation strategies.

5.2.2 The ODD detector

The ODD detector is responsible for determining the specific ODD region in which the autonomous system is currently operating. This crucial functionality enables the system's controller to make informed decisions and take appropriate actions based on the identified region.

To accurately detect the ODD region, the detector relies on different sensor in-

puts. For instance, it may utilise a GPS sensor to gather geographical information, allowing the system to recognise changes in location and adapt its behaviour accordingly. By leveraging GPS data, the ODD detector can identify when the system transitions between different geographic regions with distinct operational constraints or requirements.

Moreover, weather condition sensors such as an anemometer can be employed as part of the ODD detector's sensor suite. These sensors enable the system to perceive and account for changes in weather conditions, including factors like wind speed, which may impact the vehicle's performance or necessitate adjustments to its driving behavior.

In addition to the aforementioned sensors, the ODD detector can be further enhanced through the integration of a camera sensor. The camera enables visual perception, allowing the system to analyse the surrounding environment and detect critical elements such as fog, rain, or alterations in lighting conditions. By leveraging the camera's capabilities, the ODD detector can gather real-time visual information and effectively determine the presence of adverse weather conditions or other visual cues about environmental changes that might affect the system's decision-making.

The ODD detector functions include: (i) ODD region identification—the ODD detector determines the specific ODD region in which the autonomous system is currently operating; (ii) sensor integration—the ODD detector integrates data from multiple sensors to gather information about the system's surroundings, e.g., it may incorporate inputs from sensors like GPS, cameras, LiDAR, radar, weather sensors, or other environmental sensors to enhance its understanding of the operational context; (iii) ODD boundary determination—based on the available sensor data, the ODD detector defines the boundaries and constraints of the operational design domain, and establishes the limits within which the autonomous system can safely operate and inform to make appropriate decisions; (iv) decision support—the ODD detector provides crucial information to the system's controller. By identifying the current ODD region and its associated constraints, the ODD detector assists in making informed decisions and selecting appropriate actions for the autonomous system.

In summary, the ODD detector serves as the mechanism for identifying the current ODD region in which the autonomous system is operating. It integrates sensor data, including GPS, weather condition sensors like an anemometer, and visual information from a camera, to accurately perceive geographical, weather-related, and environmental changes. This comprehensive sensory input enables the ODD detector to provide crucial information to the system's controller, empowering it to make context-aware decisions and take appropriate actions within the distinct operational design domains.

5.2.3 The controller

Based on the data provided by the ODD detector the controller needs to decide if a switching from the current RTOD ensemble is necessary and which of the available models is the most suitable. This means that the systems understands the environment where it operates and has the ability to decide its response to changes in the environment. In our approach we assume that we know in advance the set of possible ODD regions in which the system is expected to work (the environment) and

that for each of them a risk analysis has been carried out as described in Chapter 4. The risk analysis includes firstly the consultation of the domain expert to assess the LoE for each relevant class as well as the impact of misclassification for each class-pair. For example, on a motorway we expect LoE to be high for classes such as car, truck or bus. In contrast, classes such as person or bicycle are not expected to be encountered regularly. Similarly, the impact of misclassification changes according to the new region, for example, making a misclassification of a motorbike as a person could have a higher impact in the motorway ODD than in the city centre ODD as this implies that the vehicle should brake severely causing an accident, in the city centre the vehicle is expected to be driving considerably slower and braking is less likely to be problematic. Secondly we identify risk concerns relevant for each ODD which will be determined according to the maximum tolerated risk threshold τ. With this information at hand we can synthesise Pareto-optimal risk-aware ensembles with respect to risk and performance metrics that successfully mitigate the set of risk concerns. The intuition is that as the system moves from one ODD region to another the set of concerns and the amount of risk for each of them changes degrading the performance of the current risk-aware ensemble which may compromise the safety of the system. The switching to a new ensemble ensures that the safety concerns are kept by selecting a model that deals with the new risk concerns identified.

5.2.4 The decision framework

This component gets the set of predictions from the object detector and based on the prediction an action will be taken. Each ground-truth class $c_j \in \{1, 2, \ldots, n\}$ has assigned an action $a_i \in \{1, 2, \ldots, s\}$ that will affect the speed of the system and the action a_i will be selected by the decision framework every time the prediction c'_j contains the relevant class. For instance, if class *car* has assigned the action *fast*, every time the object detector reports class *car* the speed of the vehicle will be increased for a period of time t and after the time has elapsed the speed will be back to a *nominal* speed; as expected, every time the object detector makes a misclassification the wrong action will be executed. If the object detector misses the object, then no action will be reported and we assume a crash happens. This information is recorded during the run of the system and allows us to make an evaluation of how well the chosen risk-aware ensemble performed on the ODD region.

5.2.5 The real-time adaptation system (RAS)

The RAS is a crucial component that operates in a continuous feedback loop, interacting with the environment, reporting its state to the decision framework, and executing appropriate actions based on the provided instructions.

One of the primary functions of the RAS is to continuously monitor and report its state to the decision framework. This involves gathering relevant data about its internal state, sensor readings, and other contextual information. By providing this real-time information, the RAS enables the decision framework to have an up-to-date understanding of the system's current situation, allowing for informed decision-making.

Based on the instructions received from the decision framework, the RAS ex-

ecutes the corresponding actions. These actions can include executing different behaviours, maneuvers, or responses depending on the goals and tasks assigned to the autonomous system. For instance, in the case of a self-driving vehicle, the RAS may be responsible for controlling the vehicle's speed, steering, braking, and other relevant functions necessary for safe and efficient operation.

The RAS also interacts with the environment by observing it and acting upon it to achieve its assigned tasks. This involves processing data from various sensors, such as cameras, LiDAR, radar, or other environmental sensors, to perceive and understand the surroundings. By observing the environment, the RAS can gather critical information about road conditions, obstacles, traffic, and other relevant factors that influence its decision-making and actions.

Furthermore, the RAS actively acts upon the environment based on its assigned tasks and objectives. This can include actions such as changing lanes, maintaining a safe distance from other vehicles, responding to traffic signals, or navigating complex traffic scenarios. The RAS uses its perception capabilities and decision-making algorithms to make informed decisions and execute actions that align with its goals.

5.3 Evaluation

5.3.1 Evaluation methodology

In order to evaluate our proposed approach we built a simulator into which we deployed a vehicle using our method. We then ran multiple episodes and evaluated the system to assess the effectiveness of ensemble switching as the ODD region changes. We assessed effectiveness with respect to safety by measuring the number of crashes, and number of correct actions taken (*slow* and *fast*), during 10 journeys of the vehicle. Each journey is composed of six laps of a simple circular track by the vehicle in the simulation. Using a circular track allows us to easily simulate longer journeys. For each lap, we randomly deploy cubes with images mapped to their faces which represent objects, in scenes, which are expected to be seen in the current ODD region. The simulator is constructed with a set of light sources distributed on the ceiling which, combined with the speed of the robot and angles of detection, simulate environmental disturbances. An example of the simulator is shown in Figure 5.2.

We found that when detecting objects at a distance the number of misclassifications was very high. We then implemented a simple decision framework in which the robot was assumed to be able to calculate the distance to the detected object and hence disregard any classification further than a threshold from the robot. When the robot is in the detection area shown in Figure 5.3 and an object is detected we record the prediction and the object cube is removed from the simulation. Based on the detection the robot will take an action (i) if the predicted classes are bus, dog or bicycle the vehicle will take the action *slow*; if the predicted classes are car or motorbike the vehicle will increase its speed or take the action *fast*; and if the prediction is any other class then the vehicle will *ignore* which means no change to the current speed of the vehicle. The action taken will be executed during the three seconds after the detection was made after which it will return to a nominal speed. Beyond the detection area we defined the crash area also represented in Figure 5.3, when the vehicle enters this area a crash is registered and the image is removed from

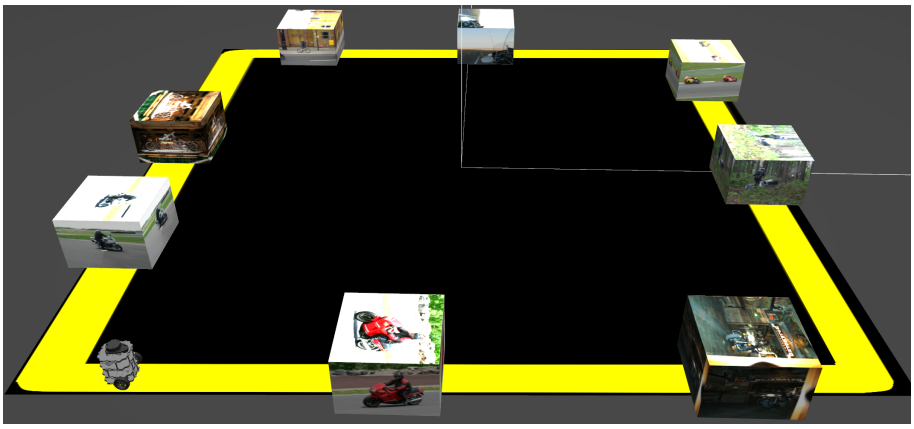


Figure 5.2: Simple track-circuit with different images displayed on cubes to be identified by the autonomous agent as it completes laps on the Gazebo simulator.

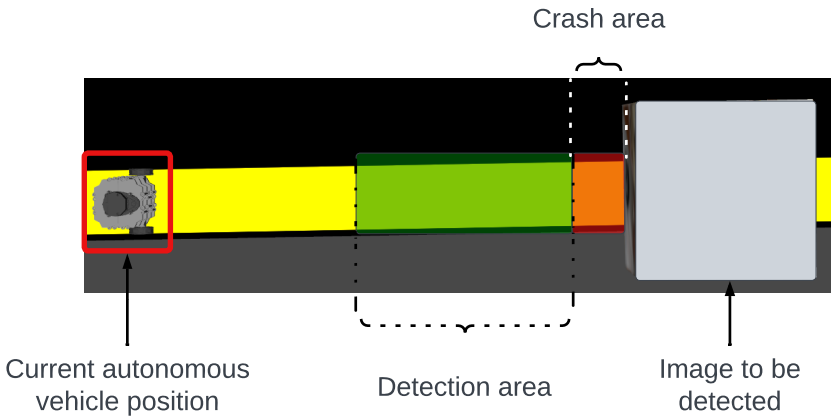


Figure 5.3: Detection and crash area for the autonomous vehicle. A detection is registered when the vehicle is in the green-shaded area, if the vehicle goes beyond this point and a detection was not registered (red-shaded area) a crash is recorded.

the simulation.

We selected 16 images for the classes mentioned above (bus, car, motorbike, dog and bicycle) from the test set of the PASCAL VOC 2007 [42] challenge. Our evaluation considers two different ODD regions: (i) motorway with the classes bus, car and motorbike; and (ii) town with the classes motorbike, person, dog, cat and bicycle. The 10 journeys of the autonomous vehicle were randomly assembled with 6 laps each, where each lap could contain either motorway or town classes.

Our system is provided with three synthesised ensembles one specific ensemble for each ODD and a base ensemble which considers all of the concerns for both ODDs simultaneously. We also assessed the best performing risk-aware and risk-oblivious models using an acceptable risk level of $\tau = H(3)$. The base ensemble is designed to be applied to both of the different ODD regions defined i.e. motorway and town. It attempts to mitigate five risk concerns: (i) motorbike predicted as car; (ii) bicycle predicted as person; (iii) dog predicted as cat; (iv) motorbike predicted as person; and (v) bus predicted as car. The second ensemble is specific to the motorway ODD and includes only two concerns: (i) motorbike predicted as car; and (ii) bus

Table 5.1: LoE (that corresponds to the actual class) and impact levels (that corresponds to the class pairs) for the three synthesised ensembles. Observe how the values vary from one ODD to the other.

Id	Concern		Base Ensemble		Motorway ensemble		Town ensemble	
	Actual	Predicted	LoE	Impact	LoE	Impact	LoE	Impact
1	Motorbike(13)	Car(6)	VH	VH	VH	VH	H	VL
2	Bicycle(1)	Person(14)	VH	VH	VL	H	VH	VH
3	Dog(11)	Cat(7)	VH	VH	VL	H	VH	VH
4	Motorbike(13)	Person(14)	VH	VH	L	VL	VH	VH
5	Bus(5)	Car(6)	VH	VH	VH	VH	VH	L

predicted as car. Finally, the third ensemble was synthesised for a town ODD and considers three concerns: (i) bicycle predicted as person; (ii) dog predicted as cat; and (iii) motorbike predicted as person. The single risk-aware and risk-oblivious models are reused from Chapter 4; however, the ensembles for the specific ODDs are newly created.

In Table 5.1 we show the *LoE* values for each of the classes across the different ensembles as well as the *impact* levels. The Table reflects that from one ODD to another classes are expected to be encountered more or less often, for instance the class bicycle has a very-high likelihood of encounter in the ODD town; whereas, in the ODD motorway its LoE is very-low as we normally do not expect to find bicycles in a motorway. Similarly, the class dog has a very-low *LoE* in the motorway ODD but it is very likely to be found in town. The impact of misclassifications also varies, in our example, misclassifying a bus as a car in a motorway has a very-high impact as we believe that if an assumption is made based on the size of the vehicle in front to plan an action such as overtaking, and the wrong size is predicted this could cause an accident; however, in town the misclassification has a low impact as we assume the vehicle is traveling at low speed and overtaking is a less dangerous maneuver. The values that we chose could be arguable different, in fact, in a real-world scenario expert domain input is required; however, the values we propose help to create scenarios with different concerns to evaluate our approach.

Finally, we provide some technical details of the simulation. It has been built in the popular simulator Gazebo [74] version 9.0, [2] a 3D dynamic multi-robot environment capable of recreating complex worlds. Inside the Gazebo world we deployed a TurtleBot3 [3] as our autonomous vehicle running the Robot Operating System (ROS) [4] melodic equipped with a camera and running an implementation of YoloV3 in TensorFlow 2.0 [5]. This implementation was modified accordingly to allow the use of our risk-aware real-time object detection method described in Chapter 4 and to bridge the communication between the TurtleBot and the YoloV3 [119] object detector.

[2] `https://gazebosim.org`
[3] `https://www.turtlebot.com/`
[4] `https://www.ros.org/`
[5] `https://github.com/zzh8829/yolov3-tf2`

Table 5.2: Risk values before and after the synthesised base, motorway and town ensembles for the relevant concerns in each of them. The concerns are the class pairs with risk values above $\tau = H(3)$ and the asterisk in ensembles motorway and town indicates that the class pairs did not require mitigation as their initial risk value was below τ.

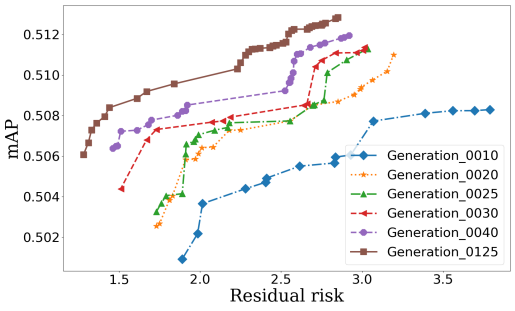
Id	Concern		Base ensemble		Motorway ensemble		Town ensemble	
	Actual	Predicted	Initial risk	Mit risk	Initial risk	Mit risk	Initial risk	Mit risk
1	Motorbike(13)	Car(6)	3.01	1.36	3.01	1.4	1.67	*
2	Bicycle(1)	Person(14)	3.12	1.8	1.04	*	3.12	1.81
3	Dog(11)	Cat(7)	3.2	2.8	1.06	*	3.2	2.11
4	Motorbike(13)	Person(14)	4.31	3.97	1.65	*	4.31	3.28
5	Bus(5)	Car(6)	4.68	3.11	4.68	3.89	1.26	*
Residual risk			3.32	1.08	1.69	0.89	1.63	0.28

5.3.2 Experimental results

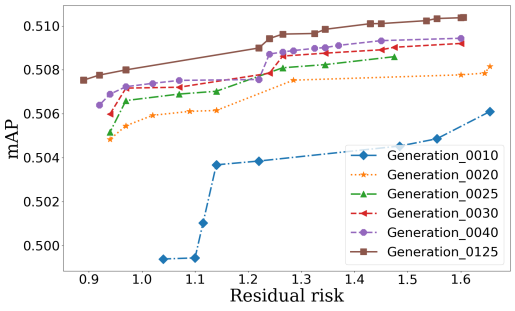
Table 5.2 shows the initial risk values for the identified concerns. We observe that the risk varies for the concerns in the different ensembles due to the provided risk information (LoE and impact values in Table 5.1) and its variation in each ODD, for instance, the class pair dog predicted as cat in the base and town ensembles has an initial risk of 3.2 which makes it a concern as this value is above $\tau = H(3)$; the same class pair in the motorway ensemble has a value of 1.06 which means it is not a concern in this ODD. Table 5.2 also shows the mitigated risk values for each of the concerns, i.e. the risk value obtained with the synthesised ensemble. We can see how the synthesised ensembles successfully decrease the risk values and this is reflected in the residual risk in the last row of the table. In the base ensemble the residual risk was reduced from 3.32 to 1.08, in the motorway ensemble the residual risk was reduced from 1.69 to 0.89 and in the town ensemble the residual risk was reduced from 1.63 to 0.28. The asterisks in motorway and town ensembles indicate that no mitigation was required for those class pairs as their initial risk values were below τ and did not, therefore, require mitigation.

We show the ensemble learning history of the three synthesised ensembles in Figure 5.4. Training for the three ensembles ran for 2500 generations of the GA and we can observe how as generations elapsed the GA found better solutions with a reduction in the amount of risk and improved mAP. We note that the residual risk in the base ensemble, shown in Figure 5.4a, was reduced to 1.08 and is the ensemble with the highest residual risk when compared to the other two. Because the base ensemble is dealing with five concerns at the same time, the GA struggles to find solutions that mitigate the risk for all concerns below τ. This is expected since more concerns mean that the GA starts with a larger residual risk. Nevertheless, the ensemble successfully mitigated the risk to some extent for all concerns.

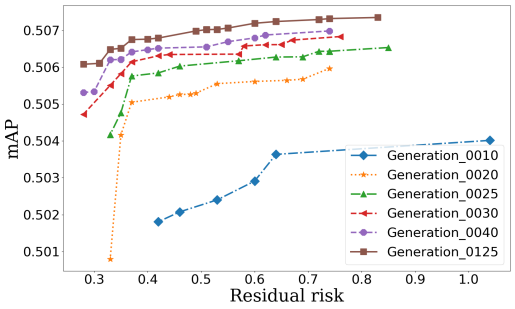
We allowed each ensemble to make use of 4 models and when we investigated model choice we identified the following: For the base ensemble the GA chose three risk-aware models for the concerns (i) bicycle as person, (ii) dog as cat and (iii) bus as car, and one risk-oblivious model. For the motorway ensemble the GA selected



(a) Ensemble learning base



(b) Ensemble learning motorway



(c) Ensemble learning town

Figure 5.4: Ensemble learning history of the three synthesised ensembles for the evaluation of the proposed approach.

two risk-aware models for the concerns (i) motorbike as car and (ii) bus as car and two risk-oblivious models. Finally, for the town ensemble three risk-aware models were selected (i) bicycle as person, (ii) dog as cat and (iii) motorbike as person and one risk-oblivious model. This suggests that the GA was able to identify and use the risk-aware models created for the specific concerns and use them to mitigate the risk. For the ODDs motorway and town the GA was able to fit one risk-aware model per concern as well as the risk-oblivious models; however, for the base ensemble, only three concerns got a risk-aware model and because the risk-aware models are specialist in one type of misclassification, they normally do not perform very well for the rest of the class pairs, that is why the GA, selected two risk-oblivious models, to account for those cases, leaving two concerns without specialist, and we believe this contributes to how much the residual risk can be minimised.

The synthesised ensembles were deployed into the simulation described in Section 5.3.1 and the number of crashes was registered for each of them as well as the number of times the correct action was taken during the 10 journeys of the autonomous vehicle. We also ran an experiment dynamically switching between models as the autonomous vehicle entered to a new ODD, all of the results are shown in Table 5.3. As we can observe, dynamically switching between the ensembles reduces the mean number of crashes compared to using a single ensemble or single model with the mean reducing from a maximum of 16.30 for a single model to 12.60 when using adaptation.

When we consider the systems ability to execute the correct action we found that for the *fast* action adaptation is better than using the motorway, town or

Table 5.3: Mean values of crashes, safe action fast and safe action slow during the 10 journeys of the autonomous vehicle when dynamically switching between models (adaptation), using only the motorway ensemble, the town ensemble, the base ensemble and the best performing risk-oblivious single model.

Model	Mean crashes	Mean safe action fast	Mean safe action slow
Adaptation	12.60	9.20	14.30
Motorway ensemble	14.90	8.20	14.30
Town ensemble	15.70	8.90	12.70
Base ensemble	15.30	8.60	13.80
Single model	16.30	10.10	10.90

base ensembles with a mean of 9.20 safe actions per journey vs 8.20, 8.90 and 8.60 when adaptation was not used. Surprisingly, single model has higher precision than adaptation. We believe this is due in part to the challenging environment in which the vehicle is operating, i.e. the image is shrunk to fit on the cubes; the lighting conditions are not as one finds in the training set and; the angle of view and speed of the vehicle distorts the image. This means that some of the models in the ensemble are adversely affected and the performance of the ensemble suffers. The single model is less prone to this error and is not being combined with other models.

For the safe action *slow* we found that both the adaptation and motorway ensembles have the highest value per journey. Across the range of conditions we see then that switching ensembles for each ODD is better as we have reduced the number of crashes and improved the safe actions *fast* and *slow* when compared to using only one ensemble for the two different ODD regions.

The benefit of model switching is also shown in Figure 5.5 which compares the number of crashes of the autonomous vehicle when switching models with the motorway, town and base ensemble as well as the best performing risk-oblivious single model for all journeys considered. We can see that using our approach lowers the variance and mean of the number of crashes recorded and we also obtained the minimum value recorded in one journey, We also note that the motorway ensemble is better than the town and base ensembles, however, the motorway ensemble has a higher variance. We can also see that town in one run has the worst performance of all models (outlier) with 24 crashes and has no benefit over the base ensemble. Lastly, single model has the worst performance when compared to our approach and the rest of the ensembles. These results confirm that by using our approach the safety of the vehicle has been improved in a challenging environment.

5.3.3 Discussion

The evaluation of our approach shows that by dynamically switching ensembles as the autonomous vehicle enters a new ODD region has the benefit of improving the safety of the vehicle by decreasing the number of crashes and by choosing the correct action based on the detection. However, we still observe how crashes and

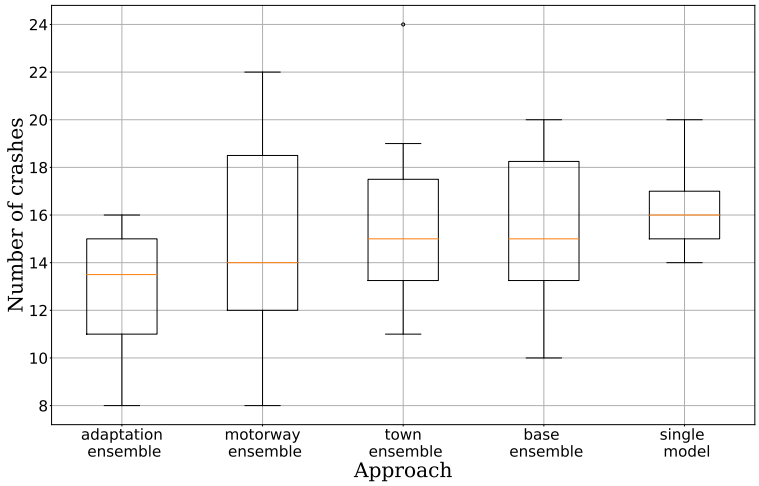


Figure 5.5: Comparison of number of crashes when using adaptation, motorway ensemble, base ensemble and single model for the 10 journeys of the autonomous vehicle in the gazebo simulation.

misclassifications continue to happen which is not a surprise as we are deploying the models in a challenging context with environmental disturbances that come naturally from the lighting conditions of the simulation such as brighter/darker areas and shadows created for the angle in which the light sources are displayed. The size of the images also makes object detection more challenging as we are mapping the images onto cubes which deforms them by changing their original size. Similarly, what the camera perceives, and sends to the ensembles, is not only the image on the cube itself but also noise such as bits of the road and other objects in the simulation. The fact that the robot is moving around the circuit-track also has an effect on the angle of perception of the image which leads to misclassifications and missed objects affecting the safety of the vehicle. Because of all mentioned disturbances we found that even the same ensemble would some times correctly classify the images and in a different lap either completely miss it or misclassify it. Despite the challenges discussed we emphasise that our approach effectively improves the safety of the autonomous vehicle when compared to a solution that does not considers dynamic ensemble switching.

5.3.4 Threats to Validity

Construct validity. These threats are associated with the assumptions made during the implementation of the simulator used to evaluate the proposed approach. Specifically, they may be caused by a potential discrepancy between the simulator and the real-world environment, as well as by the accuracy of the synthesised ensembles in representing the intended ODD regions. To address these threats, we conducted our experiments in the Gazebo simulator, which is widely recognised as the best suited platform for simulating autonomous vehicles performing tasks.

Gazebo offers several key features that mitigate these concerns. Firstly, it provides a realistic environment, allowing for the creation of immersive virtual scenarios that closely resemble real-world conditions. Secondly, Gazebo supports the integra-

tion and simulation of various sensors commonly used in autonomous vehicles, such as cameras, lidars, and radars. This capability enables realistic perception modeling, allowing autonomous vehicles to accurately sense and interpret their virtual surroundings. Finally, Gazebo incorporates robust physics engines that accurately simulate vehicle dynamics and control systems. It takes into account important factors like inertia, friction, suspension, and aerodynamics, resulting in realistic vehicle behavior and response to control inputs. This feature enables accurate testing and tuning of autonomous vehicle control algorithms, enhancing the reliability and performance of the proposed approach.

To ensure the accuracy of the synthesised ensembles in representing the ODD regions, we carefully selected the relevant classes for each ODD. We derived the risk information values by considering the frequency of class instances within each ODD, the impact of misclassifications in different ODDs, and how these values vary across different ODDs. We drew inspiration from the Road Accidents and Safety Statistics reports [6] from the UK to guide our determination of these values.

Internal validity. These threats can originate from the potential of obtaining inaccurate results and the introduction of bias during data collection and analysis. To mitigate them, we conducted experiments encompassing a diverse set of scenarios. These scenarios varied in terms of the number of deployed images at random locations, the quantity of synthesised ensembles, the range of mitigated concerns, and the number of GA generations utilised.

Furthermore, to enhance the reliability and robustness of our findings, the reported results were gathered over multiple independent journeys of the autonomous vehicle. This approach helped ensure that the outcomes were not influenced by specific instances or biased data points, thus reducing the risk of skewed or unreliable results.

External validity Threats emerge from the challenges associated with applying the proposed evaluation methodology to RTOD ensembles synthesised from data sets other than the one used in our case study. To mitigate this threat, we introduced the ensemble synthesis as a separate and independent component of the methodology. We recommend deriving ensembles using balanced data sets and incorporating input from domain experts to provide accurate risk information.

Additionally, we propose the utilisation of a reusable simulator that enables the performance evaluation of the methodology to be decoupled from the specific case study at hand. Such a simulator offers the flexibility to deploy customised objects, and allows users to load their own RTOD algorithms into the autonomous vehicle, enhancing the versatility and adaptability of the evaluation process.

By separating the ensemble synthesis and leveraging a reusable simulator, we aimed to mitigate the threats associated with the generalisability and applicability of our framework. We therefore envisage that the framework can be applied to diverse data sets and scenarios, fostering broader adoption and facilitating the assessment of the proposed approach in different contexts. Nevertheless, the evaluation of the framework through additional case studies is needed to confirm this hypothesis.

[6] For more details please refer to `https://www.gov.uk/government/collections/road-accidents-and-safety-statistics`

5.4 Related work

Ensemble methods have effectively been used to improve the accuracy of single-DNN models particularly for tasks such as self-driving vehicles [154], advanced driver assistance systems [159], vehicle detection [155], etc. However, the majority of these solutions do not consider different ODD regions and their corresponding risk mitigation strategies. Moreover, the existing solutions are not making the dynamic selection of ensembles as we do.

An efficient ensemble algorithm for object detection on a dataset for autonomous vehicles under adverse weather conditions is presented in [154]. In this work they propose assembling multiple baseline deep learning models under different voting strategies for object detection and the utilisation of data augmentation to boost the models' performance. The applied techniques demonstrated an increase in accuracy over the baseline models and were able to identify objects from the images captured in the adverse foggy and rainy weather conditions. Unlike our work they do not make an identification of different ODDs and do not apply online switching of ensembles.

A real-time fusion model approach based on the dynamic analysis of agreement among object detection outputs with data augmentation to enhance the model's accuracy is proposed in [159]. The approach is demonstrated in the context of cone, pedestrian and box detection for advanced driver assistance systems. Although this approach applies ensembles to improve the performance of the models, it does not support the synthesis of different ensemble models applied to different context.

Wang et al. introduce a weighted ensemble method called soft-weighted-average [155] . The proposed method is attenuated by object detection confidence, and it penalises the detection result of the corresponding relationship by the confidence attenuation. The proposed method can further reduce the vehicle misdetection of the target detection algorithm, obtaining a better detection result, similarly to the previous work. This method does not considers the synthesis of ensembles for different ODD regions and does not supports dynamic model selection.

DeepCert, a tool-supported method for verifying the robustness of deep neural network image classifiers to contextually relevant perturbations such as blur, haze, and changes in image contrast is presented in [108], their focus is on verifying DNN robustness to small perturbations in the images being classified. DeepCert addresses the verification problem by supporting (i) the encoding of real-world image perturbations; (ii) the systematic evaluation of contextually relevant DNN robustness; (iii) the generation of contextually relevant counterexamples; and (iv) the selection of DNN image classifiers suitable for the operational context during design or when the system is deployed. As we can see, the similarity with our approach is that this method derives DNN models for operational contexts or different ODDs; nevertheless, their approach only makes a recommendation of which model to use but they do not apply dynamic selection of the models as we do, another difference is that they work with single models applied to image detection and our work focuses on risk-aware ensembles applied to object detection.

5.5 Summary

In this chapter we presented an approach that advocates the use of a collection of risk-aware ML ensembles with dynamic switching between models as a RAS moves

from one ODD region to another. We described the five components of the approach and we then moved to its evaluation using a simulator in which a vehicle moves round a circuit in which context dependent objects are present. The vehicle used RTOD ensembles that we synthesised with details of risk information and risk values for the concerns that each of the models were designed to mitigate. We assessed the effectiveness of ensemble switching by measuring safety as the number of crashes and number of correct actions taken. The obtained results suggest that by using our approach the safety of the vehicle has been improved in a challenging environment.

Chapter 6

Conclusion and Further Work

6.1 Conclusions

This thesis highlighted the need to systematically assess and mitigate the risk introduced by deep neural networks (DNN) in safety critical applications. In particular, the loss function minimised by the training of DNN classifiers, and the assessment of their performance, are oblivious of the risks associated with the intended use of such classifiers. Traditionally, DNNs place equal emphasis on the detection accuracy of classes, disregarding any context-relevant misclassifications. We propose an approach in which we identify class-pairs whose misclassifications are of concern from a safety perspective. We achieve concern identification through the derivation of a risk profile for each misclassification that combines the likelihood of misclassification with risk information such as impact and likelihood of encounter where each risk factor is a result of the context in which the system will be deployed.

The approaches introduced in this work consider DNNs applied to image classification as well as real-time object detection. We address limitations of existing solutions by considering risk factors to identify class pairs with risk values over an acceptable threshold, which we term concerns. Concerns are ranked in order to identify those which require immediate treatment and those which can be tolerated. Once the concerns are identified we propose an approach to mitigate risk concerns by modifying the loss function and obtaining a set of specialist models. These models are then combined with traditional, generalist models to create risk-aware ensembles whose training is guided by a GA that seeks to minimise risk and to maximise performance metrics. The evaluation of the proposed approaches shows an improvement in terms of both the risk minimisation and the performance metrics.

This thesis also explored the possibility of using run-rime adaptation in which risk-aware ensembles are dynamically switched as the system moves from one Operational Design Domain (ODD) region to another. The simulation environment which we constructed allows us to evaluate the performance of risk-aware ensembles in a challenging context with environmental disturbances. The evaluation of our approach shows the potential for improvement in the safety of an autonomous vehicles by reducing the number of crashes and increasing the number of correct actions taken.

In the remainder of this chapter we summarise the main contributions of this thesis.

Firstly, in Chapter 3, we presented an approach for identifying and mitigating

risks associated with misclassifications to synthesize risk-aware DNN image classifiers for use in safety critical applications. The effectiveness of the approach was assessed through the application of two case studies, CIFAR-10 [77] and a subset of the GTSRB data set [141]. The results obtained for both data sets suggest that our method can can mitigate contextually relevant risk concerns using Pareto-optimal ensembles of risk-aware and risk-oblivious DNN image classifiers. We note the tendency for the residual risk to decrease and the F1 score to increase as the number of models allowed in the ensembles increased.

Secondly, in Chapter 4 we extended our approach to development risk-aware ML ensembles for real-time object detection. Since object detectors are concerned not only with image classification, but also defining the position of an object within a scene, this required a consideration of the bounding boxes in the identification of risk concerns. The validity of our method was evaluated using the PASCAL visual object classes challenge (VOC) 2007 dataset, a common used dataset in object detection tasks. The experimental results presented in this chapter suggest that our method can effectively mitigate risk, supporting the development of dependable RTOD-based systems for safety-critical applications.

Finally, we presented in Chapter 5 an approach that advocates the use of risk-aware ML ensembles with dynamic switching between models at run-time as the system moves from one ODD region to another. In order to assess the effectiveness of our proposed approach we built a simulator into which we deployed an autonomous vehicle using our method. Using this simulated environment we evaluated the system to measure the effectiveness of ensemble switching as the ODD region changed. We assessed the effectiveness of our approach by measuring the number of crashes and number of correct actions taken by the system when objects were detected. This evaluation shows that our approach has the benefit of improving the safety of the vehicle by decreasing the number of crashes and more frequently choosing the correct action based on the detection.

6.2 Directions for Future Work

The research presented in this thesis can be refined and extended in multiple directions as described below. Exploring these directions could improve the effectiveness of the methods that we introduced in this PhD project.

6.2.1 Mitigating Risk in Neural Network Classifiers

In the evaluation of this approach we have demonstrated that better image classifiers can be constructed by synthesising Pareto-optimal ensembles that mitigate risk and improve performance metrics. To further assess the generalisability of this approach we would like to examine other domains, for example health care, and apply the approach to a data set such as Chestx-ray8 [157]. If such direction is taken it will impose new challenges, for instance we would require a close collaboration with specialist medics to provide the risk information needed for stage two of the approach.

Whilst the experiments carried out so far indicate that increasing the number of models used in the ensemble increases system performance, it is unclear where the limits of such improvement lie. As the search space increases it may become

infeasible for the genetic algorithm approach to find suitable solutions. It is also unclear if such limits are context dependent. Further work is required to identify the limits of our approach and provide guidance on the construction of ensembles for different contexts.

Additionally, we propose as further work alternative training strategies. For instance, developing novel training strategies that target the mitigation of multiple identified risks. This can involve exploring techniques such as active learning [29, 128, 129], where the training process focuses on high-risk instances or incorporating cost-sensitive learning [39, 89] to assign different penalties for misclassifications based on their associated risks.

Additionally, as part of the directions we propose for future research, we recommend the optimisation of the process for synthesising risk-aware ensembles using multi-objective genetic algorithms. To achieve this, it is essential to explore new avenues that can improve the diversity and performance of the ensembles. One area of investigation involves delving into novel genetic operators that can introduce greater variation and exploration within the ensemble population. Additionally, studying alternative selection mechanisms can help strike a balance between exploiting high-performing models and exploring risk-mitigating solutions.

Finally, we recommend as future work the development of innovative ensemble generation approaches. By investigating new methods for creating ensembles, we can effectively leverage the power of multi-objective GAs to enhance their diversity and performance. These approaches may involve techniques such as ensemble pruning or ensemble weighting based on risk severity. Through these advancements, one can create risk-aware ensembles that succeed in both mitigating risks and maintaining high performance.

6.2.2 Risk-aware Real-time Object Detection

The experimental results presented in this chapter of the thesis suggest that our method can effectively mitigate risk, supporting the development of dependable RTOD-based systems for safety-critical applications.While the PASCAL VOC 2007 dataset has provided valuable insights into the effectiveness of our approach, future research should explore the use of additional and more diverse data sets for evaluation for instance it would be interesting to evaluate the Microsoft COCO [87] dataset which gathers images of complex everyday scenes containing common objects in their natural context or the nuScenes [18] dataset which contains images from the autonomous driving domain. This would help validate the generalisability of the approach across different object categories, environmental conditions, and detection scenarios.

Stage four of our method (ensemble synthesis) provides as part of the prediction a bounding box calculated as the average of all bounding boxes predicted by the models allowed in the ensemble.

To deal with bounding boxes, approaches such as weighted boxes fusion introduced in [140] or the approach introduced in [162] explore different ideas for bounding box selection such as weighting of boxes based on their confidence score and the use of an algorithm called non-maximum suppression to disregard low scoring boxes.

Combining these ideas with the ensemble structure that we propose could refine the method to minimise the error in the final bounding box provided by learning

which models in the ensemble are better at predicting boxes for different classes in the dataset.

The approach can also be extended in its *Risk-aware ensemble synthesis and verification* step by investigating advanced fusion techniques, such as ensemble averaging, weighted voting, or adaptive combination methods, which can lead to improved ensemble performance and risk mitigation. This is particularly crucial in the context of RTOD, where multiple objects with varying locations are present in images from video streams.

Real-time object detection systems operate in dynamic and evolving environments, where new objects or changes in object behavior can occur. Future work should focus on developing adaptive ensemble generation techniques that can dynamically incorporate new knowledge from base learners and adapt to changing conditions. This could involve incorporating online learning methods or incremental ensemble learning approaches to continuously update the ensemble's knowledge and adapt to evolving object detection challenges.

6.2.3 Dynamic Selection of Risk-aware Object Detection Ensembles

A possible way in which the approach can be refined is through increasing the number of ODDs in which the autonomous driving will be deployed, this requires the identification of risk information for the new ODDs and synthesising ensembles with the ability to handle risk for them. Different ODD regions have been proposed to study context perturbation and disturbances in [108, 154]. We can use some of the ideas presented in these research to get inspiration and refine the evaluation of our approach. Moving on this direction will also require the selection of the relevant classes from the dataset and map the images onto the cubes to be deployed during the simulation. Testing the approach in a variety of regions of operation will help to prove the generality of the approach.

Another direction of further research is combining the dynamic switching of ensembles with verification techniques such as model checking. For instance, we can use a Markov decision process (MDP) model to synthesise a controller that seeks to minimise the time to perform journeys of the self-driving vehicle while keeping the a cumulative incident costs below a certain value. We can get inspiration from work such as [48, 54, 98, 153, 158].

6.2.4 Further Research Directions

This thesis proposed two different methods to be applied to image classification and object detection integrating a risk analysis to guide the training and synthesis of risk-aware ensembles. At the core of our approach we modified the loss function of the deep neural network training algorithms underpinning the image classification and the object detection tasks. DNNs are only one type of machine learning approach for classification and further work is needed to extend our approach to other classification algorithms. In the future it would be interesting to see how risk-mitigation approaches could be applied to decision trees [71] or binary classification algorithms [78], and how they can be integrated into self-adaptive systems that use a combination of AI and control-theoretic components [19, 160].

At the moment we are only considering the camera sensor in our autonomous system, however, autonomous agents such as self-driving vehicles require a variety of sensors including ultrasonic sensors, radio detection and lidar as well as other vision sensors such as stereo cameras, thermal cameras and event-driven cameras [51]. Further research is required to understand how risk mitigation might be used when we have a combination of multiple sensors on the autonomous system such as those described above.

Appendix A

Chapter 3 - Supplementary Material

This appendix shows the DNN architectures used to train the risk-oblivious and risk-aware models on the CIFAR-10 dataset and on the subset of the GTSRB from Chapter 3.

A.1 DNN architectures used to train the models on the CIFAR-10 dataset

```
from __future__ import print_function
import keras
from keras.datasets import cifar10
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
import json
import os

batch_size = 32
num_classes = 10
epochs = 100
data_augmentation = False
num_predictions = 20
model_id = 0
save_dir = os.path.join(os.getcwd(), '/saved_models/Unweighted')

for model_id in range(30):
    model_name = 'unweighted_model_{}.h5'.format(model_id)

    # The data, split between train and test sets:
    (x_train, y_train), (x_test, y_test) = cifar10.load_data()

    # Convert class vectors to binary class matrices.
    y_train = keras.utils.to_categorical(y_train, num_classes)
    y_test = keras.utils.to_categorical(y_test, num_classes)

    model = Sequential()
    model.add(Conv2D(32, (3, 3), padding='same',
```

```
                    input_shape=x_train.shape[1:]))
    model.add(Activation('relu'))
    model.add(Conv2D(32, (3, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Conv2D(64, (3, 3), padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(64, (3, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Flatten())
    model.add(Dense(512))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes))
    model.add(Activation('softmax'))

    # initiate RMSprop optimizer
    # Let's train the model using Adam
    model.compile(loss='categorical_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])

    x_train = x_train.astype('float32')
    x_test = x_test.astype('float32')
    x_train /= 255
    x_test /= 255

    if not data_augmentation:
        print('Not using data augmentation.')
        model.fit(x_train, y_train,
                  batch_size=batch_size,
                  epochs=epochs,
                  validation_data=(x_test, y_test),
                  shuffle=True)
    else:
        print('Using real-time data augmentation.')
        # This will do preprocessing and realtime data augmentation
    :
        datagen = ImageDataGenerator(
            # set input mean to 0 over the dataset
            featurewise_center=False,
            # set each sample mean to 0
            samplewise_center=False,
            # divide inputs by std of the dataset
            featurewise_std_normalization=False,
            # divide each input by its std
            samplewise_std_normalization=False,
            # apply ZCA whitening
            zca_whitening=False,
            # epsilon for ZCA whitening
            zca_epsilon=1e-06,
            # randomly rotate (degrees, 0 to 180)
            rotation_range=0,
```

```
            # randomly shift images horizontally
            width_shift_range=0.1,
            # randomly shift images vertically
            height_shift_range=0.1,
            # set range for random shear
            shear_range=0.,
            # set range for random zoom
            zoom_range=0.,
            # set range for random channel shifts
            channel_shift_range=0.,
            # set mode for filling points
            #outside the input boundaries
            fill_mode='nearest',
            cval=0.,
            # randomly flip images
            horizontal_flip=True,
            vertical_flip=False,
            rescale=None,
            preprocessing_function=None,
            data_format=None,
            validation_split=0.0)

        datagen.fit(x_train)

        # Fit the model on the batches generated by datagen.flow().
        model.fit_generator(datagen.flow(x_train, y_train,
                            batch_size=batch_size),
                            epochs=epochs,
                            validation_data=(x_test, y_test),
                            workers=4)

    # Save model and weights
    if not os.path.isdir(save_dir):
        os.makedirs(save_dir)
    model_path = os.path.join(save_dir, model_name)
    model.save(model_path)
    print('Saved trained model at %s ' % model_path)

    # Score trained model.
    scores = model.evaluate(x_test, y_test, verbose=1)
    print('Test loss:', scores[0])

    # Now save the model architecture and weights
    json_string = model.to_json()

    architecture_name = 'model_architecture_{}.txt'.format(model_id
)
    save_path = os.path.join(save_dir, architecture_name)

    with open(save_path, 'w') as outfile:
        json.dump(json_string, outfile)

    weights_name = 'model_weights_{}.h5'.format(model_id)
    save_path = os.path.join(save_dir, weights_name)
    model.save_weights(save_path)
```

Listing A.1: DNN architecture used to train the risk-oblivious models on the CIFAR-10 dataset.

```
from __future__ import print_function
import keras
from keras.datasets import cifar10
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
import json
import os

import numpy as np
import tensorflow as tf
from keras import backend as K
import datetime


def class_weighted_loss(y_true, y_pred, **kwargs):
  w_flat = np.array(weights).flatten()
  WEIGHTS = tf.constant(w_flat, dtype=tf.float32)
  y_classT = K.argmax(y_true, axis=1)
  y_classP = K.argmax(y_pred, axis=1)

  idx = y_classT + 10*y_classP

  w = tf.gather(WEIGHTS, idx)
  loss = keras.losses.categorical_crossentropy(y_true, y_pred)
  result = loss * w
  return result

#set of concerns
data=[[0, 1, 2,'1as0_w2'],[0, 1, 5,'1as0_w5'],
     [0, 1, 10,'1as0_w10'],[1, 9, 2,'9as1_w2']...]


for j in range(0, len(data)):
  predicted=data[j][0]
  actual=data[j][1]
  weight=data[j][2]

  #create directory
  os.mkdir('/savedModels/weighted'+data[j][3])
  saveDirectory='/savedModels/weighted'+data[j][3]

  weights = np.ones((10,10))
  weights[predicted,actual] = weight

  f = weights.shape[0]*(weights.shape[0])
  weights /= np.sum(weights)/f

  save_dir = os.path.join(os.getcwd(), saveDirectory)

  batch_size = 32
  num_classes = 10
  epochs = 100
  data_augmentation = False
  num_predictions = 20


```

```
for i in range(10):
    model_id = i
    model_name = 'model_{}.h5'.format(model_id)

    # The data, split between train and test sets:
    (x_train, y_train), (x_test, y_test) = cifar10.load_data()

    # Convert class vectors to binary class matrices.
    y_train = keras.utils.to_categorical(y_train, num_classes)
    y_test = keras.utils.to_categorical(y_test, num_classes)

    model = Sequential()
    model.add(Conv2D(32, (3, 3), padding='same',
                     input_shape=x_train.shape[1:]))
    model.add(Activation('relu'))
    model.add(Conv2D(32, (3, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Conv2D(64, (3, 3), padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(64, (3, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Flatten())
    model.add(Dense(512))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes))
    model.add(Activation('softmax'))

    opt = keras.optimizers.RMSprop(learning_rate=0.0001, decay=1e
-6)

    model.compile(optimizer='adam',
                  loss=class_weighted_loss,
                  metrics=['accuracy'])

    x_train = x_train.astype('float32')
    x_test = x_test.astype('float32')
    x_train /= 255
    x_test /= 255

    if not data_augmentation:
        print('Not using data augmentation.')
        model.fit(x_train, y_train,
                  batch_size=batch_size,
                  epochs=epochs,
                  validation_data=(x_test, y_test),
                  shuffle=True)
    else:
        print('Using real-time data augmentation.')
        # This will do preprocessing
        #and realtime data augmentation:
        datagen = ImageDataGenerator(
```

```
            # set input mean to 0 over the dataset
            featurewise_center=False,
             # set each sample mean to 0
            samplewise_center=False,
            # divide inputs by std of the dataset
            featurewise_std_normalization=False,
            # divide each input by its std
            samplewise_std_normalization=False,
            # apply ZCA whitening
            zca_whitening=False,
            # epsilon for ZCA whitening
            zca_epsilon=1e-06,
            # randomly rotate (degrees, 0 to 180)
            rotation_range=0,
            # randomly shift images horizontally
            width_shift_range=0.1,
            # randomly shift images vertically
            height_shift_range=0.1,
            # set range for random shear
            shear_range=0.,
            # set range for random zoom
            zoom_range=0.,
             # set range for random channel shifts
            channel_shift_range=0.,
            # set mode for filling points
            # outside the input boundaries
            fill_mode='nearest',
            cval=0.,
            # randomly flip images
            horizontal_flip=True,
            vertical_flip=False,
            rescale=None,
            preprocessing_function=None,
            data_format=None,
            validation_split=0.0)

        datagen.fit(x_train)

        # Fit the model on the batches
        #generated by datagen.flow().
        model.fit_generator(datagen.flow(x_train, y_train,
                                         batch_size=batch_size),
                                         epochs=epochs,
                                         validation_data=(x_test, y_test),
                                         workers=4)

    # Save model and weights
    if not os.path.isdir(save_dir):
        os.makedirs(save_dir)
    model_path = os.path.join(save_dir, model_name)
    model.save(model_path)
    print('Saved trained model at %s ' % model_path)

    # Score trained model.
    scores = model.evaluate(x_test, y_test, verbose=1)
    print('Test loss:', scores[0])

    # Now save the model architecture and weights
```

```
174         json_string = model.to_json()
175 
176         architecture_name = 'model_architecture_{}.txt'.format(
    model_id)
177         save_path = os.path.join(save_dir, architecture_name)
178 
179         with open(save_path, 'w') as outfile:
180             json.dump(json_string, outfile)
181 
182         weights_name = 'model_weights_{}.h5'.format(model_id)
183         save_path = os.path.join(save_dir, weights_name)
184         model.save_weights(save_path)
```

Listing A.2: DNN architecture used to train the risk-aware models on the CIFAR-10 dataset.

A.2 DNN architectures used to train the models on the subset of the GTSRB dataset

```
1  import pickle
2  import pandas as pd
3  import numpy as np
4  from keras.models import Sequential
5  from keras.optimizers import Adam
6  from keras.layers import Dense
7  from keras.layers import Flatten, Dropout
8  from keras.utils.np_utils import to_categorical
9  from keras.layers.convolutional import Conv2D, MaxPooling2D
10 from keras.preprocessing.image import ImageDataGenerator
11 import cv2
12 import os
13 import json
14 import sys
15 import random
16 
17 #set size to maximum to avoid trucation
18 np.set_printoptions(threshold=sys.maxsize)
19 
20 #Load pickle subset data
21 base='/SubsetOfTrafficSign/german-trafficSubSetData/'
22 #Train data
23 with open(base+'y_train.pkl', 'rb') as input:
24     y_train = pickle.load(input)
25 with open(base+'X_train.pkl', 'rb') as input:
26     X_train = pickle.load(input)
27 
28 #Test data
29 with open(base+'y_test.pkl', 'rb') as input:
30     y_test = pickle.load(input)
31 with open(base+'X_test.pkl', 'rb') as input:
32     X_test = pickle.load(input)
33 
34 #Validation data
35 with open(base+'y_val.pkl', 'rb') as input:
36     y_val = pickle.load(input)
37 with open(base+'X_val.pkl', 'rb') as input:
38     X_val = pickle.load(input)
```

```
39
40
41 datagen = ImageDataGenerator(width_shift_range=0.1,
42                               height_shift_range=0.1,
43                               zoom_range=0.2,
44                               shear_range=0.1,
45                               rotation_range=10.)
46
47 datagen.fit(X_train)
48
49 batches = datagen.flow(X_train, y_train, batch_size=15)
50 X_batch, y_batch = next(batches)
51
52 y_train = to_categorical(y_train)
53 y_test = to_categorical(y_test)
54 y_val = to_categorical(y_val)
55
56
57 # create risk-oblivious models from subset
58 save_dir = os.path.join(os.getcwd(), '/SubsetOfTrafficSign/
       saved_models/unweighted')
59 def make_model():
60     model = Sequential()
61     model.add(Conv2D(60, (5, 5), input_shape=(32, 32, 1),
       activation='relu'))
62     model.add(Conv2D(60, (5, 5), activation='relu'))
63     model.add(MaxPooling2D(pool_size=(2, 2)))
64
65     model.add(Conv2D(30, (3, 3), activation='relu'))
66     model.add(Conv2D(30, (3, 3), activation='relu'))
67     model.add(MaxPooling2D(pool_size=(2, 2)))
68
69     model.add(Flatten())
70     model.add(Dense(500, activation='relu'))
71     model.add(Dropout(0.5))
72     #this int is the no of classes
73     model.add(Dense(8, activation='softmax'))
74
75     model.compile(Adam(lr=0.001),
76                   loss='categorical_crossentropy',
77                   metrics=['accuracy'])
78
79     return model
80
81
82 for model_id in range(30):
83     model = make_model()
84     print(model.summary())
85
86     history = model.fit_generator(datagen.flow(X_train, y_train,
87                           batch_size=50),
88                           steps_per_epoch=X_train.shape[0]/50,
89                           epochs=100,
90                           validation_data=(X_val, y_val), shuffle=1)
91
92     # Now save the model architecture and weights
93     json_string = model.to_json()
94
```

```
    architecture_name = 'model_architecture_{}.txt'.format(model_id)
    save_path = os.path.join(save_dir, architecture_name)

    with open(save_path, 'w') as outfile:
        json.dump(json_string, outfile)

    weights_name = 'model_weights_{}.h5'.format(model_id)
    save_path = os.path.join(save_dir, weights_name)
    model.save_weights(save_path)
```

Listing A.3: DNN architecture used to train the risk-oblivious models on the subset of the GTSRB dataset.

```
import numpy as np
import keras
import tensorflow as tf
from keras import backend as K
from keras.optimizers import Adam
from keras.layers import Dense
from keras.layers import Flatten, Dropout
from keras.utils.np_utils import to_categorical
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.models import Sequential
import utils.library as lib
import pickle
import pandas as pd
from keras.preprocessing.image import ImageDataGenerator
import os
import json

def class_weighted_loss(y_true, y_pred, **kwargs):
  w_flat = np.array(weights).flatten()
  WEIGHTS = tf.constant(w_flat, dtype=tf.float32)
  y_classT = K.argmax(y_true, axis=1)
  y_classP = K.argmax(y_pred, axis=1)

  idx = y_classT + 8*y_classP

  w = tf.gather(WEIGHTS, idx)
  loss = keras.losses.categorical_crossentropy(y_true, y_pred)
  result = loss * w
  return result


# create model
def make_model():
    model = Sequential()
    model.add(Conv2D(60, (5, 5), input_shape=(32, 32, 1), activation='relu'))
    model.add(Conv2D(60, (5, 5), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(30, (3, 3), activation='relu'))
    model.add(Conv2D(30, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Flatten())
```

```
    model.add(Dense(500, activation='relu'))
    model.add(Dropout(0.5))
    #int no of classes
    model.add(Dense(8, activation='softmax'))

    model.compile(Adam(lr=0.001), loss=class_weighted_loss, metrics
   =['accuracy'])
    return model

#load the data of the subset traffic sign
base='/home/misael/Documents/riskAware/SubsetOfTrafficSign/german-
   trafficSubSetData/'
#Train data
with open(base+'y_train.pkl', 'rb') as input:
    y_train = pickle.load(input)
with open(base+'X_train.pkl', 'rb') as input:
    X_train = pickle.load(input)

#Test data
with open(base+'y_test.pkl', 'rb') as input:
    y_test = pickle.load(input)
with open(base+'X_test.pkl', 'rb') as input:
    X_test = pickle.load(input)

#Validation data
with open(base+'y_val.pkl', 'rb') as input:
    y_val = pickle.load(input)
with open(base+'X_val.pkl', 'rb') as input:
    X_val = pickle.load(input)

num_of_samples = []
cols = 5
num_classes = 8


datagen = ImageDataGenerator(width_shift_range=0.1,
                             height_shift_range=0.1,
                             zoom_range=0.2,
                             shear_range=0.1,
                             rotation_range=10.)

datagen.fit(X_train)

batches = datagen.flow(X_train, y_train, batch_size=15)
X_batch, y_batch = next(batches)

y_train = to_categorical(y_train,8)
y_test = to_categorical(y_test,8)
y_val = to_categorical(y_val,8)


################################config parameters and files
noOfEpochs=100
################################end config param and files

#concerns
data=[[4, 2, 2, '2as4_w2'],[4, 2, 5, '2as4_w5'],[4, 2, 10, '2
   as4_w10'],
```

```
99        [0, 4, 2, '4as0_w2'],[0, 4, 5, '4as0_w5'],[0, 4, 10, '4
    as0_w10'],
100       [7, 6, 2, '6as7_w2'],[7, 6, 5, '6as7_w5'],[7, 6, 10, '6
    as7_w10'],
101       [1, 4, 2, '4as1_w2'],[1, 4, 5, '4as1_w5'],[1, 4, 10, '4
    as1_w10']]
102
103
104 for i in range(0, len(data)):
105   print("starting model--------",i)
106   predictedClass=data[i][0]
107   actualClass=data[i][1]
108   weight=data[i][2]
109   os.mkdir('/SubsetOfTrafficSign/saved_models/weightedModels/
      weighted'+data[i][3])
110   saveDirectory='/SubsetOfTrafficSign/saved_models/weightedModels/
      weighted'+data[i][3]
111
112   weights = np.ones((8,8))
113   weights[predictedClass,actualClass] = weight
114
115   f = weights.shape[0]*(weights.shape[0])
116   weights /= np.sum(weights)/f
117
118   save_dir = os.path.join(os.getcwd(), saveDirectory)
119   model_id = 0
120
121   for model_id in range(10):
122       model = make_model()
123
124       history = model.fit_generator(datagen.flow(X_train, y_train,
125                     batch_size=50),
126                     #total elements of the training set/batch_size
127                     steps_per_epoch=X_train.shape[0]/50,
128                     epochs=noOfEpochs,
129                     validation_data=(X_val, y_val), shuffle=1)
130
131       # Now save the model architecture and weights
132       json_string = model.to_json()
133
134       architecture_name = 'model_architecture_{}.txt'.format(
    model_id)
135       save_path = os.path.join(save_dir, architecture_name)
136
137       with open(save_path, 'w') as outfile:
138           json.dump(json_string, outfile)
139
140       weights_name = 'model_weights_{}.h5'.format(model_id)
141       save_path = os.path.join(save_dir, weights_name)
142       model.save_weights(save_path)
```

Listing A.4: DNN architecture used to train the risk-aware models on the subset of the GTSRB dataset.

Appendix B

Chapter 4 - Supplementary Material

This appendix shows the YoloV3 architecture used to train the risk-aware and risk-oblivious models from Chapter 4. This implementation of YoloV3 in TensorFlow 2.0 is suggested in [112] and the code is publicly available at `https://github.com/zzh8829/yolov3-tf2`. Then we show the loss function of YoloV3 used to synthesise the risk-oblivious and the modified loss function used to train the risk-aware models.

```
from absl import logging
import numpy as np
import tensorflow as tf
import cv2

YOLOV3_LAYER_LIST = [
    'yolo_darknet',
    'yolo_conv_0',
    'yolo_output_0',
    'yolo_conv_1',
    'yolo_output_1',
    'yolo_conv_2',
    'yolo_output_2',
]

YOLOV3_TINY_LAYER_LIST = [
    'yolo_darknet',
    'yolo_conv_0',
    'yolo_output_0',
    'yolo_conv_1',
    'yolo_output_1',
]


def load_darknet_weights(model, weights_file, tiny=False):
    wf = open(weights_file, 'rb')
    major, minor, revision, seen, _ = np.fromfile(wf, dtype=np.
    int32, count=5)

    if tiny:
        layers = YOLOV3_TINY_LAYER_LIST
    else:
        layers = YOLOV3_LAYER_LIST

```

```
    for layer_name in layers:
        sub_model = model.get_layer(layer_name)
        for i, layer in enumerate(sub_model.layers):
            if not layer.name.startswith('conv2d'):
                continue
            batch_norm = None
            if i + 1 < len(sub_model.layers) and \
                    sub_model.layers[i + 1].name.startswith('batch_norm'):
                batch_norm = sub_model.layers[i + 1]

            logging.info("{}/{} {}".format(
                sub_model.name, layer.name, 'bn' if batch_norm else 'bias'))

            filters = layer.filters
            size = layer.kernel_size[0]
            in_dim = layer.get_input_shape_at(0)[-1]

            if batch_norm is None:
                conv_bias = np.fromfile(wf, dtype=np.float32, count=filters)
            else:
                # darknet [beta, gamma, mean, variance]
                bn_weights = np.fromfile(
                    wf, dtype=np.float32, count=4 * filters)
                # tf [gamma, beta, mean, variance]
                bn_weights = bn_weights.reshape((4, filters))[[1, 0, 2, 3]]

            # darknet shape (out_dim, in_dim, height, width)
            conv_shape = (filters, in_dim, size, size)
            conv_weights = np.fromfile(
                wf, dtype=np.float32, count=np.product(conv_shape))
            # tf shape (height, width, in_dim, out_dim)
            conv_weights = conv_weights.reshape(
                conv_shape).transpose([2, 3, 1, 0])

            if batch_norm is None:
                layer.set_weights([conv_weights, conv_bias])
            else:
                layer.set_weights([conv_weights])
                batch_norm.set_weights(bn_weights)

    assert len(wf.read()) == 0, 'failed to read all data'
    wf.close()


def broadcast_iou(box_1, box_2):
    # box_1: (..., (x1, y1, x2, y2))
    # box_2: (N, (x1, y1, x2, y2))

    # broadcast boxes
    box_1 = tf.expand_dims(box_1, -2)
    box_2 = tf.expand_dims(box_2, 0)
    # new_shape: (..., N, (x1, y1, x2, y2))
    new_shape = tf.broadcast_dynamic_shape(tf.shape(box_1), tf.shape(box_2))
```

```
    box_1 = tf.broadcast_to(box_1, new_shape)
    box_2 = tf.broadcast_to(box_2, new_shape)

    int_w = tf.maximum(tf.minimum(box_1[..., 2], box_2[..., 2]) -
                       tf.maximum(box_1[..., 0], box_2[..., 0]), 0)
    int_h = tf.maximum(tf.minimum(box_1[..., 3], box_2[..., 3]) -
                       tf.maximum(box_1[..., 1], box_2[..., 1]), 0)
    int_area = int_w * int_h
    box_1_area = (box_1[..., 2] - box_1[..., 0]) * \
        (box_1[..., 3] - box_1[..., 1])
    box_2_area = (box_2[..., 2] - box_2[..., 0]) * \
        (box_2[..., 3] - box_2[..., 1])
    return int_area / (box_1_area + box_2_area - int_area)


def draw_outputs(img, outputs, class_names):
    boxes, objectness, classes, nums = outputs
    boxes, objectness, classes, nums = boxes[0], objectness[0],
    classes[0], nums[0]
    wh = np.flip(img.shape[0:2])
    for i in range(nums):
        x1y1 = tuple((np.array(boxes[i][0:2]) * wh).astype(np.int32
    ))
        x2y2 = tuple((np.array(boxes[i][2:4]) * wh).astype(np.int32
    ))
        img = cv2.rectangle(img, x1y1, x2y2, (255, 0, 0), 2)
        img = cv2.putText(img, '{} {:.4f}'.format(
            class_names[int(classes[i])], objectness[i]),
            x1y1, cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 0, 255),
    2)
    return img


def draw_labels(x, y, class_names):
    img = x.numpy()
    boxes, classes = tf.split(y, (4, 1), axis=-1)
    classes = classes[..., 0]
    wh = np.flip(img.shape[0:2])
    for i in range(len(boxes)):
        x1y1 = tuple((np.array(boxes[i][0:2]) * wh).astype(np.int32
    ))
        x2y2 = tuple((np.array(boxes[i][2:4]) * wh).astype(np.int32
    ))
        img = cv2.rectangle(img, x1y1, x2y2, (255, 0, 0), 2)
        img = cv2.putText(img, class_names[classes[i]],
                          x1y1, cv2.FONT_HERSHEY_COMPLEX_SMALL,
                          1, (0, 0, 255), 2)
    return img


def freeze_all(model, frozen=True):
    model.trainable = not frozen
    if isinstance(model, tf.keras.Model):
        for l in model.layers:
            freeze_all(l, frozen)
```

Listing B.1: YoloV3 architecture used to train the risk-aware and risk-oblivious models from Chapter 4.

```
def YoloLoss(anchors, classes=80, ignore_thresh=0.5):
    def yolo_loss(y_true, y_pred):
        # 1. transform all pred outputs
        # y_pred: (batch_size, grid, grid, anchors, (x, y, w, h, obj, ...cls))
        pred_box, pred_obj, pred_class, pred_xywh = yolo_boxes(
            y_pred, anchors, classes)
        pred_xy = pred_xywh[..., 0:2]
        pred_wh = pred_xywh[..., 2:4]

        # 2. transform all true outputs
        # y_true: (batch_size, grid, grid, anchors, (x1, y1, x2, y2, obj, cls))
        true_box, true_obj, true_class_idx = tf.split(
            y_true, (4, 1, 1), axis=-1)
        true_xy = (true_box[..., 0:2] + true_box[..., 2:4]) / 2
        true_wh = true_box[..., 2:4] - true_box[..., 0:2]

        # give higher weights to small boxes
        box_loss_scale = 2 - true_wh[..., 0] * true_wh[..., 1]

        # 3. inverting the pred box equations
        grid_size = tf.shape(y_true)[1]
        grid = tf.meshgrid(tf.range(grid_size), tf.range(grid_size))
        grid = tf.expand_dims(tf.stack(grid, axis=-1), axis=2)
        true_xy = true_xy * tf.cast(grid_size, tf.float32) - \
            tf.cast(grid, tf.float32)
        true_wh = tf.math.log(true_wh / anchors)
        true_wh = tf.where(tf.math.is_inf(true_wh),
                           tf.zeros_like(true_wh), true_wh)

        # 4. calculate all masks
        obj_mask = tf.squeeze(true_obj, -1)
        # ignore false positive when iou is over threshold
        best_iou = tf.map_fn(
            lambda x: tf.reduce_max(broadcast_iou(x[0], tf.boolean_mask(
                x[1], tf.cast(x[2], tf.bool))), axis=-1),
            (pred_box, true_box, obj_mask),
            tf.float32)
        ignore_mask = tf.cast(best_iou < ignore_thresh, tf.float32)

        # 5. calculate all losses
        xy_loss = obj_mask * box_loss_scale * \
            tf.reduce_sum(tf.square(true_xy - pred_xy), axis=-1)
        wh_loss = obj_mask * box_loss_scale * \
            tf.reduce_sum(tf.square(true_wh - pred_wh), axis=-1)
        obj_loss = binary_crossentropy(true_obj, pred_obj)
        obj_loss = obj_mask * obj_loss + \
            (1 - obj_mask) * ignore_mask * obj_loss
        # TODO: use binary_crossentropy instead
        class_loss = obj_mask * sparse_categorical_crossentropy(
            true_class_idx, pred_class)

        # 6. sum over (batch, gridx, gridy, anchors) => (batch, 1)
        xy_loss = tf.reduce_sum(xy_loss, axis=(1, 2, 3))
        wh_loss = tf.reduce_sum(wh_loss, axis=(1, 2, 3))
```

```
        obj_loss = tf.reduce_sum(obj_loss, axis=(1, 2, 3))
        class_loss = tf.reduce_sum(class_loss, axis=(1, 2, 3))

        return xy_loss + wh_loss + obj_loss + class_loss
    return yolo_loss
```

Listing B.2: YoloV3 loss function used to train the risk-oblivious models from Chapter 4.

```
def YoloLoss(anchors, classes=80, ignore_thresh=0.5,actualClass,
    predClass,weight):
    def yolo_loss(y_true, y_pred):
        # 1. transform all pred outputs
        # y_pred: (batch_size, grid, grid, anchors, (x, y, w, h,
    obj, ...cls))
        pred_box, pred_obj, pred_class, pred_xywh = yolo_boxes(
            y_pred, anchors, classes)
        pred_xy = pred_xywh[..., 0:2]
        pred_wh = pred_xywh[..., 2:4]

        # 2. transform all true outputs
        # y_true: (batch_size, grid, grid, anchors, (x1, y1, x2, y2
    , obj, cls))
        true_box, true_obj, true_class_idx = tf.split(y_true, (4,
    1, 1), axis=-1)
        true_xy = (true_box[..., 0:2] + true_box[..., 2:4]) / 2
        true_wh = true_box[..., 2:4] - true_box[..., 0:2]

        # give higher weights to small boxes
        box_loss_scale = 2 - true_wh[..., 0] * true_wh[..., 1]

        # 3. inverting the pred box equations
        grid_size = tf.shape(y_true)[1]
        grid = tf.meshgrid(tf.range(grid_size), tf.range(grid_size)
    )
        grid = tf.expand_dims(tf.stack(grid, axis=-1), axis=2)
        true_xy = true_xy * tf.cast(grid_size, tf.float32) - \
            tf.cast(grid, tf.float32)
        true_wh = tf.math.log(true_wh / anchors)
        true_wh = tf.where(tf.math.is_inf(true_wh),
                           tf.zeros_like(true_wh), true_wh)

        # 4. calculate all masks
        obj_mask = tf.squeeze(true_obj, -1)
        # ignore false positive when iou is over threshold
        best_iou = tf.map_fn(
            lambda x: tf.reduce_max(broadcast_iou(x[0], tf.
    boolean_mask(
                x[1], tf.cast(x[2], tf.bool))), axis=-1),
            (pred_box, true_box, obj_mask),
            tf.float32)
        ignore_mask = tf.cast(best_iou < ignore_thresh, tf.float32)

        # 5. calculate all losses
        xy_loss = obj_mask * box_loss_scale * \
            tf.reduce_sum(tf.square(true_xy - pred_xy), axis=-1)
        wh_loss = obj_mask * box_loss_scale * \
            tf.reduce_sum(tf.square(true_wh - pred_wh), axis=-1)
```

```
        obj_loss = binary_crossentropy(true_obj, pred_obj)
        obj_loss = obj_mask * obj_loss + \
            (1 - obj_mask) * ignore_mask * obj_loss
        # TODO: use binary_crossentropy instead
        class_loss = obj_mask *sparse_categorical_crossentropy(
    true_class_idx, pred_class)

        ##########################
        #1) define weights matrix
        weights = np.ones((classes,classes))
        actual=actualClass
        predicted=predClass
        weights[predicted,actual] = weight

        #2) transform to arrays
        weights = np.array(weights)
        f = weights.shape[0]*(weights.shape[0])
        weights /= np.sum(weights)/f

        true_class_idxHot=tf.one_hot(tf.cast(true_class_idx, tf.
    int32),classes)#to categorical
        #delete 1 dimention as it was 16*13*13*3*1*20
        #and I need it 16*13*13*3*20
        true_class_idxHot=tf.squeeze(true_class_idxHot)

        #3)encode weights in formula
        #true_class_idxHot and pred class
        #both have the shape (16*13*13*3*20)
        w_flat = np.array(weights).flatten()
        WEIGHTS = tf.constant(w_flat, dtype=tf.float32)
        # this is equivalent to passing axis=4,
        #i.e. the max in the predicted class
        y_classT = K.argmax(true_class_idxHot)
        y_classP = K.argmax(pred_class)

        idx = y_classT + classes*y_classP
        w = tf.gather(WEIGHTS, idx)

        class_loss = class_loss * w
        '''End modify'''

        ##End of weighted section
        # 6. sum over (batch, gridx, gridy, anchors) => (batch, 1)
        xy_loss = tf.reduce_sum(xy_loss, axis=(1, 2, 3))
        wh_loss = tf.reduce_sum(wh_loss, axis=(1, 2, 3))
        obj_loss = tf.reduce_sum(obj_loss, axis=(1, 2, 3))
        class_loss = tf.reduce_sum(class_loss, axis=(1, 2, 3))
        return xy_loss + wh_loss + obj_loss + class_loss
    return yolo_loss
```

Listing B.3: Modified YoloV3 loss function used to train the risk-aware models from Chapter 4.

Appendix C

Chapter 5 - Supplementary Material

This appendix shows interesting scenarios from the simulation in Chapter 5. We show a crash in Figure C.1, misclassifications made by the ensembles synthesised for the motorway and town ODDs in Figures C.2 and C.3. And we show when the ensembles correctly classifies the objects in Figures C.4 and C.5.

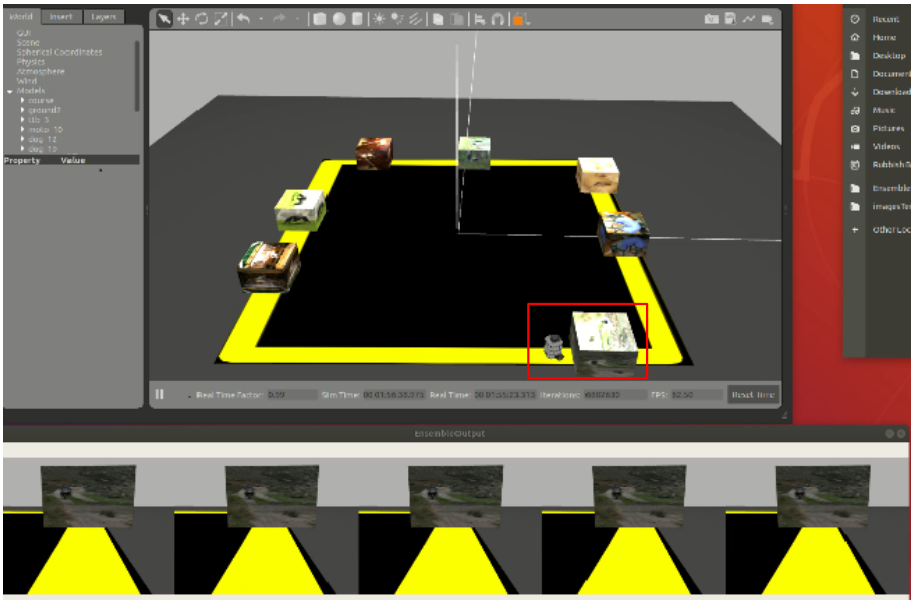


Figure C.1: A crash registered when using the town synthesised ensemble i.e. no adaptation (see also Figure 5.3).

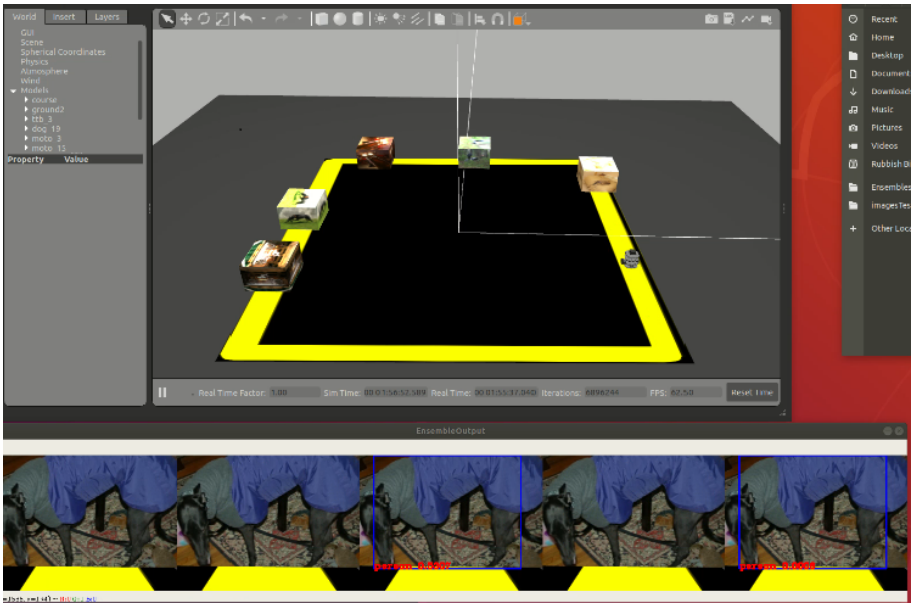


Figure C.2: The class dog misclassified as person by the motorway ensemble.

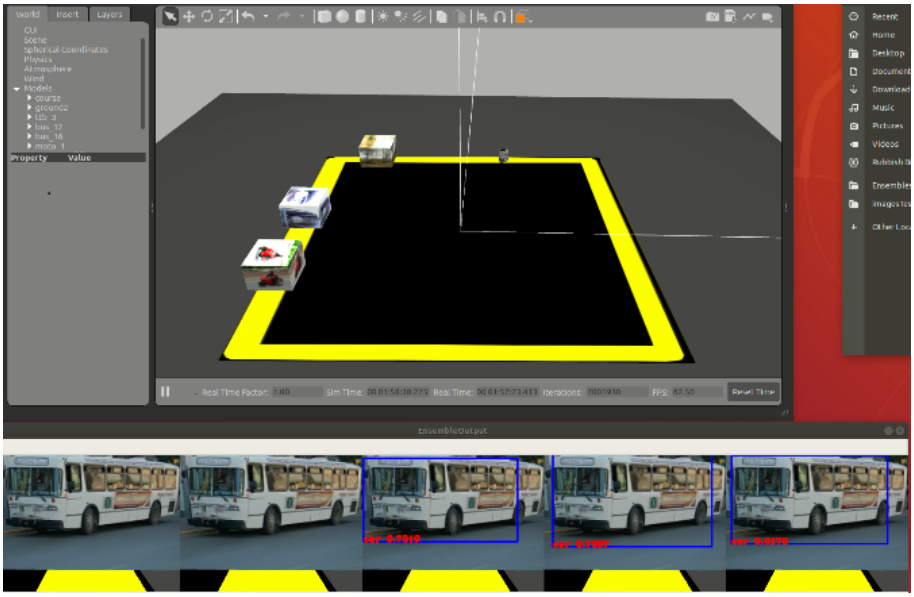


Figure C.3: The class bus misclassified as car by the town ensemble.

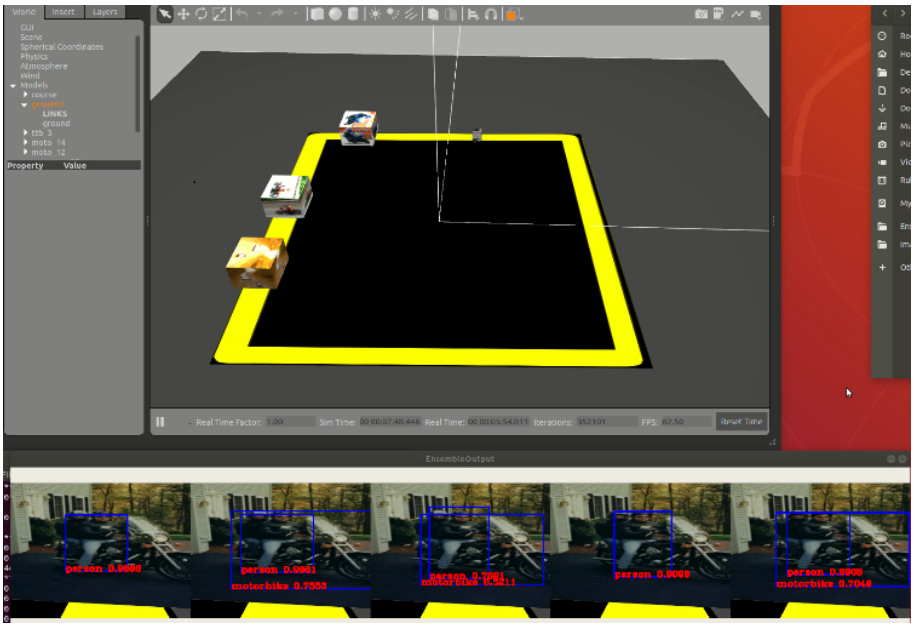


Figure C.4: Correct classification made by the ensemble when using adaptation.

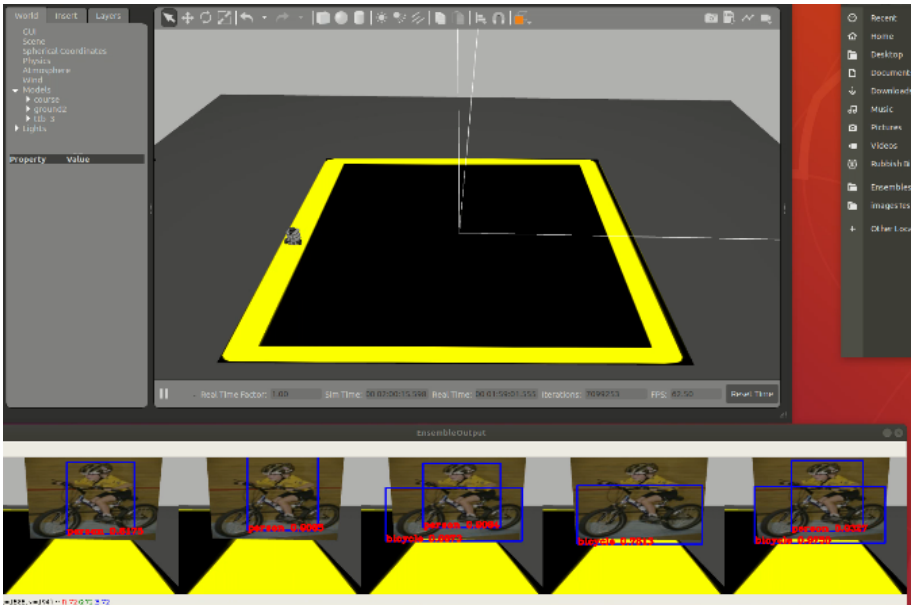


Figure C.5: Correct classification made by the ensemble when using adaptation.

Bibliography

[1] Rima Al-Ali, Lubomír Bulej, Jan Kofroň, and Tomáš Bureš. A guide to design uncertainty-aware self-adaptive components in cyber–physical systems. *Future Generation Computer Systems*, 128:466–489, 2022.

[2] Berat Mert Albaba and Sedat Ozer. Synet: An ensemble network for object detection in uav images. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 10227–10234, 2021.

[3] Wissam A. Albukhanajer, Yaochu Jin, and Johann A. Briffa. Classifier ensembles for image identification using multi-objective pareto features. *Neurocomputing*, 238:316–327, 2017.

[4] Hosein Alizadeh, Muhammad Yousefnezhad, and Behrouz Minaei Bidgoli. Wisdom of crowds cluster ensemble. *Intelligent Data Analysis*, 19(3):485–503, 2015.

[5] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3366–3375, 2017.

[6] Misael Alpizar Santana, Radu Calinescu, and Colin Paterson. Mitigating risk in neural network classifiers. In *48th Euromicro Conference Series on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2022.

[7] Misael Alpizar Santana, Radu Calinescu, and Colin Paterson. Risk-aware real-time object detection. In *2022 18th European Dependable Computing Conference (EDCC)*, pages 105–108. IEEE, 2022.

[8] Naomi Altman and Martin Krzywinski. Ensemble methods: bagging and random forests. *Nature Methods*, 14(10):933–935, 2017.

[9] MA Anusuya and Shriniwas K Katti. Speech recognition by machine, a review. *arXiv preprint arXiv:1001.2267*, 2010.

[10] Álvaro Arcos-García, Juan A Alvarez-Garcia, and Luis M Soria-Morillo. Deep neural network for traffic sign recognition systems: An analysis of spatial transformers and stochastic optimisation methods. *Neural Networks*, 99:158–165, 2018.

[11] Taiwo Oladipupo Ayodele. Types of machine learning algorithms. *New advances in machine learning*, 3:19–48, 2010.

[12] Kamyar Azizzadenesheli, Anqi Liu, Fanny Yang, and Animashree Anandkumar. Regularized learning for domain adaptation under label shifts. *arXiv preprint arXiv:1903.09734*, 2019.

[13] Qifang Bi, Katherine E Goodman, Joshua Kaminsky, and Justin Lessler. What is machine learning? a primer for the epidemiologist. *American journal of epidemiology*, 188(12):2222–2239, 2019.

[14] Chris Bogdiukiewicz, Michael Butler, Thai Son Hoang, Martin Paxton, James Snook, Xanthippe Waldron, and Toby Wilkinson. Formal development of policing functions for intelligent systems. In *28th International Symposium on Software Reliability Engineering (ISSRE)*, pages 194–204. IEEE, 2017.

[15] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[16] Peter Bühlmann. Bagging, boosting and ensemble methods. In *Handbook of computational statistics*, pages 985–1022. Springer, 2012.

[17] Jonathon Byrd and Zachary Lipton. What is the effect of importance weighting in deep learning? In *Int. Conf. on ML*, pages 872–881, 2019.

[18] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[19] Ricardo Diniz Caldas, Arthur Rodrigues, Eric Bernd Gil, Genaína Nunes Rodrigues, Thomas Vogel, and Patrizio Pelliccione. A hybrid approach combining control theory and ai for engineering self-adaptive systems. In *Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 9–19, 2020.

[20] Radu Calinescu, Raffaela Mirandola, Diego Perez-Palacin, and Danny Weyns. Understanding uncertainty in self-adaptive systems. In *2020 ieee international conference on autonomic computing and self-organizing systems (acsos)*, pages 242–251. IEEE, 2020.

[21] Manuel Carranza-García, Jesús Torres-Mateo, Pedro Lara-Benítez, and Jorge García-Gutiérrez. On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data. *Remote Sensing*, 13(1):89, 2020.

[22] Ángela Casado-García and Jónathan Heras. Ensemble methods for object detection. In *ECAI 2020*, pages 2688–2695. IOS Press, 2020.

[23] PR Caseley. Claims and architectures to rationate on automatic and autonomous functions. In *11th International Conference on System Safety and Cyber-Security (SSCS 2016)*, pages 1–6. IET, 2016.

[24] Yashvi Chandola, Jitendra Virmani, H.S. Bhadauria, and Papendra Kumar. Chapter 5 - Hybrid computer-aided classification system design using end-to-end CNN-based deep feature extraction and ANFC-LH classifier for chest radiographs. In Yashvi Chandola, Jitendra Virmani, H.S. Bhadauria, and Papendra Kumar, editors, *Deep Learning for Chest Radiographs*, Primers in Biomedical Imaging Devices and Systems, pages 141–156. Academic Press, 2021.

[25] Davide Chicco and Giuseppe Jurman. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):1–13, 2020.

[26] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[27] François Chollet. *Deep Learning with Python*. Manning, November 2017.

[28] Radoslaw M. Cichy and Daniel Kaiser. Deep neural networks as scientific models. *Trends in Cognitive Sciences*, 23(4):305–317, 2019.

[29] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996.

[30] Krzysztof Czarnecki. Operational world model ontology for automated driving systems–part 1: Road structure. *Waterloo Intelligent Systems Engineering Lab (WISE) Report*, 2018.

[31] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. *Advances in neural information processing systems*, 29, 2016.

[32] Truong Dang, Tien Thanh Nguyen, and John McCall. Toward an ensemble of object detectors. In *International Conference on Neural Information Processing*, pages 458–467. Springer, 2020.

[33] Abhishek Das, Mihir Narayan Mohanty, Pradeep Kumar Mallick, Prayag Tiwari, Khan Muhammad, and Hongyin Zhu. Breast cancer detection using an ensemble deep learning method. *Biomedical Signal Processing and Control*, 70:103009, 2021.

[34] Jeffrey De Fauw, Joseph R Ledsam, Bernardino Romera-Paredes, Stanislav Nikolov, Nenad Tomasev, Sam Blackwell, Harry Askham, Xavier Glorot, Brendan O'Donoghue, Daniel Visentin, et al. Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nature medicine*, 24(9):1342–1350, 2018.

[35] Kalyanmoy Deb, Ram Bhushan Agrawal, et al. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 1995.

[36] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[37] Naiyang Deng, Yingjie Tian, and Chunhua Zhang. *Support vector machines: optimization based theory, algorithms, and extensions*. CRC press, 2012.

[38] David M Dutton and Gerard V Conroy. A review of machine learning. *The knowledge engineering review*, 12(4):341–367, 1997.

[39] Charles Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001.

[40] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2147–2154, 2014.

[41] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[42] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[43] Peter Flach. Performance evaluation in machine learning: The good, the bad, the ugly, and the way forward. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:9808–9814, 07 2019.

[44] International Organization for Standardization. IEC 31010:2019. Risk management — Risk assessment techniques, 2019.

[45] Stephanie Forrest. Genetic algorithms. *ACM Computing Surveys (CSUR)*, 28(1):77–80, 1996.

[46] Félix-Antoine Fortin et al. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, Jul 2012.

[47] Alexander Fraser and Daniel Marcu. Measuring word alignment quality for statistical machine translation. *Computational Linguistics*, 33(3):293–303, 2007.

[48] Douglas Fraser, Ruben Giaquinta, Ruth Hoffmann, Murray Ireland, Alice Miller, and Gethin Norman. Collaborative models for autonomous systems controller synthesis. *Formal Aspects of Computing*, 32(2):157–186, 2020.

[49] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

[50] Jing Gao, Wei Fan, Jiawei Han, and Philip S Yu. A general framework for mining concept-drifting data streams with skewed distributions. In *Proceedings of the 2007 siam international conference on data mining*, pages 3–14. SIAM, 2007.

[51] Alexandros Gazis, Evangelos Ioannou, and Elefteria Katsiri. Examining the sensors that enable self-driving vehicles. *IEEE Potentials*, 39(1):46–51, 2019.

[52] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[53] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[54] Mario Gleirscher, Radu Calinescu, James Douthwaite, Benjamin Lesage, Colin Paterson, Jonathan Aitken, Rob Alexander, and James Law. Verified synthesis of optimal safety controllers for human-robot collaboration. *Science of Computer Programming*, 218:102809, 2022.

[55] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[56] J. Gorodkin. Comparing two k-category assignments by a k-category correlation coefficient. *Computational Biology and Chemistry*, 28(5):367–374, 2004.

[57] Yuzhuo Han, Xiaofeng Liu, Zhenfei Sheng, Yutao Ren, Xu Han, Jane You, Risheng Liu, and Zhongxuan Luo. Wasserstein loss-based deep object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 998–999, 2020.

[58] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

[59] Richard Hawkins, Colin Paterson, Chiara Picardi, Yan Jia, Radu Calinescu, and Ibrahim Habli. Guidance on the assurance of machine learning in autonomous systems (amlas). *arXiv preprint arXiv:2102.01564*, 2021.

[60] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[61] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[62] Sara M Hezavehi, Danny Weyns, Paris Avgeriou, Radu Calinescu, Raffaela Mirandola, and Diego Perez-Palacin. Uncertainty in self-adaptive systems: a research community perspective. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 15(4):1–36, 2021.

[63] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[64] Brody Huval, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, Joel Pazhayampallil, Mykhaylo Andriluka, Pranav Rajpurkar, Toki Migimatsu, Royce Cheng-Yue, et al. An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716*, 2015.

[65] ISO/IEC Standard 31010:2019. Risk Management—Risk Assessment Techniques, 2019.

[66] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[67] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[68] Kyle D Julian, Jessica Lopez, Jeffrey S Brush, Michael P Owen, and Mykel J Kochenderfer. Policy compression for aircraft collision avoidance systems. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pages 1–10. IEEE, 2016.

[69] Giuseppe Jurman, Samantha Riccadonna, and Cesare Furlanello. A Comparison of MCC and CEN Error Measures in Multi-Class Prediction. *PLoS ONE*, 7(8):e41882, 2012.

[70] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International conference on computer aided verification*, pages 97–117. Springer, 2017.

[71] Carl Kingsford and Steven L Salzberg. What are decision trees? *Nature biotechnology*, 26(9):1011–1013, 2008.

[72] Michal Kit, Ilias Gerostathopoulos, Tomas Bures, Petr Hnetynka, and Frantisek Plasil. An architecture framework for experimentations with self-adaptive cyber-physical systems. In *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 93–96. IEEE, 2015.

[73] David G Kleinbaum, K Dietz, M Gail, Mitchel Klein, and Mitchell Klein. *Logistic regression*. Springer, 2002.

[74] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.

[75] Quan Kong, Naoto Akira, Bin Tong, Yuki Watanabe, Daisuke Matsubara, and Tomokazu Murakami. Multimodal Deep Neural Networks Based Ensemble Learning for X-Ray Object Recognition. In *Asian Conference on Computer Vision*, pages 523–538. Springer, 2018.

[76] Bartosz Krawczyk and Michał Woźniak. Cost-sensitive neural network with ROC-based moving threshold for imbalanced classification. In *Int. Conf. Intell. Data Eng. and Automated Learning*, pages 45–52, 2015.

[77] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

[78] Roshan Kumari and Saurabh Kr Srivastava. Machine learning: A review on binary classification. *International Journal of Computer Applications*, 160(7), 2017.

[79] George Kyriakides and Konstantinos G Margaritis. *Hands-On Ensemble Learning with Python: Build highly optimized ensemble machine learning models using scikit-learn and Keras*. Packt Publishing Ltd, 2019.

[80] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[81] David D Lewis, Yiming Yang, Tony Russell-Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

[82] Hui Li, Xuesong Wang, and Shifei Ding. Research and development of neural network ensembles: a survey. *Artificial Intelligence Review*, 49(4):455–479, 2018.

[83] Jun Li, Xue Mei, Danil Prokhorov, and Dacheng Tao. Deep neural network for structural prediction and lane detection in traffic scene. *IEEE transactions on neural networks and learning systems*, 28(3):690–703, 2016.

[84] Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[85] Jihye Lim, Jungyoon Kim, and Songhee Cheon. A deep neural network-based method for early detection of osteoarthritis using statistical data. *International journal of environmental research and public health*, 16(7):1281, 2019.

[86] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[87] Tsung-Yi Lin et al. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pages 740–755, 2014.

[88] Tsung-Yi Lin et al. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

[89] Charles X. Ling and Victor S. Sheng. *Cost-Sensitive Learning*, pages 231–235. Springer US, Boston, MA, 2010.

[90] Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. Detecting and correcting for label shift with black box predictors. In *International conference on machine learning*, pages 3122–3130. PMLR, 2018.

[91] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single Shot multibox Detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[92] Xin Liu, Yanju Zhou, and Zongrun Wang. Recognition and extraction of named entities in online medical diagnosis data based on a deep neural network. *Journal of Visual Communication and Image Representation*, 60:1–15, 2019.

[93] Donghuan Lu, Karteek Popuri, Gavin Weiguang Ding, Rakesh Balachandar, Mirza Faisal Beg, Alzheimer's Disease Neuroimaging Initiative, et al. Multiscale deep neural network based analysis of fdg-pet images for the early diagnosis of alzheimer's disease. *Medical image analysis*, 46:26–34, 2018.

[94] Chao Luo, Xiaojie Li, Lutao Wang, Jia He, Denggao Li, and Jiliu Zhou. How does the data set affect cnn-based image classification performance? In *2018 5th International conference on systems and informatics (ICSAI)*, pages 361–366. IEEE, 2018.

[95] Amalia Luque, Alejandro Carrasco, Alejandro Martín, and Ana de las Heras. The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*, 91:216–231, 2019.

[96] Mathilde Machin, Jérémie Guiochet, Hélène Waeselynck, Jean-Paul Blanquart, Matthieu Roy, and Lola Masson. SMOF: A safety monitoring framework for autonomous systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(5):702–715, 2016.

[97] Batta Mahesh. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9:381–386, 2020.

[98] Alejandro Masrur, Michał Kit, Vladimír Matěna, Tomáš Bureš, and Wolfram Hardt. Component-based design of cyber-physical applications with safety-critical requirements. *Microprocessors and Microsystems*, 42:70–86, 2016.

[99] Georgios Mastorakis. Human-like machine learning: limitations and suggestions, 2018.

[100] Brian W Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.

[101] Marvin Minsky and Seymour A Papert. *Perceptrons, Reissue of the 1988 Expanded Edition with a new foreword by Léon Bottou: An Introduction to Computational Geometry*. MIT press, 2017.

[102] Akinori Mitani, Abigail Huang, Subhashini Venugopalan, Greg S Corrado, Lily Peng, Dale R Webster, Naama Hammel, Yun Liu, and Avinash V Varadarajan. Detection of anaemia from retinal fundus images via deep learning. *Nature Biomedical Engineering*, 4(1):18–27, 2020.

[103] Tadahiko Murata and Hisao Ishibuchi. MOGA: Multi-objective genetic algorithms. In *IEEE International Conference on Evolutionary Computation*, pages 289–294, 1995.

[104] Mahyar Najibi, Mohammad Rastegari, and Larry S Davis. G-CNN: an iterative grid based object detector. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2369–2377, 2016.

[105] Ali Bou Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, and Khaled Shaalan. Speech recognition using deep neural networks: A systematic review. *IEEE access*, 7:19143–19165, 2019.

[106] Bongjin Oh and Junhyeok Lee. A case study on scene recognition using an ensemble convolution neural network. In *2018 20th International Conference on Advanced Communication Technology (ICACT)*, pages 351–353. IEEE, 2018.

[107] Rafael Padilla, Sergio L Netto, and Eduardo AB Da Silva. A survey on performance metrics for object-detection algorithms. In *2020 international conference on systems, signals and image processing (IWSSIP)*, pages 237–242. IEEE, 2020.

[108] Colin Paterson, Haoze Wu, John Grese, Radu Calinescu, Corina S Păsăreanu, and Clark Barrett. DeepCert: Verification of contextually relevant robustness for neural network image classifiers. In *International Conference on Computer Safety, Reliability, and Security*, pages 3–17. Springer, 2021.

[109] P Pendharkar. Misclassification cost minimizing fitness functions for genetic algorithm-based artificial neural network classifiers. *Journal of the Operational Research Society*, 60(8):1123–1134, 2009.

[110] Parag C Pendharkar and Gary J Koehler. A general steady state distribution based stopping criteria for finite length genetic algorithms. *European Journal of Operational Research*, 176(3):1436–1451, 2007.

[111] Ignazio Pillai, Giorgio Fumera, and Fabio Roli. Designing multi-label classifiers that maximize f measures: State of the art. *Pattern Recognition*, 61:394–404, 2017.

[112] Benjamin Planche and Eliot Andres. *Hands-On Computer Vision with TensorFlow 2*. Packt Publishing Ltd, 2019.

[113] Ryan Poplin, Avinash V Varadarajan, Katy Blumer, Yun Liu, Michael V McConnell, Greg S Corrado, Lily Peng, and Dale R Webster. Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning. *Nature Biomedical Engineering*, 2(3):158, 2018.

[114] David MW Powers. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2020.

[115] Cambridge University Press. Risk, 2022.

[116] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.

[117] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[118] Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7263–7271, 2017.

[119] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[120] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 28:91–99, 2015.

[121] Brian D Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007.

[122] Lior Rokach. Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography. *Computational statistics & data analysis*, 53(12):4046–4072, 2009.

[123] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[124] SoAEI SAE. J3016-taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. *Surface Vehicle Recommended Practive*, 2016.

[125] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

[126] R. Saravanan and Pothula Sujatha. A state of art techniques on machine learning algorithms: A perspective of supervised learning approaches in data classification. In *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 945–949, 2018.

[127] Abhishek Sarda, Shubhra Dixit, and Anupama Bhan. Object detection for autonomous driving using YOLO [You Only Look Once] algorithm. In *2021 Third international conference on intelligent communication technologies and virtual mobile networks (ICICV)*, pages 1370–1374. IEEE, 2021.

[128] Burr Settles. Active learning literature survey. 2009.

[129] Burr Settles. From theories to queries: Active learning in practice. In *Active learning and experimental design workshop in conjunction with AISTATS 2010*, pages 1–18. JMLR Workshop and Conference Proceedings, 2011.

[130] Guofan Shao et al. Introducing Image Classification Efficacies. *IEEE Access*, 9:134809–134816, 2021.

[131] Neha Sharma, Vibhor Jain, and Anju Mishra. An analysis of convolutional neural networks for image classification. *Procedia computer science*, 132:377–384, 2018.

[132] Zhiqiang Shen, Zhuang Liu, Jianguo Li, Yu-Gang Jiang, Yurong Chen, and Xiangyang Xue. DSOD: Learning deeply supervised object detectors from scratch. In *Proceedings of the IEEE international conference on computer vision*, pages 1919–1927, 2017.

[133] Yong-Jun Shin, Joon-Young Bae, and Doo-Hwan Bae. Concepts and models of environment of self-adaptive systems: A systematic literature review. In *2021 28th Asia-Pacific Software Engineering Conference (APSEC)*, pages 296–305. IEEE, 2021.

[134] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[135] Atma Prakash Singh, Ravindra Nath, and Santosh Kumar. A survey: Speech recognition approaches and techniques. In *2018 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, pages 1–4. IEEE, 2018.

[136] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation. In *Australasian joint conference on artificial intelligence*, pages 1015–1021. Springer, 2006.

[137] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437, 2009.

[138] Gabriela Félix Solano, Ricardo Diniz Caldas, Genaína Nunes Rodrigues, Thomas Vogel, and Patrizio Pelliccione. Taming uncertainty in the assurance process of self-adaptive systems: a goal-oriented approach. In *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 89–99. IEEE, 2019.

[139] Shiva Soleymanpour et al. CSCNN: Cost-Sensitive Convolutional Neural Network for encrypted traffic classification. *Neural Processing Letters*, 53(5):3497–3523, 2021.

[140] Roman Solovyev, Weimin Wang, and Tatiana Gabruseva. Weighted boxes fusion: Ensembling boxes from different object detection models. *Image and Vision Computing*, 107:104117, 2021.

[141] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: A multi-class classification competition. In *The 2011 International Joint Conference on Neural Networks*, pages 1453–1460, 2011.

[142] Stanford Artificial Intelligence Laboratory et al. Robotic operating system.

[143] Neville A. Stanton, Paul M. Salmon, Guy H. Walker, and Maggie Stanton. Models and methods for collision analysis: A comparison study based on the Uber collision with a pedestrian. *Safety Science*, 120:117–128, 2019.

[144] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[145] Domen Tabernik and Danijel Skočaj. Deep learning for large-scale traffic-sign detection and recognition. *IEEE Transactions on Intelligent Transportation Systems*, 21(4):1427–1440, 2019.

[146] Xu-Sheng Tang, Zhe-Lin Shi, De-Qiang Li, Long Ma, and Dan Chen. Bagging-adaboost ensemble with genetic algorithm post optimization for object detection. In *2009 Fifth International Conference on Natural Computation*, volume 4, pages 528–534, 2009.

[147] B Templeton. Tesla in Taiwan crashes directly into overturned truck, ignores pedestrian, with autopilot on. *Forbes, Editors pic, June*, 2:2020, 2020.

[148] Alaa Tharwat. Classification assessment methods. *Applied Computing and Informatics*, 2020. `https://doi.org/10.1016/j.aci.2018.08.003`.

[149] Aastha Tiwari, Anil Kumar, and Goswami Mansi Saraswat. Feature extraction for object recognition and image classification. *International Journal of Engineering Research & Technology (IJERT)*, 2(10):2278–0181, 2013.

[150] Nathalie-Sofia Tomov and Stanimire Tomov. On deep neural networks for detecting heart disease. *arXiv preprint arXiv:1808.07168*, 2018.

[151] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Random k-labelsets for multilabel classification. *IEEE transactions on knowledge and data engineering*, 23(7):1079–1089, 2010.

[152] Grigorios Tsoumakas, Ioannis Partalas, and Ioannis Vlahavas. A taxonomy and short review of ensemble selection. In *Workshop on Supervised and Unsupervised Ensemble Methods and Their Applications*, pages 1–6, 2008.

[153] Gricel Vazquez Flores, Radu Calinescu, and Javier Camara. Scheduling of missions with constrained tasks for heterogeneous robot systems. In *Fourth Workshop on Formal Methods for Autonomous Systems*. York, 2022.

[154] Rahee Walambe, Aboli Marathe, Ketan Kotecha, and George Ghinea. Lightweight object detection ensemble framework for autonomous vehicles in challenging weather conditions. *Computational Intelligence and Neuroscience*, 2021, 2021.

[155] Hai Wang, Yijie Yu, Yingfeng Cai, Xiaobo Chen, Long Chen, and Yicheng Li. Soft-weighted-average ensemble vehicle detection method based on single-stage and two-stage deep learning models. *IEEE Transactions on Intelligent Vehicles*, 6(1):100–109, 2020.

[156] Qi Wang, Yue Ma, Kun Zhao, and Yingjie Tian. A comprehensive survey of loss functions in machine learning. *Annals of Data Science*, 9(2):187–212, 2022.

[157] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2097–2106, 2017.

[158] Junqing Wei, John M Dolan, Jarrod M Snider, and Bakhtiar Litkouhi. A point-based mdp for robust single-lane autonomous driving behavior under uncertainties. In *2011 IEEE International Conference on Robotics and Automation*, pages 2586–2592. IEEE, 2011.

[159] Pan Wei, John E Ball, and Derek T Anderson. Fusion of an ensemble of augmented image detectors for robust object detection. *Sensors*, 18(3):894, 2018.

[160] Danny Weyns, Bradley Schmerl, Masako Kishida, Alberto Leva, Marin Litoiu, Necmiye Ozay, Colin Paterson, and Kenji Tei. Towards better adaptive systems by combining mape, control theory, and machine learning. In *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 217–223. IEEE, 2021.

[161] Eyal Wirsansky. *Hands-on genetic algorithms with Python: applying genetic algorithms to solve real-world deep learning and artificial intelligence problems*. Packt Publishing Ltd, 2020.

[162] Jie Xu, Wei Wang, Hanyuan Wang, and Jinhong Guo. Multi-model ensemble with rich spatial information for object detection. *Pattern Recognition*, 99:107098, 2020.

[163] Bo Yang et al. Lessons learned from accident of autonomous vehicle testing: An edge learning-aided offloading framework. *IEEE Wireless Communications Letters*, 9(8):1182–1186, 2020.

[164] Donggeun Yoo, Sunggyun Park, Joon-Young Lee, Anthony S Paek, and In So Kweon. Attentionnet: Aggregating weak directions for accurate object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2659–2667, 2015.

[165] Wei Yu, Jing Chang, Cheng Yang, Limin Zhang, Han Shen, Yongquan Xia, and Jin Sha. Automatic classification of leukocytes using deep neural network. In *2017 IEEE 12th international conference on ASIC (ASICON)*, pages 1041–1044. IEEE, 2017.

[166] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824, 2019.

[167] Cha Zhang and Yunqian Ma. *Ensemble machine learning: methods and applications*. Springer, 2012.

[168] Chong Zhang, Kay Chen Tan, Haizhou Li, and Geok Soon Hong. A cost-sensitive deep belief network for imbalanced classification. *IEEE transactions on neural networks and learning systems*, 30(1):109–122, 2018.

[169] Dan Zhang, Licheng Jiao, Xue Bai, Shuang Wang, and Biao Hou. A robust semi-supervised svm via ensemble learning. *Applied Soft Computing*, 65:632–643, 2018.

[170] Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, 2019.

[171] Xiangzeng Zhou, Lei Xie, Peng Zhang, and Yanning Zhang. An ensemble of deep neural networks for object tracking. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 843–847. IEEE, 2014.

[172] Zhi-Hua Zhou and Xu-Ying Liu. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on knowledge and data engineering*, 18(1):63–77, 2005.

[173] Weiwei Zong, Guang-Bin Huang, and Yiqiang Chen. Weighted extreme learning machine for imbalance learning. *Neurocomputing*, 101:229–242, 2013.