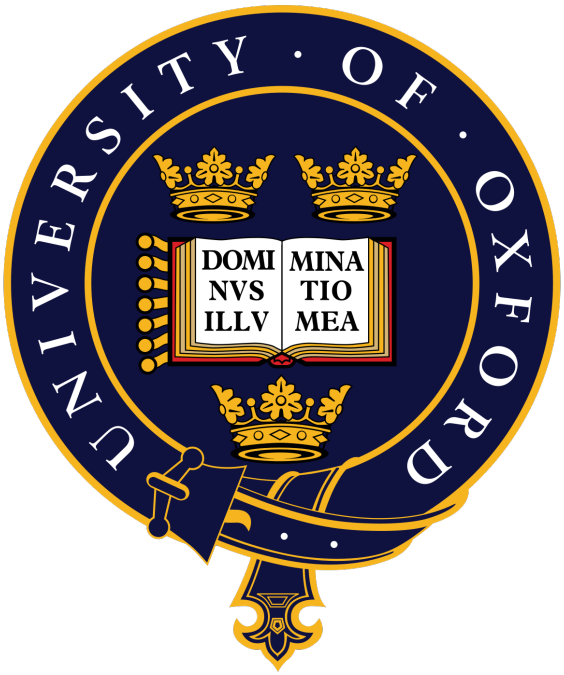
Learning Distributional Uncertainty Estimation for Semantic Segmentation

David Williams

Worcester College



A thesis submitted for the degree of *Doctor of Philosophy*.

Acknowledgements

I would firstly like to thank my supervisor, Paul Newman, for providing the means and the environment that allowed me to pursue this DPhil. I left each of our meetings with a greater sense of perspective and zeal for our projects, and this was no doubt a considerable help for the times when it wasn't clear a polished research product was in sight. Your rootedness in the practical aspects of robotics always gave me the confidence that we were solving something worthwhile.

Secondly, a thanks to our MRG postdocs, Daniele and Matt. Our conversations were extremely important in keeping the work on track and your constant presence as a sounding board for ideas was invaluable. Along with Matt and Daniele, a thanks also has to go out to the wider MRG and ORI community. Aspects of doing a PhD are inherently solitary, but being alongside people on the same journey is a great antidote to this.

Finally, I want to thank those who have helped me outside of the academic community: my girlfriend Chloe, my friends and my family. Whilst Oxford is a brilliant place to be, spending time outside of the city with you all kept me going and lent a sense of perspective to the whole process.

A special mention goes to my parents and grandparents, whose unwavering support and emphasis on the value of education have been fundamental to my journey. For this, I am truly grateful.

David Williams
5th May 2024

Abstract

Given that robots take consequential actions in the real-world, it should be ensured that their deployment, insofar as possible, is safe and trustworthy by design. Accordingly, this thesis tackles a problem known as distributional shift, which occurs when a deep learning system is exposed to data that is shifted from the data distribution it was trained on, and can result in unpredictable and unintended deployment scenarios. For the task of semantic segmentation, this thesis investigates how a system can detect when error occurs due to distributional shift in order to prevent these dangerous scenarios.

After a discussion of both the nature of distributional uncertainty, i.e. that which causes error due to distributional shift, and the existing literature, this thesis presents three methods that perform distributional uncertainty estimation alongside semantic segmentation for driving data.

The first method poses the problem as a large-scale out-of-distribution detection problem, where a large-scale image dataset is used to train a segmentation neural network to separate in-distribution and out-of-distribution training instances. The training method for this involves a contrastive loss function and a data augmentation procedure that reduces the difference in appearance between in-distribution and out-of-distribution instances.

The second method takes learnings from the first, in that it uses out-of-distribution training images that are inherently less distributionally-shifted from the in-distribution images, rather than relying on data augmentation. This makes the task of separating them more challenging, and therefore the learned uncertainty estimation more robust. For this reason, this method is designed to use an unlabelled distributionally-shifted driving dataset and proposes a training procedure to account for the lack of labels.

Finally, the third method combines ideas from the previous two approaches by using both large-scale image data to learn a general feature representation and an unlabelled distributionally-shifted driving dataset to tailor this representation to distributional uncertainty estimation for driving images.

Contents

1 Introduction **9**

1.1 Motivation 9

1.2 Semantics in Robotics 10

1.3 Mitigating Distributional Shift 11

1.3.1 Reduction of Distributional Uncertainty 11

1.3.2 Estimation of Distributional Uncertainty 12

1.3.3 To reduce or detect? 13

1.4 Thesis Structure 13

2 Introduction to Semantic Segmentation **16**

2.1 Semantic Segmentation Preliminaries 17

2.1.1 The Task 17

2.1.2 Deep Semantic Segmentation Networks 17

2.1.3 Neural Network Training 18

2.1.4 State-of-the-Art Methods 19

2.2 Overfitting 21

2.3 Calibration of Deep Neural Networks 22

2.3.1 Calculating Calibration 23

2.3.2 Observations of Miscalibration 24

2.3.3 Causes of Neural Network Miscalibration 25

2.4 Conclusion 28

3 Uncertainty Estimation in Deep Learning **30**

3.1 Sources of Uncertainty 31

3.2 Origins of Distributional Uncertainty . . . 32
3.3 Epistemic Uncertainty Estimation . . . 37
3.3.1 Principles for Epistemic Uncertainty Estimation . . . 37
3.3.2 Methods for Epistemic Uncertainty Estimation . . . 39
3.3.3 Discussion . . . 41
3.4 Aleatoric Uncertainty Estimation . . . 41
3.4.1 Learned Loss Attenuation . . . 42
3.4.2 Direct Error Estimation . . . 45
3.4.3 Generative Modelling . . . 47
3.4.4 Test-Time Augmentation . . . 48
3.4.5 Discussion . . . 49
3.5 Out-of-Distribution Detection . . . 50
3.5.1 Pretrained Methods . . . 51
3.5.2 Regularisation-Based Methods . . . 52
3.5.3 Self-Supervised Learning for OoD Detection . . . 53
3.5.4 Proxy Task Methods . . . 54
3.5.5 Deep Generative Models . . . 55
3.5.6 Use of Out-of-Distribution Data . . . 56
3.5.7 Discussion . . . 57
3.6 Conclusion . . . 58

4 Model Evaluation and Datasets **60**
4.1 Model Evaluation & Metrics . . . 61
4.1.1 Misclassification Detection . . . 62
4.1.2 Metrics: Definitions . . . 62
4.1.3 Metrics: Discussion . . . 65
4.2 SAX Semantic Segmentation Dataset . . . 67
4.2.1 Dataset Motivation . . . 68
4.2.2 Semantic Definitions . . . 69
4.2.3 Inclusion of multiple target domains . . . 70
4.2.4 Curation of the unlabelled SAX training datasets . . . 71

4.3 Other Driving Datasets . . . 72
4.3.1 Cityscapes . . . 72
4.3.2 Berkeley DeepDrive . . . 72
4.3.3 WildDash . . . 73
4.3.4 KITTI . . . 73

5 Learning OoD Detection from Large-Scale Datasets 80
5.1 Motivation . . . 81
5.1.1 Contrastive Learning . . . 82
5.1.2 Training Data . . . 84
5.2 Proposed System Design . . . 86
5.2.1 Overview . . . 86
5.2.2 Objective Function . . . 86
5.2.3 Masking Label Noise . . . 89
5.2.4 Data Augmentation . . . 89
5.3 Experimental Setup . . . 91
5.3.1 Datasets . . . 91
5.3.2 Network Architecture . . . 92
5.4 Experiments and Results . . . 93
5.4.1 Data Augmentation Experiments . . . 94
5.4.2 Data Augmentation Results . . . 94
5.4.3 Objective Function Experiments . . . 94
5.4.4 Objective Function Results . . . 95
5.4.5 Data Diversity Experiments . . . 95
5.4.6 Data Diversity Results . . . 96
5.5 Conclusion . . . 96

6 Learning Uncertainty Estimation from Uncurated Domain Data 101
6.1 Motivation . . . 103
6.1.1 Using Unlabelled Out-of-Distribution Driving Data . . . 103
6.1.2 Introduction to γ-SSL . . . 105

6.1.3 How to tailor self-supervised methods for uncertainty estimation? . . 106
6.2 Preliminaries . . . 107
6.2.1 Segmentation via Prototypes . . . 107
6.2.2 Uncertainty Estimation via a Feature-Space Threshold . . . 108
6.3 Crop & Resize Data Augmentation . . . 109
6.3.1 Method . . . 109
6.4 Training Architecture . . . 110
6.5 Training Objective . . . 112
6.5.1 Calculating γ . . . 113
6.5.2 Learning $\mathtt{E}$ . . . 113
6.5.3 Learning the Task . . . 115
6.5.4 Preventing Feature Collapse . . . 115
6.6 Training Procedure . . . 117
6.6.1 Model Pretraining . . . 117
6.6.2 Domain-based Curriculae . . . 118
6.7 Network Architecture . . . 119
6.8 Baselines . . . 119
6.9 Evaluating uncertainty estimation on narrow target domains . . . 121
6.9.1 Source: Cityscapes, Target: SAX Test Datasets . . . 122
6.9.2 Effect of distributional shift . . . 124
6.9.3 Source: Cityscapes, Target: KITTI & BDD . . . 126
6.9.4 Source: BDD, Target: SAX Test Datasets . . . 128
6.10 Evaluating uncertainty estimation on a general target domain . . . 132
6.10.1 Target: WildDash . . . 132
6.11 Miscellaneous experiments . . . 133
6.11.1 Calculation of the optimal threshold . . . 134
6.11.2 Calculating thresholds across domains . . . 135
6.11.3 Latency Evaluation . . . 136
6.12 Qualitative Results . . . 137
6.13 Ablation Studies . . . 139

6.13.1 Estimating Uncertainty via Distance to Prototypes . . . 139
6.13.2 Importance of target domain images . . . 139
6.13.3 Importance of $\mathbf{M}^{\gamma}$ in the training objective . . . 141
6.13.4 Importance of Branch Asymmetry . . . 141
6.13.5 Importance of L^{u} and L^{p} . . . 142
6.13.6 Importance of Crop-and-Resize Data Augmentation . . . 142
6.13.7 Importance of using hard $\mathbf{M}^{\gamma}$ . . . 143
6.13.8 Possibility of class-wise thresholds . . . 143
6.13.9 The need for large batch sizes to calculate prototypes . . . 144
6.14 Conclusion . . . 146

7 Learning Uncertainty Estimation with Masking & Foundation Models 147
7.1 Foundation Models . . . 149
7.1.1 Preliminaries on Foundation Models . . . 149
7.1.2 Foundation Models for Semantic Segmentation . . . 149
7.1.3 Model Distillation . . . 151
7.2 Uncertainty Estimation for Foundation Models . . . 152
7.3 Training Framework . . . 153
7.4 Uncertainty Training . . . 154
7.4.1 Learning Uncertainty Estimation . . . 155
7.4.2 Avoiding Feature Collapse . . . 158
7.5 Masked Image Modelling for Uncertainty Estimation . . . 159
7.5.1 Background . . . 159
7.5.2 Motivation . . . 160
7.5.3 Masking Policy . . . 161
7.6 Experimental Setup . . . 162
7.6.1 Network Architecture . . . 162
7.6.2 Network Initialisation . . . 162
7.6.3 Data . . . 163
7.6.4 Baselines . . . 164
7.7 Experiments and Results . . . 165

7.7.1 Different Target Domains 166
7.7.2 Freezing E 167
7.7.3 Comparing perturbation methods 168
7.7.4 Freezing $\mathtt{f}_\theta$ 169
7.8 Conclusion 169

8 Conclusion **171**
8.1 Summary 171
8.2 Closing Remarks 174

Bibliography **174**

Chapter 1

Introduction

Contents

1.1 Motivation **9**

1.2 Semantics in Robotics **10**

1.3 Mitigating Distributional Shift **11**

1.3.1 Reduction of Distributional Uncertainty 11

1.3.2 Estimation of Distributional Uncertainty 12

1.3.3 To reduce or detect? 13

1.4 Thesis Structure **13**

1.1 Motivation

Robots take actions in the real-world, and so it is of paramount importance that the safety of their deployment is carefully considered. One way of doing this uses the concept of an Operational Design Domain (ODD), which defines a set of operating conditions under which a robot has been designed and empirically shown to operate safely [1].

More and more, robotic systems are being designed to make use of deep learning, where complex tasks can be learned directly from training data. This includes perception tasks [2]–[4], planning tasks [5]–[7], control tasks [8]–[10], or the entire system from perception to action [11]–[13]. Deep learning systems typically achieve their best task performance for input data that is similar to the training data. Therefore, in order to deploy a system, we prepare

a training dataset from the operating conditions in the ODD, and empirically evaluate the robot in these conditions to make sure performance is sufficient.

However, if we consider deploying a robot into a dynamic real-world environment, there is no guarantee that these operating conditions remain constant for the duration of the deployment. For example, in outdoor settings, weather may change, the sun may come out, the street furniture may be modified, never-before-seen dynamic objects may appear etc. This causes a *distributional shift* of the data input to the system, and can lead to dangerous and unpredictable robot behaviour unless mitigated.

One option to mitigate this is to extend the ODD to include each of these additional factors of variation. However, preparing a dataset of this kind is inherently difficult for dynamic and diverse environments.

Therefore, this thesis investigates the alternative option of designing a system that has a strong sense of the boundaries of its ODD, and is able to detect when it is no longer operating within it. The focus of this thesis is constrained to the important perception task of semantic segmentation of RGB images, in which every pixel is assigned to a semantic class.

1.2 Semantics in Robotics

It is vital for a robot to have not just a geometric, but a semantic understanding of its surroundings. To appreciate this, we will describe the importance of semantics in three basic steps of robot operation.

Firstly, robots must understand where they are in the environment. Semantics are useful for this as they provide a high-level description of a scene. This can be used to determine which objects are dynamic or ephemeral, and which objects are static and permanent. The latter are the elements to focus on to solve the problems of localisation and odometry, while the former introduce noise and uncertainty to these problems, as discussed in [14]. Additionally, describing a scene in terms of high-level semantics is more memory efficient than using appearance-based methods, resulting in smaller maps and lower cost localisation, e.g. [15].

Secondly, robots need to plan a path towards a goal location. Traversing man-made environments often involves following rules, such as staying within lane markings, following

traffic signs or sticking to the pavement. These rules require the extraction of semantic information from the static world, such as performed in [16]–[19]. On the way to a goal location, detecting dynamic objects is a key step in obstacle avoidance, and can be aided by the delineation between semantic objects, such as in [20], [21].

Thirdly, once the robot has reached its location, it has to perform a task. Most tasks that we might want robots to perform are best defined in terms of higher-level semantics, rather than geometry, e.g. pick the *apples* in an orchard [22], detect *people* in a search and rescue setting [23], clean the *dirt* on the floor in an office environment [24]. Therefore, identifying semantic entities is often a necessary step in solving the task of interest.

For these reasons, it is a key robotics task to localise task-relevant semantic classes within images. Therefore, this thesis considers the task of semantic segmentation, which assigns a class to each pixel in an image. This thesis focusses on this task in the context of autonomous driving, and how it might be solved with a consideration of its safety-critical nature.

1.3 Mitigating Distributional Shift

As introduced in Section 1.1, it is vitally important that distributional uncertainty – i.e. that which causes errors due distributional shift – does not lead to dangerous unknown or unintended scenarios. Consequently, one or both of the following steps ought to be taken: (1) the distributional uncertainty is reduced, (2) the distributional uncertainty is detected, and risk-mitigating measures are taken.

1.3.1 Reduction of Distributional Uncertainty

Reducing the distributional uncertainty is equivalent to increasing the breadth of the safe operating conditions defined in the ODD. This can be achieved by collecting and annotating a larger and more diverse training dataset for the segmentation model. However, to ensure safe deployment, this labelled dataset needs to be an approximation of any and all of the possible scenes that the robot is likely to come across during a deployment. The scale of this dataset would therefore be vast, and so would the resource cost be. For example, the annotation and quality control for a single image of the Cityscapes dataset [25] took 1.5

hours, meaning the entire dataset of 5000 images took 7500 hours, or 312.5 days to annotate.

An alternative solution to the problem is domain adaptation. In this framework, there is a labelled dataset from one domain, named the *source* domain. There is also another dataset of unlabelled images or a small number of labelled images from a distributionally-shifted domain, named the *target* domain. The domain adaptation task is to train on data from both the source domain and the target domain, in order to reduce the error rate on a test dataset in the target domain. Suppose the source domain is defined by images inside the ODD, and the target domain is a set of images of scenes outside of the ODD. There are methods that exist to train on an unlabelled target domain dataset [26]–[29], which in this case would extend the ODD.

Lastly, an increasingly prevalent method for reducing distributional uncertainty is to initially train a model on a very broad distribution of unlabelled natural images using Self-Supervised Learning (SSL), as demonstrated in [30]–[32]. This model can subsequently be leveraged to solve a range of computer vision tasks on a range of different datasets.

1.3.2 Estimation of Distributional Uncertainty

Alternatively, instead of extending the ODD, a robot can detect its limits by performing distributional uncertainty estimation. As we are interested in the pixel-wise task of semantic segmentation, we are also interested in pixel-wise distributional uncertainty estimation, i.e. detecting pixels that are incorrectly segmented due to distributional shift.

When a robot system detects an image region which is unknown and incorrectly segmented, it is then given the opportunity to improve the safety of the system by dealing with this event appropriately. As an example, if this image region overlaps with the robot's planned trajectory, then it can decide to perform a risk-minimising manoeuvre (e.g. smoothly coming to a stop) and hand-over to a human operator. Therefore, if a robot is forced out of its ODD, it can 'fail gracefully', and can be prevented from making decisions using an incorrect understanding of its surroundings.

1.3.3 To reduce or detect?

This thesis focusses on the estimation of distributional uncertainty, but ultimately both approaches are important research directions, as they reduce the error rate in orthogonal manners. The justification for the focus of this thesis is based on the fact that distributional uncertainty cannot currently be fully reduced, and perhaps never will be for all settings.

Due to the recent advent of foundation models in computer vision [30], [31], it is increasingly possible to train models to accurately perform computer vision tasks on an extremely broad distribution of images. However, we suggest that there will still be utility in specialist models trained to perform well on specific image domains. This could be true for reasons of (1) accuracy, i.e. models fine-tuned on a specific image domain outperform models trained on all images, or (2) inference cost, i.e. smaller neural networks use less memory and have a lower latency, but are less expressive and so only perform well on a subset of all natural images, or (3) training cost, i.e. training on large datasets is time-consuming and costly, therefore smaller or more specialist models have utility in these settings.

In these contexts, these smaller or more specific models have no need to be able to perform their task for the entirety of the natural image distribution, and therefore high-quality uncertainty estimation will be a key driver of their safe deployment into the world.

A second reason is that it is likely to be easier to detect distributional uncertainty than reduce it, as will be discussed further in Section 3.2 and Section 5.1. Broadly, this is because detecting distributional uncertainty requires the model only to discriminate between the classes within the ODD, and those outside of it. Crucially, it removes the requirement to discriminate between each of the semantic classes that are not within the ODD, and thus this task is less complex than reducing distributional uncertainty.

1.4 Thesis Structure

This thesis is about taking steps to solve: **the estimation of distributional uncertainty to mitigate the effects of distributional shift on the task of semantic segmentation for mobile robotics** . This is achieved by proposing three methods, which are presented in Chapter 5, Chapter 6 and Chapter 7, as well as in the following publications:

- D. Williams, M. Gadd, D. De Martini, and P. Newman, "Fool Me Once: Robust Selective Segmentation via Out-of-Distribution Detection with Contrastive Learning", IEEE International Conference on Robotics and Automation (ICRA), 2021.
- D. Williams, D. De Martini, M. Gadd, and P. Newman, "Mitigating Distributional Shift in Semantic Segmentation via Uncertainty Estimation from Unlabelled Data", IEEE Transactions on Robotics (T-RO), 2024.
- D. Williams, M. Gadd, P. Newman, and D. De Martini, "Masked γ-SSL: Learning Uncertainty Estimation via Masked Image Modeling", IEEE International Conference on Robotics and Automation (ICRA), 2024.

As will be discussed throughout this thesis, the thread that ties these methods together is that they each seek to learn distributional uncertainty estimation directly from out-of-distribution (OoD) training data, however they are tailored to use different types of training datasets to achieve this.

Before these methods are presented, Chapter 2, Chapter 3 and Chapter 4 provide crucial background information on the task of interest, and this ultimately motivates the methods in the later chapters.

Chapter 2 introduces the task of semantic segmentation, the deep learning methods used to solve it, and the notation used in this thesis. It then discusses the effect of distributional shift of these methods, and the concept of neural network miscalibration, which limits these methods' ability to detect distributional shift. Finally, it discusses the implications of miscalibration in the context of safety-critical robotics.

Chapter 3 firstly investigates the nature of distributional uncertainty, and contributes a useful framing of it. The existing literature related to distributional uncertainty estimation, namely uncertainty estimation and OoD detection, is then presented. The discussion of this literature is conditioned on this chapter's framing of distributional uncertainty, as well as the constraints of mobile robotics. The chapter concludes with a discussion of the themes of research that will be found in our proposed methods in Chapter 5, Chapter 6 and Chapter 7.

Chapter 4 discusses the datasets and model evaluation strategies used in this thesis, with a focus on our mobile robotics setting. This includes a dataset that has been developed over the course of this thesis, for the specific problem of measuring the quality of distributional

uncertainty estimation across a range of magnitudes of distributional shift. This chapter also introduces misclassification detection, which is the task used to measure the quality of uncertainty estimation.

Chapter 5 introduces the first of our proposed methods, where the training task is formulated as a large-scale pixel-wise OoD detection problem. The key contributions are: (1) a training algorithm that uses contrastive learning to leverage a large-scale image recognition training dataset, (2) a data augmentation technique that combines in-distribution and OoD instances within the same images, and reduces the distributional shift between the OoD dataset and the in-distribution dataset, allowing for more pixel-wise robust OoD detection.

Next, Chapter 6 presents the second of our proposed methods. Instead of using a large-scale image recognition dataset as in Chapter 5, this work instead uses a dataset that is distributionally shifted from the source domain, but is still real-world driving data. This is because this type of dataset naturally contains in-distribution and OoD instances within the same image, and the OoD instances are more subtly different. This choice, however, brings many challenges as there is no supervision provided by this dataset. The key contributions of this chapter are: (1) a training algorithm that uses *unlabelled* target domain data to learn distributional uncertainty estimation, (2) an extensive set of experiments that evaluate the quality of uncertainty estimation for a wide variety of benchmarks.

Finally, Chapter 7 presents the last of our proposed methods. This uses a similar formulation to the previous chapter, but introduces the use of foundation models, which emerged over the course of this thesis. In this sense, this method uses a large-scale image recognition dataset in addition to the distributionally-shifted driving dataset used in the previous chapter. The key contributions for this work are: (1) a framework for fine-tuning a foundation model to solve a specific task, while also learning to maintain generality for uncertainty estimation (2) a method using masked image modelling, instead of the data augmentation used in Chapter 6, to train a fine-tuned segmentation network to perform uncertainty estimation without the requirement for ground-truth, (3) an empirical investigation on the effect of foundation model pre-training on quality of uncertainty estimation.

Finally, Chapter 8 summarizes the contributions of this thesis.

Chapter 2

Introduction to Semantic Segmentation

Contents

2.1 Semantic Segmentation Preliminaries 17

2.1.1 The Task 17

2.1.2 Deep Semantic Segmentation Networks 17

2.1.3 Neural Network Training 18

2.1.4 State-of-the-Art Methods 19

2.2 Overfitting 21

2.3 Calibration of Deep Neural Networks 22

2.3.1 Calculating Calibration 23

2.3.2 Observations of Miscalibration 24

2.3.3 Causes of Neural Network Miscalibration 25

2.4 Conclusion 28

This chapter introduces the task of semantic segmentation, the current methods used to solve it, and the limitations of these methods for robots operating in diverse and dynamic environments. In Section 2.1, the notation to be used throughout this thesis is presented, along with detail about how deep segmentation networks are designed and trained. Section 2.2 discusses how large neural networks are prone to overfitting and Section 2.3 describes the concept of calibration and the empirical findings of neural network miscalibration. The chapter ends with Section 2.4, which discusses the implications of neural network

calibration when deep semantic segmentation networks are used as components of robotic systems.

2.1 Semantic Segmentation Preliminaries

2.1.1 The Task

Semantic segmentation of RGB images is a task that requires the estimation of the semantic class of every pixel in an image. We firstly define a set of K semantic classes, $\mathbb{K} = \{k_1, \dots k_K\}$, that are of interest for a given setting. These classes are then visually defined using N natural RGB images, $\mathbf{X} \in \mathbb{R}^{N \times 3 \times H \times W}$ with corresponding pixel-wise labels $\mathbf{Y}^* \in \mathbb{R}^{N \times H \times W}$, resulting in a dataset $\mathbb{D} = \{\mathbf{X}, \mathbf{Y}^*\}$.

The semantics are encoded in a given label $\mathbf{y}^* \subset \mathbf{Y}^*$ by assigning each pixel to the index of the semantic class, $\mathbf{y}^* \in \{n \in \mathbb{Z} \mid 1 \leq n \leq K\}^{H \times W}$. The dataset $\mathbb{D}$ jointly defines which classes are of interest, and the appearance they take.

For a given RGB image $\mathbf{x} \in \mathbb{R}^{3 \times H \times W}$, the semantic segmentation task is to return a segmentation map, $\mathbf{y} \in \{n \in \mathbb{Z} \mid 1 \leq n \leq K\}^{H \times W}$, which is identical to the corresponding label $\mathbf{y}^* \in \{n \in \mathbb{Z} \mid 1 \leq n \leq K\}^{H \times W}$.

2.1.2 Deep Semantic Segmentation Networks

Semantic segmentation performance has increased enormously over the last decade due to the application of deep learning to the problem. The availability of large pixel-wise annotated datasets, innovation in neural network architectures, and decrease in cost of parallel compute has made this possible.

Therefore, a typical solution to the semantic segmentation problem is to train a neural network $\mathtt{f}_\theta$, which is parameterised by θ. A segmentation neural network $\mathtt{f}_\theta$ typically returns unnormalised log-probabilities for each pixel and each class, which are referred to as logits, $\mathbf{l} = \mathtt{f}_\theta(\mathbf{x}) \in \mathbb{R}^{K \times H \times W}$. From this, we can obtain the estimated segmentation map via $\mathbf{y} = \operatorname{argmax}(\mathbf{l})$.

Additionally, we might also want to consider the per-pixel categorical distribution $\mathbf{p} \in [0, 1]^{K \times H \times W}$. For a pixel location i, pixel value $x = \mathbf{x}_i \in \mathbb{R}^3$ and pixel-wise segmentation

$y = \mathbf{y}_i \in \mathbb{R}^1$, this distribution is given by $\mathbf{p}_i = p(y|x) \in [0,1]^K$. Note that in truth, the categorical distribution for a given pixel is conditioned on the entire input image, so should be represented as $\mathbf{p} = p(y|\mathbf{x})$, however $p(y|x)$ is used for the sake of simplicity. This is expanded as:

$$\mathbf{p}_i = p(y|x) = [p(y = k_1|x), \ldots, p(y = k_K|x)] \in [0,1]^K \tag{2.1}$$

This is typically calculated as:

$$p(y|x) = \text{softmax}_\tau(\mathbf{l}_i) = \text{softmax}_\tau([\mathbf{l}_{i,1}, \ldots, \mathbf{l}_{i,K}]) \tag{2.2}$$

This is using the softmax function softmax, which can be calculated for a given temperature τ as:

$$\text{softmax}_\tau(z_j) = \frac{\exp(z_j/\tau)}{\sum_{k=1}^{K} \exp(z_k/\tau)} \tag{2.3}$$

If the subscript τ is omitted, then assume that $\tau = 1$.

This is of great interest to this thesis, as $p(y|x)$ communicates the degree of certainty with which a given pixel is assigned a class by either considering its entropy, where entropy is higher for less certain estimates, or via the max softmax score, $\mathbf{p}_{\text{max}} = \max[p(y|x)]$ which represents the model's confidence, for which higher values indicate lower uncertainty.

2.1.3 Neural Network Training

For supervised semantic segmentation training, we have a labelled training dataset $\mathbb{D} = \{\mathbf{X}, \mathbf{Y}^*\}$ where an image-label pair can be sampled as the n^{th} element: $(\mathbf{x}, \mathbf{y}^*) = (\mathbf{X}_n, \mathbf{Y}_n^*)$, where $\mathbf{x} \in \mathbb{R}^{3\times H\times W}$ and corresponding pixel-wise labels $\mathbf{y}^* \in \{n \in \mathbb{Z} \mid 1 \leq n \leq K\}^{H\times W}$.

A given segmentation network with parameters θ, is represented by the function $\mathtt{f}_\theta$. The model parameters can be estimated by using Maximum Likelihood Estimation (MLE), such that the likelihood of the observed data is maximised.

The likelihood function is defined as:

$$L(\theta; \mathbb{D}) = \prod_{n=1}^{N} \prod_{i=1}^{H\times W} p(y|x; \theta)$$

In practice, we maximize the log-likelihood, which turns products into sums and is nu-

merically more stable. This is made possible as $\log$ is a monotonically increasing function, and so the optima of g are the same as that of $\log(\mathsf{g})$.

$$\log L(\theta; \mathbb{D}) = \sum_{n=1}^{N} \sum_{i=1}^{H \times W} \log p(y|x; \theta)$$

This is ultimately equivalent to minimising the cross-entropy loss, $\mathcal{H}$, where $\mathcal{H}[p, q] = -\sum q \log(p)$ and q represents one-hot encoded labels.

Therefore the parameters can be estimated by minimising:

$$L_{\mathrm{s}} = \sum_{n=1}^{N} \sum_{i=1}^{H \times W} \bar{y}^* \log p(y|x) \tag{2.4}$$

Where $\bar{y}^* \in \{0, 1\}^K$ is the one-hot encoded label for a pixel location i, $\bar{y}^* = \bar{\mathbf{y}}^*_i$, where $\bar{\mathbf{y}}^* \in \{0, 1\}^{K \times H \times W}$. Using L_{s}, the parameters θ are updated iteratively using a variant of stochastic gradient descent:

$$\theta \leftarrow \theta - \alpha \frac{\partial L_{\mathrm{s}}}{\partial \theta}$$

where α is the learning rate, and $\frac{\partial L_{\mathrm{s}}}{\partial \theta}$ is the gradient of the loss with respect to θ.

An illustration of this method can be seen in Figure 2.1, in which a 2D toy problem is solved using a linear model, in contrast to the higher-dimensional problem of semantic segmentation of RGB images, solved with a significantly more expressive non-linear models.

2.1.4 State-of-the-Art Methods

The primary aspects of a method for maximising semantic segmentation performance is to use as large and diverse a labelled training dataset as possible, in conjunction with a large and expressive deep semantic segmentation network.

There has been considerable innovation in the field of neural network architectures used for semantic segmentation. Typically, an encoder-decoder network architecture is used. An encoder $\mathtt{E}_\theta : \mathbb{R}^{3 \times H \times W} \rightarrow \mathbb{R}^{F \times h \times w}$ embeds an image as a feature map of high latent dimension F and smaller spatial dimensions (h, w). The decoder $\mathtt{D}_\theta : \mathbb{R}^{F \times h \times w} \rightarrow \mathbb{R}^{K \times H \times W}$ then transforms this feature map into per-pixel unnormalised log-probabilities, i.e. the logits. Overall, we have $\mathbf{l} = \mathtt{D}_\theta \circ \mathtt{E}_\theta(\mathbf{x})$, where $\circ$ represents the function composition operator.

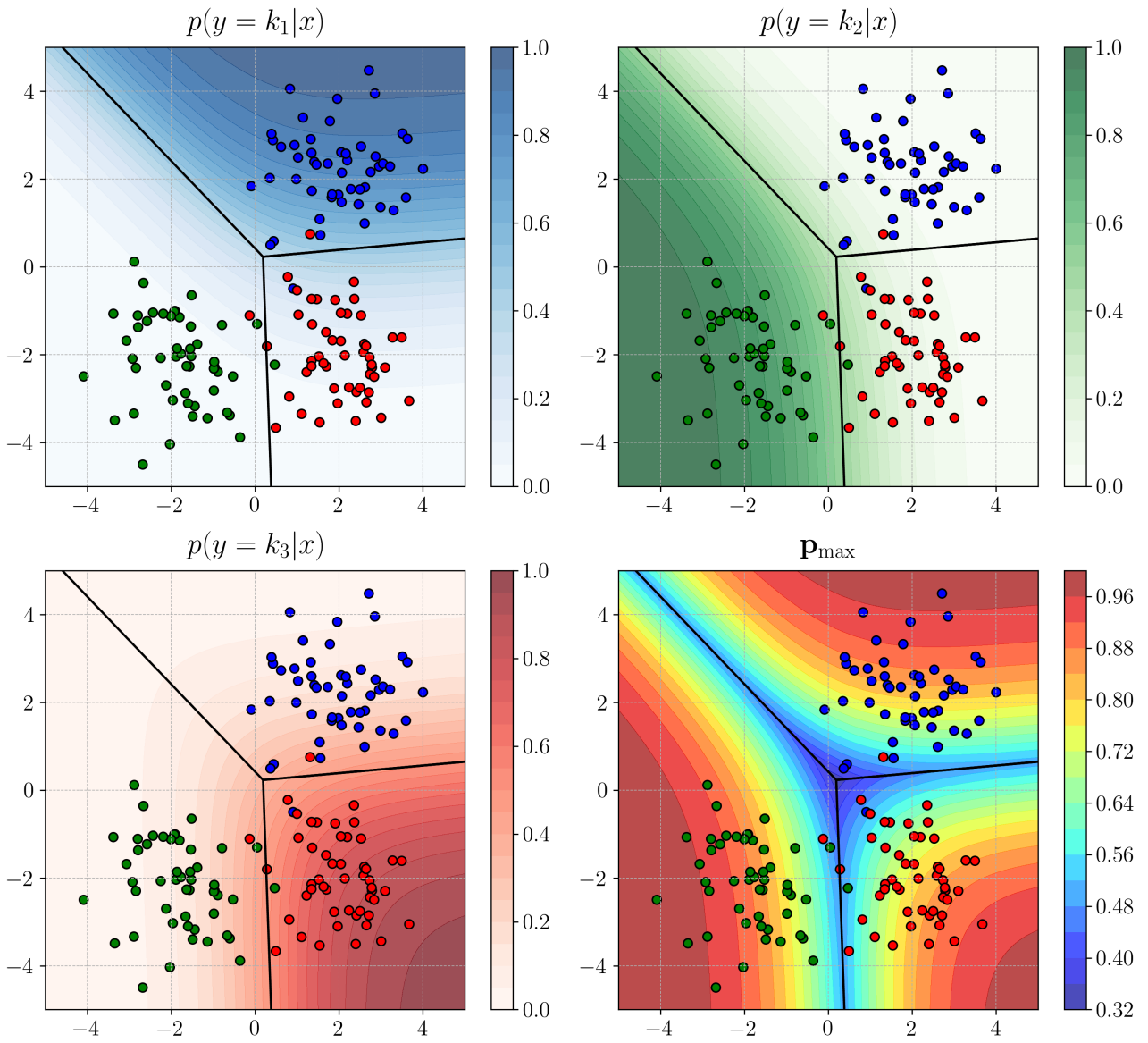


Figure 2.1: A toy problem for which a multi-class logistic regression model is trained with maximum likelihood estimation. Each class probability is plotted separately, along with the decision boundaries and the model confidence, $\mathbf{p}_{\max}$.

The differences in semantic segmentation methods often relates to differences in the design of $\mathtt{E}_\theta$ and $\mathtt{D}_\theta$. The first approaches of this type used fully-convolutional networks [33], which adapted image classification networks by removing the flattening and multilayer perceptron (MLP) layers. More recently, the encoder has been chosen to be the best-performing encoder for supervised, or self-supervised image classification. This is because in image classification, the semantic object of interest could exist across many scales, and be anywhere in the image, therefore the encoder must extract high-level semantic features from across the entirety of the image. This is evidenced by top-performing segmentation networks [33],

[34][1] being initialised with the encoder weights from classification networks trained on the ImageNet.

For many years, the best-performing image classification networks and semantic segmentation networks were based on the ResNet encoder [35] architecture. Currently, SOTA encoders are now variants of the Vision Transformer (ViT) architecture [36], which applies the Transformer architecture [37], which originates from the Natural Language Processing (NLP) community, to computer vision. As for decoders, there are a great number of possible choices [38]–[40], and the decision made in each our presented methods are described in their relevant chapter.

2.2 Overfitting

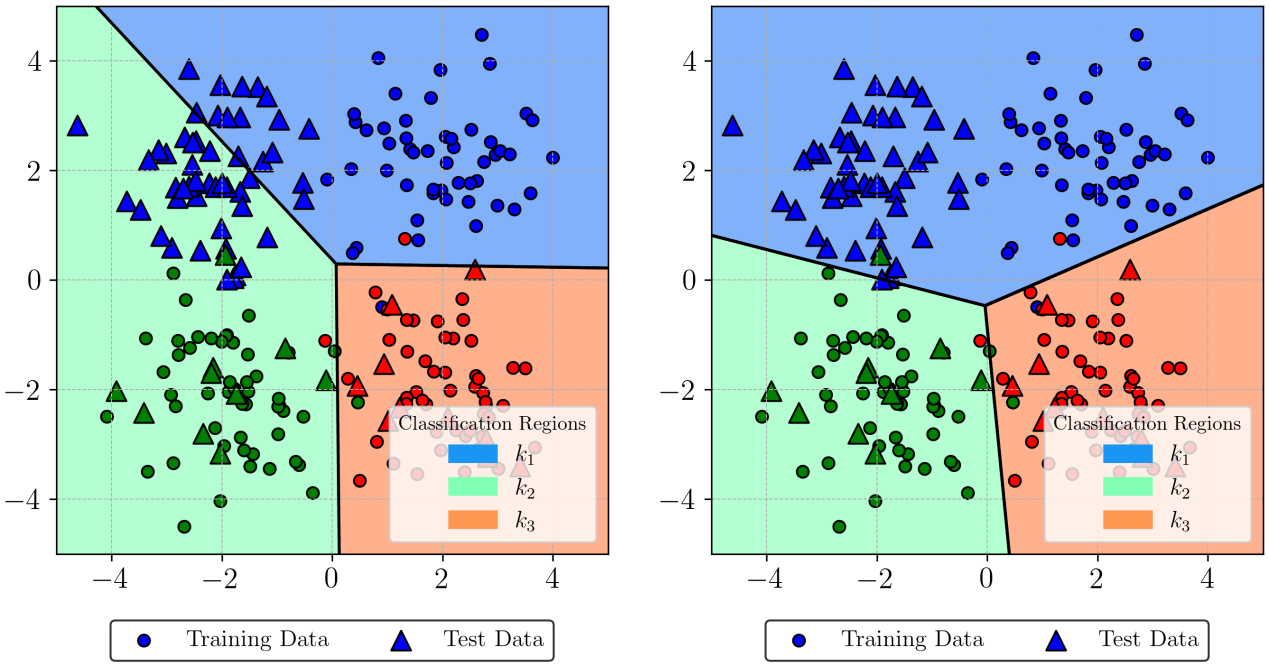
The parameters θ of $\mathtt{f}_\theta$ define a hypothesis space of functions, each of which are a possible solution to a problem. Supervised training on a labelled training dataset via a suitable optimisation technique, e.g. stochastic gradient descent, allows us to search this hypothesis space for the optimal function $\mathtt{f}^*_\theta$ for the task at hand.

Given that the labelled training dataset is inherently a limited representation of the data distribution of all possible images, $\mathtt{f}_\theta$ typically learns a bespoke function which is optimal only for this set of images. This means that even if the test images come from the same data distribution as the training images, the loss and error rate will be higher on these test images, as they do not belong to the training set. In Figure 2.2, the problem of overfitting is illustrated by presenting two models, which are optimised using different sets of data.

As described in Section 2.1.4, $\mathtt{f}_\theta$ is typically a very expressive neural network with a large number of parameters. This means that the hypothesis space of functions is very large, and it has often observed that this readily leads to overfitting, as shown in [41], [42].

A variety of methods can be used to prevent this, such as regularisation and use of validation data for early stopping. It is nonetheless inevitable when we perform empirical risk minimisation (i.e. choosing the model that has the lowest loss on the provided data), that $\mathtt{f}^*_\theta$ will only be optimal for a subset of all possible data.

[1]Both Fully-Convolutional Networks [33] and Mask R-CNN [34] were state-of-the-art (SOTA) methods upon publishing in 2015 and 2018 respectively.



(a) Overfit model

(b) Optimal Model

Figure 2.2: Two models illustrating the problems of overfitting. In (a) a logistic regression model is trained to maximise the likelihood of the training data. In (b) another logistic regression model is trained to maximise the likelihood for all data, demonstrating what an optimal model looks like. In contrast to (b), the model in (a) is optimal only for the training data, and not the test data which belongs to the same class, but is differently distributed. This illustration shows that models trained with maximum likelihood estimation are biased towards only being optimal on the provided data, and thus prone to overfitting.

There are two possible ways in which the loss can be high on unseen data, it can either relate to (1) confident but incorrect class assignment with low entropy and high $\mathbf{p}_{\text{max}}$, or (2) ambiguous class assignment, i.e. high entropy and low $\mathbf{p}_{\text{max}}$. The former is a safety concern, as without labels this is indistinguishable from confident and correct class assignment. However, the latter provides us with information about the limits of the ODD. The relationship between a network's $\mathbf{p}_{\text{max}}$ and accuracy is the topic of model calibration, which is discussed in the next section.

2.3 Calibration of Deep Neural Networks

A neural network's calibration refers to how model confidence $\mathbf{p}_{\text{max}}$ correlates with accuracy. It is desired that the confidence is lower when the model is less accurate, and higher when the model is accurate. This is illustrated in Figure 2.3, where a well-calibrated model and a poorly calibrated model are presented.

It has been shown in many experiments [43]–[45] that, for the task of classification, the calibration of many neural networks is poor for both independent and identically distributed (i.i.d.) and OoD data. Given that most methods treat semantic segmentation as a pixel-wise classification problem, these findings directly carry over to our problem of interest.

Firstly, Section 2.3.1 describes how calibration is formally measured. Then in Section 2.3.2, we will discuss the empirical findings in the literature, and then possible causes in Section 2.3.3. Finally, Section 2.4 discusses miscalibration in the context of robotics.

2.3.1 Calculating Calibration

In order to measure the calibration of a segmentation network $\mathtt{f}_\theta$, we firstly use $\mathtt{f}_\theta$ to segment a set of N images, $\mathbf{Y} = \mathtt{f}_\theta(\mathbf{X})$ where $\mathbf{Y} \in \{1, \dots, K\}^{N \times H \times W}$, $\mathbf{X} \in \mathbb{R}^{N \times 3 \times H \times W}$. We then split the predictions into M bins based on the confidence of the predictions, $\mathbf{P}_{\max} = \max \circ \operatorname{softmax} \circ \mathtt{f}_\theta(\mathbf{X})$, such that the bins $B_{1:M}$ contain predictions with confidence $[0 : \frac{1}{M}, \frac{1}{M} : \frac{2}{M}, \dots, \frac{M-1}{M} : 1]$. The accuracy and confidence is then calculated for each bin:

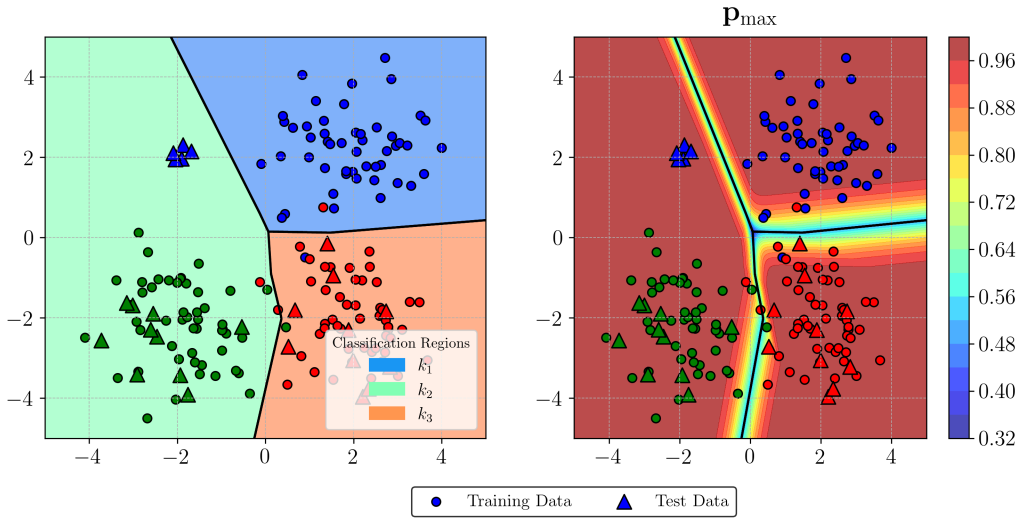
$$\operatorname{Acc}(B_m) = \frac{M}{NHW} \sum_{i=1}^{\frac{NHW}{M}} \mathbb{1}[\mathbf{y}_i = \mathbf{y}_i^*] \tag{2.5}$$

$$\operatorname{Conf}(B_m) = \frac{M}{NHW} \sum_{i=1}^{\frac{NHW}{M}} [\mathbf{p}_{\max}]_i \tag{2.6}$$

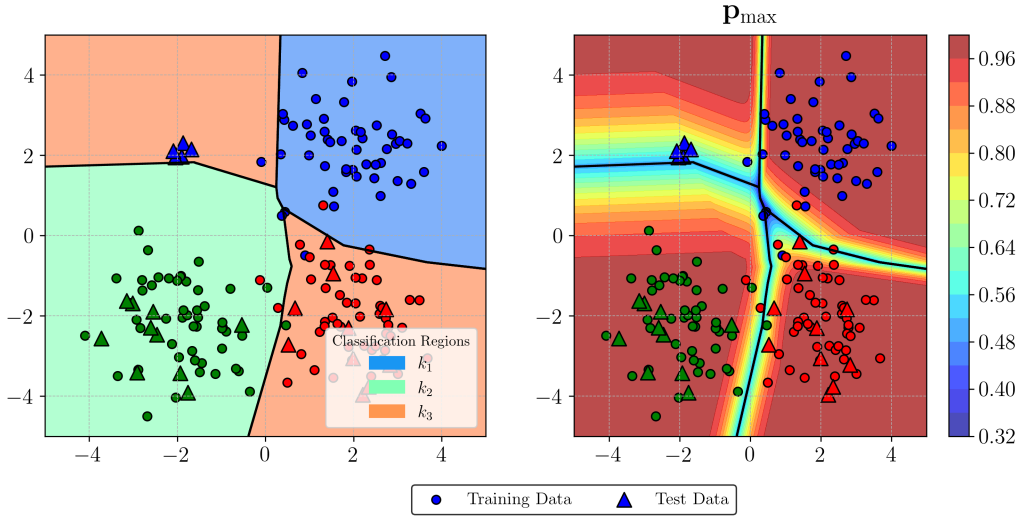
Finally, the model's calibration can by evaluated with the expected calibration error, ECE, as seen in [43], [44]:

$$\mathrm{ECE} = \frac{1}{M} \sum_{m=1}^{M} |\operatorname{Acc}(B_m) - \operatorname{Conf}(B_m)| \tag{2.7}$$

This means that the model calibration will be high when the model's confidences broadly correlates with the model's accuracy. Note that this does not directly measure if the model confidence for a single pixel can be used to estimate the model's accuracy for that pixels, but, in contrast with metrics presented later in this thesis, it takes a broader and more statistical view.



(a) Poorly calibrated model



(b) Well calibrated model

Figure 2.3: In this figure, two MLPs are depicted as being trained to similar accuracies on the training data. In (a), the model is well-calibrated as it misclassifies a cluster of test data in the top left, however it is also has low confidence about this classification, and therefore confidence is predictive of accuracy. However in (b), the model similarly misclassifies the top-left cluster of test data, but it is as confident about this inaccurate classification as it is for accurate classifications.

2.3.2 Observations of Miscalibration

In [43], it is empirically determined that the calibration of modern[2] neural networks is very poor compared with the older smaller networks for classification tasks. Calibration is measured on test datasets that are i.i.d. with respect to the training dataset, and over-confidence is consistently observed for a number of different classification network architectures. This shows that MLE pushes up the probability mass for the training examples, resulting in high

[2]modern as described in 2017

$\mathbf{p}_{\max}$ values, but it also increases the probability of unseen examples resulting in poor calibration. A great number of works (e.g. [43], [46]–[49]) make similar observations when testing on i.i.d. data.

There are also a number of different works that test the calibration of neural networks on OoD test data. In this setting, model accuracy will likely drop significantly between training and testing, and so the question is therefore: to what extent is there also an appropriate drop in model confidence? In a broad set of experiments, [45] shows that the calibration of neural networks becomes poorer as distributional shift increases. This is also not mitigated by tuning the temperature parameter in the softmax function on an i.i.d. validation dataset, such that the mean confidence on this data equals the mean accuracy. This implies either that the network cannot detect OoD instances or that it can, but ignores this information in the calculation of $\mathbf{p}_{\max}$. As a result, ranking pixels by their $\mathbf{p}_{\max}$ value is not an effective method for detecting error due to distributional shift. This observation is corroborated by the findings in [44], which experiments over different classification tasks and a large range of recent neural network architectures (e.g. Vision Transformers [36], ResNet variants [35], non-convolutional MLP Mixers [50]).

For classification, and therefore pixel-wise classification, the fact that $\mathbf{p}_{\max}$ is a poor estimator of the decrease in accuracy due to distributional shift is a key motivator for the methods presented in this thesis.

2.3.3 Causes of Neural Network Miscalibration

For the task of classification (and therefore also for pixel-wise classification), there are a number of possible factors that are suggested to contribute to neural network miscalibration.

Effect of the Cross-Entropy Objective

We can reason about the effects of the cross-entropy objective on model miscalibration in the following way. If a neural network is estimating $p(y|x)$ via the softmax function, then $p(y_j|x) = \frac{\exp(l_j)}{\sum_{k=1}^{K} \exp(l_k)}$. The targets are one-hot encoded labels, and so to fully minimise the cross-entropy, it is required that either $\exp(l_j)$ tends to infinity, or $\sum_{k \in K,\, j \neq k} \exp(l_k)$ tends to zero. This means that for a correctly classified image, the neural network can best reduce

the loss by outputting logits values l_j with very large magnitudes (positive for the logit corresponding to the correct class, and negative for the rest). For incorrectly classified images, the neural network should produce a high entropy $p(y|x)$ to minimise the loss, as the loss increases the more confident an incorrect class assignment is. This can be achieved with logits that can be large or small, as long as they are roughly equal due to the 'translation invariance'[3] of the softmax function, discussed in [51]. This incentive for the model to produce large logit values leads to miscalibration.

As a result of this reasoning, the motivation of [48], [49], [51] is to address the effect of the cross-entropy objective by proposing alternative objective functions. Label smoothing is proposed in [52] where the model is optimised to minimise the cross-entropy between $p(y|x)$ and the one-hot labels, but also to minimise the distance between $p(y|x)$ and the uniform distribution. The effects of this for calibration are described in [48]. The focal loss, originally proposed in [53], can be used in a similar manner, as described in [49]. The latter also discusses how using only the cross-entropy objective leads to weight magnification, and suggests regularisation as a way to mitigate this problem.

Effect of Model Size and Architecture

In [43], large model capacity is cited as a driver of miscalibration when used in conjunction with the cross-entropy loss. Typically, large expressive neural networks will achieve a high accuracy on the training set, and so there will be a particularly large bias in the logit distribution towards very large logit values. [43] reports that while the higher capacity models have a better test accuracy and so generalise better in terms of accuracy, they do not in terms of calibration.

The testing in [43] is performed on in-distribution data exclusively, but [44] investigates the calibration of large models for in-distribution *and* OoD, in addition to using more recent image classification backbones, such as Vision Transformers [36] and MLP Mixer [50].

The findings in [44] corroborate the findings in [43], in that the calibration of higher capacity models on in-distribution data is worse than lower capacity models. It shows that this is also the case is you use temperature scaling to calibrate the mean confidence value on validation data. It, however, shows that higher capacity models perform better in terms of

[3] i.e. $\mathrm{softmax}(x) = \mathrm{softmax}(x + \delta)$, therefore the values are determined by relative, not absolute, magnitude.

calibration on OoD data.

It is worth noting that in [44], all models were pretrained on large diverse datasets, before being fine-tuned in a supervised manner on ImageNet. Therefore, both the training and pretraining expose the models to a large diversity of natural images, which acts as a regulariser. This is in contrast to training or fine-tuning large capacity models on less diverse datasets, e.g. driving datasets, and therefore these reported results are relevant primarily for large-scale image recognition tasks.

Another finding of [44] is that neural network architecture had an effect on model calibration when adjusted for model capacity and the extent of model pretraining. Vision transformers and MLP Mixers tended to be more calibrated than convolutional architectures, such as ResNet-based architectures.

Effect of Activation Function

[54] suggests that another factor which contributes to miscalibration on OoD data is the Rectified Linear Unit (ReLU) activation function. This choice of activation function produces piece-wise linear decision boundaries, which produce polytopes[4] in feature space assigned to each class. They describe how this results in the possibility of yielding features assigned to a known class, but which are infinitely far away from the training data. Therefore, if a OoD data point is represented as very dissimilar to the training data, it is possible for it to be assigned a large logit value for one of the classes, and therefore to be very confidently and incorrectly classified.

Other possible factors

In addition to discussing the cross-entropy objective and large model capacity, [43] also cites batch normalisation [55] and a lack of weight decay as causes of miscalibration.

Batch normalisation is designed to ensure that, during training, each batch of extracted features are similarly distributed. During evaluation, the running mean and variance calculated during training are used to normalise each feature map. By reducing the spread of the features, it is possible that images that have a strong appearance change from the rest of the

[4] A polytope is a N-dimensional extension of a polygon.

training dataset, and thus would benefit from lower confidence, are represented similarly to the prototypical class images, for which a high confidence is appropriate. In this way, it is conceivable that batch normalisation contributes to assigning high confidences to almost all inputs.

[43] describes that increasing weight decay improves model calibration, and yet decreases test accuracy. For this reason, it describes how weight decay is increasingly not used to the detriment of model calibration. In the experiments in both [43] and [49], it is shown that neural networks with smaller weight magnitudes are better calibrated, with the latter performing implicit weight regularisation with the focal loss.

2.4 Conclusion

In this thesis, our objective is to design deep semantic segmentation networks that are suited to being components of robotic systems. Therefore, the neural networks must be able to detect when they are within their ODD, and thus have a sufficiently high level of accuracy for safe autonomous operation.

For this reason, the increasing miscalibration of neural networks due to distributional shifts is a great concern. It also appears that, in many experiments, this is a problem not of scale but of ranking. This means that model confidence is not miscalibrated on OoD data because the mean confidence is inappropriate, but the ranking of pixels by confidence does not relate to the likelihood of accurate classification, and therefore simple scaling fixes such as adjusting the softmax temperature are not sufficient. It is therefore of great importance for us to investigate methods that allow for higher quality estimation of predictive uncertainty, such as seen in the epistemic uncertainty estimation and OoD detection literature. This will be investigated in detail in the next chapter.

In addition to robotic systems needing to understand their limits to facilitate safe operation, it is also important that robotic systems are trusted by the humans they interact with. One way of doing this is to perform accurate uncertainty estimation, and report these estimates to humans. Humans are typically able to interpret probabilities on an intuitive level [56], and, if the model predictions are matched with appropriate levels of model confidence, then this can build trust between humans and the systems they share environments

with, or are being serviced by. Trust is a key ingredient in the adoption of robots in the setting of personal transport due to its safety-critical nature, and so the reporting of high quality uncertainty estimates is an important research direction for this reason alone.

In Chapter 3, we therefore describe the different types of uncertainty, effective methods for their estimation, and how these relate to our mobile robotics setting.

Chapter 3

Uncertainty Estimation in Deep Learning

Contents

3.1 Sources of Uncertainty 31

3.2 Origins of Distributional Uncertainty 32

3.3 Epistemic Uncertainty Estimation 37

3.3.1 Principles for Epistemic Uncertainty Estimation 37

3.3.2 Methods for Epistemic Uncertainty Estimation 39

3.3.3 Discussion 41

3.4 Aleatoric Uncertainty Estimation 41

3.4.1 Learned Loss Attenuation 42

3.4.2 Direct Error Estimation 45

3.4.3 Generative Modelling 47

3.4.4 Test-Time Augmentation 48

3.4.5 Discussion 49

3.5 Out-of-Distribution Detection 50

3.5.1 Pretrained Methods 51

3.5.2 Regularisation-Based Methods 52

3.5.3 Self-Supervised Learning for OoD Detection 53

3.5.4 Proxy Task Methods 54

3.5.5 Deep Generative Models 55
3.5.6 Use of Out-of-Distribution Data 56
3.5.7 Discussion 57
3.6 Conclusion 58

The conclusion of Chapter 2 is that deep segmentation networks are not inherently good at detecting distributional shift, and that this is a problem for mobile robotics applications. For this reason, this chapter presents and discusses the different methods that have been developed to mitigate this problem, spanning the fields of uncertainty estimation and OoD detection.

Firstly, Section 3.1 introduces a dichotomous framework for thinking about uncertainty, by splitting it into aleatoric and epistemic. This is important as much of the uncertainty estimation literature frames its work in these terms. Secondly, Section 3.2 discusses how distributional uncertainty fits into this framework.

Then, for the rest of the chapter, the themes and methods for epistemic uncertainty estimation, aleatoric uncertainty estimation, and OoD detection are presented. Each are discussed both for their broad suitability for mobile robotics, and with reference to the discussion in Section 3.2.

Finally, in Section 3.6, the most promising themes are discussed in more detail. These themes are key influences on the methods presented in Chapter 5, Chapter 6, and Chapter 7.

3.1 Sources of Uncertainty

Error in computer vision tasks is often presented as originating from two sources of uncertainty: aleatoric and epistemic. Epistemic uncertainty (related to the Ancient Greek, *episteme*, for knowledge), is uncertainty that is theoretically under the control of the modeller (i.e. the person designing the model) and can be reduced by decisions in the model design. Under this definition, if only epistemic uncertainty exists, then the following is true: there theoretically exists a model architecture defined by $\mathtt{f}^*$, which can be trained by a optimal dataset and optimiser to yield parameters θ^*, such that the trained model $\mathtt{f}^*_\theta$[1] reduces the test error

[1]Strictly, this should be $\mathtt{f}^*_{\theta^*}$, as both the architecture and the parameters are optimal, however $\mathtt{f}^*_\theta$ is used for simplicity of presentation.

to zero. For this reason, epistemic uncertainty is due to the suboptimality of the model and its parameters rather than the input data, and can be reduced by an improved training procedure, such as a more diverse training dataset, better network architecture, better optimiser etc.

In contrast, aleatoric uncertainty is uncertainty than cannot be reduced by the modeller. Etymologically, aleatoric is related to the Latin, *alea*, for dice, bringing up the idea that aleatoric uncertainty causes error due to the inherent randomness of the observations, rather than imperfections in the model.

This means that no combination of model, dataset, optimizer can reduce the error on the test dataset to zero. In outdoor robotics, examples of sources of aleatoric uncertainty include: noise from the sensor, environmental factors such as fog or darkness, lack of resolution of objects in the distance, sensor adherents such as rain, snow, or smudges on a camera lens.

The distinction between the types of uncertainty is important for two reasons. Firstly, knowledge of the uncertainty type should be used to design the method for its estimation, e.g. if uncertainty primarily originates in the data, then a method should focus its attention on the data and not the model parameters. Secondly, since epistemic uncertainty is reducible, estimating it for a dataset of unlabelled images gives the modeller information about which images to use to improve the model's accuracy, for example by labelling them and then training on them. Therefore, estimating epistemic and aleatoric uncertainty separately can be useful in an active learning setting to find training examples that are maximally informative, i.e. those that have high epistemic uncertainty and low aleatoric uncertainty. The latter is important as images with high aleatoric uncertainty are, by definition, uninformative, and thus cannot improve the model's accuracy.

3.2 Origins of Distributional Uncertainty

We have presented a dichotomous framework which defines uncertainty as being either reducible or irreducible by the modeller. As described in Chapter 1, we are interested in estimating distributional uncertainty, and so it is worth determining how distributional uncertainty fits into this framework.

In the context of semantic segmentation, distributional uncertainty can derive from two

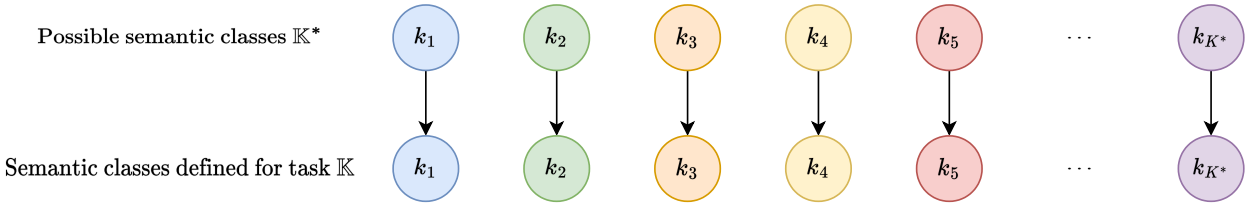
components: (1) uncertainty due to an instance of an unknown class, i.e. a class not defined in the labelled training dataset (2) uncertainty due to an instance of known class, but with an appearance distinct from the instances of the same class in the labelled training dataset, often called covariate shift [57]–[59].

Theoretically, it can be concluded that distributional uncertainty is epistemic. This is because it is theoretically possible to design a model that can segment all possible presentations of all possible semantic classes, represented in Figure 3.1a. A dataset could be collected that contains diverse presentations of each of the finite number of semantic concepts[2]. Then, a sufficiently expressive neural network architecture could be defined with an appropriate optimizer, and this model could be trained such that the segmentation error on any possible test dataset would be zero. Note that this is true with the proviso that each semantic object needs to be sufficiently well captured in the test images that no aleatoric uncertainty is present. To some degree, this type of large-scale training has been attempted in works such as DINOv2 [31], CLIP [30] and Segment Anything [60].

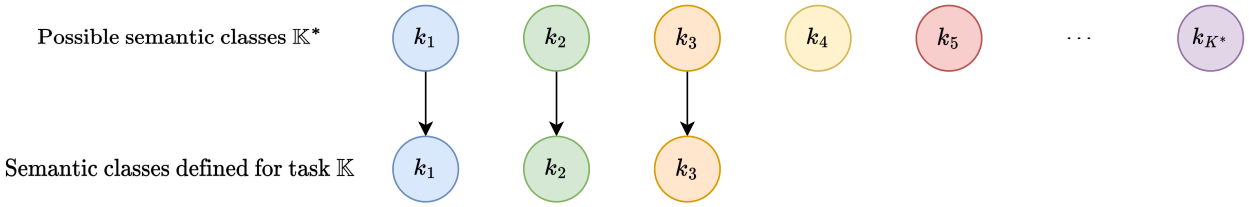
However, for the practical setting introduced in Chapter 1, we can say that distributional uncertainty cannot be fully reduced by the modeller. Firstly, this is because large-scale vision models in question cannot classify or segment any test dataset with zero error. Secondly, the recognition of every possible semantic class requires a model capacity that is far larger than needed. For example, a 10 class classification problem CIFAR-10 can be solved with $92.74\,\%$ top-1 accuracy by kMobileNet V3 Large 16ch [61] with $400\,\mathrm{k}$ parameters, while a 1000 class classification problem ImageNet-1k requires Florence-CoSwin-H [62] with $893\,\mathrm{M}$ parameters to achieve a $90.05\,\%$ top-1 accuracy, with the latter incurring significantly more latency and GPU memory usage. In addition, the cost of this large-scale training – in terms of cost of GPUs or cloud credits – is prohibitively expensive for many settings. For example, CLIP [30] was trained on 256 NVIDIA V100 GPUs for two weeks, equivalent to costing in the order of $100,000s. This means that even if these types of model did allow us to fully reduce distributional uncertainty, there are currently significant challenges with both training and deploying them.

In summary, this shows that distributional uncertainty is theoretically epistemic, how-

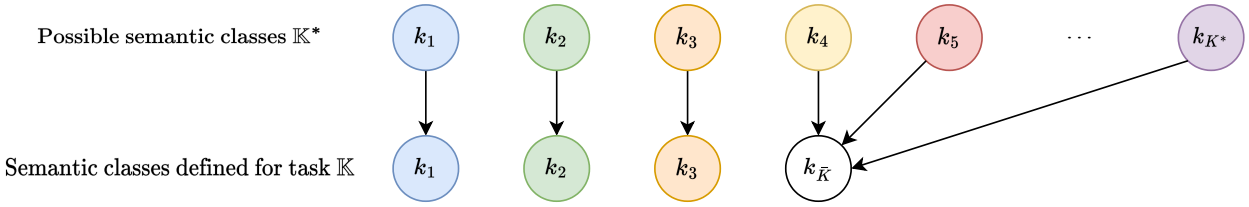
[2]If we consider language to be finite and that semantics are ultimately described in language, we can convince ourselves that the number of possible semantic concepts are also finite.



(a) Modelling all semantic classes, thus any error is due to epistemic uncertainty.



(b) Modelling only the semantic classes of interest, thus errors due to classes $k_{4:K^*}$ relate to *aleatoric* uncertainty.



(c) Modelling only the semantic classes of interest, thus errors due to classes $k_{4:K^*}$ relate to *epistemic* uncertainty.

Figure 3.1: Suppose the set of all semantic classes is $\mathbb{K}^* = k_1, \ldots, k_{K^*}$, but we define a segmentation task in which only the set, $\mathbb{K} = \{k_1, k_2, k_3\}$ are of interest. This figure represents the possible ways in which we can approach this problem: (a) seeks to segment all classes, (b) simply ignores the classes not of interest, while (c) adds an additional unknown class $\bar{K}$ to represent all irrelevant classes. As described this impacts the definition of component (1) of distributional uncertainty as aleatoric or epistemic.

ever a tractable implementation would not reduce the test error to zero. This raises the question: Factoring in the current state-of-the-art in computer vision and the constraints of mobile robotics, *is it the case that distributional uncertainty is best treated as epistemic?*

In the following sub-sections, the described components of distributional uncertainty will be discussed in detail. This will subsequently be used to argue that aleatoric methods can also be used to effectively model distributional uncertainty.

Uncertainty due to unknown classes

It is firstly worth considering the nature of the models and data in use. A labelled training dataset for semantic segmentation defines a finite set of known semantic classes $\mathbb{K} =$

$\{k_1, k_2, \ldots, k_K\}$, and pixels in the dataset that do not belong to those classes or remain unlabelled due to ambiguity are assigned to an undefined class $k_{\bar{K}}$, often referred to as `void`.

Typically in benchmarks such as [25], [63], segmentation networks are evaluated with the test dataset on their ability to segment the known classes only, i.e. the segmentations of the pixels labelled as $k_{\bar{K}}$ are ignored. For this reason, it is common for a segmentation network to produce K logits, one for each of the semantic classes defined by the training dataset. In this setting, a segmentation network trained on these classes therefore estimates which of the finite set of known classes a given pixel belongs to, i.e. for each pixel, the model outputs $p(y|x) = [p(y = k_1|x), \ldots p(y = k_K|x)] \in [0, 1]^K$, introduced in Section 2.1.2.

In robotics, we need to consider the entirety of the image, and so these void pixels will contribute to the distributional uncertainty of the test dataset. Under this previous definition, it is not possible for the segmentation network to correctly segment pixels not in $\mathbb{K}$. For this model definition, this means that component (1) of distributional uncertainty is aleatoric, see Figure 3.1b.

However, we could also design a model that produces $K + 1$ logits, where for each pixel $p(y|x) = [p(y = k_1|x), \ldots, p(y = k_K|x), p(y = k_{\bar{K}}|x)] \in [0, 1]^{K+1}$. In this case, the final class $\bar{K}$ represents any semantic class that is not a known and defined semantic class, see Figure 3.1c. In this case, and given that labelled pixels for this class exist in typical semantic segmentation datasets as `void`, component (1) of distributional uncertainty is epistemic, as more training data of pixels belonging to class $k_{\bar{K}}$ will improve segmentation accuracy.

This therefore suggests that this component of distributional uncertainty might be considered as either epistemic or aleatoric depending on the formulation used.

Uncertainty due to intra-class variation

The second component of distributional uncertainty is due to the intra-class visual dissimilarity between the labelled training distribution (i.e. the source domain) and the distributionally-shifted test distribution (i.e. the target domain). On the surface, this is clearly epistemic as adding more diversity to the dataset of existing classes will improve segmentation performance.

However, much like component (1), this is also a matter of definition, as semantic classes



(a) Cityscapes



(b) SAX New Forest

Figure 3.2: Images from two datasets demonstrating the subjectivity of semantic classes. Suppose (a) is an example of an image from a labelled training dataset that defines the semantic classes. If we consider the `road` class, how does this semantic concept relate to (b)? Is `road` defined by the type of road surface, or by the notion of traversability, or by the rules of the road in this specific geographic location? And therefore which of regions of (b) is it appropriate to call `road`? The image in (b) is from our dataset presented later in Chapter 4.

are inherently subjective (illustrated in Figure 3.2). For this reason, it is down to the judgement of the modeller to decide whether an appearance change constitutes the creation of a new class, or whether this new appearance change is within the definition of the original class. For example in Figure 3.2b, do all pixels from traversable regions relate to `road`, or is it only those from the paved sections of the road? If the former is true, then the distributional uncertainty associated with those pixels is epistemic as more training images of unpaved roads would improve segmentation accuracy. However, if the latter is true and ground-truth label for the pixels to the left of the paved road is `unpaved_road` and not `road`, then distributional uncertainty associated with these pixels is aleatoric. This is because the model cannot segment pixels as `unpaved_road`, and therefore this uncertainty cannot be reduced.

Therefore, much like the previous component, whether distributional uncertainty due to intra-class variation is aleatoric or epistemic is a matter of perspective.

Summary

Considering both components of our defined components, this analysis has shown us whether distributional uncertainty is epistemic or aleatoric is conditioned on the modeller's decisions rather than being an intrinsic property. Based on this understanding, this thesis also considers aleatoric uncertainty estimation methods unlike many other works. In fact, given the computational benefits of aleatoric uncertainty estimation (detailed in Section 3.4.5), these methods are of particular interest.

3.3 Epistemic Uncertainty Estimation

3.3.1 Principles for Epistemic Uncertainty Estimation

As epistemic uncertainty is intrinsic to the model, and not the test data, epistemic uncertainty estimation methods seek to capture how the model parameters relate to model performance for a given input. One framing of this is Bayesian, whereby instead of treating the model as deterministic, the model parameters are instead treated as random variables, yielding a Bayesian Neural Network (BNN), such as in [64], [65].

For this, the weight posterior distribution $p(\theta|\mathbb{D})$ is considered where θ represents the model parameters, and $\mathbb{D}$ the training dataset. Bayes' Rule tells us that the weight posterior can be computed as:

$$p(\theta|\mathbb{D}) = \frac{p(\mathbb{D}|\theta)p(\theta)}{p(\mathbb{D})} = \frac{p(\mathbb{D}|\theta)p(\theta)}{\int_\theta p(\mathbb{D}|\theta)p(\theta)d\theta}$$

In this setting, the likelihood $p(\mathbb{D}|\theta)$ is a measure of how well a given set of parameters fit the data. The prior $p(\theta)$ represents our beliefs about the weights before any data is observed. $p(\mathbb{D})$ is the model evidence and represents the likelihood of the observed data under all possible parameterisations of the model, i.e. it is an aggregation of likelihoods $p(\mathbb{D}|\theta)$ evaluated for all possible values of θ. Each of these are used to calculate the weight posterior $p(\theta|\mathbb{D})$, which represents which weight values best explain the observed data.

The predictive distribution is calculated by evaluating the model predictions $p(y|x,\theta)$ for all possible weight values:

$$p(y|x,\mathbb{D}) = \int_\theta p(y|x,\theta)p(\theta|\mathbb{D})d\theta \tag{3.1}$$

Epistemic uncertainty is then commonly calculated as either the Predictive Entropy (PE) or Mutual Information (MI), as described in [66]:

$$\mathrm{PE} = \mathcal{H}[p(y|x,\mathbb{D})] = -\sum_{k=1}^{K} p(y=k|x,\mathbb{D}) \log p(y=k|x,\mathbb{D}) \tag{3.2}$$

$$\text{MI} = \mathcal{H}[p(y|x,\mathbb{D})] + \mathbb{E}_{p(\theta|\mathbb{D})}\left[-\mathcal{H}[p(y|x,\theta)]\right] \tag{3.3}$$

$$= \text{PE} + \mathbb{E}_{p(\theta|\mathbb{D})}\left[\sum_{k=1}^{K} p(y=k|x,\theta)\log p(y=k|x,\theta)\right] \tag{3.4}$$

PE is the entropy of the predictive distribution, and represents the total predictive uncertainty. PE is high when either: (1) the posterior distribution is broad, causing a large spread in $p(y=k|x,\mathbb{D})$, or (2) the posterior distribution is sharp, however for each possible parameterisation, the prediction $p(y|x,\theta)$ is of low confidence. As an example, for a binary classification problem, suppose we sample parameters from $p(\theta|\mathbb{D})$ as $[\theta^1, \theta^2, \ldots, \theta^\infty]$.

The scenario (1) relates to sampling a wide range of parameter values yielding variable predictions, such as:

$$[p(y|x,\theta^1), p(y|x,\theta^2), p(y|x,\theta^3), \ldots, p(y|x,\theta^\infty)] = [(1,0),(0,1),(0,1),\ldots,(1,0)]$$

which give a predictive distribution of $p(y|x,\mathbb{D}) = (0.5, 0.5)$, and thus a $\text{PE} = 0.3$.

The scenario (2) relates to sampling parameter values that are very similar, which give very consistent predictions, but in this case these predictions are consistently unconfident:

$$[p(y|x,\theta^1), p(y|x,\theta^2), p(y|x,\theta^3), \ldots, p(y|x,\theta^\infty)] = [(0.5,0.5),(0.5,0.5),(0.5,0.5),\ldots,(0.5,0.5)]$$

which also give a predictive distribution of $p(y|x,\mathbb{D}) = (0.5, 0.5)$, and thus a $\text{PE} = 0.3$.

In contrast, MI focusses specifically on the uncertainty in the weight posterior, by calculating the difference between the total predictive uncertainty PE and the uncertainty related to average entropy of each specific model parameterisation $\mathbb{E}_{p(\theta|\mathbb{D})}\left[\mathcal{H}[p(y|x,\theta)]\right]$. In scenario (1), $\text{MI} = \text{PE}$, as each of $\mathcal{H}[p(y|x,\theta)] = 0$, meaning that the uncertainty due to the weight posterior makes up the entire predictive uncertainty. However in scenario (2), $\text{MI} = 0$, as PE and $\mathcal{H}[p(y|x,\theta)]$ are the same for each θ, and thus the uncertainty due to the weight posterior does not contribute to the total predictive uncertainty. MI is therefore more commonly used when we want to specifically measure the uncertainty from the parameter distribution, such as in active learning [66].

Unfortunately, evaluating the weight posterior with exact Bayesian analysis is not tractable due to the required integration over all possible weight values in the model evidence, and

the very high-dimensionality of the model parameter space, θ. A number of different approximations are made to solve this, and these are discussed in the next section.

3.3.2 Methods for Epistemic Uncertainty Estimation

One such approximation, named Monte Carlo Dropout (MCD) [64], approximates the weight posterior $p(\theta|\mathbb{D})$ with $q(\theta)$ using dropout [41]. Here, a network is trained with dropout layers yielding N parameters, $\theta = \{\theta_1, \ldots, \theta_N\}$, and then at inference time, sets of weights are sampled from the weight posterior, such that the m^{th} sample is given by $\hat{\theta}^m = \{\hat{\theta}_1^m, \ldots, \hat{\theta}_N^m\}$, where $\hat{\theta}_i^m = \theta_i * z_i$ and $z_i \sim \text{Bernoulli}(p_{\text{drop}})$. If M sets of parameters are sampled, the predictive distribution is given by:

$$p(y|x, \mathbb{D}) = \int_\theta p(y|x, \theta)p(\theta|\mathbb{D})d\theta \approx \int_\theta p(y|x, \theta)q(\theta)d\theta \approx \frac{1}{M}\sum_{m=1}^{M} p(y|x, \hat{\theta}^m) \tag{3.5}$$

And then PE and MI can be redefined for M samples as:

$$\text{PE} = -\sum_{k=1}^{K}\left(\frac{1}{M}\sum_{m=1}^{M} p(y|x, \hat{\theta}^m)\right)\log\left(\frac{1}{M}\sum_{m=1}^{M} p(y|x, \hat{\theta}^m)\right) \tag{3.6}$$

$$\text{MI} = \text{PE} + \frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K} p(y = k|x, \hat{\theta}^m)\log p(y = k|x, \hat{\theta}^m) \tag{3.7}$$

Another approach, Bayes-by-Backprop, instead approximates the weight posterior with a distribution that is itself parameterised, i.e. $q(\theta, \alpha)$, where α are the parameters defining that shape of $q(\cdot)$. The general form for how to use variational inference for neural networks is to fit α by minimising the the variational free energy, $\mathcal{F}(\mathbb{D}, \alpha)$, such that $\alpha^* = \arg\min_\alpha \mathcal{F}(\mathbb{D}, \alpha)$ and:

$$\mathcal{F}(\mathbb{D}, \alpha) = \text{KL}[q(\theta|\alpha), p(\theta)] - \int q(\theta|\alpha)\log p(\mathbb{D}|\theta)d\theta \tag{3.8}$$

The first term represents the distance between the variational distribution $q(\theta|\alpha)$ and the true prior $p(\theta)$, while the second term represents how well the variational distribution explains the observed data.

Bayes-by-Backprop is approximates this expression with:

$$\mathcal{F}(\mathbb{D}, \alpha) \approx \sum_{m=1}^{M} \log q(\hat{\theta}^m|\alpha) - \log p(\hat{\theta}^m) - \log p(\mathbb{D}|\hat{\theta}^m) \tag{3.9}$$

Where M is the number of Monte Carlo samples, and $\hat{\theta}^m$ is a set of model parameters sampled from $q(\hat{\theta}^m|\alpha)$. In this way, this method is able to approximate the weight posterior $p(\theta|\mathbb{D})$ with $q(\hat{\theta}^m|\alpha)$ and calculate uncertainty as the variance of the predictive distribution for each of the sampled $\hat{\theta}^m$. In [65], the authors experiment with both Gaussian and scale mixture distributions as the priors for Gaussian distributed posteriors.

This method tends to be challenging to train to a high accuracy and tends to increase training time substantially. This is due to the need to sample sets of parameters and so there is a trade-off between diversity of sampled weights $\hat{\theta}^m$ and how many forward passes are required to compute a batch, as discussed in [67]. In order to see as many possible weight perturbations as possible and to reduce the variance of the gradient updates, we would want to sample a different $\hat{\theta}^m$ for each batch element. This, however, is extremely computationally intensive. Additional methods have been devised to try to deal with this problem, e.g. Flipout [67].

In contrast to methods that draw on a Bayesian underpinning, ensembles have been used to produce very high-quality epistemic uncertainty estimates [68]. Instead of sampling from $q(\cdot)$, samples from the weight distribution $\{\hat{\theta}^1, \ldots, \hat{\theta}^M\}$ are obtained by training M neural networks independently on the same training data. In order to obtain diversity in the neural networks, each member of the ensemble is initialised with a different set of random weights and the training data is presented in a different order. Not only is the training in this method straightforward, but it also often leads to higher model accuracy.

Similar to the approaches using BNNs, the epistemic uncertainty can be calculated using PE and MI. Despite the lack of direct Bayesian interpretation, this approach has been very successful at producing accurate epistemic uncertainty estimates.

3.3.3 Discussion

These approaches have been shown to produce high-quality uncertainty estimates (especially MCD and deep ensembles). They do however have a major drawback relating to their computational efficiency at training and, most importantly for this thesis, at testing.

In order to obtain an uncertainty estimate for a single image, we require M forward passes, relating to the M different parameterisations of the neural network. Therefore in order to run this on a robotic platform, either the memory requirement increases by a factor of M to maintain latency, or the latency increases by a factor of M. It is typically reported that an appropriate value for M is 5 or more. Therefore the former is often not possible due to limited hardware deployed on edge devices, e.g. a NVIDIA Jetson Nano Developer Kit has only $4\,\mathrm{GB}$ of GPU memory which cannot support 5 high accuracy segmentation networks in parallel. The latter significantly affects safety and usefulness, as a $5\times$ increase in latency would typically make a segmentation network unsafe for a number of different tasks, e.g. path planning in dynamic environments.

In order to mitigate this, methods such as [69], [70], distil the uncertainty estimates of MCD networks and ensembles into a single deterministic neural network. In both cases, measures of the variance are used as targets, which a neural network is trained to approximate in a single forward pass. There is naturally a corresponding drop in performance, but makes the discussed epistemic approaches feasible for mobile robotics settings.

3.4 Aleatoric Uncertainty Estimation

The presence of aleatoric uncertainty means that it is not possible for the modeller to design a model that fully reduces the error on the test dataset, because aleatoric uncertainty is inherent to the data. For this reason, methods that seek to estimate aleatoric uncertainty focus on the relationship between appearance of image regions and error, rather than the model parameters and error.

Suppose we are considering a classification task for which we have ground-truth labelling, a neural network can be trained to perform both classification and aleatoric uncertainty estimation as follows. As the cross-entropy classification loss decreases, the error

rate on the training and validation data decreases, and thus the epistemic uncertainty has decreased. This continues to occur until the error rate plateaus at a non-zero value, and the epistemic uncertainty cannot be further reduced. It can be assumed that for the images that are still incorrect, there is insufficient discriminative information contained in them for the model to be accurate[3]. This might be because of class ambiguity, sensor noise, fog, darkness, sensor adherents, sensor resolution etc. It is therefore appropriate for a method to be able to detect these images, and to output a high aleatoric uncertainty for them.

3.4.1 Learned Loss Attenuation

One way of doing this is to have the task model output both a task estimate, e.g. a classification estimate, and also an uncertainty estimate. This is often achieved by placing a prior distribution over the networks outputs, and using a set of methods named *learned loss attenuation* [71], but also presented previously in [72]. For classification tasks, this is slightly awkward to think about, and so for readability, this type of method will first be explained for a regression task, e.g. depth estimation. This sub-section will then subsequently describe the extension to classification and segmentation tasks.

We define a model g_θ which produces both a regression estimate and uncertainty estimate for a given image $\mathbf{x}$: $[\mu,\ \sigma^2] = \mathrm{g}_\theta(\mathbf{x})$. For each image, there exists a ground-truth regression label $\mathbf{y}^*$, such that error can be measured as $\|\mu - \mathbf{y}^*\|_2$. For this formulation, the model can be trained to perform regression and uncertainty estimation by assuming a Gaussian likelihood. Supposing we have defined a pixel-wise regression task, we can calculate the loss for each pixel location i as:

$$L_i^{\text{MLE}} = \frac{1}{2}\sum_i \frac{\|\mu_i - \mathbf{y}_i^*\|_2^2}{\sigma_i^2} + \frac{1}{2}\log(\sigma_i^2) \tag{3.10}$$

The model can minimise this loss function in two ways: (1) either it can reduce the regression error $\|\mu_i - \mathbf{y}_i^*\|_2$ (minimising the numerator) and estimate a low variance σ_i^2, minimising $\log(\sigma_i^2)$, or (2) it can estimate a large value for σ_i^2 (maximising the denominator). The former of the two options trains the model to solve the regression task, while the latter trains the

[3]Note that this is an assumption, and it could also be attributed to sub-optimal model architecture, optimizer, training dataset, etc.

model to output high uncertainty values when the model cannot sufficiently reduce the error for a pixel. The $\frac{1}{2}\log(\sigma_i^2)$ term prevents the loss from being minimised by estimating a large variance for all pixels, as this term is minimised by $\sigma_i \to 0$.

The formulation above is extended to classification and segmentation problems in [71] by placing a Gaussian distribution over the logits $\mathbf{l}$, before obtaining the categorical distribution $\mathbf{p}$ with the softmax function. As before, we define a neural network to produce explicit uncertainty estimates: $[\mathbf{l},\ \sigma^2] = \mathbf{g}_\theta(\mathbf{x})$, with logits and covariances for each pixel i as $\mathbf{l}_i \in \mathbb{R}^K$ and $\sigma_i^2 \in \mathbb{R}^{K\times K}$ respectively, where the latter is diagonal. The logit distribution is given by:

$$\tilde{\mathbf{l}}_i \sim \mathcal{N}(\mathbf{l}_i, \sigma_i^2) \tag{3.11}$$

Where now $\mathbf{p}_i = \mathrm{softmax}(\tilde{\mathbf{l}}_i)$. The loss function again takes the form of the negative log-likelihood:

$$L_i^{\text{MLE}} = -\log \mathrm{softmax}(\tilde{\mathbf{l}}_{i,k=\mathbf{y}^*}) \tag{3.12}$$

Unfortunately, there is no analytical solution to combine the softmax function and Gaussian distribution, and so each $\tilde{\mathbf{l}}_i$ needs to be sampled as $\tilde{\mathbf{l}}_{i,m,k} = \mathbf{l}_{i,k} + \epsilon_{m,k}$ where $\epsilon_{m,k} \sim \mathcal{N}(0, \sigma_k^2)$. Incorporating the sampling and expanding for clarity gives us:

$$L_i^{\text{MLE}} = \frac{1}{M}\sum_{m=1}^{M}\left(-\tilde{\mathbf{l}}_{i,m,k=\mathbf{y}^*} + \log\sum_{k=1}^{K}\exp(\tilde{\mathbf{l}}_{i,m,k})\right) \approx \frac{1}{M}\sum_{m=1}^{M}\left(\max_k[\tilde{\mathbf{l}}_{i,m,k}] - \tilde{\mathbf{l}}_{i,m,k=\mathbf{y}^*}\right) \tag{3.13}$$

The interpretation of this loss is aided by recalling that the $\log\text{-}\sum\text{-}\exp$ is a smooth approximation to the $\max$ function, as shown in Equation (3.13). The first way in which the model can reduce the loss is by ensuring that the sampled logit which corresponds to the correct class is the largest. This can be done by expressing a low corresponding value of σ_k^2 and large deterministic logit value $\mathbf{l}_k = \mathbf{y}^*$. This allows the model to learn the classification problem.

Similar to the regression case, the model also learns to estimate uncertainty by using its estimate σ^2 to reduce the loss for misclassified images. A carefully chosen increase to σ^2 can increase the noise of the sampled logits, which decreases the confidence of the model prediction. On average, this can reduce the difference between $\max_k[\tilde{\mathbf{l}}_{i,m,k}]$ and $\tilde{\mathbf{l}}_{i,m,k=\mathbf{y}^*}$, and thus minimise the negative log-likelihood in Equation (3.12).

Note that this is similar to simply performing MLE without a Gaussian distribution over the logits. The key difference in this formulation is that the model learns to reduce the loss via the explicit estimation of uncertainty σ^2, whereas simpler MLE without the Gaussian logits trains a model to reduce the loss via implicitly representing uncertainty as high entropy categorical distributions[4].

The formulation in [73] is similar to the classification case for a binary segmentation problem, but uses an approximation instead of resorting to Monte Carlo sampling from the Gaussian across the softmax function. It states that the Monte Carlo sampling required 10^4 samples to converge to the true distribution, and so this approximation is a great help during training.

In [74], aleatoric uncertainty estimates are learned alongside learning surface normal estimation by using learned loss attenuation with the von Mises-Fisher distribution, a spherical equivalent to the Gaussian distribution. A 3D super-resolution task is defined in [75], and it trains a neural network with learned loss attenuation to also estimate aleatoric uncertainty with a Gaussian regression formulation.

The methods discussed so far have used ground-truth supervision in order to measure where error has occurred. In contrast, [76], [77] define self-supervised tasks and probabilistic self-supervised objectives to solve them. A neural network is trained in [76] to estimate the 3D geometry of objects from a single image. The self-supervised task involves learning to estimate depth and viewpoint via supervision from egomotion and depth from Structure from Motion (SfM) applied to videos. The estimation of aleatoric uncertainty is motivated by the need to reduce the noise in the self-supervised task, which is achieved by attenuating the loss for uncertain depth or viewpoint estimates.

In [77], a neural network is trained to extract geometric features by defining a self-supervised data augmentation task. Similar to [76], it learns aleatoric uncertainty estimation in order to mitigate the gradient contributions of image regions where a stable match is not possible, i.e. where there is insufficient discriminative information to describe a region, such as feature-less surfaces. For this task, it also uses the learned loss attenuation formulation,

[4]Note that this is not true for typical regression problems with a mean squared error loss function. In this case, if the model thinks its estimate is erroneous, there is no recourse for mitigating the loss. In contrast, classification problems always implicitly model the uncertainty with a categorical distribution, which results in learning to mitigate large loss values due to misclassification through high entropy estimates.

where, in this case, the network output parameterises an exponential distribution.

The learned loss attenuation formulation is built upon in [78] and it introduces one main idea. It proposes a framework that parameterises each set of intermediate features throughout the network as a distribution. In this way, not only does the network output its uncertainty in its task estimate, but it also outputs its uncertainty for each feature. The motivation for this is that when a model only outputs its uncertainty in its output, it must in someway encode the uncertainty in its features into the features themselves. However, this method seeks to disentangle this information.

To explain this by way of an example, suppose a cat/dog classification model is given an image of a monkey. In order to express uncertainty at the output, the model's features must describe the monkey in sufficient detail that its features are distinct from that of cats and dogs. However with this framework, the features can encode the task-specific information about the image, i.e. how cat-like or dog-like the monkey looks, while the uncertainty over the features can tell the model to ultimately ignore this information as it looks like neither.

It does this by using variational expectation propagation, in which a neural network layer's outputs parameterise a chosen distribution $q(\cdot)$. This is done by using moment matching (a.k.a. assumed density filtering), as shown below. This means that the distribution at the output of ith layer of neural network $\mathtt{f}_\theta$ with input $z^{(i-1)} \sim q(z^{(i-1)})$ is parameterised by:

$$\mu^{(i)} = \mathbb{E}_{q(z^{(i-1)})}[\mathtt{f}_\theta^{(i)}(z^{(i-1)})],\ \sigma^{(i)} = \mathbb{V}_{q(z^{(i-1)})}[\mathtt{f}_\theta^{(i)}(z^{(i-1)})] \tag{3.14}$$

Which for a linear layer $\mathtt{f}_\theta^{(i)}(z^{(i-1)}) = Wz^{(i-1)} + b$, is given by $\mathbb{E}[\mathtt{f}_\theta^{(i)}] = W\mu^{(i-1)} + b$, $\mathbb{V}[\mathtt{f}_\theta^{(i)}] = (W \odot W)\sigma^{(i-1)}$, where $\odot$ is a element-wise product. This work also gives formulae for how uncertainty can be propagated through other commonly used layers.

In addition to this, when Gaussian distributions are chosen for $q(\cdot)$, [78] shows how the Dirichlet distribution can be used as the output distribution for classification in conjunction with these propagated Gaussian feature distributions.

3.4.2 Direct Error Estimation

Instead of using a probabilistic formulation to model the error as a distribution, it is also possible to simply measure and directly estimate the error during training.

For the task of camouflaged object detection, a challenging binary image segmentation task, [79] does this exactly this. The error for the segmentation network is measured, which is then used to supervise the training of a separate 'Online Confidence Estimation Network', which poses error estimation as a binary classification problem.

The Segment Anything Model is introduced in [60], which is a foundation model for image segmentation, and can have its estimated segmentations conditioned on a text or point-wise prompt. It is trained on a very diverse image segmentation dataset which contains annotation masks that use a range of definitions for what an object is, and thus where the segmentation boundaries should be. On top of this, while easy for human interaction, point-wise prompts are sufficiently un-descriptive, that it is often unclear what the correct segmentation should be, i.e. should it be of the whole object, or the object part, or the object sub-part? These factors induce ambiguity in the segmentation task, which this work solves by modelling this uncertainty. The model is trained to output multiple segmentations for a given input, and then a small neural network is trained on top of the model to predict the segmentation quality, in terms of Intersection over Union (IoU), of each of these possible segmentations. During inference, the estimated IoUs are used to then rank the output, where only the most high-quality segmentation is returned. If an image is given to this model, and the highest-quality segmentation it can produce has a low IoU, then it can be said that this was due to aleatoric uncertainty. The model was trained on a vast dataset, and is itself extremely expressive, therefore it is very likely in this case that it was inherently ambiguous where the segmentation boundaries should be.

Finally, [80], [81], also focus on producing image-wise measures of segmentation quality, the Jaccard Index and Dice Similarity Coefficient respectively. These are designed for the clinical setting, where poor quality segmentations can be automatically flagged, leading to closer inspection by a clinician.

Albeit in a different setting, each of these works broadly align with the objective of this thesis, as they seek to estimate when segmentation error is higher (or segmentation quality is lower), such that the effects of this can be mitigated for the broader system.

3.4.3 Generative Modelling

Section 3.4.1 considers how to train a neural network to parameterise a distribution at the network output that captures the variability in its accuracy, and is supervised by the ground-truth error as provided by a labelled training dataset. An alternative approach is to train a segmentation network to parameterise a distribution, which captures the variability in plausible segmentations for a given image. The major motivation for this method is to represent uncertainty via the diversity of the set of plausible and coherent segmentations, rather than a pixel-wise map of uncertainties. This type of method is often seen in the context of medical imaging, where experts frequently disagree on the location of object boundaries, and this disagreement is a key aspect of ensuring good clinical decisions and outcomes.

The key difference between this setting and the previous is that these methods are required to model the joint probability of class assignment over all pixels. This is in contrast to standard cross-entropy training and learned loss attenuation, where the pixels are often modelled independently, therefore sampling from high uncertainty regions leads to high-frequency noise rather than diverse coherent segmentations.

When segmentation networks are trained with this method on a large labelled dataset, the resulting variability at test-time will likely be due to the inherent ambiguity in the input image, rather than ambiguity due to an insufficient dataset. Therefore in this setting, we can treat the segmentation variability as a measure of aleatoric uncertainty.

This distribution of plausible segmentations can be achieved by either (1) defining a distribution in latent space, from which samples are decoded into segmentations, as in [82]–[84], or (2) explicitly modelling the joint distribution at the output of the neural network, as in [85].

In [82], the Probabilistic U-Net is proposed, which combines a conditional Variational Autoencoder (VAE) [86] with a U-Net segmentation network [87]. It is trained such that, during inference, a plausible segmentation is sampled by: (1) using the prior network to parameterise an F_1-dimensional Gaussian conditioned on the input image (2) extracting features $\mathbb{R}^{F_2 \times h \times w}$ from the image with the segmentation network (3) sampling a $\mathbb{R}^{F_1}$ vector from F_1-dim Gaussian, tiling it to the spatial dimensions of the U-Net features, and then concatenating the sample and the U-Net features $\mathbb{R}^{(F_1+F_2) \times h \times w}$ (4) decoding this concatenated vector

into a segmentation map. This work was improved in [83] by folding the prior network into the segmentation network, and having a hierarchy of latent distributions in the decoding stages of the U-Net.

They evaluated the model by injecting ambiguity into the Cityscapes dataset, by randomly flipping class labels with a certain probability e.g. `sidewalk` is flipped to `sidewalk-2` with a probability of $^{8}/_{17}$. The samples from the learned distribution were shown to reflect this ambiguity, as the frequency of class assignment across samples reflected the class flip probabilities.

Another approach that aims to learn a distribution of plausible segmentations is [85]. Instead of using latent distributions, this work seeks to train a network to parameterise a Gaussian distribution; but in contrast to the learned loss attenuation methods, this distribution should model the joint distribution over pixels and classes, $\mathbf{l} \sim \mathcal{N}(\mu, \Sigma)$, where $\mu \in \mathbb{R}^{HW \times K}$ and $\Sigma \in \mathbb{R}^{(HW \times K)^2}$ and Σ is non-diagonal. The enormity of this required distribution makes it intractable to use, so they instead use a low-rank parameterisation of the Σ. Much like [71], they do use Monte Carlo sampling through the softmax function instead of an analytical solution.

3.4.4 Test-Time Augmentation

In Section 3.4.1 and Section 3.4.2, neural networks were trained to tell us at test-time the likelihood of a pixel being correctly segmented based on its appearance. The neural networks learned this by observing when error occurs, and learning to relate appearance and error.

In contrast, another method trains a segmentation network normally, and only investigates the relationship between appearance and error at test-time. It does this with image augmentation, which changes the appearance of an image but leaves the underlying semantic content unchanged. From the distribution of possible image augmentations it is possible to obtain a distribution of possible segmentations, where, the more variable the segmentation of a given pixel, the more likely the pixel is to be segmented incorrectly. As before, if a segmentation network is trained with a training dataset that captures the test distribution well, then inconsistent segmentation is likely due to an inherent lack of information in the image from which to infer where segmentation boundaries are located.

This approach essentially applies perturbations to the input data in order to measure how the model output varies, in contrast to epistemic approaches which apply perturbations to the model parameters instead. This is reflective of the difference in the nature of aleatoric and epistemic uncertainty.

For the task of segmenting 2D and 3D MRI scans, [88] applies flipping, rotating, scaling and random intensity noise as augmentations to a given input scan, yielding 20 perturbed samples. They measure uncertainty from this distribution of segmentations by using the predictive entropy, and show that, for their task, this test-time augmentation outperformed Monte Carlo Dropout in terms of quality of uncertainty estimation.

The work in [89] uses a similar approach for the task of detecting Diabetic Retinopathy in high-resolution images of the retina. It applies both the spatial transforms of the previous, but due to using RGB images, it also uses colour-space transforms such as randomly changing the hue. Instead of 20 samples, this method samples 128 augmentations, and measures uncertainty as the median max class assignment probability.

3.4.5 Discussion

Aleatoric uncertainty estimation methods use training and test data that is entirely in-distribution. This means that when error or segmentation variability is found, it is inferred that this is due to the data's inherent ambiguity, rather than distributional shift, poor model architecture choices, poor model optimization, or any other reason. However, suppose that we used a dataset containing OoD data. In this case, when error or segmentation variability is measured, we might instead attribute it to distributional uncertainty, as discussed in Section 3.2. This means that while the specific experiments in the cited works are not entirely relevant, many of the methods can still be broadly of interest to us if used with OoD data.

Both learned loss attenuation and direct error estimation methods are of interest, as they allow a model to output uncertainty estimates in a single forward pass. This means that memory usage and latency are not meaningfully increased from a standard segmentation network, which is a very desirable characteristic in the context of mobile robotics. The downside to many of these methods is that they require ground-truth in order to determine where error occurs. If the training dataset contains in-distribution and OoD instances within the

same dataset, these methods would require pixel-wise annotation, which is costly in terms of resources.

The cited works using deep generative modelling for aleatoric uncertainty use a labelled training dataset, which contains various possible segmentations of the same image. This is largely because they were from the field of medical imaging and used the lung image database consortium (LIDC) dataset [90], which contains lung scans, each with four annotations from expert clinicians. The curation of this type of dataset is very resource intensive, and likely not feasible for a large scale OoD dataset, as discussed in Chapter 4.

One solution to the problem of a lack of pixel-wise annotations is to use test-time augmentation, which does not involve uncertainty-specific training, i.e. the segmentation network just needs to be trained in a standard manner. This means that error can be discovered without the use of any labels. Therefore, a segmentation network could be trained on a labelled dataset from one domain, and then error due to distributional shift could be estimated by this method at test-time on test data from a different domain. The downside to test-time augmentation is that there is a significant computational cost at test-time, as a given test image needs to be augmented and then segmented many times. A possible solution to this is to use augmentation during training and to distill the distribution of segmentations into a single model, similar to methods that do the same for ensembles or MCD.

3.5 Out-of-Distribution Detection

OoD detection is the task of identifying data points that do not belong to a given data distribution. While the focus of uncertainty estimation is to identify which of the model's attempts to solve a task are incorrect (and sometimes this is due to a distributional shift), by contrast, the focus of OoD detection is to identify instances of distributional shift (and often this leads to error for the task model). There is, however, clearly a significant amount of overlap between the two, and a given OoD detection method's estimated 'OoD-ness' score can immediately be interpreted as an estimate of distributional uncertainty.

There are many different ways to approach this problem, and we will discuss each in the following sub-sections.

3.5.1 Pretrained Methods

One set of methods for OoD detection constrains itself to the use of a frozen pre-trained network, where different inference methods are designed to leverage the learned representation and to produce a OoD score. In this setting, training data is used to train a neural network for classification or segmentation, and the training data distribution is defined as in-distribution. OoD instances are then detected by using the model's learned features, logits or categorical distributions to compare how similar a given test image is to the training images.

A common baseline, [91], calculates the max softmax score, $\mathbf{p}_{\max}$, where the temperature parameter in the softmax function is typically tuned with validation data. [92] improve on this method by adversarially perturbing the input test images, as this empirically resulted in greater separation between the in-distribution and OoD examples. [93] calculates the OoD score based on the Mahalanobis distance between the embeddings of the test image and the labelled training dataset, and does this in a series of feature spaces throughout the network.

In [94], a distinction is made between methods that use features and methods that use logits or max softmax scores. They argue that logits and $\mathbf{p}_{\max}$ contain only class-dependent information, i.e. they only consider the similarity of a given test image to each of the in-distribution classes. By contrast, feature-based methods contain more class-agnostic information.

An experiment in [94] uses the iNaturalist dataset [95] as a OoD dataset (as in [96]) with ImageNet-1k [97] defining in-distribution, where iNaturalist in this setting contains 110 plant classes that are not contained in ImageNet-1k. In this instance, feature-based methods struggle more as they focus less on the class-specific detail that differentiates between the plants in ImageNet and the OoD plants. The class-dependent information in the logits and $\mathbf{p}_{\max}$ methods contain more on the required level of specificity, and therefore OoD detection performance was better, as the model could not decide which of the ImageNet classes to assign the iNaturalist image to.

Then, [94] uses the Describable Textures Dataset [98] as OoD, which contains a diverse set of images that define textures, rather than describe semantic objects. On this dataset they showed that the methods using class-dependent information performed poorly compared

to the feature-based, class-agnostic methods. This shows that features are generally better at describing broader, less semantically driven differences between images. They finally developed a method using logits and features to reap the benefits of both.

An alternative method, presented in [99], applies density estimation to a feature space in order to assess whether a given test feature is in-distribution or OoD. For features z and classes y, the method calculates $p(y)$ and then models $p(z|y)$ with a Gaussian Mixture Model, before calculating the OoD scores as $-\log(p(z))$, where $p(z) = \int p(z|y)p(y)dy$. This work notes that it would be preferable to train the network, such that its representation contained 'task-agnostic' information as well as task-specific information. However, it constrains itself to only developing an inference procedure for OoD detection for a given model.

This therefore suggests that, in addition to the logits and $\mathbf{p}_{\max}$, the features from pre-trained networks may also be largely class- or task-specific, as they are trained to extract information that is salient to the task. This means that these approaches are likely to fail when there is not sufficient information in the training dataset for the model to learn to appropriately describe OoD instances in the features, logits or $\mathbf{p}_{\max}$.

3.5.2 Regularisation-Based Methods

Methods in this sub-section build on those in Section 3.5.1, by adding a loss that regularises neural network training on labelled data. The objective of this is to embed more task-agnostic information into the network, in order to improve OoD detection.

Deterministic Uncertainty Methods

These methods add regularisation to model training by using spectral normalisation [100]. Spectral normalisation layers constrain the Lipschitz constant of a network, where the Lipschitz constant, M, of a function is a value that satisfies:

$$\mathrm{d_A}(\mathtt{f}(x_1), \mathtt{f}(x_2)) \leq M \mathrm{d_B}(x_1, x_2)$$

where $\mathrm{d_A}, \mathrm{d_B}$ are metrics on the sets A and B respectively, for a function $\mathtt{f} : \mathrm{A} \to \mathrm{B}$. The Lipschitz constant is therefore a measure of the extent to which a function amplifies or attenuates

the distance between points x_1, x_2 in their different metric spaces.

The goal of training in this way is to yield a model with a learned representation in which distance between features $z_1 = \mathtt{f}(x_1)$ and $z_2 = \mathtt{f}(x_2)$ is a measure of the semantic difference between pixels x_1 and x_2. If this is true, then we would expect an OoD pixel to be embedded very far away from in-distribution pixels, while in-distribution pixels would be close to other in-distribution pixels of the same class. This therefore allows OoD detection to be performed by comparing the distance between the embedded in-distribution dataset and a given pixel feature from a test image.

Each Deterministic Uncertainty Method (DUM) is trained similarly by using spectral normalisation to perform this regularisation, but each performs inference in a different manner. [101] replaces the output layer of a classification network with a Gaussian Process. [102] measures the similarity between features using Radial Basis Function (RBF) kernels. [103] instead shows that these more complex approaches are not strictly necessary, and that a simpler post-hoc per-class Gaussian Mixture Model achieves similar OoD performance.

Other Regularisation-Based Methods

[104] presents Deep Variational Information Bottleneck, which adds the Gaussian prior loss from VAE [86] training to the latent space in a classifier in order to add regularisation to model training and prevent adversarial attacks. In [105], the authors then showed that using this method improves OoD detection, when the OoD score is taken as the $\mathbf{p}_{\max}$. This shows that this alternative type of regularisation is also effective at improving the representation by encoding task-agnostic information.

3.5.3 Self-Supervised Learning for OoD Detection

In Section 3.5.2, spectral norm was used to ensure that feature space distance is semantically meaningful. This is also the objective of many SSL methods, and so these methods can also used for OoD detection.

SSL methods for computer vision train neural networks to learn a representation from image data without the use of any annotations. Thanks to the lack of labelling requirement, SSL methods are typically paired with large and diverse datasets, such as ImageNet [97].

Therefore, for these works, a 'good' representation is one which can be used to solve a wide range of computer vision problems, e.g. classification, semantic segmentation, depth estimation and object detection.

One way in which this can be achieved is by using data augmentation along with either an instance classification or a clustering task. Data augmentation allows us to change the appearance of an image, whilst keeping its semantics the same. Instance classification tasks then pose network training as a nearest neighbour classification problem, where a pair of augmented images defines its own class or 'instance', as seen in [32], [106], [107]. Clustering tasks instead train a network to assign an image and its augmentation to the same cluster, such as in [108].

Another set of methods instead corrupt a single image, and then train a network to recover the information content of the image from this corrupted input. Examples of this include re-colourisation tasks [109], patch reordering [110], [111] or, most prevalent at the moment, masking tasks [31], [112], [113].

Empirically, it has been shown that many of these methods can extract rich semantic information from a diverse set of images, for example by exhibiting good nearest-neighbour classification performance on ImageNet, or high quality semantic segmentation on ADE-20k [114] or Cityscapes [25] by training only a linear layer on top of the learned representation. Therefore, for the task of OoD detection, this learned representation can be used to detect the differences between in-distribution and OoD instances, as suggested in [115].

A related work is [116] which uses the instance classification framework in [32]. However, the difference is that they do not use a large-scale dataset, but instead solve instance classification as well as the classification task on only the task-specific in-distribution dataset. The objective here is to learn 'task-agnostic' features with the instance classification loss, in addition to the 'task-specific' using the cross-entropy task loss, with these ideas discussed previously in Section 3.5.1.

3.5.4 Proxy Task Methods

An alternative approach is to design a task which requires a semantic understanding of an image to solve, such that a model can successfully solve it for the in-distribution data used

for training, but struggle on OoD data. One such method [117] samples a transform from a set of pre-defined geometric transforms (rotations, horizontal flips, large translations with mirroring), and uses this to augment each training image. The task is then framed as a classification problem, where the model must estimate which of the pre-defined transforms was applied to the original image. The OoD score is then computed as $\mathbf{p}_{\max}$ for a given test image, i.e. it is predicted that the model will not be able to confidently determine which transform was applied to the OoD image, due to a lack of recognition of high-level features such as objects or object parts and no knowledge of how they are usually spatially distributed.

[115] uses a similar approach and defines purely rotational augmentations from the set of rotation angles: $\{0°, 90°, 180°, 270°\}$. The difference is that this work does this in combination with the standard supervised classification training, and uses the classification $\mathbf{p}_{\max}$ as the OoD score. The argument here is that this self-supervised rotation task forces the model to learn to extract features that describe more than just texture, but also detect high-level attributes such object parts. Then, by having a 'higher-quality' representation of the in-distribution data, the classification model is better equipped to detect when an image does not come from this distribution.

3.5.5 Deep Generative Models

Deep generative models seek to learn a high-dimensional representation of the data distribution for the training images, from which additional samples from the same underlying distribution can be sampled. Examples of such techniques include VAEs [86], Generative Adversarial Networks (GANs) [118], auto-regressive models [119], Flow-based models [120], [121], and diffusion models [122]–[124]. Each of these methods provide a way of doing OoD detection as we can query how well a given test image aligns with the learned representation of the in-distribution data. The method for measuring this 'alignment' is typically dependent on which type of generative model is being used.

VAEs, auto-regressive models such as PixelCNN, and Normalising Flows provide a way to evaluate the likelihood of any given test image, and the effectiveness of this for OoD detection was investigated in [125]. The conclusion from this work, is that we have to be careful when doing this, as likelihood appeared to be a poor measure of distributional shift.

Methods such as [126], [127] use reconstruction-based scores instead for a VAE and GAN respectively, and showed this to be an effective method for OoD detection. In a way, this is similar to the literature on using proxy tasks for OoD detection presented in Section 3.5.4, as the generative models are trained by performing reconstruction tasks, and the measure of 'OoD-ness' is how well they can perform this task on a test image.

3.5.6 Use of Out-of-Distribution Data

Each of the previously described OoD detection and uncertainty estimation methods train neural networks using in-distribution data only. They learn a representation from this in-distribution data, and are tasked with detecting differences between the training data and test images. In contrast, the methods described in this section use an OoD training dataset in conjunction with the in-distribution training data.

In principle, these OoD datasets are curated such that each image is OoD, which is mostly achieved by ensuring that the set of classes defined in the OoD dataset is disjoint from the set of classes in the in-distribution dataset. A loss function is then defined which trains the model to learn a separable representation of in-distribution and OoD data.

[128] introduces the common framework for doing this, naming it Outlier Exposure. If the in-distribution dataset $\mathbb{D}_{\text{in}}$ defines a distribution p_{in} and the OoD dataset defines a distribution p_{out}, then the outlier exposure objective can be defined as:

$$L^{\text{OE}} = \mathbb{E}_{p_{\text{in}}}[\mathcal{H}(\mathbf{y}^*, p(y|x))] + \mathbb{E}_{p_{\text{out}}}[L^{\text{OoD}}(p(y|x))] \tag{3.15}$$

Where $L^{\text{OoD}}(p(y|x)) = \mathcal{H}[\text{Uniform}\{0, \text{K}-1\}, p(y|x)]$ is chosen for classification problems, where $\text{Uniform}\{a, b\}$ is the discrete uniform distribution. The first term maximises the sharpness of $p(y|x)$ at the correct class for in-distribution data, while the second term maximises the uniformity of $p(y|x)$ for OoD data. Another common choice for L^{OoD} is $L^{\text{OoD}}(p(y|x)) = \text{KL}[\text{Uniform}\{0, \text{K}-1\} \parallel p(y|x)]$, e.g. [129]. [51], [130] use the same idea, but instead formulate the objective for a Dirichlet-distributed output.

The OoD dataset in these works often derive from a dataset collected for a different task, e.g. a dataset for the classification of a different set of semantic classes. This means that there is a large distributional shift between the in-distribution and OoD dataset. It has been

noted in [128], [129], that the larger this distributional shift, the less well these OoD detection methods work. This is likely because the task of learning a separable representation of two very different sets of data can be fairly trivial, and thus the model can easily overfit to the training datasets. For a more robust separation, the obtainment of *near-distribution* or *boundary* training examples is therefore important, with the hope that the resultant learned representation is significantly more general. These are training images which are OoD enough to cause model error, without appearing so visually distinct from in-distribution images that they are trivially detected. [129] generates such a near-distribution dataset using a GAN, which is trained to generate images for which the classifier is unconfident (i.e. $\mathrm{KL}[\mathrm{Uniform}\{0, \mathrm{K}-1\} \parallel p(y|x)]$ is large) while also fooling the discriminator into thinking the images are in-distribution. If these losses are balanced, [129] shows that the images are both realistic looking, while also OoD, albeit it on small-scale datasets.

3.5.7 Discussion

It is interesting to compare the literature involving the use of an OoD dataset for OoD detection (which we will call outlier exposure for convenience) to the idea of using learned loss attenuation methods with OoD data. In both cases, they train a neural network to learn to detect the differences between in-distribution and OoD data such that they can represent the difference at the network output and reduce the loss. While outlier exposure requires a dataset which explicitly describes the desired separation between certain and uncertain, learned loss attenuation methods measures the error, and thus the desired separation between certain and uncertain is defined as a function of the model. This means that the latter provides noisier supervision as the model's parameterisation changes throughout training, however it also allows for a more flexible definition of the problem, i.e. it does not force you to label which data points should be certain or uncertain.

The lack of flexibility in the problem definition allows for the outlier exposure problem to use the simple formulation of a softmax distribution. By contrast, the additional definitional flexibility typically forces learned loss attenuation problems to use a more complex objective to account for the lack of direct supervision. Note that this is not true for direct error estimation for aleatoric uncertainty estimation, but this is a relatively small subset of the

literature, and is perhaps not widely practised due to the simpler objective's sub-optimality when the noise in supervision (i.e. the label noise) is high.

Deep generative models come with the potential that they can simultaneously learn a rich representation from a large dataset due to a lack of labelling requirement, while also being performant for specific discriminative tasks. The former allows for higher quality OoD detection as larger, more diverse datasets allow neural networks to extract more salient information from images leading to the accentuation of the important differences between in-distribution and OoD images. The problem with this is that, as discussed in [131], generative models are not able to match the discriminative performance of discriminative models. For this reason, a deep generative model would need to be used in parallel to the segmentation network, which is not efficient from the point of view of computational requirements.

The proxy task methods are predicated on the idea that data augmentation can be used to distinguish between in-distribution and OoD instances, much like for test-time augmentation methods for aleatoric uncertainty estimation. While both assume that the model is more invariant to the perturbations to in-distribution data than for OoD data, the cited OoD detection methods use proxy task training to further increase the separation between the two. As stated in Section 3.4.5, using augmentation is a useful way of determining erroneous image regions without the need for ground-truth labelling. The literature using these augmentation-based tasks adds evidence that this could work well for OoD data in the same way that it does with in-distribution data for aleatoric uncertainty estimation.

The final point of interest from this section is from [115], which gives empirical evidence to the idea that improving the representation with self-supervised learning benefits OoD detection.

3.6 Conclusion

In this chapter, we have introduced the concepts of epistemic and aleatoric uncertainty, and situated distributional uncertainty within this framework. We argue that distributional uncertainty can just as easily be viewed as aleatoric uncertainty, as it can epistemic uncertainty. For this reason, both sets of methods are open to investigation, and the key determinant is really the practicality of each method and empirical performance. In terms of practical-

ity, the promising uncertainty estimation methods include: (1) distilling epistemic methods into a single model, (2) aleatoric learned loss attenuation methods applied to OoD data (3) distillation of test-time augmentation based methods, also applied to OoD data.

We have also discussed OoD detection methods, and find that many of these methods have similarities to aleatoric uncertainty estimation methods. This gives evidence that methods such as a combination of learned loss attenuation and outlier exposure methods or OoD proxy task methods and test-time augmentation may well be very well suited to the problem of computationally-light distributional uncertainty estimation.

In the next chapter, we discuss how this thesis measures the quality of a model's uncertainty estimates and the datasets used to do this. This is followed by the presentation of our proposed methods in Chapter 5, Chapter 6 and Chapter 7, which are motivated by the discussions in this chapter.

Chapter 4

Model Evaluation and Datasets

Contents

4.1 Model Evaluation & Metrics 61

4.1.1 Misclassification Detection 62

4.1.2 Metrics: Definitions 62

4.1.3 Metrics: Discussion 65

4.2 SAX Semantic Segmentation Dataset 67

4.2.1 Dataset Motivation 68

4.2.2 Semantic Definitions 69

4.2.3 Inclusion of multiple target domains 70

4.2.4 Curation of the unlabelled SAX training datasets 71

4.3 Other Driving Datasets 72

4.3.1 Cityscapes 72

4.3.2 Berkeley DeepDrive 72

4.3.3 WildDash 73

4.3.4 KITTI 73

This chapter describes the method for evaluating the quality of a segmentation model's uncertainty estimation and the data used to train and test the models.

Section 4.1 introduces the method for evaluation: misclassification detection, where uncertainty estimates are expected to directly detect erroneous pixels. The metrics used to

evaluate a model's ability to perform misclassification detection are discussed in detail in this section, with reference to the robotics context of this thesis.

Then, in Section 4.2, the *SAX Semantic Segmentation Dataset* is introduced, which was collected, curated and then annotated over the course of this thesis to serve as a benchmark for distributional uncertainty estimation. Lastly, in Section 4.3, the other driving datasets used to train and test models are described, such that we can qualitatively understand the distributional shift between them.

4.1 Model Evaluation & Metrics

In this thesis, we are interested in evaluating how a model can perform misclassification detection alongside semantic segmentation on distributionally-shifted data. This means that the uncertainty estimate values are directly interpreted as the model's estimate of the likelihood that a given pixel is $\mathrm{accurate}$ or $\mathrm{inaccurate}$.

This is in contrast to other possible methods for evaluation such as considering model calibration or OoD detection performance, seen in [43], [44] and [128] respectively. As described in Section 2.3, model calibration evaluates how a model's confidence correlates with accuracy in a frequentist manner, i.e. for a batch of N pixels each with similar confidences, it compares the mean confidence to the mean accuracy. These summary statistics are useful for getting a general sense of the quality of a model's uncertainty estimation, but we suggest that they are not suitable for a robotics setting. This is because for a robotic system, every captured image leads to an updated understanding of an environment, and this understanding directly conditions the system's behaviour at a given moment. Therefore, we need to consider the direct relationship between image and error, as errors can immediately lead to dangerous robot behaviour.

OoD detection performance is typically measured by evaluating how the OoD scores are predictive of whether a given image or pixel is $\mathrm{in\text{-}distribution}$ or OoD, as introduced in Section 3.5. This is similar to misclassification detection, however in OoD detection, $\mathrm{in\text{-}distribution}$ and OoD are being conflated with $\mathrm{accurate}$ and $\mathrm{inaccurate}$ respectively. In our setting, the safety of the system is more closely related to whether image regions are inaccurate or not, than whether they are OoD or not.

4.1.1 Misclassification Detection

Misclassification detection is a binary classification problem whereby the uncertainty estimates 'classify' each pixel as either $\mathrm{certain}$ or $\mathrm{uncertain}$ and this is compared to whether the pixel was segmented accurately, giving the label states $\mathrm{accurate}$ and $\mathrm{inaccurate}$ – as can also be found in [91]. Clearly, the ideal model estimates uncertainty such that each pixel is either $(\mathrm{certain}, \mathrm{accurate})$ or $(\mathrm{uncertain}, \mathrm{inaccurate})$. Here is a confusion matrix that defines the possible states:

		Predicted	
		$[\mathrm{certain}]_t$	$[\mathrm{uncertain}]_t$
Actual	accurate	$[\mathrm{TP}]_t$	$[\mathrm{FN}]_t$
	inaccurate	$[\mathrm{FP}]_t$	$[\mathrm{TN}]_t$

We define the $\mathrm{accurate}$ and $\mathrm{inaccurate}$ label states as positive and negative respectively. Each of the states: TP, TN, FP, FN, are calculated for a specific threshold t on a model's real-valued uncertainty estimates, as illustrated by $[.]_t$. Therefore which pixels are $\mathrm{certain}$ and $\mathrm{uncertain}$ is also determined by the threshold t. From now on, this notation is omitted for: TP, TN, FP, FN, in the interest of brevity.

The best model can be chosen from a set of imperfect models by a wide range of metrics, owing to the fact that this is ultimately a binary classification problem. Ultimately, the best metric to use for evaluating uncertainty estimation is dependent on the context in which the model is being deployed. We therefore consider a range of possible metrics and justify the use of each in the context of robotics.

4.1.2 Metrics: Definitions

ROC Curves

A common way to evaluate binary classification algorithms is to use Receiver Operating Characteristic (ROC) and Precision-Recall (PR) curves, and this also extends to misclassification detection, as seen in [91]. ROC curves plot two quantities: the True Positive Rate

(TPR) versus the False Positive Rate (FPR):

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \ \mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$$

The TPR represents the proportion of accurate pixels that are estimated to be certain, while the FPR is the proportion of inaccurate pixels that are erroneously estimated to be certain. These quantities are calculated separately for the accurate and inaccurate pixels, i.e. the pixels that belong to the positive and negative class respectively. They are thus independent of the class distribution, and so are not affected by the semantic segmentation accuracy. In order to calculate a single quantity, TPR and FPR are calculated for a range of thresholds, and then the area under this curve is calculated, known as the AUROC. This is maximised for the ideal model, where $\mathrm{AUROC}^* = 1$.

PR Curves

PR curves are generated by plotting Precision versus Recall, where each are calculated as:

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \ \mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

These curves are often used in context of information retrieval, where the task is to retrieve the positive class instances, while retrieving as few of the negative class instances as possible. Using this interpretation for misclassification detection, PR curves measure how effective a model's uncertainty estimates are at finding only the accurate pixels, while ignoring the inaccurate pixels (see above that accurate is defined as the positive class, not inaccurate). Similarly, these curves can be summarised by calculating the area under them, giving the AUPR. Again, the ideal model also maximises this metric, with $\mathrm{AUPR}^* = 1$.

F_β Scores

Another method for aggregating PR curves is to calculate the F_β score:

$$\mathrm{F}_\beta = \frac{(1 + \beta^2)\mathrm{TP}}{(1 + \beta^2)\mathrm{TP} + \mathrm{FP} + \beta^2\mathrm{FN}}$$

The benefit of this metric is that it includes a hyperparameter β that can be used to weigh the importance of precision and recall for any given application. When $\beta < 1$, greater importance is given to $\mathrm{Precision}$, while for $\beta > 1$, greater importance is given to Recall[1]. Typically, the values used for β are $\frac{1}{2}$, 1 and 2, giving the $\mathrm{F}_{1/2}$, F_1 and F_2 scores.

Misclassification Detection Accuracy

Given our binary classification problem of misclassification detection, we can also simply consider the accuracy of the model on this problem, given the name A_{MD} (not to be confused with the semantic segmentation accuracy), and is calculated as follows:

$$\mathrm{A}_{\mathrm{MD}} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

Under this metric, an ideal model is one which assigns the greatest number of pixels to one of two 'safe' states: $(\mathrm{accurate}, \mathrm{certain})$ and $(\mathrm{inaccurate}, \mathrm{uncertain})$.

Single-threshold versus All-threshold metrics

The possible states (TP, TN, FP, FN) are all calculated for a single value of the uncertainty threshold. However, when we calculate ROC and PR curves and take the area under these curves, we are summarising the performance over all thresholds.

It is however also useful to calculate metrics for single values of the threshold, i.e. to evaluate the effectiveness of a method at a single point on the curve. For this type of analysis, we consider the threshold that maximises F_β and A_{MD} scores, known as MaxF_β and $\mathrm{MaxA}_{\mathrm{MD}}$ scores.

An additional consideration is needed for this analysis, however, as it is possible that a model with very low segmentation accuracy on a challenging target domain might end up with a very good MaxF_β or $\mathrm{MaxA}_{\mathrm{MD}}$ by expressing uncertainty over all pixels. It might be the case that this model looks better in terms of these metrics than a model that segments the target domain better, but still largely rejects each pixel as uncertain.

For this reason, we pair the MaxF_β and $\mathrm{MaxA}_{\mathrm{MD}}$ scores with the $\mathrm{p}(\mathrm{a}, \mathrm{c})$, which is the

[1]Note that as $\beta \to 0$, $\mathrm{F}_\beta \to \mathrm{Precision}$, and as $\beta \to \infty$, $\mathrm{F}_\beta \to \mathrm{Recall}$

proportion of pixels that are $\mathrm{accurate}$ and $\mathrm{certain}$:

$$\mathrm{p(a, c)} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

In this way, we can also see if the model is actually solving the semantic segmentation task, as well as being able to detect distributional shift. We present $\mathrm{MaxF}_\beta \ @ \ \mathrm{p(a, c)}$ and $\mathrm{MaxA_{MD}} \ @ \ \mathrm{p(a, c)}$.

4.1.3 Metrics: Discussion

In this thesis, we are primarily interested in situating these evaluations in the context of semantic segmentation for real-world mobile robotics applications. Therefore, it is important that we consider the following:

1. What are the misclassification costs for pixels that are FP versus FN in real-world safety-critical settings?

2. Should the metrics we use to select the best model be independent of the class distribution? By class distribution, we mean the proportion of pixels that are accurate, a.k.a. $\mathrm{p(accurate)}$, which is the same as the semantic segmentation accuracy.

Naturally, the answers to these questions are context-dependent, both in terms of the nature of the robot deployment, and also how the semantic segmentation maps are used for robotic planning and control. However, we can still make some general statements.

For the first question, we can say there is a higher importance of $\mathrm{certain}$ pixels being $\mathrm{accurate}$, than $\mathrm{uncertain}$ pixels being $\mathrm{inaccurate}$. Therefore, there should be a higher misclassification cost to FP pixels than FN. Put simply, dangerous situations involving robots are often arise when perception systems have an over-confident and inaccurate understanding their surroundings.

In contrast, broadly speaking, perception systems that are under-confident are typically overly-conservative in their estimation of what is a safe action. This leads to systems that are possibly inefficient, but are less dangerous.

For these reasons, we argue that precision is more important than recall for misclassification detection tasks, where we are trying to evaluate the quality and usefulness of uncer-

tainty estimation for robotics. This means that we ideally want no FP pixels, even if that means we have fewer TP pixels.

In order to encode this preference into our model evaluation, we use the F_β scores. More specifically, we use the $\mathrm{F}_{1/2}$ score to represent our greater interest in precision over recall.

What about the other metrics?

PR and ROC curves show us the misclassification performance over the full range of thresholds and the full range of misclassification costs. For this reason, the summary metrics AUROC and AUPR aggregate the performance over all misclassification costs. This is an important consideration as it is therefore possible to choose a model based on these metrics that might be sub-optimal for a specific context (with its specific weighting of different misclassification costs). Nonetheless, they are useful metrics for giving a broad sense of the quality of uncertainty estimation.

Misclassification detection accuracy, A_{MD}, similarly expresses no preference between FP and FN. In contrast to the aforementioned curves, we consider the threshold at which A_{MD} is maximised, i.e. $\mathrm{MaxA}_{\mathrm{MD}}$, as opposed to aggregating the performance over all thresholds. This gives a different perspective on quality of uncertainty estimation, and is useful when we do not care about the relative misclassification costs. An example of when this might occur is when the semantic segmentations are being used for localisation or mapping. In the former of these cases, further post-processing steps such as RANSAC are being used, which can reject FP pixel themselves. Similarly, in the latter case, majority voting can reduce the segmentation noise, turning possible FP pixels into TP.

We make the case that, in these applications, choosing the model that has the highest $\mathrm{MaxA}_{\mathrm{MD}}$ is more appropriate. An important note, is that typically the values of $\mathrm{p}(\mathrm{a}, \mathrm{c})$ at which we have $\mathrm{MaxA}_{\mathrm{MD}}$ are typically higher than that of $\mathrm{MaxF}_{1/2}$, which is likely to be useful in these applications, without compromising our wider trust of the system, as can been seen in Section 5.4, Section 6.9 and Section 6.10.

Discussion: Class Distribution

It is an often cited benefit that PR and ROC curves are independent of the class distribution of the test data. This means that in a binary classification problem, metrics based on these curves are independent of the underlying proportions of how many data points belong to each of the positive and negative classes.

Take the problem of disease detection for example. In this case, the class distribution is represented by: $\mathrm{p(disease)}$ and $\mathrm{p(healthy)}$ in the population of patients. We do not want the metrics to evaluate the best method to be conditioned on the rate of disease for one given dataset, as then the method would be sub-optimal for another dataset. For this reason, is important that the metrics are independent of $\mathrm{p(disease)}$ and $\mathrm{p(healthy)}$.

Our misclassification detection problem is a little different, as the class distribution is represented by: $\mathrm{p(accurate)}$ and $\mathrm{p(inaccurate)}$. Although it is clearly true that we want to pick the model with the best uncertainty estimation, we cannot choose it while ignoring that we also need a model that can return pixels that are accurately segmented. Therefore, the metrics we use must also allow us to pick model that produces high-quality semantic segmentations, relating to having a high $\mathrm{p(accurate)}$. AUROC and AUPR obfuscate this information, which leads us to also considering $\mathrm{MaxF_{1/2}} \mathbin{@} \mathrm{p(a, c)}$ and $\mathrm{MaxA_{MD}} \mathbin{@} \mathrm{p(a, c)}$.

$\mathrm{MaxF_{1/2}}$ and $\mathrm{MaxA_{MD}}$ allow us to consider the quality of uncertainty estimation for two sets of misclassification costs, while $\mathrm{p(a, c)}$ informs us of how many pixels are actually assigned to the known classes as opposed to assigned to unknown. The former represents how *safe* the model is, while the latter tells how *useful* the model is.

In addition to giving values for $\mathrm{MaxF_{1/2}} \mathbin{@} \mathrm{p(a, c)}$ and $\mathrm{MaxA_{MD}} \mathbin{@} \mathrm{p(a, c)}$ in tables, we also plot the full range of $\mathrm{F_{1/2}}$ and $\mathrm{A_{MD}}$ versus $\mathrm{p(a, c)}$.

4.2 SAX Semantic Segmentation Dataset

As part of the Sense-Assess-eXplain (SAX) project [132], a semantic segmentation dataset was created, known as the SAX Semantic Segmentation Dataset. The purpose of the SAX project was to develop methods along three themes: (1) the robust perception of environments, (2) the self-assessment of model performance for a given task and, (3) the explanation

of performance to a human user. The work conducted in this thesis focusses strongly on the first two of these themes.

In order to evaluate how methods satisfy each of the three themes, this project involved collecting large amounts of driving data in unusual and challenging environments, and to produce datasets with ground truth that can be used for model evaluation.

My contributions to this dataset were as follows: (1) The overall design of the dataset - how the broader dataset is split into training and test datasets, which domains to include data from, and specifically which images to use, (2) A significant proportion of the pixel-wise labelling, which was done by hand using an internal tool of the Oxford Robotics Institute.

4.2.1 Dataset Motivation

In the interest of investigating how semantic segmentation quality degrades due to distributional shift, and how methods can detect this, a pixel-wise labelled segmentation dataset has been created from some of this collected data. The dataset is split up by geographic domain, so that we can compare metrics between each of the domains.

Due to the significant amount of time and effort required to label each image in a pixel-wise manner, it was only feasible to label a set of images in the order of hundreds, and these labelled images are therefore used as a test dataset. Segmentation networks can therefore be trained on a labelled dataset from one domain, and then tested on the challenging SAX domains to determine (1) the extent to which the performance degrades and (2) the extent to which this can be mitigated via uncertainty estimation.

In addition to the labelled test datasets, the SAX Semantic Segmentation Dataset also includes a large unlabelled dataset for each domain. This is motivated by the literature and discussion in Chapter 3, where the idea of training with unlabelled OoD images is raised. This can be investigated with the 100,000 unlabelled images included for each domain.

The successive sub-sections discuss the characteristics of the dataset, and how they allow us to evaluate segmentation and uncertainty estimation quality for a range of novel methods for training segmentation networks.

4.2.2 Semantic Definitions

As described above, the desired use for this dataset is in conjunction with a pixel-wise labelled training dataset from another domain. We are interested in evaluating how the model extends the visual semantic concepts defined in the labelled training dataset to the distributionally shifted images in the test dataset. Therefore, across both the training and test datasets, we want the pixels to be annotated with the same semantic classes.

For this reason, the test dataset was annotated in accordance with the labelling policy used in the Cityscapes dataset [25]. This is because this labelling policy is a commonly used standard for driving data, allowing interoperability with other datasets as well, such as with BerkelyDeepDrive [63], KITTI [133] and WildDash [134].

In addition to how the images are labelled, it is also important to determine which images should be labelled from the large pool of collected images to make an informative test dataset.

One possible objective for segmentation datasets is that training and test datasets are prepared such that all the information required to perfectly segment the test dataset is provided in the training dataset, i.e. the uncertainty is purely epistemic, and can be fully reduced for the test dataset. This means that weather and illumination conditions do not corrupt the images and no instances of OoD classes are present, or, if there are, they are labelled `void` and ignored during training and testing. This is a useful setting for the evaluation of segmentation network architectures, however is not reflective of deploying robot systems into the real world.

By contrast, for the SAX test datasets, a conscious effort was made to include objects that do not belong to any of the defined semantic classes. This is in addition to objects that belong to the known classes, but look very different, which are very prevalent when the geographic domain significantly changes. Another possible source of distributional uncertainty for the SAX datasets is the sensor type, for which a PointGrey Bumblebee XB3 stereo camera was used. By contrast, the Cityscapes dataset used a stereo camera with $\frac{1}{3}$-inch CMOS $2\,\mathrm{MP}$ sensors, namely OnSemi AR0331s. Each of these factors ensure that there will be distributional uncertainty for us to investigate in the test datasets.

4.2.3 Inclusion of multiple target domains

As discussed, this dataset is made up of sub-datasets, which are each collected in a given geographic domain. Specifically, three geographic domains are considered: London, the New Forest, and Scottish Highlands. These represent a spectrum of distributional shift from an urban driving dataset, such as Cityscapes, with London being the most similar and the Scottish Highlands being most different.

These datasets allow us to investigate how the quality of segmentation and uncertainty estimation vary with the magnitude of the distributional shift. They also allow us to investigate how training with unlabelled data from each of these domains can help to mitigate performance degradation on both of these fronts.

The rest of this sub-section describes each of the SAX domains in order to get a sense of the nature of the distributional shift between these domains and other public datasets. For examples from each domain see: Figure 4.1, Figure 4.2, Figure 4.3 and Figure 4.4.

SAX London

This domain is most similar in appearance and spatial distribution of classes to other urban driving datasets. The likely differences in the appearance of the defined semantic classes from other urban driving datasets include: markings on the roads, different architecture of buildings, different signage. There are also a number of objects of undefined class, such as bus stops, dogs, and road works. The conditions for this dataset vary from dry and overcast, to dark and lightly raining, with the latter likely causing a larger distributional shift than the former.

SAX New Forest

The New Forest domain is less urban than the London domain, as the images were collected in small towns and their surroundings in a rural region of Southern England[2]. A major difference between the New Forest and other urban driving domains is the distribution of classes, i.e. terrain and vegetation are much more frequent in the former, while buildings, traffic lights, and pedestrians are much more frequent in the latter. Given the rural nature of

[2] `https://www.thenewforest.co.uk/`

this domain, the road boundaries are less clear, and it contains many instances of unknown classes such as horses, cows and donkeys, which are allowed to roam freely[3], and so are commonly found in and besides the road.

SAX Scotland

Images for this domain were collected in and around the Ardverikie Estate[4] in the Scottish Highlands, and are the least similar to other urban driving datasets. This is because many of the images were collected away from public roads and in very rural settings. This means that the distribution of classes is different, as well as the classes themselves looking quite different. A notable example of this is that the class `road` is often represented by dirt tracks or gravel roads rather than paved streets. Another is that the `terrain` class in a dataset such as Cityscapes refers to patches of grass, whereas in the Scottish Highlands the most similar instances of `terrain` would instead be heather moorland[5].

4.2.4 Curation of the unlabelled SAX training datasets

As mentioned, each SAX labelled test dataset has an accompanying unlabelled training dataset. Each of these unlabelled datasets contain approximately 100,000 images. Given a dataset of this size, careful curation of which images should and should not be included would be extremely time-consuming. Therefore, the only curation performed was to remove any images that are too close in time or location to the labelled test images.

The dataset was obtained by uniformly sampling at a frequency of $4\,\mathrm{Hz}$ from a pool of videos collected by the data collection vehicle. Which videos are chosen was broadly guided to get a range of different environments within a domain, and the number of videos was chosen such that sampling in this manner yields 100,000 images.

[3] `https://www.newforestnpa.gov.uk/discover/commoning/the-animals/`
[4] `https://ardverikie.com/`
[5] `https://en.wikipedia.org/wiki/Moorland`

4.3 Other Driving Datasets

Additional datasets are used, which define source and target domains, where the source domain provides labelled training images and the target domain is distributionally shifted from the source domain, and provides labelled test images and (optionally) unlabelled training images.

4.3.1 Cityscapes

The Cityscapes dataset [25] contains 5000 pixel-wise labelled images, with a $60\,\%$ - $10\,\%$ - $30\,\%$ split between train-val-test. They are collected in 49 German cities and one Swiss city in the daytime, in bright but not sunny conditions and during the spring, summer, and autumn. The semantic class definitions used in Cityscapes are used in all of our experiments, and are defined as follows (the colour denotes the visualisation colour seen in figures): Examples from this dataset can be seen in Figure 4.5.

- road
- pole
- sky
- bus
- sidewalk
- traffic light
- person
- train
- building
- traffic sign
- rider
- motorcycle
- wall
- vegetation
- car
- bicycle
- fence
- terrain
- truck
- void

4.3.2 Berkeley DeepDrive

Berkeley DeepDrive (BDD) [63] uses the same semantic definitions as Cityscapes, although the classes are visually different due to collection in Berkeley, New York, San Francisco, and the Bay Area. Additionally, images are collected in the day and night, and in rainy, snowy, and foggy conditions, in addition to clear and overcast. This means that BDD is a much more diverse dataset than Cityscapes. It is also larger, as it contains 10,000 pixel-wise labelled images with a $70\,\%$ - $10\,\%$ - $20\,\%$ split for train-val-test. There is also a larger BDD dataset, named BDD-100k, which contains 100,000 unlabelled images.

We use BDD in a variety of ways including: (1) as a target test domain, where only the test split is used, and the labelled training dataset from another domain is used, (2) as a target training and test domain in which the 100,000 unlabelled images are used for training, while the labelled test split is used for testing, (3) as a source training domain, where the labelled train split is used for training instead of Cityscapes. Examples from this dataset can be seen in Figure 4.6.

4.3.3 WildDash

WildDash [134] is a dataset of 4256 pixel-wise labelled images that are collected from dash-cams mounted on vehicles all around the world. In contrast to the other datasets, this dataset is designed to be used only as a labelled test dataset. It is the most diverse of any of the datasets considered as it contains rural and urban scenes from each continent, and in a variety of weather and illumination conditions. It is therefore a significant challenge to produce both high-quality segmentations and uncertainty estimates, and is a useful benchmark in assessing the generality of both. Examples from this dataset can be seen in Figure 4.6.

4.3.4 KITTI

KITTI [133] is collected in Karlsruhe, Germany and the semantic segmentation portion contains 400 pixel-wise annotated images in a $50\,\%$ - $50\,\%$ train-test split. This is therefore not a dataset with significant diversity, as it is a small dataset collected within urban and semi-urban Karlsruhe. In addition to this dataset, there are also KITTI raw logs containing many more unlabelled images, split into separate sequences for each sub-domain: City, Residential, Road, Campus, Person, Calibration.

We use KITTI domain data in the following ways: (1) it is used as a target test dataset in which the train split is used, as the test labels are withheld for testing on their servers, (2) the KITTI raw images are also used as an unlabelled target training dataset, with testing occurring on the labelled train split as in (1). The raw images for this unlabelled dataset are chosen such that the images are from the same sub-domains, but not from the same sequence as the images in the used test dataset (again, we in fact use the train split). Examples from this dataset can be seen in Figure 4.6.



(a) SAX London



(b) SAX New Forest



(c) SAX Scotland

Figure 4.1: Example test images (left) and ground-truth (right) from each of the domains in the SAX Semantic Segmentation Dataset



Figure 4.2: Examples from the SAX London test dataset (with the labels omitted). Images are chosen for each row, such that the instances in these images cause the rows to exhibit increasing distributional shift from top to bottom.



Figure 4.3: Examples from the SAX New Forest test dataset (with the labels omitted). Images are chosen for each row, such that the instances in these images cause the rows to exhibit increasing distributional shift from top to bottom.



Figure 4.4: Examples from the SAX Scotland test dataset (with the labels omitted). Images are chosen for each row, such that the instances in these images cause the rows to exhibit increasing distributional shift from top to bottom.



Figure 4.5: Examples from the Cityscapes training dataset.



(a) Berkeley DeepDrive



(b) KITTI



(c) WildDash

Figure 4.6: Example test images from Berkeley DeepDrive, KITTI and WildDash datasets. The examples are chosen such that the distributional shift with respect to Cityscapes increases from left to right.

Chapter 5

Learning OoD Detection from Large-Scale Datasets

Contents

5.1 Motivation 81

5.1.1 Contrastive Learning 82

5.1.2 Training Data 84

5.2 Proposed System Design 86

5.2.1 Overview 86

5.2.2 Objective Function 86

5.2.3 Masking Label Noise 89

5.2.4 Data Augmentation 89

5.3 Experimental Setup 91

5.3.1 Datasets 91

5.3.2 Network Architecture 92

5.4 Experiments and Results 93

5.4.1 Data Augmentation Experiments 94

5.4.2 Data Augmentation Results 94

5.4.3 Objective Function Experiments 94

5.4.4 Objective Function Results 95

5.4.5 Data Diversity Experiments 95
5.4.6 Data Diversity Results 96
5.5 Conclusion 96

This chapter presents a method that uses a large-scale image recognition dataset as a source of OoD data to learn distributional uncertainty estimation via OoD detection. It achieves this by defining a challenging OoD detection task using data augmentation to fuse in-distribution and OoD images into the same image. A segmentation network is then trained to solve this OoD detection task using a variation on the typical contrastive loss, and is evaluated on its ability to detect error on distributionally-shifted test datasets. This method was published as:

- D. Williams, M. Gadd, D. De Martini & P. Newman, Fool Me Once: Robust Selective Segmentation via Out-of-Distribution Detection with Contrastive Learning, *International Conference on Robotics and Automation (ICRA), 2021*

5.1 Motivation

This work primarily draws on the outlier exposure literature, presented in Section 3.5.6. In addition to in-distribution data, these works train on OoD data such that the representation between the two is separable. As previously stated in Section 3.5.6, the objective for this is often formulated as follows:

$$L^{\text{OE}} = \mathbb{E}_{p_{\text{in}}}[\mathcal{H}(\mathbf{y}_i^*, p(y|x))] + \mathbb{E}_{p_{\text{out}}}[\text{KL}[\text{Uniform}\{0, \text{K}-1\} \parallel p(y|x)]] \tag{5.1}$$

Where the first term of L^{OE} is the classification loss which minimises the entropy of $p(y|x)$ over p_{in}, i.e. the data distribution defined by the in-distribution dataset. The second term minimises the distance between $p(y|x)$ and the uniform distribution over p_{out}, i.e. the data distribution defined by the OoD dataset, thereby maximising its entropy over this data distribution.

As presented in [128], [129], the key observation in this literature is that different OoD datasets result in very different OoD performance, and so the nature of the data is a key

factor to consider. This motivates the method in this chapter to leverage existing datasets that are very large and diverse, in combination with data augmentation.

Datasets such as ImageNet [97] or LAION [135] attempt to approximate the set of all natural images, and are typically used to train neural networks to extract general semantic features from natural images that can be fine-tuned to solve many computer vision tasks. These trained neural networks are often evaluated on large-scale image recognition challenges, such as ImageNet [97].

The idea to leverage these datasets comes from framing OoD detection as a simplified version of large-scale image recognition. While these recognition problems require class assignment to some approximation of all possible classes, OoD detection requires assignment either to the classes defined in the in-distribution training dataset or to the 'unknown' class. The fact that the sub-classes of the 'unknown' class, i.e. all semantic classes besides those defined as in-distribution, need not be differentiated is what leads to the simplification as compared to large-scale image recognition problems, i.e. we might say that the OoD instances need to be *detected*, but not *recognised*.

Although simpler than large-scale recognition, this OoD detection task is still very challenging due to the enormity of the set of OoD instances. In this chapter, we therefore hypothesise that large-scale datasets can be a key part of solving this problem. A popular method for leveraging this type of dataset uses both contrastive learning and data augmentation. The work in this chapter draws inspiration from this work, and adapts it for the task of OoD detection.

Another key difference between large-scale image recognition challenges, such as ImageNet, and the task in this chapter, is that we seek to detect in-distribution and OoD instances on a pixel-wise basis. This requires adapting both the OoD detection training task, and also the objective used to solve this task.

5.1.1 Contrastive Learning

Contrastive learning defines a set of objectives which compare data points that are either positive or negative, where the positive data points should all be represented similarly and differently from negative data points. For computer vision, the positive and negative data

points are typically generated using data augmentation in what is referred to as an instance classification task. Each image has a single positive pairing, which is the augmented version of itself, while all other images would be considered negative.

The type of data augmentation used generally involves using a crop and resize to the original spatial dimensions, along with a wide array of colour-space augmentations. This yields two images that contain the same semantic content on an image-wise basis, while looking very different (often referred to as different 'views' of the same image). Therefore, if a neural network represents both of these images similarly, then it must have learned an image-wise semantic representation of images.

Negatives are used to prevent a training failure mode known as 'feature collapse'. This occurs when the model learns to reduce the loss by ignoring the input images, and outputting the same feature vector for any input. When the loss is only calculated between positives, this is a solution that minimises the loss, and so occurs frequently. Therefore, the loss between positives and negatives is included to prevent this failure mode.

For a pair of positives, we cannot ensure that each of the negatives are of a different semantic class, as we do not have labels for the training dataset. This is what makes this formulation 'instance classification', rather than image classification, as each image is essentially defining its own class. Nonetheless, this approximation has been shown to empirically lead to representations suited to large-scale image recognition [32], [106]. Additionally, [136] explores how negatives provide a component of the loss that maximises the uniformity of the learned feature representation, which is shown to be a useful characteristic. In a similar vein, [116] uses contrastive learning to learn task-agnostic information, which is needed to discriminate between instances belonging to the same class. This is shown to improve a network's representation for OoD detection.

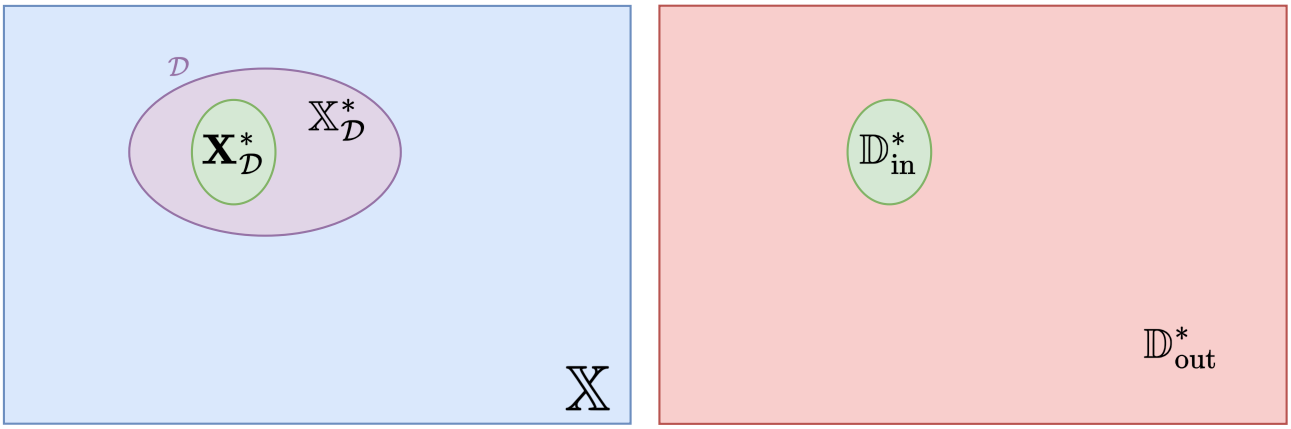
The contrastive objective can be calculated for a pair of positive features (z_i, z_j) denoted by (i, j):

$$L_{i,j}^{\text{Con}} = -\log \frac{\exp(z_i^\top z_j/\tau)}{\sum_{k,k\neq i} \exp(z_i^\top z_k/\tau)} \tag{5.2}$$

This can be viewed as a cross-entropy classification objective applied to instance classification, where the logits are calculated by feature similarity. The cosine similarity is typically used, and so $\|z\|_2 = 1$.

The method in this chapter uses this type of loss for the problem of learning a separable representation between in-distribution and OoD data. More specifically, it needs to do this at a pixel-wise level, such that the pixel-wise embeddings of in-distribution pixels are distinct from OoD pixels.

5.1.2 Training Data



(a) (b)

Figure 5.1: An illustration of the definitions of the in-distribution and OoD datasets in Section 5.1.2. In (a), $\mathbb{X}$ is the set of all possible images, and within it we can define a domain $\mathcal{D}$, which is defined by a set of classes $\mathbb{K}$ and their appearance. $\mathbb{X}^*_{\mathcal{D}}$ is the set of all images that contain instances of classes in $\mathbb{K}$ with the appearance defined by domain $\mathcal{D}$. $\mathbf{X}^*_{\mathcal{D}}$ is the set of pixels within $\mathbb{X}^*_{\mathcal{D}}$ which belong to the classes in $\mathbb{K}$. Finally, the ideal datasets, $\mathbb{D}^*_{\text{in}}$ and $\mathbb{D}^*_{\text{out}}$, can be defined by $\mathbf{X}^*_{\mathcal{D}}$ and the rest of pixel space respectively, as shown in (b), along with their ground-truth.

Firstly, along with Figure 5.1, this section considers what we mean by in-distribution and OoD by defining idealised datasets of each: $\mathbb{D}^*_{\text{in}}$ and $\mathbb{D}^*_{\text{out}}$. They are defined as ideal, in the sense that they would allow us to fully solve the task of OoD detection[1] via straightforward supervised learning.

For the in-distribution dataset, let us define a domain $\mathcal{D}$, which is a high-level description of the context and conditions in which a set of semantic classes $\mathbb{K}$ are captured with a given sensor, e.g. a domain could be: RGB images of driving in London, in dry, overcast conditions, during summer. Note that the domain describes the *appearance* of the semantic classes in the data collected for this domain, and is not exclusive to conditions of the domain, i.e. a car imaged in Manchester could still be in the example domain above if it has the same

[1] Assuming we have a sufficiently expressive neural network and an appropriate optimizer.

appearance as those in London and in the described conditions.

Using the domain $\mathcal{D}$, we can then define $\mathbb{X}^*_{\mathcal{D}}$ (see Figure 5.1) which is the set of all images which include instances from the domain $\mathcal{D}$. Note that the images in $\mathbb{X}^*_{\mathcal{D}}$ might not entirely be composed of instances with a semantic class in $\mathbb{K}$, therefore not every pixel is considered in-distribution. Therefore, we define the in-distribution dataset with $\mathbb{D}^*_{\text{in}} = \{\mathbf{X}^*_{\mathcal{D}}, \mathbf{Y}^*_{\mathcal{D}}\}$, where $\mathbf{X}^*_{\mathcal{D}} = \{\mathbf{x}_i \mid (\mathbf{x}_i \in \mathbf{x}_{\mathcal{D}}, \mathbf{x}_{\mathcal{D}} \in \mathbb{X}^*_{\mathcal{D}}), \mathbf{y}^*_i \in \mathbb{K}\}$, i.e. all the pixels in $\mathbb{X}^*_{\mathcal{D}}$ that correspond to the classes in $\mathbb{K}$, and $\mathbf{Y}^*_{\mathcal{D}}$ are the corresponding pixel-wise labels describing which of the $\mathbb{K}$ classes the pixels belong to.

Considering the definition of what is in-distribution, we can then define the set of all OoD images, as any image that is not from the domain $\mathcal{D}$, therefore approximately $\mathbb{X}^*_{\text{out}} = \mathbb{X} - \mathbb{X}^*_{\mathcal{D}}$, where $\mathbb{X}$ is the set of all images. However, for completeness, we also have to define this on a pixel-wise basis as there were pixels that were OoD in $\mathbb{X}^*_{\mathcal{D}}$. Therefore, $\mathbf{X}^*_{\text{out}} = \{\mathbf{x}_i \mid (\mathbf{x}_i \in \mathbf{x}_{\mathcal{D}}, \mathbf{x}_{\mathcal{D}} \in \mathbb{X}^*_{\mathcal{D}}), \mathbf{y}^*_i \notin \mathbb{K}\} \cup \{\mathbf{x}_i \mid (\mathbf{x}_i \in \mathbf{x}_{\text{out}}, \mathbf{x}_{\text{out}} \in \mathbb{X}^*_{\text{out}})\}$. The OoD dataset is then given by $\mathbb{D}^*_{\text{out}} = \{\mathbf{X}^*_{\text{out}}, \mathbf{Y}^*_{\text{out}}\}$, where $\mathbf{Y}^*_{\text{out}}$ are the corresponding pixel-wise labels, which simply denote that each pixel is OoD.

This definition of $\mathbb{D}^*_{\text{out}}$ demonstrates the enormous size and diversity of the ideal OoD dataset. Factoring in the difficulties of pixel-wise labelling, the resource cost of obtaining a good approximation of $\mathbb{D}^*_{\text{out}}$ is prohibitively expensive. For this reason, what is instead available is small datasets of pixel-wise labelled images for a given task, or large and diverse datasets of natural images that contain image-wise or no annotation.

Given the size of $\mathbb{D}^*_{\text{out}}$, we make the determination that small specific datasets are insufficient for this task, and so must consider how the latter can be used. The problem with this type of dataset is that the lack of pixel-wise labelling means that we do not know which pixels are OoD in any image. For this reason, we must make the approximation that the entirety of this dataset is OoD, introducing label noise into the problem. It also means that instead of in-distribution and OoD instances being within the same image as is the case in the test datasets, they are separated into different images, i.e. we have image-wise labels for in-distribution and OoD. Training on this type of dataset will generalise poorly to pixel-wise distributional uncertainty estimation at test-time.

Each of these problems are addressed and mitigated in the proposed method, by using a

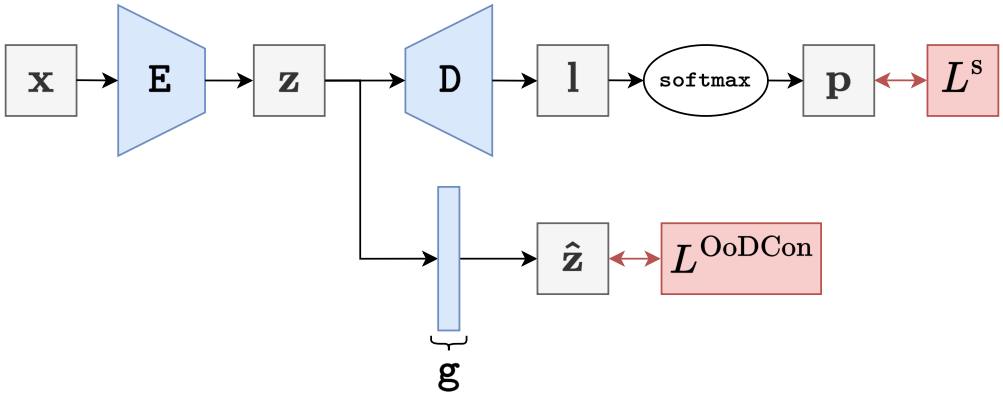


Figure 5.2: For the method described in Section 5.2, this figure depicts the network architecture and the losses used to train the model.

new objective to combat label noise, and data augmentation to combine in-distribution and OoD images.

5.2 Proposed System Design

5.2.1 Overview

The network architecture for this method is found in Figure 5.2. The encoder $\mathtt{E}$ embeds an image $\mathbf{x} \in \mathbb{R}^{3\times H\times W}$, resulting in feature map $\mathbf{z} \in \mathbb{R}^{F\times \hat{H}\times \hat{W}}$, where H, W and $\hat{H}, \hat{W}$ are the original and reduced spatial dimensions respectively. As part of the contrastive task, these features $\mathbf{z}$ are projected by a two-layer MLP projection network $\mathtt{g}$, yielding projected features $\hat{\mathbf{z}} \in \mathbb{R}^{F_{\mathtt{g}}\times \hat{H}\times \hat{W}}$, of the same spatial dimensions, but different feature length. A decoder $\mathtt{D}$ takes the features $\mathbf{z}$ as input, and segments them to give logits $\mathbf{l} \in \mathbb{R}^{K\times H\times W}$.

5.2.2 Objective Function

Firstly let us consider the loss L^{SupCon}, presented in [137], which is a supervised contrastive loss applied to image classification and a generalisation of the contrastive loss presented as L^{Con} in Equation (5.2). It *maximises* the cosine similarity between features belonging to the same class, and *minimises* the cosine similarity between features belonging to different classes. It reduces to the instance classification contrastive loss L^{Con} when each image and its augmentation is its own class.

Let's consider a batch element $a \in \mathbb{B}$, where $\mathbb{B} \in \mathbb{R}^{B\times 3\times H\times W}$ is a batch of B images. For a,

all features belonging to the same class as a are considered as positives, which are: z_b where $b \in \mathbb{B}$ and $y_b = y_a$. Then, since all features vectors are normalised as $\|z\|_2 = 1$, it calculates the cosine similarity between the feature for the element in question z_a and the features of the same class z_b as $z_a^\top z_b$. The cosine similarity is also calculated between z_a and features belonging to other classes, i.e. z_c where $c \in \mathbb{B}$ and $y_c \neq y_a$, which are treated as negatives.

These similarities are combined in the form of the softmax function, where the scores to be maximised are on the numerator, while all scores are on the denominator, and all are exponentiated. Therefore the way to minimise the loss is to maximise the similarity between z_a and z_b, and to minimise the similarity between z_a and z_c.

This loss is calculated for a given element in the batch $a \in \mathbb{B}$:

$$L_a^{\text{SupCon}} = \sum_{b \in B : b \neq a, y_b = y_a} -\log \frac{\exp\left(z_a^\top z_b / \tau\right)}{\sum_{c \in B : y_c \neq y_a} \exp\left(z_a^\top z_c / \tau\right) + \sum_{b \in B : b \neq a, y_b = y_a} \exp\left(z_a^\top z_b / \tau\right)} \tag{5.3}$$

The total loss per batch, L^{SupCon}, is calculated over batch elements as:

$$L^{\text{SupCon}} = \sum_{a \in B} \frac{1}{N_{y_a} - 1} L_a^{\text{SupCon}} \tag{5.4}$$

where N_{y_a} is the number of features vectors that belong to the same class as a, therefore $N_{y_a} - 1$ is the number of positives.

Application to OoD Detection

OoD detection can be viewed as a binary classification problem, where the two classes are OoD and in-distribution.

Applying L^{SupCon} to this binary classification problem naïvely would result in clustering both in-distribution and OoD instances in feature space. Given that the OoD dataset is extremely diverse, clustering in this way is very challenging. Therefore, in designing the objective for this method, we instead treat OoD instances only as negatives, such that, in feature space, they are pushed away from positive in-distribution instances, but are not pulled towards each other. The result of this is a one-class contrastive learning objective, with the in-distribution class being the superset of all classes defined in the semantic segmentation problem.

If 'in-distribution' is defined as the positive class ($y = 1$) in the classification problem, we can define the objective, L^{OoDCon} as:

$$L^{\text{OoDCon}} = \sum_{a \in B: y_a = 1}^{N} \frac{1}{N_{y_a=1} - 1} L_a^{\text{OoDCon}} \tag{5.5}$$

$$L_a^{\text{OoDCon}} = \sum_{b \in B: b \neq a, y_b = 1} -\log \frac{\exp\left(z_a^\top z_b / \tau\right)}{\sum\limits_{c \in B: c \neq a} \exp\left(z_a^\top z_c / \tau\right)} \tag{5.6}$$

In this way, the cosine similarity is never maximised between OoD features. As for the maximisation of the similarity of features that are in-distribution, this means that we are training the model to minimise the semantic class differences. By contrast, the supervised learning objective encourages the classes to be maximally separated, therefore it is possible that L^{OoDCon} could decrease segmentation performance. This is, however, not the case, thanks in part due to the projection network, meaning L^{OoDCon} is calculated on features projected from those used in semantic segmentation. This projection is performed by a 2-layer MLP projection network g such that $\hat{\mathbf{z}} = \text{g}(\mathbf{z})$ and $\hat{\mathbf{z}} \in \mathbb{R}^{F_\text{g} \times \hat{H} \times \hat{W}}$, where F_g is the projected feature length. Additionally, [138] suggests that maximising the similarity between the embeddings of all classes leads to useful regularisation and learning of class- and task-agnostic information, which is useful for OoD detection as discussed in Section 3.5.

Therefore, in actual fact, L^{OoDCon} is given by:

$$L^{\text{OoDCon}} = \sum_{a \in \mathbb{B}: y_a = 1}^{N} \frac{1}{N_{y_a=1} - 1} L_a^{\text{OoDCon}} \tag{5.7}$$

$$L_a^{\text{OoDCon}} = \sum_{b \in \mathbb{B}: b \neq a, y_b = 1} -\log \frac{\exp\left(\hat{z}_a^\top \hat{z}_b / \tau\right)}{\sum\limits_{c \in \mathbb{B}: c \neq a} \exp\left(\hat{z}_a^\top \hat{z}_c / \tau\right)} \tag{5.8}$$

Where $\hat{z} \in \mathbb{R}^{F_\text{g}}$ is a projected pixel-wise feature.

Overall Objective

This contrastive objective is combined with the cross-entropy segmentation objective L^s to give the following overall objective, L:

$$L = L^\text{s} + \lambda L^{\text{OoDCon}} \tag{5.9}$$

Where $L^{\text{s}} = \sum_{n=1}^{N} \sum_{i=1}^{H \times W} \bar{y}^* \log p(y|x)$, as described in Section 2.1.3, and $\lambda = 0.1$ to balance the magnitudes of each constituent objective, which we found to work well empirically.

5.2.3 Masking Label Noise

As we use a large-scale unlabelled image dataset, there will possibly be some semantic overlap between this and the smaller labelled dataset. If this is true, L^{OoDCon} then minimises the similarity between features from images that are ultimately very similar, thereby resulting in the learning of features that are less sensitive to semantics. In order to avoid this, we introduce a method for mitigating this problem.

We augment the L^{OoDCon} objective by introducing $\mathbf{M}^{\text{LN}}$ in the following way:

$$L_a^{\text{OoDCon}} = \sum_{b \in \mathbb{B}: b \neq a, y_b = 1} -\log \frac{\exp\left(\hat{z}_a^\top \hat{z}_b / \tau\right)}{\sum\limits_{c \in \mathbb{B}: c \neq a} \mathbf{M}_{ac}^{\text{LN}} \exp\left(\hat{z}_a^\top \hat{z}_c / \tau\right)} \tag{5.10}$$

where this binary mask $\mathbf{M}^{\text{LN}}$, for a given in-distribution element a and OoD element c, is calculated as:

$$\mathbf{M}_{ac}^{\text{LN}} = \begin{cases} 0 & \text{if} \quad \max(\hat{z}_a^\top \hat{z}_c) > t_R \text{ and } y_c = 0 \\ 1 & \text{otherwise} \end{cases} \tag{5.11}$$

This mask is 0 when the similarity between an in-distribution feature vector and OoD feature vector exceeds a threshold t_R. Therefore, t_R is a hyperparameter that rejects a certain proportion R of the OoD features in the batch. Here, R is referred to as the rejection ratio.

In [137], it is noted that the gradient contributions of hard positives and negatives dominate the batch updates. Therefore $\mathbf{M}^{\text{LN}}$ reduces the impact of false hard negatives, i.e. features that are similar according to ground-truth but are falsely associated with different classes in our approximate labelling.

5.2.4 Data Augmentation

As discussed previously, the datasets we have for training are comprised of labelled images that are in-distribution and unlabelled images that are defined as entirely OoD. If the training task were set up such that the model had to classify pixel-wise features as either in-distribution or OoD but the input images were entirely of either class, then this task is no



(a) In-Distribution → In-Distribution



(b) In-Distribution → OoD



(c) OoD → In-Distribution



(d) OoD → OoD

Figure 5.3: Examples of training images generated via our OoDMix data augmentation. Each combination of crop and background is shown across the sub-figures (denoted by *Crop* → *Background*)

more difficult than image-wise OoD detection. Therefore, it is highly likely that the model would, at test time, generalise poorly to difficult pixel-wise distributional uncertainty estimation.

For this reason, the in-distribution and OoD images need to be combined during training in such a way to make the training task more difficult. This is achieved by devising a data augmentation scheme specifically for this setting. It initially comprises of cropping images from one dataset into another, i.e. OoD into in-distribution, in-distribution into OoD.

This, however, only makes the problem slightly more difficult, as this can be solved by detecting the crop and then treating crop and the background separately, and thus is again similar to image-wise OoD detection. Detecting the location of the crop is simple due to the large gradients at the crop boundary, and the large colour-space differences between the in-distribution and OoD dataset.

We take several steps to make this more challenging. Firstly, we can also crop in-distribution into in-distribution and OoD into OoD, as this prevents the detection of a crop's origin from its background. We can also remove the large gradients found at the crop boundaries by combining the crop and background with a Gaussian blur kernel. Lastly, the images are transformed into HSV space, and the colour of the crop is adjusted such that the mean hue and value are the same for the crop as that of the pixels they are replacing in the background.

Each of these steps blend the OoD crop into in-distribution images and vice versa, in such a way that the problem of representing them differently is made much harder. The result is thus a model that can more robustly separate in-distribution and OoD instances. See Figure 5.3 for examples of augmented images.

As per more traditional data augmentation, the final step is then to perform random colour-space augmentations, namely `ColorJitter` in PyTorch [139].

5.3 Experimental Setup

This section describes the specifics of the datasets and neural network architectures used in the experiments presented in Section 5.4.

5.3.1 Datasets

The labelled in-distribution, i.e. source, dataset in this work is Cityscapes [25], as described in Section 4.3.1. The source dataset also contains a small proportion of pixels that are considered as `void`, i.e. they were not annotated due to ambiguity or not belonging to the defined classes. For this reason, they are treated as OoD.

ImageNet is used as the unlabelled OoD, i.e. target, training dataset, which contains 1000 classes with approximately 1000 images for each. Image-wise labels are provided for

these images, however we do not make use of them. This is because we have developed our training algorithm to not require such labelling, allowing it to use unlabelled datasets, which are typically much larger and more diverse, e.g. LAION-5B [135]. Not only this, an image defined as the ImageNet class `hummingbird` can have many pixels belonging to the Cityscapes semantic class `vegetation`, meaning that according to the image-wise labels, the entire image is OoD, however the majority of the pixels are indeed in-distribution. ImageNet also contains images of the Cityscapes known classes, e.g. cars, buildings, and some of these will have similar appearance as those in Cityscapes, which contributes to the label noise discussed and mitigated in Section 5.2.3.

Testing primarily occurs on the driving dataset WildDash [134]. As discussed in Section 4.3.3, WildDash is an extremely diverse driving dataset, and so is the most appropriate dataset to evaluate the distributional uncertainty estimation performance across the breadth of driving data.

We perform additional testing on the SAX test datasets, discussed in more detail in Section 4.2, which are narrower datasets with an increasing distributional shift with respect to Cityscapes in the following order: SAX London, SAX New Forest, SAX Scotland. These test datasets allows us to investigate how the magnitude of distributional shift affects the performance of each method.

5.3.2 Network Architecture

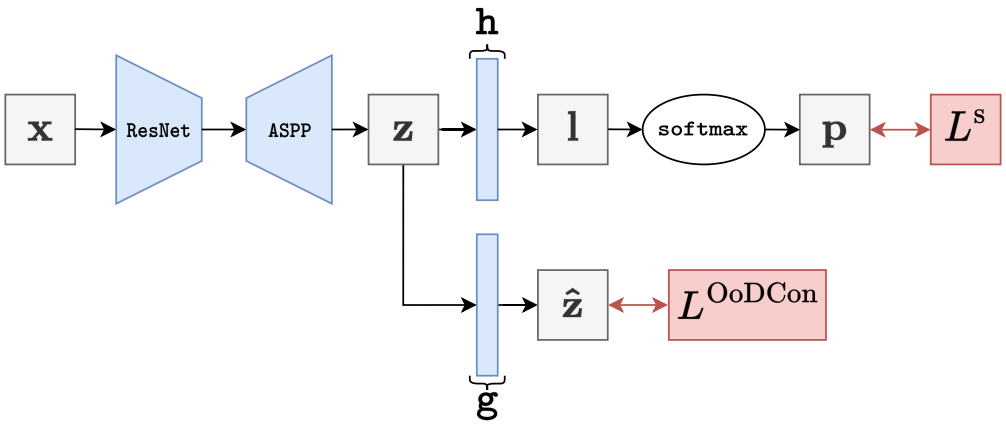


Figure 5.4: A depiction of the network architecture including the specific modules used to define the encoder and decoder: $\texttt{E} = \texttt{ASPP} \circ \texttt{ResNet}$ and $\texttt{D} = \texttt{h}$ respectively.

The segmentation network architecture used in the experiments for this method is DeepLabV3+

[140]. It is split into encoder and decoder, for which a ResNet-18 [35] and Atrous Spatial Pyramid Pooling (ASPP) module is used respectively. In the implementation used in this work[2], an additional segmentation head is used on top of the ASPP decoder.

In contrast to the definitions in [140], we define the encoder as $\texttt{E} = \texttt{ASPP} \circ \texttt{ResNet}$, i.e. as both the ResNet and the ASPP module. The decoder $\texttt{D}$ is then simply the segmentation head, $\texttt{h}$. These changes can be seen in Figure 5.4, which elaborates on Figure 5.2.

The encoder $\texttt{E}$ is therefore represented by the following transform: $\texttt{E} : \mathbb{R}^{3\times H\times W} \rightarrow \mathbb{R}^{256\times \hat{H}\times \hat{W}}$, i.e. the dimensionality of the features are 256 and the downsampled spatial dimensions are $(\hat{H}, \hat{W})$, where $\hat{H} = \frac{H}{4}$ and $\hat{W} = \frac{W}{4}$.

The decoder $\texttt{D}$, implemented as the segmentation head $\texttt{h}$, is a single 1×1 convolutional layer which returns the downsampled logits. Therefore, $\texttt{D} : \mathbb{R}^{256\times \hat{H}\times \hat{W}} \rightarrow \mathbb{R}^{K\times \hat{H}\times \hat{W}}$, which is followed by bilinear upsampling to give the final logits of size $\mathbb{R}^{K\times H\times W}$.

As mentioned in Section 5.2.2, this method also employs a projection network $\texttt{g}$, which is a two-layer MLP, represented by the following transform: $\texttt{g} : \mathbb{R}^{256\times \hat{H}\times \hat{W}} \rightarrow \mathbb{R}^{128\times \hat{H}\times \hat{W}} \rightarrow \mathbb{R}^{F_\texttt{g}\times \hat{H}\times \hat{W}}$, where $F_\texttt{g} = 128$. This projection network $\texttt{g}$ is regularised with a dropout layer with a dropout ratio of 0.5.

By using our definition of $\texttt{E}$, we calculate the contrastive loss on features that are projected from much deeper in the network. This enforces $\mathbf{z}$, which is only a linear transformation away from the logits $\mathbf{l}$, to encode information suited to solving the contrastive task, as well as semantic segmentation-specific information. Empirically, this leads to better distributional uncertainty estimation with $\mathbf{p}_{\text{max}}$, because this additional semantic segmentation-agnostic information helps to spread out probability mass in $\mathbf{p}$ for OoD instances. This is in contrast to using features after the ResNet, i.e. $\texttt{E} = \texttt{ResNet}$, which were empirically less effective as the ASPP module is likely filtering out all but the semantic segmentation-specific information, leading to decreased misclassification detection performance.

5.4 Experiments and Results

This section details the experiments designed to evaluate the method presented in this chapter, and presents the results in Table 5.2, Table 5.1, Table 5.3, Table 5.4 and Figure 5.5.

[2]Based on `https://github.com/qubvel/segmentation_models.pytorch`

5.4.1 Data Augmentation Experiments

In this chapter, we have introduced a data augmentation scheme that combines in-distribution and OoD images into the same image, in such a way that the task of separating the two is made more difficult. By increasing the difficulty of the OoD detection training task, this chapter has argued that we can expect increased robustness from our model for this task.

We empirically determine that this is true by training a model on three variants of data augmentation: `ImageWise`, `CutMix` and `OoDMix`. In `ImageWise`, in-distribution and OoD images are included as separate images in a training batch, as is seen in [51], [128]. `CutMix` is based on a regularisation method presented in [141] and crops images into other images, but does not align them in colour-space or blur the edges of the boundaries between crop and background. Finally, `OoDMix` is our proposed data augmentation scheme, described in Section 5.2.4.

5.4.2 Data Augmentation Results

Using our proposed training algorithm, we train models using each of the data augmentation schemes described in Section 5.4.1. In Table 5.1, we can see that using `OoDMix` results in significantly better misclassification detection in terms of AUPR for each of the distributionally shifted domains compared to `CutMix` and `ImageWise`. The same is also true for $\mathrm{MaxF}_{1/2}$ and $\mathrm{MaxA}_{\mathrm{MD}}$, as shown in Table 5.3 and Table 5.4. As an example, for WildDash, the $\mathrm{MaxF}_{1/2}$ for `OoDCon-OoDMix` is 0.801 versus 0.634 and 0.670 for `OoDCon-Cutmix` and `OoDCon-ImageWise` respectively, and for the much higher $\mathrm{p(a, c)}$ of 0.432 versus 0.234 and 0.223. These results show that increasing the difficulty of the training task via the data augmentation scheme proposed in this chapter yields a model that is better able to discern between in-distribution and OoD pixels, and therefore detect error on distributionally shifted test datasets.

5.4.3 Objective Function Experiments

In order to investigate the effectiveness of our proposed objective function L^{OoDCon}, we compare it to baselines which use the same data augmentation scheme. The first baseline is `KL`, seen in [129] and Equation (5.1). Secondly, we consider `VoidSeg`, where the model outputs

additional predictions for the `void` class, and is trained to segment the data augmented OoD pixels as `void`. Instead of $\mathbf{p}_{\max}$, this method uses $(1 - p(y = \texttt{void}|x))$ as the model confidence. Finally, this method also considers a network trained only with the cross-entropy segmentation loss L^{s}, named `Vanilla`.

Additionally, we consider two variants of our proposed objective: `OoDCon` and `OoDCon-LN`, with the latter having a rejection ratio R of 0.2, and the former of 0, i.e. every element of $\mathbf{M}^{\text{LN}}$ is 1, and thus label noise is in no way mitigated.

5.4.4 Objective Function Results

Table 5.1, Table 5.3 and Table 5.4, suggest that `OoDCon-LN` is the best performing objective function for SAX New Forest, SAX Scotland and WildDash, which are the most distributionally shifted domains. For WildDash, the $\text{MaxF}_{1/2}$ @ $\text{p}(\text{a}, \text{c})$ of `OoDCon-LN-OoDMix` is 0.809 @ 0.439, versus 0.801 @ 0.432 for `OoDCon-OoDMix`, therefore, for this threshold, more segmentation error is being detected and fewer pixels are being rejected as unknown as a result of the label noise mask.

However, `OoDCon`, i.e. when $R = 0$, performs better in terms of $\text{MaxF}_{1/2}$ and MaxA_{MD} for SAX London, which is relatively similar to Cityscapes. This is possibly due to the inherent trade-off in using the label noise mask, which is between the quantity of label noise and the magnitude of the enforced separation between the in-distribution and OoD dataset. When R is lower, the effective label noise for the task is higher, however there is a greater enforced separation between the two datasets. In the case of smaller distributional shifts on the test dataset, it is perhaps more important to enforce a more aggressive separation between datasets during training than to remove label noise. By contrast, for larger distributional shifts on the test dataset, it is perhaps more important to remove label noise as there already exists a larger separation between source and target domains.

5.4.5 Data Diversity Experiments

In this chapter, we have opted to use a large-scale image dataset due to the discussion in Section 5.1.2, which demonstrates the enormity of the set of possible OoD images. In order to empirically determine the benefit of this diversity in the OoD training dataset, we use a

subset of the ImageNet dataset.

This dataset includes 10 classes instead of 1000, resulting in a dataset of $13,000$ images. The 10 classes were selected as the closest classes to those in the CIFAR-10 dataset (as determined by [142]), which is a commonly used small-scale image classification dataset. We use this subset of ImageNet instead of CIFAR-10, due to the latter's significantly smaller image resolution. The classes considered are as follows: `airliner`, `beach wagon`, `hummingbird`, `siamese cat`, `ox`, `golden retriever`, `tailed frog`, `zebra`, `container ship`, `trailer truck`.

This dataset is used with `OoDMix` data augmentation for direct comparison with the `OoDCon-OoDMix` experiments, and the results for this experiment can be found with the name `SubSet`.

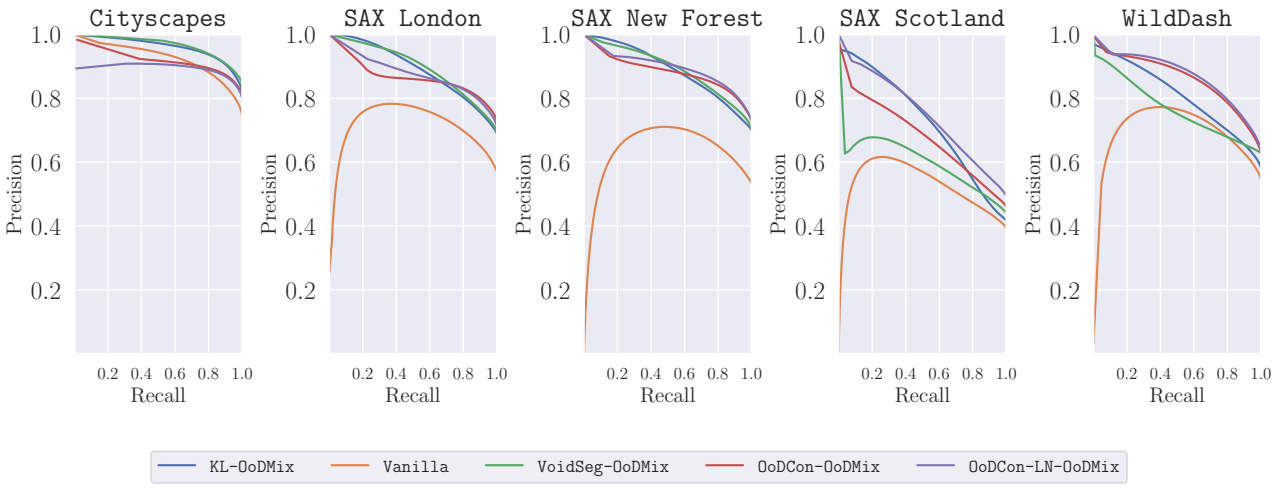
5.4.6 Data Diversity Results

By comparing results for `OoDCon-OoDMix` and `OoDCon-SubSet`, we can see that adding diversity to the OoD training dataset improves distributional uncertainty estimation. For almost every distributionally-shifted domain, the addition of the extra 990 classes improved almost every misclassification detection metric. Specifically, for WildDash, the $\mathrm{MaxF}_{1/2}$ @ $\mathrm{p}(\mathrm{a}, \mathrm{c})$ was 0.801 @ 0.432 for `OoDCon-OoDMix`, while it was only 0.761 @ 0.370 for `OoDCon-SubSet` with the reduced dataset, therefore fewer inaccurate pixels are detected and fewer pixels were confidently and correctly segmented by the latter.

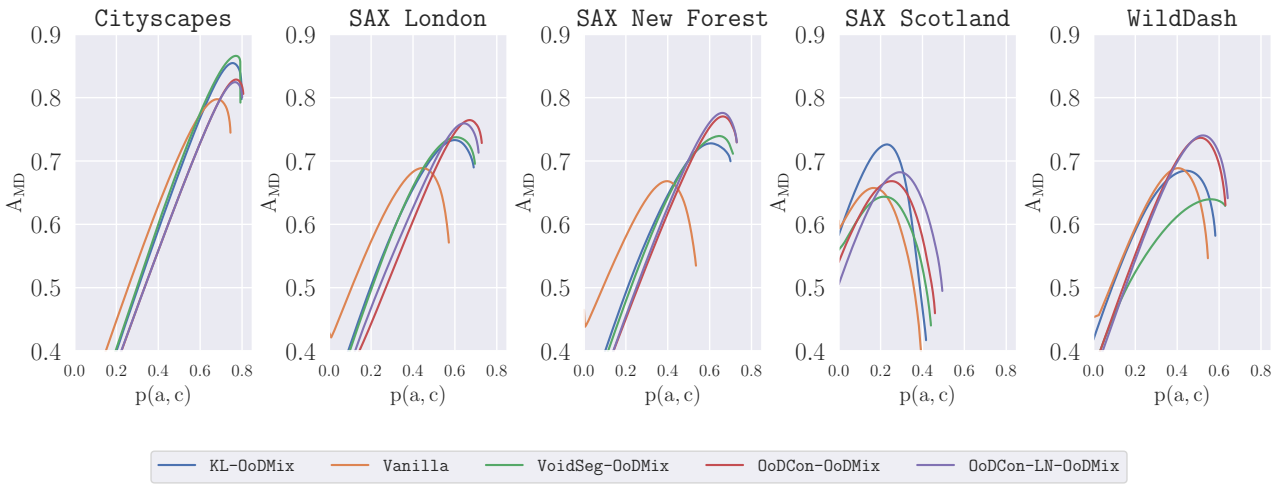
The performance using the ImageNet subset still outperformed `Vanilla` on all metrics, showing that using the reduced ImageNet dataset was beneficial, however it is simply less beneficial than using a larger-scale dataset.

5.5 Conclusion

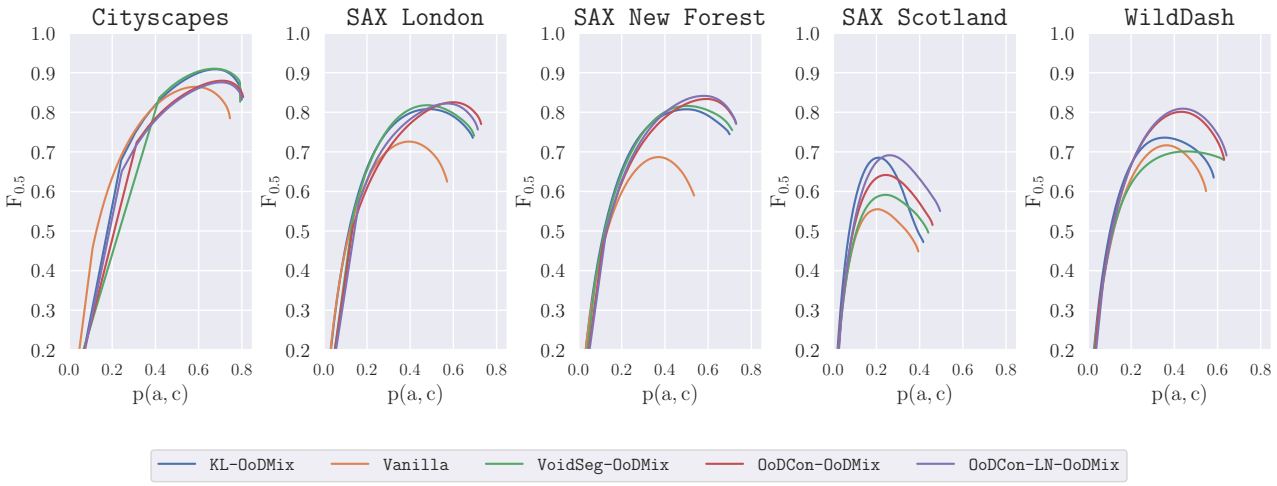
This chapter presented a method that posed distributional uncertainty estimation as a simplification of a large-scale image recognition problem, and correspondingly leveraged a large-scale image dataset. The method defines a training task that combines in-distribution and OoD images using data augmentation, and then a segmentation network is trained to



(a) Precision versus Recall



(b) A_{MD} versus p(a, c)



(c) $F_{1/2}$ versus p(a, c)

Figure 5.5: Misclassification detection performance evaluated for three metrics over a range of domains. The left-most domain is the in-distribution Cityscapes domain, and provides the labelled training dataset for learning semantic segmentation. From left to right, the SAX domains are increasingly distributionally-shifted from Cityscapes, and so these plots measure the models' ability to detect regions of segmentation error which are increasingly arising from left to right due to distributional shift. The WildDash plots present the models' quality of uncertainty estimation on a very diverse distributionally-shifted driving dataset.

		AUPR				
	Method	CS	LDN	NF	SCOT	WD
Baselines	Vanilla	0.926	0.717	0.635	0.527	0.712
	KL-OoDMix	**0.960**	**0.891**	0.889	0.728	0.810
	VoidSeg-OoDMix	0.941	0.896	0.892	0.602	0.767
Ablations	KL-ImageWise	0.907	0.636	0.434	0.239	0.610
	KL-CutMix	0.938	0.545	0.392	0.200	0.528
	KL-Subset	0.935	0.856	0.865	0.680	0.775
	OoDCon-ImageWise	0.908	0.767	0.714	0.472	0.711
	OoDCon-CutMix	0.940	0.573	0.488	0.255	0.680
	OoDCon-Subset	0.914	0.843	0.810	0.639	0.820
Ours	OoDCon-OoDMix	0.904	0.864	0.882	0.684	0.862
	OoDCon-LN-OoDMix	0.893	0.878	**0.896**	**0.750**	**0.872**

Table 5.1: Misclassification Detection performance in terms of AUPR for a range of test domains.

differentiate between the two. Although the training task is OoD detection, the trained segmentation networks are evaluated on their ability to perform misclassification detection, i.e. the test setting considers uncertainty estimation, rather than OoD detection.

The experiments in this chapter show that the following factors are important for learning pixel-wise distributional uncertainty estimation: (1) the magnitude of the visual difference between in-distribution and OoD pixels, and (2) the diversity of the OoD dataset.

The first factor is supported by the literature [128], [129], and makes intuitive sense, as when the in-distribution and OoD are more similar, the training task of separating them is more challenging, yielding a more robust model. As for the second factor, adding diversity to a dataset is a common method for achieving a more general model, however, this is even more important for this problem due to the enormous diversity of the set of all OoD instances, as discussed in Section 5.1.2.

By choosing a large-scale image dataset in this chapter, we have optimised for the second factor and have tried to optimise for the first factor using data augmentation. Nevertheless, we hypothesize that there are better ways of optimising for the first factor, i.e. methods for defining a more challenging training task of this kind. As discussed in Section 1.4, this hypothesis is a key driver of work in Chapter 6 and Chapter 7, in which distributionally shifted driving data is used instead of a large-scale image recognition dataset.

		AUROC				
	Method	CS	LDN	NF	SCOT	WD
Baselines	`Vanilla`	0.824	0.723	0.686	0.674	0.728
	`KL-OoDMix`	0.874	0.777	0.765	**0.760**	0.751
	`VoidSeg-OoDMix`	**0.889**	0.785	0.769	0.677	0.653
Ablations	`KL-ImageWise`	0.827	0.738	0.653	0.587	0.713
	`KL-CutMix`	0.847	0.732	0.682	0.641	0.748
	`KL-Subset`	0.850	0.726	0.751	0.754	0.736
	`OoDCon-ImageWise`	0.857	**0.790**	0.764	0.705	0.762
	`OoDCon-CutMix`	0.848	0.697	0.679	0.600	0.717
	`OoDCon-Subset`	0.788	0.740	0.726	0.736	0.775
Ours	`OoDCon-OoDMix`	0.759	0.732	0.761	0.713	0.795
	`OoDCon-LN-OoDMix`	0.745	0.758	**0.782**	0.738	**0.799**

Table 5.2: Misclassification Detection performance in terms of AUROC for a range of test domains.

		$\mathrm{MaxF}_{1/2}$ @ $\mathrm{p}(\mathrm{a}, \mathrm{c})$				
	Method	CS	LDN	NF	SCOT	WD
Baselines	`Vanilla`	0.864 @ 0.585	0.726 @ 0.395	0.687 @ 0.370	0.555 @ 0.205	0.717 @ 0.364
	`KL-OoDMix`	0.909 @ 0.678	0.808 @ 0.485	0.808 @ 0.501	0.685 @ 0.210	0.736 @ 0.354
	`VoidSeg-OoDMix`	**0.910** @ 0.672	0.818 @ 0.480	0.816 @ 0.503	0.592 @ 0.241	0.701 @ 0.456
Ablations	`KL-ImageWise`	0.844 @ 0.506	0.603 @ 0.180	~~0.428 @ 0.100~~	~~0.261 @ 0.048~~	0.572 @ 0.156
	`KL-CutMix`	0.872 @ 0.561	~~0.522 @ 0.082~~	~~0.397 @ 0.060~~	~~0.253 @ 0.016~~	~~0.524 @ 0.085~~
	`KL-Subset`	0.900 @ 0.678	0.793 @ 0.528	0.796 @ 0.497	0.637 @ 0.184	0.710 @ 0.312
	`OoDCon-ImageWise`	0.888 @ 0.592	0.722 @ 0.236	0.676 @ 0.233	~~0.457 @ 0.092~~	0.670 @ 0.223
	`OoDCon-CutMix`	0.891 @ 0.638	0.552 @ 0.160	0.482 @ 0.104	~~0.271 @ 0.034~~	0.634 @ 0.234
	`OoDCon-Subset`	0.884 @ 0.690	0.805 @ 0.544	0.791 @ 0.518	0.618 @ 0.176	0.761 @ 0.370
Ours	`OoDCon-OoDMix`	0.88 @ 0.708	**0.825** @ 0.599	0.834 @ 0.594	0.642 @ 0.245	0.801 @ 0.432
	`OoDCon-LN-OoDMix`	0.876 @ 0.705	0.822 @ 0.573	**0.841** @ 0.579	**0.691** @ 0.263	**0.809** @ 0.439

Table 5.3: Misclassification Detection performance in terms of $\mathrm{MaxF}_{1/2}$ @ $\mathrm{p}(\mathrm{a}, \mathrm{c})$ for a range of test domains. Entries are struck out if their $\mathrm{p}(\mathrm{a}, \mathrm{c})$ is too low, as this threshold value leads to the assignment of class to too few pixels.

		$\mathrm{MaxA_{MD}}$ @ $\mathrm{p(a, c)}$				
	Method	CS	LDN	NF	SCOT	WD
Baselines	`Vanilla`	0.798 @ 0.682	0.689 @ 0.444	0.668 @ 0.396	0.657 @ 0.164	0.689 @ 0.405
	`KL-OoDMix`	0.855 @ 0.755	0.733 @ 0.599	0.728 @ 0.606	**0.726 @ 0.231**	0.685 @ 0.444
	`VoidSeg-OoDMix`	**0.866** @ 0.770	0.738 @ 0.606	0.739 @ 0.645	0.644 @ 0.217	0.640 @ 0.562
Ablations	`KL-ImageWise`	0.771 @ 0.603	0.707 @ 0.170	~~0.732 @ 0.031~~	~~0.825 @ 0.001~~	0.701 @ 0.134
	`KL-CutMix`	0.799 @ 0.657	~~0.773 @ 0.064~~	~~0.795 @ 0.020~~	~~0.893 @ 0.000~~	~~0.790 @ 0.065~~
	`KL-Subset`	0.847 @ 0.755	0.722 @ 0.610	0.724 @ 0.595	0.721 @ 0.188	0.672 @ 0.368
	`OoDCon-ImageWise`	0.818 @ 0.674	0.738 @ 0.266	0.713 @ 0.253	~~0.769 @ 0.041~~	0.715 @ 0.241
	`OoDCon-CutMix`	0.830 @ 0.716	0.692 @ 0.129	~~0.736 @ 0.061~~	~~0.836 @ 0.006~~	0.671 @ 0.238
	`OoDCon-Subset`	0.829 @ 0.758	0.744 @ 0.613	0.732 @ 0.582	0.715 @ 0.174	0.711 @ 0.437
Ours	`OoDCon-OoDMix`	0.829 @ 0.771	**0.765 @ 0.670**	0.770 @ 0.663	0.668 @ 0.252	0.737 @ 0.511
	`OoDCon-LN-OoDMix`	0.824 @ 0.767	0.759 @ 0.644	**0.776 @ 0.662**	0.682 @ 0.292	**0.740 @ 0.524**

Table 5.4: Misclassification Detection performance in terms of $\mathrm{MaxA_{MD}}$ @ $\mathrm{p(a, c)}$ for a range of test domains. Entries are struck out if their $\mathrm{p(a, c)}$ is too low, as this threshold value leads to the assignment of class to too few pixels.

Chapter 6

Learning Uncertainty Estimation from Uncurated Domain Data

Contents

6.1 Motivation 103

6.1.1 Using Unlabelled Out-of-Distribution Driving Data 103

6.1.2 Introduction to γ-SSL 105

6.1.3 How to tailor self-supervised methods for uncertainty estimation? . 106

6.2 Preliminaries 107

6.2.1 Segmentation via Prototypes 107

6.2.2 Uncertainty Estimation via a Feature-Space Threshold 108

6.3 Crop & Resize Data Augmentation 109

6.3.1 Method 109

6.4 Training Architecture 110

6.5 Training Objective 112

6.5.1 Calculating γ 113

6.5.2 Learning $\texttt{E}$ 113

6.5.3 Learning the Task 115

6.5.4 Preventing Feature Collapse 115

6.6 Training Procedure 117

6.6.1 Model Pretraining . . . 117
6.6.2 Domain-based Curriculae . . . 118
6.7 Network Architecture . . . 119
6.8 Baselines . . . 119
6.9 Evaluating uncertainty estimation on narrow target domains . . . 121
6.9.1 Source: Cityscapes, Target: SAX Test Datasets . . . 122
6.9.2 Effect of distributional shift . . . 124
6.9.3 Source: Cityscapes, Target: KITTI & BDD . . . 126
6.9.4 Source: BDD, Target: SAX Test Datasets . . . 128
6.10 Evaluating uncertainty estimation on a general target domain . . . 132
6.10.1 Target: WildDash . . . 132
6.11 Miscellaneous experiments . . . 133
6.11.1 Calculation of the optimal threshold . . . 134
6.11.2 Calculating thresholds across domains . . . 135
6.11.3 Latency Evaluation . . . 136
6.12 Qualitative Results . . . 137
6.13 Ablation Studies . . . 139
6.13.1 Estimating Uncertainty via Distance to Prototypes . . . 139
6.13.2 Importance of target domain images . . . 139
6.13.3 Importance of $\mathbf{M}^{\gamma}$ in the training objective . . . 141
6.13.4 Importance of Branch Asymmetry . . . 141
6.13.5 Importance of L^{u} and L^{p} . . . 142
6.13.6 Importance of Crop-and-Resize Data Augmentation . . . 142
6.13.7 Importance of using hard $\mathbf{M}^{\gamma}$ . . . 143
6.13.8 Possibility of class-wise thresholds . . . 143
6.13.9 The need for large batch sizes to calculate prototypes . . . 144
6.14 Conclusion . . . 146

This chapter presents a method that learns distributional uncertainty estimation through training on OoD driving data, which contains in-distribution, near-distribution and OoD instances all within a single image. This is in contrast to Chapter 5, where a large-scale image dataset was used, where the entirety of every image was treated as OoD. This new type of OoD data is challenging to train with, however the previous chapter suggests that a smaller distributional shift leads to more generality and robustness than using a OoD dataset with a large distributional shift.

This method was first presented in:

- D. Williams, D. De Martini, M. Gadd, and P. Newman, "Mitigating Distributional Shift in Semantic Segmentation via Uncertainty Estimation from Unlabelled Data", IEEE Transactions on Robotics (T-RO), 2024.

6.1 Motivation

The method in this chapter is influenced by insights from Chapter 5, where it was shown that the nature of the data used to learn distributional uncertainty estimation, i.e. the target domain dataset, is important.

In the previous chapter, the target domain dataset was a large-scale image recognition dataset, where many of the images were very different to the source domain, i.e. the labelled training dataset. We found in the literature, and through experiment, that the more distinct the source and target domains are, the lower the quality of learned distributional uncertainty estimation as a result of a less robust separation of in-distribution and OoD data. These findings motivated us to look at what alternatives could be used as the target domain dataset.

6.1.1 Using Unlabelled Out-of-Distribution Driving Data

The literature in Section 3.5.6 and empirical evidence discussed in Chapter 5 advocate the use of a training dataset containing so-called 'near-distribution' instances, i.e. instances that are OoD, but similar to the in-distribution data in terms of semantics or appearance. This

type of training dataset allows for more general and robust distributional uncertainty estimation.

For our setting, a type of data that contains 'near-distribution' instances alongside in-distribution and OoD is distributionally-shifted driving data. For images collected by road vehicles, the majority of pixels often remain in-distribution, e.g. roads, vegetation, sky can often look similar in different lighting and geographic locations. However, due to the setting's outdoor and open nature, the images also contain many instances that are visually or semantically different, which are adjacent to the in-distribution pixels. Distributionally-shifted driving images from a given target domain therefore perhaps provide the opportunity to learn a representation that is more suited to general distributional uncertainty estimation on a number of different target domains.

Another benefit of this type of training on images from a given target domain, is that the uncertainty estimation performance is likely to be particularly good in that target domain due to the reduction in gap between training and testing. Therefore, if one type of distributional shift from the robot's ODD is likelier to occur, e.g. the same geographic location, but in bad weather, or the same conditions but slightly outside the geographic location of the ODD, the model can be made especially good at detecting error due to this shift.

Distributionally-shifted driving data sits in between the two types of data seen in aleatoric uncertainty estimation and learned OoD detection. For aleatoric uncertainty estimation methods, discussed in Section 3.4.5, the model would be trained to detect error on in-distribution driving data, i.e. there is no distributional shift or target domain considered. By contrast, as discussed in Section 3.5.6 and practised in Chapter 5, for learned OoD detection, distributionally shifted data is considered, however it is no longer driving data. The argument made in this section is that by drawing on aspects from both methods, and training using distributionally-shifted driving data, uncertainty estimation can be improved both on the specific target domain chosen, as well as more generally for any given target domain.

Challenges

Using data with in-distribution, near-distribution and OoD instances all within the same image is desirable, however it also introduces a significant challenge. For the learned methods

discussed, pixel-wise ground-truth for the training images is required. For aleatoric uncertainty estimation, ground-truth is used to measure error, and thus determine the image regions for which high uncertainty is appropriate. For learned OoD detection, pixel-wise ground truth is needed to determine which pixels are in-distribution and which are OoD.

This requirement for pixel-wise ground-truth would severely limit the diversity of the dataset, which, in turn, limits the quality of distributional uncertainty estimation, as discussed in Section 5.4. For this reason, this chapter presents a method to train a model to learn uncertainty estimation using *unlabelled* images, in contrast to methods discussed in Section 3.4.5 and Section 3.5.6. As a result, this type of method is tractable in robotics settings, because all that is required for the collection of a large training dataset is: the availability of a robot, and access to domains which are distinct from the source domain.

6.1.2 Introduction to γ-SSL

The first thing to consider is how to generate a signal, without the use of labels, that can be used train the model such that image pixels that lead to segmentation error are separable from pixels that are correctly segmented.

One productive vein of research in computer vision uses self-supervised learning to learn a representation without labels, as discussed previously in Section 3.5.3 and Chapter 5. In self-supervised learning methods, models are often trained with the objective of learning a semantic representation, without these semantic classes being defined by annotation, and instead by using data augmentation. This has been shown to work extremely well in a large-scale setting, where the neural networks and datasets are both very large.

One method for evaluating the 'quality' of the learned representation is to perform linear probing experiments. These experiments train a linear layer on top of a learned representation in a supervised manner, and show very high performance on large-scale image classification such as ImageNet [31], [32], [106], [108]. This shows that this type of self-supervised training is very effective at training large neural networks to cluster image data, such that the clusters are linearly separable and semantically meaningful.

For this chapter, the significance of this is that these works have showed that using data augmentation is a very effective way of circumventing the requirement for ground-truth

annotation. In supervised learning, labels are used to measure classification error, which the model can then be trained to minimise; however in self-supervised learning, the data augmentation is instead approximately measuring classification error. Instead of measuring and *minimising* error, this raises the possibility of using data augmentation to measure and *detect* error.

6.1.3 How to tailor self-supervised methods for uncertainty estimation?

In this chapter, we consider the self-supervised task of crop-and-resize data augmentation. The method works by having one image, which you crop, resize to its previous size, then apply a colour-space transform. This is done such that the image-wise semantic content is preserved between the two images (or views), while also looking significantly different. Then, by training a model to learn a feature space which is invariant to this form of data augmentation, the assumption is that the model produces embeddings which describes semantics. A model's inconsistency in embedding is measured, and the SSL objective minimises this inconsistency.

A method for learning uncertainty estimation can therefore be devised by (1) measuring a model's inconsistency with respect to data augmentation, (2) designing a mechanism whereby the model can express uncertainty, and (3) defining an objective that trains the model to output high uncertainty for pixels where the model is inconsistent and low uncertainty when the model is consistent.

The aims of (1) and (2) are common between learning uncertainty estimation and SSL, however the implementation choices between the method in this chapter and SSL literature are different.

For (1), instead of measuring inconsistency in the model embeddings, i.e. considering the distance between high-dimensional features, our method measures the inconsistency between segmentations. This is because segmentation inconsistency is a more useful representation of segmentation error than the more abstract embedding inconsistency. As will be described in Section 6.5, segmentation inconsistency is represented by a binary mask $\mathbf{M}^{c}$ relating to whether the most probable classes are equal or not, in a way that is not possible with embedding consistency without introducing a threshold.

For (2), we have chosen to design a model that outputs uncertainty as distance in feature space, however it is the distance from a given pixel-wise feature to its nearest 'prototype'. The prototypes are calculated from the labelled source images, and are the mean unit feature vectors for each of the semantic classes. In this way, the distance between a pixel-wise feature vector and its nearest prototype describes the difference between this image region and the labelled training data for its most similar class. If this distance is large, then this image region is unlike anything seen in the labelled training images. This is similar to the idea of using a per-class Gaussian Mixture Model (GMM) seen in DUMs in Section 3.5.2, where the covariance matrix is fixed to the identity matrix.

How this measure of uncertainty is trained to be predictive of segmentation inconsistency for (3) is discussed in Section 6.5.

6.2 Preliminaries

The ultimate aim of this method is to train a neural network that can: (1) segment an image $\mathbf{x} \in \mathbb{R}^{3\times H\times W}$ into a set of K semantic classes, giving $\mathbf{y} \in \{n \in \mathbb{Z} \mid 1 \leq n \leq K\}^{H\times W}$, where each pixel is assigned to a member of the set of known classes is given by $\mathbb{K} = \{k_1, \dots k_K\}$, and (2) perform uncertainty estimation and yield an uncertainty mask, $\mathbf{M}^\gamma \in \{0, 1\}^{H\times W}$ which assigns a given pixel to 1 for certain, and 0 for uncertain. As described in Chapter 4, the model will be evaluated in terms of its misclassification detection performance, where, for the ideal model at a pixel location i, $\mathbf{M}^\gamma_i = 1$ when $\mathbf{y}_i = \mathbf{y}^*_i$, and $\mathbf{M}^\gamma_i = 0$ when $\mathbf{y}_i \neq \mathbf{y}^*_i$.

6.2.1 Segmentation via Prototypes

An image is embedded using both an encoder $\mathtt{E} : \mathbb{R}^{3\times H\times W} \to \mathbb{R}^{F\times h\times w}$ and a projection network $\mathsf{g}_\rho : \mathbb{R}^{F\times h\times w} \to \mathbb{R}^{F\times h\times w}$, where F is the feature length, and h and w are the downsampled spatial dimensions.

From these embeddings, prototypes $\pi_\mathrm{S} \in \mathbb{R}^{F\times K}$ are calculated which represent the class-wise mean feature vector. These are calculated using a batch of N images $\mathbf{X}_\mathrm{S} \in \mathbb{R}^{N\times 3\times W\times H}$, with corresponding one-hot labels $\bar{\mathbf{Y}}^*_\mathrm{S} \in \mathbb{R}^{N\times h\times w\times K}$, which have been downsampled to the same spatial dimensions as the embeddings. The batch of images $\mathbf{X}_\mathrm{S}$ are embedded as above

to given a batch of embeddings, $\mathbf{Z}_S \in \mathbb{R}^{N\times h\times w\times F}$. The prototypes can then be calculated for each class as:

$$\pi_S = \frac{\mathbf{Z}_S^\top \bar{\mathbf{Y}}_S^*}{\|\mathbf{Z}_S^\top \bar{\mathbf{Y}}_S^*\|_2} \tag{6.1}$$

These prototypes can then be used to segment an image by calculating the similarity between each pixel feature of a given target image and the prototypes. The low-resolution logits $\tilde{\mathbf{l}}$ of the i-th pixel embedding $\mathbf{z}_i \in \mathbb{R}^{F\times 1}$ is given by:

$$\tilde{\mathbf{l}}_i = \mathbf{z}_i^\top \pi_S \tag{6.2}$$

As both $\mathbf{z}_i$ and each of π_S are of unit-length, the logits are the cosine similarities between a pixel feature and each of the class prototypes.

In order to produce full-resolution segmentation maps, the downsampled logits $\tilde{\mathbf{l}} \in \mathbb{R}^{h\times w\times K}$ are bilinearly upsampled to $\mathbf{l} \in \mathbb{R}^{H\times W\times K}$. Finally, per-pixel categorical distributions, $\mathbf{p} \in [0,1]^{H\times W\times K}$, and segmentation map $\mathbf{y} \in \{n \in \mathbb{Z} \mid 1 \leq n \leq K\}^{H\times W}$ are calculated as:

$$\mathbf{p} = \mathrm{softmax}_\tau(\mathbf{l}),\ \mathbf{y} = \mathrm{argmax}(\mathbf{l}) \tag{6.3}$$

6.2.2 Uncertainty Estimation via a Feature-Space Threshold

Distributional uncertainty can be thought of as the uncertainty of a pixel not being assigned correctly to one of the known classes, i.e. $p(y \notin \mathbb{K}|x)$. Therefore we need a way to relate distance (or in this case similarity) in feature space, i.e. the logit values, to $p(y \notin \mathbb{K}|x)$. This method represents uncertainty by concatenating a threshold parameter γ to the logits:

$$p(y|x) = \mathrm{softmax}_\tau(\mathbf{l}_i \oplus \gamma) \in \mathbb{R}^{K+1} \tag{6.4}$$

Where $p(y|x) = [p(y = k_1), \ldots, p(y = k_K|x), p(y \notin \mathbb{K}|x)] = \mathrm{softmax}_\tau([\mathbf{l}_{i,1},\ \ldots\ , \mathbf{l}_{i,K}, \gamma])$

If we consider the maximum logit prior to concatenation, $\max(\mathbf{l}_i)$, this is the similarity between the pixel feature and its nearest prototype, which is a measure of the model's confidence. However, after concatenation, if γ exceeds this estimate of the model confidence then $\mathrm{argmax}(\mathbf{l}_i \oplus \gamma) = K{+}1$. This means that instead of being assigned to a known class $k \in \mathbb{K}$,

the pixel is assigned to 'unknown'.

Therefore, γ is operating as a threshold on the model's confidence, and represents a region around each prototype, within which the model is confident, and outside of which the model is uncertain. Formally, we can calculate the aforementioned uncertainty mask in this manner as:

$$\mathbf{M}_i^{\gamma} = 1 - \mathbb{1}[\text{argmax}(\mathbf{l}_i \oplus \gamma) = K{+}1] \tag{6.5}$$

Where $\mathbb{1}$ is the indicator function.

6.3 Crop & Resize Data Augmentation

In lieu of supervision, data augmentation is used to generate an approximation of where errors likely occur. Typically, data augmentation takes the form of applying both a crop-and-resize transform, along with a colour-space transform. In the context of learning a representation for image classification, it is important that, while different in appearance, both crops primarily describe one semantic class. Therefore, both crops should be embedded to the same location in a semantic feature space.

The difference in our context is that our task requires evaluating a model's feature space in a pixel-wise manner. This, therefore, requires us to compare the semantics between the two crops in a pixel-wise manner. We devise the following method, described in Section 6.3.1 and Figure 6.1.

6.3.1 Method

Firstly, an unlabelled image $\mathbf{x}$ is randomly cropped with the corresponding transform $\mathcal{T}_1^G$, producing a global crop. This is then separately transformed by both $\mathcal{T}_1^L$ or $\mathcal{T}_2^L$ giving two crops, $\bar{\mathbf{x}}$ and $\bar{\mathbf{x}}'$. $\mathcal{T}_1^L$ and $\mathcal{T}_2^L$ are sampled such that one is always the identity transform, while the other is a crop-and-resize transform. Different colour-space transforms, $\mathcal{C}_1$ and $\mathcal{C}_2$, are then applied to each of these two crops. Intuitively, the result of these operations is to give two crops, where one is a colour-transformed crop of the other, which has been resized to match its spatial dimensions.

After this, two functions, $\mathtt{f}$ and $\mathtt{g}$, segment each of the generated crops. Pixel-wise

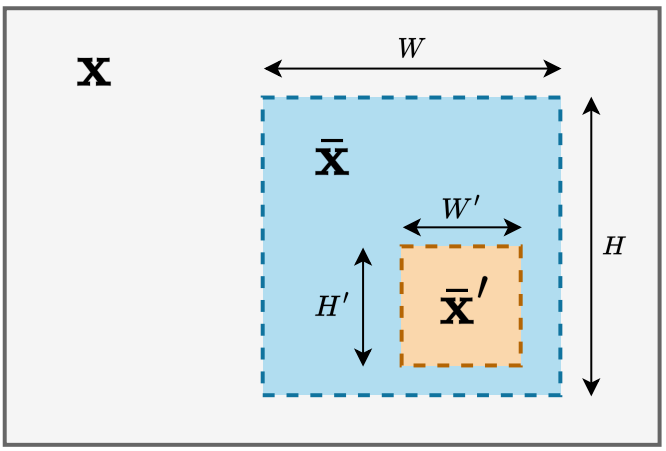


Figure 6.1: A depiction of the crop-and-resize scheme used. Firstly, a global crop $\bar{\mathbf{x}}$ is obtained with spatial dimensions (H, W), which are image spatial dimensions used for the entirety of this chapter. Subsequently, a local crop $\bar{\mathbf{x}}'$ is obtained from within the global crop with initial spatial dimensions (H', W'), before being bilinearly interpolated to (H, W). In this way, we have two images of the same spatial dimensions representing two distinct views of the instances contained within the region of $\bar{\mathbf{x}}'$, where $\bar{\mathbf{x}}$ contains significantly more context than $\bar{\mathbf{x}}'$. $\bar{\mathbf{x}} = \mathcal{T}_1^{\mathcal{L}} \circ \mathcal{T}_1^{\mathcal{G}}(\mathbf{x})$ and $\bar{\mathbf{x}}' = \mathcal{T}_2^{\mathcal{L}} \circ \mathcal{T}_1^{\mathcal{G}}(\mathbf{x})$ if $\mathcal{T}_1^{\mathcal{L}}$ and $\mathcal{T}_2^{\mathcal{L}}$ are sampled as the identity and local crop-and-resize respectively.

aligned segmentations are then generated by applying the local crop-and-resize operation that was not applied before, i.e.:

$$\begin{aligned} \mathbf{l}' &= \mathcal{T}_2^{\mathcal{L}} \circ \mathtt{f} \circ \mathcal{C}_1 \circ \mathcal{T}_1^{\mathcal{L}} \circ \mathcal{T}_1^{\mathcal{G}}(\mathbf{x}) \\ \mathbf{l} &= \mathcal{T}_1^{\mathcal{L}} \circ \mathtt{g} \circ \mathcal{C}_2 \circ \mathcal{T}_2^{\mathcal{L}} \circ \mathcal{T}_1^{\mathcal{G}}(\mathbf{x}) \end{aligned} \tag{6.6}$$

Logits $\mathbf{l}$ and $\mathbf{l}'$ are pixel-wise aligned and, from this, the training objective is designed.

6.4 Training Architecture

Using the above data augmentation scheme, training entails comparing the segmentations of two crops of the same image. In this scheme, we have described two segmentation functions $\mathtt{f}$ and $\mathtt{g}$ that define two branches, each of which segments one of the crops. These functions could represent the neural networks with the same architecture, however in this method they do not. In this method, the branches are defined by the following, and described in Figure 6.2 and Figure 6.3:

$$\mathtt{f}(\cdot) = \mathtt{f}_\psi \circ \mathtt{E}(\cdot) \tag{6.7}$$

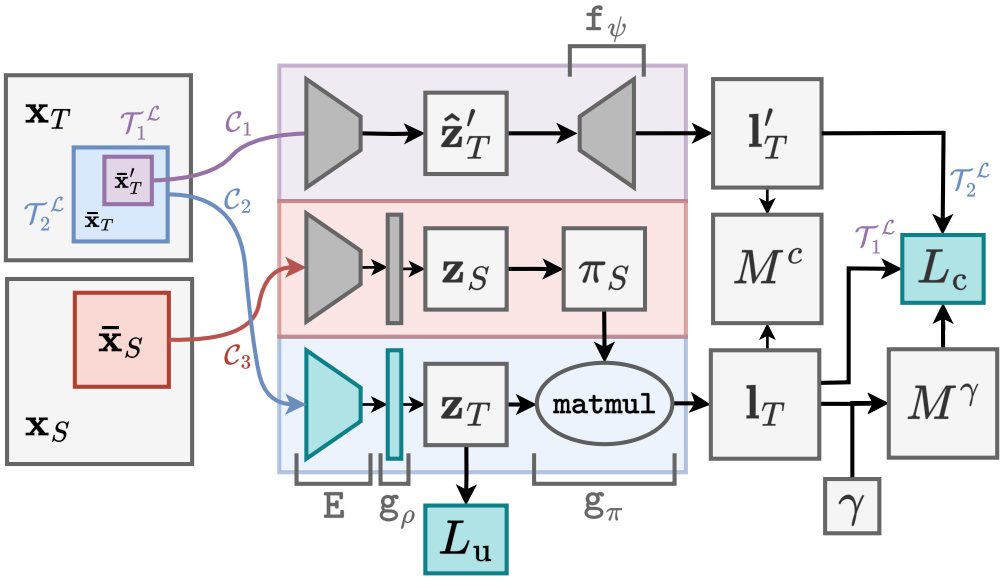


Figure 6.2: Illustration of the architectures of the branches when training on target domain images, and the context in which losses L^c and L^u are calculated. Networks are coloured in aquamarine when the gradients w.r.t. the depicted losses for that forward pass are non-zero, and in grey when no gradients are calculated.

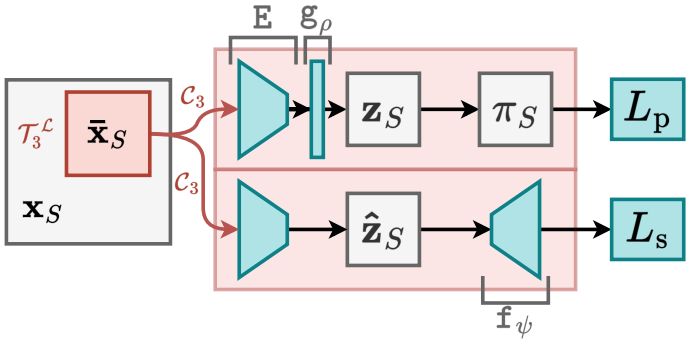


Figure 6.3: Illustration of the architectures of the branches when training on source domain images, and the context in which losses L^p and L^s are calculated. Gradient with respect to the depicted losses are calculated for all networks, hence their colouring in aquamarine.

$$\mathtt{g}(\cdot) = \mathtt{g}_\pi \circ \mathtt{g}_\rho \circ \mathtt{E}(\cdot) \tag{6.8}$$

Where $\mathtt{E}$ is an encoder, $\mathtt{f}_\psi$ is a segmentation head, $\mathtt{g}_\rho$ is a projection network and $\mathtt{g}_\pi$ represents prototype segmentation. The function $\mathtt{g}$ uses prototype segmentation in a way that has been described in Section 6.2.1. The reason as to why $\mathtt{f}$ does not share the architecture of $\mathtt{g}$ will be described in Section 6.5.4 with reference to the training objective.

6.5 Training Objective

Using the above mechanism, the method should train a model so that it can express uncertainty accurately with $\mathbf{M}^{\gamma}$, such that the model is either accurate and certain, or the model is inaccurate and uncertain.

The first step in designing the training method is that we assume segmentation consistency between two images that have been generated via crop-and-resize data augmentation, is a good approximation of segmentation accuracy. In the context of this work: if we have two pixels that correspond to the same location in the original image, and they have both been segmented as the same class by two differently parameterised functions, then it is highly likely that the class assignment is correct, otherwise it is probably not.

For two segmentations, $\{\mathbf{l}, \mathbf{l}'\} \in \mathbb{R}^{H \times W \times K}$ the consistency between these segmentations can be represented by consistency mask $\mathbf{M}^{\mathrm{c}} \in \{0, 1\}^{H \times W}$:

$$\mathbf{M}^{\mathrm{c}}_i = \begin{cases} 1 & \underset{k \in \mathbb{K}}{\operatorname{argmax}}(\mathbf{l}'_i) = \underset{k \in \mathbb{K}}{\operatorname{argmax}}(\mathbf{l}_i) \\ 0 & \text{otherwise} \end{cases} \tag{6.9}$$

This method then relates the uncertainty maps $\mathbf{M}^{\gamma}$ and consistency maps $\mathbf{M}^{\mathrm{c}}$ with a procedure consisting of two steps:

1. γ is solved for such that the number of certain pixels in $\mathbf{M}^{\gamma}$ is equal to the number of consistent pixels in $\mathbf{M}^{\mathrm{c}}$. In this way, the rate of certainty is calibrated by the rate of consistency, which is taken as an approximation of task difficulty and accuracy.

2. In a model training step, the consistency is maximised for the pixels that are certain according to $\mathbf{M}^{\gamma}$. As a result of this step, both $\mathbf{M}^{\mathrm{c}}$ and $\mathbf{M}^{\gamma}$ become better estimators of accuracy.

If every training iteration serves to improve $\mathbf{M}^{\mathrm{c}}$ and $\mathbf{M}^{\gamma}$ as approximations of ground-truth accuracy, then a positive feedback loop is established whereby high-quality uncertainty estimation is learned. In order to fully achieve this feedback loop, a number of additional objectives are required, which are discussed in Section 6.5.4.

6.5.1 Calculating γ

The threshold parameter γ is calculated so that the number of certain pixels in $\mathbf{M}^{\gamma}$ is equal to the number of consistent pixels in $\mathbf{M}^{c}$, see Algorithm 1 for more detail. The approximation being made in this work is that pixel-wise segmentation consistency is an estimate of pixel-wise segmentation accuracy. However, the calculation of γ relies on a looser version of this approximation, that the mean consistency of a batch is an estimate of the mean accuracy of a batch.

It is important that the value for γ is appropriate given how it is used in the second step. If the training objective maximises the consistency between too many pixels, then without ground-truth to keep the segmentation network grounded, it will start to erroneously assign a consistent class to pixels at the same location in the two crops, when they should be uncertain. This is a problem, because an erroneous increase in mean consistency, leads to an erroneous increase in mean certainty via γ, and thus the proportion of pixels that are consistent, certain and inaccurate increases. Once this begins to happen, it is difficult to undo as this is a path to reducing the loss, even though it is a incorrect solution to the problem from our perspective. This type of collapse is a common phenomenon in SSL. Therefore, it is key that $\mathbf{M}^{\gamma}$ is certain about as few pixels that are consistent and inaccurate as possible, and empirically, we find that this method of solving for γ works effectively.

This is perhaps because, initially, the consistency between the segmentations is very low. This therefore sets up a training dynamic where the certainty is also initially low, and consistency is only maximised for the very most confident pixels. Therefore, as the objective increases the consistency over time, it is less likely that pixels are consistent and inaccurate, as the objective was initially very selective and conservative.

6.5.2 Learning $\mathtt{E}$

The training of the encoder $\mathtt{E}$ should result in segmentations that are either certain and accurate *or* uncertain and inaccurate. The objective seeks to achieve this by drawing on ideas in learned loss attenuation (discussed in Section 3.4.1), and to provide two paths for the network to decrease the loss. Either it can produce a consistent (and therefore high-quality) segmentation or it can express uncertainty.

Algorithm 1 Algorithm to calculate γ. Here, $\text{Max}S_T$ contains the largest classification scores for each pixel, i.e. largest similarity with prototypes. As $\mathbf{M}^{\text{c}}$ is a binary mask of consistency, and p_c is its average, $(1 - p_c)$ is the proportion of inconsistent pixels in the batch. Line 9 then chooses γ so that certain pixels have the same proportion as consistent.

1: **Inputs:**
2: Consistency mask: $\mathbf{M}^{\text{c}} \in \mathbb{R}^{N \times H \times W}$
3: Classification scores: $S_T \in \mathbb{R}^{N \times K \times H \times W}$
4: **function** CALCULATE_GAMMA($\mathbf{M}^{\text{c}}, S_T$)
5: $\quad \text{Max}S_T = \texttt{flatten(max(}S_T\texttt{, dim="K"))}$
6: $\quad \text{Max}S_T = \texttt{sort(}\text{Max}S_T\texttt{, ascending=True)}$
7: $\quad p_c = \texttt{mean(}M_c\texttt{)}$ ▷ % consistent pixels
8: $\quad \text{R} = (1 - p_c) * N * H * W$ ▷ Num. uncertain pixels
9: $\quad \gamma = \text{Max}S_T\texttt{[int(R)]}$
10: $\quad$ **return** γ
11: **end function**

The first step in calculating the loss is to compute the soft consistency between the two segmentations. For two crops, $\bar{\mathbf{x}}$ and $\bar{\mathbf{x}}'$, generated with the data augmentation scheme discussed in Section 6.3.1, with pixels at locations i given by $\bar{x} = \bar{\mathbf{x}}_i$ and $\bar{x}' = \bar{\mathbf{x}}'_i$, the categorical distributions for the ith pixel are given by $p(y|\bar{x})$ and $p(y|\bar{x}')$. Then, the soft consistency can be calculated with the cross-entropy function as $\mathcal{H}[p(y|\bar{x}), p(y|\bar{x}')]$.

The next step is to mask out regions of the loss, which the model estimates are likely to be inconsistent, i.e. regions that are uncertain according to $\mathbf{M}^{\gamma}$. The final loss is therefore given by:

$$L^{\text{c}} = \frac{\sum_i^{NHW} \mathbf{M}_i^{\gamma} \, \mathcal{H}[p(y|\bar{x}'_T), p(y|\bar{x}_T)]}{\sum_j^{NHW} \mathbf{M}_j^{\gamma}} \tag{6.10}$$

Where the soft consistency is minimised where the model is certainty ($\mathbf{M}_i^{\gamma} = 1$), and the loss is attenuated where it is uncertain ($\mathbf{M}_i^{\gamma} = 0$).

Another important aspect of this loss, is that the $p(y|\bar{x}'_T)$ is derived from `f`, which uses a segmentation head, rather than prototype segmentation. As a result of this, the entropy of $p(y|\bar{x}'_T)$ is much lower than $p(y|\bar{x}_T)$ due to its training with supervised cross-entropy L^{s} (discussed in Section 6.5.3), therefore this training objective also reduces the entropy of the latter. As $p(y|\bar{x}_T)$ is computed by calculating the cosine similarity between the source prototypes and the target features, a decrease in its entropy relates to the target features being pulled closer to its nearest neighbour prototype. This increases the separation between certain and uncertain pixels because the entropy is only minimised for certain pixels, where $\mathbf{M}_i^{\gamma} = 1$.

6.5.3 Learning the Task

In addition to training the model to perform uncertainty estimation, we also need to train the model to perform the task at hand, i.e. the accurate semantic segmentation of the source domain. For this, the supervised cross-entropy loss L^{s} is used, which for a pixel $x_{\mathrm{S}} = \mathbf{x}_{\mathrm{S},i}$ is given by:

$$L_i^{\mathrm{s}} = -\sum_{k \in \mathbb{K}} \mathbf{y}^*_{\mathrm{S},i,k} \log(p(y = k | x_{\mathrm{S}})) \tag{6.11}$$

Where $\mathbf{y}^*_{\mathrm{S},i} \in \{0, 1\}^K$ is a one-hot ground-truth label for a pixel location i. This segmentation is produced by the segmentation head, therefore this objective updates both the encoder $\mathtt{E}$ and the segmentation head $\mathtt{f}_\psi$, as illustrated in Figure 6.3.

6.5.4 Preventing Feature Collapse

If we were to optimise the model only using L^{c}, then it is likely that feature collapse will occur, as the simplest solution that minimises L^{c} is one in which the model outputs the same segmentation for any image, and across both branches. This happens in SSL when a model loses sensitivity to the input, and produces a model output that satisfies the consistency objective, while being independent of the input. A simple example of this for Siamese networks is when the model produces the same feature vector for all possible images. In our case, the model could produce the same segmentation for all target images, while segmenting the source domain accurately. This would result in consistent (and thus ultimately confident) segmentations that are entirely inaccurate.

This problem is often discussed in SSL literature, such as in [143], [144], and can be solved with both training objectives and architectural changes. This method leverages both.

Additional Objectives

When feature collapse occurs with prototype segmentation, all features are embedded on top of a single or a few prototypes. Therefore, when it occurs there is a characteristic concentration of features in feature space. L^{u} is used to prevent this by softly constraining the model to embed features uniformly on the unit hypersphere, proposed in [136].

For a batch of projected and downsampled features $\tilde{\mathbf{Z}}_T \in \mathbb{R}^{N \times F \times h_u \times w_u}$, the objective

maximises the pairwise distance between features using an RBF kernel:

$$L^{\mathrm{u}} = \frac{1}{N h_{\mathrm{u}} w_{\mathrm{u}}} \sum_{i \neq j} e^{-t||\tilde{\mathbf{z}}_{T,i} - \tilde{\mathbf{z}}_{T,j}||_2^2} \tag{6.12}$$

Where (h_u, w_u) are the downsampled spatial dimensions of the features used in this loss. After projection, the features are average pooled to a quarter of the resolution of the projected features, $(h_u, w_u) = (\frac{h}{4}, \frac{w}{4})$. This is done purely to reduce the memory usage incurred in calculating pairwise distances between all features in a batch of size N. The RBF kernel has a maximum value of 1 when $\tilde{\mathbf{Z}}_{T,i} = \tilde{\mathbf{Z}}_{T,j}$, and thus any non-zero distance is minimised by the loss L^{u}.

L^{u} is only calculated on target images, and so to prevent additional failure modes, uniformity also needs to be encouraged on the source images. An observed failure mode occurs where the prototypes are concentrated and the majority of the target features collapse onto the prototypes (and are thus certain and minimise L^{c}). Meanwhile, the distance between the certain and the smaller proportion of uncertain features are maximised, minimising L^{u}.

Uniformity for the source domain is implemented by maximising the distance between the source prototypes, in the form found in [145]. To do this, the cosine similarity is computed between each of the prototypes as $\pi_S^\top \pi_S \in \mathbb{R}^{K \times F} \times \mathbb{R}^{F \times K} = \mathbb{R}^{K \times K}$. If $\mathbf{I}$ is the identity matrix, then for $(\pi_S^\top \pi_S - 2\mathbf{I})$, the largest possible value for the diagonal self-similarity terms is -1. The maximum non-self pair-wise similarity can thus be calculated for each prototype, for example:

$$\max \left[\underbrace{\begin{pmatrix} 0.94 & 0.31 & 0.23 \\ 0.31 & 0.91 & 0.56 \\ 0.23 & 0.56 & 0.99 \end{pmatrix}}_{\pi_S^\top \pi_S} - \underbrace{\begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}}_{2\mathbf{I}} \right] = \max \left[\begin{pmatrix} -1.06 & 0.31 & 0.23 \\ 0.31 & -1.09 & 0.56 \\ 0.23 & 0.56 & -1.01 \end{pmatrix} \right] = \begin{pmatrix} 0.31 \\ 0.56 \\ 0.56 \end{pmatrix}$$

These maximum pair-wise similarity are minimised, so that L^{p} is maximising the dis-

tance between nearest-neighbour prototypes. Formally, the loss is given by:

$$L^{\mathrm{p}} = \frac{1}{K} \sum_{i=1}^{K} \max_{j \in K} [\pi_S^\top \pi_S - 2\mathbf{I}]_{ij} \tag{6.13}$$

Asymmetric Branches

A second way to prevent feature collapse is to parameterise the branches `f` and `g` differently, e.g. [106], or define each with a different architecture, e.g. [107]. In this work, we do the latter, as previously described in Section 6.4.

The branches share an encoder `E`, however each map the features to a segmentation map differently. Therefore, for both models to produce an identical output and thereby maximise consistency, the following needs to be true: $\mathtt{f}_\psi = \mathtt{g}_\pi \circ \mathtt{g}_\rho$. For this architecture, collapse is not observed for the following possible reasons: (1) $\mathtt{f}_\psi$ and the prototypes used in $\mathtt{g}_\pi$ are not updated by L^{c}, therefore are not encouraged to collapse, (2) $\mathtt{g}_\rho$ is updated with L^{u} in addition to L^{c}.

6.6 Training Procedure

6.6.1 Model Pretraining

Each of the networks are pretrained by L^{s} and L^{u} before training with L^{c}. This means that before training proper starts, the segmentation network $\mathtt{f} = \mathtt{f}_\psi \circ \mathtt{E}$ is trained to segment the source domain, and have some level of performance the in target domain. So long as the network has not overfit, this is a good initialisation for segmenting the target domain, and helps to speed up convergence.

In addition to this the projection network has only been updated with L^{u}, and so has very successfully maximised the uniformity, and thus ultimately the inconsistency between branches `f` and `g`. This initially makes it more difficult for the networks to maximise consistency, and thus makes it less likely that corresponding pixels at a given location are consistent and inaccurate. This helps to maintain the assumption that consistency is a good approximation of ground truth accuracy, and benefits the positive feedback loop, the importance of which we stressed in Section 6.5.1.

6.6.2 Domain-based Curriculae

Each target domain provides a set of unlabelled training images, and so we end up with a trained model for each target domain. Testing primarily occurs in the labelled test dataset from that same target domain.

However, we also perform experiments where a model is trained on multiple unlabelled target datasets. These experiments are based on the assumption (and ultimately the empirical observation) that it is more difficult to learn to perform uncertainty estimation if the target domain is significantly shifted from the source domain. Therefore, our solution to this is to first train a model on a target domain that is more similar to the source domain, but is still similar to the target domain of interest. By splitting training into two steps from source $\rightarrow$ to intermediate target $\rightarrow$ final target, we reduce the effective size of the distributional shift for each training step.

This is related to the motivations in curriculum learning, where it is shown that increasing the difficulty throughout training results in a more performant model. In this work, the difficulty of any given training example is defined in a very coarse manner, i.e. just by the geographic domain it comes from. Nonetheless, it would be possible to use more fine-grained and possibly image-wise metrics to describe the difficulty of training examples.

As previously discussed, we are interested in training a model that is performant in the robot's ODD, and can detect when it has left it. This ODD is defined by the labelled source domain dataset, which has a set of operating conditions, e.g. sunny/overcast, day/night, rainy/dry, summer/winter. Therefore what is out of this ODD and thus OoD is defined as the negative of any one of these choices. We can therefore imagine that a clear definition of a ODD makes it fairly clear how to design a curriculum. The easiest examples are when most of the operating conditions are the same, and the most difficult are when they are all different.

For this reason, we argue that designing and training with a curriculum is simple, and just requires access to a data collection platform (e.g. the deployment robot) to collect data (which requires no annotation) when these conditions occur.

6.7 Network Architecture

The segmentation network used in this method has the DeepLabV3+ [140] architecture, which comprises of a ResNet [35] encoder and a ASPP decoder. We used as ResNet-18 encoder as it is the smallest commonly used ResNet, and we are ultimately interested in deploying such a segmentation network on a mobile robot, therefore low latency and memory usage are very important. This also has the additional benefit of reducing the memory usage during our experimentation, allowing for more experiments to be run in parallel, and thus speeding up development.

In the implementation used[1], there is also a segmentation head that maps from the output of the ASPP module to the segmentation maps. In this method the features used $\hat{z}$ come from the output of the ASPP, i.e. those before the final layer, such that the encoder actually comprises both the ResNet and the ASPP layer:

$$\texttt{E} = \texttt{ASPP} \circ \texttt{ResNet} : \mathbb{R}^{3\times H\times W} \to \mathbb{R}^{F\times h\times w}$$

Where the spatial dimensions of the input are $(H, W) = (256, 256)$, and the downsampled dimensions are $(h, w) = (\frac{H}{4}, \frac{W}{4}) = (64, 64)$, and the feature dimension $F = 256$.The segmentation head is represented by $\texttt{f}_\psi : \mathbb{R}^{F\times h\times w} \to \mathbb{R}^{K\times h\times w}$. The features $\hat{z}$ are projected by projection network g_ρ yielding z of the same dimensionality as $\hat{z}$, and so the projection network is given by $g_\rho : \mathbb{R}^{F\times h\times w} \to \mathbb{R}^{F\times h\times w}$. It is a two-layer MLP which is applied to each pixel feature independently, such that before projected, a batch of N features is reshaped as follows: $\mathbb{R}^{N\times F\times h\times w} \to \mathbb{R}^{Nhw\times F}$ and the MLP operates on each of the individual Nhw features.

6.8 Baselines

It is important to consider the performance of related methods, to see how the results compare and reason about where the improved performance comes from.

There are two high-level methods for how to detect segmentation error for distribution-

[1]DeepLabV3+ implementation found here: `https://github.com/qubvel/segmentation_models.pytorch`

ally shifted images: epistemic uncertainty estimation methods (shortened to epistemic methods) and representation learning based methods (i.e. representation methods). The former investigates the distribution of model parameters in order to quantify when the learned parameters are inadequate for the task at hand as discussed in Section 3.3.2. The latter hypothesize that a model's learned representation ought to have learned to represent the differences between the source images and any other image, therefore this distance in feature space likely correlates with the likelihood of error and thus is a useful measure of uncertainty, discussed in Section 3.5.

Epistemic Baselines

The epistemic methods considered in this work are Monte Carlo Dropout (referred to as `MCD` in the results) and Deep Ensembles (referred to as `Ens`). We use two measures of uncertainty: the predictive entropy (PE) and the mutual information (MI). The architecture used for the Monte Carlo Dropout baseline is based on Bayesian DeepLab [146], which is then adapted for the ResNet-18 used in this work. We test over a range of possible dropout probabilities and number of samples, and conclude that 0.2 and 8 work effectively. Additionally, for the Deep Ensemble baseline, we consider ensembles of size 5 and 10.

As has been discussed, these baselines are not feasible to run in real-time on a mobile robot, however we use the method in [69] to distil the distribution of segmentations from a MCD network into a deterministic network, that can express uncertainty in a single forward pass (known as `MCD-DSL` in the results).

Representation Baselines

The representation baselines considered are OoD detection methods and DUMs, discussed in Section 3.5.1 and Section 3.5.2 respectively.

The OoD detection methods follow the following formula: train a network on the source domain data, then leverage the learned representation by using an inference method that calculates an OoD score that describes the difference between a given image and the source dataset. Therefore, each of these methods uses the same segmentation network trained in a supervised manner on the source domain. [91] simply calculates $\mathbf{p}_{\max}$ (referred to as

Softmax). ODIN in [92] calculates $\mathbf{p}_{\max}$ but with a tuned softmax temperature and adversarially attacked images (referred to as $\texttt{Softmax}_{\texttt{A}}$). The method in [93] computes the Mahalanobis distance between a given image features and a multi-dimensional Gaussian fitted to the mean source domain features (referred to as FeatDist), which can also leverage adversarially attacked images (referred to as $\texttt{FeatDist}_{\texttt{A}}$). Finally, ViM proposed in [94] computes a score based on both the extracted features and the logits.

For a direct comparison, the methods that use features use the same feature space is used as in our proposed method, described in Section 6.7. For the methods using adversarially attacked images, a range of ϵ are tried, and results are presented for the best performing value.

As for the DUMs, we use the simple but effective method proposed in [103], which calculates a post-hoc GMM with a mixture component for each class. The implementation for the spectral normalised network is based on [147][2].

Our Proposed Models

The models trained with the method proposed in this chapter are referred to as γ-SSL and $\gamma\text{-}\texttt{SSL}_{\texttt{iL}}$. The latter is trained according to the curriculum suggested in Section 6.6.2, and the intermediate domain used is SAX London (hence the subscript iL referring to **i**nit **L**ondon).

6.9 Evaluating uncertainty estimation on narrow target domains

With results found in Section 6.9.1, the first set of experiments train our proposed models with the labelled training dataset from Cityscapes and unlabelled training dataset from one of the SAX domains. These models are then evaluated with the test dataset from the same SAX domain as the unlabelled training dataset. The aim of this is to evaluate how the proposed method is able to detect error due to a specific type of distributional shift, by using training data that contains instances of this specific type of shift.

The experiments are repeated with KITTI and BDD as the distributionally shifted tar-

[2]Implementation found at https://github.com/jhjacobsen/invertible-resnet

get domains, with results found in Section 6.9.3. Additionally, experiments with the SAX domains as the target domains are also performed with BDD providing the labelled source training dataset, with results in Section 6.9.4.

6.9.1 Source: Cityscapes, Target: SAX Test Datasets

Target: SAX London

Looking at Figure 6.4, the γ-SSL model performs best in each plot for SAX London. Its precision is higher for all values of recall, which relate to increases of $19\,\%$ and $8\,\%$ over the next best baseline for AUROC and AUPR respectively (more detail found in Table 6.3 and Table 6.2). γ-SSL also returns the highest $\mathrm{MaxA_{MD}}$ and $\mathrm{MaxF_{1/2}}$ scores, with corresponding values of $\mathrm{p(a, c)}$ that are higher than all but $\mathrm{MCD_{0.2}}$.

$\mathrm{MCD_{0.2}}$ generally is the best baseline, returning the highest AUROC, AUPR, $\mathrm{MaxA_{MD}}$ and $\mathrm{MaxF_{1/2}}$ scores of the baselines. For this baseline, MI instead of PE was on the whole more performant. It is worth noting that the $\mathrm{MCD_{0.2}}$ models also have higher segmentation accuracies, presumably owing to the dropout layers providing effective regularisation during training, leading to better generalisation than a standard segmentation network.

The quality of uncertainty estimation for the epistemic and representation methods was fairly similar according to AUROC and AUPR. When looking at $\mathrm{MaxA_{MD}}$ and $\mathrm{MaxF_{1/2}}$ scores, these were also quite similar between the two method types, however the values of $\mathrm{p(a, c)}$ at which the scores occur for representation methods is lower on average.

The γ-$\mathrm{SSL_{iL}}$ models don't exist for this dataset configuration as for this SAX London as the target domain, they are the same models as γ-SSL.

Target: SAX New Forest

In terms of AUROC and AUPR, the γ-SSL and γ-$\mathrm{SSL_{iL}}$ models produce similar results, with the γ-$\mathrm{SSL_{iL}}$ models better on the margins (AUPR of 0.942 vs. 0.921). Similarly for the $\mathrm{MaxA_{MD}}$ and $\mathrm{MaxF_{1/2}}$ scores the γ-$\mathrm{SSL_{iL}}$ models are slightly better with increases of $2.4\,\%$ and $3.5\,\%$ respectively. Underlying these metrics, there is however a large difference as the values of $\mathrm{p(a, c)}$ at which the γ-$\mathrm{SSL_{iL}}$ scores are reported are far larger, with increases of $25.9\,\%$ and $30.7\,\%$ over γ-SSL. For this reason, the γ-$\mathrm{SSL_{iL}}$ are much more useful, which

is not something that AUROC or AUPR capture, thereby demonstrating the importance of presenting the results as we do. Part of the reason for this is that the curriculum learning has improved the segmentation accuracy for this domain.

The $\texttt{MCD}_{0.2}$ models have lower AUROC and AUPR than γ-SSL models, and slightly lower MaxA_{MD} and $\text{MaxF}_{1/2}$ scores. However, the latter is tarnished by the fact that the scores for γ-SSL are for values of $\text{p}(\text{a}, \text{c})$ that are $22\,\%$ lower. In order to pick between $\texttt{MCD}_{0.2}$ and γ-SSL, it is therefore important to weigh up 'usefulness' versus 'safety'.

The same cannot be said for the γ-$\texttt{SSL}_{\texttt{iL}}$ model, which has significantly better MaxA_{MD} and $\text{MaxF}_{1/2}$ scores with increases of $9.0\,\%$ and $7.3\,\%$ respectively for minor changes in $\text{p}(\text{a}, \text{c})$ of $-2.7\,\%$ and $1.7\,\%$ compared with $\texttt{MCD-MI}_{0.2}$.

No different to SAX London, the $\texttt{MCD}_{0.2}$ models perform the best out of the baselines. Again the epistemic methods were better than the representation methods, with higher AUROC and AUPR, and similar values of MaxA_{MD} and $\text{MaxF}_{1/2}$ but at higher values of $\text{p}(\text{a}, \text{c})$.

Target: SAX Scotland

The key finding in this domain is that the increase in performance from γ-SSL to γ-$\texttt{SSL}_{\texttt{iL}}$ is at its greatest. The γ-$\texttt{SSL}_{\texttt{iL}}$ far exceeds the quality of uncertainty estimation of the other models, as easily seen in Figure 6.4. For the MaxA_{MD}@$\text{p}(\text{a}, \text{c})$ and $\text{MaxF}_{0.5}$@$\text{p}(\text{a}, \text{c})$ metrics, the increases over the next best baseline is as follows: $9.1\,\%$ @ $65.8\,\%$, $20.2\,\%$ @ $72.1\,\%$. Similar to the New Forest, but much more pronounced, is the large increase in the quality of both segmentation accuracy and the uncertainty estimation as a result of the curriculum learning.

As for γ-SSL, its results are similar to the best baseline $\texttt{MCD}_{0.2}$. It is clear that the extremely large distributional shift causes a significant challenge to the semi-supervised task – perhaps demonstrated by the difference between γ-SSL and γ-$\texttt{SSL}_{\texttt{iL}}$.

As a result of the distributional shift, the epistemic methods are better than the representation methods by the largest margin out of the domains. Quantitatively, this is represented by the increase in AUPR of $34.8\,\%$. It is clear that a representation solely learned on Cityscapes is not appropriate to perform OoD detection in a domain as distributionally shifted as SAX Scotland.

6.9.2 Effect of distributional shift

For the SAX domains, the order of increasing distributional shift to Cityscapes is: London, New Forest, Scotland. Qualitatively, this was thought to be true, which is backed up quantitatively by the decreasing segmentation quality of a segmentation network trained on Cityscapes only, with segmentation accuracies being: 0.571, 0.538, 0.394 for London, New Forest and Scotland respectively. We investigate the changes in the quality of uncertainty estimation for each model as a function of distributional shift in Table 6.1. The AUROC and AUPR are presented as they are independent of the segmentation accuracy of each method, which decreases as the distributional shift increases. Thus, it is expected that these metrics should not change as distributional shift increases.

We are particularly interested in how the quality of uncertainty estimation degrades for the type of method: epistemic and representation, and comparing that to our proposed methods.

Table 6.1: AUROC and AUPR Percentage Change at Increasing Distributional Shift, $\%\Delta\mathrm{ROC}$ and $\%\Delta\mathrm{PR}$ Respectively

		LDN → NF		NF → SCOT	
	Method	$\%\Delta$ROC	$\%\Delta$PR	$\%\Delta$ROC	$\%\Delta$PR
Epistemic	Ens-PE$_5$	2.3	2.7	-2.5	-9.6
	Ens-MI$_5$	11.1	11.2	-13.0	-22.7
	Ens-PE$_{10}$	4.2	2.0	-3.2	-12.4
	Ens-MI$_{10}$	13.7	12.8	-16.3	-26.3
	MCD-PE$_{0.2}$	1.8	0.9	-0.6	-14.2
	MCD-MI$_{0.2}$	2.6	1.3	-6.4	-16.4
	MCD-DSL	6.5	4.2	-8.9	-26.4
	– Mean –	6.0	5.0	-7.3	-18.3
Representation	Softmax	-5.1	-11.2	-1.8	-16.8
	Softmax$_{\mathrm{A}}$	-5.1	-11.2	-1.9	-17.0
	FeatDist	-2.3	-6.1	0.3	-22.1
	FeatDist$_{\mathrm{A}}$	-0.4	-3.1	-9.0	-34.8
	ViM	10.3	0.5	-8.5	-29.2
	DUM	16.9	-1.4	-9.7	-43.6
	– Mean –	2.4	-5.4	-5.1	-27.3
Ours	γ-SSL	-1.7	-3.0	-11.8	-21.2
	γ-SSL$_{\mathrm{iL}}$	-1.7	-0.7	-2.4	-5.8

A negative value represents a decrease in misclassification detection performance as distributional shift increases.

Epistemic Methods

Considering the shift from London to New Forest, the AUROC and AUPR increase by $6.0\,\%$ and $5.0\,\%$ respectively. As epistemic methods are designed specifically to detect epistemic uncertainty, this could be a result of an increase in pixels that are wrong as a result of epistemic uncertainty, e.g. by not having trained on the right training dataset. This, however, does not happen when the distributional shift increases further, in the shift from New Forest to Scotland. For this shift, the AUROC and AUPR decrease for every method, and for some very significantly, leading average decreases of $7.3\,\%$ and $18.3\,\%$ respectively.

The conclusion from this is that, as the proportion of pixels that are incorrect as a result of distributional shift increases, there might be some immediate benefits to uncertainty estimation quality. However, ultimately, distributional shift can increase to a magnitude that can greatly reduce the quality of the detection of these errors, which is a concern.

Representation Methods

On average, methods that rely on leveraging the representation learned on a labelled dataset tend to produce uncertainty estimates that are less robust to distributional shift. This is demonstrated quantitatively by the decrease in AUPR of $5.4\,\%$ and $27.3\,\%$ for the two shifts considered. Considering AUROC, the change was less clear, however the majority of representation methods decreased in AUROC across both shifts. $\mathrm{MaxA}_{\mathrm{MD}}$, $\mathrm{MaxF}_{1/2}$, and the $\mathrm{p}(\mathrm{a}, \mathrm{c})$ at which these scores occur also greatly reduce in the shift from New Forest to Scotland, meaning that both the segmentation quality has decreased, but also the ability for the model to delineate between the accurate and inaccurate pixels has degraded.

These quantitative results demonstrate that relying on the task-specific representation of these pretrained segmentation networks leads to increasingly poor misclassification detection as the magnitude of the distributional shift increases. This justifies the focus on methods, such as ours in this chapter, that use additional data to learn a more task-agnostic representation to prevent this severe degradation.

γ-SSL Methods

Similar to the representation methods, the γ-SSL and γ-SSL_{iL} methods decrease in uncertainty estimation quality for both AUROC and AUPR and both shifts. Considering only γ-SSL to begin with, the metrics are inconclusive about the robustness of uncertainty estimation to shifts, with the representation methods being preferable for AUROC and γ-SSL being preferable for AUPR.

However this cannot be said for γ-SSL_{iL}, where the robustness to distributional shift between New Forest and Scotland for this method is significantly better than the epistemic and representation methods, and indeed the best of any individual method. This shows that the curriculum approach massively the robustness of uncertainty estimation in a way that is independent of the increases in segmentation accuracy that it also provides.

Clearly, the quantitative results show that this additional training on distributionally shifted data greatly improves the representation by including the type of task-agnostic information required for misclassification detection.

6.9.3 Source: Cityscapes, Target: KITTI & BDD

In order to show that the proposed method is not overfit to the SAX datasets, we also use KITTI and BDD as target domains. We can determine how distributionally shifted these domains are from Cityscapes by considering the segmentation accuracy for a γ-SSL model on each of the test sets. The segmentation accuracies for KITTI, SAX London, BDD, SAX New Forest are: 0.817, 0.703, 0.684, 0.595. This suggests that the ordering of increasing segmentation accuracy is: KITTI, SAX London, BDD, SAX New Forest, SAX Scotland.

KITTI

γ-SSL exceeds the uncertainty estimation quality of each of the baselines for AUROC, MaxA_{MD}, $\text{MaxF}_{1/2}$, while the MCD-DSL methods has a better AUPR. The epistemic methods are better on average for KITTI, and each epistemic method outperforms each representation method.

BDD

For each metric considered, γ-SSL outperforms the baselines on BDD. Similar to KITTI, each epistemic method has higher AUROC and AUPR than each representation method.

According to our analysis, SAX New Forest is more distributionally shifted than BDD, however the metrics for BDD are lower than for SAX New Forest. This is possible because BDD is significantly more diverse than SAX New Forest, which therefore makes learning uncertainty estimation more of a challenge.

Table 6.2: Misclassification Detection AUROC with Source: Cityscapes

		AUROC				
	Method	LDN	NF	SCOT	KITTI	BDD
Epistemic	Ens-PE$_5$	0.630	0.645	0.629	0.845	0.758
	Ens-MI$_5$	0.551	0.612	0.532	0.705	0.696
	Ens-PE$_{10}$	0.603	0.629	0.608	0.828	0.755
	Ens-MI$_{10}$	0.554	0.629	0.527	0.690	0.713
	MCD-PE$_{0.2}$	0.727	0.739	0.735	0.864	0.801
	MCD-MI$_{0.2}$	0.755	0.774	0.725	0.815	0.801
	MCD-DSL	0.697	0.742	0.676	0.855	0.761
	– Mean –	0.645	0.681	0.633	0.800	0.755
Representation	Softmax	0.723	0.686	0.674	0.785	0.706
	Softmax$_A$	0.722	0.685	0.672	0.784	0.705
	FeatDist	0.62	0.605	0.607	0.575	0.641
	FeatDist$_A$	0.645	0.642	0.585	0.585	0.648
	ViM	0.635	0.700	0.640	0.714	0.745
	DUM	0.483	0.565	0.510	0.501	0.502
	– Mean –	0.638	0.647	0.615	0.657	0.658
Ours	γ-SSL	**0.895**	0.880	0.776	**0.888**	**0.835**
	γ-SSL$_{iL}$	-	**0.880**	**0.859**	-	-

Table 6.3: Misclassification Detection AUPR with Source: Cityscapes

		AUPR				
	Method	LDN	NF	SCOT	KITTI	BDD
Epistemic	Ens-PE$_5$	0.714	0.733	0.663	0.956	0.854
	Ens-MI$_5$	0.663	0.737	0.570	0.910	0.830
	Ens-PE$_{10}$	0.710	0.724	0.634	0.952	0.857
	Ens-MI$_{10}$	0.667	0.752	0.554	0.905	0.840
	MCD-PE$_{0.2}$	0.851	0.859	0.737	0.926	0.816
	MCD-MI$_{0.2}$	0.878	0.889	0.744	0.956	0.904
	MCD-DSL	0.785	0.818	0.602	**0.957**	0.851
	– Mean –	0.753	0.787	0.643	0.937	0.850
Representation	Softmax	0.711	0.631	0.525	0.840	0.686
	Softmax$_A$	0.711	0.631	0.524	0.840	0.686
	FeatDist	0.693	0.650	0.507	0.660	0.718
	FeatDist$_A$	0.721	0.699	0.456	0.637	0.727
	ViM	0.705	0.708	0.502	0.800	0.797
	DUM	0.630	0.622	0.350	0.367	0.319
	– Mean –	0.695	0.657	0.477	0.691	0.656
Ours	γ-SSL	**0.949**	0.921	0.726	0.951	**0.911**
	γ-SSL$_{iL}$	-	**0.942**	**0.887**	-	-

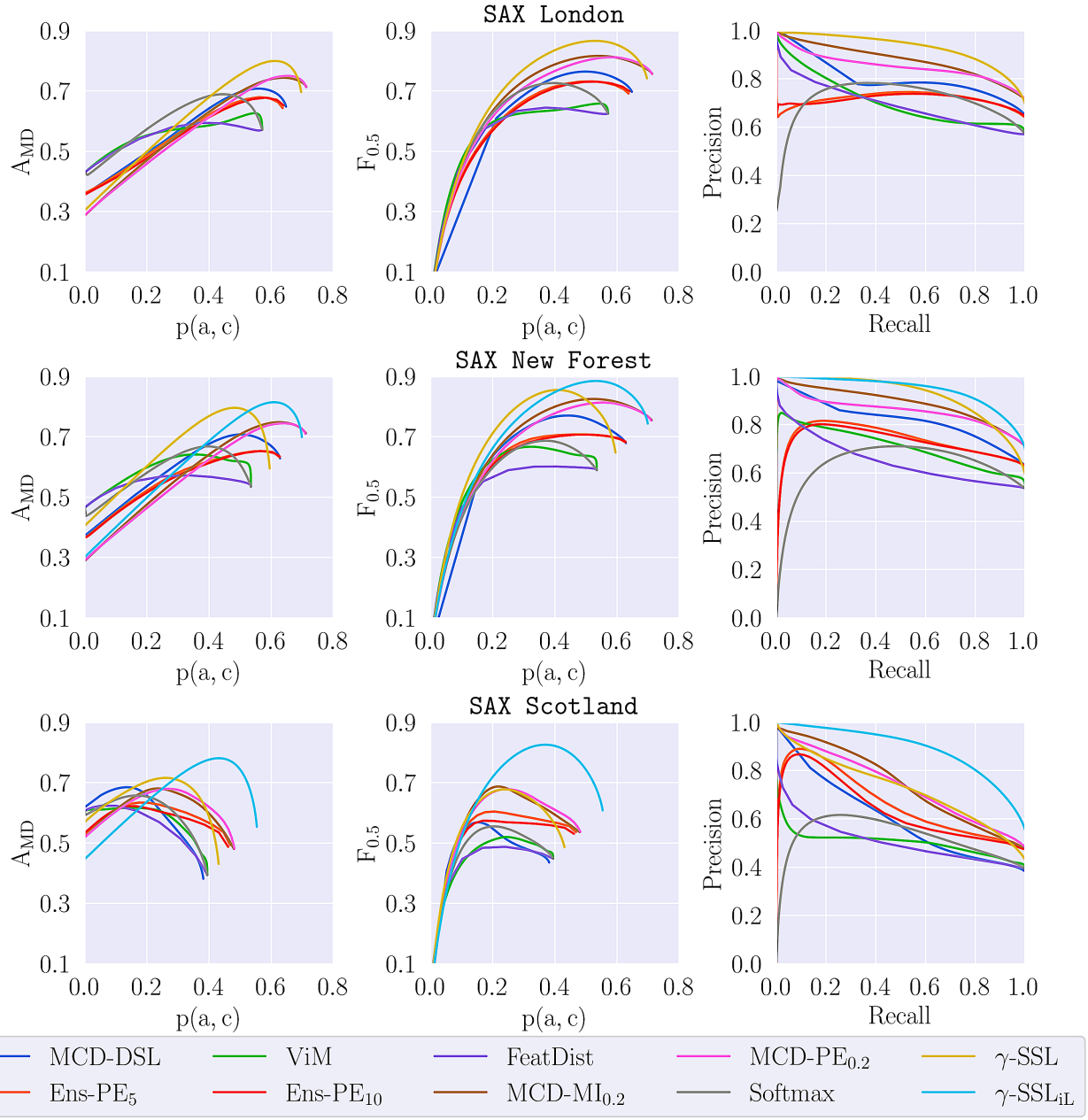


Figure 6.4: For each SAX domain, a row of plots describes the misclassification detection performance of a series of baselines and the proposed methods, γ-SSL and γ-SSL$_{\text{iL}}$. A_{MD} and $\text{F}_{1/2}$ (shown in the plots as $\text{F}_{0.5}$) aggregate performance into a single metric, where a larger value of each represents a more 'introspective' model. They are plotted versus $\text{p}(\text{a}, \text{c})$, the proportion of pixels that are accurate and certain, as this represents the amount of accurate and useful semantic information the model can extract from images; also a metric maximised by the ideal model. Note that the maximum value of $\text{p}(\text{a}, \text{c})$ is equal to the segmentation accuracy, $\max[\text{p}(\text{a}, \text{c})] = \text{p}(\text{accurate})$.

6.9.4 Source: BDD, Target: SAX Test Datasets

We also train our methods and the baselines on the labelled dataset from BDD. This is to help up investigate the generality of each of the approaches, and to see how a different source domains affects uncertainty estimation.

Table 6.4: Maximum $\mathrm{A_{MD}}$ and $\mathrm{p(a, c)}$ with Source: Cityscapes

		$\mathrm{MaxA_{MD}}$ @ $\mathrm{p(a, c)}$				
	Method	LDN	NF	SCOT	KITTI	BDD
Epistemic	Ens-$\mathrm{PE_5}$	0.679 @ 0.560	0.652 @ 0.581	0.634 @ 0.175	0.818 @ 0.751	0.734 @ 0.607
	Ens-$\mathrm{MI_5}$	0.645 @ 0.620	0.643 @ 0.613	0.580 @ 0.076	0.798 @ 0.797	0.692 @ 0.668
	Ens-$\mathrm{PE_{10}}$	0.678 @ 0.582	0.653 @ 0.564	0.622 @ 0.142	0.812 @ 0.768	0.728 @ 0.602
	Ens-$\mathrm{MI_{10}}$	0.654 @ 0.622	0.650 @ 0.611	0.578 @ 0.079	0.798 @ 0.798	0.698 @ 0.624
	MCD-$\mathrm{PE_{0.2}}$	0.750 @ 0.646	0.745 @ 0.645	0.679 @ 0.270	0.849 @ 0.792	0.765 @ 0.608
	MCD-$\mathrm{MI_{0.2}}$	0.744 @ 0.640	0.748 @ 0.625	0.681 @ 0.234	0.833 @ 0.823	0.754 @ 0.607
	MCD-DSL	0.708 @ 0.559	0.708 @ 0.503	0.685 @ 0.136	0.830 @ 0.745	0.738 @ 0.571
	– Mean –	0.694 @ 0.604	0.686 @ 0.592	0.637 @ 0.159	0.795 @ 0.782	0.730 @ 0.612
Representation	Softmax	0.689 @ 0.444	0.668 @ 0.396	0.657 @ 0.164	0.718 @ 0.434	0.692 @ 0.479
	$\mathrm{Softmax_A}$	0.689 @ 0.446	0.668 @ 0.401	0.656 @ 0.167	0.717 @ 0.435	0.692 @ 0.480
	FeatDist	0.594 @ 0.371	0.572 @ 0.334	0.624 @ 0.055	0.588 @ 0.574	0.620 @ 0.489
	$\mathrm{FeatDist_A}$	0.617 @ 0.449	0.603 @ 0.378	0.636 @ 0.032	0.564 @ 0.384	0.626 @ 0.502
	ViM	0.626 @ 0.546	0.641 @ 0.353	0.617 @ 0.153	0.655 @ 0.386	0.687 @ 0.480
	DUM	0.625 @ 0.625	0.618 @ 0.556	0.649 @ 0.000	0.733 @ 0.732	0.637 @ 0.635
	– Mean –	0.640 @ 0.480	0.628 @ 0.403	0.640 @ 0.095	0.663 @ 0.491	0.659 @ 0.511
Ours	γ-SSL	**0.83** @ 0.625	0.796 @ 0.483	0.716 @ 0.260	**0.856** @ 0.767	**0.770** @ 0.568
	γ-$\mathrm{SSL_{iL}}$	-	**0.815** @ 0.608	**0.781** @ 0.431	-	-

Table 6.5: Maximum $\mathrm{F_{1/2}}$ Score and $\mathrm{p(a, c)}$ with Source: Cityscapes

		$\mathrm{MaxF_{1/2}}$ @ $\mathrm{p(a, c)}$				
	Method	LDN	NF	SCOT	KITTI	BDD
Epistemic	Ens-$\mathrm{PE_5}$	0.732 @ 0.520	0.708 @ 0.460	0.604 @ 0.206	0.889 @ 0.613	0.801 @ 0.496
	Ens-$\mathrm{MI_5}$	0.694 @ 0.609	0.691 @ 0.605	0.528 @ 0.461	0.832 @ 0.795	0.758 @ 0.472
	Ens-$\mathrm{PE_{10}}$	0.730 @ 0.539	0.707 @ 0.517	0.574 @ 0.168	0.880 @ 0.608	0.795 @ 0.492
	Ens-$\mathrm{MI_{10}}$	0.702 @ 0.603	0.696 @ 0.597	0.522 @ 0.460	0.833 @ 0.761	0.767 @ 0.488
	MCD-$\mathrm{PE_{0.2}}$	0.812 @ 0.584	0.813 @ 0.559	0.678 @ 0.248	0.914 @ 0.686	0.836 @ 0.530
	MCD-$\mathrm{MI_{0.2}}$	0.816 @ 0.541	0.825 @ 0.523	0.687 @ 0.215	0.891 @ 0.669	0.836 @ 0.500
	MCD-DSL	0.764 @ 0.499	0.770 @ 0.441	0.568 @ 0.155	0.900 @ 0.644	0.802 @ 0.491
	– Mean –	0.750 @ 0.556	0.744 @ 0.529	0.594 @ 0.273	0.877 @ 0.682	0.799 @ 0.496
Representation	Softmax	0.726 @ 0.395	0.687 @ 0.370	0.555 @ 0.205	0.777 @ 0.366	0.735 @ 0.429
	$\mathrm{Softmax_A}$	0.726 @ 0.397	0.687 @ 0.373	0.554 @ 0.207	0.777 @ 0.366	0.735 @ 0.430
	FeatDist	0.645 @ 0.371	0.601 @ 0.408	0.487 @ 0.240	0.640 @ 0.571	0.672 @ 0.434
	$\mathrm{FeatDist_A}$	0.672 @ 0.360	0.643 @ 0.296	0.449 @ 0.240	0.612 @ 0.525	0.680 @ 0.428
	ViM	0.658 @ 0.543	0.667 @ 0.326	0.520 @ 0.243	0.721 @ 0.324	0.735 @ 0.391
	DUM	0.676 @ **0.625**	0.664 @ 0.528	0.419 @ 0.315	0.774 @ 0.731	0.686 @ 0.635
	– Mean –	0.684 @ 0.449	0.658 @ 0.384	0.497 @ 0.242	0.717 @ 0.481	0.707 @ 0.458
Ours	γ-SSL	**0.893** @ 0.548	0.855 @ 0.407	0.678 @ 0.239	**0.920** @ 0.676	**0.843** @ 0.475
	γ-$\mathrm{SSL_{iL}}$	-	**0.885** @ 0.532	**0.826** @ 0.370	-	-

Training the baselines on BDD increases the quality of uncertainty estimation for epistemic and representation methods according to each of the metrics. For the $\mathrm{MaxF_{1/2}}$ scores of the baselines, the increase from Cityscapes to BDD for SAX London, SAX New Forest, SAX Scotland are: $9.1\,\%$, $14.6\,\%$, $26.2\,\%$. This corresponds with the fact that there is a smaller distributional shift between BDD and SAX Scotland.

Target: SAX London

γ-SSL has the highest quality uncertainty estimation out of all of the methods for this target domain, characterised by a $6.5\,\%$ increase in $\mathrm{MaxF}_{1/2}$ over the next best score (held jointly by Softmax, $\mathrm{Softmax}_{\mathrm{A}}$ and $\mathrm{MCD\text{-}PE}_{0.2}$).

In contrast to what is true for the baselines, the uncertainty estimation performance for the γ-SSL models are higher using Cityscapes as the source domain than BDD. The segmentation accuracy for γ-SSL is also higher for Cityscapes than for BDD, with 0.703 and 0.688 respectively.

Target: SAX New Forest

Contrary to SAX London, the segmentation accuracy for γ-SSL using BDD as the source domain is higher than that of using Cityscapes (0.666 and 0.595 respectively). This implies smaller distributional shift between BDD and SAX New Forest than Cityscapes and SAX New Forest.

γ-SSL and $\gamma\text{-}\mathrm{SSL}_{\mathrm{iL}}$ models outperform each of the baselines on all metrics apart from AUPR. Unlike what is predicted according to the relative distributional shifts, the uncertainty estimation performance of our models trained on BDD as source are not consistently better than those trained on Cityscapes. This is perhaps again because of the significant diversity in BDD compared with Cityscapes. Therefore, the appearance and semantics of the latter are more concisely described, making distributional uncertainty estimation easier.

Target: SAX Scotland

Using BDD as the source domain for γ-SSL makes a significant positive change to the uncertainty estimation performance over using Cityscapes. This is also true for $\gamma\text{-}\mathrm{SSL}_{\mathrm{iL}}$ but to a lesser extent. Looking at the segmentation accuracies with BDD and Cityscapes as source, gives us 0.495 and 0.431 respectively. Therefore, as described above, BDD is closer to SAX Scotland than Cityscapes, and this helps explain the benefit of using it in this case.

In this experiment, the epistemic methods are much better than the representation methods and perform comparably with $\gamma\text{-}\mathrm{SSL}_{\mathrm{iL}}$.

Conclusions

These experiments help us describe how uncertainty estimation performance changes depending on which training dataset and test dataset considered. They show that our proposed methods perform very favourably to the baselines, and are typically the best performing method even when considering those methods that are too computationally heavy to use for mobile robotics applications (namely the epistemic methods apart for MCD-DSL).

Table 6.6: Misclassification Detection AUROC and AUPR with Source: BDD

		AUROC		
	Method	LDN	NF	SCOT
Epistemic	Ens-PE$_5$	0.781	0.854	0.862
	Ens-MI$_5$	0.744	0.809	0.818
	Ens-PE$_{10}$	0.785	0.854	**0.866**
	Ens-MI$_{10}$	0.761	0.827	0.861
	MCD-PE$_{0.2}$	0.834	0.841	0.822
	MCD-MI$_{0.2}$	0.808	0.816	0.739
	MCD-DSL	0.821	0.803	0.805
	– Mean –	0.791	0.829	0.825
Representation	Softmax	0.825	0.819	0.810
	Softmax$_A$	0.825	0.819	0.810
	FeatDist	0.605	0.599	0.563
	FeatDist$_A$	0.625	0.631	0.598
	ViM	0.663	0.664	0.560
	DUM	0.480	0.431	0.398
	– Mean –	0.671	0.661	0.623
Ours	γ-SSL	**0.889**	0.876	0.833
	γ-SSL$_{iL}$	-	**0.882**	0.855

		AUPR		
	Method	LDN	NF	SCOT
Epistemic	Ens-PE$_5$	0.896	0.941	0.885
	Ens-MI$_5$	0.876	0.915	0.837
	Ens-PE$_{10}$	0.899	**0.942**	0.880
	Ens-MI$_{10}$	0.884	0.925	0.870
	MCD-PE$_{0.2}$	0.903	0.881	0.798
	MCD-MI$_{0.2}$	0.880	0.863	0.728
	MCD-DSL	0.902	0.914	0.849
	– Mean –	0.891	0.912	0.835
Representation	Softmax	0.901	0.904	0.816
	Softmax$_A$	0.901	0.905	0.816
	FeatDist	0.724	0.731	0.547
	FeatDist$_A$	0.758	0.770	0.604
	ViM	0.799	0.797	0.547
	DUM	0.549	0.530	0.321
	– Mean –	0.772	0.773	0.608
Ours	γ-SSL	**0.936**	0.935	0.858
	γ-SSL$_{iL}$	-	0.928	**0.900**

Table 6.7: Maximum A_{MD} and $p(a, c)$ with Source: BDD

		MaxA_{MD} @ $p(a, c)$		
	Method	LDN	NF	SCOT
Epistemic	Ens-PE$_5$	0.739 @ 0.630	0.783 @ 0.621	0.790 @ 0.347
	Ens-MI$_5$	0.716 @ 0.623	0.766 @ 0.637	0.753 @ 0.343
	Ens-PE$_{10}$	0.741 @ 0.619	0.783 @ 0.627	0.792 @ 0.331
	Ens-MI$_{10}$	0.727 @ 0.619	0.774 @ 0.629	**0.797** @ 0.351
	MCD-PE$_{0.2}$	0.753 @ 0.488	0.760 @ 0.390	0.766 @ 0.217
	MCD-MI$_{0.2}$	0.729 @ 0.454	0.742 @ 0.362	0.729 @ 0.176
	MCD-DSL	0.756 @ 0.549	0.761 @ 0.638	0.727 @ 0.387
	– Mean –	0.737 @ 0.569	0.767 @ 0.558	0.765 @ 0.307
Representation	Softmax	0.761 @ 0.553	0.760 @ 0.573	0.755 @ 0.275
	Softmax$_A$	0.761 @ 0.554	0.761 @ 0.574	0.754 @ 0.275
	FeatDist	0.673 @ 0.609	0.677 @ 0.674	0.603 @ 0.154
	FeatDist$_A$	0.706 @ 0.646	0.727 @ 0.668	0.590 @ 0.265
	ViM	0.662 @ 0.662	0.683 @ 0.608	0.588 @ 0.114
	DUM	0.562 @ 0.562	0.596 @ 0.596	0.604 @ 0.000
	– Mean –	0.688 @ 0.598	0.701 @ 0.616	0.649 @ 0.181
Ours	γ-SSL	**0.818** @ 0.599	0.805 @ 0.567	0.762 @ 0.331
	γ-SSL$_{iL}$	-	**0.809** @ 0.578	0.769 @ 0.417

Table 6.8: Maximum $F_{1/2}$ Score and $p(a, c)$ with Source: BDD

		$\text{MaxF}_{1/2}$ @ $p(a, c)$		
	Method	LDN	NF	SCOT
Epistemic	Ens-PE_5	0.810 @ 0.488	0.872 @ 0.500	0.829 @ 0.307
	Ens-MI_5	0.792 @ 0.484	0.839 @ 0.528	0.777 @ 0.290
	Ens-PE_{10}	0.814 @ 0.488	0.872 @ 0.500	0.825 @ 0.284
	Ens-MI_{10}	0.802 @ 0.483	0.855 @ 0.519	0.815 @ 0.309
	MCD-$PE_{0.2}$	0.833 @ 0.403	0.814 @ 0.328	0.748 @ 0.199
	MCD-$MI_{0.2}$	0.820 @ 0.380	0.803 @ 0.311	0.685 @ 0.163
	MCD-DSL	0.829 @ 0.460	0.839 @ 0.525	0.769 @ 0.319
	– Mean –	0.814 @ 0.455	0.842 @ 0.459	0.778 @ 0.267
Representation	Softmax	0.833 @ 0.465	0.835 @ 0.483	0.764 @ 0.241
	Softmax_A	0.833 @ 0.468	0.836 @ 0.484	0.764 @ 0.241
	FeatDist	0.730 @ 0.564	0.731 @ 0.566	0.534 @ 0.191
	FeatDist_A	0.755 @ 0.614	0.778 @ 0.621	0.592 @ 0.285
	ViM	0.732 @ 0.433	0.747 @ 0.541	0.511 @ 0.214
	DUM	0.616 @ 0.562	0.648 @ 0.596	0.451 @ 0.396
	– Mean –	0.750 @ 0.518	0.763 @ 0.549	0.603 @ 0.261
Ours	γ-SSL	**0.887** @ 0.515	0.873 @ 0.495	0.797 @ 0.284
	γ-SSL_{iL}	-	**0.885** @ 0.501	**0.831** @ 0.348

6.10 Evaluating uncertainty estimation on a general target domain

As discussed in Section 6.1.1, we are interested in evaluating the extent to which using distributionally shifted driving images from one domain improves quality of uncertainty estimation for a wide range of driving domains, i.e. investigating the extent to which our proposed models have generalised. This is achieved by training using labelled images from Cityscapes with unlabelled images from each of the SAX domains, and then testing on the WildDash dataset. The results for this are found in the next section.

6.10.1 Target: WildDash

In the preceding evaluations, γ-SSL and γ-SSL_{iL} have been tested in domains for which the model had access to unlabelled training images for that domain. In this experiment, each of the baselines, γ-SSL and γ-SSL_{iL} are tested in an entirely unseen and very challenging domain, namely WildDash, [134] (see Section 4.3 for more detail).

This dataset is very challenging, but is a good way of testing the extent to which a given model can perform general distributional uncertainty estimation. As discussed in Section 6.1.1, we are interested in evaluating the extent to which using a *specific* distributionally shifted driving dataset from one domain improves *general* distributional uncertainty estimation, as well as for the domain of the unlabelled target training dataset.

The results for this can be found in Figure 6.5. It can be seen in this plot that γ-SSL$_{\text{iL}}$-SCOT outperforms each of the baselines in terms of MaxA$_{\text{MD}}$, MaxF$_{1/2}$ and PR, with this backed up by scores of AUROC and AUPR of 0.852 and 0.896 respectively compared with scores of 0.803 and 0.868 for the best performing baseline, Ens-PE$_5$.

The key thing to conclude from this experiment is that although the γ-SSL$_{\text{iL}}$ models have been trained to detect segmentation error in a specific target domain, the representation learned from this generalises well to general driving datasets. Therefore, this suggests that this method is a promising approach for mitigating segmentation error for driving datasets generally, as was hypothesised in the discussion in Section 6.1.1.

Another aspect worth noting is that misclassification detection metrics when testing γ-SSL$_{\text{iL}}$-SCOT on WildDash are lower than that of testing on SAX Scotland. This confirms that, if possible, it is still better to leverage unlabelled images from the domain which could be encountered during a deployment, as the highest quality uncertainty estimation can be learned in this way.

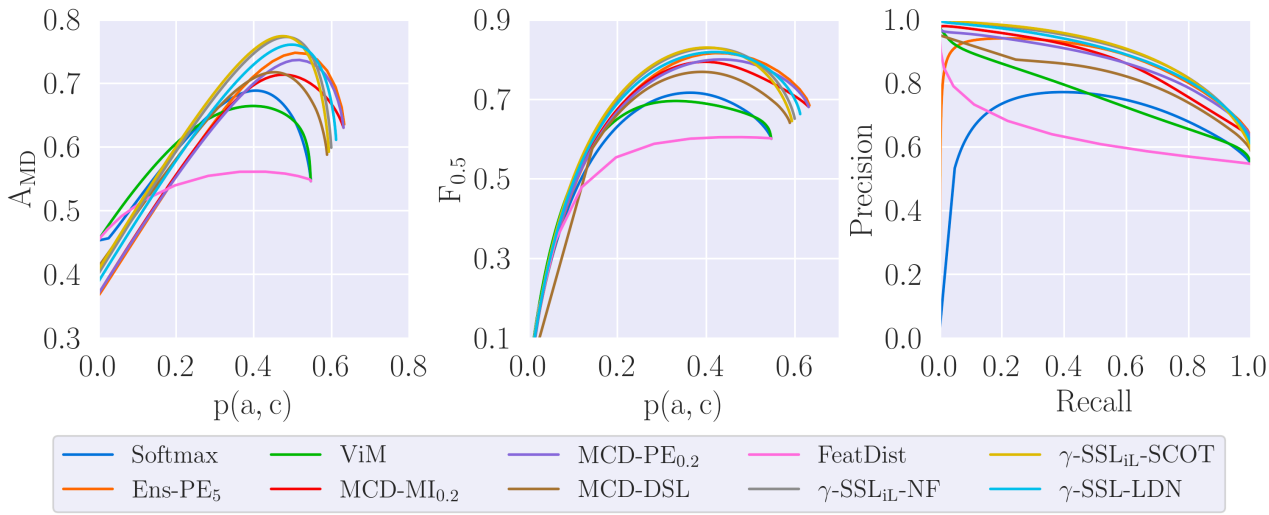


Figure 6.5: Misclassification detection results on the WildDash Dataset [134]. γ-SSL-LDN refers to a γ-SSL model trained on the SAX London unlabelled dataset, whereas γ-SSL$_{\text{iL}}$-NF, γ-SSL$_{\text{iL}}$-SCOT refer to γ-SSL$_{\text{iL}}$ models that are trained on the SAX New Forest and SAX Scotland unlabelled datasets, while also using SAX London as part of a curriculum.

6.11 Miscellaneous experiments

In addition to the improvements in uncertainty estimation quality, we investigate a number of other facets of our proposed method.

Firstly, the results in Section 6.11.1 show how the number of labelled validation images used to calculate an optimal threshold affects the quality of uncertainty estimation. The proposed method calculates a threshold during training, and this section shows that this threshold provides similar uncertainty estimation quality, without the requirement for labelled validation images.

Secondly, the results in Section 6.11.2 show the extent to which the optimal threshold calculated for one domain is optimal for testing on another domain. They show that there is insignificant degradation in quality of uncertainty estimation due to using a threshold from another domain.

Finally, Section 6.11.3 presents the latency of each of the baselines (described in Section 6.8), and shows that our proposed models are significantly faster than high quality alternatives.

6.11.1 Calculation of the optimal threshold

When considering the $\mathrm{A_{MD}}$ and $\mathrm{F_{1/2}}$ scores, there is an obvious point at which these scores are maximised (seen in Figure 6.4), which corresponds to an optimal threshold. This threshold can be calculated with a set of labelled validation images from the domain of interest. This requires using additional labelled validation images, and if this validation set is not representative of the test set, the uncertainty estimation quality could be worse.

For this reason, we investigate the effect of the number of validation images on uncertainty estimation quality for the γ-SSL models. We do this by holding out a validation set from the test images, and testing on these remaining images. This is done over 100 difference combinations of validation and test set for each size of validation set. Interestingly, given that γ-SSL also calculates a threshold during training, we can get a threshold without using any validation images and this is also investigated. The mean and spread of the $\mathrm{A_{MD}}$ metric is shown as a box plot, Figure 6.6, for $[0, 1, 5, 20]$ validation images.

Figure 6.6 shows that the $\mathrm{A_{MD}}$ metrics are much more variable for small numbers of validation images, as the validation set is less likely to represent the test set. It also shows that there is no significant drop in uncertainty estimation performance by using 0 validation images versus using 20, which demonstrates a very useful characteristic of the γ-SSL



Figure 6.6: Box plot representing the achieved A_{MD} by calculating the uncertainty threshold with varying numbers of validation examples for a γ-SSL model. The dashed lines represent the values of A_{MD} achieved when using the entire test dataset to calculate the optimal threshold, before then testing on it.

method.

6.11.2 Calculating thresholds across domains

It is also important to look at the effect of using a threshold calculated to maximise uncertainty estimation performance in one domain on the uncertainty estimation performance in another domain. The concern would be that a threshold that is chosen on one domain, and is therefore chosen for deployment, might be very sub-optimal for another domain, leading to dangerously poor uncertainty estimation.

This is investigated for a given domain, by comparing the $F_{1/2}$ at the optimal threshold to the $F_{1/2}$ score obtained using a threshold calculated from different domains. The results for this can be found in Table 6.9. For each γ-SSL_{iL} model trained with unlabelled data its target domain (e.g. γ-SSL_{iL}-LDN is trained using SAX London unlabelled images) we calculate the $F_{1/2}$ score using the optimal threshold for that given model in its target domain and report the percentage change as Δ.

This table shows that there is only a very slight decrease in uncertainty estimation quality as a result of using the threshold that is optimal for a different domain.

Table 6.9: Cross-Domain Threshold Testing Results

(a) γ-SSL$_{\text{iL}}$-LDN

	$\text{MaxF}_{1/2}$	$\text{F}_{1/2}$ with γ-LDN	Δ
LDN	0.8930	0.8930	0%
NF	0.8827	0.8822	-0.058%
SCOT	0.7853	0.7839	-0.175%

(b) γ-SSL$_{\text{iL}}$-NF

	$\text{MaxF}_{1/2}$	$\text{F}_{1/2}$ with γ-NF	Δ
LDN	0.8834	0.8832	-0.017%
NF	0.8849	0.8849	0%
SCOT	0.7952	0.7949	-0.038%

(c) γ-SSL$_{\text{iL}}$-SCOT

	$\text{MaxF}_{1/2}$	$\text{F}_{1/2}$ with γ-SCOT	Δ
LDN	0.8768	0.8764	-0.038%
NF	0.8806	0.8805	-0.007%
SCOT	0.8257	0.8257	0%

6.11.3 Latency Evaluation

We have made the argument that the epistemic methods can provide high quality uncertainty estimates, however are computationally very demanding. To prove this quantitatively, we have provided the frequency at which different methods can operate in Table 6.10. We run the methods on two very different pieces of hardware, namely a NVIDIA V100 GPU and the CPU on a MacBook Pro containing a M2 Pro CPU.

Table 6.10: Timing Results

Method	GPU [Hz]	CPU [Hz]
Vanilla	159.12	1.53
MCD	19.62	0.46
Ens-5$^{\text{LM}}$	5.27	0.75
Ens-5$^{\text{HM}}$	27.22	0.75
Ens-10$^{\text{LM}}$	2.99	0.38
Ens-10$^{\text{HM}}$	16.64	0.38
γ-SSL	183.37	1.52

The Vanilla method is a DeepLabV3+ segmentation network and thus represents the representation methods apart from DUM. The difference between Vanilla and γ-SSL (which is architecturally the same as γ-SSL$_{\text{iL}}$) is that the latter uses prototype segmentation rather than a segmentation head. The definitions of MCD and Ens are the same as those in previous experiments, where 8 samples are taken from the MCD model and the size of the ensemble is

shown in the method names.

For the ensemble methods, we use two different types of inference, denoted by the superscript LM and HM. These stand for **L**ow **M**emory and **H**igh **M**emory, where the former only loads one member of the ensemble on the GPU at the same time, and the latter loads each member onto the GPU then performs inference on the members sequentially. The LM method is therefore slower but uses less memory compared with the HM method.

The key finding here is the confirmation that the epistemic methods have a significantly lower frequency of operation, and therefore a higher latency. Our γ-SSL models are the same speed or faster than a standard segmentation network.

6.12 Qualitative Results

In Figure 6.7, we present test images from the SAX Test Datasets with an image from the source domain, alongside their semantic segmentation over the known classes (middle) and then segmentation into the known classes *or* uncertain (right).

As can be seen, the γ-SSL model is trained to semantically segment the source domain very accurately, however when presented with distributionally shifted images, the segmentation quality begins degrade (middle). This is however mitigated by the γ-SSL models by expressing uncertainty over regions of the segmentation that are inaccurate. To describe a few of the examples:

- In the first SAX London image, there is a street sign which is unfamiliar to a model given the semantic definitions in Cityscapes
- In the second SAX New Forest image, there is a red telephone box, the appearance of which would not be found in the German cities in which Cityscapes is collected, and is not part of the known classes.
- In the first SAX Scotland image, there is a pile of wood, which is unlikely to be found in a German city, and not one of the known classes.

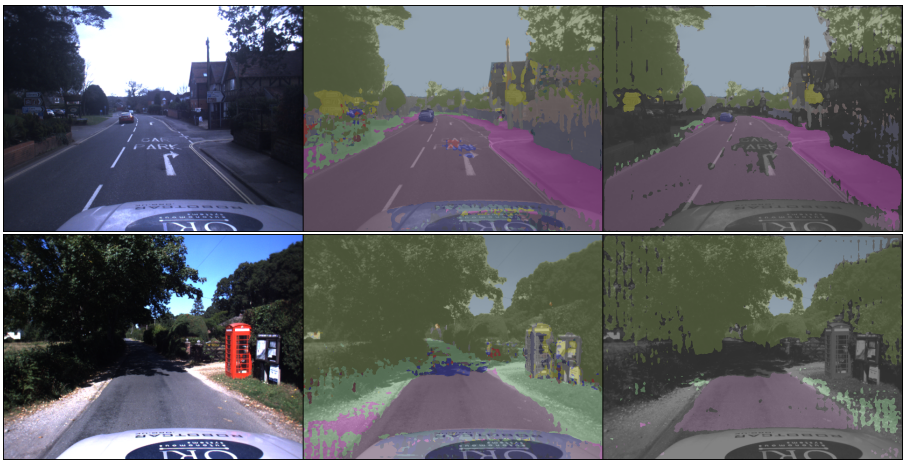
By looking at the right column in Figure 6.7, each of these instances are detected and assigned high uncertainty.



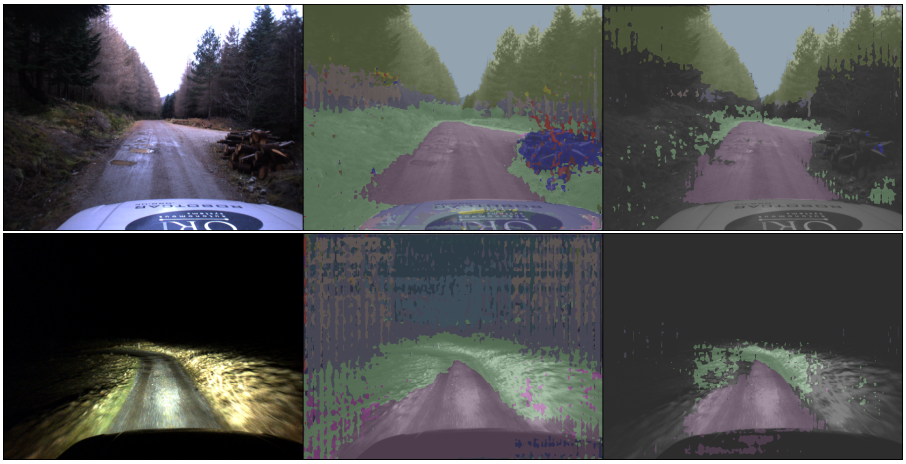
(a) Cityscapes



(b) London



(c) New Forest



(d) Scotland

Figure 6.7: Qualitative results for Cityscapes and the SAX domains. As the SAX RGB images (left) become more dissimilar from Cityscapes (from top to bottom), the corresponding semantic segmentations (centre) decrease in quality. *However*, for these poorly segmented regions, high uncertainty is largely expressed over them, shown in black (right).

6.13 Ablation Studies

In order to investigate the effect of each component of the proposed system, we perform a series of ablation studies. From these experiments, we are able to confirm that the choices made in the design of the system are beneficial.

6.13.1 Estimating Uncertainty via Distance to Prototypes

This experiment investigates to what extent does the γ-SSL work as a result of the improvement in the learned representation or the method for calculating uncertainty, i.e. by calculating the mean feature vector per class and computing the distance between the prototypes and a given pixel-wise feature of a test image.

This investigation is implemented by training a network on the source domain alone, and then calculating prototypes as we do in γ-SSL. In this way, the representation being used is the same as the representation baselines (apart from DUM) and so a direct comparison over uncertainty estimation inference procedures can be made. The results for this ablated method are under the name NoSSL, as it is γ-SSL without performing the semi-supervised learning on the unlabelled target images.

By comparing Table 6.11 to Table 6.2 and Table 6.3, we can see that this ablation outperforms the representation methods. This means that estimating uncertainty in this manner is clearly a good method, and supports the assumption that calculating M^{γ} in the way that is done is an important part of setting up the positive feedback loop.

However the key finding is that the performance of NoSSL is significantly worse than γ-SSL and γ-$\mathrm{SSL}_{\mathrm{iL}}$. This is important evidence that the training in these methods allow the model to learn a representation that is more suited to performing uncertainty estimation.

6.13.2 Importance of target domain images

In the proposed method, we have performed a different type of training with a different type of images. It is therefore worth determining if the benefits of the training method are independent of the type of data used. In this experiment, we use Cityscapes images instead of the unlabelled target domain images. The results for these are therefore under the name

Table 6.11: Misclassification Detection results over three metrics for the models trained with ablated versions of the method proposed in this work.

		AUROC			AUPR		
	Method	LDN	NF	SCOT	LDN	NF	SCOT
Ablations	NoSSL	0.803	0.774	0.72	0.872	0.829	0.686
	NoSAX	0.793	0.771	0.669	0.824	0.794	0.455
	$\texttt{M}^{\gamma=-\infty}$	0.805	0.761	0.711	0.9	0.827	0.646
	SymParam	0.851	0.598	0.622	0.8	0.539	0.486
	SymNonParam	0.643	0.643	0.621	0.382	0.235	0.165
	NoRegL	0.815	0.827	0.72	0.882	0.847	0.621
	MCD-SSL	0.776	0.81	0.705	0.794	0.857	0.592
Ours	γ-SSL	**0.895**	0.88	0.776	**0.949**	0.921	0.726
	γ-$\texttt{SSL}_{\texttt{iL}}$	-	**0.88**	**0.859**	-	**0.942**	**0.887**

(a)

		MaxA_{MD} @ p(a, c)		
	Method	LDN	NF	SCOT
Ablations	NoSSL	0.727 @ 0.482	0.706 @ 0.424	0.680 @ 0.204
	NoSAX	0.729 @ 0.465	0.717 @ 0.463	0.653 @ 0.003
	$\texttt{M}^{\gamma=-\infty}$	0.754 @ 0.586	0.692 @ 0.426	0.685 @ 0.174
	SymParam	0.790 @ 0.253	0.596 @ 0.368	0.756 @ 0.053
	SymNonParam	0.752 @ 0.015	~~0.864 @ 0.000~~	~~0.902 @ 0.000~~
	NoRegL	0.766 @ 0.552	0.748 @ 0.391	0.665 @ 0.201
	MCD-SSL	0.747 @ 0.594	0.748 @ 0.482	0.707 @ 0.153
Ours	γ-SSL	**0.83** @ 0.625	0.796 @ 0.483	0.716 @ 0.260
	γ-$\texttt{SSL}_{\texttt{iL}}$	-	**0.815** @ 0.608	**0.781** @ 0.431

~~Struck through~~ results are discounted as no pixels are confidently segmented.

(b)

		$\text{MaxF}_{1/2}$ @ p(a, c)		
	Method	LDN	NF	SCOT
Ablations	NoSSL	0.790 @ 0.380	0.753 @ 0.364	0.626 @ 0.203
	NoSAX	0.769 @ 0.401	0.756 @ 0.403	0.505 @ 0.220
	$\texttt{M}^{\gamma=-\infty}$	0.826 @ 0.492	0.746 @ 0.353	0.602 @ 0.184
	SymParam	0.731 @ 0.219	0.608 @ 0.381	0.495 @ 0.057
	SymNonParam	0.394 @ 0.088	0.281 @ 0.030	0.210 @ 0.017
	NoRegL	0.806 @ 0.507	0.781 @ 0.332	0.569 @ 0.241
	MCD-SSL	0.809 @ 0.518	0.804 @ 0.416	0.589 @ 0.159
Ours	γ-SSL	**0.893** @ 0.548	0.855 @ 0.407	0.678 @ 0.239
	γ-$\texttt{SSL}_{\texttt{iL}}$	-	**0.885** @ 0.532	**0.826** @ 0.370

(c)

NoSAX.

Looking at Table 6.11, it can be seen that the AUROC and AUPR of NoSAX are worse than that of γ-SSL. This makes it clear that using target domain images is important to performing good uncertainty estimation. This puts into focus the importance of, not just training algorithm, but the type of data in this method.

6.13.3 Importance of M^{γ} in the training objective

In Section 6.5, we make the point that is important to only maximise the consistency over pixels that are likely to be accurate (approximated by those that are certain). In this experiment, we question this and apply more standard semi-supervised learning in which we maximise the consistency over all pixels. It can be argued that the semi-supervised learning will improve the representation of the target domain, such that uncertainty estimation will be improved, and thus it is not necessary to specialise the objective for uncertainty estimation. This experiment is called $\mathrm{M}^{\gamma=-\infty}$.

The results for this, seen in Table 6.11, show that quality of uncertainty estimation for γ-SSL is better than that of $\mathrm{M}^{\gamma=-\infty}$. Therefore, we can conclude that there is benefit to specialising the objective specifically for uncertainty estimation.

6.13.4 Importance of Branch Asymmetry

In Section 6.5.4, we make the point that having two segmentation functions f and g that are not the same helps to prevent feature collapse. To empirically investigate this, we use two different methods: SymNonParam and SymParam. Both of these methods have symmetric branches, but for the former, both branches use prototype segmentation (hence symmetric non-parametric segmentation), while for the latter, they both use segmentation heads (hence symmetric parametric segmentation).

In each method's case, the same feature collapse nearly always occurs, where pixels are segmented typically as one or two classes in a fixed pattern for all images, where one of these classes is almost always road. The only exception to this is for SymNonParam on the SAX New Forest dataset. The characteristics of this collapse is that the segmentation accuracy becomes very poor, even while the AUROC and AUPR are reasonably good. So firstly, these

experiments provide us with evidence that having asymmetric branches helps to prevent feature collapse, as can been seen in Table 6.11. Secondly, it gives a good illustration of how important it is not just to look at AUROC and AUPR when evaluating a model, and that incorporating $\mathrm{p}(\mathrm{a}, \mathrm{c})$ into $\mathrm{MaxF}_{1/2}$ and $\mathrm{MaxA}_{\mathrm{MD}}$ is a useful way of doing this.

6.13.5 Importance of L^{u} and L^{p}

We presented the use of L^{u} and L^{p} as objectives to help prevent feature collapse and to benefit uncertainty estimation. By removing these losses, we can investigate their effect, in a method named `NoRegL`.

When using this method, as can be seen in Table 6.11, there is not a complete feature collapse due to the presence of using branch asymmetry as well, but the quality of uncertainty estimation is significantly reduced. This confirms for us the benefit of uniformity in the representation for uncertainty estimation, and backs the hypothesis that this encourages selectivity in which pixels can be near the prototypes and which cannot.

6.13.6 Importance of Crop-and-Resize Data Augmentation

The entirety of this method is designed using the foundation of crop-and-resize data augmentation. The argument is made that we can investigate the limits of the segmentation network's knowledge by perturbing images in this way and looking at the resulting segmentation performance, as performed in SSL.

Alternatively, as per the epistemic methods, it is also possible to consider the model parameter distribution, by perturbing the model parameters instead of the training images. Doing this, we can define a method that uses dropout instead of data augmentation, and then compare segmentations produced by two perturbation of the model parameters. For this method, the dropout probability of 0.2 is used as this was found to be optimal for the baselines. This method is given the name `MCD-SSL`.

As seen in Table 6.11, this method does largely work, however it does not produce the quality of uncertainty estimates as γ-`SSL` does. This justifies the use of data augmentation in order to induce the distribution of possible segmentations, which encodes distributional uncertainty.

6.13.7 Importance of using hard $\mathbf{M}^\gamma$

In the method, we make binarize the uncertainty estimations to get $\mathbf{M}^\gamma \in \{0,1\}^{H\times W}$. We can try and determine if this was a good design choice by investigating the alternative which is to use soft $\tilde{\mathbf{M}}^\gamma \in \mathbb{R}^{H\times W}$. To do this, we propose to calculate $\tilde{\mathbf{M}}^\gamma$ for a given pixel i in the following way:

$$\tilde{\mathbf{M}}_i^\gamma = \mathrm{norm}(\max[\mathrm{softmax}_\tau(\mathbf{l}_i)]) \in [0,1] \tag{6.14}$$

Where $\max[\mathrm{softmax}_\tau(\mathbf{l}_i)]$ is $\mathbf{p}_{\max}$, and $\mathrm{norm}(\cdot)$ is a normalisation function. This normalisation function is needed as the scaling of the uncertainty estimates is not in any way guaranteed to be optimal for use in training. norm is chosen to normalise each $\tilde{\mathbf{M}}^\gamma \in \mathbb{R}^{H\times W}$, such that the minimum and maximum values are 0 and 1.

This experiment is performed using Cityscapes as the source domain and SAX London as the target domain. When testing this model again on the SAX London domain, the $\mathrm{MaxF}_{1/2}$ score is 0.862 at a value of $\mathrm{p(a,c)}$ of 0.421, with a corresponding segmentation accuracy of 0.576. The corresponding γ-SSL model with a binary $\mathbf{M}^\gamma$ has a $\mathrm{MaxF}_{1/2}$ score of 0.893 with a $\mathrm{p(a,c)}$ of 0.548, and a segmentation accuracy of 0.70.

This shows that the quality of uncertainty estimation is worse using $\tilde{\mathbf{M}}^\gamma$ and the segmentation quality even more significantly reduced. Therefore, the soft certainty mask adds noise to the consistency task in a way that makes it much more difficult to learn a good representation of the target domain.

6.13.8 Possibility of class-wise thresholds

In the γ-SSL methods, we define a single value as the threshold for each of the class prototypes. It can however be argued that it might be more optimal to have a different threshold for each class prototype, as different classes are likely to be distributed differently, owing to their different prevalence's and appearance differences. For example, the `road` class is very prevalent, and yet is almost always looks the same, and so you might expect it to be embedded quite tightly. However a class such as `traffic sign` varies much more in appearance and it corresponds to far fewer pixels in the training datasets, therefore you can imagine that this class would have a much larger spread.

To investigate whether this is an important aspect of the problem to consider, we develop a method that solves for a different threshold for each class. The thresholds for each class are calculated such the per-class certainty $[p_\gamma]_k$ and per-class consistency $[p_c]_k$ are equal, just as in the above method.

$$[p_\gamma]_k = \sum_{i=1}^{NHW} \frac{\mathbb{1}[\text{argmax}(\mathbf{l}_i) = k] \odot \mathbf{M}_i^\gamma}{\sum_{j=1}^{NHW} \mathbb{1}[\text{argmax}(\mathbf{l}_j) = k]} \tag{6.15}$$

$$[p_c]_k = \sum_{i=1}^{NHW} \frac{\mathbb{1}[\text{argmax}(\mathbf{l}_i) = k] \odot \mathbf{M}_i^\text{c}}{\sum_{j=1}^{NHW} \mathbb{1}[\text{argmax}(\mathbf{l}_j) = k]} \tag{6.16}$$

By equating these we can calculate the per-class thresholds $\Gamma = [\gamma_1, \gamma_2, ...\gamma_K] \in \mathbb{R}^K$. Then the certainty mask $\mathbf{M}^\gamma$ is calculated for every pixel i:

$$\mathbf{M}_i^\gamma = \mathbb{1}[\max(\mathbf{l}_i) > \gamma_{k=\text{argmax}(\mathbf{l}_i)}] \tag{6.17}$$

Using this described method, the uncertainty estimation quality was significantly worse than for γ-SSL with a single threshold. It reported a $\text{MaxF}_{1/2}$ and $\text{p}(\text{a}, \text{c})$ of 0.772 @ 0.432 versus 0.893 @ 0.548 for γ-SSL.

Therefore, using a class-wise threshold degrades the learning of good uncertainty estimation, and so it is both easier and more performant to use a single threshold as in our method.

6.13.9 The need for large batch sizes to calculate prototypes

In the γ-SSL method, the prototypes are re-calculated at each iteration from a sampled batch of labelled source images. Therefore, it is possible to argue that the γ-SSL method is too computationally demanding due to the need to have large batch sizes (and thus large GPU memory usage) in order to prevent the prototypes from being too noisy.

The concerns for this problems are diminished by the fact that the batch size used in training was 12, meaning that the training could fit on a single GPU with 12GB of vRAM. We can however also consider how small the batch size can be in order to successfully learn to perform uncertainty estimation, in order to investigate if the training would work with less good hardware or scale to larger image sizes.

This method implements a scheme that prevents the issues associated with the limits caused by calculating the prototypes at every training iteration. When sampling a batch of labelled images, there is no guarantee that this batch contains pixels that are labelled with every one of the known classes. Therefore, without counting for this, the model cannot assign pixels in the target image to one of these left-out classes. To prevent this we use a history of class prototypes, where if a class is not present in the source batch, then the most recently calculated prototype for that class is used. This helps to prevent the problems associated with the need for large batch sizes.

We perform an experiment to investigate this problem, where the number of images used to calculate the prototypes is varied, however the batch size for the rest of the training method is kept the same. Then during testing, the prototypes were calculated using the entirety of the labelled source dataset, as was done in all previous testing. This allows us to investigate the effect of smaller batch sizes for the prototypes on the learned representation. In Table 6.12, we report the $\mathrm{MaxF}_{1/2}$ and $\mathrm{p}(\mathrm{a}, \mathrm{c})$ metrics for a range of prototype batch sizes, along with segmentation accuracy.

Table 6.12: Results for varying Training Prototype Batch Size on SAX London

Prototype Batch Size	Use history?	$\mathrm{MaxF}_{1/2}$ @ $\mathrm{p}(\mathrm{a}, \mathrm{c})$	Segmentation Accuracy
12	Yes	0.893 @ 0.548	0.703
8	Yes	0.888 @ 0.559	0.719
6	Yes	0.892 @ 0.546	0.712
2	Yes	0.882 @ 0.518	0.693
2	No	0.827 @ 0.538	0.690

Table 6.12 shows that if we maintain a history of prototypes, then the quality of uncertainty estimation and semantic segmentation does not significantly depend on the used batch size. However when the history is not used, there is a significant hit to the quality of uncertainty estimation. This is because, despite there being prototypes for the most prevalent classes, e.g. road, building and sky, and the segmentation accuracy can be mostly the same, the model has prototypes which are vectors of zeros for the rest, therefore the categorical distribution from which the uncertainty is calculated is significantly affected. However, it is interesting that the uncertainty estimation quality does not completely degrade in this setting, perhaps owing to the robustness of using cosine distance as a measure of uncertainty.

6.14 Conclusion

This chapter has presented a method to perform distributional uncertainty estimation which uses an unlabelled target domain training dataset, which contains instances that are in-distribution, near-distribution and OoD within the same image. While challenging to work with due to the lack of segmentation ground-truth, it promised to lead to higher-quality distributional uncertainty estimation when compared with using a training dataset where every image is significantly distributionally shifted from the source domain.

Empirically, it has been shown that this does indeed lead to high-quality distributional uncertainty estimation. The significant challenge in this work became apparent when using an unlabelled target domain that was significantly shifted from source domain, namely SAX Scotland. For this case, the quality of both segmentation and uncertainty estimation degraded due to the large distributional shift. In this work, this was solved by using a curriculum, and firstly training on a less distributionally shifted domain, namely SAX London.

Solving this problem is of great interest to us, because it is very important that quality of uncertainty estimation does not degrade in the same manner as the quality of segmentation. This problem will be further investigated in the next chapter, with a method that firstly uses large-scale self-supervised training to learn a general feature representation, such that large distributional shifts are not as damaging to learned uncertainty estimation.

Chapter 7

Learning Uncertainty Estimation with Masking & Foundation Models

Contents

7.1 Foundation Models 149

7.1.1 Preliminaries on Foundation Models 149

7.1.2 Foundation Models for Semantic Segmentation 149

7.1.3 Model Distillation 151

7.2 Uncertainty Estimation for Foundation Models 152

7.3 Training Framework 153

7.4 Uncertainty Training 154

7.4.1 Learning Uncertainty Estimation 155

7.4.2 Avoiding Feature Collapse 158

7.5 Masked Image Modelling for Uncertainty Estimation 159

7.5.1 Background 159

7.5.2 Motivation 160

7.5.3 Masking Policy 161

7.6 Experimental Setup 162

7.6.1 Network Architecture 162

7.6.2 Network Initialisation 162

7.6.3 Data 163
7.6.4 Baselines 164
7.7 Experiments and Results 165
7.7.1 Different Target Domains 166
7.7.2 Freezing E 167
7.7.3 Comparing perturbation methods 168
7.7.4 Freezing $\mathtt{f}_\theta$ 169
7.8 Conclusion 169

This chapter presents a method that builds on the ideas presented in both Chapter 5 and Chapter 6. It considers the same problem setting of Chapter 6, where labelled images from a source domain and unlabelled images from a distributionally shifted target domain are used to train a model to perform uncertainty estimation. However, much like Chapter 5, it also considers the use of large-scale image datasets for learning a general task-agnostic representation.

The way in which these two approaches are combined is influenced by the recent advent of foundation models in computer vision. These are neural networks that are trained on a broad distribution of image data, in order to be capable of solving many different computer vision tasks with minimal additional training.

This chapter investigates the way in which these models can be used to improve the quality of distributional uncertainty estimation, and presents a training framework for this in Section 7.3. The crucial final step in this framework is a method that we present in Section 7.4, which trains a model to learn uncertainty estimation from unlabelled target domain data using a Masked Image Modeling (MIM) task.

Experiments, described in Section 7.6, are performed in order to quantify the benefits of using a foundation model for uncertainty estimation, and the additional improvements seen when using our proposed method. The results for these experiments are presented in Section 7.7.

This work was first presented in:

- D. Williams, M. Gadd, P. Newman, and D. De Martini, "Masked γ-SSL: Learning Uncertainty Estimation via Masked Image Modeling", IEEE International Conference on

Robotics and Automation (ICRA), 2024.

7.1 Foundation Models

7.1.1 Preliminaries on Foundation Models

Foundation models are deep neural networks that are trained such that they can easily be adapted to solve a wide range of downstream tasks on a broad distribution of data. In order for this to be possible, they need to learn a general feature representation, i.e. they need to extract salient information from a diverse set of data. In the case of computer vision, foundation models need to be able to extract semantic and geometric information from any given natural image.

Currently, the requirements for achieving this are: (1) a very large neural network must be used, and (2) it must be trained on a very large and diverse dataset. As a result of the second of these requirements and the cost of annotating data, self-supervised learning is used to train these models, which has been discussed previously in Section 3.5.3, Chapter 5 and Chapter 6.

For computer vision, as well as other modalities, foundation models are based on the transformer neural network architecture [37]. This architecture has been adapted from its NLP origins for computer vision applications, with the primary change being to how the input is tokenised. In architectures such as ViT [36] and its variants, an input image is split into patches, which are each individually embedded. The patch embeddings are then processed by the standard transformer self-attention blocks proposed in [37].

As a result, computer vision foundation models take the form of a transformer encoder, which is represented by $\mathtt{E} : \mathbb{R}^{3\times H\times W} \rightarrow \mathbb{R}^{F\times \hat{H}\times \hat{W}}$, where F is the length of the output features and $(\hat{H}, \hat{W})$ are the spatial dimensions. Due to the patch tokenisation, $(\hat{H}, \hat{W}) = (\frac{H}{P}, \frac{W}{P})$ where P is the patch size.

7.1.2 Foundation Models for Semantic Segmentation

It is possible to apply these types of foundation models to a task without any adaptation via non-parametric inference with the features, e.g. nearest-neighbour classification or pro-

totype segmentation (described in Section 6.2.1). However, in order to maximise the performance on a given task, the parameterisation of the model is typically changed in some way. A simple example of this is the addition of a linear layer on top of the model for classification, which outperforms nearest-neighbour classification on ImageNet, as seen in [31], [32], [106].

A number of different methods can be used to adapt a transformer encoder for semantic segmentation. The simplest is to freeze the parameters of the encoder, and train a segmentation decoder on top with a pixel-wise labelled dataset. Many different segmentation decoders have been proposed to do this, using both convolutional neural network (CNN) and transformer blocks, e.g. [38]–[40], [148].

The above approach changes only how the features $\mathbf{z}$, where $\mathbf{z} = \mathtt{E}(\mathbf{x})$, are processed into segmentations, $\mathbf{z} \rightarrow \mathbf{y}$. Alternatively, the encoder itself can be adapted in a number of different ways. Firstly, the encoder can be fine-tuned with straightforward supervised learning along with the decoder to solve a specific task. This, however, incurs a large training cost for very large neural networks and so parameter-efficient fine-tuning (PEFT) methods, such as LoRA [149], can be used.

LoRA leaves the original encoder parameters unchanged, but modifies the forward pass by adding trainable low-rank weight matrices $\Delta\theta$, such that for the i^{th} layer $\mathtt{E}_i$ of the encoder $\mathtt{E}$, $\mathtt{E}_i(\mathbf{x}) = \theta_i\mathbf{x} + [\Delta\theta]_i\mathbf{x}$. Crucially, for $\theta_i \in \mathbb{R}^{d\times k}$, these additional low-rank weight matrices are composed as $\Delta\theta_i = BA$, where $B \in \mathbb{R}^{d\times r}, A \in \mathbb{R}^{r\times k}$ and r is small, e.g. $r = 8$. This means that a small number of additional parameters need to be trained, leading to (1) lower training cost, (2) no inference cost as θ and $\Delta\theta$ can be combined, and (3) lower storage costs as a model can be fine-tuned for a range of tasks with the only difference between these task models being the addition of a small number of parameters, which are trivial to store.

A related approach to this is to augment the encoder architecture by adding additional parameterised layers, such as in the adapter methods [150], [151]. Along with the decoder, these layers are only trained with task-specific supervised learning.

The method in this chapter uses a segmentation decoder of the form found in Mask2Former [148]. As will be discussed in Section 7.6, we perform experiments both with and without encoder fine-tuning, in order to investigate how this affects uncertainty estimation (discussed

further in Section 7.2).

7.1.3 Model Distillation

The methods discussed in Section 7.1.2 adapt the model to maximise its performance on a specific task. Another reason to adapt the model is to reduce its size. Foundation models are very large, and so incur a significant cost at inference time, which may be too great for deployment on a mobile robot. As a result, it may be necessary to distill the representation learned in the large foundation model into a model with fewer parameters.

Knowledge distillation, presented in [152], is a method that improves the supervised training of a small model (named the *student*) by approximating the output of a larger, more accurate pretrained model (named the *teacher*). More specifically, soft pseudo-labels are generated by the teacher model by applying a softmax function with a high temperature to the logits, which are used to train the student alongside the supervised training. A similar approach is seen in [153], where the supervised distillation task is tailored to the transformer architecture.

This idea is also applied to self-supervised learning, where, much like in supervised learning, smaller models perform less well in training and testing. In [154], a self-supervised method for distilling the representation of a large self-supervised model into a small model is presented. A similar method is presented in DINOv2 [31], where a ViT-Giant with 1 billion parameters is distilled into a smaller ViT-Small model with 22 million parameters. This distillation method was shown in [154] and [31] to be significantly more effective than performing SSL training from a random initialisation.

In this chapter, we are interested in training a model that has a low inference cost, while also benefitting from recently published foundation models. For this reason, we use the distilled ViT-Small presented in DINOv2. This is important, as it allows us to investigate the use of a model that is very general, but that is not prohibitively large for a mobile robotics context.

7.2 Uncertainty Estimation for Foundation Models

This thesis is interested in mitigating distributional shift and, as such, we must consider the impact of recent foundation models on this.

There are two key considerations: (1) the extent to which fine-tuning a foundation model degrades its generalisation ability, and (2) the extent to which foundation models can help improve distributional uncertainty estimation as a result of their general representation.

As discussed in Section 7.1.2, in order to solve specific segmentation tasks, foundation models need to be fine-tuned with a labelled training dataset. In doing this, the model is optimised on what is likely to be a relatively small dataset[1], and therefore the span of data over which the model is optimal is likely to shrink as a result of fine-tuning (see the discussion in Section 2.2). This means that, although the original foundation model may not suffer greatly from distributional shift, the resultant fine-tuned model will suffer to a greater extent. This is shown in [155], where, as a model is fine-tuned on one dataset, the classification accuracy decreases substantially for seven different datasets.

Additionally, given the arguments made in Section 2.3.3 and Section 3.5.1 that neural networks typically cannot detect decreases in accuracy, and that supervised training does not promote task-agnostic representations, it is also likely that the quality of the model's uncertainty estimates will decrease. Experiments described in Section 7.6, with results presented in Section 7.7, investigate whether this is empirically true.

As for the second consideration, given that foundation models are trained in such a way as to learn task-agnostic information from diverse data, they also hold promise for being very good at distributional uncertainty estimation. This has been previously discussed in Section 3.5.2 and Section 3.5.3 in the context of OoD detection, and then further in Chapter 5 and Chapter 6.

We present a training framework in Section 7.3, and a method in Section 7.4, that address these considerations in order to train a semantic segmentation model which performs high-quality distributional uncertainty estimation.

[1] Certainly small relative to the pretraining dataset owing to the cost of annotation.

7.3 Training Framework

We propose a three-step framework that yields a foundation model fine-tuned for the semantic segmentation of a given domain, that also produces high-quality distributional uncertainty estimates.

The steps are (1) Pretraining, (2) Task Learning, (3) Uncertainty Training. The details of these steps will now be discussed, with an illustration of each found in Figure 7.1.

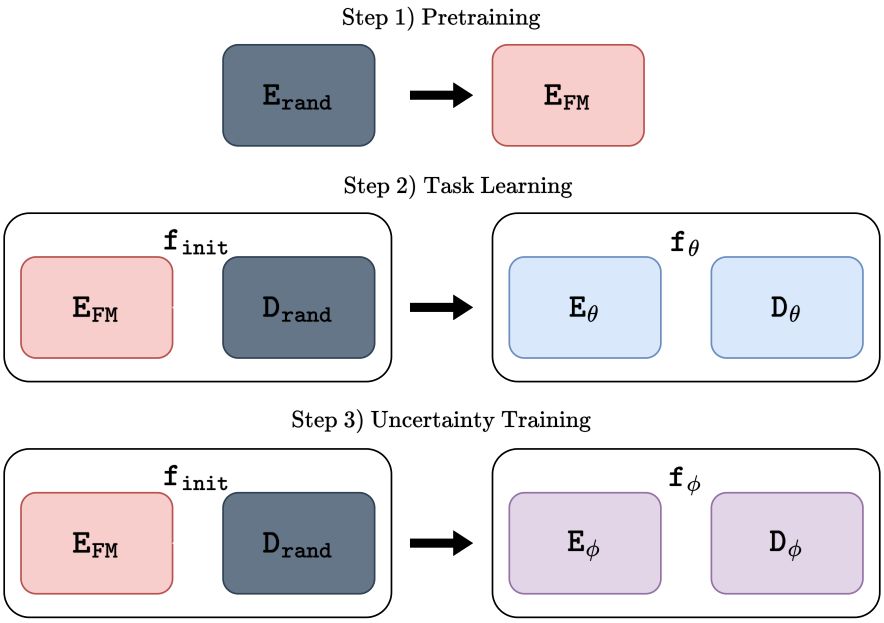


Figure 7.1: A illustration of the three training steps of encoders $\mathtt{E}$ and decoders $\mathtt{D}$, where the subscript denotes their parameterisation. Firstly, an encoder is trained to produce a good feature representation of a diverse set of natural images. This encoder forms the initialisation of steps 2 and 3, where the encoder is fine-tuned, and a decoder is trained from scratch. Each $\mathtt{E}$ and $\mathtt{D}$ are architecturally the same, but parameterised differently.

(1) Pretraining

This step trains an encoder $\mathtt{E}_{\mathtt{FM}}$ from scratch such that it can encode salient semantic and geometric information from a broad distribution of natural images, i.e. it trains a foundation model. As discussed in Section 7.1.3, if the deployment scenario necessitates a smaller model, this step can also involve distilling the learned representation into a smaller model. The resultant encoder $\mathtt{E}_{\mathtt{FM}}$ will be used to initialise the models that are trained in the next two steps, such that each can benefit from this general representation.

(2) Task Learning

In this step, a segmentation network is defined with the foundation model encoder $\mathtt{E}_{\mathtt{FM}}$, and a randomly initialised decoder $\mathtt{D}_{\mathtt{rand}}$. This segmentation network will be trained to perform a task of interest, resulting in the network $\mathtt{f}_\theta = \mathtt{D}_\theta \circ \mathtt{E}_\theta$.

In this work, the task is semantic segmentation of the source domain, which is achieved with the corresponding source domain labelled training dataset. Note that by 'task', we mean both whether we are considering image classification, depth regression, semantic segmentation etc., but also the domain of data being used. In order to maximise the task performance, i.e. maximising the quality of semantic segmentations on the source domain, the encoder is also fine-tuned, hence the transform from $\mathtt{E}_{\mathtt{FM}} \rightarrow \mathtt{E}_\theta$.

(3) Uncertainty Training

For this step, we propose a novel method, which is described in more detail in Section 7.4. After the network $\mathtt{f}_\theta$ has been trained to maximise its segmentation performance on the source domain, it is frozen.

Due to the task-specific training in the second step, the representation of $\mathtt{f}_\theta$ will be less task-agnostic, resulting in poorer quality segmentation and uncertainty estimation for distributionally-shifted target domains. As a result, we train another segmentation network $\mathtt{f}_\phi$, which can segment the source domain as well as $\mathtt{f}_\theta$, but is also able to detect error due to distributional shift. This is done by initialising the encoder of $\mathtt{f}_\phi$ with $\mathtt{E}_{\mathtt{FM}}$, and then jointly performing uncertainty training and task learning (from step (2)) to give $\mathtt{f}_\phi = \mathtt{D}_\phi \circ \mathtt{E}_\phi$, where again the encoder is fine-tuned from $\mathtt{E}_{\mathtt{FM}} \rightarrow \mathtt{E}_\phi$. While task learning promotes learning task-specific features, the aim of uncertainty training is to maintain and refine the task-agnostic features produced by $\mathtt{E}_{\mathtt{FM}}$, as per the requirements set out in Section 3.5.2 and Section 3.5.3.

In the next section, we detail the method we have designed for this final step.

7.4 Uncertainty Training

The goal of this method is to expose the task-specific model $\mathtt{f}_\theta$ to distributionally shifted images and to find regions where segmentations are likely to be incorrect. Then, while

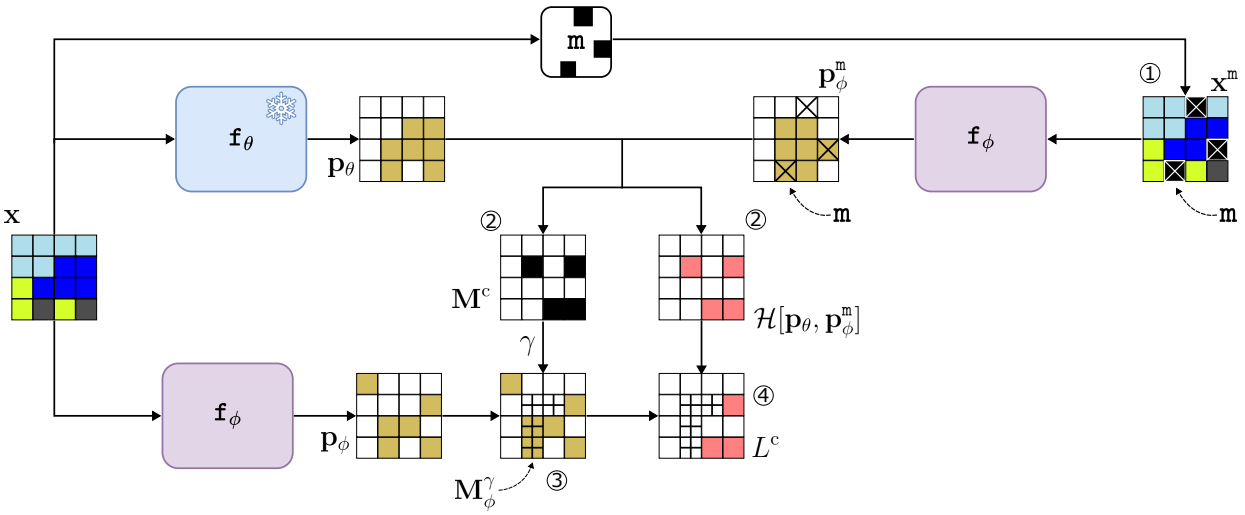


Figure 7.2: Overview for the training on $\mathtt{f}_\phi$ with the loss L^{c}. The training image $\mathbf{x}$ is perturbed by masking to give $\mathbf{x}^{\mathtt{m}}$ at ①. Then both images are segmented, and the soft consistency $\mathcal{H}[\mathbf{p}_\theta, \mathbf{p}_\phi^{\mathtt{m}}]$ and hard consistency $\mathbf{M}^{\mathrm{c}}$ between these segmentations are calculated, as seen at ②, where the latter is used to calculate the uncertainty threshold γ. The model $\mathtt{f}_\phi$ then segments the unmasked image and estimates its uncertainty over this segmentation, which is thresholded by γ, to give $\mathbf{M}_\phi^\gamma$ at ③. Finally, the loss L^{c} is calculated as the soft consistency masked by the binary uncertainty estimate $\mathbf{M}_\phi^\gamma$, seen at ④.

finding these erroneous image regions, the goal is to train a new network $\mathtt{f}_\phi$ to learn the appearance of these regions and to express uncertainty over them.

Designed to achieve this goal, the method presented in this chapter builds on the method in Chapter 6 by introducing the following changes to improve the training process:

- A masking policy is introduced to perturb the input image, such that incorrect segmentations can be found in a way that is less dependent on hyperparameters than a crop-and-resize task.
- The encoder is initialised with weights from a foundation model, which aids the training of a model that learns the relationship between image appearance and distributional uncertainty.
- One of the training branches is frozen, resulting in a training method that is less susceptible to feature collapse, a problem discussed in Section 6.5.4.

7.4.1 Learning Uncertainty Estimation

The objective for training $\mathtt{f}_\phi$ to learn distributional uncertainty estimation in this chapter is similar to the objective presented in Section 6.5. They rely on the assumption that the

consistency in segmentation of two images, which are perturbations of each other, can serve as an approximation of ground-truth segmentation accuracy. We describe the objective in a series of steps, which correspond to the numbers, e.g. ①, in Figure 7.2.

(1) Perturbing the input

In this work, the perturbed images are $\mathbf{x}$ and $\mathbf{x}^{\mathtt{m}}$, where the latter is a masked version of the former, i.e. $\mathbf{x}^{\mathtt{m}} = \mathtt{m}(\mathbf{x})$ for a masking function $\mathtt{m}$, as shown at ① in Figure 7.2. Therefore, instead of using a crop-and-resize policy as in Chapter 6, this work uses a masking policy in order to perturb the input image and find regions of likely segmentation error. This follows from the SSL literature, which is increasingly using masking as well as, or instead of, crop-and-resize tasks in order to learn representations from unlabelled images, as seen in [31], [112], [113]. The motivation and details of this masking policy are discussed in Section 7.5.

As previously discussed in Chapter 6, the aim of this type of training is to train a network to detect inconsistency as a proxy for detecting error, in much the same way as SSL tasks train a model to minimise inconsistency as a proxy for minimising task error. Therefore, considering consistency with respect to masking is an appropriate choice for this type of training.

(2) Calculating segmentation consistency

The perturbed images are segmented as $\mathbf{p}_\theta = \mathtt{softmax} \circ \mathtt{f}_\theta(\mathbf{x})$ and $\mathbf{p}_\phi^{\mathtt{m}} = \mathtt{softmax} \circ \mathtt{f}_\phi(\mathbf{x}^{\mathtt{m}})$, shown in Figure 7.2. The frozen segmentation network $\mathtt{f}_\theta$ has been trained to maximise segmentation quality in the source domain, and we are interested in determining when this network erroneously segments distributionally-shifted images. Depicted as ② in Figure 7.2, this is estimated by considering both the soft consistency $\mathcal{H}[\mathbf{p}_\theta, \mathbf{p}_\phi^{\mathtt{m}}] \in \mathbb{R}^{H \times W}$, where $\mathcal{H}[\cdot, \cdot]$ is the cross-entropy function, and the hard consistency $\mathbf{M}^{\mathtt{c}} \in \{0, 1\}^{H \times W}$, where for a pixel location i:

$$\mathbf{M}_i^{\mathtt{c}} = \begin{cases} 1 & \text{if } \operatorname{argmax}[\mathbf{p}_{\theta,i}] = \operatorname{argmax}[\mathbf{p}_{\phi,i}^{\mathtt{m}}] \\ 0 & \text{otherwise} \end{cases} \tag{7.1}$$

The consistency mask $\mathbf{M}^{\mathtt{c}}$ is used to calculate the uncertainty threshold γ in the same manner as in Section 6.5.1, where the proportion of pixels that are consistent is used as

an approximation of the proportion of pixels are certain, and therefore are estimated to be correctly segmented. This uncertainty threshold γ is used in the next step.

(3) Estimating uncertainty

In order to train $\mathtt{f}_\phi$ to estimate high uncertainty over erroneous regions using the approach presented in Section 6.5, we allow $\mathtt{f}_\phi$ to reduce the soft consistency loss via the binary uncertainty estimate $\mathbf{M}_\phi^\gamma \in \{0,1\}^{H\times W}$. In this way, if the network estimates that a segmentation is likely to be inconsistent, then it can express high uncertainty and mask out these pixels. Shown at ③ in Figure 7.2, this binary uncertainty estimate is calculated by thresholding the model's uncertainty estimate, which is calculated for a pixel location i as follows:

$$\mathbf{M}_{\phi,i}^\gamma = \begin{cases} 1 & \text{if } \max \circ \operatorname{softmax} \circ \mathtt{f}_\phi(\mathbf{x})]_i > \gamma \\ 0 & \text{otherwise} \end{cases} \tag{7.2}$$

Note that $\mathbf{M}_\phi^\gamma$ is calculated from a separate segmentation $\mathbf{p}_\phi = \operatorname{softmax} \circ \mathtt{f}_\phi(\mathbf{x})$, i.e. the segmentation by $\mathtt{f}_\phi$ of the *unmasked* image $\mathbf{x}$. This is important, as it is described in [31] that there is a performance gap for neural networks trained on masked images and tested with unmasked images, which is solved by full fine-tuning on unmasked images. In order to avoid this performance gap, $\mathtt{f}_\phi$ is directly trained to estimate uncertainty on unmasked images.

(4) Mitigating inconsistency

Finally, the objective L^{c} can be calculated as follows (seen at ④ in Figure 7.2):

$$L^{\mathrm{c}} = \frac{\sum_i^{NHW} \mathbf{M}_{\phi,i}^\gamma \, \mathcal{H}[\rho_{\mathrm{T}}(\mathbf{p}_{\theta,i}), \mathbf{p}_{\phi,i}^{\mathtt{m}}]}{\sum_{j=1}^{NHW} \mathbf{M}_{\phi,j}^\gamma} \tag{7.3}$$

Similar to Section 6.5, the model can reduce the loss by minimising the inconsistency between the segmentations $\mathbf{p}_\theta$ and $\mathbf{p}_\phi^{\mathtt{m}}$, or by expressing uncertainty via $\mathbf{M}_\phi^\gamma$.

An additional difference between this objective and that seen in Section 6.5, is the sharpening function $\rho_{\mathrm{T}}(\cdot)$. This is presented in [156] and, for a vector $a \in \mathbb{R}^K$, is formulated as $\rho_{\mathrm{T}}(a_k) = \frac{a_k^{1/T}}{\sum_j^K a_j^{1/T}}$. This work uses $T = 0.5$, and so ρ reduces the entropy of soft target

$\mathbf{p}_{\theta,i}$. This is motivated for the same reasons as in Section 6.5, that it further increases the separation between certain and uncertain pixels, however in the previous chapter, this was achieved with the asymmetric architecture as opposed to the sharpening function ρ – as such this method is simpler.

7.4.2 Avoiding Feature Collapse

A significant challenge in Chapter 6 was preventing feature collapse from occurring, as detailed in Section 6.5.4, where, despite satisfying the objective, the feature representation was no longer dependent on the input and contained no semantic information. Feature collapse is a possible side-effect when maximising a consistency objective across branches, where in this chapter, these branches are parameterised by $\mathtt{f}_\theta$ and $\mathtt{f}_\phi$. A key benefit of the method in this chapter is that feature collapse cannot occur, because, during uncertainty training, the parameters of $\mathtt{f}_\theta$ are not updated, i.e. $\mathtt{f}_\theta$ is frozen.

This is made possible by the practise of initialising the encoders of $\mathtt{f}_\theta$ and $\mathtt{f}_\phi$ with foundation model parameters. Without this practise, as is true in Chapter 6, the feature representation would be overly task-specific after the task learning step. The problem this causes is that, initially, the segmentation consistency would be very low for target domain images as a result of the initially poor representation of the target domain. This means that, initially, the objective L^c maximises the consistency between very few pixels, and so the representation of the target domain would not greatly improve. If one of the branches is frozen, then this remains true, and as a result, the uncertainty estimation quality does not greatly improve throughout training.

However, in Chapter 6, this is avoided by updating the parameters of both branches, and so, while the consistency is initially low, it increases over the course of training and therefore so does the quality of the representation and of uncertainty estimation. In this chapter, initialising $\mathtt{f}_\theta$ with foundation model parameters circumvents this problem, as even after fine-tuning, the representation is good enough for the consistency to be high enough to further improve the representation of $\mathtt{f}_\phi$ using the target domain images.

7.5 Masked Image Modelling for Uncertainty Estimation

As discussed in the previous section, the uncertainty training method presented in this chapter involves perturbing training images via a masking policy. This section will further motivate this decision, and discusses the specifics of the masking policy used.

7.5.1 Background

This thesis, much like the SSL literature, uses both crop-and-resize and masking tasks in order to perturb the appearance of training images, whilst preserving their semantic content.

'Masked signal modelling' – where a portion of an input signal is masked out, and a model is trained to reconstruct the masked signal – has been a successful approach for NLP for a number of years, by the name of Masked Language Modeling (MLM). The application of this approach to computer vision is more challenging for a number of reasons. Firstly, language is discrete and semantically dense, and so reconstruction is formulated as a classification problem, which requires the sophisticated modelling of semantic information. In contrast, image reconstruction requires predicting continuous pixel values, for which high-frequency local information needs to be modelled. Therefore, successful MIM tasks require a different formulation to MLM tasks, such as seen in iBOT [113], SimMiM [157] and BEiT [158].

Secondly, CNNs previously dominated computer vision and are not suited to MIM tasks, as it is not clear how to encode the image masking task. This is in contrast to masking for transformer architectures, such as ViT [36], where masking tokens and positional encodings allow for the use of a similar formulation to that seen in MLM.

For the cited works in this section, it is specifically semantic information that is learned because the masking policy removes information that can only be reconstructed with a wider semantic understanding of the image, as opposed to low-level high-frequency information. This can be achieved by (1) randomly sampling a high proportion of patches, such as in [112], [157], where it is argued that masking tasks for images are easy due to the redundancy of information, therefore requiring an aggressive masking policy to make the task sufficiently difficult, or (2) removing a smaller proportion of semantically important patches, such that the remaining unmasked patches are less informative of the whole, re-

quiring significant semantic understanding to reconstruct the image, such as in [159]–[161], which use a range of methods to inform the masking policy. Methods such as [113], [158] randomly mask out blocks of multiple patches in a local region, making this masking policy similar to the latter of the two previous strategies, albeit noisier due to its more random and un-informed nature.

7.5.2 Motivation

A particular motivation for masking tasks over crop-and-resize tasks, such as in [32], [106], [162], is that the objective is defined on a patch-wise basis, rather than an image-wise basis. This means that the network is directly trained to encode patch-wise semantics, as opposed to image-wise semantics. This is demonstrated in DINOv2 [31], where an iBOT-based objective is used and is shown to have significantly better fine-tuned semantic segmentation performance over DINOv1 [162] and OpenCLIP [163], both of which use image-wise objectives. This suggests that masking could be a better alternative for estimating, and then learning to detect, segmentation error, instead of adapting crop-and-resize tasks to segmentation as seen in Chapter 6.

An additional benefit of masking tasks is their simplicity. Defining a crop-and-resize task involves defining the size of the smaller crop relative to the larger crop. If this is made too small, then there will not be sufficient context to recognise objects or object parts, and the network is likely to focus on low-level images statistics. If the smaller crop is made too large, then the task will be made too easy, and the network will not be required to learn robust semantic features. There is a similar consideration for masking tasks, based on the scale of masked regions and the proportion of the image that is masked. However, for crop-and-resize tasks, appearance transforms must also be defined – such as augmenting the hue, augmenting the saturation, adding random noise – along with hyperparameters associated with the magnitude of each of these. Therefore, defining a masking task is simpler and involves considerably fewer hyperparameters.

7.5.3 Masking Policy

The aim of the masking policy is such that the segmentations produced by $\mathtt{f}_\theta$ and $\mathtt{f}_\phi \circ \mathtt{m}$ are consistent when the class assignment for that pixel is correct, and they are inconsistent otherwise. A major difference between the masking task in this method and those found in literature is that the output space in which comparisons between masked and unmasked are made is semantic segmentation space, and not an abstract feature space or RGB space.

The masking policy is implemented as $\mathtt{m}(\mathbf{x}) = \mathbf{M} \odot [\mathtt{E}]_{0:1}(\mathbf{x})$, where $[\mathtt{E}]_{0:1} : \mathbb{R}^{3\times H\times W} \rightarrow \mathbb{R}^{F_{\text{pe}}\times \hat{H}\times \hat{W}}$ is the patch embedding of encoder $\mathtt{E}$, $\mathbf{M} \in \{0,1\}^{\hat{H}\times\hat{W}}$ is the patch-wise mask, and $\odot$ is the element-wise product.

Another difference with commonly presented masking tasks is the nature of the data used. In contrast to datasets such as ImageNet, for driving datasets such as Cityscapes there exists a greater range of scale within classes and between classes. For example, a traffic light in the distance and a traffic light up close occupy a very different number of pixels. Additionally the classes: `building`, `road`, `sky` take up much more of images than `pedestrian`, `traffic sign`, `traffic light`, `pole`.

In masking tasks, it is generally important to mask out semantically consistent regions of images, e.g. object parts. The significance of the range of scales is therefore that it is much harder to hand-craft a policy that gets the scale of the masks correct. More concretely, the challenge is the following: either you (1) mask out large regions, and suffer the case that entire semantic objects are masked out, with no context information to infer that they existed, or (2) you mask out small regions, and the masking task is too easy, and the model can solve the task by local information and interpolation. If the task is too easy or too hard then this breaks the assumption that consistency approximates accuracy, and prevents the learning of high-quality uncertainty estimation.

The solution that was found to be the most effective is to mask patches independently, such that for a patch i, the mask is given by:

$$\mathbf{M}_i = \text{Bernoulli}(p_{\text{mask}}) \tag{7.4}$$

Where $p_{\text{mask}} = 0.5$ worked effectively. The fact that patches are masked independently helps

to minimise the likelihood that small semantic instances are entirely masked out, while also providing a suitable level of difficulty.

A number of more complex masking policies were tried in order to try and maximise the effectiveness of the masking task. These included: (a) masking only the most uncertain pixels, in order to focus the task on the image regions that were most difficult to segment (b) learning a masking policy on a pretrained network that maximises the segmentation inconsistency (c) experiments where the loss was only backpropagated w.r.t. the masked or unmasked regions. None of these policies were able to improve upon the performance of the simple mask, and so due to their complexity, they are not proposed as part of the method. The benefit of a simple method is also that it is unlikely to be biased to the datasets considered in the experiments in this chapter.

7.6 Experimental Setup

7.6.1 Network Architecture

The segmentation network architecture used in this work is the Mask2Former [148] architecture. For a description of the details of this architecture, see Figure 7.3. The image size used during training in this method is $(H, W) = (224, 224)$, and the dimensions of the mask features are $(\bar{H}, \bar{W}) = (64, 64)$.

It is worth noting that the query embeddings act in a very similar manner to prototypes in prototype segmentation (described in Section 6.2.1), however instead of being calculated from labelled images, they are initialised and progressively refined by a transformer.

The encoder used with Mask2Former is the DeiT transformer [153], as we want to use the weights from the training of DINOv1 [162] and DINOv2 [31].

7.6.2 Network Initialisation

The primary foundation model used is the ViT-Small from DINOv2 [31], which is distilled from a ViT-Giant foundation model. In order to investigate the importance of these weights, the results are compared to initialising with ViT-Small weights from DINOv1 [162]. The weights from DINOv1 [162] and DINOv2 [31] represent different levels of generality in fea-

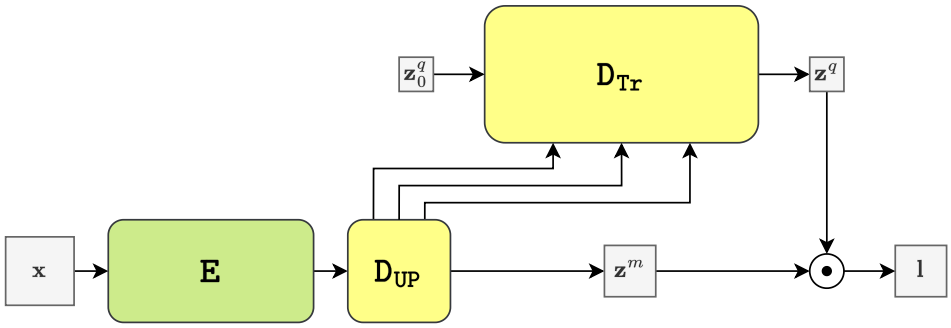


Figure 7.3: Illustration of the Mask2Former segmentation network, $\mathbf{l} = \mathtt{f}(\mathbf{x})$, used in this work. The encoder $\mathtt{E}$ embeds an image $\mathbf{x} \in \mathbb{R}^{3\times H\times W}$, which is then progressively upsampled by a decoder $\mathtt{D_{UP}}$, yielding a series of feature maps with the final one being the mask features $\mathbf{z}^m \in \mathbb{R}^{256\times \bar{H}\times \bar{W}}$. All but the last of these feature maps are used to condition a transformer decoder $\mathtt{D_{Tr}}$ to transform initial query embeddings $\mathbf{z}_0^q \in \mathbb{R}^{K\times 256}$ into query embeddings $\mathbf{z}^q \in \mathbb{R}^{K\times 256}$. These final query embeddings are then dot-producted, illustrated as $\odot$, with mask features $\mathbf{z}^m \in \mathbb{R}^{256\times \bar{H}\times \bar{W}}$ to give the logits $\mathbf{l} \in \mathbb{R}^{K\times \bar{H}\times \bar{W}}$. The logits are then finally bilinearly upsampled to the original image spatial dimensions $\mathbf{l} \in \mathbb{R}^{K\times H\times W}$.

ture representation, with the former being less general than the latter. This is judged on the transfer learning performance found in each paper. This difference is a result of DINOv2 used an improved training procedure on a larger, more diverse dataset.

The DINOv1 model is trained on ImageNet-22k without labels, therefore containing approximately 14 million images, while DINOv2 is trained on a dataset containing 142 million images. The dataset for DINOv2 is comprised of a set of image classification datasets (including ImageNet-22k), segmentation, depth estimation and image retrieval datasets. Additionally, images are sampled from a pool of internet-scraped images based on their similarity with these datasets, thus further increasing the diversity. For example, an additional 57 million images are sampled that bear resemblance to the 14 million images in ImageNet-22k.

7.6.3 Data

The evaluation of models for this method is similar to that of the previous chapter. We use labelled training images to define the source domain, and then unlabelled training images and labelled images from target domains. In this work, Cityscapes is the source domain, and the following target domains are used: SAX London, SAX New Forest, SAX Scotland, BDD (described in Chapter 4).

We also make the point to test a model trained with unlabelled data from a given target domain on each of the other labelled test datasets. This allows us to examine how uncer-

tainty training in one domain generalises to other target domains. We also again test on WildDash [134], which is a domain that contains no unlabelled training images, and is itself very diverse.

We also perform testing on the validation set in Cityscapes in order to compare how uncertainty estimation performance in the source domain compares to that of the target domains.

7.6.4 Baselines

In order to evaluate the performance of the method proposed in this chapter, we consider a number of baselines. Unlike the baselines in Chapter 6, each of the baselines in this method are either initialised with DINOv1 or DINOv2 weights, as will be described in their model name. This makes for fair comparison to our proposed method, and provides an interesting contrast with the previous chapter's baseline, as it tells us the effect of general representations on a range of different methods. As before, we consider both epistemic uncertainty estimation techniques and OoD detection-based representation methods.

For the former, we train both MCD networks and sets of ensembles. It is again worth noting that these methods are computationally heavy compared with methods such as ours and representation-based methods.

As for the representation-based methods, we consider methods that are trained in a supervised manner on the source dataset, and then define a specific inference procedure in order to estimate uncertainty. The inference procedures considered are (1) calculating the max softmax score, named as `MaxS-xxx`, and (2) and calculating Mahalanobis distance between an extracted target pixel feature and the mean class-wise source domain features, names as `GMM-xxx`. These representation baselines allow us to examine how the proposed uncertainty training provides an improved representation than either full supervised fine-tuning of encoder and decoder, or leveraging the general encoder with supervised trained decoder.

7.7 Experiments and Results

This section firstly describes the experiments performed in this chapter. Subsequently, the quantitative results are discussed and shown in Table 7.2, Table 7.1 and Figure 7.4. Additional qualitative results can be found in Figure 7.5.

Data: We train models with different unlabelled target datasets, and evaluate which datasets worked best for learning uncertainty estimation for each test domain. The dataset used can be found in the model name, e.g. `xxx-LDN-xxx` is trained with unlabelled SAX London images.

Initialisation: The models trained are either initialised with DINOv1 or DINOv2 weights for the encoder. This allows us to investigate how impactful a more general representation is to uncertainty estimation performance. This information can be seen in the model names: `xxx-xxx-d2` uses DINOv2 weights and `xxx-xxx-d1` uses DINOv1.

Input Augmentation: The method in this chapter uses a masking task instead of a crop-and-resize task to calculate segmentation consistency, therefore we investigate the benefits of each. The models trained with a masking task are named `Mask-xxx-xxx`, while the models trained with crop-and-resize are named `C&R-xxx-xxx`.

Freezing $\mathtt{f}_\theta$: We investigate the effect of not freezing $\mathtt{f}_\theta$, as this a key difference between this method and that in Chapter 6. The equivalent branch to $\mathtt{f}_\theta$ in the method in Chapter 6 was not frozen, but updated along with the branch equivalent to $\mathtt{f}_\phi$, as discussed in Section 7.4.2. If $\mathtt{f}_\theta$ is frozen, this can been seen in the method name as `xxx-`$\mathtt{f}_\theta^+$.

Freezing `E`: As described in the framework in Section 7.3, in order to maximise the segmentation quality on the source domain, the encoder `E` is fine-tuned. This reduces the amount of task-agnostic information in the encoder's representation, and therefore we hypothesized in Section 7.2 that this would reduce the quality of uncertainty estimation. We investigate this by performing experiments where the encoder is frozen during training on the source domain. Results for this are found in the method names as `xxx-`$\mathtt{E}^+$.

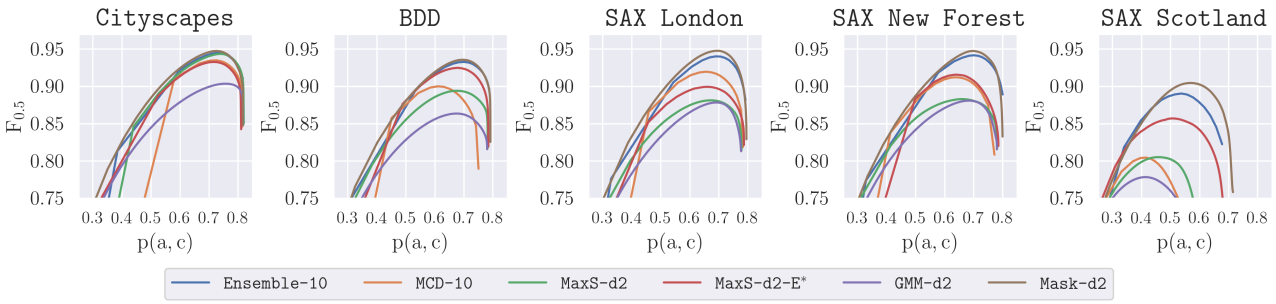


Figure 7.4: In these plots, we measure misclassification detection performance using $F_{0.5}$ scores plotted against the proportion of pixels that are certain and accurate $p(a, c)$. The baselines are trained only with labelled Cityscapes data, while our proposed model, `Mask-d2`, leverages unlabelled images from the domain in which testing is occurring. All models are able to perform uncertainty estimation similarly well for Cityscapes, however when tested on the distributionally shifted target domains, `Mask-d2`'s performance exceeds that of the baselines. The gap in $\text{MaxF}_{0.5}$ score between `Mask-d2` and `MaxS-d2`, `MaxS-d2-E*` is descriptive of the benefit of our proposed uncertainty training.

7.7.1 Different Target Domains

Firstly, it can be seen in Figure 7.4 that the proposed `Mask-d2` models outperform the baselines in terms of $\text{MaxF}_{1/2}$. In Table 7.1, we can see that the model with highest $\text{MaxF}_{1/2}$ score on each test dataset is the `Mask-d2` model that is trained with unlabelled images from same domain as the test dataset. This is also very nearly true for the AUPR metric, shown in Table 7.2. The differences between the `Mask-d2` models trained on different unlabelled domains, however, are not that large, and it is common for a different `Mask-d2` model to outperform each of the baselines, showing that the proposed models generalise well to performing uncertainty estimation in different domains.

This is shown particularly well by looking at the performance of the `Mask-d2` models on the WildDash dataset for which no unlabelled images are available. The best performing model was `Mask-d2-BDD`, but each of the other models also performed well compared with the baselines. The reason for the BDD dataset outperforming the rest is perhaps due to its greater diversity, due to its collection in a wider range of conditions and wider range of locations. It is nonetheless true, that for each domain, using domain-specific training unlabelled images is the best performing strategy, rather than opting for a more general dataset such as BDD.

		$\mathrm{MaxF}_{1/2}$ @ $\mathrm{p(a, c)}$					
	Method	CS	LDN	NF	SCOT	BDD	WD
Baselines	Ensemble-5-d2	0.944 @ 0.721	0.939 @ 0.699	0.94 @ 0.697	0.894 @ 0.552	0.932 @ 0.695	0.912 @ 0.624
	Ensemble-10-d2	0.945 @ 0.727	0.94 @ 0.689	0.942 @ 0.701	0.891 @ 0.537	0.933 @ 0.7	0.913 @ 0.626
	MCD-5-d2	0.935 @ 0.726	0.919 @ 0.656	0.912 @ 0.64	0.803 @ 0.416	0.926 @ 0.679	0.903 @ 0.599
	MCD-10-d2	0.935 @ 0.726	0.92 @ 0.656	0.912 @ 0.639	0.805 @ 0.415	0.926 @ 0.679	0.903 @ 0.599
	MaxS-d2-E$^+$	0.933 @ 0.714	0.899 @ 0.658	0.916 @ 0.641	0.857 @ 0.507	0.925 @ 0.678	0.908 @ 0.609
	MaxS-d2	0.944 @ 0.739	0.881 @ 0.674	0.883 @ 0.659	0.805 @ 0.458	0.894 @ 0.674	0.87 @ 0.603
	GMM-d2	0.904 @ 0.757	0.878 @ 0.692	0.881 @ 0.683	0.778 @ 0.415	0.864 @ 0.674	0.805 @ 0.567
	MaxS-d1-E$^+$	0.912 @ 0.67	0.859 @ 0.579	0.899 @ 0.612	0.82 @ 0.359	0.886 @ 0.586	0.86 @ 0.483
	MaxS-d1	0.936 @ 0.716	0.858 @ 0.61	0.887 @ 0.628	0.79 @ 0.343	0.896 @ 0.622	0.853 @ 0.497
	GMM-d1	0.894 @ 0.751	0.789 @ 0.639	0.823 @ 0.61	0.687 @ 0.281	0.827 @ 0.656	0.743 @ 0.572
	C&R-NF-d2	0.928 @ 0.703	0.911 @ 0.641	0.925 @ 0.662	0.869 @ 0.518	0.912 @ 0.669	0.891 @ 0.61
	C&R-NF-d2-f$^+_\theta$	0.931 @ 0.69	0.894 @ 0.658	0.902 @ 0.66	0.863 @ 0.524	0.923 @ 0.676	0.908 @ 0.618
	C&R-NF-d1	0.917 @ 0.666	0.867 @ 0.567	0.901 @ 0.607	0.825 @ 0.364	0.898 @ 0.618	0.874 @ 0.522
	C&R-NF-d1-f$^+_\theta$	0.919 @ 0.67	0.894 @ 0.599	0.907 @ 0.612	0.752 @ 0.338	0.891 @ 0.622	0.852 @ 0.5
Ours	Mask-LDN-d2	0.945 @ 0.735	**0.948** @ 0.693	**0.948** @ 0.697	0.889 @ 0.481	0.934 @ 0.694	0.913 @ 0.605
	Mask-NF-d2	**0.947** @ 0.724	0.941 @ 0.679	**0.948** @ 0.696	0.903 @ 0.517	0.927 @ 0.677	0.902 @ 0.582
	Mask-SCOT-d2	0.934 @ 0.706	0.924 @ 0.645	0.939 @ 0.693	**0.905** @ 0.568	0.92 @ 0.662	0.899 @ 0.579
	Mask-BDD-d2	0.938 @ 0.725	0.931 @ 0.675	0.942 @ 0.698	0.891 @ 0.501	**0.936** @ 0.696	**0.918** @ 0.631
	Mask-LDN-d1	0.931 @ 0.693	0.9 @ 0.623	0.919 @ 0.641	0.848 @ 0.361	0.91 @ 0.63	0.879 @ 0.508
	Mask-SCOT-d1	0.926 @ 0.679	0.881 @ 0.599	0.919 @ 0.636	0.851 @ 0.397	0.907 @ 0.624	0.87 @ 0.502

Table 7.1: Misclassification Detection performance described by $\mathrm{MaxF}_{1/2}$ @ $\mathrm{p(a, c)}$ for a range of test domains.

		AUPR					
	Method	CS	LDN	NF	SCOT	BDD	WD
Baselines	Ensemble-5-d2	0.98	0.976	0.979	0.933	0.967	0.948
	Ensemble-10-d2	0.981	0.976	0.978	0.932	0.967	0.946
	MCD-5-d2	0.977	0.971	0.963	0.778	0.971	0.959
	MCD-10-d2	0.977	0.971	0.963	0.786	0.971	0.959
	MaxS-d2-E$^+$	0.975	0.958	0.968	0.931	0.97	0.961
	MaxS-d2	0.982	0.933	0.942	0.861	0.94	0.919
	GMM-d2	0.936	0.894	0.914	0.838	0.896	0.845
	MaxS-d1-E$^+$	0.966	0.92	0.953	0.889	0.942	0.925
	MaxS-d1	0.98	0.913	0.942	0.859	0.949	0.918
	GMM-d1	0.921	0.781	0.866	0.741	0.835	0.739
	C&R-NF-d2	0.972	0.967	0.975	0.936	0.954	0.937
	C&R-NF-d2-f$^+_\theta$	0.978	0.927	0.932	0.92	0.965	0.96
	C&R-NF-d1	0.972	0.929	0.954	0.883	0.951	0.935
	C&R-NF-d1-f$^+_\theta$	0.972	0.949	0.957	0.794	0.941	0.915
Ours	Mask-LDN-d2	0.985	**0.987**	**0.985**	0.947	**0.973**	0.966
	Mask-NF-d2	**0.988**	0.984	**0.985**	0.958	0.964	0.958
	Mask-SCOT-d2	0.977	0.975	0.98	**0.96**	0.965	0.957
	Mask-BDD-d2	0.98	0.977	0.981	0.952	0.972	**0.967**
	Mask-LDN-d1	0.979	0.956	0.969	0.91	0.957	0.937
	Mask-SCOT-d1	0.977	0.936	0.966	0.92	0.959	0.933

Table 7.2: Misclassification Detection performance summarised over all possible thresholds described by AUPR for a range of test domains.

7.7.2 Freezing E

Table 7.1 and Table 7.2 show that by fine-tuning the encoder, the uncertainty quality is improved for the distributionally shifted test domains. Both the $\mathrm{MaxF}_{1/2}$ and AUPR metrics in these tables are higher for MaxS-d2-E$^+$ than for MaxS-d2. This confirms our hypothesis, and is explained by the fact that the fine-tuning removes task-agnostic information from the

representation.

When considering the less general representation of DINOv1, this effect is similar, but much less pronounced, i.e. there is only a very small benefit to uncertainty estimation in not fine-tuning the DINOv1 representation. In fact, the fully fine-tuned model performs slightly better for the BDD test dataset. This observation can be seen by comparing `MaxS-d1-E+` and `MaxS-d1` in Table 7.1 and Table 7.2, and can be explained by the fact that there was significantly less task-agnostic information in the DINOv1 representation to begin with. This means that there is less of this information to remove by fine-tuning, and therefore less of a drop in quality of uncertainty estimation.

Comparing `Mask-d2` to `MaxS-d2-E+` in Table 7.1 and Table 7.2 shows us that the representation learned with our uncertainty training has improved upon the general pretraining by learning task-specific attributes with are important to uncertainty estimation, while also not over-fitting to the training data. This is equally true when comparing `Mask-d1` to `MaxS-d1-E+`.

7.7.3 Comparing perturbation methods

It is of interest to us to consider how the performance of the `C&R` approaches compare to that of the `Mask` approaches. We show in Table 7.1 and Table 7.2, the performance of the `Mask` approaches are generally better than that of `C&R` approaches, but there is an additional benefit which is important to highlight, relating to the discussion in Section 7.5.2.

Using `C&R` clearly requires designing both a cropping and colour-space augmentation scheme, and it is important that, particularly the latter, is tuned in order to be optimal for the data used. We show that this is however less true for our masking task, which only has one hyperparameter, which it is quite robust to.

We trained two models for each of `Mask` and `C&R` each with different hyperparameters on the SAX New Forest domain. For the former, we use masking policies with $p_{\text{mask}} = 0.25$ and $p_{\text{mask}} = 0.75$. For the latter, we train one model where less colour-space augmentation, and one with a cropping policy with different hyperparameters.

For the masking task with $p_{\text{mask}} = 0.25$ and $p_{\text{mask}} = 0.75$, the $\text{MaxF}_{1/2}$ @ $\text{p}(\text{a}, \text{c})$ scores were 0.945 @ 0.705 and 0.951 @ 0.713 respectively. For the `C&R` task, the $\text{MaxF}_{1/2}$ @ $\text{p}(\text{a}, \text{c})$ scores

were 0.926 @ 0.655 and 0.891 @ 0.648 respectively.

We can see that the scores for the C&R task varies considerably more than that of the masking task. For this reason, it is much simpler and easier to use a masking task to obtain high-quality uncertainty estimates.

7.7.4 Freezing $\mathtt{f}_\theta$

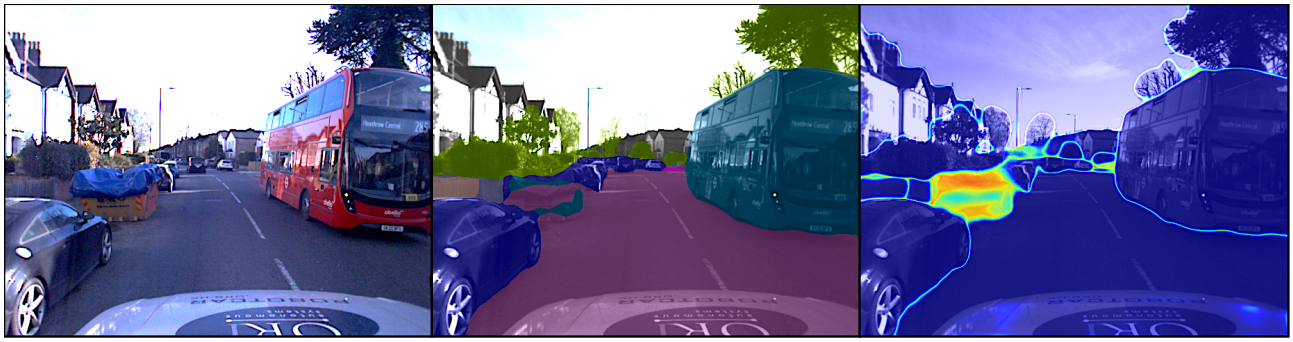
We would like to determine the effects of freezing $\mathtt{f}_\theta$ as it prevents a significant training failure mode (as discussed in Section 7.4.2), but it is possible that it could impact quality of learned uncertainty estimation. As per our previous chapter, we test this on the C&R task, and do so in the SAX New Forest domain. These models can be found in Table 7.1 and Table 7.2 as C&R-NF-d1-$\mathtt{f}_\theta^+$ and C&R-NF-d2-$\mathtt{f}_\theta^+$.

When training with DINOv1 weights, we see that there is a improvement in uncertainty estimation quality when the $\mathtt{f}_\theta$ is not frozen. This benefit is however not seen when using DINOv2. This means that when the initialisation is sufficiently general, we can simplify the training procedure greatly by freezing $\mathtt{f}_\theta$ with no cost in performance.

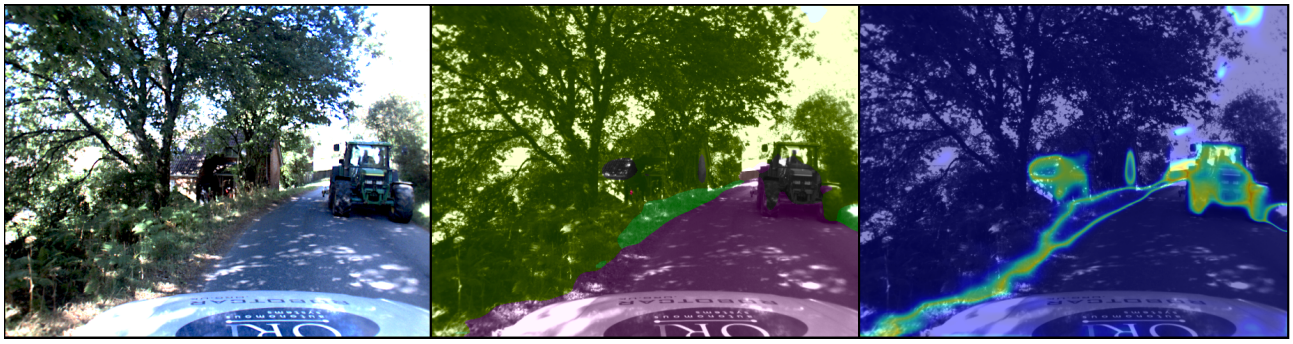
7.8 Conclusion

This final method has used a combination of large-scale training and driving-domain-specific uncertainty training to learn distribution uncertainty estimation. It is therefore a combination of the ideas presented in Chapter 5 and Chapter 6. Instead of learning a large-scale OoD detection problem like in Chapter 5, this work used a model that is distilled from a network trained with more typical large-scale self-supervised learning. This chapter has shown that these approaches are complementary and lead to very high quality distributional uncertainty estimation.

In addition to the changes in architecture and network initialisation, this chapter also advocates for the use of masking instead of crop-and-resize data augmentation. It suggests that when using an encoder with a transformer architecture, masking may well be a more convenient choice due to decreased reliance on tuned hyperparameters.



(a) SAX London



(b) SAX New Forest



(c) SAX New Forest

Figure 7.5: Qualitative results for the proposed `Mask-d2` model, presenting (left) an RGB image, (middle) the semantic segmentation and (right) the estimated uncertainty in the `jet` colour map, where red is uncertain and blue is certain. For each distributionally shifted image, the incorrect segmentations are effectively detected by the model's estimated uncertainty. The variant of `Mask-d2` model used is that which was trained on the same domains as the test image shown (test domain described in sub-caption).

Chapter 8

Conclusion

Contents

8.1 Summary 171

8.2 Closing Remarks 174

8.1 Summary

It is vitally important that safety and trust are considered in the deployment of robotic systems, and thus should be a key motivator for their system design. As such, the objective of this thesis has been to tackle the problem of distributional shift, where data unlike that seen in model training causes an increase in error rate and potentially dangerous deployment scenarios.

With a specific focus on the task of semantic segmentation, this thesis has investigated solutions to the problem of distributional shift that involve the detection of test error, i.e. uncertainty estimation. This is an orthogonal avenue of mitigating test error to that of reducing it, and this thesis argues that both are important steps in the effort to ensure safety and trust when using deep learning systems for safety-critical applications.

For mobile robotics applications, this thesis has concluded that, in order to reduce the computational cost at inference time, uncertainty estimates should ideally be output by a segmentation network alongside its estimated semantic segmentations. In light of the discussion of model miscalibration in Chapter 2, the literature regarding uncertainty estimation

and OoD detection was investigated and discussed in Chapter 3, with the aim to identify methods that provide insight for the training of a segmentation network to output high-quality uncertainty estimates in a single forward pass.

A conclusion from Chapter 3 was that this thesis should focus on methods that learn uncertainty estimation from training data that is distributionally-shifted from the labelled segmentation training dataset. This is key, because the representation learned from supervised semantic segmentation training is specific to the task and the domain defined by the labelled dataset used, which makes uncertainty estimation challenging on a shifted domain. For this reason, distributional uncertainty estimation can be improved by using additional training data to learn a representation that can better detect the relevant semantic differences between the labelled segmentation dataset and a given test image.

Following Chapter 3 is a search for the answers to the following questions: (a) What data should be used to tailor a segmentation network to perform high-quality pixel-wise uncertainty estimation on distributionally-shifted test data? (b) What does a training procedure look like that allows us to train on this data for this purpose? In answering these questions, this thesis has presented three methods that learn distributional uncertainty estimation from a distributionally-shifted training dataset.

The first method, presented in Chapter 5, trains a segmentation network to perform large-scale OoD detection by training on a large-scale image recognition dataset, in addition to semantic segmentation. Given the availability of very large and diverse datasets such as ImageNet [97] or LAION-5B [135] which are comprised almost entirely of natural images from outside of the domain of the labelled segmentation training dataset, it is possible to expose a segmentation network to a diverse set of OoD images. This method defines an OoD detection task based on this to train a network to learn a separable representation of the two types of training data: in-distribution labelled driving images from a given domain, and unlabelled images from a diverse natural image dataset. The key contributions of this method are as follows:

- A data augmentation procedure that combines in-distribution and OoD images on a pixel-wise basis and effectively reduces the appearance difference between them. This results in an pixel-wise OoD detection task that is sufficiently challenging to train a

segmentation network to robustly perform distributional uncertainty estimation.

- A contrastive loss function that trains a segmentation network to learn a representation in which in-distribution and OoD instances are separable, whilst accounting for the label noise induced by using large-scale unlabelled image datasets.

Our primary insight from this chapter was as follows: the smaller the difference between distributionally-shifted training images and in-distribution training images, the higher the quality of the learned uncertainty estimation.

The second method, presented in Chapter 6, draws on this insight and uses distributionally-shifted *driving domain* images during training. This is because these images are inherently more similar to the in-distribution labelled driving images than images from a large-scale dataset, and can therefore lead to more robust learned distributional uncertainty estimation. The contributions of this method are as follows:

- A training task based on crop-and-resize data augmentation to detect regions of likely segmentation error.

- An inference procedure to output uncertainty as a feature space distance, coupled with a loss function which improves the segmentation network's representation for distributional uncertainty estimation.

This method is evaluated with an extensive set of experiments to measure the quality of the proposed model's uncertainty estimates, and to ablate the method to investigate how its components affect the empirical performance.

The final method, presented in Chapter 7, makes use of both the distributionally-shifted driving domain images from the previous chapter and diverse natural images from a unlabelled large-scale dataset. In doing so, it combines the benefits of the previous two methods. This method's contributions are as follows:

- A proposed framework for training a segmentation network, involving three steps. The first step trains an encoder using a large diverse image dataset in a self-supervised manner, yielding a foundation model. Secondly, this encoder is used to initialise the segmentation network, which is fine-tuned to perform semantic segmentation on a

given domain. Finally, a new segmentation network, also initialised with the encoder from the first step, is trained to detect when error occurs on unlabelled distributionally-shifted driving images. This final step fine-tunes the model for both distributional uncertainty estimation and semantic segmentation, and therefore maintains more of the task-agnostic information in the encoder's representation than is the case for standard fine-tuning.

- A method for the final step that involves a masking task, which is simpler and more effective than using crop-and-resize data augmentation.

8.2 Closing Remarks

In the broadest terms, this thesis seeks to design deep learning systems for which safety and trust are of paramount importance. This is a significant challenge, and one which ought to be central to the field of robotics in the current era. It is also a challenge that can benefit greatly from the increasing efforts into training and open-sourcing large-scale computer vision models, as discussed in Chapter 7. At the nexus of foundation models, model distillation and uncertainty training, we believe there is great promise for the further reduction in error rate for robotic perception systems.

Bibliography

[1] *Assuring the operational safety of automated vehicles – Specification*. Standard. British Standards Institution, 2022.

[2] Wayve Ltd. *Driving computer vision with deep learning*. 2018. URL: `https://wayve.ai/thinking/driving-computer-vision-with-deep-learning/`.

[3] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, et al. "End-to-end learning of geometry and context for deep stereo regression". In: *Proceedings of the IEEE international conference on computer vision*. 2017.

[4] Jyh-Jing Hwang, Henrik Kretzschmar, Joshua Manela, et al. "Cramnet: Camera-radar fusion with ray-constrained cross-attention for robust 3d object detection". In: *European Conference on Computer Vision*. Springer. 2022, pp. 388–405.

[5] Jiankun Wang, Tianyi Zhang, Nachuan Ma, et al. "A survey of learning-based robot motion planning". In: *IET Cyber-Systems and Robotics* (2021).

[6] Moataz Samir, Chadi Assi, Sanaa Sharafeddine, et al. "Age of information aware trajectory planning of UAVs in intelligent transportation systems: A deep learning approach". In: *IEEE Transactions on Vehicular Technology* (2020).

[7] Peide Cai, Yuxiang Sun, Yuying Chen, et al. "Vision-Based Trajectory Planning via Imitation Learning for Autonomous Vehicles". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019.

[8] Siddhant Gangapurwala, Mathieu Geisert, Romeo Orsolino, et al. "RLOC: Terrain-Aware Legged Locomotion Using Reinforcement Learning and Optimal Control". In: *IEEE Transactions on Robotics* (2022).

[9] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, et al. “Learning dexterous in-hand manipulation”. In: *The International Journal of Robotics Research* (2020).

[10] Sumedh A Sontakke, Jesse Zhang, Sébastien MR Arnold, et al. “RoboCLIP: One Demonstration is Enough to Learn Robot Policies”. In: *arXiv preprint arXiv:2310.07899* (2023).

[11] Alex Kendall, Jeffrey Hawke, David Janz, et al. “Learning to drive in a day”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019.

[12] Anthony Brohan, Noah Brown, Justice Carbajal, et al. “RT-1: Robotics Transformer for Real-World Control at Scale”. In: *arXiv preprint arXiv:2212.06817*. 2022.

[13] Sergey Levine, Chelsea Finn, Trevor Darrell, et al. “End-to-end training of deep visuomotor policies”. In: *The Journal of Machine Learning Research* (2016).

[14] Dan Barnes, Rob Weston, and Ingmar Posner. “Masking by Moving: Learning Distraction-Free Radar Odometry from Pose Information”. In: *Conference on Robot Learning (CoRL)*. 2019.

[15] Georgi Pramatarov, Daniele De Martini, Matthew Gadd, et al. “BoxGraph: Semantic place recognition and pose estimation from 3D LiDAR”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 7004–7011.

[16] Ping-Rong Chen, Shao-Yuan Lo, Hsueh-Ming Hang, et al. “Efficient road lane marking detection with deep learning”. In: *IEEE 23rd International Conference on Digital Signal Processing (DSP)*. 2018.

[17] David Williams, Daniele De Martini, Matthew Gadd, et al. “Keep off the grass: Permissible driving routes from radar with weak audio supervision”. In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE.

[18] Saturnino Maldonado-Bascon, Sergio Lafuente-Arroyo, Pedro Gil-Jimenez, et al. “Road-Sign Detection and Recognition Based on Support Vector Machines”. In: *IEEE Transactions on Intelligent Transportation Systems* (2007).

[19] Felix Stache, Jonas Westheider, Federico Magistri, et al. “Adaptive Path Planning for UAV-based Multi-Resolution Semantic Segmentation”. In: *2021 European Conference on Mobile Robots (ECMR)*. 2021.

[20] Minjie Hua, Yibing Nan, and Shiguo Lian. "Small obstacle avoidance based on RGB-D semantic segmentation". In: *Proceedings of the IEEE/CVF international conference on computer vision workshops*. 2019.

[21] Jungseok Hong, Karin de Langis, Cole Wyethv, et al. "Semantically-aware strategies for stereo-visual robotic obstacle avoidance". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 2450–2456.

[22] Yunong Tian, Guodong Yang, Zhe Wang, et al. "Apple detection during different growth stages in orchards using the improved YOLO-V3 model". In: *Computers and Electronics in Agriculture* (2019).

[23] Eleftherios Lygouras, Nicholas Santavas, Anastasios Taitzoglou, et al. "Unsupervised human detection with an embedded vision system on a fully autonomous UAV for search and rescue operations". In: *Sensors* (2019).

[24] Richard Bormann, Florian Weisshardt, Georg Arbeiter, et al. "Autonomous dirt detection for cleaning in office environments". In: *IEEE international conference on robotics and automation*. 2013.

[25] Marius Cordts, Mohamed Omran, Sebastian Ramos, et al. "The cityscapes dataset for semantic urban scene understanding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[26] Jun-Yan Zhu, Taesung Park, Phillip Isola, et al. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017).

[27] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, et al. "Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.

[28] Yaroslav Ganin and Victor Lempitsky. "Unsupervised domain adaptation by backpropagation". In: *International Conference on Machine Learning*. PMLR. 2015.

[29] Mohammad Sabokrou, Mohammad Khalooei, Mahmood Fathy, et al. "Adversarially learned one-class classifier for novelty detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[30] Alec Radford, Jong Wook Kim, Chris Hallacy, et al. “Learning transferable visual models from natural language supervision”. In: *International Conference on Machine Learning*. 2021.

[31] Maxime Oquab, Timothée Darcet, Theo Moutakanni, et al. *DINOv2: Learning Robust Visual Features without Supervision*. 2023.

[32] Ting Chen, Simon Kornblith, Mohammad Norouzi, et al. “A simple framework for contrastive learning of visual representations”. In: *International Conference on Machine Learning*. 2020.

[33] J. Long, E. Shelhamer, and T. Darrell. “Fully convolutional networks for semantic segmentation”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3431–3440.

[34] Kaiming He, Georgia Gkioxari, Piotr Dollár, et al. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[36] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *ICLR* (2021).

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. “Attention is all you need”. In: *Advances in neural information processing systems* (2017).

[38] Enze Xie, Wenhai Wang, Zhiding Yu, et al. “SegFormer: Simple and efficient design for semantic segmentation with transformers”. In: *Advances in Neural Information Processing Systems* (2021).

[39] Tete Xiao, Yingcheng Liu, Bolei Zhou, et al. “Unified perceptual parsing for scene understanding”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018.

[40] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, et al. “Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021.

[41] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* (2014).

[42] Yuichi Yoshida and Takeru Miyato. “Spectral norm regularization for improving the generalizability of deep learning”. In: *arXiv preprint arXiv:1705.10941* (2017).

[43] Chuan Guo, Geoff Pleiss, Yu Sun, et al. “On Calibration of Modern Neural Networks”. In: *Proceedings of the 34th International Conference on Machine Learning*. 2017.

[44] Matthias Minderer, Josip Djolonga, Rob Romijnders, et al. “Revisiting the calibration of modern neural networks”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 15682–15694.

[45] Yaniv Ovadia, Emily Fertig, Jie Ren, et al. “Can you trust your model’s uncertainty? Evaluating predictive uncertainty under dataset shift”. In: *Advances in Neural Information Processing Systems*. 2019.

[46] Dan Hendrycks, Norman Mu, Ekin D Cubuk, et al. “Augmix: A simple data processing method to improve robustness and uncertainty”. In: *arXiv preprint arXiv:1912.02781* (2019).

[47] Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, et al. “On mixup training: Improved calibration and predictive uncertainty for deep neural networks”. In: *Advances in Neural Information Processing Systems* (2019).

[48] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. “When does label smoothing help?” In: *Advances in Neural Information Processing Systems*. 2019, pp. 4694–4703.

[49] Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, et al. “Calibrating deep neural networks using focal loss”. In: *Advances in Neural Information Processing Systems* (2020).

[50] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, et al. “Mlp-mixer: An all-mlp architecture for vision”. In: *Advances in neural information processing systems* 34 (2021), pp. 24261–24272.

[51] Andrey Malinin and Mark John Francis Gales. “Predictive Uncertainty Estimation via Prior Networks”. In: *2018 Conference on Neural Information Processing Systems (NIPS)*. 2018.

[52] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, et al. “Rethinking the inception architecture for computer vision”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[53] Tsung-Yi Lin, Priya Goyal, Ross Girshick, et al. “Focal Loss for Dense Object Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[54] M. Hein, M. Andriushchenko, and J. Bitterwolf. “Why ReLU Networks Yield High-Confidence Predictions Far Away From the Training Data and How to Mitigate the Problem”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[55] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. 2015.

[56] Leda Cosmides and John Tooby. “Are humans good intuitive statisticians after all? Rethinking some conclusions from the literature on judgment under uncertainty”. In: *Cognition* (1996).

[57] Hidetoshi Shimodaira. “Improving predictive inference under covariate shift by weighting the log-likelihood function”. In: *Journal of statistical planning and inference* (2000).

[58] Arthur Gretton, Alex Smola, Jiayuan Huang, et al. “Covariate shift by kernel mean matching”. In: *Dataset shift in machine learning* (2009).

[59] Ashkan Rezaei, Anqi Liu, Omid Memarrast, et al. “Robust fairness under covariate shift”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2021.

[60] Alexander Kirillov, Eric Mintun, Nikhila Ravi, et al. “Segment anything”. In: *arXiv preprint arXiv:2304.02643* (2023).

[61] Mingxing Tan and Quoc Le. “Efficientnetv2: Smaller models and faster training”. In: *International conference on machine learning*. 2021.

[62] Lu Yuan, Dongdong Chen, Yi-Ling Chen, et al. “Florence: A new foundation model for computer vision”. In: *arXiv preprint arXiv:2111.11432* (2021).

[63] Fisher Yu, Wenqi Xian, Yingying Chen, et al. “BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling”. In: *arXiv preprint arXiv:1805.04687* (2018).

[64] Yarin Gal and Zoubin Ghahramani. “Dropout as a bayesian approximation: Representing model uncertainty in deep learning”. In: *international conference on machine learning*. 2016, pp. 1050–1059.

[65] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, et al. “Weight uncertainty in neural network”. In: *International conference on machine learning*. PMLR. 2015, pp. 1613–1622.

[66] Yarin Gal. “Uncertainty in Deep Learning”. PhD thesis. University of Cambridge, 2016.

[67] Yeming Wen, Paul Vicol, Jimmy Ba, et al. “Flipout: Efficient Pseudo-Independent Weight Perturbations on Mini-Batches”. In: *6th International Conference on Learning Representations, (ICLR)*. 2018.

[68] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. “Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, et al. Curran Associates, Inc., 2017, pp. 6402–6413.

[69] Corina Gurau, Alex Bewley, and Ingmar Posner. “Dropout Distillation for Efficiently Estimating Model Confidence”. In: *ArXiv* abs/1809.10562 (2018).

[70] Andrey Malinin, Bruno Mlodozeniec, and Mark Gales. “Ensemble Distribution Distillation”. In: *International Conference on Learning Representations*. 2020.

[71] Alex Kendall and Yarin Gal. “What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?” In: *Conference on Neural Information Processing Systems (NIPS)*. 2017.

[72] D.A. Nix and A.S. Weigend. “Estimating the mean and variance of the target probability distribution”. In: *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN’94)*. 1994.

[73] Rob Weston, Sarah H. Cen, Paul Newman, et al. “Probably Unknown: Deep Inverse Sensor Modelling Radar”. In: *2019 International Conference on Robotics and Automation (ICRA)* (2019), pp. 5446–5452.

[74] Gwangbin Bae, Ignas Budvytis, and Roberto Cipolla. “Estimating and exploiting the aleatoric uncertainty in surface normal estimation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021.

[75] Ryutaro Tanno, Daniel E Worrall, Aurobrata Ghosh, et al. “Bayesian image quality transfer with CNNs: exploring uncertainty in dMRI super-resolution”. In: *Medical Image Computing and Computer Assisted Intervention (MICCAI)*. 2017.

[76] David Novotny, Diane Larlus, and Andrea Vedaldi. “Learning 3d object categories by looking around them”. In: *Proceedings of the IEEE international conference on computer vision*. 2017.

[77] David Novotny, Samuel Albanie, Diane Larlus, et al. “Self-supervised learning of geometrically stable features through probabilistic introspection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[78] Jochen Gast and Stefan Roth. “Lightweight Probabilistic Deep Networks”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 3369–3378.

[79] Jiawei Liu, Jing Zhang, and Nick Barnes. “Modeling Aleatoric Uncertainty for Camouflaged Object Detection”. In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2022.

[80] Terrance DeVries and Graham W Taylor. “Leveraging uncertainty estimates for predicting segmentation quality”. In: *arXiv preprint arXiv:1807.00502* (2018).

[81] Robert Robinson, Ozan Oktay, Wenjia Bai, et al. “Real-time prediction of segmentation quality”. In: *Medical Image Computing and Computer Assisted Intervention (MICCAI)*. 2018.

[82] Simon Kohl, Bernardino Romera-Paredes, Clemens Meyer, et al. “A probabilistic u-net for segmentation of ambiguous images”. In: *Advances in neural information processing systems* (2018).

[83] Simon AA Kohl, Bernardino Romera-Paredes, Klaus H Maier-Hein, et al. “A hierarchical probabilistic u-net for modeling multi-scale ambiguities”. In: *arXiv preprint arXiv:1905.13077* (2019).

[84] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, et al. "Toward multimodal image-to-image translation". In: *Advances in neural information processing systems* 30 (2017).

[85] Miguel Monteiro, Loıc Le Folgoc, Daniel Coelho de Castro, et al. "Stochastic segmentation networks: Modelling spatially correlated aleatoric uncertainty". In: *Advances in neural information processing systems* (2020).

[86] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[87] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional networks for biomedical image segmentation". In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. 2015.

[88] Guotai Wang, Wenqi Li, Michael Aertsen, et al. "Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks". In: *Neurocomputing* (2019).

[89] Murat Seckin Ayhan and Philipp Berens. "Test-time Data Augmentation for Estimation of Heteroscedastic Aleatoric Uncertainty in Deep Neural Networks". In: *Medical Imaging with Deep Learning*. 2018.

[90] Samuel G. Armato III, Geoffrey McLennan, Luc Bidaut, et al. "The Lung Image Database Consortium (LIDC) and Image Database Resource Initiative (IDRI): A Completed Reference Database of Lung Nodules on CT Scans". In: *Medical Physics* (2011).

[91] Dan Hendrycks and Kevin Gimpel. "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks". In: *International Conference on Learning Representations*. 2017.

[92] Shiyu Liang, Yixuan Li, and R. Srikant. "Enhancing the reliability of out-of-distribution image detection in neural networks". In: *International Conference on Learning Representations*. 2018.

[93] Kimin Lee, Kibok Lee, Honglak Lee, et al. "A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks". In: *Advances in Neural Information Processing Systems*. 2018.

[94] Haoqi Wang, Zhizhong Li, Litong Feng, et al. “ViM: Out-Of-Distribution with Virtual-logit Matching”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.

[95] Grant Van Horn, Oisin Mac Aodha, Yang Song, et al. “The inaturalist species classification and detection dataset”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8769–8778.

[96] Rui Huang and Yixuan Li. “Mos: Towards scaling out-of-distribution detection for large semantic space”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8710–8719.

[97] Olga Russakovsky, Jia Deng, Hao Su, et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* (2015).

[98] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, et al. “Describing textures in the wild”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 3606–3613.

[99] Janis Postels, Hermann Blum, Yannick Strümpler, et al. “The hidden uncertainty in a neural networks activations”. In: *arXiv preprint arXiv:2012.03082* (2020).

[100] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, et al. “Spectral Normalization for Generative Adversarial Networks”. In: *International Conference on Learning Representations*. 2018.

[101] Jeremiah Liu, Zi Lin, Shreyas Padhy, et al. “Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, et al. 2020, pp. 7498–7512.

[102] Joost R. van Amersfoort, Lewis Smith, Yee Whye Teh, et al. “Simple and Scalable Epistemic Uncertainty Estimation Using a Single Deep Deterministic Neural Network”. In: *ICML*. 2020.

[103] Jishnu Mukhoti, Andreas Kirsch, Joost van Amersfoort, et al. “Deterministic Neural Networks with Inductive Biases Capture Epistemic and Aleatoric Uncertainty”. In: *arXiv preprint arXiv:2102.11582* (2021).

[104] Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, et al. “Deep Variational Information Bottleneck”. In: *International Conference on Learning Representations*. 2017.

[105] Alex Alemi, Ian Fischer, and Josh Dillon, eds. *Uncertainty in the Variational Information Bottleneck*. 2018. URL: https://arxiv.org/abs/1807.00906.

[106] Kaiming He, Haoqi Fan, Yuxin Wu, et al. “Momentum contrast for unsupervised visual representation learning”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020.

[107] Jean-Bastien Grill, Florian Strub, Florent Altché, et al. “Bootstrap your own latent - a new approach to self-supervised learning”. In: *Advances in Neural Information Processing Systems* (2020).

[108] Mathilde Caron, Ishan Misra, Julien Mairal, et al. “Unsupervised learning of visual features by contrasting cluster assignments”. In: *Advances in Neural Information Processing Systems* (2020).

[109] Richard Zhang, Phillip Isola, and Alexei A Efros. “Colorful image colorization”. In: *European conference on computer vision*. 2016.

[110] Mehdi Noroozi and Paolo Favaro. “Unsupervised learning of visual representations by solving jigsaw puzzles”. In: *European conference on computer vision*. 2016.

[111] Ishan Misra and Laurens van der Maaten. “Self-supervised learning of pretext-invariant representations”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020.

[112] Kaiming He, Xinlei Chen, Saining Xie, et al. “Masked autoencoders are scalable vision learners”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 16000–16009.

[113] Jinghao Zhou, Chen Wei, Huiyu Wang, et al. “Image BERT Pre-training with Online Tokenizer”. In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=ydopy-e6Dg.

[114] Bolei Zhou, Hang Zhao, Xavier Puig, et al. “Semantic understanding of scenes through the ade20k dataset”. In: *International Journal of Computer Vision* (2019).

[115] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, et al. “Using self-supervised learning can improve model robustness and uncertainty”. In: *Advances in neural information processing systems* (2019).

[116] Jim Winkens, Rudy Bunel, Abhijit Guha Roy, et al. “Contrastive training for improved out-of-distribution detection”. In: *arXiv preprint arXiv:2007.05566* (2020).

[117] Izhak Golan and Ran El-Yaniv. “Deep anomaly detection using geometric transformations”. In: *Advances in Neural Information Processing Systems*. 2018.

[118] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, et al. “Generative adversarial networks”. In: *Communications of the ACM* (2020).

[119] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. “Pixel recurrent neural networks”. In: *International conference on machine learning*. PMLR. 2016.

[120] Laurent Dinh, David Krueger, and Yoshua Bengio. “Nice: Non-linear independent components estimation”. In: *arXiv preprint arXiv:1410.8516* (2014).

[121] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. “Density estimation using Real NVP”. In: *International Conference on Learning Representations*. 2017.

[122] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, et al. “Deep unsupervised learning using nonequilibrium thermodynamics”. In: *International Conference on Machine Learning*. 2015.

[123] Yang Song and Stefano Ermon. “Generative Modeling by Estimating Gradients of the Data Distribution”. In: *Advances in Neural Information Processing Systems*. 2019.

[124] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851.

[125] Eric T. Nalisnick, A. Matsukawa, Y. Teh, et al. “Do Deep Generative Models Know What They Don’t Know?” In: *arXiv preprint arXiv:1810.09136* (2019).

[126] Jinwon An and Sungzoon Cho. “Variational Autoencoder based Anomaly Detection using Reconstruction Probability”. In: 2015.

[127] Houssam Zenati, Chuan Sheng Foo, Bruno Lecouat, et al. “Efficient GAN-Based Anomaly Detection”. In: *arXiv preprint arXiv:1802.06222* (2018).

[128] Dan Hendrycks, Mantas Mazeika, and Thomas G. Dietterich. “Deep Anomaly Detection with Outlier Exposure”. In: *arXiv preprint arXiv:1812.04606* (2018).

[129] Kimin Lee, Honglak Lee, Kibok Lee, et al. “Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples”. In: *2018 International Conference on Learning Representations (ICLR)*. 2018.

[130] Andrey Malinin and Mark Gales. “Reverse KL-Divergence Training of Prior Networks: Improved Uncertainty and Adversarial Robustness”. In: *NeurIPS*. 2019.

[131] Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, et al. “Your classifier is secretly an energy based model and you should treat it like one”. In: *International Conference on Learning Representations*. 2020.

[132] M Gadd, D de Martini, L Marchegiani, et al. “Sense-Assess-eXplain (SAX): Building Trust in Autonomous Vehicles in Challenging Real-World Driving Scenarios”. In: *2020 IEEE Intelligent Vehicles Symposium (IV): Workshop on Ensuring and Validating Safety for Automated Vehicles (EVSAV)*. 2020.

[133] Hassan Alhaija, Siva Mustikovela, Lars Mescheder, et al. “Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes”. In: *International Journal of Computer Vision (IJCV)* (2018).

[134] Oliver Zendel, Katrin Honauer, Markus Murschitz, et al. “WildDash - Creating Hazard-Aware Benchmarks”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[135] Christoph Schuhmann, Romain Beaumont, Richard Vencu, et al. “Laion-5b: An open large-scale dataset for training next generation image-text models”. In: *arXiv preprint arXiv:2210.08402* (2022).

[136] Tongzhou Wang and Phillip Isola. “Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere”. In: *Proceedings of the 37th International Conference on Machine Learning*. Proceedings of Machine Learning Research. PMLR.

[137] Prannay Khosla, Piotr Teterwak, Chen Wang, et al. “Supervised contrastive learning”. In: *arXiv preprint arXiv:2004.11362* (2020).

[138] Nicholas Frosst, Nicolas Papernot, and Geoffrey Hinton. "Analyzing and improving representations with the soft nearest neighbor loss". In: *International conference on machine learning*. 2019.

[139] Adam Paszke, Sam Gross, Francisco Massa, et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* (2019).

[140] Liang-Chieh Chen, Yukun Zhu, George Papandreou, et al. "Encoder-decoder with atrous separable convolution for semantic image segmentation". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 801–818.

[141] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, et al. "Cutmix: Regularization strategy to train strong classifiers with localizable features". In: *IEEE/CVF international conference on computer vision*. 2019.

[142] Hanxun Huang, Xingjun Ma, Sarah Monazam Erfani, et al. "Unlearnable Examples: Making Personal Data Unexploitable". In: *International Conference on Learning Representations*. 2021.

[143] Yuandong Tian, Xinlei Chen, and Surya Ganguli. "Understanding self-supervised learning dynamics without contrastive pairs". In: *Proceedings of the 38th International Conference on Machine Learning*. 2021.

[144] Li Jing, Pascal Vincent, Yann LeCun, et al. "Understanding dimensional collapse in contrastive self-supervised learning". In: *arXiv preprint arXiv:2110.09348* (2021).

[145] Pascal Mettes, Elise Van der Pol, and Cees Snoek. "Hyperspherical prototype networks". In: *Advances in neural information processing systems* (2019).

[146] Jishnu Mukhoti and Yarin Gal. "Evaluating bayesian deep learning methods for semantic segmentation". In: *arXiv preprint arXiv:1811.12709* (2018).

[147] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, et al. "Invertible residual networks". In: *International conference on machine learning*. PMLR. 2019.

[148] Bowen Cheng, Ishan Misra, Alexander G Schwing, et al. "Masked-attention mask transformer for universal image segmentation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 1290–1299.

[149] Edward J Hu, Yelong Shen, Phillip Wallis, et al. “LoRA: Low-Rank Adaptation of Large Language Models”. In: *International Conference on Learning Representations*. 2022.

[150] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. “Vision transformers for dense prediction”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021.

[151] Zhe Chen, Yuchen Duan, Wenhai Wang, et al. “Vision Transformer Adapter for Dense Predictions”. In: *The Eleventh International Conference on Learning Representations*. 2023.

[152] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the knowledge in a neural network”. In: *arXiv preprint arXiv:1503.02531* (2015).

[153] Hugo Touvron, Matthieu Cord, Matthijs Douze, et al. “Training data-efficient image transformers & distillation through attention”. In: *International conference on machine learning*. PMLR. 2021.

[154] Quentin Duval, Ishan Misra, and Nicolas Ballas. “A simple recipe for competitive low-compute self supervised vision models”. In: *arXiv preprint arXiv:2301.09451* (2023).

[155] Jishnu Mukhoti, Yarin Gal, Philip HS Torr, et al. “Fine-tuning can cripple your foundation model; preserving features may be the solution”. In: *arXiv preprint arXiv:2308.13320* (2023).

[156] Mahmoud Assran, Mathilde Caron, Ishan Misra, et al. “Semi-supervised learning of visual features by non-parametrically predicting view assignments with support samples”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021.

[157] Zhenda Xie, Zheng Zhang, Yue Cao, et al. “SimMIM: A Simple Framework for Masked Image Modeling”. In: *International Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[158] Hangbo Bao, Li Dong, Songhao Piao, et al. “BEiT: BERT Pre-Training of Image Transformers”. In: *International Conference on Learning Representations*. 2022.

[159] Gang Li, Heliang Zheng, Daqing Liu, et al. “Semmae: Semantic-guided masking for learning masked autoencoders”. In: *Advances in Neural Information Processing Systems* (2022).

[160] Yuge Shi, N Siddharth, Philip Torr, et al. "Adversarial masking for self-supervised learning". In: *International Conference on Machine Learning*. 2022.

[161] Ioannis Kakogeorgiou, Spyros Gidaris, Bill Psomas, et al. "What to hide from your students: Attention-guided masked image modeling". In: *European Conference on Computer Vision*. 2022.

[162] Mathilde Caron, Hugo Touvron, Ishan Misra, et al. "Emerging properties in self-supervised vision transformers". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021.

[163] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, et al. *OpenCLIP*. Version 0.1. DOI: 10.5281/zenodo.5143773. URL: https://doi.org/10.5281/zenodo.5143773.