

Predicting and Improving Perception Performance for Robotics Applications



Corina Gurău

Wolfson College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2018

Acknowledgements

There are many people who have helped me throughout my DPhil and without whom this thesis would not have been possible. First and foremost, I would like to thank my supervisor Ingmar Posner for the incredible support and freedom he gave me throughout the entire process. In between our first and last meetings I have gained a lot of knowledge thanks to his kind guidance, dedication and trust.

This research would also not have been possible without the invaluable advice and mentorship from Chi Hay Tong, Alex Bewley and Dushyant Rao. I am extremely grateful for the time you took to meet, discuss and share ideas with me and I can only hope to be as big of a help to the next generation of researchers. I would also like to thank Peter Ondruska, Winston Churchill and Letizia Marchegiani for opportunity to work together.

No words can express the depth of my gratitude towards the entire ORI community. I am privileged to have had the opportunity to work in such a unique environment and to have been surrounded by bright, motivated and kind people on a daily basis.

To Matt, Jeff, Geoff and Chris – I feel lucky to have started at the same time as you, as I could not have asked for a more fun group of people to go through deadlines with. The only thing that saddens me is thinking about how it would have been like if Raja were still with us today.

A big thank you to Julie, Hillary, Stefan, Ollie, Dan, Markus and Hugo for having been there for me when I needed to figure things out but also for having fun with me along the way. By living together, travelling or simply spending time in and outside of the lab, you made my past years some of the most enjoyable yet.

Thanks to everyone at the Oxford University Karate Club for keeping me fit and sane for all these years. Training for Varsity with you has been one of my favourite things in Oxford. A special thanks to Emily - I will forever treasure our salad dinners and veggie market runs.

To keep things balanced, I am also incredibly grateful for all the parties, trips and late night conversations with my Wolfson College friends. Lydia, Claudio,

Luke, Claudia, Maurits, Nina, thanks for being a constant source of entertainment and showing me the fun side of being a graduate student. I am glad we never shied away from discussing controversial topics and I am thankful for your different perspectives. You kept me happy even when research was not going so well.

Through home-cooked meals, civic engagement, activism and contagious kindness, the Romanian community in Oxford managed to bring home closer to me, which I am incredibly thankful for.

I am also indebted to Jack for putting a smile on my face even in the toughest of days. Your constant encouragement and support made the final years of my DPhil much easier than they would have been otherwise and our adventures on various continents will stay with me for a lifetime.

To my friends at home – Mihaela, Erika, Andreea – thanks for keeping a close connection and welcoming me with open arms despite the physical distance and the years we've been apart. Our jokes seem to never grow old.

Last but not least, to my parents Dana and Horia, thank you for all the sacrifices you made for me and for your unconditional love and support. I owe everything to you.

Abstract

Perception systems are often the core component of a robotics framework as their ability to accurately interpret sensor data is essential for autonomy. The goal of this thesis is to estimate and improve the perception performance of a mobile robot across large areas of operation, particularly when there are no guarantees that the testing data distribution will match the training distribution. Such situations are prevalent for autonomous mobile robots operating outdoors under a variety of environmental conditions.

This thesis explores the *adaptability* of vision systems by training place-specific models which outperform generic ones. We show that it is possible to train such models in a self-supervised fashion using geometric scene constraints without relying on costly image annotations.

This thesis also explores the *awareness* that vision systems have of their own capability to make correct predictions at any given moment in time. We approach this problem from two different vantage points: firstly, through *performance records* which model perception performance as a function of location and appearance and, secondly, through intrinsic model uncertainty, or *introspection* as introduced by [Grimmett et al., 2016].

Performance records allow an autonomous agent to estimate the likelihood of making a mistake during future traversals of the same place. In a use-case scenario regarding offering or denying autonomy, we show that an agent is able to estimate when its confidence levels are low, deny autonomy, and reduce the number of perception mistakes made.

Introspection refers to the ability of a model to associate an appropriate assessment of confidence with any test case. We introduce an efficient way to obtain well-calibrated and reliable uncertainty scores from neural networks. Our method is more computationally efficient than Bayesian neural networks or model ensembles which, despite being well-calibrated, are more cumbersome to train and slower to test. Additionally, we believe that we are the first to propose more introspective detectors within a state of the art object detection framework such as Faster R-CNN.

This thesis proposes vision systems that are not only more accurate but also whose failures can be more reliably predicted. In doing so, we advocate practical solutions that often make use of tools specific to robotics such as additional sensing modalities or localisation maps pertaining to an autonomous vehicle, but we also touch upon machine learning techniques such as Bayesian deep learning.

While striving for high accuracy remains a crucial endeavour, given the safety-critical nature of robot perception, we believe that estimating reliability, introspection, and diagnosing failure are indispensable when operating in cluttered, complex, and ever-changing environments.

Contents

List of Figures	x
List of Tables	xii
1 Introduction	1
1.1 Motivation	1
1.2 Contributions	3
1.3 Thesis Structure	4
1.4 Publications	5
2 Preliminaries	7
2.1 Supervised Learning	8
2.1.1 Loss Function	8
2.1.2 Model Choice	10
2.1.3 Training	12
2.2 Fundamentals of Object Detection	13
2.2.1 Modern Detectors	14
2.2.2 Sliding-Window Detectors	15
2.2.3 Feature Representation	16
2.2.4 Non-Maximum Suppression	18
2.2.5 Hard Negative Mining	18
2.3 Evaluation Metrics	19
2.3.1 Accuracy	19
2.3.2 Calibration	21
2.4 Conclusion	23

3	Improving Robot Perception Through Experience	24
3.1	The Experience Paradigm	25
3.2	Motivation and Background	27
3.3	Framework Description	29
3.3.1	Object Detection	30
3.3.2	Scene Filtering using Map Priors	31
3.3.3	Memory Bank and Model Retraining	36
3.4	Experimental Evaluation	37
3.4.1	Sensor Suite	37
3.4.2	Baseline Detection Model	38
3.4.3	Datasets	39
3.4.4	Environment-Specific Tuning	39
3.4.5	Comparison with Hard Negative Mining	44
3.5	Discussion	46
3.5.1	Knowing When to Retrain	46
3.5.2	Sources of Error	46
3.6	Conclusions	47
4	Predicting the Performance of Object Detectors	49
4.1	Motivation	50
4.2	Related Work	52
4.2.1	Relation to Introspection	54
4.3	Framework Description	56
4.3.1	Estimating Perception Performance	57
4.3.2	Decision Making using a Performance Record	59
4.3.3	Performance Records and the Experience Paradigm	61
4.4	Experimental Results	63
4.4.1	Pedestrian Detector	65
4.4.2	Surrogate Ground Truth	65
4.4.3	Performance Estimation and Decision Making	68
4.5	Discussion	75
4.5.1	Reasoning with Model Uncertainty	75
4.5.2	Correcting Mistakes	76
4.6	Conclusions	77

5	Dropout Distillation for Efficiently Estimating Model Confidence	78
5.1	Motivation	79
5.2	Related Work	80
5.3	Approach	81
5.3.1	MC-Dropout	82
5.3.2	Knowledge Distillation	83
5.3.3	The Distilled Dropout Network	84
5.4	Classification Experiments	85
5.4.1	Dataset and Network Architecture	85
5.4.2	Results	87
5.4.3	CIFAR-10 Qualitative Examples	92
5.5	Object Detection Experiments	95
5.5.1	Dataset and System Architecture	95
5.5.2	Results	97
5.5.3	Faster R-CNN Qualitative Examples	99
5.5.4	Faster R-CNN with Out-of-Distribution Data	102
5.6	Discussion	102
5.6.1	Approximating an Ensemble	102
5.6.2	System I and System II	104
5.6.3	Probability Density Distillation	105
5.7	Conclusion	105
6	Conclusions and Future Work	107
6.1	Summary of Contributions	107
6.2	Future Work	109
6.2.1	Predicting Detection Failure	109
6.2.2	Anticipating Failure in Advance	110
6.2.3	Expert Detectors According to Performance Records	110
6.2.4	Testing DDN on Out-of-Distribution Data	111
6.3	Closing Remarks	112
	Bibliography	113

List of Figures

3.1	Figure showing a variety of experiences	25
3.2	Proposed EBC framework	27
3.3	EBC system diagram	29
3.4	Figure illustrating the EBC scene filtering step	31
3.5	Figure showing image detections rejected by the scene filter	35
3.6	The Wildcat vehicle and a map depicting the routes used for experiments	38
3.7	Precision-Recall and Miss Rate - False Positives Per Image plot for multiple EBC cycles	40
3.8	Average Precision (AP) plot for EBC detector tested on North Oxford January test set	41
3.9	Average precision for model trained on all data available.	42
3.10	Figure shows the performance increase provided by the scene filter (referred to as ‘validated’ detector) when applied to both the base detector and the final EBC detector on the North Oxford January test dataset.	43
3.11	Precision-Recall curves comparing HNM to EBC iterations for a portion of the North Oxford January dataset	45
4.1	Example data encountered by a robot as it traverses an urban environment	51
4.2	Overview of the proposed performance prediction method	57
4.3	Figure shows the expected loss of choosing an action for an example posterior distribution	61
4.4	A simple illustration of the experience-based navigation framework	63
4.5	Example images from a test route in Milton Keynes as selected by LOC and APP	64
4.6	Figure showing the platform and the route chosen for experiments .	65
4.7	Vote3d pedestrian detection example	66

4.8	Heatmap showing a strong correlation between performance scores computed using 2D ground truth annotations and 3D laser detections	68
4.9	Number of mistakes after the decision making process	70
4.10	Example images and their corresponding pedestrian detections using MS-CNN	73
4.11	Example images and their corresponding pedestrian detections using the SVM+ACF detector	74
5.1	CIFAR-10 training and testing accuracy	87
5.2	Histograms of output probability scores on CIFAR-10 dataset . . .	88
5.3	Calibration plot on CIFAR-10 test data	89
5.4	Risk-Coverage plot on CIFAR-10 test data	89
5.5	Qualitative examples of DDN on the CIFAR-10 test set	93
5.6	Qualitative examples of DDN on the CIFAR-10 test set	94
5.7	Calibration plot on Faster R-CNN test data	98
5.8	MSCE, MCE and NLL scores for different weight values λ on the distillation loss.	99
5.9	Qualitative examples for DDN on the COCO dataset	100
5.10	Qualitative examples for DDN on the COCO dataset	101
5.11	Figure shows the calibration of the baseline and DDN models on Central Oxford data.	103
5.12	Calibration plot on CIFAR-10 test data	104
6.1	Proposed model for predicting detection failure using appearance information only	110

List of Tables

3.1	A summary of the datasets used for training and testing the Experience-Based Classification framework	39
4.1	Percentage of decision making mistakes (both type I and type II) for two different detectors and two different action costs	71
4.2	Percentage of decision mistakes for an equal amount of the route driven autonomously	72
5.1	DDN calibration and timing results on CIFAR-10 test set	90
5.2	DDN negative log-likelihood and entropy on CIFAR-10 test set . . .	91
5.3	DNN (within Faster R-CNN) calibration and accuracy results on COCO test set	97

1

Introduction

1.1 Motivation

Perceiving and understanding the visual world is a fundamental requirement for robot autonomy - without accurately interpreting sensor data, a robot could not act autonomously in our surroundings. Some robots are physically embodied in our world and their perception poses an additional important challenge: extremely low acceptable error rates. Even with the recent breakthroughs in the fields of computer vision and machine learning, failures of vision systems are still too common to enable safe autonomous operation.

The goal of this work is to improve the perception capabilities of computer vision systems operating in the real world. To do so, we touch upon two key aspects of their operation: *adaptation* and *awareness*. Through adaptation, a computer vision

system (for instance, an image-based object detector) quickly improves in its place of operation, in order to respond appropriately to the environment it is currently in. For instance, one way to do so is by updating existing models such that they are better suited for specific environments or contexts. Through awareness, we equip the system with an understanding of its limitations and therefore an ability to ask for help when it is not able to make a reliable prediction. We believe that the problem of instilling an appropriate amount of confidence in a robot’s prediction has not yet received the attention it deserves from the robotics community. We tackle this problem from two different vantage points.

Firstly, we employ tools commonly used in robotics such as semantic maps and localisation in order to keep track of performance over time and space. The goal is to build probabilistic models that aid the robot in predicting its success (or failure) at performing a task. We propose a framework that allows it to do so by *retrospectively* looking at its performance on encounters of a similar environment. This approach decouples the confidence assessment from the task itself, being more similar to a meta-model that simply observes the behaviour of the main system.

Secondly, we employ tools from the machine learning community such as Bayesian deep learning and propose a more efficient way to obtain uncertainty scores from neural networks. In this setting, the model that performs the task of interest is additionally equipped with *introspection* - the ability to estimate its own confidence levels.

Both introspection and retrospection are used in decision making scenarios in which the robot is given the option to perform a visual task autonomously or ask a human for help. We show that the robot performs better at choosing the action which minimises the number of perceptual mistakes when it has a more appropriate measure of uncertainty.

1.2 Contributions

We make the following contributions:

- Learning *place-specific models* that adapt to the operational environment of the robot (Chapter 3). We show that by making efficient use of cycles of interaction with the world we can learn in a self-supervised way more accurate object detectors. Our framework does not require additional manual annotations which are difficult to obtain, but instead leverages scene constraints. While using a single, highly generalisable model for object detection remains a noble goal, our experiments show that training a collection of models such that each serves as a local expert is not only a more practical approach, but also one that currently yields better results.
- The concept of *performance records*, a framework for gathering perception performance estimates over time and space (Chapter 4). They confirm the belief that a single estimate of performance (i.e. task accuracy on a static benchmark) is insufficient for establishing the reliability of a vision system for large scale outdoor operation. We propose two approaches for building performance records (one of which makes use of the visual appearance of a place), as well as a use-case scenario regarding the disabling of autonomy when our records indicate poor performance. Incorporating such information into the way a robot acts gives us more control over potential errors (i.e. type I and type II) and choosing to perform actions with a lower cost associated to them.
- Efficiently learning to predict *better calibrated confidence scores* (Chapter 5). Neural networks are known to be overconfident in their predictions, which makes them difficult to deploy in real-world robotics applications. We propose a model that is more reliable than standard (non-Bayesian) neural networks and computationally more efficient than Bayesian neural networks,

which despite being better calibrated, are computationally expensive to use in practice. To our knowledge, we are the first to propose better calibrated object models within a state of the art detection framework (Faster R-CNN [Ren et al., 2017a]).

1.3 Thesis Structure

Chapter 2 outlines the image-based object detection task – one of the main applications addressed throughout this thesis. We describe the learning process as well as the metrics used for the evaluation in classification and detection. We also use this chapter to introduce the reader to techniques such as hard negative mining (Section 2.2.5) and non-maxima suppression (Section 2.2.4) whose usage will become more clear in the context of future chapters. All sections in Chapter 2 are referenced outside of the chapter itself such that the reader may skip some of the concepts and come back whenever necessary.

In Chapter 3 we take up the problem of adaptability of object detectors. We describe a framework for learning environment-specific object detectors relevant for domains that can encounter shift in the data distribution, such as autonomous platforms operating outdoors over large areas and extended periods of time. We show that place (location/weather) specific models outperform generic ones when used in the place for which they are tailored. We envision any workspace that is significantly different to all others having its own expert detector.

In Chapter 4 we propose to keep track of the performance of an object detector via a *performance record* in order to estimate the likelihood of a perception failure online, during operation. We show that detectors are persistent in their mistakes across visual experiences and argue that it is valuable to know when and where they happen. Our experiments demonstrate that if the robot could, in a decision-making scenario, choose whether to operate autonomously or hand over control to a user,

then it could lower the number of perception mistakes and have overall better perception performance by occasionally denying autonomy.

Chapter 5 touches upon a different measure of trust in machine learning, namely intrinsic model uncertainty. In order to be introspective (i.e. have a reliable, well-calibrated confidence prediction), neural networks require multiple samples from the distribution of the model parameters. Sampling is time-consuming and despite making models more accurate and introspective, it is unsuitable for real-time applications in robotics. To overcome this problem, we propose a model that learns an approximation to a model ensemble, is fast to use at test time and, most importantly, makes trustworthy predictions.

Finally, in Chapter 6 we summarise the contributions reported in this thesis and propose future directions of research inspired by our findings.

1.4 Publications

The research described in Chapter 3 on place-specific object detectors has appeared in the following publications:

- **Corina Gurău**, Jeffrey Hawke, Chi Hay Tong, Ingmar Posner, “Learning on the Job: Improving Robot Perception Through Experience”, Neural Information Processing Systems in (NIPS) Workshop: Autonomously Learning Robots, 2014.
- Jeffrey Hawke, **Corina Gurău**, Chi Hay Tong, Ingmar Posner, “Wrong Today, Right Tomorrow: Experience-Based Classification for Robot Perception”, Field and Service Robotics (FSR), 2015.

The work presented in Chapter 4 on performance estimation has appeared in:

- **Corina Gurău**, Dushyant Rao, Chi Hay Tong, Ingmar Posner, “Learn from Experience: Probabilistic Prediction of Perception Performance to Avoid Failure”, The International Journal of Robotics Research (IJRR), 2017.

- **Corina Gurău**, Chi Hay Tong, Ingmar Posner, “Fit for Purpose? Predicting Perception Performance based on Past Experience”, International Symposium on Experimental Robotics, 2016.

Additionally, although their content is not present in this document, the following publications have also had an influence on the work described in this thesis:

- Oliver Bartlett, **Corina Gurău**, Letizia Marchegiani, Chi Hay Tong, Ingmar Posner, “Enabling Intelligent Energy Management for Robots using Publicly Available Maps”, International Conference on Intelligent Robots and Systems (IROS), 2016.
- Peter Ondruska, **Corina Gurău**, Letizia Marchegiani, Chi Hay Tong, Ingmar Posner, “Scheduled Perception for Energy-Efficient Path Following”, International Conference on Robotics and Automation (ICRA), 2015.
- Winston Churchill, Chi Hay Tong, **Corina Gurău**, Ingmar Posner, Paul Newman, “Know Your Limits: Embedding Localiser Performance Models in Teach and Repeat Maps”, International Conference on Robotics and Automation (ICRA), 2015.

2

Preliminaries

Visual object detection is a fundamental problem in computer vision. It deals with identifying all instances of a physical or semantic class in a static image and reporting their location via, for instance, bounding boxes or pixel segmentation. When applied to robotics and autonomous vehicles in particular, common classes of interest are pedestrians, cars and cyclists, as they are the most likely to interact with the robot. Most successful approaches use machine learning and require large datasets of examples. The accuracy of the trained model at the detection task depends on many factors such as the data used, the choice of model and the training procedure for finding the best model parameters.

The purpose of this chapter is to introduce the reader to the background knowledge relied upon in the rest of this thesis. It details concepts or procedures specific to supervised learning from image data which we will occasionally refer

back to in subsequent chapters without breaking the flow of the specific research question addressed in that chapter.

2.1 Supervised Learning

This section covers the essential parts of supervised learning: common loss functions, choices of models and the optimisation procedures used for learning.

2.1.1 Loss Function

The goal of supervised learning is finding a set of weights $\mathbf{w}$ which parametrise a function $f_w : \mathcal{X} \rightarrow \mathcal{Y}$ such that $\mathbf{y} = f_w(\mathbf{x})$ best models the dataset $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$, where $\mathbf{x} \in \mathcal{X}$ are the input observations and $\mathbf{y} \in \mathcal{Y}$ are the output labels. How well f_w fits $\mathcal{D}$ is measured by a scalar-valued loss function $\mathcal{L}(\mathbf{w})$, composed of a data term $\mathcal{L}_{\mathcal{D}}$ and a regularisation term $\mathcal{L}_{\mathcal{W}}$ whose importance is controlled by parameter λ :

$$\mathcal{L}(\mathbf{w}) = \mathcal{L}_{\mathcal{D}}(\mathbf{w}) + \lambda \mathcal{L}_{\mathcal{W}}(\mathbf{w}). \quad (2.1)$$

The goal of learning is to find $\mathbf{w}^*$ by solving the optimisation problem:

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \mathcal{L}(\mathbf{w}). \quad (2.2)$$

The data term $\mathcal{L}_{\mathcal{D}}$, also called the empirical risk term, is computed over all N examples in the training set. In the case of regression, the observation labels $\mathbf{y} \in \mathcal{Y}$ have continuous values and a common choice for the loss is the L_2 squared loss:

$$\mathcal{L}_{\mathcal{D}}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N \|\mathbf{y}_i - f_w(\mathbf{x}_i)\|^2 \quad (2.3)$$

In the case of binary classification, a common choice of loss function is the hinge loss:

$$\mathcal{L}_D(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N \max(0, 1 - \mathbf{y}_i f_w(\mathbf{x}_i)) \quad (2.4)$$

We make use of the hinge loss in Chapter 3 in order to train a support vector machine (SVM) for binary classification. When the model f_w has a probabilistic output, a common choice for the loss function is the negative log likelihood:

$$\mathcal{L}_D(\mathbf{w}) = -\frac{1}{N} \sum_{i=1}^N \log f_w(\mathbf{x}_i). \quad (2.5)$$

In this case the optimisation is equivalent to maximum likelihood estimation (MLE):

$$\mathbf{w}^* = \mathbf{w}_{ML} = \arg \max_{\mathbf{w}} p(\mathcal{D}|\mathbf{w}), \quad (2.6)$$

where the function $p(\mathcal{D}|\mathbf{w})$ is the model likelihood. In a Bayesian setting, the model parameters $\mathbf{w}$ are treated as random variables themselves. If a prior distribution $p(\mathbf{w})$ over model parameters $\mathbf{w}$ is assumed, one can use the posterior distribution instead of the likelihood in order to obtain the maximum a posteriori estimate (MAP):

$$\mathbf{w}^* = \mathbf{w}_{MAP} = \arg \max_{\mathbf{w}} p(\mathcal{D}|\mathbf{w})p(\mathbf{w}). \quad (2.7)$$

The optimisation criteria is usually defined in log-space instead of using the density values themselves in order to be numerically more stable. Both $\mathbf{w}_{ML}$ and $\mathbf{w}_{MAP}$ are point estimate approximations to the posterior $p(\mathbf{y}|\mathbf{x}, \mathcal{D})$. Bayesian inference uses instead the full posterior distribution over the weights $p(\mathbf{w}|\mathcal{D})$ to compute the posterior predictive distribution:

$$p(\mathbf{y}|\mathbf{x}, \mathcal{D}) = \int p(\mathbf{y}|\mathbf{x}, \mathbf{w})p(\mathbf{w}|\mathcal{D})d\mathbf{w}. \quad (2.8)$$

The difference between point estimates commonly used by many machine learning

models (neural networks, SVMs, etc.) and Bayesian inference is that the latter provides a quantification of uncertainty in the model parameters $\mathbf{w}$ through the probability distribution $p(\mathbf{w}|\mathcal{D})$.

When trying to describe the input data in an interpretable way or when assessing the output of a computer vision system we often rely on this type of uncertainty (referred to as *epistemic uncertainty* in [Kendall and Gal, 2017]). We will address the reliability of this estimate in Chapter 5.

2.1.2 Model Choice

We exemplify the choice of model with two parametric models that have been used in this thesis, namely linear models such as support vector machines (SVMs) and deep neural networks (DNNs).

Support Vector Machines

The simplest model is a linear mapping $f_w(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$, $\mathbf{w} \in \mathbb{R}^n, w_0 \in \mathbb{R}$ applied directly to the input data $\mathbf{x} \in \mathbb{R}^n$. In the case of inputs that are not linearly separable such as pixel values, a linear model is unable to capture sufficiently complex patterns. In order to improve linear separability as well as encode invariance to lighting and colour, input vectors are non-linearly mapped to a very high-dimensional, potentially infinite, feature space through a feature transform ϕ :

$$f_w(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + w_0. \quad (2.9)$$

Introduced by [Cortes and Vapnik, 1995], a support vector machine (SVM) is an example of such a model. The feature transform ϕ is implicitly specified by a kernel function $k : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ which dictates the shape of the decision surface. For every positive definite kernel $k(\mathbf{x}, \mathbf{x}')$ there exists a mapping $\phi(\mathbf{x})$ of the input data to a Hilbert space $\mathcal{H}$ such that $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}$ ([Scholkopf and Smola, 2001]).

The SVM classification function f_w can be written in its dual and primal form as:

$$f_w(\mathbf{x}) = \underbrace{\mathbf{w}^T \phi(\mathbf{x}) + w_0}_{\text{primal}} = \sum_{i=1}^m \underbrace{\alpha_i y_i k(\mathbf{x}_i, \mathbf{x})}_{\text{dual}} + w_0, \quad (2.10)$$

where the α_i are the Lagrange multipliers associated to the data and $\mathbf{x}_i$ are the m representative data points found during training (the support vectors). The kernel can be either a linear or a non-linear function with the most common of the latter being the squared exponential kernel also known as the radial basis function (RBF).

Neural Networks

In recent years however, (deep) neural networks have become a more popular model choice due to their increased learning capacity. DNNs learn a hierarchy of L layer functions $\{f^{(l)}\}_{l=1}^L$ defined by the following recursive procedure:

$$f^{(l)}(\mathbf{x}) = \phi^{(l)}(\mathbf{w}^{(l)} f^{(l-1)}(\mathbf{x}) + b^{(l)}), \quad (2.11)$$

where $f^0(\mathbf{x}) = \mathbf{x}$ and ϕ is a non-linear function. The classifier function is $f_w(\mathbf{x}) = f^{(L)}(\mathbf{x})$ with model parameters $w = \{\mathbf{w}^{(l)}, b^{(l)}\}_{l=1}^L$. While SVMs learn a linear function on top of a fixed non-linear feature transform (typically specified implicitly via a kernel), DNNs learn a non-linear function of the raw input. Some of the most commonly used non-linear (also called activation) functions are the logistic sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \quad (2.12)$$

and the rectified linear units (ReLU):

$$\text{ReLU}(x) = \max(0, x). \quad (2.13)$$

2.1.3 Training

The SVM dual objective is conveniently a convex function and its solution (i.e. the parameters α_i) is most commonly found using dual stochastic coordinate ascent ([Hsieh et al., 2008]). With large datasets however, finding the support vectors can be slow, in which case SVMs are preferably solved in the primal form instead, using stochastic gradient descent ([Shalev-Shwartz and Tewari, 2011]).

Stochastic gradient descent (SGD) is also most suitable for training neural networks on large datasets. In particular, SGD with randomised subsets of the training data (mini-batches) reduces the variance of the parameter updates and leads to a more stable convergence. The change in the $\mathbf{w}$ is given by:

$$\mathbf{w}_\tau = \mathbf{w}_{\tau+1} - \eta \frac{d}{d\mathbf{w}} \mathcal{L}(\mathbf{w}_\tau), \quad (2.14)$$

where τ denotes the iteration number and η the learning rate parameter which is usually decayed after having performed half of the training iterations. The initial model weights $\mathbf{w}_0$ are drawn randomly from a gaussian distribution centred at 0. This process is repeated until convergence.

Regularisation

Model regularisation, imposed through the regularisation loss $\mathcal{L}_W$ in Equation 2.1, helps reduce the model complexity such that it improves its generalisation to unseen data. One of the most popular losses for weight regularisation is the L_1 loss:

$$\mathcal{L}_W(\mathbf{w}) = \sum_{i=1}^n |w_i|, \quad (2.15)$$

which encourages zero weights (and therefore model sparsity), as well as the L_2 loss:

$$\mathcal{L}_W(\mathbf{w}) = \sum_{i=1}^n w_i^2, \quad (2.16)$$

which encourages small weights and can additionally be interpreted as having a gaussian prior on the weights (and therefore computing the maximum a posteriori estimate).

Additionally, stochastic regularisation techniques have been shown to improve model generalisation, particularly for neural networks with a large number of parameters. They do not modify the loss but rather the network itself. One such approach is dropout ([Srivastava et al., 2014]) which amounts to setting the units in the network to zero with a certain probability p during training in order to prevent their co-adaptation. Dropout has been traditionally used only in training but more recently it has been interpreted as a technique for performing approximate inference in Bayesian neural networks ([Gal and Ghahramani, 2015]).

Cross Validation

Cross validation is used to assess how well a model will generalise to new, unseen data. It is performed on a set of data which is distinct from the test set in order to find optimal hyperparameters such as the λ regularisation weight. In cases where data are scarce, the training set is partitioned into multiple subsets (folds) and each acts in turn as a validation set. This is called k -fold cross validation, where k is the number of folds. For multiple hyperparameters and time-consuming training procedures, the technique is less practical as it can require a number of training runs that is exponential in the number of hyperparameters.

2.2 Fundamentals of Object Detection

Vision-based object detection is used as an application of the research ideas presented throughout all the chapters in this thesis. The background and related works on object detection are introduced here rather than in each chapter separately for a more holistic view.

2.2.1 Modern Detectors

Modern object detectors employ Convolutional Neural Networks (CNNs) in order to jointly learn, in a supervised fashion, a feature representation and a model that predicts the class and image location of the objects of interest. Two different approaches to object detection with CNNs have gained attention: *two-stage detectors*, which first identify a set of candidate locations or Regions of Interest (RoIs) and *one-stage detectors* which make dense predictions at all spatial locations across the image.

The former commonly make use of a Region Proposal Network (RPN) to generate candidate object locations in the first stage. The RPN predicts class-agnostic ‘objectness’ scores at a dense set of locations and is designed to ensure high detection recall. In the second stage, the RoIs that have been proposed by the RPN are evaluated independently in order to classify the candidate as one of the classes of interest or background/no detection. Similar in spirit to a two level cascade, this approach improves accuracy of detection making two-stage detectors currently state of the art. It has been popularised by the series of papers on Regions with Convolutional Neural Networks (R-CNN) such as [Ren et al., 2017b], [Dai et al., 2016], [Cai et al., 2016], [Girshick, 2015] and [Girshick et al., 2014].

The second approach consists of one-stage detectors such as YOLO ([Redmon et al., 2016], [Redmon and Farhadi, 2016]) or SSD ([Liu et al., 2016], [Fu et al., 2017]) which are considered to be faster and simpler because they bypass the object proposal stage. They are implemented as fully convolutional networks which effectively compute the same function at all locations in an image. Currently, however, one-stage detectors are being outperformed by the two-stage approaches but they are occasionally preferred due to their significant computational speedup. The trade-off between the two approaches to CNN detection is nicely illustrated in [Redmon and Farhadi, 2018]. One reason for two-stage approach having better accuracy is the fact that RoIs take away from the classifier the option of making easy mistakes on image background which can be very difficult to model ([Lin et al., 2017]).

2.2.2 Sliding-Window Detectors

Prior to CNNs for object detection, the task commonly consisted of applying linear models on a manually designed non-linear feature representation of the image. Linear models are fast to apply even at a densely sampled set of locations on a grid dividing the image. Some examples of past state of the art detectors are [Dollar et al., 2014], [Benenson et al., 2013a], [Benenson et al., 2012], [Malisiewicz et al., 2011], [Dollar et al., 2012] and [Dollar et al., 2010].

The appeal of the sliding window approach is that there exists an efficient way of computing a scoring function f that linearly maps a feature map $\phi(\mathbf{x})$ of an image $\mathbf{x}$, to a detection score. This is efficiently computed as a convolution operation between $\phi(\mathbf{x})$ and the model weights $\mathbf{w}$ (the kernel to be convolved). The convolution operation computes the inner product between two signals $\mathbf{x}$ and $\mathbf{y}$ at all relative shifts as:

$$(\mathbf{x} * \mathbf{y})[u] = \sum_{v \in \mathcal{Z}} \mathbf{x}[v] \mathbf{y}[u - v], \quad u \in \mathcal{Z}. \quad (2.17)$$

The Convolution Theorem states that this operation in the spatial domain is equivalent to an element-wise multiplication in the frequency domain, which is easily parallelisable and can be exploited by a graphics processing unit (GPU).

However, while the speed is not the bottleneck of sliding window detectors, their performance ultimately comes from the ability to find a feature representation that is both invariant to different image transformation as well as highly discriminative of a particular class, such that an image patch is correctly classified. Within-class variation, pose variation, viewpoint variation, occlusions, truncations - all contribute to the difficulty of this task. Additionally, the subwindows presented to the classifier at test time may be overwhelmingly negative samples as the background class dominates the positive class which is prone to affect the learning process by biasing the model towards the dominating class.

2.2.3 Feature Representation

An appropriate feature representation of the input is therefore one of the most important choices in the design of a computer vision system. In recent years, manually designed image encodings such as Bag-of-Visual-Words (BoVW), Histogram of Oriented Gradients ([Dalal and Triggs, 2005]) or Aggregated Channel Features ([Dollár et al., 2014]) have been significantly outperformed by neural networks that acquire representations automatically from large labelled datasets ([Krizhevsky et al., 2012]).

In Chapters 3 and 4 we make use of the ACF feature representation detailed below. As they build upon HOG features, HOG are also described for completeness. At the end of the section, we comment on the features learnt by state of the art neural networks used in Chapters 4 and 5.

Histogram of Oriented Gradients (HOG)

One of the first successful and widely used visual representations is the Histogram of Oriented Gradients (HOG) introduced by [Dalal and Triggs, 2005] and winner of the 2006 Pascal Challenge ([Everingham et al., 2006]). HOG partitions the input image into pixel blocks (originally 8×8 cells, where a cell is comprised of multiple pixels) and computes a histogram of gradient orientations for each block. The blocks are contrast normalised which makes them invariant to changes in lighting. They are, however, unreliable when dealing with occlusions, truncations or different poses of the object as they seek to match rigid templates. It has been shown that ‘clean’ training samples are extremely important in obtaining good HOG features ([Zhu et al., 2016]). For objects whose appearance depends on the viewing angle, such as vehicles, multiple rigid templates are required for good performance (one for each view). Datasets with no class-specific viewpoint labelling might additionally require a label clustering step in order to obtain training examples to learn each intra-class template model. Deformable Part Models (DPMs) have been

introduced to deal with these shortcomings as they use HOG for individual object parts rather than the whole object ([Felzenszwalb et al., 2010b], [Felzenszwalb et al., 2010a], [Girshick et al., 2011]), making the features feasible for detection from all viewpoints and orientation of objects.

Aggregated Channel Features (ACF)

Channel features add to HOG more image channels obtained by various linear and non-linear transformations. One of the main benefits is that the features are computed efficiently as local sums and histograms over local rectangular regions. The multiple channels provide a richer source of input, while the summations are extremely fast to compute as they simply correspond to lookups in a summed-area table also known as an integral image. Instead of working with a regular grid like the HOG descriptor, ACF features can work with irregular cells that are selected during learning ([Benenson et al., 2013b]). This, as well as the additional feature channels (for instance, LUV color channels), explains the boost in performance. However, similar to HOG, ACF features still suffer from the problems of low-level rigid templates.

Deep Learning Representations (DL)

Starting with [Krizhevsky et al., 2012], deep learning features have shown a significant performance improvement over their hand designed alternatives, often referred to as ‘shallow’. Learnt in a supervised way using convolutional neural networks (CNNs), DL features currently dominate state of the art for a variety of tasks ([Razavian et al., 2014]) beyond image classification and detection. CNNs are typically composed of several layers of non-linear feature extractors with a capacity to learn a much more sophisticated structure in the data. They are directly trained to predict the outcome of a certain task (classification, segmentation, etc) in an end-to-end setting which improves the discrimination ability of the overall model.

Because of this setup, the boundary between the feature encoding and task-related model has become less clear. Capabilities such as colour, edge and orientation detection (encountered in HOG, ACF) are still present in the filters learnt by the early layers of the network. The filters learnt at the final layers of the network are more concerned with high-level semantics or concepts so they can deal a variety of poses and illumination conditions. From the point of view of a robotics platform operating in highly dynamic environmental conditions, the improved generalisation ability of deep learning representations greatly reduces the errors caused by changes in appearance. However, despite recent progress, even state of the art vision systems misbehave on the encounter of an out-of-sample input.

2.2.4 Non-Maximum Suppression

Non-Maximum Suppression (NMS) refers to the process of finding the best scoring window out of a set of overlapping candidate detections. Both sliding window and region of interest approaches produce high scores for multiple regions that overlap (with shifts of a couple of pixels or close scales) the true detection bounding boxes, as they share similar features. The candidate detections are sorted based on their scores, and starting with the lowest scoring ones, they are suppressed if there is a sufficient overlap with the higher scoring boxes. A local maximum believed to best cover that particular object instance is returned as the output of the NMS procedure.

2.2.5 Hard Negative Mining

Hard negative mining (HNM) refers to incorporating difficult negative examples in the training process, in order to converge to an optimal template model and bias faster. As the set of training samples not containing the object of interest highly exceeds the set of positive samples, it is important to find the subset of negatives that are the most relevant. HNM is applied to a labelled set of data (containing ground truth annotations of correct detections), over multiple iterations. Firstly, an

initial detector is trained using a random subset of windows. Then, at each iteration the detector is used in order to identify high scoring false positive detections and a new detector is trained on the initial dataset as well as the newly detected false positives (added as negative examples). Usually only the top scoring false positives are added. In support vector machine training, they can be interpreted as the support vectors. The detection performance is shown to improve with each iteration, although there is no guarantee that old hard negatives do not get discarded in subsequent iterations of the procedure. HNM is typically applied to limited labelled datasets. In robotics, where the distribution of the data a robot encounters changes rapidly, HNM has limited potential because it requires manually labelling the image data in order to identify hard negatives. HNM does not represent a viable solution for lifelong learning, an issue which we approach in Chapter 3.

2.3 Evaluation Metrics

Throughout this thesis we make use of evaluation metrics specific to image classification, object detection, decision making as well as introspectiveness of learnt models. The reader can skip this section and only come back to it when it is referenced in subsequent chapters.

2.3.1 Accuracy

In classification, two metrics are typically used to measure the task accuracy: precision and recall. Precision represents the proportion of true positive (TP) assignments out of all positive assignments:

$$precision = \frac{TP}{TP + FP}, \quad (2.18)$$

while recall represents the proportion of true positive assignments retrieved out of all relevant instances:

$$recall = \frac{TP}{TP + FN}, \quad (2.19)$$

where TP , FP and FN are true positives, false positives and false negatives respectively. A false positive error is also called a type I error while a false negative is called a type II error. The two metrics can also be used to evaluate an object detector by assigning its bounding box detections to ground truth boxes if they have enough overlap as measured by their Intersection over Union (IoU). The IoU threshold is commonly set to 0.5 unless otherwise specified.

Average precision (AP) computes the area under a precision-recall curve. To obtain this curve, assignments are ranked by their score and their precision is plotted as a function of recall, at every position in this ranking (by varying a threshold). AP is then the average value of $precision(recall)$, or $p(r)$ to simplify notation, where recall varies from 0 to 1:

$$AP = \int_0^1 p(r) dr. \quad (2.20)$$

Another aggregate metric for classification is the F_1 score, the harmonic mean of precision and recall:

$$F_1 = \left(\frac{precision^{-1} + recall^{-1}}{2} \right)^{-1}. \quad (2.21)$$

Detectors can additionally be evaluated using per-image statistics, for instance the Miss Rate (MR):

$$MR = \frac{FN}{TP + FP} \quad (2.22)$$

computed against the average number of false positives per image (FPPI). The FPPI is usually presented in the log-space (ranging from 10^{-2} to 10^0). Unassigned detection bounding boxes decrease the precision of the model and increase the FPPI, while unassigned ground truth boxes decrease the recall and increase the miss rate.

Breaking down scalar-summary statistics such as AP into precision and recall allows pointing out strengths and weaknesses of different models and choosing a trade-off that is appropriate for the task. For instance, false negative detections are dangerous in the context of self driving vehicles but not so in the context of injecting coral-eating starfish in the Great Barrier Reef ([Dayoub et al., 2015]), where false positive detections have more significant consequences.

2.3.2 Calibration

When measuring the quality of probabilistic output, one of the main measures of interest is *calibration*. Accuracy and calibration are orthogonal measures. For instance, in a classification setting where the output vector $f_w(\mathbf{x})$ is a categorical distribution over C output classes (where f_w^c indicates the probability assigned to class c), accuracy is concerned with the precision of the model’s class predictions $\hat{y}$, where

$$\hat{y} = \arg \max_c f_w^c(\mathbf{x}) \quad (2.23)$$

while calibration is concerned with the quality of the probability value assigned to that prediction: $f_w^{\hat{y}}(\mathbf{x}) = \max_c(f_w^c(\mathbf{x}))$. For instance, an introspective classifier assigning a probability value of 0.8 to its predictions is expected to be correct roughly 8 out of 10 times.

A frequentist approach to measuring calibration is by computing the difference between the confidence predicted by a model and the empirical frequencies of successful classifications made by that model. By assigning all output probabilities $f_w^{\hat{y}}(\mathbf{x})$ corresponding to the predicted classes $\hat{y}$ to K equally sized bins such that D_k

is the set of predictions for which $f_w^{\hat{y}}(\mathbf{x}) \in \left(\frac{k-1}{K}, \frac{k}{K}\right]$, we can compute the success frequency of the predictions in bin D_k :

$$\text{frequency}(D_k) = \frac{1}{\|D_k\|} \sum_{\mathbf{x}_i \in D_k} \mathbb{1}(\hat{y}_i = y_i), \quad (2.24)$$

where y_i are the true classes, and the confidence of the predictions in bin D_k :

$$\text{confidence}(D_k) = \frac{1}{\|D_k\|} \sum_{\mathbf{x}_i \in D_k} f_w^{\hat{y}}(\mathbf{x}) \quad (2.25)$$

This calibration measure has been introduced in [DeGroot and Fienberg, 1983]. We analyse it in Chapter 5 via calibration diagrams which plot the success frequency and confidence in each bin. They are a common choice for measuring the quality of predictive uncertainty used in works such as [Kendall and Gal, 2017], [Guo et al., 2017], [Gal et al., 2017] and [Ondruska and Posner, 2014]. We also report the Mean Squared Calibration Error (MSCE) computed as:

$$\text{MSCE} = \sum_{i=1}^K \frac{|D_i|}{N} |\text{freq}(D_i) - \text{conf}(D_i)|^2. \quad (2.26)$$

Maximum Calibration Error (MCE) is similar to MSCE but instead of the mean it computes the maximum error or the worst-case deviation between confidence and success frequency. The lower MSCE and MCE are, the better calibrated the model is.

Two Bayesian measures of uncertainty are the normalised negative log-likelihood (NLL) shown in Equation 2.5 and the entropy H of the prediction:

$$H(\mathbf{w}) = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^C f_w^j(\mathbf{x}_i) \log f_w^j(\mathbf{x}_i) \quad (2.27)$$

where y is the true class, N is the number of examples and C is the number of classes.

Another measure of interest, particularly for applications where refusing operation is preferable to an incorrect prediction, is *risk*. The risk of employing a

model amounts to the percentage of mistakes made for a particular coverage of the dataset. The coverage contains only the samples where the maximum confidence assigned by the model to any class is above a user-specified threshold. If the model has the option not to classify samples which it deems as uncertain, then the overall accuracy of the task can significantly increase by using only the samples with a prediction confidence greater than the specified threshold. We make use of this metric in a decision making framework in Chapter 4 as well in Chapter 5 in order to quantify the uncertainties assigned by different models.

2.4 Conclusion

This chapter has introduced the fundamental problem of object detection and some of the prerequisite knowledge for the remaining chapters. In Chapter 3 we aim to learn more accurate models while in Chapter 5 we aim to learn better calibrated models, both building upon Section 2.1. The models learnt are part of object detection frameworks that follow the two-stage paradigm described in Section 2.2.1 (Chapters 4 and 5) as well as more traditional approaches such as the sliding-window classifiers in Section 2.2.2 (Chapters 3 and 4). We also touch upon standard detection techniques such as hard negative mining (HNM) and non-maximum suppression (NMS) and introduce some of the evaluation metrics used in the following chapters such as accuracy, calibration and risk.

3

Improving Robot Perception Through Experience

Aspirations of lifelong perception and navigation in outdoor environments require autonomous agents to deal with great variation in scene appearance. Season, time of day and weather conditions can drastically change the appearance of a place and implicitly the performance of a robot executing a visual task in that workspace. The work presented in this chapter aims to improve object detectors by tailoring them to the exact workspace of the robot. Each place the robot traverses can benefit from an expert detector that, regardless of the performance anywhere else, excels in that particular place of operation.



Figure 3.1: The same place under various environmental conditions. Experiences encode both the location and the visual features of a place. Example figure produced by [Linegar et al., 2015].

3.1 The Experience Paradigm

The idea behind this work is inspired by the *experience paradigm* which captures different visual modes in the appearance of a robot’s environment. It was introduced under the framework of Experience-based Navigation ([Linegar, 2016], [Churchill and Newman, 2013a]), where *experiences* act as maps that enable visual feature matching between the robot’s live experience and a memory of the same place. The difference between standard maps and experiences is that the latter incorporate the visual appearance of a place such that if the same location is visually different then a new experience map is generated. Figure 3.1 shows an example of the same place under various environmental conditions, or experiences. Extensive research work in this direction has been conducted by [Linegar, 2016].

The robot’s performance on any visual task, be it feature matching or object detection, is highly influenced by the visual appearance of the robot’s workspace ([Peng et al., 2017]). The need for several diverse experiences is due to the fact that learnt models optimise for high performance on the data seen by the robot in the training phase. Unlike fixed datasets (or benchmarks), where the distribution of the training set is static, for practical application in mobile robotics the testing distribution cannot be guaranteed to match the training distribution. This is because with time, outdoor data are likely to change appearance, due to various factors (difficult to model) such as: weather conditions, illumination conditions, foliage or structural changes in the appearance of a place. Performing a visual task under sunny weather might require a different set of tools to performing it under heavy rain. The only way to operate long range and over large periods of time is by continuously learning and adapting to the changes encountered. There are two ways one can envision this happen: either by training a general model that we can trust to operate in all circumstances or by training a collection of models that each serve as a local expert.

The problem with the first approach is that it is difficult (if not impossible) to guarantee that a training set is complete and there will be no out-of-distribution data samples encountered in a future test scenario. Additionally, traditional machine learning models are known to ignore the tail the of data distribution in order to generalise better to new data ([Horn and Perona, 2017], [Wang et al., 2017]). So even if the training data contained all visual imagery a robot can encounter over its lifetime, the model learnt might be biased towards the high probability data and choose to ignore examples that have a low probability of occurring.

On the other hand, local experts focus on a single part of the space: the one that the robot is currently operating in. Inspired by the success of Experience-based Navigation (EBN) ([Churchill and Newman, 2013b], [Linegar et al., 2015] and [Paton et al., 2017]), we instead train local experts and compare this approach to training a single model.



Figure 3.2: Illustration of the EBC framework: we propose detection models that learn from the robot’s experience over multiple outings. Agents are often constrained in terms of their place of operation. We take advantage of this in order to improve experience-specific performance.

3.2 Motivation and Background

We introduce Experience-Based Classification (EBC) to tackle the object detection problem using insights from the Experience Paradigm introduced in Section 3.1. This work proposes local object detectors that act as experts in a certain environment (see Figure 3.2). We take advantage of the fact that, in robotics, physical systems are tied to their area of operation, whether it is autonomous vehicles operating on streets, service robots in homes or offices, or industrial robots in factories. The same workspaces will be traversed over and over again, therefore it seems practical

to tailor the visual representation to specific application domains. To narrow this representation even further, we propose that a certain detector model will not only be utilised in urban environments, but it will be tied to specific routes. The hypotheses we test in this chapter are the following:

- **Learning a single model for object detection is insufficient for robust performance everywhere.** As data encountered by the robot is likely to change over time, a model learnt using a finite number of samples might no longer be representative of the data the robot is experiencing. We experimentally show that a model learnt on all the data available is outperformed by a place-specific model when the latter is used in the right place.
- **Operation in dynamic environments implies the need for a robot to adapt over time.** We propose to make use of higher-level information, such as the place and time of operation, in order to decide when and where to learn (or use) new experts. In this chapter the place and time are manually selected; however, we envision learning to predict it automatically using a framework such as the one described in Chapter 4.
- **A robot should make effective use of its feedback from cycles of interaction with the world.** We believe that this is key to lifelong operation in changing environments - the ability to effortlessly improve with each experience. We show that the accuracy of a detector improves with each outing (up to a limit set by the learning model capacity).

In order to build place-specific detection experts we incorporate data that the robot encounters in the same, or a similar environment. In particular, we incorporate the mistakes made in the past into the learning procedure, such that if the same environment is traversed over and over again, past mistakes would not repeat. Doing so is typically achieved through *hard-negative mining* ([Felzenszwalb et al., 2010a], [Sung and Poggio, 1998], [Henriques et al., 2013], [Kanezaki et al., 2014]), a

technique described in more detail in Section 2.2.5. HNM consists of determining hard examples by performing multiple iterations over a set of labelled data and giving them a higher weight when learning the model. Retraining happens after every iteration until convergence. However, this is time-consuming and costly when applied to a robotics platform that regularly collects new data ([Maddern et al., 2017]) due to its requirement of manual annotations. Instead, the paradigm we propose explores geometric constraints in order to reason about detections in a self-supervised manner and continuously gather training samples in the real world without requiring additional ground truth labels.

Our work also share similarities with the self-supervised Track-Learn-Detect framework ([Kalal et al., 2012]) where constraints are introduced for online detector improvement. They introduce P - N learning: a P -expert estimates missed detections, while an N -expert estimates false alarms. Our geometric constraints are analogous to their N constraints.

3.3 Framework Description

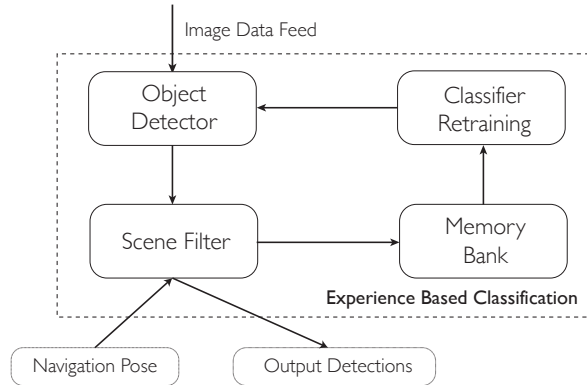


Figure 3.3: Proposed Experience-based Classification (EBC) framework. The detection process is followed by a scene filtering step (informed by a navigation feed typically available on a robotics platform). The scene filter produces false positive detections which can be stored in a memory bank for later retraining of a place-specific classifier. Since such detections scored high enough to pass the detection stage but did not satisfy scene constraints, they are considered to be *hard* examples for the classifier.

We augment the standard perception pipeline by introducing a scene filtering step, a memory bank of negative samples (place-specific hard examples) and a classifier retraining step. Our implementation of this system is depicted in Figure 3.3 and its components are:

- **An Object Detector:** The detector extracts image features from an incoming image feed and, in a sliding window fashion, uses a classifier to obtain a dense map of classification scores. If the bounding box detection score is a local maximum and survives a non-maximum suppression (NMS) procedure, it is deemed to be a positive detection and streamed down the pipeline.
- **A Scene Filter:** The raw bounding box detections are interpreted in the context of the scene, which adds additional constraints. We make use of these constraints to determine potential mistakes made by the detector and filter them out.
- **A Memory Bank:** A collection of false positive detections are gathered in order to be used for detector retraining.
- **A Retraining Step:** Detectors are retrained to make use of the data in the memory bank, which ties them to specific locations where the robot has travelled.

3.3.1 Object Detection

The object detector processes an image stream and produces detections in the form of bounding boxes. Each image is processed individually by first converting it into a dense feature map. As a feature descriptor we use Aggregated Channel Features (ACF) introduced by [Dollár et al., 2014] (see Section 2.2.3). At the time of this work, ACF features were the most commonly used features for pedestrian detection due to being highly discriminative and fast to compute. Current state of the art methods use convolutional neural networks to learn image features.

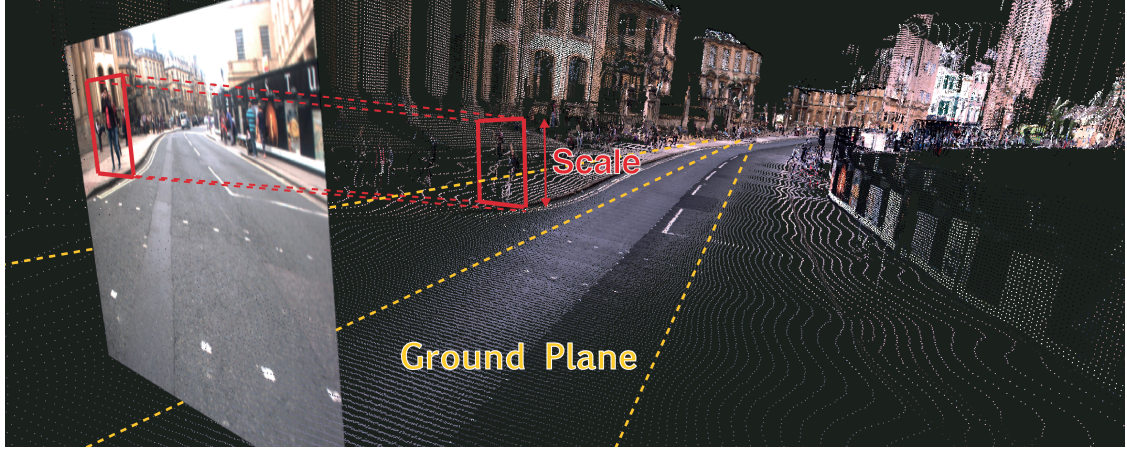


Figure 3.4: Figure illustrating the scene filtering step. Bounding boxes scored highly by the detector reach the second step of the EBC pipeline. They are processed by the scene filter using heuristics such as scale feasibility and horizon intersection. This step makes use of a scene prior map built at an apriori traversal of the environment and a pose feed that informs the robot of its location with respect to the scene prior.

However, even deep learning approaches have been shown to suffer from (training) dataset bias and could benefit from the EBC framework. We believe the proposed framework is agnostic to the detector.

After the feature computation step, we employ a sliding window approach to obtain classification scores as described in Section 2.2.2. Multi-scale detection is performed by resizing the image and repeating the process for multiple scales. After aggregating the detections for all desired set of scales, non-maximum suppression with an IoU of 0.5 is used to filter out overlapping detections. Finally, the output is a set of bounding boxes which correspond to subwindows that score above a detection threshold, which are deemed to be candidate positive detections.

3.3.2 Scene Filtering using Map Priors

The scene filter is the core component of the EBC framework as it enables finding and filtering out some of the candidate detections in a self-supervised way. Rather than abandoning geometry, we are leveraging the geometry of vision to provide feedback to object detectors that are solely based on machine learning techniques.

We assume that the robot operates in an environment previously mapped (either by the robot itself or by a survey vehicle) and a 3D prior map has been generated. This prior map, referred to as a *scene prior* is a set of map segments constructed using a visual odometry pose chain. The segments are locally metric 3D pointclouds collected using a lidar sensor. An example of a scene prior segment can be seen in Figure 3.4. Once the scene prior is built ([Stewart, 2014]), it is stored in a database together with other scene information which allows fast retrieval access. By finding the robot pose with respect to this prior map (e.g. [Stewart, 2014]), we can avoid online computation and simply retrieve scene information for the scene prior segment that the robot is currently in (typically covering a 30m distance length).

Specifically, the information we need to retrieve at test time is the ground plane location with respect to the robot in order to avoid online computation. To fit a ground plane model to the pointcloud generated by the robot’s planar laser scanner we use Maximum Likelihood Estimation SAmple Consensus (MLESAc) by [Torr and Zisserman, 2000]. The ground plane parameters $\hat{n}$ and d , where $\hat{n}$ is the plane unit normal and d the distance to the origin such that $x \cdot \hat{n} + d = 0$, are retrieved for every segment that the robot localises in. The ground plane together with the camera intrinsics, extrinsics and the robot’s current imagery are used in order to aid two heuristics for identifying false positives produced by the detector, corresponding to:

- detections that lie off the ground plane
- detections that despite being located on the ground plane do not have a viable scale

The application of the two heuristics is illustrated in Figure 3.4.

Ground Plane Intersection

The first type of detections to be rejected are straight-forward to identify by estimating a horizon line (in the image plane) and finding its position with respect

to the detection bounding boxes. If the detection box lies entirely above the horizon line, we simply discard the detection as it corresponds to an object that is above the ground plane. We first choose two points on the ground plane $\{[x_i, y_i, z_i]\}_{i \in \{1,2\}}$ such that they are far enough away from the robot in the forward direction (by setting $x_1 = 1000m$, $x_2 = 1000m$, $y_1 = 50m$ and $y_2 = -50m$).

Points $\{[x_i, y_i, z_i]\}_{i \in \{1,2\}}$ in the robot frame are then converted to $\{[x_i^c, y_i^c, z_i^c]\}_{i \in \{1,2\}}$ points in the camera frame using the extrinsic camera matrix. Using the intrinsics of the camera we then convert the camera points to the image coordinates $\{[u_i, v_i]\}_{i \in \{1,2\}}$:

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & \gamma & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i^c \\ y_i^c \\ z_i^c \end{bmatrix}, \quad (3.1)$$

where f_u and f_v are the focal lengths and the u_0 and v_0 are the principal point offsets. The horizon line follows the equation $y = mx + b$, where:

$$m = \frac{v_2 - v_1}{u_2 - u_1}, \quad b = v_2 - mu_2. \quad (3.2)$$

Given a detection bounding box defined by $[p_u, p_v, w, h]$, where p_u and p_v are the coordinates of the bottom left corner, we check if the point $[u', v']$ defined by $[u', v'] = [p_u + \frac{w}{2}, p_v]$ is above or below the horizon line. If

$$v' \geq mu' + b, \quad (3.3)$$

the point is below the horizon and the detection is accepted by the ground plane filter. Otherwise, it is deemed to be a false positive detection and stored in the memory bank.

Feasible Scale

To obtain the height of the detection we first project into 3D space the middle of the lower resp. the upper edge of the image bounding box and compute the euclidian distance between the two 3D points.

We first compute $L_c = [x^c, y^c, z^c]$, a line from the camera origin to a point $[u, v]$ in the image using the inverse of intrinsics matrix in Equation 3.1:

$$\begin{bmatrix} x^c \\ y^c \\ z^c \end{bmatrix} = \begin{bmatrix} f_u & \gamma & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \quad (3.4)$$

To then compute the 3D intersection point between line L_c and the ground plane, we convert the line coordinates into the robot reference frame using the extrinsics camera matrix. Given a line of equation $p = l_0 + dl$ and a plane $(p - p_0) \cdot \hat{n} = 0$, their intersection is computed by finding d :

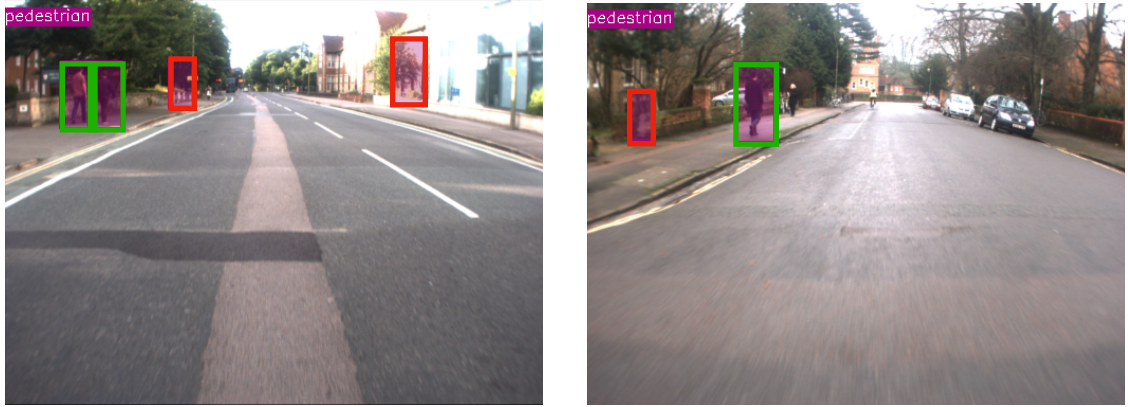
$$d = \frac{(p_0 - l_0) \cdot \hat{n}}{l \cdot \hat{n}}, \quad (3.5)$$

where p_0 is a point we choose on the plane, l is a vector in the direction of the line and l_0 is a point on the line.

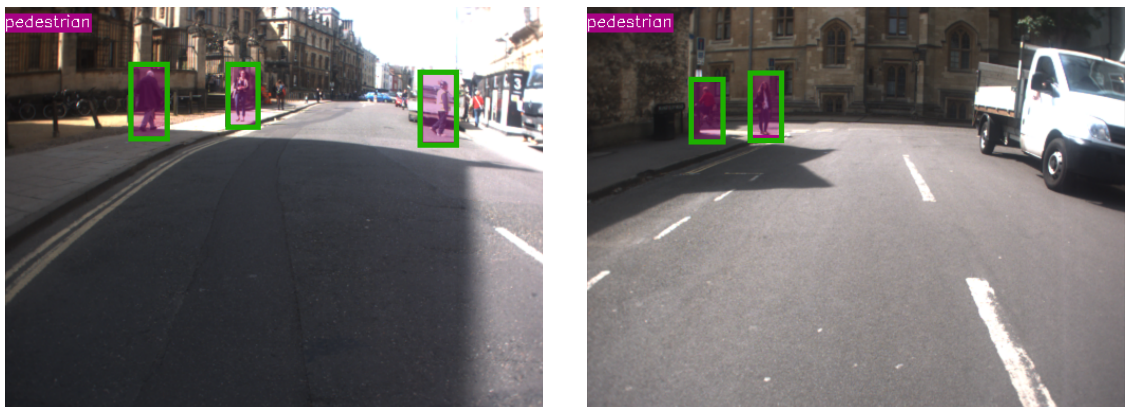
By finding d , we can compute two points on the ground plane corresponding to the bottom corners of the image bounding box. The two points and the ground plane normal $\hat{n}$ determine a plane, orthogonal to the ground plane, that contains the detection bounding box in 3D. We then compute the intersection of L_c with this plane for two image points: the middle of the lower resp. the upper edge of the bounding box in order to find their corresponding 3D estimates. The distance between the two points determine our candidate object's height (see Figure 3.4). Bounding boxes having heights that are infeasible (too tall or too short to be a 3D object of interest) are rejected.



(a) Detection bounding boxes that have the bottom edge above the horizon line are rejected and stored in the memory bank.



(b) The false positive detections (shown in red) do not have a feasible scale. They are rejected and stored in the memory bank by this step in the EBC pipeline.



(c) Example image where no detections are rejected by the scene filter.

Figure 3.5: Using EBC geometric constraints to filter out false positive detections. In green we show true positive detections and in red we show false positives. The orange line is the estimated horizon line which must not be below the bottom of any detection bounding box.

Figure 3.5 shows some example detections rejected according to the two heuristics as well as some successful detections. Accepted detections are passed on to the remainder of the perception pipeline, while the rejected ones are stored in the memory bank. We employ a conservative scene filtering in order to avoid false negative samples being added to the memory bank as this can lead to semantic drift ([Curran et al., 2007]).

We are able to run the detector and scene filter in real time (20Hz) while the robot is operating and require relatively limited resources in order to retrain the detector models (offline).

3.3.3 Memory Bank and Model Retraining

By applying the scene filtering step, EBC augments the original training set with the rejected detection samples. These false positive detections are incorporated as negative examples during the retraining procedure, after the robot has finished operating. Since the samples were detections that scored highly in a certain environment but were filtered out as false positives, they will act as hard negatives during retraining. In each subsequent training cycle the detector doesn't only improve performance by learning from its previous mistakes, but it also adapts the classifier to the specific environment where the false positives were detected.

We note that as data streams gathered from mobile robotic platforms tend to be spatially and temporally correlated, it is best to subsample the detections from the memory bank (for instance, by taking a limited number of false positives per image or discarding consecutive images). This is because most classifiers assume independent, identically distributed data. To do so we use a fixed number for the top false positives per frame and apply NMS to the detections.

The initial set of negative samples is not preserved as a separate term but rather combined with the set of hard negatives. All the negative samples (the initial set and place-specific hard negatives from multiple iterations of this procedure)

are weighted equally by the retraining algorithm. However, since over time the set of negative examples will expand, we do make sure that the classifier loss appropriately weights the two classes. In our case, the loss to be minimised in the classifier retraining step is:

$$\begin{aligned} \mathcal{L}(\mathbf{w}) = & \alpha \frac{1}{|\mathcal{D}^+|} \sum_{(\mathbf{x}, y) \in \mathcal{D}^+} \max(0, 1 - y f_w(\mathbf{x})) + \\ & (1 - \alpha) \frac{1}{|\mathcal{D}^-|} \sum_{(\mathbf{x}, y) \in \mathcal{D}^-} \max(0, 1 - y f_w(\mathbf{x})) + \frac{1}{2} \lambda \|\mathbf{w}\|^2 \end{aligned} \quad (3.6)$$

where f_w is the classifier function $f_w(x) = \mathbf{w}^T \mathbf{x}$, $(\mathbf{x}, y) \in \mathcal{D}^+$ and $(\mathbf{x}, y) \in \mathcal{D}^-$ the ACF features $\mathbf{x}$ and binary classification labels y of the positive respectively negative samples, λ the regularisation parameter and α a weighting of the two classes.

3.4 Experimental Evaluation

3.4.1 Sensor Suite

The platform used for experiments (shown in Figure 3.6) is equipped with:

- Point Grey Research (PGR) Bumblebee2 Stereo Camera: imagery used for visual object detection, robot localisation in the scene prior, pose chain generation using visual odometry (captured at 20Hz);
- SICK LMS-151 2D Lidar: planar scans used to generate a 3D poincloud scene prior (captured at 50Hz);
- Oxford Technical Solutions (OxTS) RT 3042 Dual-Antenna INS: used for georeferencing of data with UTM coordinates (captured at 100Hz);
- Point Grey Research (PGR) Ladybug2 Omnidirectional Camera: 360° imagery used to colour scene prior maps (captured at 10Hz);



Figure 3.6: The Wildcat vehicle (left) used to gather the image data, and a map (right) depicting the routes for our datasets, where we gathered images in January, May, and August. The difference in time of year provided seasonal variation, which affects the visual appearance of the scene. The two North Oxford routes, illustrated in blue, provided variation in season but same physical location, and the Central Oxford route, depicted in red, provided a difference in location.

3.4.2 Baseline Detection Model

Our baseline pedestrian classifier is a linear SVM trained using dual coordinate descent on the INRIA Pedestrian Dataset ([Dalal and Triggs, 2005]) with Aggregate Channel Features (ACF). We additionally performed 10-fold cross validation on a training set consisting of 1237 cropped positive pedestrian samples, and 12180 randomly sampled negatives (10 windows per negative image). To ensure the quality of this baseline detector we performed a bootstrapping step consisting of ten consecutive cycles of HNM using the INRIA negative images (images not containing the class of interest). In each HNM cycle the classifier was presented with 10000 random negative cropped samples extracted from the negative images. Misclassified hard negatives were saved and used as additional training data to retrain the classifier. All training is performed using the open source Liblinear library ([Fan et al., 2008]).

Route	Info	Train 1	Train 2	Train 3	Train 4	Train Total	Test
North Oxford January	Distance (km)	2.60	2.01	1.92	1.92	8.45	1.99
	Image frames	27.66	12.38	41.31	38.69	6.19	9155
	Time (min)	27.66	12.38	41.31	38.69	6.19	7.63
North Oxford May	Distance (km)	1.43	1.95	1.01	1.01	5.40	–
	Image frames	6066	8676	4001	3977	22720	–
	Time (min)	5.06	7.23	3.33	3.31	18.9	–
Central Oxford August	Distance (km)	6.91	6.79	6.68	6.56	26.94	–
	Image frames	36472	27720	27607	23463	115262	–
	Time (min)	30.4	23.1	23.0	19.6	96.1	–

Table 3.1: A summary of the datasets used for training and testing. We use two different routes in Oxford for spatial variation and three different seasons for visual variation.

3.4.3 Datasets

To show the impact of environmental variation and to evaluate our self-supervised learning approach, we used twelve different urban driving training datasets gathered using the Bumblebee2 stereo camera mounted on a Wildcat vehicle shown in Figure 3.6 (left) driving around Oxford. These unlabelled datasets were gathered from two different routes from successive outings at different times of year. We allocated these datasets to three categories based on the route and time of year (season), with 4 training datasets per category. These categories are referred to throughout the rest of this chapter as **North Oxford January**, **North Oxford May** and **Central Oxford August**. A map of the routes is provided in Figure 3.6 (right). The North Oxford and Central Oxford routes have no overlap. The detection was performed on the left Bumblebee2 stereo image only. For evaluation, we have manually labelled a test dataset from the **North Oxford January** category. The experiments therefore use a total of 40km of unlabelled training data for model learning and 2km of labelled data for testing. The datasets are summarised in Table 3.1.

3.4.4 Environment-Specific Tuning

Our goal is to investigate the effects of tuning models to specific environments. Figure 3.7 shows the precision-recall, as well as the miss rate and false positives

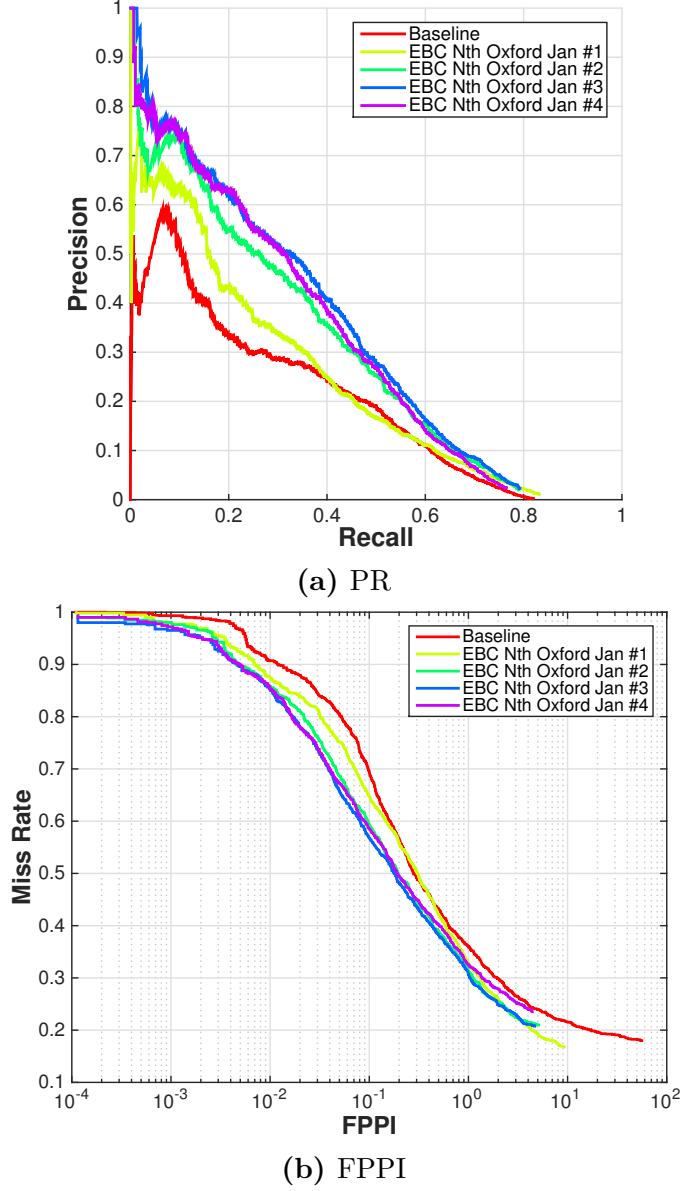


Figure 3.7: The precision-recall (3.7a) and miss rate-false positives per image (3.7b) for multiple EBC cycles on the North Oxford January training dataset. All plots show an increase in performance on the North Oxford January test set. The PR curve performance increases with the first three datasets observed, moving to the top right corner of the graph. This then settles with a very slight performance drop on the fourth round. The same trend is visible in the MR-FPPI graph with the curves moving to the lower left corner.

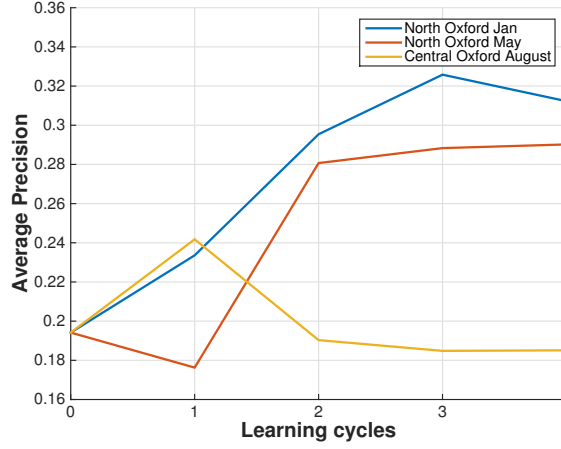


Figure 3.8: The average precision (obtained by computing the area under a PR curve) for three detectors trained using EBC, evaluated on the North Oxford January test dataset. Learning from operating on the same route (North Oxford) improves performance over the baseline detector, with the detector shown data from the same season as the evaluation set performing the best (North Oxford January). The detector which learned from operation on a different route as well as time of the year (Central Oxford August) does not improve performance.

per image for four training iterations on the North Oxford January dataset. The evaluation is performed on the North Oxford January test set and shows that indeed we are able to improve the performance of a detector using the self-supervised EBC framework. The baseline model is trained on the INRIA dataset and described in more detail in Section 3.4.2. The model is then retrained for the future using the false positives that it encountered at test time. Both PR and MR-FPPI curves show an increase in performance for the first three iterations. We attribute the diminished gains in performance after that to either model capacity or error sources not modelled by the EBC framework. The latter are addressed in Section 3.5.

EBC models

The three categories the data are divided into represent different perceptual experiences and each will train its own specific classifier with EBC. We start by training three different models from the same baseline model, one per category, and refer to by their category name: North Oxford January, North Oxford May and

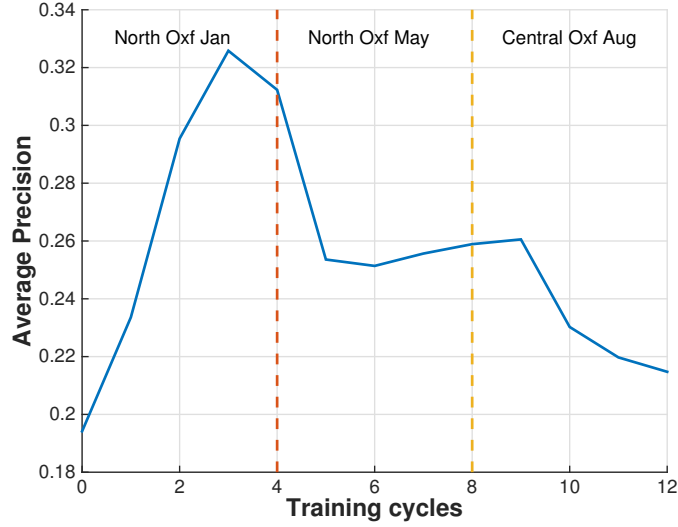


Figure 3.9: Average precision displayed for every dataset added to the training procedure. Starting from a baseline model, North Oxford January data is added, followed by North Oxford May and finally Central Oxford August. The evaluation is performed on the North Oxford test dataset. Drop in performance as we add more data from dissimilar environments, changing season in May and changing both season and route in Central Oxford.

Central Oxford August. Each model is evaluated against the separate test dataset from North Oxford January. Figure 3.8 shows the average precision of the evaluation of the three models, with North Oxford January performing better than both North Oxford May and Central Oxford August. We see that both detectors trained in the same place (North Oxford) improve notably in performance, whereas the detector trained in a different place (Central Oxford) does not. This indicates that location could be a more important factor than season. However, given that the performance increase is higher in January than it is in May for the detectors trained in North Oxford, we speculate that there are some perceptual differences between the two datasets and acknowledge that season also has an effect on the model performance.

EBC vs. Single Model

To confirm the value of experience-specific classifiers, we also investigate the effects of simply using all of the available data to train a single model, with the twelve datasets processed in temporal order: first January, followed by May, followed

by August. The results in Figure 3.9 show that this model performs worse than the model trained only on data from the same place and season as the evaluation data, with performance degrading substantially as dissimilar data are observed and incorporated into the learning procedure. More specifically, the average precision decreases when May data are added and continues to do so for August data until the average precision shows a very small improvement over the baseline (the model before starting the EBC training procedure). This result adds further weight to the argument for training place-dependent models rather than incorporating all the data into a single model.

Final Detector

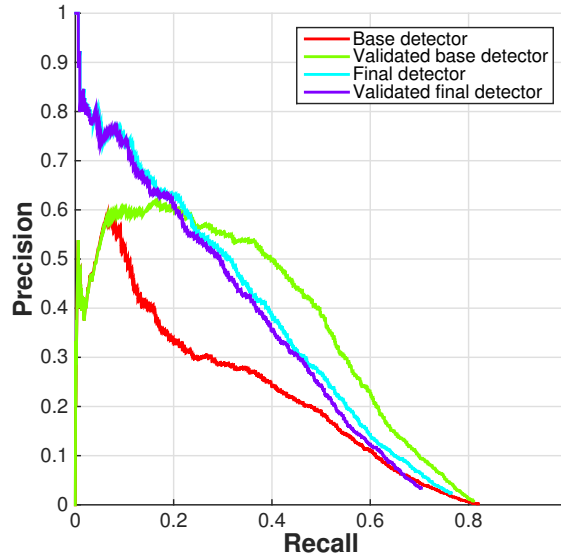


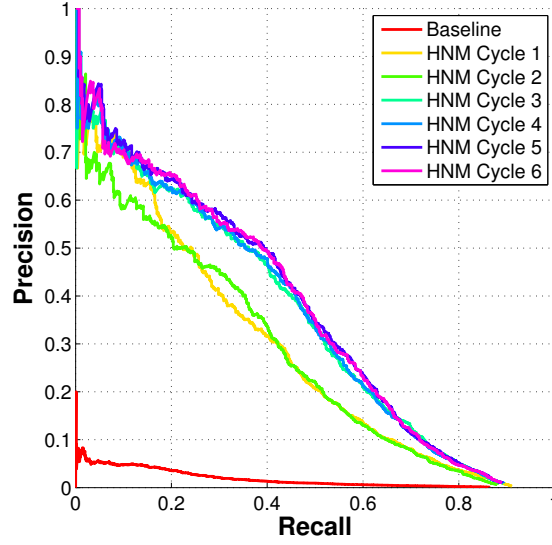
Figure 3.10: Figure shows the performance increase provided by the scene filter (referred to as ‘validated’ detector) when applied to both the base detector and the final EBC detector on the North Oxford January test dataset.

We note that we have only showed the raw detector performance in our experimental trials. Since the scene filter is already incorporated into the EBC framework, we can also validate our detections while running online if the 3D scene prior and localisation information are available. Figure 3.10 shows the precision and recall performance of a baseline detector and an EBC final detector. The additional

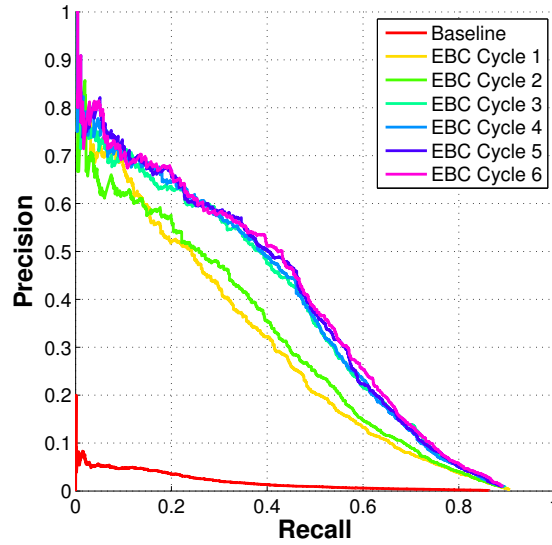
two curves show the detectors using the scene filter. We refer to a detector used together with the scene filter as ‘validated’. The baseline detector significantly improves when validated, while our final detector has shows only a very small improvement. We attribute this to the fact that the EBC detector has learnt during training from all the samples that the scene filter is able to invalidate, such that towards the final EBC training iterations the scene filter has diminishing returns. We note that a validated detector is more expensive to use as it requires additional systems running (localisation and scene filter) so in an online setting we would prefer to use a single detector - the EBC detector.

3.4.5 Comparison with Hard Negative Mining

Finally, we include an experiment showing that EBC is empirically equivalent to traditional HNM which requires manually labelled data. This is a stand-alone experiment we performed before training place-specific models in order to verify that the scene prior filter provides high quality false positive detections to be incorporated into the retraining process. The setup is similar to the experiments presented in the rest of the chapter: a baseline model is trained on a standard pedestrian dataset and improved in subsequent retraining iterations using data from the robot’s environment. As before, the dataset used in this experiment is the labelled North Oxford January dataset. Since HNM requires ground truth labels, we split the test set into two parts, with 70% allocated for training and 30% for testing. The baseline model used is a linear SVM trained on HOG features using the Liblinear ([Fan et al., 2008]) and OpenCV ([Bradski, 2000]) libraries on the Daimler Pedestrian Benchmark [Keller et al., 2011]. The training set consisting of 52112 positive and 32465 negative samples. We perform a ten-fold cross validation for SVM hyperparameter selection but not additional hard negative mining on the Daimler dataset to improve the model baseline. This is because the goal of this experiment was to asses how the EBC procedure compares to HNM when incorporating data from a new distribution



(a) Hard Negative Mining



(b) Experience-Based Classification

Figure 3.11: Precision-Recall curves comparing HNM to EBC iterations for a portion of the North Oxford January dataset. There is a dramatic improvement after the first iteration, and similar performance from both approaches. This demonstrates the value of environment-specific tuning and that EBC is a practical alternative to HNM.

and preferred that both models are assessed before incorporating HNM examples. Figure 3.11 shows six training iterations performed for each method. The behaviour of the two frameworks is similar, with the first two cycles significantly boosting

the precision and recall and the last cycle showing no increase in performance. This experiment empirically demonstrates that EBC is a practical alternative to using manually labelled data in a HNM procedure without any performance drop introduced by using a self-supervised signal.

3.5 Discussion

3.5.1 Knowing When to Retrain

While our experiments support the idea of experience-specific models, our proposed EBC framework raises questions around the necessary spatial and temporal resolution of experiences for model retraining. We have considered a small set of possible places and driving conditions but the number of distinct *experiences* a robot can encounter may significantly increase the number of models required to extend the EBC framework. Knowing when a model no longer fits the distribution of the test data and requires retraining is a research topic on its own. One way we envision this happening is by training a new model whenever a localisation system such as [Linegar et al., 2015] identifies a new experience (a change in the data distribution). One downside of this approach is that what a localisation task considers visually different and classifies as a new experience might not coincide with what an object detector would, unless they share a common image feature representation. It is very likely that in addition to weather conditions, season and route, environmental changes such as traffic or structural appearance of a place will play a bigger role for object detection.

3.5.2 Sources of Error

Our memory bank is currently limited to detections that get filtered out by two relatively conservative filters. However, failure modes of object detectors are diverse.

Firstly, if we were to categorise false detections into false positives and false negatives, only the former are incorporated into the model retraining. This is reflected in the fact that EBC can only increase model precision but not recall. False negatives could be a rich source of information if properly mined. This can be done, for instance, by using temporal cues or additional sensing modalities to find inconsistencies between detections. We can identify missed detection in a similar way to [Ramanagopal et al., 2017] and add them to the memory bank as positive samples.

Secondly, we can categorise detection errors into catastrophic and non-catastrophic errors. The latter are mistakes caused by the operating point we choose in order to classify them as such. For instance, if the detected object is small, occluded or truncated the score assigned to the detection box is typically lower compared to when the object has a better resolution. Setting a hard threshold for deciding when a detection box is a true positive can introduce non-catastrophic errors that could have been avoided by using a lower threshold. Such errors can be an additional source of information in the memory bank.

3.6 Conclusions

We have introduced a framework that provides a self-supervised, environment-dependent improvement of object detectors. EBC is particularly suitable in an urban driving scenario, where the robot is limited to certain roads that it traverses repeatedly and where geometry offers a strong prior on the location of objects of interest. Our experiments show that leveraging scene constraints is just as effective at improving the accuracy of a detector as hard negative mining, a technique which requires significant image labelling effort. We also show that the performance of object detectors varies as the robot experiences different environments and that detectors benefit from being tailored to their place of operation. EBC thus serves two purposes: replacing the expensive HNM from the learning

pipeline and exploiting environmental feedback in order to train detectors that continually improve with experience.

4

Predicting the Performance of Object Detectors

In the previous chapter, we have shown detection performance to be environment-dependent. While good in some places of operation, the same detector can result in failure once the environment or its appearance change due to weather conditions, time of day, illumination, structural changes, or anything that alters the visual appearance of a place. The shift in the data distribution which has not been accounted for at training time causes the learned model to generalise poorly to the data that it is about to encounter. The work in this chapter aims to predict this failure in advance, based on place-specific past experience, which can not only minimise the number of incidents, but also increase the user's trust in the system. Unlike in the previous chapter where we train a collection of place-specific models,

in this chapter we fix the object detector and estimate its performance over space and time in order to estimate *when* and *where* we can trust its predictions.

4.1 Motivation

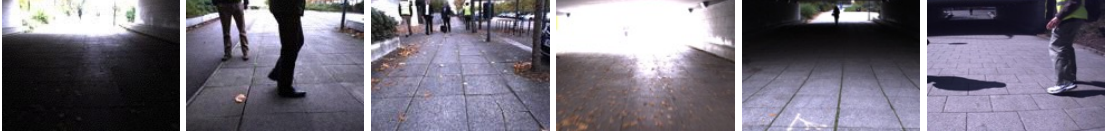
With machine learning models trained and tested on static datasets, it seems impossible to formally verify before deployment that they will behave as expected on novel scenarios at test time, when deployed on a robotics platform. Additionally, it is likely that lower-level mistakes of a perception module propagate to the higher-level decision making procedures of the autonomous system. Without any checks on the reliability of the information propagated, the safety of the robot and its surroundings would be compromised. This is why we propose to analyse and predict how robustly a perception system will function at any given moment in time.

The problem of predicting perception failure in action has not yet received the attention it deserves from the robotics community. The majority of effort is being devoted to building high-performance perception systems (e.g. [Cai et al., 2016], [Ren et al., 2017a], [He et al., 2016], [Badrinarayanan et al., 2015]) whose accuracy is measured on static benchmarks, but can have an unreliable behaviour when deployed in the real world. Additionally, a single estimate of performance can be noisy, as a result of the variance in a model that has a large number of parameters. We propose to aggregate multiple samples from the model over time and allow the perception module to flag up a warning when it believes it is about to make a mistake.

This work treats a generic object detector as a black-box and investigates the effect of changing operational conditions on its performance during multiple test runs of the same route. In particular, we are interested in discovering how many perception mistakes can be avoided if we reduce the duty cycle of the detector - rather than relying on it on all the image frames, we do so only on a select few where we are confident it can operate reliably. This setup simulates the idea of control being shared between the autonomous system and the humanoperator and it falls under



(a) Example data on which the robot decides that it is safe to operate autonomously



(b) Example data on which the robot can ask to switch the control back to a human.

Figure 4.1: Example data encountered by a robot as it traverses an urban environment in the proximity of pedestrians, cyclists and other road users. On some sections of the road on which it believes its perception system is underperforming, the robot can ask to switch control back to a human operator. Alternatively it continues to operate autonomously.

the *shared autonomy* paradigm, in which the robot is able to seamlessly take control of the vehicle and give operation back to the driver as a function of the environment. To achieve this, the robot must also fulfil the *autonomy on offer* paradigm, in which the robot characterises its own performance and determines whether or not it is confident in its abilities and hence capable of autonomous operation.

Our proposed framework is kept separate from the perception system itself and is able to predict performance before the actual perception task has been executed. It can be seen as a favourable alternative to employing the uncertainty measure typically associated with a learning algorithm in order to assess how much trust can be placed on the detector's predictions. Intrinsic uncertainty measures are often erroneous in practice, particularly when the algorithm is presented with out of sample data or when the algorithm itself is not inherently probabilistic ([Grimmett et al., 2016]). We propose to show the usefulness of our reliability measure in the context of autonomy on offer, in which the robot has to decide when to ask for help. Some example decisions can be seen in Figure 4.1. While in some cases the reasons for failing at a certain visual task are apparent - such as overexposure or underexposure of a significant part of the input image - in other cases, more analysis is needed in order to understand the underlying causes of failure. Our work is

unique in that we explicitly consider the location of the system when predicting perception performance. This means that we can predict likely failures in advance, even prior to commencing a route, and thereby highlight difficult sections of the proposed route by leveraging place-dependent cues. This is crucial, as research shows that it can take up to 15s for an operator to resume control, and up to 40s to stabilise control of the vehicle ([Merat et al., 2014]). As a result, we envision such introspection techniques to become an integral part of fleet scheduling systems for autonomous vehicles in the future. Our second approach also considers similarity of appearance between the current and previous traversals (at the expense of this ability to predict this far in advance), which ensures that prior observations that differ greatly from the current frame due to, for example, vastly different lighting or weather conditions, are removed from consideration in the prediction model. The key contributions of this chapter are:

- the concept of **performance records**: a framework which incorporates place-specific performance estimates gathered over time, in order to allow the robot at test time to estimate the likelihood of making a mistake;
- two approaches to building performance records, one of which makes use of the visual appearance of a place; and
- a view of autonomy-on-offer as a decision-making problem which provides motivation and a use-case for performance records, and allows the robot to optimally choose to offer/deny autonomy on over 60km of driving data in an urban environment.

4.2 Related Work

This lack of reliability of a perception module has also been observed by [Peynot et al., 2009] and [Peynot et al., 2010], who attribute it to sensor data integrity and analyse the effects of challenging operational conditions on the perceptual

integrity of the robot. [Hoiem et al., 2012] aim to understand and distinguish between the different failure modes of an object detector. They too acknowledge that the quality of the input plays a significant role on the outcome and show how different object characteristics correlate with the performance score. Similar problems are also reported for localisation performance. [Churchill et al., 2015] and [Dequaire et al., 2016] propose to embed spatial models of expected localiser performance in localisation maps in order to aid trajectory planners. There are several works in the literature that address the fluctuating performance levels of a machine learning system induced by a visual shift in the data. This is often due to training set bias, which can be dangerous as the resulting models generalise poorly across place and time, meaning that they may not respond well to new testing conditions. The works of [Torralba and Efros, 2011] and [Khosla et al., 2012] identified early on that the mismatch between the training and test data distributions is a major cause of poor model performance, and as a result, they advocate the use of cross-dataset evaluations.

Additionally, our experiments in Chapter 3 show the sensitivity of object detectors to factors such as weather and location, by training local experts through incorporating place specific hard negative examples in the training procedure. When generic negative training data are replaced with the detector’s mistakes, we are able to significantly improve detection results by specifically targeting areas of improvement for each location. This idea of adapting a model to the data it is most likely to observe during operation has first been proposed in the context of object detection through the process of hard negative mining ([Felzenszwalb et al., 2010a]), which can be interpreted as incorporating information about the testing data distribution into the training procedure. Similarly, in the case of training neural networks for object detection, the process of fine-tuning adjusts the model weights to be more representative of a specific set of data (e.g. [Yosinski et al., 2014]). The problem of dataset bias has also been addressed by the computer vision community through *domain adaptation*, learning a mapping between two different

domains such that a model trained on one domain performs just as well on the other when used in combination with the learned mapping ([Kulis et al., 2011]).

With the advent of deep learning techniques and subsequently, deep domain adaptation, the effort has been devoted to learning features that are invariant to this change of domain (e.g. [Ganin and Lempitsky, 2015]). In this scenario, the adaptation happens directly in the training process, where the model is penalised if it learns untransferable features that are biased towards one domain. This can be implemented under the recently proposed framework of adversarial training, whereby an additional adversarial network is trained to discriminate between the two domains given the internal representations of the first network. The original network is then also trained to fool the adversary (i.e. by adding the negative of the discriminator loss), thereby encouraging it to learn representations that are domain-invariant (e.g. [Tzeng et al., 2016]). While these methods look to adapt a model to new environments, our work instead focuses on the critical ability to characterise and predict the performance of a perception system in new conditions or environments.

4.2.1 Relation to Introspection

This higher-level characterisation of when and where an algorithm fails is similar in spirit to the concept of *introspection* introduced in [Grimmett et al., 2016]. In that work, the authors looked at the introspective capacity of different classification frameworks, which refers to a classifier’s ability to assign an appropriate measure of confidence to any test data. Mistakes are not considered catastrophic when they are made with high uncertainty as this gives the system the ability to ask for help and correct itself. [McAllister et al., 2017] also looked at model uncertainty and proposed the use of Bayesian deep learning in order to increase autonomously vehicle safety. This assessment of trust or confidence has been widely employed by the active learning community in order to guide the data selection process ([Kapoor et al., 2010], [Holub et al., 2008]). It has also been examined in the face verification domain

through standardised image quality metrics and performance assessment systems. For instance, in [Dutta et al., 2015] the authors present a generative model that maps image quality (including facial pose, illumination) to verification performance, without analysing the distribution of the classifier’s scores or its reported uncertainty. [Gurari et al., 2016] propose similar ideas for the task of predicting the accuracy of a segmentation algorithm: they use a linear regressor to predict the Jaccard index (the fraction of pixels that are common to the segment and ground truth) based on simple features characterising the shape and geometry of the segmented region.

A key benefit of our framework is that it is independent of the classification algorithm and its associated uncertainty measure. It bears some similarity with [Zhang et al., 2014], which introduces ALERT, a system used to predict the accuracy of a computer vision system on various tasks. The authors propose a regression model in order to predict task performance directly from the input data and show that incorporating these predictions can help improve the performance of a downstream application. Our work is also similar to that of [Daftry et al., 2016], in which the authors propose a convolutional neural network (CNN) architecture that uses the current image feed (and the resulting optical flow images) to predict the performance of a trajectory planner on a micro-aerial vehicle (MAV). They too motivate their work in the context of *introspection*, and also argue that performance prediction should be modelled directly as a function of the input data, so that it is independent of the system it is characterising.

We share with these methods an aspiration to reliably flag a warning when we believe that the input data (in our case, camera images) do not have the characteristics required for high performance and the robot is more likely to fail at the vision task it has to perform. However, our work stands apart from that of [Zhang et al., 2014] and [Daftry et al., 2016] as our approach is tailored specifically to robot perception and explicitly exploits location and past experiences of the robot in that place of operation. Given that autonomous vehicles are likely to traverse the same routes repeatedly, these experiences provide useful contextual

information which could guide the robot’s future decision-making process. To the best of our knowledge, this is the the first approach to predict the performance of a perception system as a function of appearance, space, and time.

4.3 Framework Description

Given that robots often perform repeat traversals of the same workspaces, as evidenced in [Furgale and Barfoot, 2010], [McManus et al., 2013], [Paton et al., 2017], and [Linegar et al., 2015], we propose to make use of the robot’s past experience in order to estimate its capabilities. In the context of autonomous vehicles, this is not an unusual problem setting, as they rarely operate in completely unknown environments. If a robot has traversed a route in the past, then we would like to leverage its past experience in order to predict performance in subsequent visits of the same place. Our work is based on the assumption that the same physical location, under similar driving conditions, leads to a similar perception outcome. Figure 4.2 illustrates this setup: we traverse the same route multiple times and gather performance estimates along it in order to build the performance records. At test time, we utilise the records to predict how the system will perform on the current traversal. To evaluate our system, we further employ the predictions made with the purpose of deciding on whether the robot should offer or deny autonomy to the driver. The performance that we want to estimate is that of an image-based pedestrian detector (more details in Section 4.4) but the idea can be extended to any learning algorithm whose performance varies with operational conditions. We describe our approach for estimating detection performance at a particular location in Section 4.3.1 and formulate offering/denying autonomy as a decision making problem in Section 4.3.2.

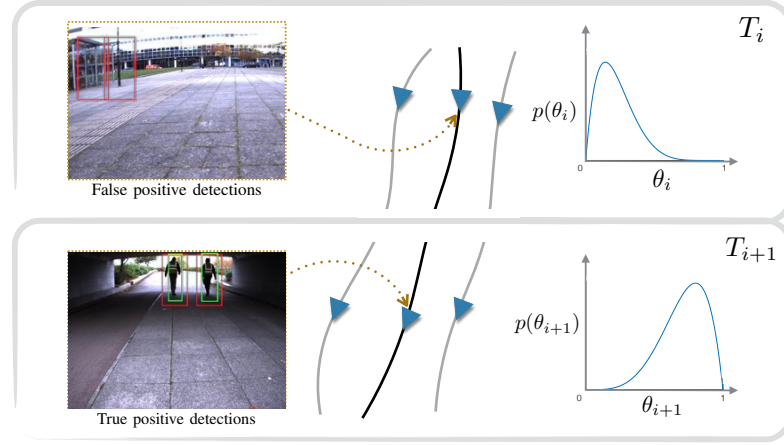


Figure 4.2: Overview of the proposed method. A new traversal (black line) of a route which has been travelled previously (grey lines) can make use of past estimates of detection performance. For instance, at Location A where we have repeatedly observed false positive detections, the performance record yields a low probability of success for the detector, while at Location B, where the detector has only produced true positive detections, the probability of success is very high.

4.3.1 Estimating Perception Performance

For a traversal $\mathcal{T} = \{\ell_1, \ell_2, \dots, \ell_N\}$ of a route, we denote as ℓ_i the i^{th} location along it. We keep track of performance at every location ℓ_i by modelling the detections at that location as events. While true positive detections indicate the success of the detection system, the false positives and false negatives indicate failure, so we record observations $x_i^j \in \{0, 1\}$ such that:

$$x_i^j = \begin{cases} 1, & \text{if the } j^{\text{th}} \text{ observation at } \ell_i \text{ is a true positive} \\ 0, & \text{if the } j^{\text{th}} \text{ observation at} \\ & \ell_i \text{ is a false positive or a} \\ & \text{false negative} \end{cases} \quad (4.1)$$

We let the observations x be modelled by a Bernoulli random variable: $x \sim \text{Ber}(\theta)$ with probability mass function:

$$f(x; \theta) = \theta^x (1 - \theta)^{1-x}, \quad x \in \{0, 1\}. \quad (4.2)$$

The parameter θ can be thought of as the probability of success of the detection system. We make the assumption that the set of observations $\mathcal{X}_i = \{x_i^1, x_i^2, \dots, x_i^{n_i}\}$ are conditionally independent given θ_i , and we explicitly condition each observation x_i on θ_i to express the likelihood of successful performance for a particular location ℓ_i as:

$$p(\mathcal{X}_i | \theta_i) = \prod_{j=1}^{n_i} p(x_i^j | \theta_i) = \theta_i^{k_i} (1 - \theta_i)^{n_i - k_i}, \quad (4.3)$$

where k_i represents the number of observations indicating good performance ($x_i = 1$) out of a total of n_i observations at location ℓ_i along the route.

Using Bayes Theorem, we calculate the probability of the detector being successful at location ℓ_i as:

$$p(\theta_i | \mathcal{X}_i) = \frac{p(\mathcal{X}_i | \theta_i) p(\theta_i)}{\int_{\theta_i} p(\mathcal{X}_i | \theta_i) p(\theta_i)}. \quad (4.4)$$

We represent the prior on the probability of success of the detector $p(\theta_i)$ as a Beta density of the form

$$p(\theta_i; \alpha, \beta) = \frac{1}{B(\alpha, \beta)} \theta_i^{\alpha-1} (1 - \theta_i)^{\beta-1}, \quad 0 \leq \theta_i \leq 1, \quad (4.5)$$

where $\alpha > 0$, $\beta > 0$ and $B(\alpha, \beta)$ is the Beta function. Our canonical prior at a new location that we see for the first time and where we have no knowledge of the success of the detector is given by $\alpha = 1, \beta = 1$.

Since the Beta distribution is a conjugate prior to the Bernoulli distribution, the posterior $p(\theta_i | \mathcal{X}_i)$ is also a Beta distribution. The hyperparameters of the

posterior are updated as:

$$\widehat{\alpha}_i = \alpha + k_i, \quad \widehat{\beta}_i = \beta + n_i - k_i. \quad (4.6)$$

This gives us a simple procedure for incorporating observations over time. We refer to all $p(\theta_i; \widehat{\alpha}, \widehat{\beta})$ at locations ℓ_i as the performance record of the detection system on a chosen route after traversal $\mathcal{T}$ and use it to estimate the likely performance of the robot at test time.

4.3.2 Decision Making using a Performance Record

We are considering the case in which the robot has to decide between operating autonomously and asking a human operator to take over control and perform the task reliably on its behalf. In this simplified scenario, the robot has only two actions available: a^0 , *denying autonomy* and a^1 , *offering autonomy* at every location along a driving route. The robot should choose action a^0 when it believes that its perception system is failing and a human operator should take over control and it should choose action a^1 when it believes that its perception system is functioning well and it can reliably operate autonomously.

We assume that there are only two states that the perception system can be in: failing (and producing false detections), or performing well (and the robot presents no risk when operating autonomously). Following standard one-shot decision theory, we assign a loss to state-action pairs, which reflects how serious it is to take action a^i when the actual state is s^j , for $i, j \in \{0, 1\}$:

$$L(a, s) = \begin{pmatrix} 0 & L_{\text{offer}} \\ L_{\text{deny}} & 0 \end{pmatrix}$$

The optimal action is the one that minimises the expected loss. This is computed as:

$$\bar{L}_\tau(a) = \sum_i p(s^i) L(a, s^i). \quad (4.7)$$

In order to compute the probability of the perception system being in either state $p(s^i)$, we introduce hyperparameter τ and denote by s^0 the event that the perception system is failing at location L_i . We compute its probability as

$$p(s^0|\theta, \tau) = p(\theta \leq \tau) = \int_0^\tau p(\theta; \hat{\alpha}, \hat{\beta}) d\theta, \quad (4.8)$$

where $p(\theta; \hat{\alpha}, \hat{\beta})$ has been estimated using the performance records proposed. We denote by s^1 the event that the perception system is performing well and compute the probability of it happening as $p(s^1|\tau) = 1 - p(s^0|\tau)$. Thus, τ can be thought of as a *decision* threshold (not to be confused with the threshold of the detection system itself), which modulates the level of confidence that is required before autonomy is offered by the robot.

The ratio of the losses associated with each action usually depends on the application domain and it ultimately comes down to a choice the system designer has to make. In this case it is a trade-off between false positive and false negative interventions so one might prefer to set the cost of inconveniencing a human operator lower than the cost of failing to detect an object. In Figure 4.3 we show the effect of adjusting the losses associated with each type of error on the actions selected. Type I, or false positive errors, correspond to the situations in which the robot denies autonomy (a^0) but its perception system is in reality performing well (s^1) and incurs a loss of L_{deny} . Type II, or false negative errors, occur when the robot fails to recognise that it is underperforming (s^0) and continues to operate autonomously (a^1). Figure 4.3 shows that by making type I errors more expensive (increasing L_{offer}), the robot is more conservative, and employs the safer action of denying autonomy more often. Ultimately, both types of errors decrease the user's trust

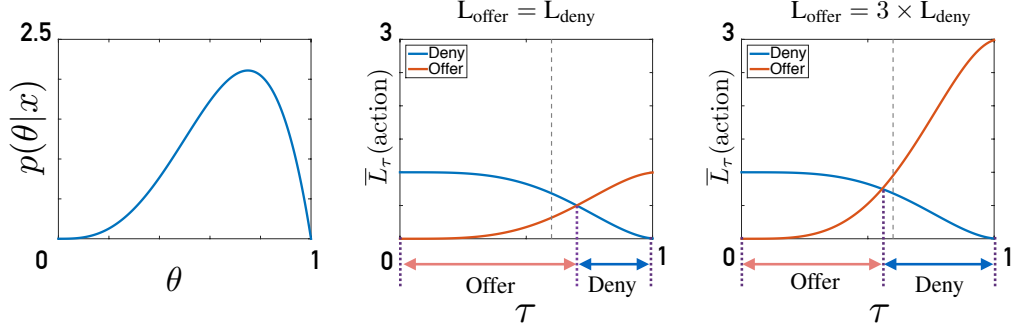


Figure 4.3: Figure shows the expected loss of choosing an action for an example posterior distribution $p(\theta|x)$ (left figure). Two different loss matrices are used, one in which the costs of the actions are equal (middle figure) and one in which L_{offer} is three times larger than L_{deny} (right figure). When $L_{\text{offer}} = L_{\text{deny}}$, for $\tau = 0.6$ (grey line), the action chosen by the robot is to offer autonomy because it has a lower expected loss $\bar{L}_{\tau=0.6}$. However, by setting $L_{\text{offer}} = 3 \times L_{\text{deny}}$, the optimal action becomes to deny autonomy. Increasing the cost of operating autonomously L_{offer} results in a more cautious system that generally prefers the safe action of denying autonomy.

in the system. Section 4.4 presents the results for equal cost on the actions as well as increased cost on offering autonomy.

4.3.3 Performance Records and the Experience Paradigm

In the initial phase of performance map building, in order to assign observations of performance to locations in the world, we make use of the robot’s navigation system, as well as a relational database framework ([Nelson et al., 2016]) to store and retrieve relevant information at runtime. The data and meta-data we are interested in, denoted by $\mathcal{D}$, consists of three-dimensional tuples at n discretised locations along a trajectory:

$$\mathcal{D} = \left\{ (\phi_i, X_i, L_i) \right\}_{i=1}^n, \quad (4.9)$$

where ϕ_i is an image, X_i is a set of observations of performance as described in Section 4.3.1 and L_i is a location along the trajectory that the robot has traversed.

At runtime, we make use of $\mathcal{D}$ in order to retrieve past performance at each test location L_{test} and estimate performance levels online. We propose two different

methods of retrieving observations associated to a location. Firstly, we look at geographical proximity given by GPS measurements and consider the observations in a local neighbourhood $N(L_{test})$, where

$$N(L_{test}) = \{i : \|L_i - L_{test}\| \leq \epsilon\}, \quad (4.10)$$

and ϵ is a prespecified radius. While this distance metric is useful for gathering all the observations close to a desired location, it does not take into account which of them are most relevant in terms of the appearance of the live frame. Imagine the following test case: while driving at night, past observations gathered during night time should be more relevant than observations gathered during day time. Similarly, detection in bright sunny conditions might have a different outcome than detection during rain. In these situations having a distance metric that also incorporates visual similarity is crucial.

The second method we propose employs Experience-Based Navigation (EBN) ([Churchill and Newman, 2013a], [Linegar et al., 2015]) in order to find the *closest location*, in terms of both appearance and location to a live frame. EBN is a vast-scale, camera-based localisation system, which gathers many representations of the same place under various conditions (*experiences*) and stores them in a graph structure. In this graph, the nodes represent landmarks and the edges between them contain six degrees of freedom transformations between the nodes. Loop closures are detected using FAB-MAP ([Cummins and Newman, 2011]). Most importantly for our application, EBN searches the experience graph, in a local neighbourhood of the robot's position estimate, in order to find the images which best matches the live frame. As in Figure 4.4, on a traversal of the trajectory, each frame may be localised against one or more frames from past traversals. Our intuition is that if two images have enough corresponding features to localise the robot, then they would produce a similar detection outcome. We denote the method of estimating performance using all past observations, regardless of the visual appearance of

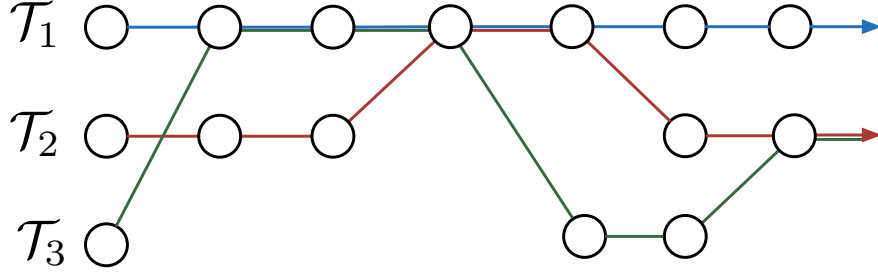


Figure 4.4: A simple illustration of the experience-based navigation framework. Each frame along a new traversal of a route can be matched to previous frames as a function of appearance. For example, individual frames in traversal 3 may be matched to frames in traversals 1 and/or 2.

the environment by **LOC**, since it only incorporates observations that are close in location. We denote the second method, which leverages EBN to explicitly distinguish between different appearances of the world and select observations from locations that are close both in physical distance and visual appearance by **APP**. This method first requires localising a live frame against an existing EBN experience-graph. Examples of the past observations selected by LOC and APP are shown in Figure 4.5. Whereas LOC selects images from past traversals that are closest according to GPS, APP collects the images that are most visually similar, in some cases compromising proximity for appearance. We expect APP to give more accurate estimates of performance as it has access to the outcome (the performance record) of images that are very similar to the live frame in terms of view-point, lighting, and even weather conditions. We further refer the reader to [Linegar et al., 2015] for a comprehensive description of the EBN framework employed.

4.4 Experimental Results

We evaluate the two methods proposed for estimating performance, LOC and APP, on 60km of driving data gathered in an urban environment in Milton Keynes over the course of six months. The same route has been traversed eight times under different environmental conditions using the data collection platform shown in Figure 4.6.

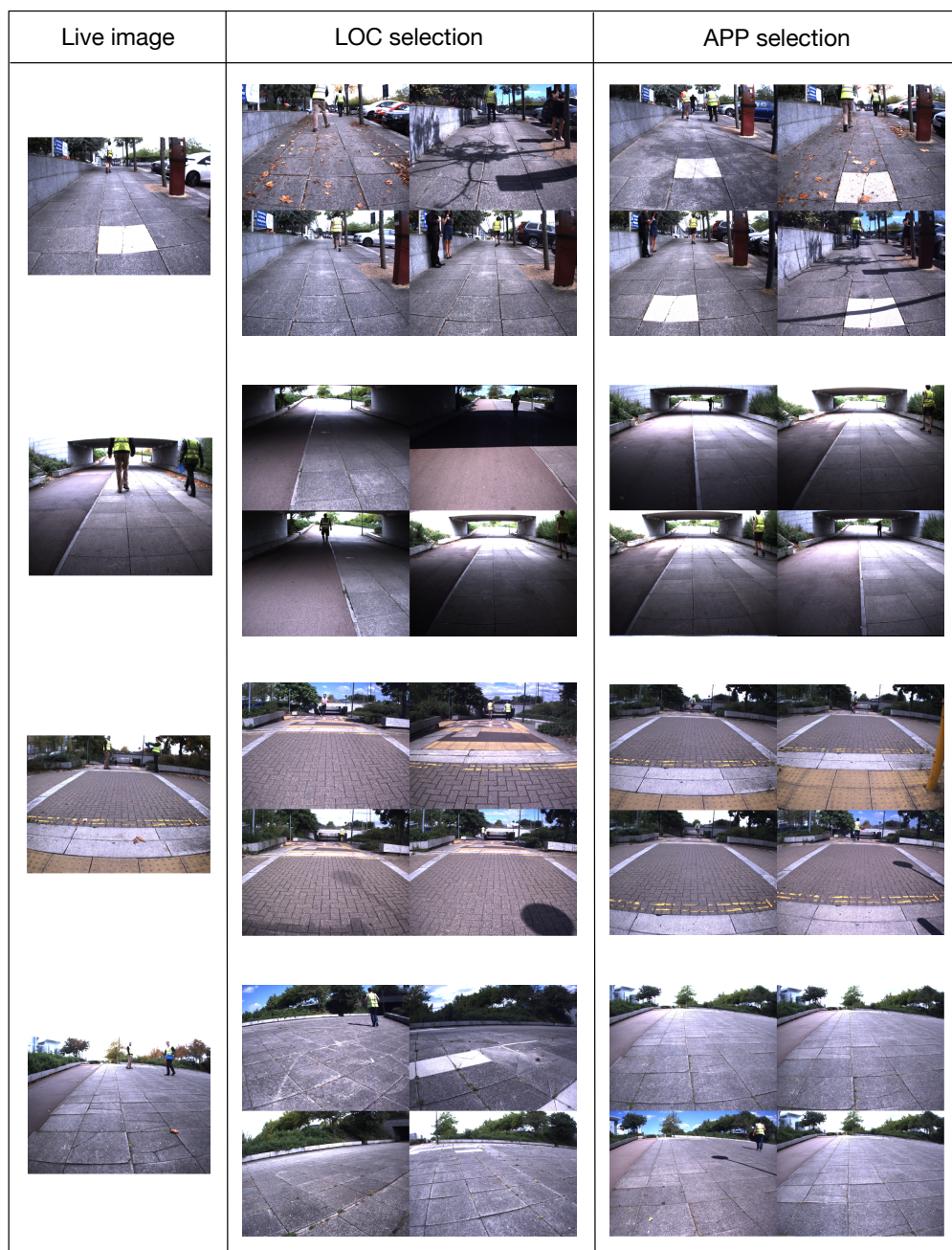


Figure 4.5: Example images from a test route in Milton Keynes. The figure shows the live frame from a test traversal (left), a set of images that LOC selects to use for building the location-specific performance record (center) and a set of images that APP selects for the same purpose though EBN, the image-based localisation system (right). Overall, the latter results in a more accurate performance record.

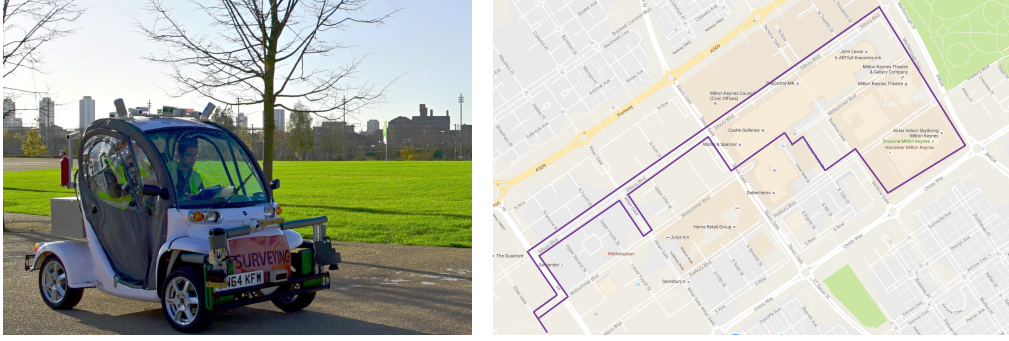


Figure 4.6: Figure showing the platform and the route chosen for experiments. The vehicle is equipped with a Bumblebee3 stereo camera, Velodyne Lidars HDL32E and an INS system used for data collection. We produce both 2D and 3D pedestrian detections in image and laser data along the route in Milton Keynes shown on the right.

4.4.1 Pedestrian Detector

We evaluate our two approaches using two distinct image-based pedestrian detectors. The first one is a support vector machine on Aggregate Channel Features (ACF) [Dollár et al., 2014] trained on the INRIA Person dataset [Dalal and Triggs, 2005] with five rounds of hard negative mining on data collected in central Oxford, which we refer to as SVM+ACF. The second detector used is MS-CNN [Cai et al., 2016], which achieves state-of-the-art performance on the KITTI [Geiger et al., 2013] and Caltech [Dollár et al., 2009] datasets. We operate MS-CNN at a detection threshold of 0.8, as this presents the best combination of precision and recall. This block can be easily substituted for any other pedestrian detector or, more generally, performance records can be built for any image-based vision system for which some kind of supervisory signal exists.

4.4.2 Surrogate Ground Truth

To assess the performance of our object detection system, as well as the quality of our predictions, we require ground truth annotations. Naturally, it is impractical to manually annotate such large datasets. Further, because our approach explicitly exploits repeat traversals of a given region or trajectory, widely used benchmarking

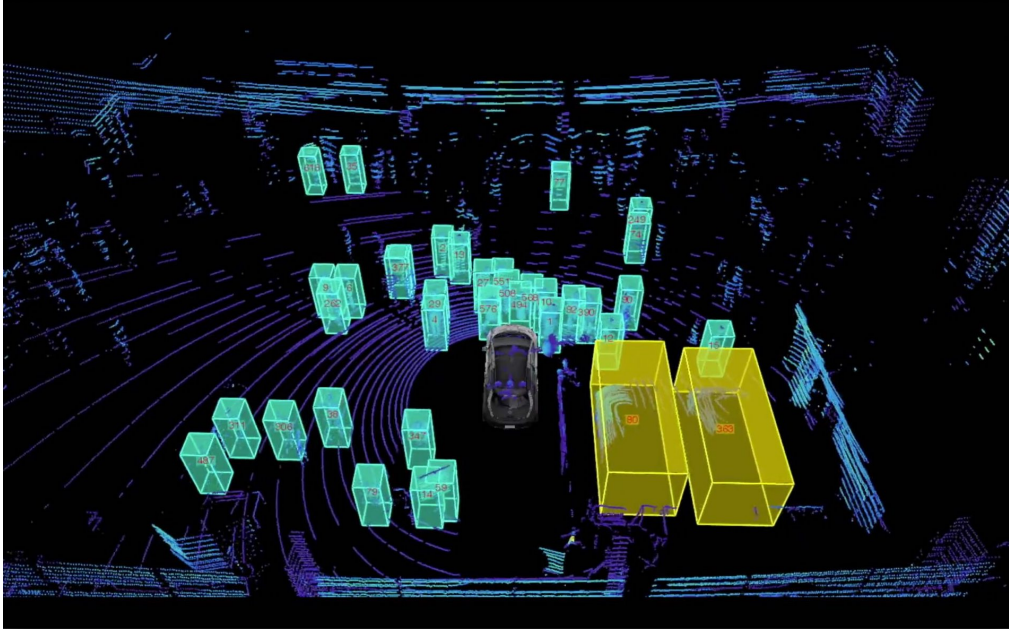


Figure 4.7: Vote3d pedestrian detection example. Laser detections are used as a surrogate metric of performance in the absence of manually annotated ground truth boxes, after empirically demonstrating that they produce a similar outcome.

datasets such as KITTI are unsuitable for the task. To be able to achieve our goal we make use of a surrogate metric of performance, which compares the image detections against laser detections (projected into the image), in order to obtain performance scores for the 2D pedestrian detector.

The laser detector used for providing the surrogate ground truth metric was trained on KITTI Velodyne data ([Geiger et al., 2013]). A thorough quantitative evaluation can be found in [Wang and Posner, 2015] and a qualitative example is shown in Figure 4.7.

To further improve the fidelity of this surrogate ground truth, the 3D laser detections are passed into a Kalman filter, ensuring that objects are consistently detected and tracked in consecutive frames. Note that although we require the laser sensor in order to build the performance record at training time, we do not require the sensor at test time. Further, by using a separate modality such as LIDAR to generate the surrogate ground truth, the labels are insensitive to the types of appearance-based variation that plague the visual perception system and lead to

degraded performance. We estimate performance and take optimal actions either using only the performance record and the location of the robot (required by LOC), or using the performance record and the incoming image feed (required by APP).

We have conducted an additional experiment to verify that treating laser detections as ground truth leads to a meaningful evaluation score. For this experiment, we use the F_1 metric in order to provide a single measurement of performance per image. The F_1 score is computed as:

$$F_1 = \frac{2TP}{2TP + FP + FN}, \quad (4.11)$$

where TP, FP and FN are true positives, false positives and false negatives respectively.

By applying the SVM+ACF detector to one dataset with (manually annotated) ground truth labels, consisting of 3627 pedestrians, we compute two sets of F_1 scores. The first set is obtained by comparing the SVM+ACF detections to ground truth annotations, and the second set is obtained by comparing them to the surrogate ground truth: the detections provided by the 3D laser. In all experiments presented in this paper, a detection is deemed to be a true positive if it overlaps a ground truth box with Intersection-over-Union (IoU) greater or equal to 0.5. Figure 4.8 shows a strong correlation between the two sets of scores which leads us to believe that our surrogate metric has empirically the same effect as using ground truth annotations. The heatmap, computed using kernel density estimation, has a peak at 1 meaning that the detector has a perfect F_1 score against both ground truth and surrogate ground truth, to which most of our detections contribute. However, the laser detector is not perfect – there exists a set of detections that score highly against ground truth but not laser. Therefore using a proper ground truth metric would further benefit the results. Nonetheless, by employing our surrogate metric of performance, the evaluation becomes entirely self-supervised. All performance scores can be retrieved effortlessly as soon as the robot has finished operating.

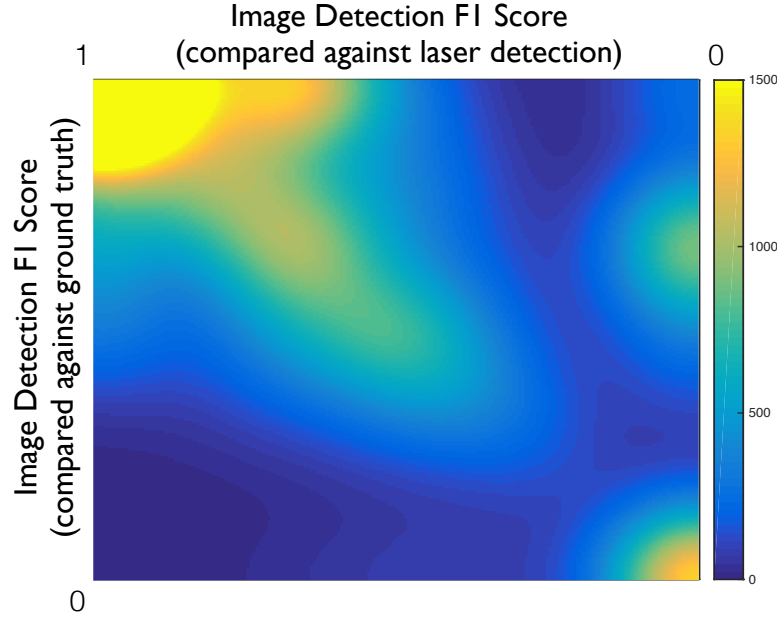


Figure 4.8: Heatmap showing a strong correlation between performance scores for the SVM+ACF detector computed in two different ways. Firstly, the 2D detection F_1 score is computed using ground truth annotations (y-axis). Secondly, the same score is computed using laser detections instead of ground truth (x-axis). The diagonal elements, as well as the peaks at 1 and 0, allow us to believe that there will be very little noise introduced by using a surrogate metric of performance.

4.4.3 Performance Estimation and Decision Making

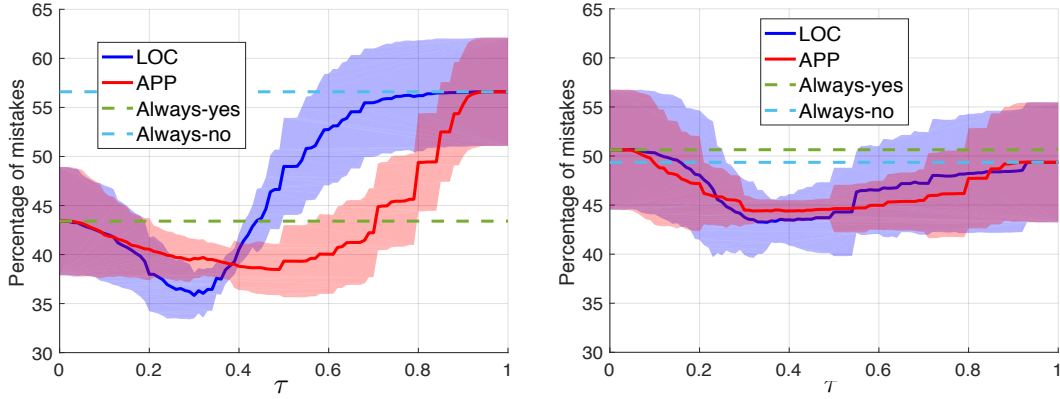
In order to evaluate the accuracy of the performance prediction methods we propose a metric that shows the usefulness of LOC and APP in a practical scenario. We compare the total number of mistakes the robot makes while employing the two methods, as described in Section 4.3. We also show two baseline cases ‘Always-yes’, corresponding to always operating autonomously, and ‘Always-no’, corresponding to always asking for help. What we refer to as *mistakes* are the outcomes of the following two cases:

- Choosing to deny autonomy when there are no false positive and no false negative detections in an image (i.e. perfect detector performance). These errors are of type I.

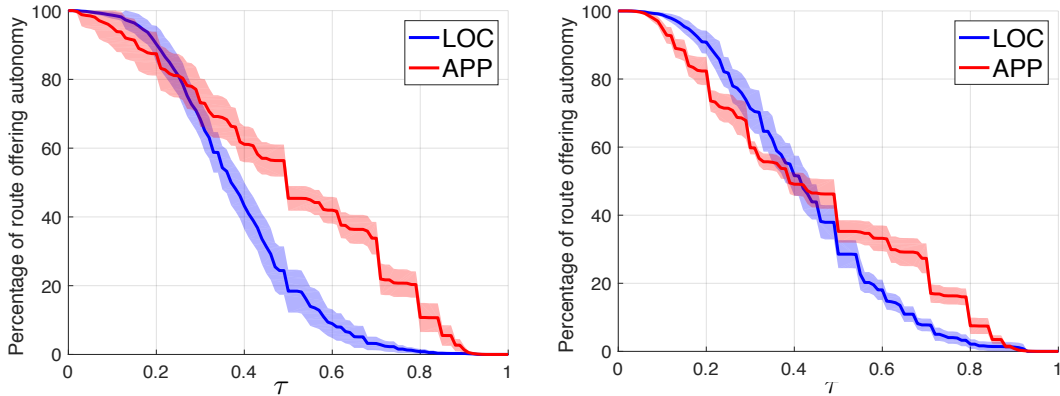
- Choosing to offer autonomy when there is at least one false detection in an image (i.e. imperfect detector performance). These errors are of type II, or critical errors, as in our situation, they can have severe consequences.

All decisions are taken on a per frame basis before the detection actually occurs. Additionally, for LOC, the estimates of performance do not rely on the live image, so they can be estimated even before the robot traverses the test route. We found that indeed mistakes at a particular location in the world are correlated (up to a difference in the appearance of the environment), and are a good indication of future performance. All figures show the results obtained in an evaluation of all traversals in a leave-one-out fashion and an equal cost ($L_{\text{offer}} = L_{\text{deny}}$) for each type of mistake.

Figure 4.9a shows the total percentage of mistakes made (in terms of incorrectly offering or denying autonomy) as a function of sliding parameter τ , the *decision* threshold, which characterises how good the past performance needs to be in order to decide to offer autonomy according to the LOC and APP methods. A value of $\tau = 0$ corresponds to the scenario of always offering autonomy (indicated as the constant *Always-yes* line) and a value of $\tau = 1$ corresponds to always denying autonomy (*Always-no*). The percentage of mistakes made for the *Always-yes* case indicates the fraction of frames in which a false positive/negative detection was made (and hence, offering autonomy was incorrect), and the *Always-no* case indicates the complement: the fraction of frames that exhibited perfect detection (such that denying autonomy was incorrect). Interestingly, the percentage of mistakes for *Always-yes* is higher for the MS-CNN than for the SVM+ACF, which indicates that the MS-CNN produces false detections on more *frames*. Based on visualisation of the detections, we suspect that this may be due to MS-CNN finding correct detections where our surrogate ground truth has failed to do so. While further analysis on this would be beneficial, we leave it to future work, since it is only tangentially relevant to the purpose of this experiment: to accurately characterise the performance of different detectors (regardless of their actual accuracy).



(a) Figure showing the percentage of total mistakes (both type I and type II) as evaluated in the decision making process using two different pedestrian detectors. SVM+ACF is shown on the left and MS-CNN is shown on the right. A decision is considered a mistake if the robot either offers autonomy when there is at least one false detection in the live frame (type II) or denies autonomy when there are only true detections (type I). *Always-yes* and *Always-no* are a function of the pedestrian detector and the input data used, which show the outcome of never switching between the autonomous system and the expert operator. Overall, we consider APP to have a better performance because it has the lowest percentage of mistakes for most values of τ .



(b) Figure showing the percentage of the route for which the robot decides to drive autonomously. For the same value of τ , the percentages correspond to the number of mistakes reported above with APP being preferable to LOC because it denies autonomy less often. The plot on the left shows the outcome of SVM+ACF being used, while the plot on the right shows the outcome of MS-CNN. In this case, *Always-yes* leads to 100% autonomy while *Always-no* leads to 0% autonomy.

Figure 4.9: Figure showing the outcome of the the decision making process. Although the robot does not offer autonomy for the entire trajectory (4.9b), the number of perception mistakes is reduced (4.9a). The evaluation is performed for all datasets in a leave-one-out fashion and for all values of the hyperparameters τ . The solid lines show the mean percentage of mistakes (4.9a) and the mean percentage of autonomy offered (4.9b). The shaded regions indicate one standard deviation from the mean.

		$L_{\text{offer}} = L_{\text{deny}}$			$L_{\text{offer}} = 3 \times L_{\text{deny}}$		
		Type I (%)	Type II (%)	A(%)	Type I (%)	Type II (%)	A(%)
SVM+ACF	LOC	51.39	1.36	6.57	54.96	0.51	6.57
	APP	27.66	12.38	41.31	38.69	6.19	24.09
MS-CNN	LOC	39.65	6.88	16.59	44.69	2.53	7.19
	APP	30.7	14.1	32.7	38.58	7.37	18.15

Table 4.1: Percentage of decision making mistakes separated into type I (denying autonomy despite performing well) and type II (offering autonomy despite performing poorly) as well as the percentage of the route driven autonomously (A). Results are shown for the two prediction methods proposed, LOC and APP, and for both pedestrian detectors employed, SVM+ACF and MS-CNN. The value of τ (hyperparameter at which the action is taken) is set to 0.6. The table also shows how increasing the cost of action L_{offer} lowers the type II errors but also reduces the percentage of the route driven autonomously. The lowest percentage of errors of each type is shown in bold. While LOC appears to outperform APP in terms of type II errors, this is because type II errors are computed solely on the occasions in which autonomy was offered (thus the amount of data is lower to begin with).

Both of these baseline options are considerably higher than the methods proposed, which encourages us to believe that if we allow the robot to deny autonomy occasionally, the overall performance on a task is improved. The proposed methods help both the SVM+ACF and the MS-CNN detector, with APP being particularly important for SVM+ACF, and less so for MS-CNN. We believe this is because the latter is a higher capacity detector which may already be appearance-invariant. Furthermore, for most values of τ , APP drives autonomously for a larger percentage of the route, as shown in Figure 4.9b. We attribute the result that APP has fewer mistakes and offers more autonomy than LOC to the fact that it more carefully selects observations from past traversals.

The choice of τ not only has an influence on the total percentage of mistakes and the amount of autonomy offered, but also on the specific type of mistakes. For lower values of τ , both methods are more permissive of driving which leads to more false negative mistakes (type II). The robot fails to recognise that the perception system is operating poorly. For higher values of τ , both methods deny autonomy more often which leads to more false positive mistakes (type I). These errors correspond to the

	30% autonomy		50% autonomy		70% autonomy	
	Type I (%)	Type II (%)	Type I (%)	Type II (%)	Type I (%)	Type II (%)
<i>LOC</i>	25.88	12.99	19.13	18.39	4.6	33.07
<i>APP</i>	21.73	12.57	17.28	15.94	4.2	29.37

Table 4.2: Percentage of decision mistakes (both type I and type II) for an equal amount of the route driven autonomously (30%, 50% and 70%). The pedestrian detector used is SVM+ACF and equal costs for each action. APP outperforms LOC in terms of both type I and type II errors.

less severe scenario of stopping the vehicle from driving despite good performance; therefore higher values of τ might be preferable in practice.

In addition to τ , encouraging the robot to take either action can be achieved by adjusting the $L_{\text{offer}}/L_{\text{deny}}$ ratio such that the action which incurs a lower cost will be selected more often (as demonstrated by Figure 4.3). For $L_{\text{offer}} = 3 \times L_{\text{deny}}$, for both prediction methods and detectors used in the experiments, we can reduce the type II errors even for the same value of τ . This is shown in Table 4.1, for $\tau = 0.6$. While it is straight-forward to see that APP produces a lower number of type I errors than LOC, for type II errors LOC seems to be preferable. This is because type II errors are computed strictly on the frames on which the decision taken was to offer autonomy, which is to begin with lower for LOC, as shown by the percentage of the route driven autonomously ($A(\%)$). This holds both for SVM+ACF and MS-CNN.

Table 4.2 shows that for a fixed percentage of the route driven autonomously (set to 30%, 50% and 70% respectively), APP also makes fewer type II errors (not seen in Table 4.1 due to varying percentage of autonomy). This result is computed for the case of $L_{\text{offer}} = L_{\text{deny}}$ and the SVM+ACF detector.

Throughout these experiments the robot has the opportunity to switch between two states of autonomy every couple of centimetres of the route it traverses, when in practice transitioning between the two would be highly inconvenient. For instance, for the MS-CNN detector with a detection threshold as well as a decision threshold of 0.7 the average length of a segment on which the robot does not switch its state is 1 meter, with the longest correct prediction of performance amounting to 55

meters and the longest incorrect prediction of performance to 30 meters. Just as false image detections are sporadic, so are the decisions to switch the current state of autonomy. This creates a segmentation of the trajectory (into on and off sections of the route) that the two methods proposed do not correct for.



(a) Example images correctly predicted to have good detection performance with high probability. All previous detections in the performance record at the same location are perfect.



(b) Example images correctly predicted to have poor detection performance with high probability. All previous detections in the performance record are incorrect.

Figure 4.10: Example images and their corresponding pedestrian detections using MS-CNN with a detector threshold of 0.8. The output of the pedestrian detector is shown in red and the ground truth is shown in green, with the goal of this work being to correctly predict the detector accuracy based on the performance records. The figure shows examples in which we correctly predict success with high probability (a), or correctly predict failure with high probability (b), using the APP approach.

Figure 4.10 shows a qualitative assessment when using the APP approach to predict the performance of the MS-CNN detector. In particular, we look at examples in which the model correctly predicts success with high probability (i.e. when the past performance records are all correct detections) in Figure 4.10a, and examples where the model correctly predicts failure with high probability (all performance records incorrect) in Figure 4.10b. The success examples in Figure 4.10a all represent straightforward conditions (without adverse lighting or weather) in which a detector could be expected to perform well. The failure examples in Figure 4.10b particularly indicate the ability of the performance records to keep track of both



(a) Example images correctly predicted to have good detection performance with high probability. All past detections in the performance record at the same location are perfect.



(b) Example images correctly predicted to have poor detection performance with high probability. All past detections in the performance record are incorrect.

Figure 4.11: Example images and their corresponding pedestrian detections using the SVM+ACF detector with a threshold of 0. The scores are not calibrated probabilistically and 0 was chosen because it provides the best trade-off between precision and recall. The output of the pedestrian detector is shown in red and the ground truth is shown in green, with the goal of this work being to correctly predict the detector accuracy based on the performance records. The figure shows examples in which we correctly predict success with high probability (a), or correctly predict failure with high probability (b), using the APP approach.

false positive detections (such as a bike rack in the rightmost image that always produces spurious detections), and false negative detections (such as those caused by under/overexposure in the left two images).

Equivalent results for the SVM+ACF detector are shown in Figure 4.11. In a similar vein, the frames correctly predicted to have good performance are under standard operating conditions, while some of the predicted failures indicate adversity in the environment (such as the pole in the second image which is falsely detected as a pedestrian), or in the conditions (over/underexposure in the last two images). Interestingly, some of the changes in lighting conditions are also tied to location: the underexposure is usually a result of entering a tunnel (or indeed, any other location likely to produce extensive shadows), while overexposure occurs when exiting these covered or shadowed regions. Such factors could therefore also be largely captured by the LOC approach.

Thus, these results illustrate the most significant benefit of the performance records: in making the performance prediction specific to place/time and appearance, it becomes possible to simultaneously consider both fixtures in the environment that may cause problems (such as the bike rack) or appearance changes that could contribute to failure (such as exposure conditions).

4.5 Discussion

4.5.1 Reasoning with Model Uncertainty

A very simple alternative to the proposed methods is to utilise the inherent uncertainty in the outputs of the model itself (i.e. the probability score of each detection) as a predictor of confidence. One may use the entropies of the detection probabilities as an uncertainty measure, or indeed the probability values themselves (as this is directly mapped to the entropy)

Such methods (i.e. [Grimmett et al., 2016]) are significantly different such that a like-for-like comparison is not straight-forward. Indeed one may use the model uncertainty in the prediction of each detection to get an estimate of how likely that detection is to be correct. However, LOC/APP solve a slightly different problem: rather than looking at the uncertainty of each detection sample (image crop), they classify an entire image captured at a certain location by looking back at past performance. We believe that making a fair comparison between retrospectively looking at past performance and introspectively looking at model uncertainty would require altering the initial frameworks in order to bring them to a comparable state.

Consider trying to characterise the performance of a pedestrian detector (with a threshold on the detection score for each box) using each of the two frameworks. On the one hand, LOC/APP do not use the incoming detections at all to take a decision (only to build the performance record). Therefore this record incorporates only observations coming from the detections above a certain threshold. On the

other hand, an uncertainty-based framework cannot take decisions based on just on a subset of the detection boxes as they would merely correspond to the detections with high probability of success, which would unfairly bias the robot towards offering autonomy (since the low probability detections have been removed). If however the low probability detections are not removed, then the performance record will be populated mostly by observations of false detections which would bias the robot towards not offering autonomy. This situation arises from the fact that introspection (as formulated by [Grimmett et al., 2016]) addresses the problem of classifier uncertainty and not sliding-window detection uncertainty.

An additional concern in using the detector score or entropy as an indicator of confidence, is that both thresholds (the *detection* threshold and the *decision* threshold) are coupled: a conservative decision threshold simply corresponds to a high detection threshold, and vice versa. Thus, we lose the ability to characterise decision-making performance as a function of τ for a single detection threshold. In other words, it is not possible to analyse decision-making performance and produce plots that are equivalent to Figure 4.9 for a single detector threshold.

4.5.2 Correcting Mistakes

Actively testing the value as well as the implications of the decision making process is vital for long range robot autonomy. We envision as future work not only predicting when a mistake is about to happen and switching the state of autonomy, but actively trying to correct for that mistake. For instance, if we separate the types of failure into false positives and false negatives and anticipate each type, we could dynamically change the operating threshold for the detection system, or intervene with place-specific experts in order to correct the detection outcome.

4.6 Conclusions

We have introduced a framework to predict the performance of a perception system on an autonomous vehicle as a function of space, time, and appearance, and determine when to relinquish control to the human user. Two approaches are proposed: one which uses a *performance record* of the detector at nearby locations on previous traversals, and a second which additionally incorporates appearance-based cues. Experiments with different detector models indicate that the proposed methods produce fewer decision-making mistakes than keeping the robot autonomy always on or off. Moreover, selecting past observations from similar environmental conditions in an appearance-based fashion further reduces the number of mistakes. Our work calibrates the outcome of a vision system to what the world looks like at the time of operation rather than an apriori (single) validation set by keeping track of its past performance. We believe that the proposed records can improve with more experience in the same workspace and represent a step towards reliable vision systems operating in the real world. An interesting direction for future work is addressing how to predict in advance exactly when the driver has to intervene (in terms of a *time to failure*) in order to notify them accordingly, whilst utilising appearance information from the current traversal.

5

Dropout Distillation for Efficiently Estimating Model Confidence

Introspection should be at the core of lifelong autonomy in order to know when a computer vision system can be trusted to operate in our environments. While in the previous chapter we estimate the influence of the outside world on the performance of an object detector, in this chapter we look at a model’s intrinsic introspective capacity, namely the accuracy of its reported uncertainty score.

Deep neural networks are now the default model choice due to their impressive performance on a wide range of tasks and benchmarks. However, predictive uncertainty estimates from standard neural networks are typically overconfident, often making them too unreliable to be deployed in real world applications ([Blundell et al., 2015], [Pawlowski et al., 2017]). Bayesian Deep Learning provides uncertainty

estimates at the cost of posing neural networks as approximate Bayesian models and drawing multiple posterior samples at test time ([Gal and Ghahramani, 2015]). The timing of this approach scales linearly with the number of samples, making Bayesian neural networks often too slow to use in practice. In this chapter we propose a method for improving the uncertainty estimates from neural networks that is simple to implement and more readily suitable to use in real-time applications.

5.1 Motivation

By treating the model parameters as a random variable, Bayesian probability theory allows us to compute uncertainty in the model predictions (see Section 2.1.1). Given a dataset $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$ and the output prediction $\mathbf{y} \in \mathcal{Y}$ for a new input $\mathbf{x} \in \mathcal{X}$, we can compute the posterior predictive distribution:

$$p(\mathbf{y}|\mathbf{x}, \mathcal{D}) = \int p(\mathbf{y}|\mathbf{x}, \mathbf{w})p(\mathbf{w}|\mathcal{D})d\mathbf{w}, \quad (5.1)$$

by using the likelihood $p(\mathbf{y}|\mathbf{x}, \mathbf{w})$ and a posterior distribution over model parameters $p(\mathbf{w}|\mathcal{D})$ which accounts for predictive uncertainty. For a non-linear neural network however, the term $p(\mathbf{w}|\mathcal{D})$ is analytically intractable. Various approximations to it have been proposed, but Bayesian Neural Networks are known to be cumbersome to train and computationally expensive to use in practice ([Blundell et al., 2015]).

[Gal and Ghahramani, 2015] have shown that dropout (a technique introduced by [Srivastava et al., 2014] and described in more detail in Section 5.3.1) also produces a Monte Carlo approximation to Equation 5.1 and that by sampling this approximation one can obtain samples from the posterior predictive distribution. These samples, which we refer to as MC-dropout samples, are often interpreted as an ensemble of models. The downside of the approach proposed by [Gal and Ghahramani, 2015] is that Monte Carlo sampling causes a linear increase in either time or resources, often making it difficult to use in real-time operation. This issue

is acknowledged in [Gal, 2016] as well as [Lakshminarayanan et al., 2016]. We propose a practical alternative for producing reliable uncertainty estimates at test time more efficiently, without the cumbersome sampling procedure.

We propose to learn, in a student-teacher paradigm, to produce accurate uncertainty values by observing the Monte Carlo dropout samples at training time only. In essence, the framework we propose amounts to training two models: one that is simply trained to maximise task accuracy (the *teacher*), and one that is specifically trained to output better uncertainties (the *student*). We refer to the latter as the *Distilled Dropout Network* (DDN), as it distills knowledge from multiple Monte Carlo samples from the teacher in order to improve the reliability of its uncertainty scores.

Our experiments empirically demonstrate that the uncertainty scores produced by DDN are indeed comparable to those provided by an ensemble of MC-dropout samples. Our model is computationally efficient by comparison as it only requires a single forward-pass and does not rely on sampling at test time. Additionally, we show that DDN also increases the task accuracy, although that is not our main objective.

5.2 Related Work

Model ensembles are known to significantly improve predictive performance ([Dietterich, 2000]) as well as to provide a way to measure uncertainty in the space of predictors. Methods for producing ensembles such as boosting, bagging (bootstrap aggregating) or stacking rely on fitting multiple models to subsets of the training data and averaging their predictions at test time. This process reduces the variance of a single prediction and, according to [Grimmett et al., 2016], leads to an increased *introspective* capacity. *Introspection* (see Section 4.2.1) refers to the ability of a model to associate an appropriate assessment of confidence to a test sample. Their hypothesis was also confirmed by [Mund et al., 2015] and [Triebl et al., 2013].

[Lakshminarayanan et al., 2016] also seek better calibrated models and propose the use of deep ensembles. They show that a non-Bayesian solution (ensembles)

can perform just as well as Bayesian inference in producing predictive uncertainty. Their research findings were encouraging in the direction of this work. However, they do not address the memory or computational increase introduced by model ensembles. Our research shows that an approximation to an ensemble is still effective at producing reliable uncertainty scores.

[Guo et al., 2017] propose to analyse some factors that cause poor calibration and identify that the loss choice, network architecture and weight decay all have an influence in this phenomenon. Our work is similar to theirs particularly in the evaluation procedure, as they measure calibration, accuracy and risk of neural networks used for image classification.

[Pereyra et al., 2017] propose penalising confident (low entropy) output distributions as a form of regularisation. They use an entropy-based loss term that penalises peaked distributions rather than using distillation or self-distillation in order to improve class confidence. While we do not address model regularisation, their loss function shares similarities with ours.

There are many works that use the outcome of an accurate, slow function that requires large computational resources as a regression target in order to learn a faster, simpler model to perform the task. Some of those works are [Ladický et al., 2015], which apply it to fluid simulations or [Ranzato et al., 2006], which apply it to efficiently learn a sparse coding model.

5.3 Approach

Our idea is inspired by [Hinton et al., 2015], which transfers knowledge from an ensemble of models to a single (smaller) model, a process for which they coin the term distillation. Our goal differs from that of [Hinton et al., 2015] in the sense that we do not seek a more compressed version of the ensemble (the teacher) model but rather a better calibrated version of it. The teacher’s prediction is the mean of the multiple dropout samples.

5.3.1 MC-Dropout

The idea of dropout was introduced by [Srivastava et al., 2014] as a way to regularise a neural network by allowing the units to be masked (*dropped*) during training in order to prevent their co-adaptation (which leads to overfitting). The mask on each weight is drawn from a Bernoulli distribution $Ber(\theta)$, where θ controls the probability of not dropping that weight. [Srivastava et al., 2014] interpret dropout as producing an ensemble of 2^n models, where n is the number of units in a single model. [Gal and Ghahramani, 2015] as well as [Kendall and Gal, 2017] have shown that the stochasticity introduced by dropout allows the network to sample the model space and estimate the uncertainty in output prediction $\hat{y}$. Assuming that f_w^θ is a network parametrised by $\mathbf{w}$ with a dropout mask generated by $Ber(\theta)$, sampling it:

$$\hat{\mathbf{y}} \sim f_w^\theta(\mathbf{x}), \quad (5.2)$$

allows us to obtain multiple output samples $\hat{\mathbf{y}}_i$ from stochastic forward-passes through the network. The predictive mean of the ensemble is the average of the M output samples,

$$\mathbb{E}_{f_w^\theta}(\mathbf{y}) \approx \frac{1}{M} \sum_{i=1}^M \hat{\mathbf{y}}_i \quad (5.3)$$

and the predictive variance is:

$$\text{Var}_{f_w^\theta}(\mathbf{y}) \approx \frac{1}{M} \sum_{i=1}^M \hat{\mathbf{y}}_i^\top \hat{\mathbf{y}}_i - \mathbb{E}(\mathbf{y})^\top \mathbb{E}(\mathbf{y}). \quad (5.4)$$

With infinite, non-i.i.d. training data, this variance converges to 0, but given the bias currently present in many computer vision datasets, models learnt on such datasets are far from having a low variance when tested in real-world robotics applications.

5.3.2 Knowledge Distillation

One of the most commonly used losses for training neural networks that output a categorical distribution is the cross entropy loss:

$$\mathcal{L}(\mathbf{w}) = - \sum_{i=1}^N \mathbf{y}_i^T \log(\sigma(f_w(\mathbf{x}_i))), \quad (5.5)$$

where $\mathbf{y}_i$ is a one-hot encoding of the ground truth labels (1 corresponding to the true class and 0 to all others) associated to sample $\mathbf{x}_i$. The softmax function σ ensures a probabilistic output of the vector $\mathbf{z}$ of predicted logits:

$$\sigma_i(\mathbf{z}) = \frac{e^{z_i}}{\sum_{j=1}^C e^{z_j}}, \quad (5.6)$$

where C is the number of output classes. In order to further clarify the notation, the difference between $\mathbf{z}$ and $\hat{\mathbf{y}}$ is that the former is the logit prediction $\mathbf{z} = f_w(\mathbf{x})$ while the latter is the predicted probability output $\hat{\mathbf{y}} = \sigma(f_w(\mathbf{x}))$.

With cross entropy, even when all predictions $f_w(\mathbf{x})$ are correct, the network can still decrease the training loss by increasing the probability assigned to the true class and decreasing the ones assigned to the rest of the classes. We hypothesise that this is what usually leads to the overconfidence of state of the art neural networks. Another problem with the cross entropy loss is that it encourages the assignment of high probability to a single class when in reality the model could be moderately confident in two or more classes (for instance, it is understandable to confuse cats and dogs, particularly when they are partially occluded).

Knowledge distillation, while having a different goal to ours, is also motivated by allowing networks to express this kind of uncertainty at training time. In [Hinton et al., 2015], the authors claim that seeing the probabilities assigned to incorrect classes as well as the correct class makes student models generalise better. In order to obtain such targets (often called ‘soft targets’), their approach is to raise

the temperature of the the teacher model’s softmax function until it produces a softer distribution over the output classes:

$$\sigma_i(\mathbf{z}) = \frac{e^{z_i/T}}{\sum_{j=1}^C e^{z_j/T}}, \quad (5.7)$$

where T a temperature parameter controlling how much to affect the output distribution. They propose to use two losses: a ‘hard’ cross entropy loss with the ground truth labels (Equation 5.5) as well as a ‘soft’ cross entropy loss with high temperature.

In this work, we propose a distillation process that uses a similar mechanism in order to produce better calibrated uncertainty scores. Rather than raising the temperature, we use the mean prediction of multiple dropout samples.

5.3.3 The Distilled Dropout Network

The Distilled Dropout Network (DDN) is trained to mimic the predictions of an ensemble, using distillation in such a way that it trains a better calibrated model. Similar to [Hinton et al., 2015] we use both the one-hot ground truth labels and a proposed soft target distribution obtained by averaging the Monte Carlo samples. However, the way we transfer knowledge is more similar to [Ba and Caruana, 2014], as we use the logits z as soft targets rather than the probabilities produced by the softmax.

Given training data $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i)\}_{i=1}^N$ containing a set of input examples $\mathbf{x}$, ground truth labels $\mathbf{y}$ and the corresponding set of distilled labels $\mathbf{z}$ obtained using a teacher model $g_{\mathbf{w}}^\theta$, we train the student model $f_{\mathbf{w}}$ by minimising the loss:

$$\mathcal{L}(\mathbf{w}) = \mathcal{L}_{\text{DD}}(\mathbf{w}) + \lambda \mathcal{L}_{\text{GT}}(\mathbf{w}) + \gamma \mathcal{L}_{\mathcal{R}}(\mathbf{w}). \quad (5.8)$$

The distilled dropout loss $\mathcal{L}_{\text{DD}}$ is defined as the L_2 loss with the teacher logits $\mathbf{z}$:

$$\mathcal{L}_{\text{DD}}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N \|f_w(\mathbf{x}_i) - \mathbf{z}_i\|_2^2, \quad (5.9)$$

$\mathcal{L}_{\text{GT}}$ is the standard cross entropy loss with the ground truth labels y :

$$\mathcal{L}_{\text{GT}}(\mathbf{w}) = -\frac{1}{N} \sum_{i=1}^N \mathbf{y}_i \log(\sigma(f_w(\mathbf{x}_i))). \quad (5.10)$$

and $\mathcal{L}_{\mathcal{R}}$ is a regularisation loss. The hyperparameters λ and γ control the importance of the additional losses.

To obtain the logit target labels $\mathbf{z}$, we average M MC-dropout predictions from the teacher model g_w^θ on exactly the same input data $\mathbf{x}$:

$$\mathbf{z}_i = \frac{1}{M} \sum_{i=1}^M g_w^\theta(\mathbf{x}_i). \quad (5.11)$$

Using additional data in order to generate more training samples for the student should also be possible. This includes even samples without ground truth annotations which could be incorporated into the student training procedure without computing $\mathcal{L}_{\text{GT}}$. This idea will be the subject of future research.

5.4 Classification Experiments

We first evaluate the outcome of our proposed training procedure on the task of image classification, and then extend its usage to the problem of object detection in Section 5.5.

5.4.1 Dataset and Network Architecture

We use the CIFAR-10 dataset ([Krizhevsky, 2009]) for image classification. The dataset consists of coloured natural images of size 32×32 pixels drawn from 10

different classes. The training and test sets contain 50k and 10k images respectively, with an equal class balance.

As a baseline we train a Dense Convolutional Network (DenseNet) model architecture introduced in [Huang et al., 2017] which has shown impressive results on image classification. The model is similar to a Deep Residual Network (ResNet) from [He et al., 2015], as they both efficiently train very deep networks with shortcut connections to allow gradients to propagate all the way through the network. The interesting idea behind DenseNet is that of feature reuse – each layer is connected to all preceding layers within a block, by using the intermediary feature-maps as inputs to subsequent layers. We use the 40-layer configuration with a growth rate of 12 described in [Huang et al., 2017]. More precisely, we train the network using stochastic gradient descent (SGD) using batches of size 64 for 300 epochs. The initial learning rate is set to 0.1, and is then divided by 10 at 50% and 75% of the total number of training epochs. We use a weight decay of 10^{-4} and a Nesterov momentum ([Sutskever et al., 2013]) of 0.9 without damping. The dropout rate (i.e. the probability of retaining a unit in the network) has been set to 0.8. All hyperparameters mentioned have been tuned and provided by the original DenseNet work ([Huang et al., 2017]).

We use the teacher model to generate 100 Monte Carlo samples per input for the knowledge distillation process. The dropout rate for generating these targets has also been set to 0.8 (typically using a dropout rate different than the one used in training causes less accurate predictions as shown in [Gal et al., 2017]). DDN is trained from scratch using an identical network architecture, training regime and hyperparameter values, the only difference being the loss function described in Equation 5.8.

At test time, both networks have a dropout rate set to 1. We occasionally refer to the baseline model as ‘No Dropout’ to indicate a standard single forward-pass through the network with no dropout. Although they have the same weights, the distinction between the baseline and the teacher is that the latter is stochastic (dropout rate set to 0.8) in order to generate multiple Monte Carlo samples.

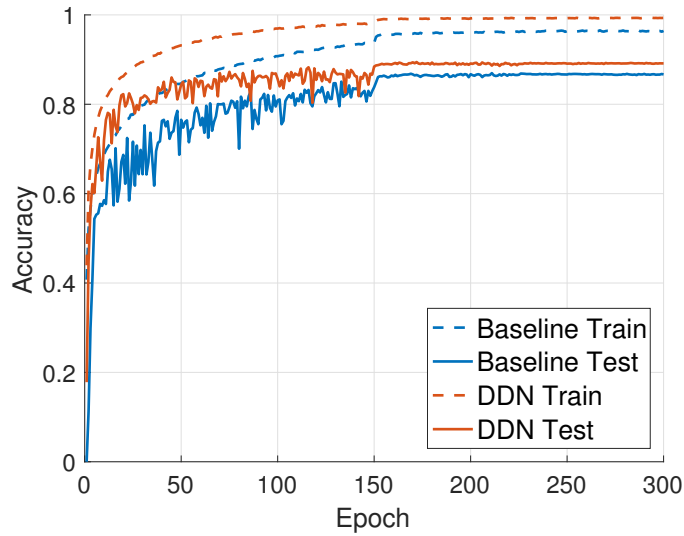


Figure 5.1: Figure showing the accuracy of the proposed model as well as a baseline teacher model during training and testing on the CIFAR-10 dataset. The student model (DDN) achieves better test accuracy than the teacher model, despite being trained on the same data with an identical training regime.

5.4.2 Results

DDN achieves better training and testing accuracy than the baseline model as shown in Figure 5.1. The final test accuracy of the teacher is 87%, while the student model can reach 90% test accuracy. This is also confirmed by the works of [Pereyra et al., 2017] and [Hinton et al., 2015]. The former claims that encouraging low-entropy predictions is a form of model regularisation which helps the model’s generalisation capabilities. The latter attributes it to the fact that the student model has knowledge of the ratios between output class confidence scores, while the teacher does not.

Figure 5.2 confirms that DDN produces higher entropy output probabilities. The histograms show the probability assigned to the predicted class for both models. Only 38% of the predictions made by DDN have a confidence score above 0.95 compared to 77% of the predictions made using the baseline model with no dropout. The baseline is not only less accurate in its predictions, but also more confident. DDN does not explicitly penalise low-entropy (overconfident)

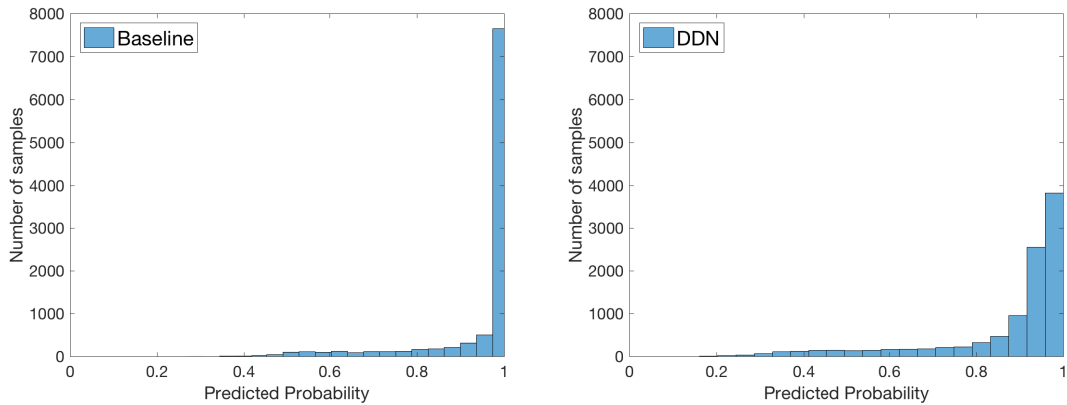


Figure 5.2: Probability scores assigned to the top scoring predicted classes for 10k test examples. The plot on the left shows the scores of the baseline model, while the plot on the right shows the ones assigned by our proposed model. The baseline model is less accurate in its predictions but more confident in the probability assigned to them.

distributions in training but rather encourages predictions to also assign some probability mass to classes which it is less sure about.

We refer the reader back to Section 2.3 which details some evaluation metrics used for measuring the quality of a probabilistic output. One of the main measures of interest is *calibration* which computes the difference between the confidence predicted by the model and empirical frequencies of successful classifications. We look at this measure via calibration plots ([DeGroot and Fienberg, 1983]) as well as the scalar summary statistics Mean Squared Calibration Error (MSCE) and Maximum Calibration Error (MCE).

We compute MSCE and MCE by binning all the prediction confidences and, for each bin, computing the accuracy and the mean and max of the probability scores assigned to that bin. Visualised on the calibration plot, a perfect calibration corresponds to a straight line from (0, 0) to (1, 1). Deviation from the diagonal towards the bottom right of the plot indicates overconfidence while deviation towards the top left indicates underconfidence in the predictions. Calibration plots are now a standard measure for predictive uncertainty quality used in works such as [Kendall and Gal, 2017], [Guo et al., 2017], [Gal et al., 2017] and [Ondruska and Posner, 2014].

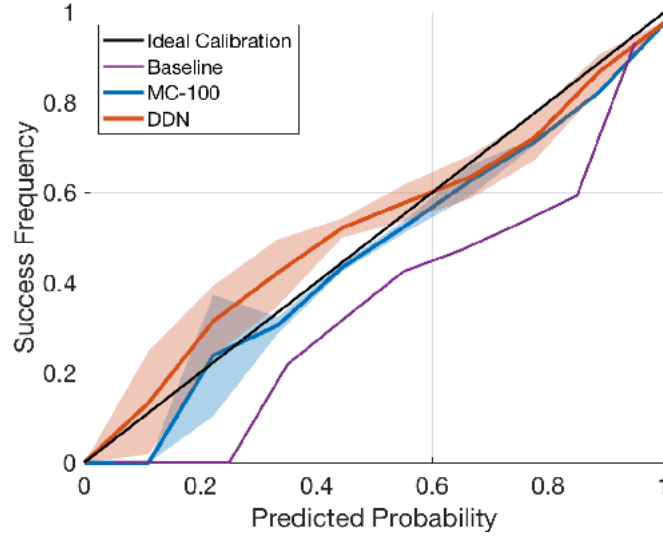


Figure 5.3: Calibration plot showing the correlation between class uncertainty and (empirical) success frequency. Perfect calibration of the model corresponds to the line $y = x$ shown in black. Baseline corresponds to using all the units in the teacher model (dropout is disabled by default at test time) and MC-100 is the outcome of sampling the model 100 times and averaging the predictions. DDN is the proposed student model that can produce predictive uncertainty scores very close to MC-100 in a single forward-pass through the network.

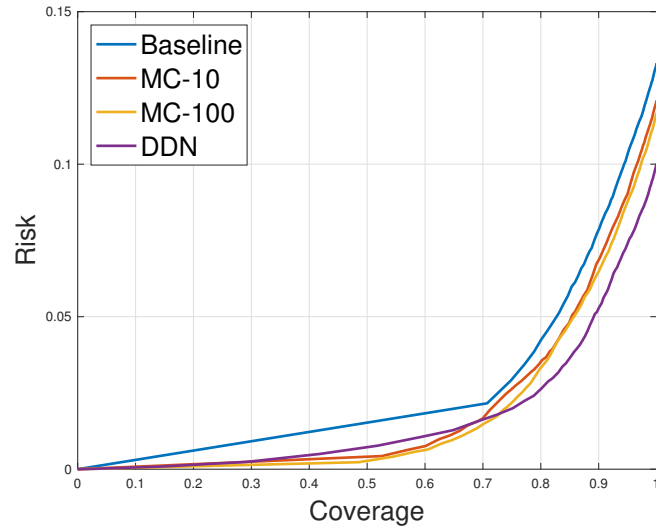


Figure 5.4: Risk-Coverage plot showing the percentage of mistakes (risk) for a given coverage of the test data. The predictions are ordered by confidence such that in the low coverage areas of the graph only the most confident predictions are used. A coverage value of 1 corresponds to using the entire test set. The baseline model presents the highest risk for all coverage values. For a coverage of 70% or more DDN has a lower risk even compared to MC-100. This is explained by the fact that DDN is a higher accuracy model.

Metric	Baseline	MC-10	MC-100	DDN
MSCE	0.9676	0.0334	0.0330	<u>0.0927</u>
MCE	5.0694	<u>0.8898</u>	0.5250	1.0790
Brier	0.2126	0.1745	0.1705	<u>0.1548</u>
Risk	0.1331	0.1209	<u>0.1169</u>	0.1002
No. forward passes	1	<u>10</u>	100	1

Table 5.1: Calibration results on CIFAR-10 test set using various metrics. Best outcome is displayed in bold and second best is underlined. The baseline model is a teacher model used with no dropout and a single forward-pass through the network. MC-10 and MC-100 are ensemble models given by 10 and 100 stochastic samples from the baseline model. DDN is our proposed student model learnt from scratch via (dropout) knowledge distillation from the baseline.

Figure 5.3 shows that the probability scores assigned to the test samples by DDN are better calibrated than those assigned by the baseline model. The mean of the DDN models trained has an MSCE score of 0.06 compared to 0.16 produced by the baseline. The training uses as targets 100 MC-dropout samples from the teacher model. In this example the teacher model has the same model weights as the baseline, the only difference being that its dropout rate has been set to 0.8 instead of 1 in order to produce stochastic samples. This figure also confirms that the DDN predictions are slightly underconfident, while the baseline model predictions are overconfident, a bias that tends to be preferred in high-risk applications. We also compare the baseline and DDN models to a very well calibrated model given by 100 Monte Carlo samples (MC-100) with an MSCE score of 0.02. The variance in the distilled models as well as MC-100 is a result of the stochasticity in the training procedure (dropout, random shuffling of training samples) observed in retraining the models. The plot shows the mean and variance of three distilled models and three MC-100 models. The higher variance in lower predicted probabilities (below 0.5) can be explained by the fact that there were fewer test samples in those bins.

To further expand our calibration analysis, Table 5.1 shows the results on the remaining calibration metrics introduced in Section 2.3, MSCE, MCE and the Brier score. Perfect calibration amounts to MSCE, MCE and Brier scores of 0.

Metric	Baseline	MC-10	MC-100	DDN
NLL	0.0634	0.1293	0.1318	<u>0.1267</u>
Entropy	0.1331	<u>0.2661</u>	0.2790	0.4445

Table 5.2: Negative log-likelihood (NLL) and entropy (H) results on CIFAR-10 test set. Most confident outcome is displayed in bold and second most confident is underlined. We note that the two metrics are not good indicators of model calibration as neural networks tend to make mistakes with high confidence.

The model’s risk, shown in Table 5.1 as well as Figure 5.4, is defined in Section 2.3 as the percentage of mistakes made for a particular coverage of the dataset (i.e. the proportion of test samples used). The coverage is computed as the percentage of test samples where the model’s confidence is above a threshold which we vary from 0 to 1 in Figure 5.4. The figure shows that at a low coverage of the dataset, a low risk is expected since only the most confident predictions are used. However, for a higher coverage, DDN has the lowest risk, followed by MC-100. The risk values in Table 5.1 correspond to a full coverage of the test set. These results are similar to the ones presented in Chapter 4 regarding offering or denying autonomy. In both cases they show that the number of mistakes can be reduced if an agent is given the possibility to ask for help rather than classify a test sample. A lower risk, or a lower number of mistakes, corresponds to a more reliable assessment of model uncertainty.

Finally, an important comparison measure is the time it takes to produce a prediction. As this depends on the model architecture, we measure it in the number of forward-passes through the network. The ensemble models require, 10 resp. 100 forward-passes, while our baseline and DDN models make a single prediction.

Table 5.2 shows that the baseline model has the best NLL score, while the other metrics in Table 5.1 indicate that it has the worst calibration. NLL and entropy measures have been introduced in Section 2.3. Lower NLL and entropy values indicate increased confidence in the prediction but do not necessarily correlate well with the accuracy of the model.

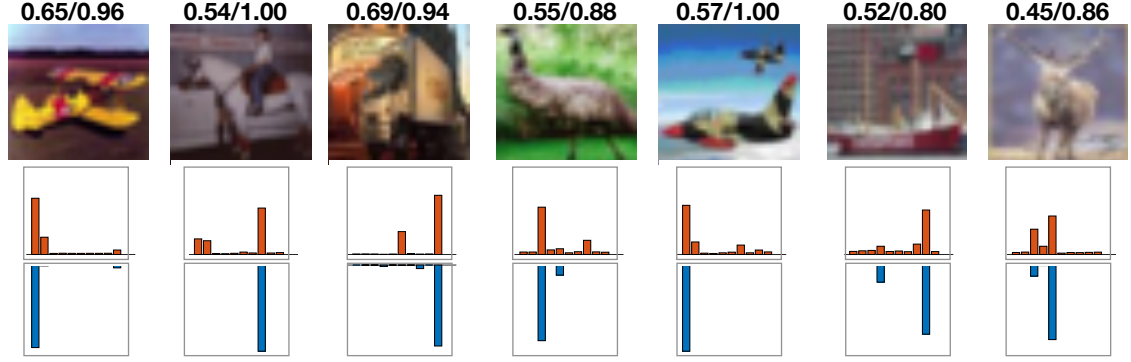
5.4.3 CIFAR-10 Qualitative Examples

Figures 5.5 and 5.6 show some examples of confidence scores assigned by our proposed model as well as the baseline on the CIFAR-10 test set. We point out some samples in which our model changes the classification outcome or has an expected behaviour such as lowering the confidence of wrong predictions. We also show some failure cases in which DDN increases the confidence in an incorrect classification.

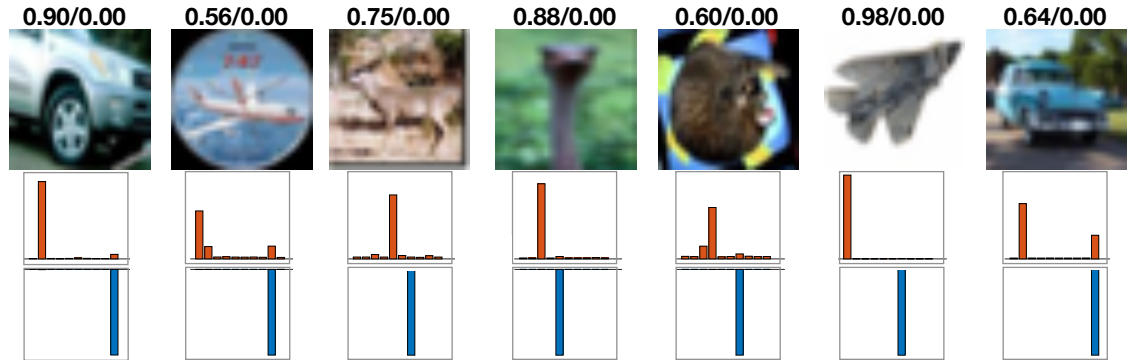
In general, the outcome distribution of our proposed model (shown in red in Figure 5.5) has higher entropy than that of the baseline (shown in blue as an inverted histogram). Figure 5.5a shows examples in which both models score the true class highest but DDN also assigns some probability to similar output classes. Such examples are most prevalent and demonstrate DDN’s capacity to be more uncertain in its predictions.

Figure 5.5b contains some of the samples for which the baseline model assigns the highest probability to an incorrect class while DDN classified the sample correctly. Such examples make up for 6.75% of the test dataset. They are the success cases of DDN as the student network is able to change the prediction outcome and increase the task accuracy in addition to improving the model calibration. We note that the opposite, i.e. the percentage of samples classified correctly by the baseline but not by DDN, only makes up for 3.45% of the test dataset. In the majority of the success case examples the increase in probability assigned to the true class between the baseline and DDN is substantial (the mean is 0.64 and the variance is 0.04). Figure 5.5b shows examples to which the baseline model assigns a confidence score higher than 0.99 to an incorrect class (overconfident and wrong).

A percentage of 6.56% of test samples are classified incorrectly by both models. On 70% of these examples, DDN increases the probability assigned to the true class. A maximum probability increase is of 0.4, despite still classifying that sample incorrectly. On the other hand, Figure 5.6a shows test samples that are classified correctly by both models. In fact, 83.23% of all test samples are in

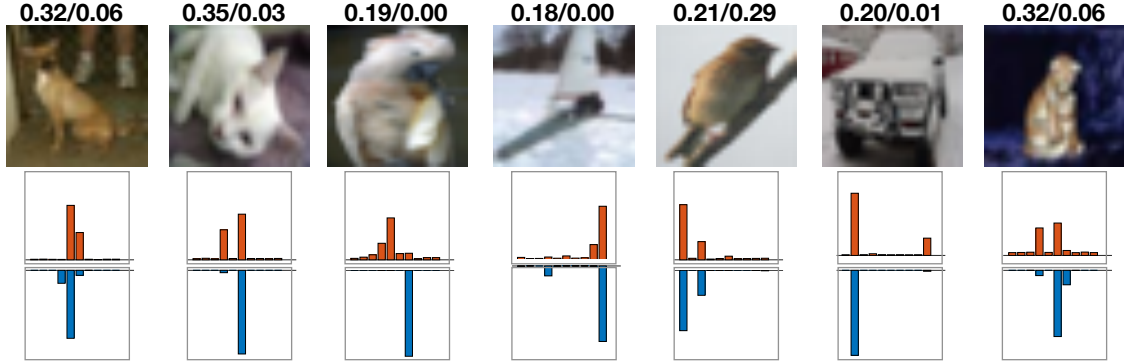


(a) Test samples where the classification outcome is correct for both the baseline model and for DDN. The samples chosen illustrate the most prevalent outcome scenario in which the output distribution predicted by DDN has higher entropy than the one predicted by the baseline.

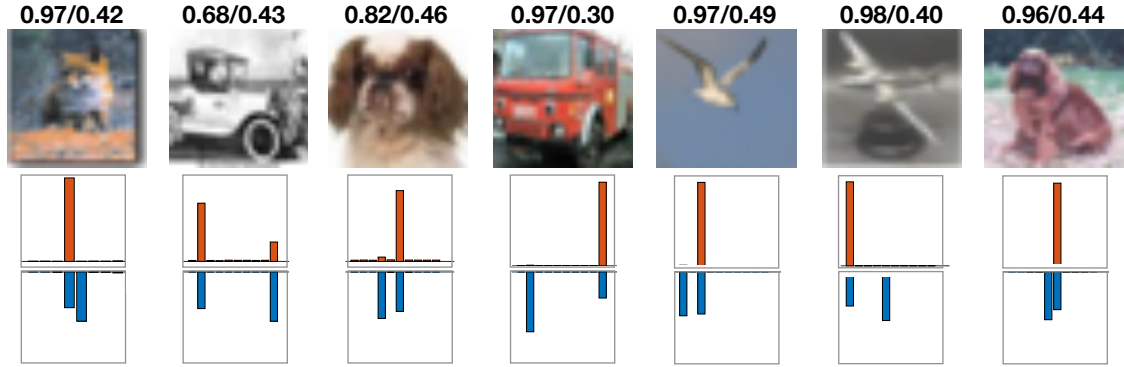


(b) Test samples for which the baseline is overconfident (and wrong) about its prediction with a confidence score higher than 0.99 assigned to a different class but for which DDN makes a correct prediction. While occasionally the output classes confused are understandable (for instance, cars and trucks), the confidence with which the mistake is made serves as the motivation behind this work.

Figure 5.5: Qualitative examples of CIFAR-10 classification. Figure 5.5a shows test examples classified correctly by both models, while Figure 5.5b shows examples which were classified incorrectly by the baseline and correctly by DDN. The probability assigned to the true class is shown on top of each image sample, with the score assigned by DDN on the left and the score assigned by the baseline model on the right. Below the test sample, we show the probability distribution assigned to all 10 output classes by DDN (in red) and by the baseline (in blue, as an inverted histogram).



(a) Test samples incorrectly classified by both models to which DDN assigns a lower probability to the wrong class. One of the essential qualities of an introspective classifier is making mistakes with lower confidence. In almost all cases, DDN assigns some probability mass to the true class, often assigning to it its second highest probability score.



(b) Test samples for which the classification outcome is incorrect for the baseline model but correct for DDN. They represent the success cases of our proposed method, in which it doesn't only improve calibration, but it also increases the accuracy of the model. In this case the baseline model makes its prediction with low confidence and its mistakes seem understandable. For instance, the first dog (third sample) is confused with a cat, while the second dog is confused with a deer. The car and truck are also confused with each other by the baseline. Interestingly, the bird is classified as an aeroplane, while the black and white image of the aeroplane is classified as a deer.

Figure 5.6: Qualitative examples of CIFAR-10 classification. Figure 5.6a shows test examples which were classified incorrectly by both the baseline and DDN, while 5.6b shows test examples which were classified correctly by DDN and wrongly by the baseline. The probability assigned to the true class is shown on top of each image sample, with the score assigned by DDN on the left and the score assigned by the baseline model on the right. Below the test sample, we show the probability distribution assigned to all 10 output classes by DDN (in red) and by the baseline (in blue, as an inverted histogram).

this category. The baseline predictions' probability scores have a mean of 0.97, while for DDN the mean is 0.93.

5.5 Object Detection Experiments

Our proposed dropout distillation method can also produce better calibrated object detector models. We show this by using the Faster R-CNN framework acknowledged for its state of the art performance ([Ren et al., 2017a]). Making minimal modifications to the original framework, we fine-tune the original models with a distillation loss until calibration is improved. Fine-tuning refers to the process of initialising the model weights with values obtained from a another, similar, training process. In our case the initial training is the baseline model training.

5.5.1 Dataset and System Architecture

All experiments are performed on the COCO dataset ([Lin et al., 2014]). The COCO dataset consists of roughly 80k training, 35k validation and 5k test images. The dataset covers 81 class categories which includes the 'background' class usually interpreted as no detection of interest.

The network architecture considered is ResNet ([He et al., 2016]). Because a Faster R-CNN detection model requires significantly more time and computation to train from scratch, the student model is only fine-tuned using the distillation loss. As teacher models (baselines) we use the original Faster R-CNN models trained on COCO: ResNet-50 and ResNet-101 trained with the open-source implementation of [Chen and Gupta, 2017].

As introduced in Section 2.2.1, Faster R-CNN is a two-stage detection framework. The first stage consists of a region proposal network (RPN) that produces class-agnostic object proposals called Regions of Interest (RoIs). The RPN is a fully convolutional network that simultaneously predicts object bounds and 'objectness'

scores at each position. The second stage of the Faster R-CNN framework consists of a bounding box classifier as well as a bounding box regressor which take as input the RoIs and outputs the final bounding box scores and locations. The two stages are trained jointly.

For our purpose, once a baseline model has been trained, we freeze the RPN model weights. The distilled dropout loss will only improve the object classifier, while the RPN will produce the same set of proposal boxes as the original models. We keep all classes and all image data exactly as in the original training procedure. As is common practice, the models are pre-trained on the ImageNet dataset ([Deng et al., 2009]) and fine-tuned on the COCO training and validation data for 490k epochs.

For each of these baseline models, we generate the DDN target logits from multiple MC samples. Due to computation constraints, we limit ourselves to 10 samples. The process of combining the scores from multiple MC samples is similar to the NMS procedure introduced in Section 2.2.4. However, instead of discarding the lowest scores we take an average of all the boxes that have an IoU overlap with ground truth higher than a set threshold and whose class prediction matches the true class label. We test different overlap threshold scores in order to generate the logit targets data (0.3, 0.5 and 0.7). For the ground truth bounding boxes that do not have any detections associated to them from the sampling procedure, we set as target for the distillation loss a uniform distribution over the output classes to encourage the student model to be very uncertain. This is motivated by the fact that since no detections were associated to it, the ground truth bounding box must be a difficult sample. Incorporating this type of information in the training procedure helps prevent DDN from making mistakes with high confidence.

All hyperparameters are set to values recommended by [Chen and Gupta, 2017]. In training both the baseline models and the DDNs, we compute losses only on boxes that score higher than 0.05, with a minibatch of 128 examples per image and a single image per batch. We use an initial learning rate of 0.001 with a reduction factor of 0.1 after 30000 epochs, a weight decay of 0.0005, a momentum of 0.9 and

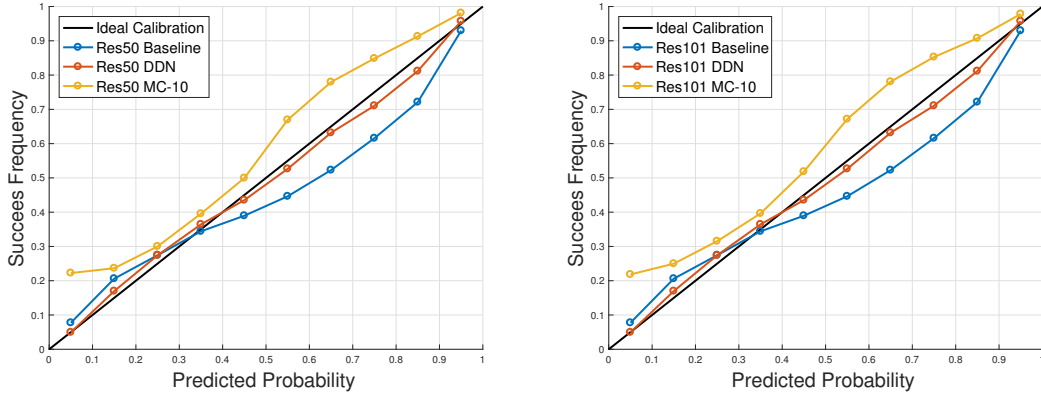
		mAP@[.5, .95]	mAR@[.5, .95]	MSCE	MCE	NLL
RES50	Baseline	0.292	0.413	0.1633	0.1988	1.1097
	DDN	0.292	0.408	0.0608	0.1158	1.4881
RES101	Baseline	0.320	0.442	0.1866	0.2118	1.1091
	DDN	0.318	0.433	0.0560	0.1043	1.5254

Table 5.3: Faster R-CNN test results for baseline models (ResNet50 and Resnet101) with no dropout and DDN models. Best outcome is shown in bold. In terms of calibration (MSCE, MCE), DDN shows a significant improvement over the baseline at a very small cost in terms of model accuracy.

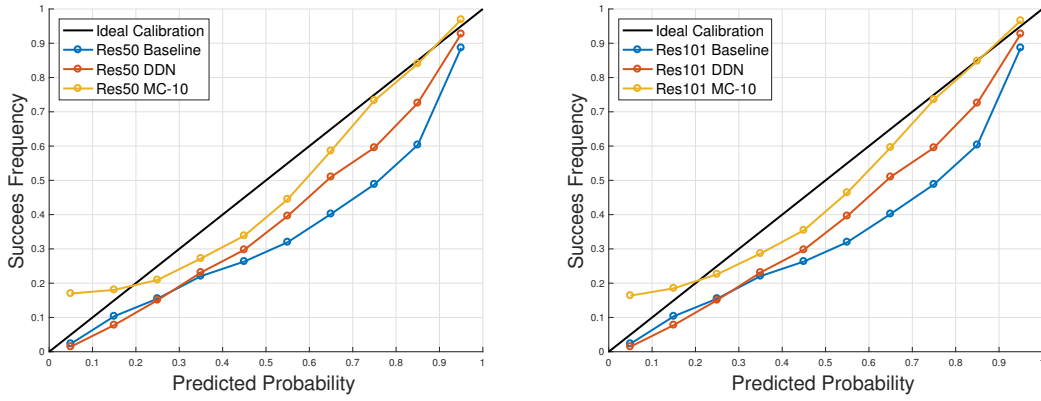
an IoU overlap between a RoI and a ground-truth bounding box of 0.5 used to compute $\mathcal{L}_{GT}$. The number of region proposals considered per image is 300. The number of training iterations to produce the baseline models is $490k$.

5.5.2 Results

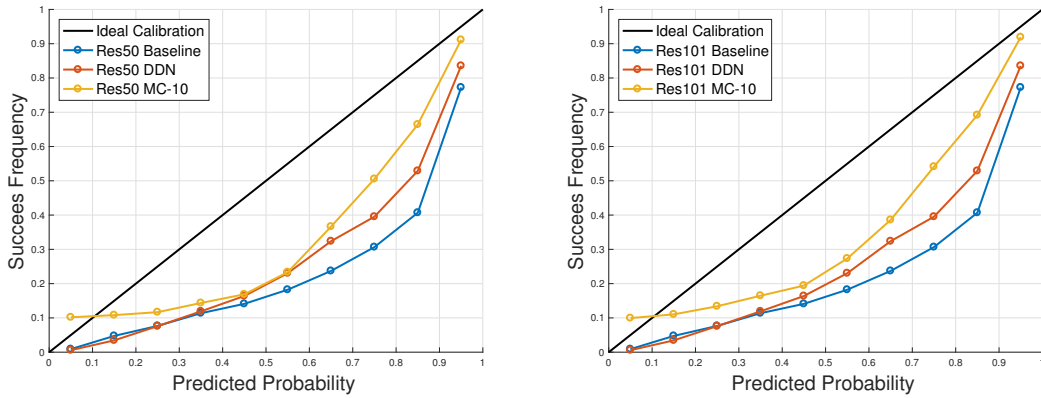
Table 5.3 shows that at a very small decrease in mean average precision (mAP) and the mean average recall (mAR), for both ResNet50 and ResNet101, DDN shows a significant improvement in MSCE and MCE scores. Both mAP and mAR are reported for all IoU overlap thresholds between 50% and 95%. The fine-tuning has been performed for an additional 1000 epochs (more fine-tuning was found to have diminishing returns) on the training and validation dataset. The evaluation results have been computed on the test dataset consisting of 5000 images. Specifically, there were a total of 85180 detection bounding boxes predicted using ResNet101 and 81317 using ResNet50. While for the classification task DDN shows an improvement in classification accuracy as well as calibration, for object detection it does not appear to make a significant difference in terms of accuracy but still shows an improvement in calibration. We believe this is due to the fact that the student model is not trained from scratch and that the training procedure requires additional losses (such as the bounding box regression loss) which also have an influence on the detection accuracy.



(a) Calibration results for an IoU threshold of 0.3.



(b) Calibration results for an IoU threshold of 0.5.



(c) Calibration results for an IoU threshold of 0.7.

Figure 5.7: Calibration plot showing the correlation between class uncertainty and (empirical) success frequency (on test data) of the baseline ResNet50 and ResNet101 models as well as DDN. MC-10 corresponds to averaging 10 Monte Carlo samples from the baseline model. The three subfigures correspond to different IoU overlap thresholds used to distinguish between true and false positive detections. Note that this IoU threshold is different than the one used for obtaining the target scores for distillation which has been set to 0.5.

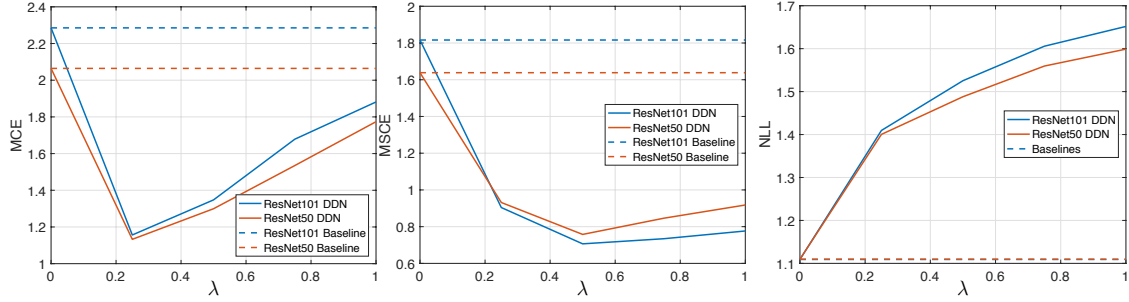


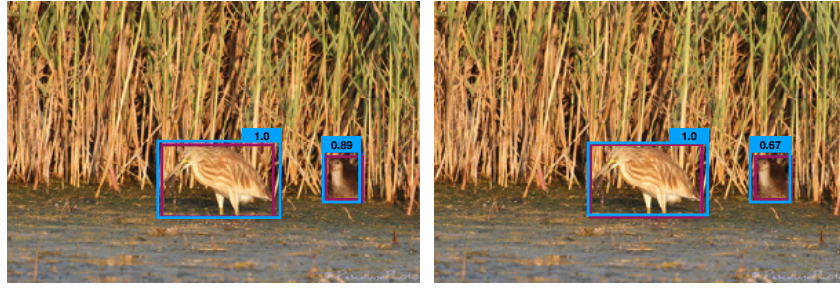
Figure 5.8: MSCE, MCE and NLL scores for different weight values λ on the distillation loss. The latter is not necessarily a good indicator of good model calibration.

Figure 5.8 shows the MSCE and MCE results for different values of the weight parameter λ on the distilled dropout loss. We also show the baseline models which do not use a distillation loss for comparison. The results in Table 5.3 are reported for $\lambda = 0.5$.

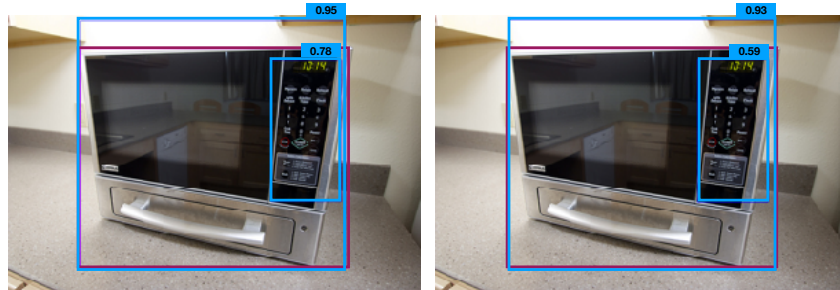
Figure 5.7 shows the calibration improvement of DDN models over the baseline Faster R-CNN models for both ResNet50 and Resnet101. Subfigures 5.7a, 5.7b and 5.7c correspond to different IoU overlap thresholds of 0.3, 0.5 resp. 0.7 used to distinguish between true and false positive detections, while the IoU threshold for obtaining the target scores for distillation which has been set to 0.5. The outcome of using the average of 10 Monte Carlo samples (MC-10) is also shown for comparison. On the object detection task, the calibration outcome (measured by MSCE) of our proposed model is less close to that of the ensemble when compared to the calibration results on the classification task. This could be due to the fact that only 10 MC-dropout samples have been used to generate the distillation training targets, while for classification the targets were generated from 100 samples.

5.5.3 Faster R-CNN Qualitative Examples

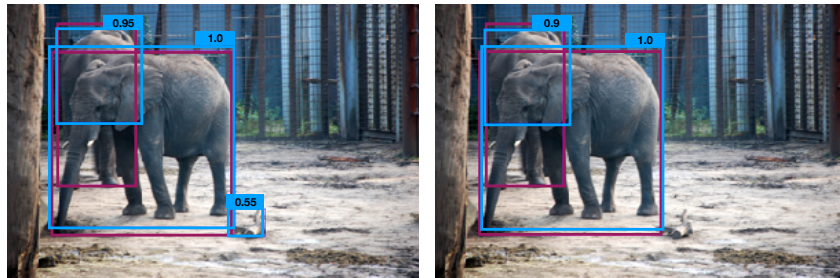
Figure 5.9 shows some example Faster R-CNN detections together with the ground truth bounding boxes using both the baseline and proposed model. Despite the region proposals being fixed, predicted bounding boxes are not identical due to bounding box regression and NMS procedure. We show both success cases such as



(a) Example detections to which DDN (right) assigns a probability of 1.0, same as the baseline (left) to the larger, centred object, while decreasing the confidence assigned to the smaller detection box which seems visually more difficult to classify correctly. In this case, the box whose confidence is decreased is a true positive detection.



(b) Example detections to which both DDN and the baseline assign a high confidence to the large, centered object. DDN also decreases the confidence assigned the more difficult (due to its fewer visual features) one. The decrease in confidence in false positive detections such as this one is the expected behaviour for the proposed model.

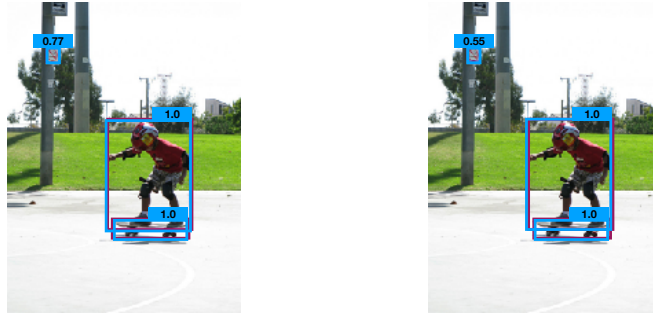


(c) Example image in which DDN maintains a high probability for the true positive detections and reduces the confidence on the false positive prediction such that it corrects a detector operating with a threshold of 0.5 on the detection scores.

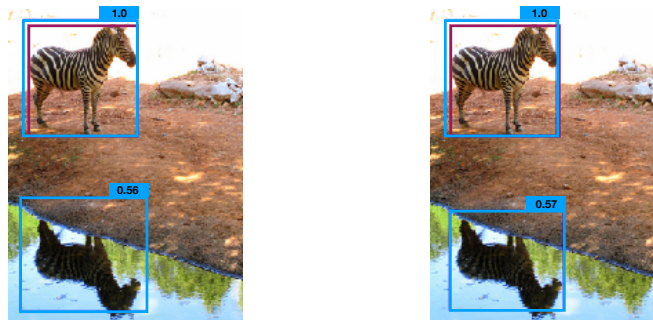
Figure 5.9: Faster R-CNN detection examples shown for a baseline model (on the left) and DDN (on the right). The magenta bounding boxes are the ground truth annotations and the blue bounding boxes are the predicted detections shown together with the class confidence score. The detection model used is ResNet50 operating with a detection threshold of 0.5.



(a) Example image where DDN assigns a lower confidence score to false positive detections (the feathers and the curtains). One of the two false positives is no longer a predictions which contributes to increasing the accuracy of a detector using a threshold of 0.5 on the bounding box predictions.



(b) A similar example of the detector decreasing its confidence in a false positive no smoking sign detection while maintaining maximum probability to the true positive detections.



(c) Example image where DDN increases its confidence on a false positive detection (failure case). This example shows an interesting classification case where the object is visually very similar to the true classification class.

Figure 5.10: Faster R-CNN detection examples shown for a baseline model (on the left) and DNN (on the right). The magenta bounding boxes are the ground truth annotations and the blue bounding boxes are the predicted detections shown together with the class confidence score. The detection model used is ResNet50 operating with a detection threshold of 0.5.

DDN lowering the confidences of false positive detections, failure cases such as doing the opposite as well as situations in which the baseline and DDN predictions are almost identical.

5.5.4 Faster R-CNN with Out-of-Distribution Data

We have considered the Central Oxford August dataset from Chapter 3, detailed in Table 3.1. The image data is significantly different than the one seen by the detector in training. The former is captured using a camera mounted on a survey vehicle driving around Oxford, while the latter consists on both indoor and outdoor images of objects captured using various cameras ([Lin et al., 2014]). We only test Faster R-CNN using the *person* class on the first labelled 8000 images of the dataset.

Figure 5.11 shows a calibration plot of the baseline models and DDN on out-of-distribution Central Oxford data.

DDN outperforms the baseline in this case as well by predicting better calibrated scores.

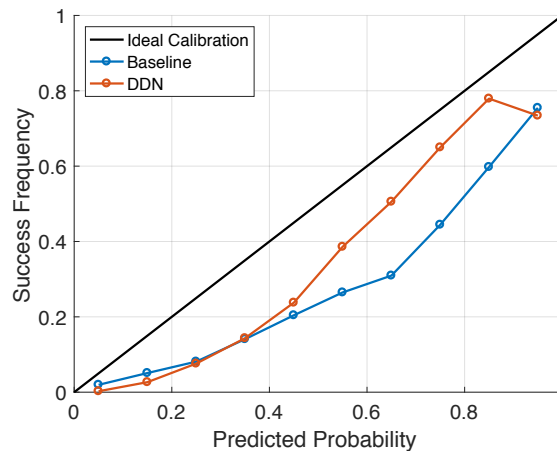


Figure 5.11: Figure shows the calibration error of the baseline and DDN models on out-of-distribution data consisting of driving imagery collected in central Oxford. While none of the detectors have seen similar data in training, DDN behaves more introspectively by having better calibrated predictions.

5.6 Discussion

5.6.1 Approximating an Ensemble

A prediction made by sampling the model space (either by using an ensemble of models or multiple MC-dropout samples) or performing full Bayesian inference takes into account the uncertainty in model parameters w . The advantage that Bayesian models have over point-estimates of the posterior predictive distribution such as MLE or MAP is that they incorporate this uncertainty into the prediction. Despite using an ensemble at training time, the model that we propose is still a point estimate that only offers an approximation to this ensemble and it might not be able to fully capture when a test sample should have high uncertainty (i.e. would be classified differently by different models in the ensemble). It remains to be investigated how DDN behaves when the test data is very different from the training data, whether it would indeed compare to Bayesian inference.

We perform a toy experiment in this regard on the CIFAR-10 classification dataset. We divide it into two clusters, one which we consider in-distribution, and use at training time, and one that is out-of-distribution and only use samples from it at test time. The clustering is performed in image space using the K-Means algorithm ([Pedregosa et al., 2011]). K-Means assigns roughly the same amount of data to each cluster in a deterministic way. At test time, we keep track of the behaviour of our models on each cluster and note that, as expected, on the data that is similar to what the model has seen in training, the model is indeed better calibrated. Figure 5.12 shows the performance of the three models, DDN, MC-100 and the baseline on the two clusters.

In future work, we hope to perform a more thorough analysis on the calibration of the proposed model in the presence of domain shift, for instance by training and testing on different datasets or differently distributed data.

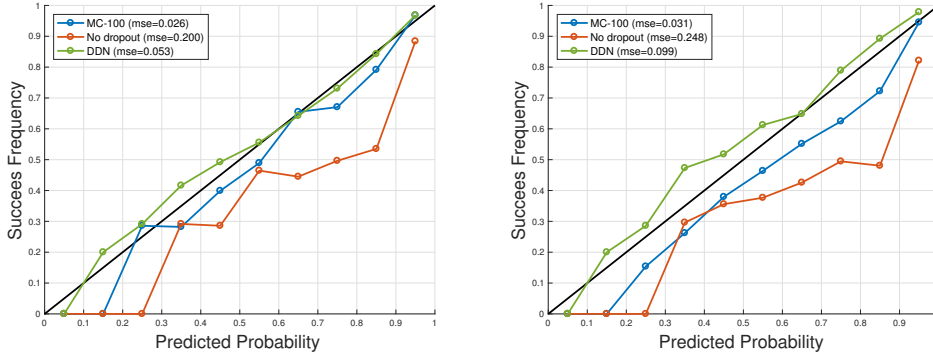


Figure 5.12: Calibration results on seen (left) and unseen (right) data from the CIFAR-10 dataset. Because test data would never be seen in training, by seen/unseen we refer to whether data from the same cluster as the test sample has been seen in training or not.

5.6.2 System I and System II

Using multiple MC-dropout samples outperforms DDN in terms of calibration. It is accurate, but more cumbersome and too slow to use in real-time applications. DDN has the advantage that it is fast, efficient, but prone to certain biases as, after all, it is a model approximation. The two approaches to inference have different strengths and could be used jointly. Assuming that in the presence of domain shift DDN’s performance is no longer competitive with that of an ensemble (due to being a point-estimate), we propose the following framework for real-time applications: Running both systems in parallel, with Monte Carlo sampling predictions running at a slower rate, and when DDN’s predictions significantly differ from the former, we simply retrain DDN. The retraining process does not require additional ground truth labels as the supervision is provided by the samples from the cumbersome model.

5.6.3 Probability Density Distillation

Rather than using the mean of the Monte Carlo samples’ output as a distillation target, we can use the parameters of the underlying distribution to generate those samples. The Dirichlet distribution is defined as a distribution over a probability simplex such a categorical model output $\sum_{i=1}^c (\sigma(f_w(\mathbf{x}_i))) = 1$, making it suitable

for our purposes. Additionally, the TensorFlow library offers an efficient way to implement this by allowing us to compute a gradient with respect to the parameters of the distribution as described in [Dillon et al., 2017]. The training would be end-to-end without the need to separately estimate the concentration parameters α . At test time we would use the mode or mean of the estimated distribution in order to make a prediction.

5.7 Conclusion

Our goal is to train neural networks that are more introspective, meaning that they are able to provide realistic estimates of their classification confidence along with the predicted class labels. Model ensembles have been shown to be more introspective than standard neural networks but are not readily fit to use in real-time applications due to their computation or memory requirements. We propose to approximate this ensemble via distillation and show that the distilled model still preserves its introspective capacity. The student-teacher framework we propose is simple - it only amounts to training an additional model (DDN) by using an additional loss term (a distilled dropout loss). We empirically demonstrate on the tasks of classification and object detection two benefits of this framework. Firstly, the confidence scores obtained are overall better calibrated - DDN models are more introspective than teacher models used for their training. Secondly, DDN shows an improvement in accuracy on the classification task, indicating that the distilled dropout loss has the potential to improve model generalisation when used within the student-teacher paradigm. We hope that future work in object detection will focus on increasing both the accuracy and the calibration of the models, rather than increasing the accuracy at the cost of having overconfident detectors that are difficult to deploy in real world applications.

6

Conclusions and Future Work

In the course of this thesis, the field of computer vision has been revolutionised by machine learning techniques, bringing us closer to deploying autonomous robots in the real world. With that, new needs regarding the reliability and safety of perception systems arise in the robotics field.

6.1 Summary of Contributions

The main contributions of this thesis are as follows:

- ***Experience-based Detectors***: We train object detectors that improve in their place of operation in a self-supervised fashion by incorporating the mistakes made in previous traversals of the same place. Using scene constraints specific to urban driving such as the intersection with the horizon line or scale

feasibility we can identify false positive detections in a self-supervised way. Our experiments establish EBC as a practical alternative to hard-negative mining which requires manually labelled data and is infeasible to use in outdoor, large scale, mobile robotics applications where the testing data distribution cannot be guaranteed to match the training distribution. While using a single, highly generalisable model for object detection remains a noble goal, our experiments show that training a collection of models such that each serves as a local expert is not only a more practical approach, but also one that currently yields better results.

- **Performance Records:** We propose a framework to gather and track place-specific performance estimates over time, in order to allow the robot to estimate the likelihood of making a mistake at test time. We treat object detectors as black-boxes and model their output as observations from a Bernoulli distribution (*success* or *failure*) that update the parameters of a Beta distribution indicative of the performance at a particular location in the robot’s map. Our experiments show two different approaches for incorporating observations into a performance record: one that only relies on spatial proximity to a location of interest (LOC) and one that relies on both spatial proximity and visual appearance of the location of interest (APP). In a use-case scenario regarding offering or denying autonomy, we found that the latter leads to taking more correct decisions. More specifically, we show that we can reduce the number of mistakes made by an object detector by denying autonomy when its confidence levels are low.
- **Distilled Dropout Network:** We introduce an efficient way to output well-calibrated, reliable, uncertainty scores from neural networks. The Distilled Dropout Network (DDN) makes standard (non-Bayesian) neural networks more introspective by adding a new training loss which prevents them from being overconfident. Our method is more efficient than Bayesian neural

networks or model ensembles which, despite being well-calibrated, are more cumbersome to train and slower to test. We evaluate DDN on the the task of image classification on the CIFAR-10 dataset and show that our calibration results are competitive even when compared to 100 Monte Carlo samples from a dropout network while they also increase the classification accuracy. We also propose better calibration within the state of the art Faster R-CNN object detection framework and show, using the COCO dataset, that DDN helps train more introspective object detectors.

6.2 Future Work

There are multiple research directions to be explored in the areas of performance prediction, introspection and ultimately trust a user places in a perception systems as follow-up works to the ideas introduced in this thesis. We briefly describe a few of them.

6.2.1 Predicting Detection Failure (using appearance only)

Failing to detect an object of interest most often means misclassifying a high dimensional feature representation. Learning a new function f_{trust} of that feature representation to predict the success or failure of the main predictor f_{det} is illustrated in Figure 6.1. While f_{det} is an object detector of choice, f_{trust} is a proposed trust model which learns a failure mask outlining the image features that lead to incorrect classifications.

This approach might require distinguishing between different detection failure modes some of which mentioned in Section 3.5.2, such as false positives and false negatives or catastrophic and non-catastrophic in order to avoid learning a mixture of effects. We hope f_{trust} picks up on clues from the environment such as tunnels, signs of roadworks or sudden weather changes that are interpretable to a human user.

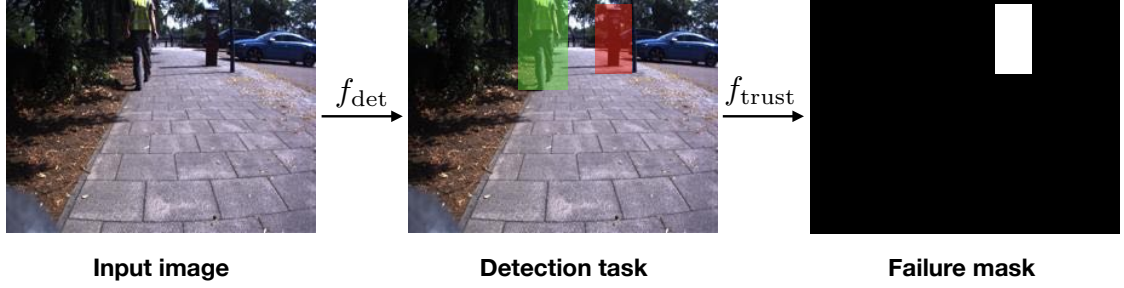


Figure 6.1: Proposed Trust Model: learning a function f_{trust} of the correct (shown in green) and incorrect (shown in red) outcome of an object detector f_{det} .

6.2.2 Anticipating Perception Failure in Advance

While performance records give us an indication of which parts of a mapped area the robot will have a low performance on, one of the methods proposed (APP) requires the robot to first localise in order to know which past experiences are more indicative of performance. Under the current framework predictions are made in real-time and there is little chance to alert the user in time to take any action. Our experiments in Chapter 4 show that perception failure can be learnt as a function of a place and its appearance, while in Section 6.2.1 we propose to learn it as a function of the input image. We take this problem setting even further and propose training a system that learns, in a supervised way, to predict a *time to failure* instead of estimating where a system fails.

6.2.3 Retraining Expert Detectors According to Performance Records

In Chapter 3 we fit object detectors to two routes and three different seasons and show that such place-specific object detectors outperform a generic one that is fitted on all of the data. We relied on the assumption that location and seasonal effects influence the performance of the detector, which we show that is indeed true in Chapter 4. However, the spatial and temporal granularity of training a local expert has not been discussed. When should we train new detectors? Future work

can investigate the usage of performance records to automatically infer when and where to train a new model (i.e. places where detector performance is consistently low). Luckily, using an autonomous vehicle means that we can actively guide the data collection process. We propose to make use of this advantage over static datasets, go back to places of low performance and collect more data in order to retrain an expert detector.

6.2.4 Testing DDN on Out-of-Distribution Data

As discussed in Section 5.6.1, DDN is still a Bayesian approximation whose capacity to estimate good confidence levels on out of distribution data is still to be explored. Firstly, an experiment we propose is to adjust the image space slowly such that we know where the breaking point between DDN and the ensemble formed by multiple dropout samples is. For instance, by gradually changing the hsv values. This breaking point is something that, when testing our proposed framework with domain shift (i.e. a different dataset with an unknown distribution) can not establish. Secondly, propose to use both models (the teacher and the student) on a robotics dataset which has an additional constraint on the time it takes to process an image frame. This would allow us to see how often we need to retrain, in practice, the student model and whether the method we propose is indeed feasible for real-time applications.

Finally, we believe it is possible to learn a model that acts as both student and teacher, replacing the need to train a separate model. Through self-improvement, the model refines its own confidence predictions, whenever it disagrees with multiple samples from itself at training time. This approach will perhaps require that at the beginning the model only trusts the ground truth samples.

6.3 Closing Remarks

The visual appearance of a robot's workspace has a significant influence on its perception performance. By explicitly accounting for such variability as part of a typical learning framework, we can build more reliable, introspective systems that are more likely to be deployed in real world applications.

Bibliography

- [Ba and Caruana, 2014] Ba, J. and Caruana, R. (2014). Do deep nets really need to be deep? In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2654–2662. Curran Associates, Inc.
- [Badrinarayanan et al., 2015] Badrinarayanan, V., Kendall, A., and Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*.
- [Benenson et al., 2012] Benenson, R., Mathias, M., Timofte, R., and Gool, L. V. (2012). Pedestrian detection at 100 frames per second. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2903–2910.
- [Benenson et al., 2013a] Benenson, R., Mathias, M., Tuytelaars, T., and Van Gool, L. (2013a). Seeking the strongest rigid detector. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '13*, pages 3666–3673, Washington, DC, USA. IEEE Computer Society.
- [Benenson et al., 2013b] Benenson, R., Mathias, M., Tuytelaars, T., and Van Gool, L. (2013b). Seeking the strongest rigid detector. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '13*, pages 3666–3673, Washington, DC, USA. IEEE Computer Society.
- [Blundell et al., 2015] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *Proceedings of the 32Nd*

- International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1613–1622. JMLR.org.
- [Bradski, 2000] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.
- [Cai et al., 2016] Cai, Z., Fan, Q., Feris, R. S., and Vasconcelos, N. (2016). A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision*, pages 354–370. Springer.
- [Chen and Gupta, 2017] Chen, X. and Gupta, A. (2017). An implementation of faster rcnn with study for region sampling. *arXiv preprint arXiv:1702.02138*.
- [Churchill and Newman, 2013a] Churchill, W. and Newman, P. (2013a). Experience-based navigation for long-term localisation. *The International Journal of Robotics Research*, 32(14):1645–1661.
- [Churchill and Newman, 2013b] Churchill, W. and Newman, P. (2013b). Experience-based navigation for long-term localisation. *International Journal of Robotics Research (IJRR)*, 32(14):1645–1661.
- [Churchill et al., 2015] Churchill, W., Tong, C. H., Gurau, C., Posner, I., and Newman, P. (2015). Know Your Limits: Embedding Localiser Performance Models in Teach and Repeat Maps. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.
- [Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Mach. Learn.*, 20(3):273–297.
- [Cummins and Newman, 2011] Cummins, M. and Newman, P. (2011). Appearance-only slam at large scale with fab-map 2.0. *The International Journal of Robotics Research*, 30(9):1100–1123.

- [Curran et al., 2007] Curran, J. R., Murphy, T., and Scholz, B. (2007). Minimising semantic drift with mutual exclusion bootstrapping. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, pages 172–180.
- [Daftry et al., 2016] Daftry, S., Zeng, S., Bagnell, J. A., and Hebert, M. (2016). Introspective perception: Learning to predict failures in vision systems. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 1743–1750. IEEE.
- [Dai et al., 2016] Dai, J., Li, Y., He, K., and Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387.
- [Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, CA, USA.
- [Dayoub et al., 2015] Dayoub, F., Dunbabin, M., and Corke, P. (2015). Robotic detection and tracking of crown-of-thorns starfish. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1921–1928. IEEE.
- [DeGroot and Fienberg, 1983] DeGroot, M. H. and Fienberg, S. E. (1983). The comparison and evaluation of forecasters. *The Statistician: Journal of the Institute of Statisticians*, 32:12–22.
- [Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Fei-fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, Miami, Florida, USA.

- [Dequaire et al., 2016] Dequaire, J., Tong, C. H., Churchill, W., and Posner, I. (2016). Off the beaten track: Predicting localisation performance in visual teach and repeat. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden.
- [Dietterich, 2000] Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg. Springer Berlin Heidelberg.
- [Dillon et al., 2017] Dillon, J. V., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., Patton, B., Alemi, A., Hoffman, M. D., and Saurous, R. A. (2017). Tensorflow distributions. *CoRR*, abs/1711.10604.
- [Dollar et al., 2014] Dollar, P., Appel, R., Belongie, S., and Perona, P. (2014). Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1532–1545.
- [Dollár et al., 2014] Dollár, P., Appel, R., Belongie, S., and Perona, P. (2014). Fast feature pyramids for object detection. *PAMI*.
- [Dollár et al., 2014] Dollár, P., Appel, R., Belongie, S., and Perona, P. (2014). Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1532–1545.
- [Dollar et al., 2010] Dollar, P., Belongie, S., and Perona, P. (2010). The fastest pedestrian detector in the west. In *Proceedings of the British Machine Vision Conference*, pages 68.1–68.11. BMVA Press. doi:10.5244/C.24.68.
- [Dollár et al., 2009] Dollár, P., Wojek, C., Schiele, B., and Perona, P. (2009). Pedestrian detection: A benchmark. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 304–311. IEEE.

- [Dollar et al., 2012] Dollar, P., Wojek, C., Schiele, B., and Perona, P. (2012). Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761.
- [Dutta et al., 2015] Dutta, A., Veldhuis, R., and Spreeuwens, L. (2015). Predicting face recognition performance using image quality. *arXiv preprint arXiv:1510.07119*.
- [Everingham et al., 2006] Everingham, M., Zisserman, A., Williams, C. K. I., and Van Gool, L. (2006). The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results.
- [Fan et al., 2008] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research (JMLR)*, 9:1871–1874.
- [Felzenszwalb et al., 2010a] Felzenszwalb, P., Girshick, R., McAllester, D., and Ramanan, D. (2010a). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(9):1627–1645.
- [Felzenszwalb et al., 2010b] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010b). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645.
- [Fu et al., 2017] Fu, C., Liu, W., Ranga, A., Tyagi, A., and Berg, A. C. (2017). DSSD : Deconvolutional single shot detector. *CoRR*, abs/1701.06659.
- [Furgale and Barfoot, 2010] Furgale, P. and Barfoot, T. D. (2010). Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics*, 27(5).
- [Gal, 2016] Gal, Y. (2016). *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge.

- [Gal and Ghahramani, 2015] Gal, Y. and Ghahramani, Z. (2015). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *arXiv:1506.02142*.
- [Gal et al., 2017] Gal, Y., Hron, J., and Kendall, A. (2017). Concrete Dropout. In *arXiv:1705.07832*.
- [Ganin and Lempitsky, 2015] Ganin, Y. and Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1180–1189.
- [Geiger et al., 2013] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237.
- [Girshick, 2015] Girshick, R. (2015). Fast r-cnn. In *International Conference on Computer Vision (ICCV)*.
- [Girshick et al., 2014] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [Girshick et al., 2011] Girshick, R. B., Felzenszwalb, P. F., and McAllester, D. A. (2011). Object detection with grammar models. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 24*, pages 442–450. Curran Associates, Inc.
- [Grimmett et al., 2016] Grimmett, H., Triebel, R., Paul, R., and Posner, I. (2016). Introspective classification for robot perception. *The International Journal of Robotics Research*, 35(7):743–762.
- [Guo et al., 2017] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. *CoRR*, abs/1706.04599.

- [Gurari et al., 2016] Gurari, D., Jain, S., Betke, M., and Grauman, K. (2016). Pull the plug? predicting if computers or humans should segment images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 382–391.
- [He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.
- [He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition.
- [Henriques et al., 2013] Henriques, J., Carreira, J., Caseiro, R., and Batista, J. (2013). Beyond hard negative mining: Efficient detector learning via block-circulant decomposition. In *2013 IEEE International Conference on Computer Vision (ICCV)*, pages 2760–2767.
- [Hinton et al., 2015] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.
- [Hoiem et al., 2012] Hoiem, D., Chodpathumwan, Y., and Dai, Q. (2012). Diagnosing error in object detectors. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part III, ECCV’12*, pages 340–353, Berlin, Heidelberg. Springer-Verlag.
- [Holub et al., 2008] Holub, A., Perona, P., and Burl, M. C. (2008). Entropy-based active learning for object recognition. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW’08. IEEE Computer Society Conference on*, pages 1–8. IEEE.
- [Horn and Perona, 2017] Horn, G. V. and Perona, P. (2017). The devil is in the tails: Fine-grained classification in the wild. *CoRR*, abs/1709.01450.

- [Hsieh et al., 2008] Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S. S., and Sundararajan, S. (2008). A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th International Conference on Machine Learning, ICML '08*, pages 408–415, New York, NY, USA. ACM.
- [Huang et al., 2017] Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
- [Kalal et al., 2012] Kalal, Z., Mikolajczyk, K., and Matas, J. (2012). Tracking-learning-detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(7):1409–1422.
- [Kanezaki et al., 2014] Kanezaki, A., Inaba, S., Ushiku, Y., Yamashita, Y., Murakami, H., Kuniyoshi, Y., and Harada, T. (2014). Hard negative classes for multiple object detection. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3066–3073, Hong Kong, China.
- [Kapoor et al., 2010] Kapoor, A., Grauman, K., Urtasun, R., and Darrell, T. (2010). Gaussian processes for object categorization. *International Journal of Computer Vision*, 88(2):169–188.
- [Keller et al., 2011] Keller, C., Enzweiler, M., and Gavrila, D. (2011). A new benchmark for stereo-based pedestrian detection. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 691–696.
- [Kendall and Gal, 2017] Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? *arXiv preprint arXiv:1703.04977*.
- [Khosla et al., 2012] Khosla, A., Zhou, T., Malisiewicz, T., Efros, A. A., and Torralba, A. (2012). Undoing the damage of dataset bias. In *European Conference on Computer Vision*, pages 158–171.

- [Krizhevsky, 2009] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report.
- [Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.
- [Kulis et al., 2011] Kulis, B., Saenko, K., and Darrell, T. (2011). What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *CVPR 2011*, pages 1785–1792.
- [Ladický et al., 2015] Ladický, L., Jeong, S., Solenthaler, B., Pollefeys, M., and Gross, M. (2015). Data-driven fluid simulations using regression forests. *ACM Trans. Graph.*, 34(6):199:1–199:9.
- [Lakshminarayanan et al., 2016] Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2016). Simple and scalable predictive uncertainty estimation using deep ensembles.
- [Lin et al., 2017] Lin, T., Goyal, P., Girshick, R. B., He, K., and Dollár, P. (2017). Focal loss for dense object detection. *CoRR*, abs/1708.02002.
- [Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham. Springer International Publishing.
- [Linegar, 2016] Linegar, C. (2016). *Vision-Only Localisation Under Extreme Appearance Change*. PhD thesis, University of Oxford, Oxford, United Kingdom.
- [Linegar et al., 2015] Linegar, C., Churchill, W., and Newman, P. (2015). Work Smart, Not Hard: Recalling Relevant Experiences for Vast-Scale but Time-Constrained Localisation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, USA.

- [Liu et al., 2016] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C., and Berg, A. C. (2016). SSD: single shot multibox detector. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I*, pages 21–37.
- [Maddern et al., 2017] Maddern, W., Pascoe, G., Linegar, C., and Newman, P. (2017). 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15.
- [Malisiewicz et al., 2011] Malisiewicz, T., Gupta, A., and Efros, A. A. (2011). Ensemble of exemplar-svms for object detection and beyond. In *ICCV*.
- [McAllister et al., 2017] McAllister, R., Gal, Y., Kendall, A., van der Wilk, M., Shah, A., Cipolla, R., and Weller, A. (2017). Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning.
- [McManus et al., 2013] McManus, C., Furgale, P., Stenning, B., and Barfoot, T. D. (2013). Lighting-invariant visual teach and repeat using appearance-based lidar. *Journal of Field Robotics*, 30(2):254–287.
- [Merat et al., 2014] Merat, N., Jamson, A. H., Lai, F. C., Daly, M., and Carsten, O. M. (2014). Transition to manual: Driver behaviour when resuming control from a highly automated vehicle. *Transportation research part F: traffic psychology and behaviour*, 27:274–282.
- [Mund et al., 2015] Mund, D., Triebel, R., and Cremers, D. (2015). Active online confidence boosting for efficient object classification. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1367–1373.
- [Nelson et al., 2016] Nelson, P., Linegar, C., and Newman, P. (2016). Building, curating, and querying large-scale data repositories for field robotics applications. In *Field and Service Robotics*, pages 517–531. Springer.

- [Ondruska and Posner, 2014] Ondruska, P. and Posner, I. (2014). Probabilistic attainability maps: Efficiently predicting driver-specific electric vehicle range. In *IEEE Intelligent Vehicles Symposium (IV)*, Dearborn, MI, USA.
- [Paton et al., 2017] Paton, M., Pomerleau, F., MacTavish, K., Ostafew, C. J., and Barfoot, T. D. (2017). Expanding the limits of vision-based localization for long-term route-following autonomy. *Journal of Field Robotics*, 34(1):98–122.
- [Pawlowski et al., 2017] Pawlowski, N., Rajchl, M., and Glocker, B. (2017). Implicit weight uncertainty in neural networks. *arXiv preprint arXiv:1711.01297*.
- [Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- [Peng et al., 2017] Peng, X., Usman, B., Kaushik, N., Hoffman, J., Wang, D., and Saenko, K. (2017). Visda: The visual domain adaptation challenge.
- [Pereyra et al., 2017] Pereyra, G., Tucker, G., Chorowski, J., Kaiser, L., and Hinton, G. E. (2017). Regularizing neural networks by penalizing confident output distributions. *CoRR*, abs/1701.06548.
- [Peynot et al., 2010] Peynot, T., Scheduling, S., and Terho, S. (2010). The marulan data sets: Multi-sensor perception in a natural environment with challenging conditions. *The International Journal of Robotics Research*, 29(13):1602–1607.
- [Peynot et al., 2009] Peynot, T., Underwood, J., and Scheduling, S. (2009). Towards reliable perception for unmanned ground vehicles in challenging conditions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1170–1176.

- [Ramanagopal et al., 2017] Ramanagopal, M. S., Anderson, C., Vasudevan, R., and Johnson-Roberson, M. (2017). Failing to learn: Autonomously identifying perception failures for self-driving cars. *CoRR*, abs/1707.00051.
- [Ranzato et al., 2006] Ranzato, M., Poultney, C., Chopra, S., and LeCun, Y. (2006). Efficient learning of sparse representations with an energy-based model. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS’06, pages 1137–1144, Cambridge, MA, USA. MIT Press.
- [Razavian et al., 2014] Razavian, A. S., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). CNN features off-the-shelf: an astounding baseline for recognition. *CoRR*, abs/1403.6382.
- [Redmon et al., 2016] Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 779–788.
- [Redmon and Farhadi, 2016] Redmon, J. and Farhadi, A. (2016). YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242.
- [Redmon and Farhadi, 2018] Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. Technical report.
- [Ren et al., 2017a] Ren, S., He, K., Girshick, R., and Sun, J. (2017a). Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149.
- [Ren et al., 2017b] Ren, S., He, K., Girshick, R. B., and Sun, J. (2017b). Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6):1137–1149.

- [Scholkopf and Smola, 2001] Scholkopf, B. and Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA.
- [Shalev-Shwartz and Tewari, 2011] Shalev-Shwartz, S. and Tewari, A. (2011). Stochastic methods for l_1 -regularized loss minimization. *J. Mach. Learn. Res.*, 12:1865–1892.
- [Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.
- [Stewart, 2014] Stewart, A. (2014). Localisation using the appearance of prior structure.
- [Sung and Poggio, 1998] Sung, K.-K. and Poggio, T. (1998). Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51.
- [Sutskever et al., 2013] Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA. PMLR.
- [Torr and Zisserman, 2000] Torr, P. H. S. and Zisserman, A. (2000). Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78:138–156.
- [Torralba and Efros, 2011] Torralba, A. and Efros, A. A. (2011). Unbiased look at dataset bias. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1521–1528. IEEE.

- [Triebel et al., 2013] Triebel, R., Grimmett, H., and Posner, I. (2013). Confidence boosting: Improving the introspectiveness of a boosted classifier for efficient learning. In *Workshop on Autonomous Learning. IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany.
- [Tzeng et al., 2016] Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. (2016). Adversarial discriminative domain adaptation. In *Advances in neural information processing systems (NIPS) workshop on adversarial learning*.
- [Wang and Posner, 2015] Wang, D. Z. and Posner, I. (2015). Voting for voting in online point cloud object detection. In *Proceedings of Robotics: Science and Systems*, Rome, Italy.
- [Wang et al., 2017] Wang, Y., Ramanan, D. K., and Hebert, M. (2017). Learning to model the tail. In *31st Conference on Neural Information Processing Systems (NIPS)*.
- [Yosinski et al., 2014] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems, NIPS’14*, pages 3320–3328, Cambridge, MA, USA. MIT Press.
- [Zhang et al., 2014] Zhang, P., Wang, J., Farhadi, A., Hebert, M., and Parikh, D. (2014). Predicting failures of vision systems. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [Zhu et al., 2016] Zhu, X., Vondrick, C., Fowlkes, C. C., and Ramanan, D. (2016). Do we need more training data? *Int. J. Comput. Vision*, 119(1):76–92.