MONASH University

Representation Learning and Semantics for Robotic Vision

Benjamin J. Meyer

Supervisor: Prof. Tom Drummond
Associate Supervisor: A. Prof. Lindsay Kleeman

A thesis submitted for the degree of Doctor of Philosophy at Monash University in 2019.

ARC Centre of Excellence for Robotic Vision,
Department of Electrical and Computer Systems Engineering.

Copyright Notice

Abstract

Vision is a powerful tool in enabling intelligent robotic systems to safely and effectively perform useful tasks. The learning of visual representations and extraction of semantic information can facilitate an understanding of the environment. This understanding is an important precursor to well-considered robotic action. There have been significant advances is the field of machine learning for vision in recent years, but robotic considerations are rare. In this thesis, several important problem domains for robotic vision are identified and explored. Learning methodologies are developed with the needs of robotic applications in mind. Although motivated from a robotic vision perspective, the proposed approaches have applications far beyond, with advantages demonstrated over existing methods on conventional computer vision problems.

Firstly, the problems of feature embedding learning and image classification are addressed by the development of a novel deep metric learning method. The proposed approach not only allows classification of new examples, but also transfers to novel classes that are withheld during training. Open set problems are then investigated, acknowledging that the true distribution of data will differ from the training distribution. Novelty detection and open set recognition using deep metric spaces is explored, allowing a classifier to detect unknown examples, rather than silently failing with incorrect predictions. Recognising that learning should not cease after initial model training, open set active learning is then investigated. A novel approach is proposed that allows a model to improve its understanding of the environment by efficiently querying for labels of unknown observations. Finally, the problem of semantic segmentation is explored. Identifying that object-level performance is often more important than pixel-level performance for robotic applications, a novel semantic segmentation system is proposed that significantly penalises false object detections. Thorough experimental evaluation of each proposed method demonstrates significant advantages over existing baselines and state-of-the-art approaches in the domains of classification, transfer learning, novelty detection, active learning and semantic segmentation.

Declaration of Authorship

This thesis is an original work of my research and contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Print Name: Benjamin J. Meyer

Date: 07/05/2019

Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Tom Drummond for his guidance and assistance throughout my candidature. I also thank my associate supervisor A. Prof. Lindsay Kleeman for his support during both my undergraduate and postgraduate studies. I am grateful to Prof. Chris McCool and A. Prof. Stephen Gould for reviewing this thesis and providing valuable feedback. Finally, special thanks to past and present lab mates, as well as family and friends, for their support and encouragement.

This research was supported by an Australian Government Research Training Program (RTP) Scholarship and the Australian Research Council Centre of Excellence for Robotic Vision (project number CE140100016).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Vision is arguably our most dominant sense. The rich information provided by sight allows us to safely navigate the world, understand our surroundings and interact with the environment. Much of our cognitive and motor responses are the direct result of inferring meaning from what we see. Research suggests that vision plays a part in up to 80% of our perception, learning and cognition [1]. Everyday actions such as walking down the street, driving a car or picking items off a shelf are largely mediated and facilitated by sight.

Sensing and understanding the environment are key requirements for many robotic applications. Given our reliance on the sense of vision, it is reasonable to pursue robotic vision as a key technology in facilitating this sensing and understanding. Complex robotic tasks such as driving a car [2], navigating a corridor [3] or picking and placing items in a warehouse [4], require rich information about the environment. This information includes the structure of the environment, the surrounding terrain, any objects present in the scene and the actions of dynamic objects such as humans, animals and vehicles. Vision provides a means to obtain and understand such information, enabling robots to carry out complex tasks and be truly useful to humans.

Robotic vision begins with the sensor that captures the required information. These sensors range from standard colour cameras to information rich multi-spectral cameras and light-field cameras. Common sensor set-ups include those which provide depth information to the robot, in addition to standard colour information [5, 6]. Passive stereo cameras allow depth to be inferred using triangulation techniques between matched pixels. Low-cost active depth sensors use structured light patterns to achieve the same result. Another sensor type is an event camera, also known as a dynamic vision sensor [7]. Unlike conventional cameras, which generally output a per-pixel intensity value for each frame, dynamic vision sensors only output

changes, or events, that occur at a given pixel location. In this thesis, vision is used to refer to the processing of information obtained by a standard monocular, colour camera. The methodologies presented, however, have clear extensions to more complicated sensor arrangements, particularly those which provide depth information.

Machine learning refers to the problem of enabling a machine to learn from data, without being explicitly programmed for a specific task. The ability to learn from visual data is an imperative step towards truly useful and reliable robotic vision. Visual data allows robots to navigate the environment by building maps and localising itself within the map. This problem, known as simultaneous localisation and mapping (SLAM), is incredibly useful in robotics, but without the ability to learn from visual data, a robot's understanding of the environment will be limited to structure and location. Machine learning facilitates a semantic understanding of the environment. In this context, *semantics* refers to the meaning afforded to objects, structures and actions in the physical world. Examples of this understanding include the ability to detect and recognise objects, which are key in facilitating robotic interaction with the environment. Further, a semantic understanding may incorporate the structure and terrain of the environment. This includes recognising road, grass, carpet, water, walls and so on, which impact the robot's navigation and path planning. Learning from visual data also allows meaningful interaction with humans, including the ability of a robot to recognise faces, recognise human actions and predict what a human is likely to do next. An additional ability facilitated by learning from visual data, is inferring meaning about objects that the robot has never before observed. Humans are very adept at this type of knowledge transfer. We are able to infer an unknown object's affordances by using knowledge of similar object categories. This may be achieved through analysing the object's attributes or by recognising the parent category to which the unknown sub-category belongs.

There has been a significant research focus on learning from visual data in the computer vision community. These works seldom make any considerations for robotic applications. Robotic vision presents numerous challenges that are not often addressed in pure computer vision solutions. An incomplete list of example challenges that are present in robotic vision problems is as follows:

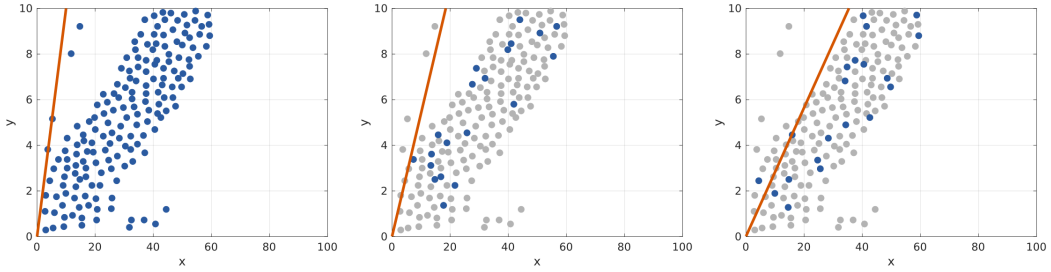
- Robots operate in dynamic environments.

- Intelligent robots need to complete several different tasks with limited compute resources.
- Mistakes in predictions have real-world consequences and can trigger undesirable robotic action.
- A robot needs to know when it doesn't know, rather than silently fail.
- Robots operate in a temporal world and can remember things they have seen before.
- A robot may need to recognise many classes with limited training data.
- Robots should be able to update their understanding on-the-fly, without costly model refinement.

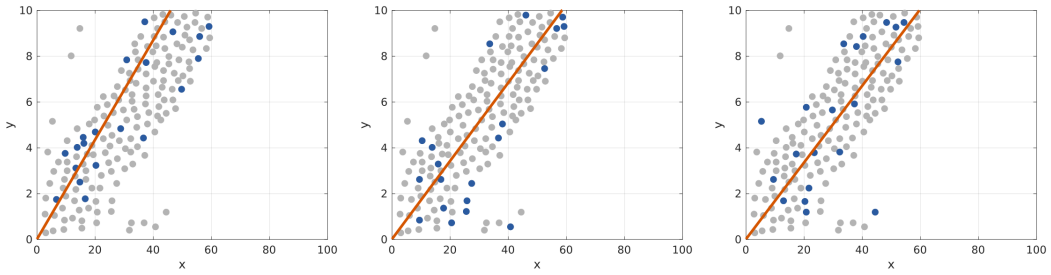
The research presented in this thesis aims to bridge part of the disconnect between computer vision and robotic applications. The computer vision problems approached are done so with robotic considerations in mind. This does not mean that the contributions of this thesis are limited to the robotics community. In fact, it is found that approaching problems with robotic considerations often leads to advantages beyond the application to robotics, improving on existing solutions in the wider computer vision field. All presented methodologies are evaluated against standard computer vision approaches on the conventional computer vision datasets. The application to robotic vision is the broad consideration of robotic needs that inspire and motivate much of the work presented in this thesis.

It is infeasible to consider all possible robotic conditions. Key robotic considerations that are not made in this thesis are mentioned here, so that readers may give thought to how such considerations may be made in future work. The work in this thesis considers only static images, not videos, and as such the temporal element of robotics is not considered in this manner. Temporal information is considered, however, in the sense that some of the proposed approaches allow the learning model to remember previously observed instances. The sensor used for all proposed methodologies is monocular, meaning there are clear extensions to the incorporation of depth information. The agency of robots, that is, the ability of a robot to *poke* or change the environment in which it operates, is not explored.

The remainder of this chapter gives a high-level introduction to the various



(A) Initial, cost = 818.32. (B) Step 1, cost = 168.63. (C) Step 2, cost = 19.86.



(D) Step 3, cost = 6.91. (E) Step 4, cost = 3.64. (F) Step 5, cost = 3.60.

FIGURE 1.1: Simple linear regression machine learning example. At each training step, a subset of data is sampled and the parameters are updated such that the cost reduces.

problems considered in this thesis. Starting with a broad overview of machine learning for vision in Section 1.1, Sections 1.2-1.6 introduce and motivate the specific areas of learning that are addressed in the subsequent chapters. A summary of the contributions of this thesis is given in Section 1.7, publications arising from the presented research are listed in Section 1.8 and the remainder of the thesis is outlined in Section 1.9.

1.1 Machine Learning from Visual Data

Machine learning is based on the concept that machines should be able to learn from data without being explicitly programmed for a specific learning task. Algorithms for machine learning are data driven; a model is presented with large amounts of data and its internal parameters are updated such that some function is optimised. This function is often referred to as an objective, cost or loss function. A simple example is shown in Figure 1.1. In this problem, the goal is to find a linear regression line that fits the global population of data. An approximation to the best fit is found since it may not be feasible to achieve the exact solution with the available compute. The algorithm

sub-samples data points and updates the parameters, which in this case are a and b from the line equation $y = ax + b$. After several updates the parameters converge. The cost function is the squared difference between the predicted values and the true values, that is, $cost = \sum_{i=1}^{N}(y_i - (ax_i + b))^2$, where N is the number of data points. If the model is well learned it should generalise to data that was unseen during training. That is, given a new value of x, the learned model should give a reasonable prediction of the associated value of y.

In this thesis, the data of interest is images and the problems are more complex than linear regression. An example problem is categorising images based on the semantic information contained within. In this case, the images are the inputs to the model and the semantic categories are the target values that the algorithm should learn to predict. Machine learning algorithms can be broadly classified into three main categories: unsupervised learning, supervised learning and reinforcement learning.

Unsupervised Learning comprises algorithms that learn from input data only. In vision, this data may be a large collection of images or videos. Unsupervised algorithms are given no target values; the input data is completely unlabelled. Common examples of unsupervised learning algorithms are those that cluster data, such as k-means [8] and Gaussian mixture models.

Supervised Learning comprises algorithms that learn not only from input data, but also target values associated with the inputs. Target values are often referred to as *labels* and hence the inputs are referred to as *labelled data*. Examples of labels for images include image-level semantic categories, or classes, as well as class labels for each pixel in an input image. Regression and classification are two key supervised learning problems. Supervised learning encompasses subcategories of learning algorithms including semi-supervised learning, in which the model learns from both labelled and unlabelled data, and weakly-supervised learning, in which the data labels are incomplete, noisy or only partially representative of the desired target values.

Reinforcement Learning is a class of algorithms and models that attempt to learn appropriate actions to take in order to maximise some reward. A process of sequential decision making, reinforcement learning is useful for problems in which training data is difficult or infeasible to acquire. This may

include a robotic arm learning how to grasp objects or perform other dexterous actions with the environment.

The research in this thesis focuses on supervised learning algorithms. In Sections 1.1.1 and 1.1.2, two common machine learning models are briefly introduced. Detailed review and discussion follows in Chapters 2 and 3.

1.1.1 Neural Networks

A commonly used model in machine learning is the neural network, which aims to very loosely model the behaviour of neurons in an animal's brain. Large numbers of artificial neurons [9], intended to vaguely emulate biological neurons, are connected together by artificial synapses, which transfer signals from one artificial neuron to another. An artificial neuron processes a received signal in some manner and transmits a subsequent signal to be processed by other artificial neurons. Generally in artificial neural networks, connections are directed and the signal processing involves a simple operation such as multiplication with a learned model parameter. Non-linearity is commonly introduced with activation functions that take the modified signal and input it to a non-linear operation, such as a sigmoid function. Although usually the case, an activation function is not required to be non-linear. Activation functions are loosely analogous to the level of action potential firing in biological neural networks. The model parameters, or *weights*, are learned by optimisation methods, involving the sampling of examples from a training dataset. This is discussed in detail in Section 3.1.

By far the most commonly used machine learning model in computer vision in recent years is the convolutional neural network (CNN) [10, 11]. The flexibility of these powerful models have allowed them to be applied to a vast array of machine learning problems. The structure of CNNs makes them particularly suitable for visual data, such as images. In a simple layout, a basic CNN consists of a series of convolutional filters that extract increasingly high-level information from the input data. The convolution weights are learned parameters of the model. As with standard artificial neural networks, the convolution outputs, or *activations*, are transformed by activation functions. The outputs of the activation functions are then transmitted to other convolutional layers for further processing. Activations are often referred to as *features*, as they should in some way describe the contents of the input data. For images, shallow-layer features, that is, those early in the

network, generally describe structural and low-level elements in the image, such as edges and colour. Deep-layer features, that is, those extracted after several consecutive convolutions, often describe higher-level semantic information. The deep, information rich features can be utilised for several useful tasks, such as classifying the input image. Neural networks with many layers are called deep neural networks and the relevant field of machine learning is referred to as deep learning. A detailed discussed on the operation of convolutional neural networks is carried out in Section 3.2.

1.1.2 Probabilistic Graphical Models

Another useful class of machine learning models are probabilistic graphical models [12, 13]. These models represent conditional dependence between random variables in a graph structure. As a result, they are useful for structured data, such as images, where contextual relationships exist between data points. For example, a pixel has spatial based contextual relationships with its neighbouring pixels. A contextual relationship also exists between pixels that are not neighbouring. These long range contextual relationships are likely different to the close range relationships. In a video sequence, a frame has a temporal contextual relationship with previous and subsequent frames. Machine learning problems that involve predictions with structured target data are known as *structured prediction*.

Perhaps the most useful probabilistic graphical model for images is a conditional random field (CRF) [13]. These models provide a means to find the state of random variables that approximately best satisfy a set of known contextual relationships, conditioned on the input data. The state of random variables may be the predicted class labels for each pixel in an image. An example contextual relationship that may be considered is *a car is more likely to be on the road than on the water*. In practice, the contextual relationships are often not hand crafted, but learned automatically from labelled data. Considering contextual relationships allows refinement of poor initial predictions.

1.2 Image Classification

A fundamental problem in supervised machine learning is classification. This involves categorising a piece of data into one out of a set of predefined

FIGURE 1.2: Examples of different image classification problems. Top: Object recognition. Middle: Fine-grained recognition. Bottom: Scene recognition.

classes. For example, an animal classifier may be trained on a set of images belonging to the classes of dog and cat. The machine learning model should be trained such that it generalises to unseen examples of the predefined classes. That is, the model should learn what makes a dog look like a dog in a general sense, not what makes the dogs in images specific to the training set look like dogs. If this is done successfully, the model will be able to classify new examples of dogs that were not present in the training set.

Classification of visual data is clearly of great importance in robotic applications. As humans, recognising objects is fundamental to our ability to interact with our environment and understand its affordances. So much of our daily life, such as driving, working, shopping and so on, relies on our ability to classify objects. This includes recognising vehicles, reading street signs, recognising our colleagues faces and finding products on the grocery store shelf. For robots to be useful on our roads, in our factories and in our homes, the ability to classify visual data is just as vital as it is to humans.

There are several subdomains of image classification. Perhaps the most common is coarse object classification. In these problems, a classifier is trained to distinguish between different coarse object categories such as bird, car, table, boat and so on. Fine-grained classification, on the other hand, aims to distinguish between semantic classes that are very similar in appearance. This includes facial recognition, classifying birds or flowers by species and

classifying cars by make and model. Such fine-grained problems are challenging and can even be difficult for humans to master, but they have many important application areas such as security, biodiversity analysis and robot-human interaction. These problems require the machine learning model to capture small visual differences between object categories that are often indistinguishable to an untrained eye. A third domain of image classification is scene recognition, which aims to classify images into categories such as kitchen, bathroom, lecture theatre, classroom and so on. This is clearly an important problem in robotics, as it provides high-level context to the robot and may greatly influence the robotic actions taken. Examples of these problems are shown in Figure 1.2.

Many existing approaches to image classification have drawbacks when it comes to robotic applications. Most commonly used methods are limited in their ability to improve knowledge on-the-fly. Generally, these approaches require large amounts of new training data and costly model re-training and refinement to improve the model's understanding of the environment. In a similar vein, often models that are trained for a specific classification set do not transfer knowledge well to novel classes.

1.3 Dense Prediction and Semantic Segmentation

Image classification problems generally pair an input image with a single target class label. Semantic segmentation on the other hand, aims to predict a class label for each pixel in an input image. This divides the image up into semantically meaningful regions. Pixels may be labelled with classes pertaining to objects, such as human, car, table or chair, as well as amorphous semantic classes such as floor, ceiling, grass and road. Example semantic segmentations are shown in Figure 1.3.

Semantic segmentation is an important problem in robotics because it simultaneously provides the robot with information about objects and structure. The labelling of object classes allows the robot to make decisions about interactions with the environment, such as picking an object off a table or interacting with a human. The labelling of amorphous, structural semantic classes allows the robot to navigate the environment, for example, traversing a specific terrain (such as the road or footpath) and avoiding another (such as grass or water).

FIGURE 1.3: Semantic segmentation examples from Cityscapes [14] (top), Pascal VOC [15] (middle) and NYU v2 [16] (bottom).

There is a significant discrepancy between the requirements of a semantic segmentation system for robotic applications and the focus of semantic segmentation systems in the computer vision community. Conventional models for pixel labelling are optimised and evaluated on pixel-level performance. This means that incorrect regions of labelling are penalised based on the size, that is, the number of pixels that are incorrectly labelled. In robotic applications, however, the size of a detected object is not of great importance; the presence or absence of any object is influential in a robotic scenario. Any object detection can trigger some sort of costly robotic action, such as further computation as the robot attempts to learn more about the detected object, such as pose or instance label, or a physical action, such as the robot interacting with the object or navigating over some terrain. For these reasons, object

or region-level performance is more important in robotic applications than pixel-level performance.

Since conventional semantic segmentation approaches are optimised at the pixel-level, they are also evaluated on pixel-level performance. Such commonly used metrics as pixel accuracy and mean intersection-over-union do not evaluate a system on the robotics-appropriate criteria outlined in this section. This mismatch between conventional semantic segmentation system optimisation and evaluation, and the requirements of a robotic implementation of such a system, is a strong motivator for part of this thesis.

1.4 Metric Learning

The problem of metric learning is the learning of a distance function between two given inputs from a set of training examples. Throughout this thesis, the distance metric is learned based on the semantic class of the associated training examples and as such, the learned metric is a measure of semantic similarity. Given a trained model, the distance metric should be small between two images of the same semantic class and larger between two images of differing semantic class. It is also expected that the distance function will encode fine-grained semantic information, such as object attributes. For example, a distance metric learned on a training set of different bird species may return a smaller measure between two examples of different species from the same genus than between two examples from different genera. A visualisation of an example mock metric space is shown in Figure 1.4. Although the example is two-dimensional, metric spaces are generally high-dimensional.

Measuring the similarity between observations is an important problem in robotics. For example, since a robot operates in a dynamic environment, it is highly likely that it will observe examples of objects that are outside of the training set distribution. A well learned distance metric allows the robot to measure the similarity between an unknown observation and known objects, enabling some sort of inference to be made. An example of this may be a domestic robot that has the ability to recognise a subset of typical household items. The robot may observe the pet rabbit, which is outside of the training set distribution. However, since the training set did contain examples of cats and dogs, the robot is able to infer that the rabbit is similar to these other animals and should be treated as such. The ability of metric learning models

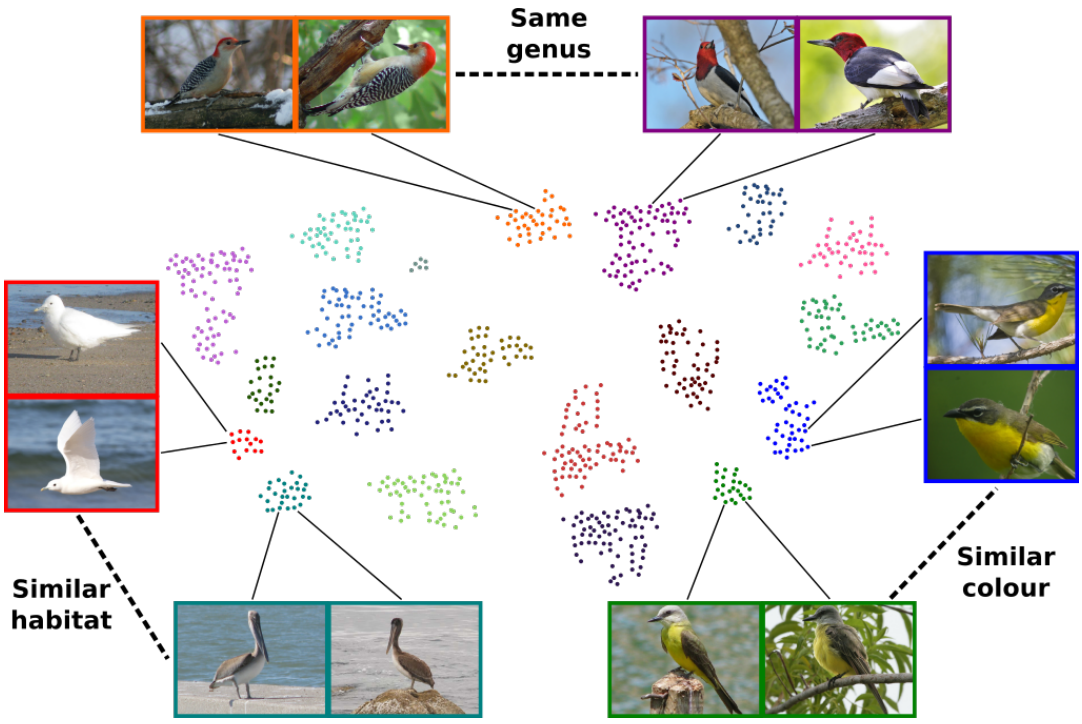


FIGURE 1.4: Example metric space. Colour represents semantic class, which in this example is the species of bird.

to transfer to unseen, out-of-distribution classes is a desirable property for robotics. Further, metric learning approaches can also be useful in the domain of few-shot learning. In this problem, models are trained on impoverished datasets, with only a few examples of each class in the training set. This is beneficial for robotics, as labelling data is labour intensive and requires human intervention. Additionally, unconstrained robots are unbounded in the number of classes they may need to recognise. As such, being able to learn from small amounts of training data is imperative for robots operating in these dynamic environments. Other applications of metric learning include ranking, retrieval, duplicate detection and weakly supervised learning.

Direct distance measures in the image domain are largely meaningless. That is, it is not expected that the Euclidean distance between a pair of images will be representative of the semantic similarity of the pair. Generally, metric learning approaches learn a transformation from the image space into a lower dimensional, semantically rich space. The goal is to learn the transformation such that Euclidean distance (or another distance measure) in the transformed space represents the notion of semantic similarity. Recently, the transformation is commonly learned with convolutional neural networks [17, 18, 19, 20, 21, 22, 23, 21]. The learned space is often referred to as a metric space, embedding space or feature space.

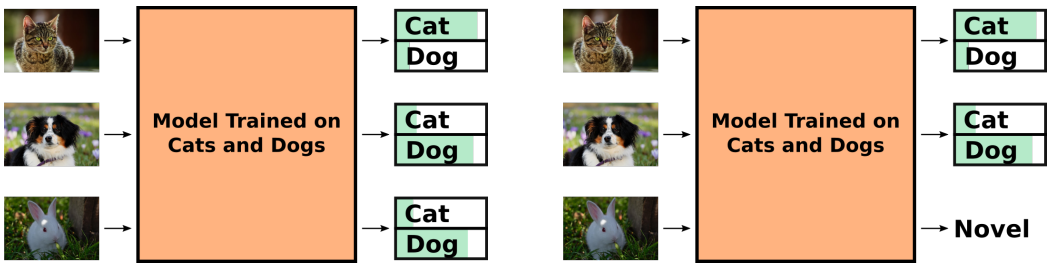
Commonly used approaches to metric learning have some drawbacks. One such drawback is that the classification performance of conventional metric learning models, beyond few-shot classification, is poor compared to state-of-the-art image classifiers. It is, however, desirable to have a single model that can perform both tasks of metric learning and classification simultaneously. This is particularly true for robotic applications, in which both problems are of great importance. A robot should be able to classify observations of known classes, as well as transfer knowledge to novel classes. As robots are often limited in compute resources, a single model that performs well in both domains is desirable. A second drawback of existing approaches is that they can be limited in their ability to represent intra-class and inter-class variations in the data.

1.5 Open Set Recognition

When applied to robotics, vision system outputs lead to decisions that have real-world consequences. The detection of a particular object can trigger robotic action that may be dangerous if that object detection is erroneous. Robotic problems are open set; a robot operating in a dynamic environment will undoubtedly encounter objects that are not represented in the training set. Conventional classification systems are closed set by design and will produce a prediction for any given input. In other words, the classifier will silently fail by categorising a novel example into one of the known training classes. Such silent failure is unacceptable in robotic applications.

A robot should be able to detect when it is observing an instance of a novel class. A novel example is one that is not represented in the training set. The problem of simultaneously classifying examples from known classes and detecting examples from novel classes is referred to as open set recognition. The difference between a closed set classifier and an open set classifier is illustrated in Figure 1.5.

Further to avoiding costly classifier errors, the handling of open set problems is important in robotics as it gives the robot the option of learning more about out-of-distribution examples and improving its understanding of the environment. Training data is designed to be a sample of the real-world distribution of data that the robot will encounter in its environment. Invariably in unconstrained environments, the training data distribution will differ from



(A) Closed set classifier. (B) Open set classifier.

FIGURE 1.5: A closed set classifier will silently fail when observing an example from outside of the training set distribution by incorrectly classifying the observation into a known class. An open set classifier can flag out-of-distribution examples as novel/unknown.

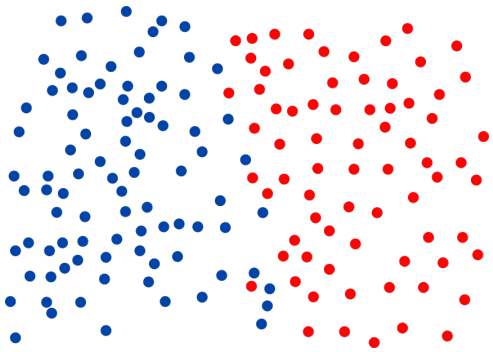
the real distribution, including missing semantic classes. Open set recognition and novelty detection provide a robotic vision system with the first step required to improve its representation of the real-world distribution.

The applications of a vision system that will not silently fail when observing out-of-distribution examples has reach far beyond robotics. Many real-world examples exist in which the output of a vision system has the potential to greatly impact users and the wider community, beyond simply attaching the incorrect label to an image. These application domains include computer-aided diagnosis in medicine, manufacturing, fraud detection and security.

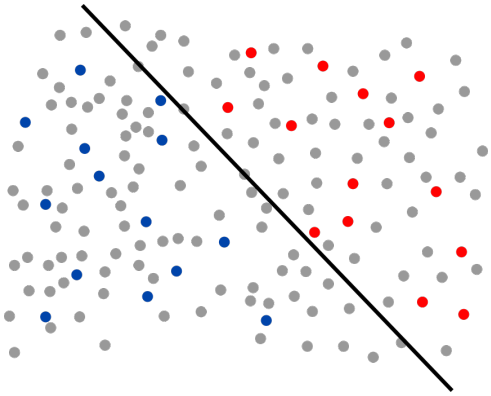
1.6 Active Learning

The conventional pipeline for a machine learning task first involves gathering and labelling training data. Labelling data requires human effort and can be laborious for large datasets or pixel-level labels. Additionally, labelling data often requires expert knowledge, for example in biodiversity datasets or medical datasets. However, not all pieces of data are created equal. Some training examples are more informative than others and human effort should be focused on these examples before the less informative examples.

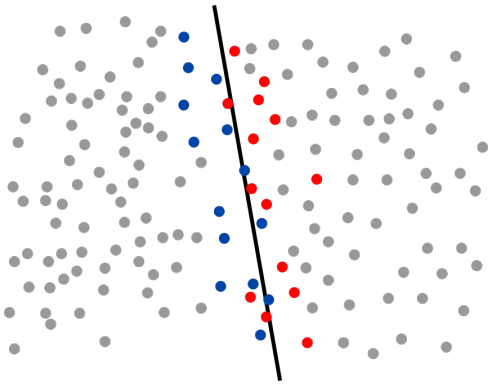
Active learning is the problem of automatically selecting the best unlabelled pieces of data that should be labelled. Given a labelling budget of b, that is, a human is able or willing to label b pieces of data, active learning aims to select the b most informative examples. Doing so should result in greater performance than randomly selecting b examples. This is illustrated with a



(A) Example space.



(B) Random query selection.



(C) Careful query selection.

FIGURE 1.6: Example two class linear decision boundary problem showing the importance of careful query selection in active learning problems.

simple two class linear decision boundary problem in Figure 1.6. The process of choosing examples for labelling is referred to as query selection. Random query selection results in a substandard decision boundary (Figure 1.6B). Selecting examples using an active learning methodology results in a superior decision boundary (Figure 1.6C). This near-ideal decision boundary is found by acquiring labels for only a fraction of the data. Active learning allows a machine learning model to efficiently learn from far fewer training examples than in a conventional learning pipeline.

A robotic vision system exploring the environment observes a wealth of new information. As discussed in Section 1.5, a training distribution is an imperfect approximation of the true, real-world distribution. A robot is likely to encounter examples of novel classes and novel examples of known classes. The initial training of a robotic machine learning model should not be the end of the robot's endeavour to understand the environment. Knowledge and understanding should be refined as the robot explores the world. This

gives rise to the idea of open set active learning in robotics, which includes incremental learning and online learning.

As a robot explores the environment and observes a plethora of new information, it should be able to discern which observations are going to provide it with the most novel information. In an active learning setting, the robot ranks observations in terms of informativeness and queries a user to label as many of the examples, in order, as the user permits. This allows the robot to update and improve its understanding of the environment. Ideally, the robot should favour selecting observations of novel examples that are not represented in the original training set distribution. Such examples represent knowledge that is completely new to the model and as such, is highly informative of the true distribution of data in the robot's operating environment.

1.7 Contributions

In this thesis, novel techniques are presented on the broad topic of machine learning for vision, with a focus on making considerations for robotic applications. Several target domain areas that are important for robotics, but lacking in existing literature in terms of robotic considerations, are identified and pursued. These include classification, metric learning, novelty detection and open set recognition, active learning and semantic segmentation. The contributions of this thesis are enumerated in the remainder of this section, grouped according to the aforementioned target domains.

1.7.1 Metric Learning and Image Classification

In chapter 4, we propose a convolutional neural network-based metric learning approach that can be applied to both feature embedding learning problems and image classification. Our method centres a Gaussian kernel on each training set feature embedding. During training, the kernels pull examples from the same class together and push examples from difference classes apart. We show how to make training feasible by introducing periodic asynchronous updates of the stored kernel centres (Section 4.7). The Gaussian kernels can also be used to classify new examples by summing the influence of nearby kernel centres (Sections 4.5.3 and 4.6.1). We introduce trainable

per-kernel weights to allow the influence of important kernel centres to be amplified (Section 4.5.3). Further, we show how to make training scalable by leveraging fast approximate nearest neighbour search (Section 4.8).

Our proposed approach is first evaluated on feature embedding learning, or distance metric learning, problems (Section 4.10.1). We investigate how well the learned distance metric transfers to novel classes that are withheld during training. Quantitative experimental results show that our method transfers well to novel classes, outperforming state-of-the-art deep metric learning approaches (Section 4.10.1.5). Qualitatively, the feature embedding spaces are visualised, showing that although the examples belong to novel classes that are withheld during training, the feature embeddings are co-located based on class and other semantic similarities (Section 4.10.1.6). Additionally, the importance of the feature embedding dimensionality is explored, finding that our approach is able to better take advantage of a larger embedding space, compared to conventional metric learning approaches (Section 4.10.1.4).

Further experimental contributions are provided by evaluating the proposed approach on the problem of image classification (Section 4.10.2). We show that compared to conventional softmax-based convolutional neural networks, our approach results in better classification performance on several datasets (Section 4.10.2.3). We also show that our approach has a significant advantage over softmax networks when the number of training examples is limited (Section 4.10.2.4). An ablation study is performed (Section 4.10.2.5), as well as an analysis of local neighbourhoods during training (Section 4.10.2.6). Finally, we investigate how accurately attributes can be propagated to nearby examples (Section 4.10.2.7).

1.7.2 Open Set Recognition and Novelty Detection

Open set problems are investigated in Chapter 5. We present a deep metric space approach to novelty detection and open set recognition (Section 5.5). Rather than silently failing when presented with out-of-distribution examples, our approach detects such examples as novel, while classifying examples from known classes (Section 5.5.2). Several distance-based novelty measures for deep metric spaces are investigated (Section 5.5.3). Our approach works on the knowledge that deep metric spaces transfer well to novel classes, allowing the detection of examples that differ from the training distribution.

Our approach is evaluated experimentally on several datasets (Section 5.7.2). We compare the approach to appropriate baselines, as well as purpose built novelty detectors and open set classifiers (Section 5.7.2.3). Results show that our approach allows more reliable detection of novel examples than the compared methods (Section 5.7.2.4). We also analyse how the open set classification accuracy for both in-distribution and the combination of in-distribution and out-of-distribution examples is affected by the novelty detection recall (Section 5.7.2.4).

1.7.3 Open Set Active Learning

Continuing with the open set theme in Chapter 5, we proposed an approach that enables the efficient active learning of observed novel classes (Section 5.6). Our approach allows a model to learn from novel observations, such that its understanding of the true distribution of data in the environment can be improved. We propose a query selection method that ensures observations that are both novel and representative of many observations are favoured (Section 5.6.2). Synthetic data is used to demonstrate the behaviour of the proposed approach. We show how to incorporate our query selection method into an active learning algorithm (Section 5.6.3). Finally, we propose an approach that allows refinement of model parameters such that the novel class representation is improved with a labelling budget of zero (Section 5.6.4).

Quantitative experimental contributions includes a comparison to other query selection approaches for metric spaces, as well as to a softmax network approach (Section 5.7.3). Results show that our approach outperforms the compared methods, with the advantage particularly large at small labelling budgets (Section 5.7.3.4). We further investigate the performance when no fine-tuning of the network weights is allowed (Section 5.7.3.4). Additionally, we investigate novel class discovery and the Kullback–Leibler divergence as a function of the number of label queries (Section 5.7.3.4). Qualitatively, we visualise the query selection process to investigate the behaviour of the proposed and compared methods (Section 5.7.3.5). Finally, the zero labelling budget learning approach is evaluated quantitatively (Section 5.7.3.7) and qualitatively (Section 5.7.3.6), comparing the method to active learning.

1.7.4 Semantic Segmentation

The problem of semantic segmentation is addressed in Chapter 6. Recognising that for robotic and other real-world problems, false object detections are often more important than pixel-level performance, we propose a region-level conditional random field model that significantly penalises such detections (Section 6.5). We define semantically meaningful image regions as nodes in the random field and show how to minimise the energy (Section 6.5.3) and learn the pairwise potentials (Section 6.5.4). Further, we propose a hierarchical region-pixel conditional random field that allows pixel-level refinement to better localise object boundaries and fine-grained image features (Section 6.6). We show how to define the pixel-level pairwise potentials such that appearance consistency and local smoothness are enforced and propose an approach to encourage semantically similar pixels to take the same class label (Section 6.6.2.2).

Identifying the shortcomings of conventional pixel-level evaluation metrics for semantic segmentation, we propose an object-aware evaluation metric that places great significance on false positive and false negative object detections (Section 6.7). We introduce an intersection-over-area term to our evaluation metric that ensures the measure cannot be gamed by the erroneous growing of predicted regions (Section 6.7.2). Our proposed evaluation measure is compared against conventional pixel measures with example segmentations, illustrating how the shortcomings of the conventional measures are overcome (Section 6.7.4).

Experimentally, our proposed conditional random field models are compared to appropriate baselines (Section 6.8). Results show that our approaches achieve significantly fewer false positive detections at a given number of false negatives (Section 6.8.3.2). Further, we show how our approach outperforms the compared methods in terms of both conventional pixel evaluation metrics and our proposed object-aware metric (Sections 6.8.3.2 and 6.8.4.2). The importance of using semantically meaningful image regions is also investigate by comparing to the use of structurally meaningful image regions (Section 6.8.3.2). Finally, we compare our approach to the baseline methods in a qualitative manner by visually inspecting example segmentations (Sections 6.8.3.3 and 6.8.4.3).

1.8 Publications

The material presented in this thesis is based on the following peer-reviewed publications:

- Benjamin J. Meyer, Ben Harwood and Tom Drummond, "Deep Metric Learning and Image Classification With Nearest Neighbour Gaussian Kernels", International Conference on Image Processing (ICIP), 2018 [24] © 2018 IEEE.
- Benjamin J. Meyer and Tom Drummond, "The Importance of Metric Learning for Robotic Vision: Open Set Recognition and Active Learning", (*to appear*) International Conference on Robotics and Automation (ICRA), 2019 [25] © 2019 IEEE.
- Benjamin J. Meyer and Tom Drummond, "Improved Semantic Segmentation for Robotic Applications with Hierarchical Conditional Random Fields", International Conference on Robotics and Automation (ICRA), 2017 [26] © 2017 IEEE.

1.9 Thesis Overview

In this chapter, the importance of vision for robotics was introduced, with a focus on learning from and understanding visual data. The difficulties of robotic vision were discussed, along with the shortcomings of many conventional computer vision approaches when viewed through a robotics lens. A broad introduction to several subtopics of learning for vision was given, including high-level motivations for the work carried out in this thesis. Finally, the contributions of the research described in this thesis were enumerated and the resultant publications were listed.

Relevant literature is reviewed in Chapter 2. This includes discussion on machine learning models, with a focus on neural networks and probabilistic graphical models. Problem specific literature is also reviewed, including image classification, metric learning, open set recognition, active learning and semantic segmentation. In an attempt to make this thesis as self contained as possible, Chapter 3 details the fundamental theory, algorithms and practices that are leveraged in the subsequent chapters.

Chapter 4 details our proposed approach to image classification and metric learning using nearest neighbour Gaussian kernels. Open set problems are explored in Chapter 5. Beginning with open set recognition and novelty detection, we present an approach that allows for the detection of out-of-distribution observations using deep metric spaces. We then turn to the problem of open set active learning by investigating how a model can learn from detected novel examples and improve its understanding of the environment. The problem of semantic segmentation is investigated in Chapter 6. Conditional random field models are proposed, as well as a robotics-appropriate evaluation metric.

The thesis is concluded in Chapter 7. We again provide a summary of the key contributions and insights of the presented research. Finally, future research opportunities that expand on the contributions of this thesis are explored.

Chapter 2

Background

This chapter provides context to the research presented in this thesis by reviewing relevant literature. Firstly, a brief overview of classical machine learning approaches is given, as well as early artificial neural networks. Deep learning is then reviewed, with a focus on the application to image classification and semantic segmentation. Metric learning, including deep neural network approaches, is then discussed. Classical and recent approaches to novelty detection and active learning are reviewed, followed by a discussion on probabilistic graphical models for semantic segmentation.

2.1 Classical Machine Learning

In this section, an overview of classical machine learning techniques is given. Here, the term "classical" is used to refer to methods outside of deep learning. In the context of computer vision problems such as image classification, these methods generally involve the extraction of hand-engineered features, which are then input to a machine learning algorithm. This is unlike the majority of deep learning approaches, which combine these two stages by learning the model parameters that extract features directly from the input data.

The discussion presented in this section is intended only as a brief overview of classical machine learning approaches, it does not attempt to discuss the vast amount of computer vision-based machine learning literature from the past several decades. Rather, this section aims to provide context for the following discussion on deep learning literature that more closely relates to the research presented in this thesis. A detailed review of classical approaches

in the context of classification problems can be found in the survey by Kotsiantis *et al.* [27].

2.1.1 Feature Descriptors

Machine learning algorithms and models, such as classifiers, often cannot operate directly on the raw data. In the case of images, the data is high-dimensional, redundant and extremely variable for a single semantic entity. As such, it is difficult for models to directly analyse image data for problems such as classification and recognition. Feature extraction refers to the process of transforming the raw data into meaningful features. Practically, a feature is a vector of numbers that in some way encodes the informative properties of the data. Features are sometimes referred to as feature descriptors, as they describe the useful information contained within the input data. In general, a feature will have significantly lower dimensionality than the raw image.

A widely used feature extraction algorithm is the scale-invariant feature transform (SIFT) proposed by Lowe [28]. The SIFT algorithm transforms an image into a collection of feature descriptors that are invariant to scale and rotation, and robust to changes in illumination and viewpoint. Scale invariance is achieved by constructing a scale space that consists of resized versions of the image, such as half size, quarter size and eighth size, each of which are progressively blurred out by Gaussian filters with increasing scale terms. Edges and corners contained within the original image are extracted by approximating the second order derivatives of the blurred images by computing the differences between blurred images at consecutive scales. Keypoints are found by locating the minima and maxima in the difference of Gaussian images and made rotation invariant by defining an orientation based on the gradients in the blurred image. A descriptor for a keypoint can be computed by analysing and encoding the gradients in sub-windows surrounding the detected keypoint.

The ideas presented in [28] are made more computationally efficient in the SURF features proposed by Bay *et al.* [29]. The improvement in execution time is largely the result of further approximating the computation of the second order derivatives by the use of box filtering. Wavelet responses are used to compute feature descriptors for detected keypoints. A similar feature descriptor to SIFT and SURF is the histogram of oriented gradients (HOG)

[30]. The image gradient values are computed using Sobel filters and small image regions are used to compute gradient orientation-based histograms.

Feature descriptors exacted from an image are often used to represent the image as a bag-of-visual-words model [31]. In this method, each feature descriptor is mapped to one of several "codewords", which are used to represent similar features found in the image. A frequency histogram can be computed from these mappings. The histogram summarises the salient features of the image and can be used for classification and recognition problems.

2.1.2 Supervised Learning

Machine learning with labelled training examples is referred to supervised learning. A commonly used supervised machine learning model is the decision tree [32], which can be used for both classification and regression problems. Data is split at each node in the tree, with an example eventually arriving at a leaf node. In the case of classification, each leaf node is assigned a class label or a probability distribution over class labels. In general, a decision tree learning algorithm will attempt to identify the most informative features and split the data accordingly. A data split should result in subsets that are as homogeneous as possible, in terms of the target values. Example decision tree learning algorithms include ID3 [33] and C4.5 [34].

The predictive power of machine learning algorithms can be improved by employing multiple models to produce several hypotheses. Such approaches are referred to as ensemble methods and operate on the premise that a set of weak learners can be combined to form a strong learner [35]. Boosting is a ensemble learning algorithm that iteratively trains weak learners and makes predictions via weighted aggregation. Perhaps the most popular boosting approach is Freund and Schapire's AdaBoost algorithm [36]. Unlike other boosting algorithms [35, 37], AdaBoost is adaptive in that subsequent learners will focus on examples that are misclassified by previously trained learners. A different ensemble learning approach is bootstrap aggregating, or bagging for short [38]. Bagging algorithms train multiple learners in parallel on different subsets of the data. The data subsets are sampled randomly and with replacement. The predictions from each of the weak learners are aggregated to produce the final prediction. A prominent example of bagging is random forests [39], which combine multiple decision trees to form a strong learner.

Support vector machines (SVMs) [40] are another of the most popular supervised machine learning models. The standard SVM model is most commonly used as a linear classifier for binary data, that is, data that belongs to either one of two classes. A support vector machine aims to find the best hyperplane that segregates input features based on class. This hyperplane can then be used to classify new examples. Support vector machines can be made non-linear classifiers by use of the kernel trick [41], which employs kernel functions to transform the input data into a space in which the data is linearly separable. A support vector machine is a binary classifier, however, multi-class classification can be achieved with SVMs [42, 43]. For an n class problem, n binary classifiers can be learned in a one-versus-all arrangement, or $n(n-1)/2$ binary classifiers in a one-versus-one arrangement.

A useful class of machine learning models are probabilistic graphical models, including Markov random fields [12] and conditional random fields [13]. These graphs allow the modelling of relationships between structured data, such as images or speech. Problems that involve prediction with structured objects are referred to as *structured prediction*. The popular conditional random field model is particularly useful for images, as it allows direct conditioning on an input, such as the pixels contained within an image. As such, a conditional random field can model the structural, contextual and appearance-based relationships that exist between pixels. A detailed discussion on probabilistic graphical models for the problem of semantic segmentation is given is Section 2.6.

2.1.3 Unsupervised Learning

Machine learning with unlabelled training examples is referred to unsupervised learning. A notable domain area of unsupervised learning is clustering, which involves assigning data points to groups, or clusters, such that examples within a group are more similar than examples from different groups. Unsupervised clustering techniques aim to discover the intrinsic structure of the underlying data. Common unsupervised clustering techniques include k-means [8], which assigns each data point to a single cluster, fuzzy C-means [44, 45], which allows clusters to overlap, and Gaussian mixture models, which take a probabilistic approach to clustering using expectation maximisation [46].

2.2 Early Artificial Neural Networks

The topic of deep learning has received great attention in the machine learning, computer vision and robotics fields in recent times. Deep learning refers to use of artificial neural networks with many layers to model abstractions of data that would be too high-level for traditional machine learning algorithms to achieve. Advances in learning techniques for artificial neural networks, as well as a vast increase in computing power, has allowed deep learning to become the state-of-the-art method for tasks including image understanding, speech recognition and natural language processing. In this section, early artificial neural networks that were a precursor to the "deep learning revolution" are discussed.

2.2.1 Artificial Neurons

Deep learning finds its origins in the perceptron first described by Rosenblatt [9]. The perceptron is a linear classifier that given an input feature vector, produces a binary output. A feature vector that in some way describes the data is operated on by a weight vector via the dot product and shifted by a bias term. The weights can be learned on a training set in a supervised manner and will ideally then be able to classify unseen data. Many complex classification tasks are non-linear, that is, the input data is not linearly separable and therefore require more complicated, non-linear classifiers to perform well.

A perceptron is a type of artificial neuron since the perceptron either "fires" or does not based on the sum of several incoming signals. This is akin to a biological neuron, where the multiplication of the input vector by the weights is comparable to the biological dendrites, the summation portion of the dot product is analogous to the soma and the firing based on comparing the dot product to a threshold is akin to the axon, which samples the soma and sends a pulse once a certain potential is measured.

2.2.2 Artificial Neural Networks

A popular model for learning complex and rich representations of data are artificial neural networks. At their core they are a network of interconnected artificial neurons. Typically, neurons are divided into layers: the first being the input layer, the middle layers as hidden layers and the final as the output

layer. The layers are made up of several artificial neurons, which take their input from the output of the previous layer's neurons, or the input vector in the case of the input layer. This type of network is described as a *feed forward* artificial neural network, as signals are only sent forward to the next layer. This is as opposed to recurrent neural networks (RNNs), which may contain loops. The perceptron has a step function as its activation function, which means that a very small change in the input feature can result in a significant change at the output. This is undesirable in many problems, for example classification tasks, where non-identical data of the same class should be able to be successfully classified. As such, other non-linear activation functions are used in practice, such as the sigmoid function and the rectified linear unit [47, 48]. Connecting multiple layers of artificial neurons together results in a non-linear classifier that has strong representational power when designed appropriately.

The first deep learning-like algorithm was described by Ivakhnenko and Lapa [49]. The proposed network consists of multiple layers of perceptrons with polynomial activation functions. Weights are trained layer by layer via statistical means, where the best features are selected and forwarded to the next layer until improvement ceases. An important breakthrough for deep learning, as it is known today, was the development of the back propagation of errors algorithm, which allows the training of large networks with many parameters to be feasible. The importance of the algorithm was first noticed by Rumelhart *et al.* [50] who noted that back propagation allows faster learning in neural networks than other techniques and opens the door to solve problems that would otherwise have been infeasible with a neural network model.

The algorithm works by finding the derivative of a cost function with respect to all weights in the network by applying the chain rule of differentiation at each layer of the network. Using classification as an example, an iteration of training with the back propagation algorithm involves first performing a forward pass through the network that has a training example as the input. The target, or ground truth, value for the training example is then presented and the error, that is, the difference between the predicted value and the actual value, is calculated. The gradient of the error with respect to the input of each layer is then back propagated through the network to find *delta* values for each weight in the network. The delta values are used to find the gradient of the error with respect to each neuron, allowing the weights to be

updated using gradient descent. This process is repeated multiple times over the training data until convergence is observed or the network's performance reaches an acceptable level. The work of LeCun *et al.* [11] is the first practical use of back propagation in neural networks, in which a network is trained to recognise handwritten U.S zip codes.

2.2.3 Convolutional Neural Networks

Perhaps the most commonly used class of model, convolutional neural networks (CNNs) are feed forward neural networks comprised of collections of neurons arranged as 3D structures with a height, width and depth. When presented with data (typically an image) or a feature map from a previous layer, the 3D structures tile the activations by operating on each patch of the input, creating an output feature map. In other words, the neurons form a kernel or filter, which is convolved with the input to produce the output feature map. The output feature map is then fed to the next layer. A convolutional layer of a convolutional neural network will typically be comprised of a number of these filters, producing the equivalent number of feature maps at the output of that layer.

Conventional CNNs consist of several layer types, four of which are introduced in this section. Detailed discussion of layer types can be found in Section 3.2.2. *Convolutional layers* perform a convolution with the input, producing a structured output feature map. *Activation layers* perform a non-linear operation on the input features, typically a rectified linear unit. *Pooling layers* downsample the data in a non-linear manner, such as taking the maximum value within a specified window size. This is done to increase the network's tolerance to local spatial variations in features and, as a result, also reduces overfitting. *Fully connected layers* are often placed after a series of convolutional, activation and pooling layers. Neurons within a fully connected layer are connected to each feature from the previous layer. Fully connected layers perform much of the high-level reasoning about the input data given the features extracted in the previous layers. Conceptually, a CNN works by learning higher-level features and data abstractions at each set of convolutional and pooling layers. For example, convolutional filters in the first layer will generally look for low-level features in image data such as edges, while convolutional layers deeper in the network will use the downsampled lower-level features to find semantically rich higher-level features.

Fukushima [10] presented the first convolutional-type neural network, however, the important features were engineered by hand. Pioneering work by LeCun *et al.* [11] uses convolutional neural networks for hand written character recognition and document analysis. Learning of the network is fully automated by use of the back propagation of errors algorithm with gradient descent. Although this work was promising, the most commonly studied methods for many more years were conventional classifiers such as support vector machines [40, 51, 41, 52] with hand engineered features including SIFT [28, 53], histograms of oriented gradients [30], SURF [29], BRIEF [54] and ORB [55].

A number of issues hindered the development of deep learning, including the vanishing gradient problem, a lack of computational power and a lack of training data. Diminishing or vanishing gradients is an exponential decay of gradients during back propagation due to the chaining of derivatives. Since the update to a weight is proportional to the gradient at that neuron, the early layers in the network learn very slowly. A number of solutions have been proposed, including long short term memory (LSTM) for recurrent neural networks [56] and pre-training for feed forward networks [57]. The pre-training in [57] is performed in an unsupervised manner and results in much quicker supervised training than random initialisation of weights.

A great amount of computational power is required to train large neural networks in a reasonable amount of time. An approach to combat this involves training the networks in a distributed manner over tens of thousands of CPU cores [58]. Advances in general purpose graphics processing units (GP-GPUs) provide another solution as neural network training is suited to the parallel nature of GPU processing. Work such as [59, 60] demonstrate how GPUs can be used to train large networks in reasonable time frames. This has the added benefit of also addressing the problem of vanishing gradients, as the early layer weights can be learned faster.

2.3 "The Deep Learning Revolution"

Deep learning refers to machine learning techniques that make use of multilayer artificial neural networks. In this sense, *depth* refers to the cascading of an often very large number of layers to extract high-level feature representations. Unlike classical machine learning techniques, which generally require

hand crafted features, deep learning techniques extract features directly from the input data. Network types used for deep learning include convolutional neural networks, recurrent neural networks and belief networks. This review focuses on the use of deep convolutional neural networks for computer vision problems. However, deep learning has also applied to many other problem domains, including natural language processing, fraud detection, recommender systems and advertising.

The deep learning revolution is a term used to refer to the rise of deep learning algorithms to become the dominant field of research in machine learning. The so-called "revolution" has been driven by the powerful compute from general purpose GPUs, the abundance of data available on the internet and an array of improvements to network architecture designs that have enabled more efficient and stable training of deep models. These three factors have allowed many of the key problems that had previously hindered the development of artificial neural networks to be largely overcome.

2.3.1 Deep learning for Image Classification

Much of the eminent deep learning research is focused on the design of network architectures and training algorithms for image classification and visual recognition. In this section, the importance of training data and the development of several key advances in network architecture design are discussed.

2.3.1.1 Challenges

Organised challenges often inspire and spur on research in a particular field. This has certainly been the case for image classification and the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [61]. The ImageNet online database [62] consists of several million images, labelled in a hierarchical structure. The collection contains images belonging to tens of thousands of synonym sets, named synsets. From this large database, just over 1.2 million training images are selected for the ILSVRC challenge. Images belonging 1000 semantic classes are represented in the challenge dataset. Performance is evaluated on both top-1 and top-5 classification error. Top-1 error only considers a classifier's top prediction per image, while top-5 evaluation allows the classifier to provide its five best guesses. Due to its large size, ImageNet

is often used as a pre-training step for deep convolutional neural networks that are then fine-tuned on a different set of data. Pre-training is explained in detail in Section 3.2.5.1.

ImageNet builds on top of another challenge that has spurred significant research in the field: Pascal VOC [15]. A smaller dataset, Pascal VOC contains images or objects belonging to only 20 semantic classes. In addition to object labels, Pascal VOC also contains pixel-level class labels. As such, the dataset is a common benchmark for semantic segmentation approaches.

2.3.1.2 AlexNet

One of the most seminal works of the so-called "deep learning revolution" was that of Krizhevsky *et al.* [60]. The CNN architecture proposed has since become known as *AlexNet*, named for the paper's first author Alex Krizhevsky. The AlexNet paper results in an error reduction of up to 10% on the ImageNet challenge. Several key contributions from [60] are responsible for this reduction in error. The AlexNet architecture consists of five convolutional layers, the first of which contains 96 kernels of size $11 \times 11 \times 3$. Pooling layers follow the first, second and fifth convolutional layers. The final pooling layer is followed by two sequential fully connected layers with a 4096-dimension output. A final fully connected layer outputs a vector with the number of dimensions equal to the number of classes. This is then fed into a softmax layer to produce a distribution across class labels.

A key element of [60] is the use of rectified linear units (ReLUs) as neuron activation functions. Prior to this work, the most common activation function was the hyperbolic tangent function. Such neuron models cause slow training time when using gradient descent optimisation. The use of ReLUs, which are modelled as $f(x) = \max(0, x)$, result in networks that are significantly faster to train. This allows networks to be deeper than otherwise possible with saturating activation functions such as $\tanh$. Derivatives for ReLUs are simple to compute. The derivative is equal to 1 for x values greater than 0, and 0 for negative values of x. Since the derivative is undefined at $x = 0$, it is usually taken to be equal to 0. Due to the non-saturating nature of the function, ReLUs ease the vanishing gradient gradient problem experienced with saturating activation functions. The problem of vanishing gradients is discussed in detail in Section 3.2.5.7. This property contributes to the facilitation of deeper networks.

Since ReLUs do not saturate as the input grows, input normalisation is not required. The authors of [60] note, however, that local normalisation reduces over-fitting and improves network generalisation. Activations at certain layers are normalised by a scaling term that considers spatially adjacent activations. This sort of normalisation is biologically inspired and results in adjacent filters that compete for large activations. Including local response normalisation reduces the ILSVRC top-1 error by 1.4%.

Pooling layers downsample feature maps by summarising local neighbourhoods of activations. This is commonly carried out with a max operation or an average operation. Pooling reduces the size of feature maps to be computationally feasible and introduces some translational invariance, which reduces a network's proneness to over-fitting. Generally, pooling neighbourhoods do not overlap, but Krizhevsky *et al.* [60] find that some overlap between adjacent neighbourhoods can be beneficial and reduce over-fitting during training.

Over-fitting is the problem of a model learning the noise in the training set. In other words, the model fits to the training examples so well that it does not generalise to examples outside of the training set. This is discussed in detail in Section 3.2.5.5. Over-fitting is further addressed in [60] by the use of data augmentation. The training set is artificially enlarged by taking crops of the input image, creating translations in both the horizontal and vertical directions. In addition, the inputs are horizontally mirrored, further augmenting the training set. The final method of data augmentation implemented involves altering the colour intensities of pixels, resulting in some invariance to illumination changes.

Dropout [63] is used as an effective tool against over-fitting. It is undesirable for network weights to co-adapt in a manner that results in neurons relying on the presence of other specific neurons in order to be useful. By randomly setting some activations to zero, or "dropping out" neurons, during training, the neurons are not able to rely on the presence of any other given neuron. This can also be thought of as randomly sub-sampling a different network architecture at each iteration of training. The authors suggest dropping out neurons with a probability of 0.5 at each training iteration. In the AlexNet architecture, dropout layers follow the first two fully connected layers.

Due to memory limitations on the GPUs used to train the network, the authors employ an interesting parallelisation scheme. Utilising two GPUs, half the network weights are stored on one GPU and half on another. The split

occurs at each layer, for example, half the the first convolutional layer kernels are on one GPU and the second half on the other. Communication between GPUs is only permitted at certain layers; some layers will take as input activations from both GPUs, while others take as input activations from only one GPU. Compared to a network trained on one GPU with half the number of convolutional kernels per layer, the multiple GPU approach reduces top-1 error by 1.7%. The GPUs described in the paper each have 3GB of RAM. Though many GPUs have enough RAM to store the entire network on one card, the parallelisation scheme presented remains applicable to larger models. The work in [64] improves on AlexNet performance by better selecting hyperparameters and tweaking the convolutional layer sizes.

2.3.1.3 Subsequent Architectures

Simonyan and Zisserman [65] propose a network architecture that demonstrates the importance of depth in convolutional neural networks. The presented architecture is similar to AlexNet [60], but consists of many more convolutional layers, with all convolutional kernels sized at 3×3. The authors investigate four different networks: 11, 13, 16 and 19 weight layers. The weight layers consist of convolutional layers and 3 fully connected layers (as in [60]). A significant reduction of 4.1% in ILSVRC error rate is seen when increasing the network depth from the shallowest to deepest depth considered. Increasing depth with this architecture appears to result in diminishing returns beyond 16 weight layers. The most commonly used model variant is the 16 layer version, generally dubbed *VGG16* in the literature. Greater network depth results in a large number of network parameters. The small 3×3 convolutional filters, compared to up to 11×11 in AlexNet, are used as a means to reduce the number of parameters.

The power of network depth is further demonstrated by Szegedy *et al.* [66]. In honour of the pioneering LeNet [11] architecture, the network proposed in [66] is named *GoogLeNet*. This network introduces a network-in-network type architecture, where each layer itself is a network-like structure. The layers consider local correlations in images by concatenating 1×1 convolutions to cover very local correlations with 3×3 and 5×5 convolutions to cover more spread out correlations. Such a structure allows the network to better deal with different scales of salient objects in an input image. These

network-in-network layers are named *Inception* modules and the architecture presented contains nine such modules. Although the network is very deep, consisting of 22 weight layers in total, the small convolutions result in a significantly reduced number of parameters. In fact, despite being made up of 14 more weight layers that AlexNet, GoogLeNet contains tens of millions fewer parameters. Due to the great depth of the network, the vanishing gradient problem is prevalent. In order to ease this effect, auxiliary losses are introduced at different stages in the network. The auxiliary losses fork off the main network after given inception modules and attempt to perform classification from these shallower parts of the network. The total loss is the sum of the main loss at the end of the network and any auxiliary losses included. The authors use two auxiliary losses, following the third and sixth inception modules.

GoogLeNet is often referred to as *Inception v1*, with subsequent updates to the architecture dubbed *Inception v2* through *Inception v4*. The first improved architecture, Inception v2 [67], rethinks the inception module with the aims of reducing computational complexity and limiting the representational bottleneck that results from subsequent convolutions reducing the feature map dimensions. The former is achieved by refactoring larger convolutions as multiple smaller convolutions. For example, a 5×5 convolution is replaced with two 3×3 convolutions. Additionally, 3×3 convolutions are replaced by a 1×3 convolution followed by a 3×1 convolution, further reducing the computational complexity. The representational bottleneck is eased by performing some convolutions in parallel, rather than sequentially. In other words, rather than making the network deeper, it is made wider. The Inception v3 architecture [67] makes small improvements on the second version, including better regularisation with label smoothing, which reduces the network's proneness to over-fitting. Another incremental improvement, Inception v4 [68] simplifies the third version by introducing a more uniform architecture.

He *et al.* [69] propose a residual neural architecture, ResNet, which surpasses human performance in the top-5 ILSVRC error rate, with an error of 3.57%. Again, the importance of network depth is shown, with 50-layer, 101-layer and 152-layer variants of the architecture presented. The key contribution of [69] is the skip connections between layers. The training of very deep networks is troublesome as the vanishing gradient problem is prevalent. Often, naively stacking additional layers in a network will degrade performance.

The skip connections used by He *et al.* [69] act as an identity mapping between layers and as such, adding more layers should not increase the training loss. Bypassing layers during training makes the initial stages of learning easier. This can be thought of as collapsing the large network into a more compact architecture. As learning progresses, the network weights will learn to mute the activations from the skip connections and focus on those from the now well learned adjacent layers. He *et al.* improve the architecture by refining the residual block in order to allow easier, unimpeded propagation of errors through the skip connections in [70]. This architecture allows very deep networks to be trained successfully, without performance degradation. For example, a 1001-layer variant is successfully trained. The *Inception-ResNet* network [68] combines ideas from the inception architectures with those from ResNet architectures.

The idea of residual blocks is extended by Huang *et al.* [71] to include skip connections between layers and all subsequent layers in the network. This densely connected architecture, dubbed *DenseNet*, allows further improved propagation of information, compared to the architecture in [70]. In addition to making network training easier, this network configuration explicitly preserves information from earlier in the network. This is compared to standard networks, which must learn to preserve relevant information. The explicit preservation provides the classifier with information extracted from both shallow layers and deep layers of the network. Further study in [72] finds that many layers contribute only a small amount of new information. This means that in order to ease computational complexity during training, layers can be randomly removed, or "dropped out", at any given training iteration. The authors find that this not only reduces training time, but also improves test performance.

The networks described in this section and Section 2.3.1.2 all use a fully connected layer and softmax layer to obtain a distribution over class labels in order to perform classification. Cross-entropy loss is used to drive the learning. However, the base network architectures are often used with other classifiers and loss functions. This includes using the networks as off-the-shelf feature extractors [73, 74], as base networks for metric learning (discussed in Section 2.4.2) and base networks for the method proposed in Chapter 4.

2.3.2 Deep Learning for Dense Prediction

The networks described in Section 2.3.1 downsample the input image with consecutive convolutions and pooling operations, followed by one or many fully connected layers. The end result, for classification problems, is a single vector representing the softmax distribution over class labels. In these networks, all spatial information from the input is lost. This is by design, as the networks aim capture information about the salient semantic entities present in the input image, regardless of the location or scale of the semantic entities. For problems that require spatial information to be preserved, for example when carrying out per-pixel prediction, such networks are not suitable. Perhaps the most common example of a per-pixel prediction problem is semantic segmentation, which aims to label each pixel in an input image. Some approaches, such as [75], are able to produce a bounding box surrounding an object prediction. However, this coarse object localisation is not sufficient for many applications requiring a pixel-level labelling.

One method to tackle the problem of dense prediction with CNNs is patch classification [76], wherein a patch of pixels surrounding a centre pixel is passed through a conventional CNN, resulting in a prediction for that centre pixel. Such an approach is inefficient as many forward passes through the network are required to obtain a pixel-wise segmentation. Other approaches [77, 78, 79] involve first generating region proposals, extracting features from the proposed regions and then performing classification. However, the remainder of this section focuses on methods that extract features for all pixels, or downsampled pixel neighbourhoods, rather than proposal-based segmentation approaches.

2.3.2.1 Fully Convolutional Networks

An influential work is the fully convolutional network (*FCN*) presented by Long *et al.* [80]. As the name suggests, the fully convolutional network does away with the fully connected layers from AlexNet [60] and VGG [65] architecture types, which discard any notion of spatiality. The fully connected layers are replaced by convolutions of 1×1 spatial extent. For example, the 4096-dimension fully connected layers in VGG are replaced with $1 \times 1 \times 4096$ convolutional layers. This means that dimensions of activations at the input to the layer are preserved in the output activations. Rather than resulting

in a single softmax distribution vector, such an approach can output an array of activations of size $h' \times w' \times n$, where h' is the downsampled height of the input, w' is the downsample width and n is the number of classes. This array of activations contains coarsely spaced softmax distributions. In other words, some spatial information is preserved, though the output space is downsized and coarse compared to the input. Removal of the fully connected layers also means that the network can handle arbitrary input size, above a minimum height and width. This is unlike the networks described in Section 2.3.1, which require inputs of a specific size.

The coarse distributions produced by the fully connected network must be upsampled and projected onto the original image resolution. A simple approach is to simply perform bilinear interpolation. Long *et al.* [80] suggest the more generalised solution of performing a series of deconvolutions. This means that the upsampling operation can be learned along with other network parameters. This type of network configuration is known as an encoder-decoder network. The encoder network takes the image as input and extracts semantic information, downsampling to coarse feature maps. The decoder network upsamples the extracted semantic information to the original image resolution. Deconvolution is also often called upconvolution. One way to think of deconvolution is as convolution with a stride of less than one. The stride of an image convolution is the number of pixels the convolutional kernel is shifted between each operation.

An important feature of these networks is that the encoder portion can be pre-trained on large image classification datasets, such as ImageNet [61], using the standard AlexNet or VGG implementations. Image-level labels are much easier to obtain than pixel-level labels, so the ability to leverage large image classification datasets for the purpose of pre-training is desirable.

2.3.2.2 Improved Encoder-Decoder Networks

Even after upsampling with deconvolutions, the segmentation produced by FCN is coarse. The down-sampling that occurs in the encoder potion of the network means that fine-grained information, such as object boundaries, is lost. In order to incorporate fine-grained information in the final output, Long *et al.* [80] introduce skip connections into the network. The skip connections allow higher resolution feature maps from shallower portions of the network to be incorporated in the final decision of the classifier. The best

performing network variant incorporates information from feature maps at three different scales.

The *SegNet* architecture [81, 82] mirrors the encoder network with a full decoder network. A key element of this architecture is the use of the encoder max-pooling indices in the decoder network. The pooling indices are used in the upsampling layers of the decoder in order to more accurately project feature maps onto the pixel space. As these upsampling layers essentially undo the pooling operations, they are often referred to as unpooling layers. Compared to simple bilinear upsampling, unpooling layers result in improved object boundary delineation.

Although the skip connections in the fully convolution network from [80] allow some fine-grained information to incorporated, the network struggles to segment objects of various scales. For example, small objects are often missed by this approach. Similarly, other fine-grained details are overly smoothed by the FCN architecture due to the coarseness of the starting point for the upsampling. These issues are addressed by Noh *et al.* [83] by learning a full decoder network. Similar to SegNet, this work mirrors the encoder network and uses unpooling layers to better project the feature maps to a larger resolution. The network operates on individual object proposals, allowing fine-grained details to be captured. This is because unlike FCN, the receptive field is not of a fixed-size.

Another similar encoder-decoder structure is proposed by Ronneberger *et al.* called *U-Nets* [84]. This network is applied specifically to biomedical image segmentation. In this application, there are commonly fine details in the input medical image that must be preserved. A second challenge in biomedical image segmentation is the scarcity of training data. Ronneberger *et al.* [84] perform large-scale data augmentation with elastic image deformations. The authors find that their approach responds well to data augmentation, while Long *et al.* [80] note that data augmentation results in limited improvement.

The success of ResNet [69] and DenseNet [71] for image classification naturally leads to these ideas being incorporated in dense prediction networks [85, 86]. The work of Drozdzal *et al.* [85] incorporates residual blocks in an encoder-decoder architecture. As with ResNet, the skip connections allow for easier training of deeper networks. Ideas from DenseNet, namely connecting each layer's activations to all subsequent layers, are incorporated into the work proposed in [86]. The skip connections allow information from all scales to be used in the reconstruction of the high resolution output.

Lin *et al.* [87] also address the problem of a obtaining a high resolution output with fine-grained information preserved. The authors use a ResNet style architecture as the encoder network. So-called *RefineNet* blocks are placed at each upsampling stage of the decoder network. These blocks fuse together features from the encoder network and upsampled features from the previous step in the decoder network. The RefineNet blocks contain local and long-range skip connections, allowing training to be carried out efficiently.

Peng *et al.* [88] note that semantic segmentation is comprised of two contradictory tasks: classification and localisation. Classification should be invariant to spatial changes in the input space, while localisation must be sensitive to such spatial changes. As with other approaches discussed in this section, the authors utilise a fully convolutional encoder network to preserve spatiality. This addresses the localisation half of the problem. For the classification half, the authors suggest that large convolutional kernels be used. Experiments are carried out with convolutional kernel sizes ranging from 3×3 (the standard size for ResNet), up to 15×15. The larger kernel sizes result in a performance improvement over the smaller kernels. Due to the added computational complexity introduced with large convolutions, an approximation similar to that in Inception v2 [67] is carried out. This approximation involves refactoring the convolutions as a sum of simpler convolutions, where a dimension of the kernel is set to a size of 1.

2.3.2.3 Dilated and Atrous Convolution

Pooling operations in the encoder network expand the receptive field, allowing more context to be incorporated into the construction of feature maps. The downside of pooling is that the resolution of the output space is significantly reduced compared to the original input image resolution. One solution to the problem of removing pooling operations but retaining a large receptive field is dilated convolutions. A normal convolutional kernel is densely arranged such that the kernel operates on all neighbouring underlying pixels. Dilated convolutions introduce a dilation rate that specifies the sparsity of the convolutional kernel. The spacing between kernel parameters are dilated by this amount, such that the kernel operates on more sparsely spaced pixels in the underlying image. This can be thought of as an approximate alternative to simply increasing the kernel size that does not increase the number of parameters. Another way to think about dilated convolutions

is as convolution with holes in the kernel. For example, a 5×5 convolutional kernel with 16 regularly spaced holes is equivalent to a 3×3 kernel with a dilation rate of 2. Dilated convolutions result in a large receptive field without the decrease in resolution introduced by pooling.

Yu and Koltun [89] use dilated convolutions in their proposed architecture. The final two pooling layers of VGG are removed and dilated convolutions are used for all following convolutional layers. Earlier convolutional layers are unmodified, allowing initialisation with weights pre-trained on a traditional VGG architecture. This approach produces higher-dimensional feature maps than with pooling, at 64×64, but the maps are still of a significantly lower resolution than the input image. The authors also propose a *context module* designed to aggregate contextual information from multiple scales. The module stacks a series of dilated convolutions with differing dilation rates and takes feature maps from the encoder as input. Results show that the context module improves segmentation performance.

Dilated convolutions are also explored in the *DeepLab* segmentation pipelines [90, 91, 92, 93]. The first and second versions of DeepLab incorporate CRFs and these elements are discussed in Section 2.6. In this section, only the convolutional neural network portion of the pipeline is reviewed. DeepLab v1 [90] and v2 [91] incorporate dilated convolutions into the network architecture in a similar way to [89]. Dilated convolutions are referred to as atrous convolutions in this work, following the naming convention in [94]. In the second version [91], the authors also address the problem of different scales in the input space. One proposed solution involves parallel networks that each take as input a differently scaled version of the same image. Weights are shared between the parallel networks. As in [89], the network used in DeepLab v2 results in a higher resolution feature map than if dilated convolutions were not used. However, the resolution is still less than the original input space and bilinear interpolation is used to upsample the features to the correct resolution. A second solution involving atrous (or dilated) convolutions is also proposed named atrous spatial pyramid pooling. This approach inputs a feature map to parallel atrous convolutional layers, each with different dilation rates. This allows information from varying spatial scales to be incorporated into the extraction of semantic information.

Atrous convolution for semantic segmentation is revisited in DeepLab v3 [92]. The authors investigate the efficacy of a cascade of atrous convolutions, with varying dilation rates. The approach is similar to that proposed in

[89], but instead of operating on softmax distributions, it operates on feature maps. Additionally, the parallel atrous spatial pyramid pooling from [91] is revisited with the inclusion of image-level features. These features capture long-range information and incorporate global context into the network, similar to ideas presented in [95, 96]. The authors also note the importance of batch normalisation [97] for training the model. Batch normalisation is explained in Section 3.2.2.5. Further improvements are made with DeepLab v3+ [93], which includes a decoder network to help refine object boundaries. An interesting feature of this work is that a user can make trade-off decisions between segmentation accuracy and run-time. This is achieved through user control of the artous convolution parameters, which set the resolution of the feature maps produced by the encoder network.

Zhao *et al.* [96] combine local and global features to incorporate short-range and long-range contextual information. Global context is considered by using a pyramid pooling module that extracts representations from the input feature maps at different scales. These representations are upsampled and concatenated with the original feature maps and pixel-level predictions are made. The authors make use of an auxiliary loss in addition to the primary loss, in order to help network training. This is similar to the auxiliary losses in GoogLeNet [66], but with a ResNet-style base network [69].

2.3.2.4 Weakly Supervised Dense Prediction

Obtaining pixel-level labels for training data is a tedious and laborious task. Weakly supervised approaches for segmentation [98, 99, 100] ease this burden by learning only from bounding box or image-level ground truth labels. Image-level labels are used as the only supervisory signal in [98]. The approach works off the knowledge that at least one pixel in the image will have a ground truth label matching the image-level label. Convolutional neural network features are aggregated, allowing the model to weight pixels that are important for classification higher than those that are less important. This approach teaches the model to discriminate the salient pixels from background pixels, allowing dense classification at test time. The *BoxSup* framework [99] uses bounding box object labels to iteratively generate proposal segmentation regions, which are then used to update the CNN parameters. As the network weights are updated, the proposed regions are improved, providing better supervision for the next updates to the CNN. The authors of [100]

treat weak supervision as a form of input label noise. The de-noising strategy proposed is a recursive training scheme that uses previous training stage predictions as supervisory signals in the current training stage.

2.4 Metric Learning

The ability to determine how similar given inputs are is an important problem in robotics and beyond. Metric learning aims to allow this by learning a distance function over inputs. In this thesis, the focus is on image data and the distance function learned should be a measure of the semantic similarity between given images. Distance measures in the image space, such as the Euclidean distance between pixel intensities, are largely meaningless in terms of high-level semantic information. As such, metric learning algorithms generally learn a transformation from the image space into a semantically meaningfully metric space. Standard distance measures, such as the Euclidean distance, between examples in the transformed metric space should be a measure of the semantic similarity between examples. In this section, commonly used loss functions are described, followed by a discussion on relevant metric learning models and algorithms.

2.4.1 Siamese Networks and Contrastive Loss

Seminal works in the domain of metric learning include those that use Siamese networks [101] and contrastive loss [102, 103]. Siamese networks consist of parallel networks that share weights. For example, a pairwise Siamese network is made up of two parallel networks that each take a single image as input. In practice, this is achieved by passing the pair of images through the same network. The objective of contrastive loss is to minimise the distance between pairwise examples of the same class and to penalise pairwise examples that belong to different classes by pushing them apart. Such contrastive loss-based metric learning methods have been applied to face verification [103] and dimensionality reduction [102].

2.4.2 Deep Metric Learning

This section discusses metric learning approaches that make use of deep convolutional neural networks.

2.4.2.1 Pairwise Networks

Deep pairwise Siamese methods [104, 105] operate on both positive input pairs (examples from the same class) and negative input pairs (examples from differing classes). In general, the pairwise loss function aims to minimise the distance between examples in positive pairs and to push apart examples from negative pairs, up to the some distance margin. Yuan *et al.* [105] propose a cascaded embedding approach that enables the sampling of different levels of *hard* image pairs. A hard pair is one that is not yet well learned by the model and will result in informative updates to the network weights. A cascade of networks with different complexities is used to perform the sampling at different levels of hardness.

Pairwise CNNs are often difficult to train. Such networks operate on absolute distance and require the selection of a fixed margin for negative pairs. This makes it difficult for pairwise networks to model variations for different classes. Loss functions that operate on relative distance, such as triplet loss (see Section 2.4.2.2), are generally easier to train and result in better performance.

2.4.2.2 Triplet Networks

Deep metric learning approaches that make use of a triplet-based loss are perhaps the most common in the literature [17, 106, 18, 19, 20, 21, 107, 108]. A triplet is a trio of examples, generally selected such that two of the examples are of the same class and the third is of a different class. One of the positive examples is referred to as the anchor. The loss function aims to pull the anchor and the other positive example nearer than the anchor and the negative example, by some margin. A hinge function is generally used such that the loss becomes zero when that condition is met. Deep triplet-based approaches share similarities with classical large margin nearest neighbour (LMNN) [109] metric learning methods, which aim to minimise the distance

between training examples and selected target neighbours, while increasing the distance to nearby examples from different classes.

Random selection of triplets and performing a single distance comparison per sampled training example result in inefficient model training. If a triplet is selected that returns a zero loss from the hinge function, no update to the network weights can be made. The process of selecting difficult triplets, that is, triplets for which the negative example is significantly closer to the anchor than the positive example, is called hard negative mining. Semi-hard negative mining is carried out in the work by Schroff *et al.* [18]. Rather than mining for hard examples over the entire training set, which is computationally expensive, Schroff *et al.* mine for hard examples within a mini-batch.

Song *et al.* [19] propose a lifted structured embedding that allows for efficient calculation of the dense distance matrix within a mini-batch. This enables the best, or hardest, negative example within the mini-batch to be utilised for each sampled pair of positive examples. Another intra-batch method, named N-pair loss, is proposed by Sohn [20]. This loss function is a generalisation of triplet loss that allows comparisons to multiple negative examples for a given positive pair. These intra-batch methods significantly increase the efficiency of the updates made to the network weights during training, compared to random sampling of triplets from the training set.

A smart mining method is proposed by Harwood *et al.* [21] that mines for triplets over the entire entire training set. This is made computationally feasible by use of a fast approximate nearest neighbour graph [110] that enables efficient approximate nearest neighbour search over large datasets in high dimensional space. The work proposed by Kumar *et al.* [23] suggests that the inclusion of a global loss term, together with triplet loss, can be beneficial to the overall performance of the model. Wang *et al.* [108] constrain the angle at the negative example point of the triangle formed by the triplet, resulting in a scale invariant loss function. The proposed method also has the advantage of better convergence during training.

2.4.2.3 Quadruplet Networks

Methods that sample four examples are referred to quadruplet approaches [111, 112, 113, 114]. In general, these approaches sample a positive pair and a negative pair, with the loss function comparing the difference between these data pairs. The *Qwise* framework proposed by Law *et al.* [111] allows the

incorporation of knowledge from complex or semantically rich class labels. This enables application to challenging problems such as relative attribute learning and class hierarchy metric learning. An example of the former problem is ranking images of faces in terms of how much the the person in the image is smiling. The latter problem involves learning a metric space that obeys class hierarchy, such that examples from sibling classes, that is, examples with the same higher-level class label, are located nearer by than examples from non-sibling classes.

Histogram loss [112] considers information from all quadruplets formed within a training mini-batch. The loss function aims to minimise the overlap of two distributions: one formed by the similarities of positive pairs and one formed by the similarities of negative pairs. The distributions, or histograms, are approximated in a simple, non-parametric fashion, enabling easy optimisation of the model. Chen *et al.* [113] apply metric learning to person re-identification problems by recognising that quadruplet networks achieve better generalisation than triplet networks. The proposed work employs an online, margin-based hard negative mining method for efficient training of the model. Huang *et al.* [114] introduce a *position-dependent deep metric unit* that enables the learning of a distance metric that is adaptive to the local structure of the feature space. The hard quadruplet sampling technique results in both faster model convergence and improved performance.

2.4.2.4 Other Approaches

Cross-entropy loss is the standard loss function used for classification problems. The work proposed by Sun *et al.* [115] trains a model jointly with contrastive loss and cross-entropy loss for face recognition problems. Experimental results show that the combination of supervisory signals results in features with larger inter-personal variations and smaller intra-personal variations, compared to training with one loss function alone.

In contrast to Siamese network metric learning approaches that focus on only the local structure of data, Song *et al.* [22] consider the global structure of the feature embedding space by directly optimising a clustering metric. This approach avoids the potentially costly data pre-processing steps of arranging examples into pairs, triplets or quadruplets.

A different metric learning approach is proposed by Rippel *et al.* [116] that maintains a model of the distributions of semantic classes in the feature embedding space. Class discrimination is achieved in the feature embedding space by a loss function that penalises overlap in class distributions. The proposed loss function is easier to train than triplet-based deep metric learning approaches and converges in significantly fewer training iterations.

2.4.2.5 Pixel-wise Metric Learning

Though the vast majority of metric learning approaches extract a single feature embedding per input image, some methods have been developed for learning dense feature embeddings for images [117, 118, 119, 120]. These methods extract more than one feature embedding per image, such as one per pixel or one per downsampled image region. Harley *et al.* [117] propose an architecture that learns pixel-level feature embeddings, where the distance between different pixel embeddings is relative to the likelihood that those pixels belong to the same image region or object. The model is used together with a CNN trained for semantic segmentation to improve per pixel classification accuracy. Rather than standard semantic segmentation, Fathi *et al.* [118] use metric learning to perform instance segmentation, wherein each instance of an object is segmented separately. The loss function aims to minimise the distance between pixel embeddings belonging to the same object instance and to push apart embeddings belonging to different instances.

Pixel-wise metric learning is used by Chen *et al.* [119] to perform object-of-interest segmentation in video sequences. A modified version of triplet loss is proposed that adapts the conventional loss function for the problem domain. The authors argue that not all positive pairs of pixels belonging to an object should be pulled close together, since an object may be composed of various parts of differing visual appearance. As such, the triplet loss is modified so that only the positive and negative examples that are nearest the anchor pixel are considered. Li *et al.* [120] perform unsupervised object segmentation in video frames by transferring the knowledge from a pixel embedding network [118] that groups together pixels belonging to the same instance. The model is trained only on static images, but is deployed on video sequences.

2.4.2.6 Unsupervised Representation Learning

Similarity learning with Siamese networks has also been used for training networks in an unsupervised manner [121]. Unsupervised learning refers to the use of training data that is unlabelled. A supervisory signal can be generated by using other relationships in the data. Wang and Gupta [121] use temporal information from videos to create this supervisory signal. An image patch containing an object is extracted from a given video frame and treated as the anchor image. The object is tracked through subsequent frames and another image patch surrounding the object is extracted as the positive example. Positive pairs are selected such that there has been sufficient movement, meaning that the anchor and the positive example will show the same object from different poses. A negative patch, that is, one that doesn't contain the anchor object, is also extracted. Triplet loss is used to pull the anchor and the positive patch nearer than the anchor and the negative patch. Results show that training a network in this unsupervised manner is an effective tool for model pre-training for problems such as object detection and surface normal estimation.

2.5 Open Set Problems

In real-world classification problems, the output of the classifier is not the end goal. For example, in robotic problems a prediction is used to enable robotic action. Further, many real-world problems are open set; it cannot be assumed that all observed data will be from the known training distribution. As such, it is important that the confidence with which predictions are made can be measured. This facilitates important functions such as the detection of examples that are from outside of the training set distribution. It also enables the ability to learn from difficult or novel examples in an active manner, such that the true distribution of data can be understood. In this section, literature relating to the problems of novelty detection, open set recognition and active learning are reviewed.

2.5.1 Novelty Detection

The task of detecting examples that belong to classes that are outside of the training set distribution is known as novelty detection. This is an important problem in many real-world applications, in which silently failing when observing unknown data is not acceptable. These applications include robotics [122], medical diagnostics [123], video surveillance [124] and natural scene analysis [125]. Novelty detection is closely related to the problems of anomaly detection and outlier detection. The three terms are often used interchangeability in the literature. In this thesis, novelty detection is used to refer to the detection of examples belonging to novel or unknown semantic classes. These novel classes are not necessarily anomalies or outliers in the true distribution of data, but are outside of the known training set distribution. On the other hand, anomaly and outlier detection are the problems of detecting examples that are in some way inconsistent with typical or expected data. This has applications in fraud and intrusion detection in electronic systems [126]. Categories of novelty detection methods include *probabilistic approaches*, *distance approaches*, *domain approaches* and *information-theoretic approaches*

2.5.1.1 Classical Novelty Detection

Probabilistic approaches to novelty detection [127, 128] often aim to estimate the probability density function of expected data. Boundaries separating expected data from novel or unknown data can be calculated using the density functions. Probabilistic methods used in this category of novelty detection approaches include mixture models [129, 130], non-parametric kernel density estimation [131, 132] and negative selection [133, 134]. Probabilistic approaches have the advantage of being based on well-grounded mathematics. However, such approaches tend to struggle in sparsely populated feature spaces that occur when the dimensionality is high or the training data is limited [135].

Novelty detection based on distance approaches [136, 137] assume that novel data will be located far from known data in the feature space. Such approaches also generally assume that examples belonging to known classes will be tightly clustered in the feature space. Example techniques include those based on neighbours [137, 138] and clustering [139]. Distance-based

novelty detection methods require meaningful distance metrics in order to measure the similarity of examples.

Domain approaches define a decision boundary based on the known data. The decision boundary is used to perform classification of examples as known or novel. In other words, domain approaches treat novelty detection as a binary classification problem. Perhaps the most common technique used to achieve classification of novel examples is a support vector machine [140, 141, 142, 124]. A detailed discussion on support vector machines can be found Section 2.1.2.

The information content of data is analysed by information-theoretic approaches to novelty detection [143, 144]. The assumption is that the information content of the dataset is significantly changed by the presence of novel data. An example approach is computing the entropy of all data, then finding the subset of examples whose removal from the dataset results in the greatest drop in entropy [135]. Such approaches are computationally expensive and can perform poorly if the number of out-of-distribution examples is significant.

2.5.1.2 Novelty Detection with Deep Learning

A simple novelty detector for deep convolutional neural networks can be formed by thresholding on the confidence with which the softmax classifier makes a prediction [145]. However, the performance of such an approach is limited, as softmax classifiers have poor confidence calibration [146] and tend to make high confidence predictions, regardless of whether or not those predictions are correct. Since the average confidence of a softmax CNN is often greater than the average accuracy, it can be difficult to discern novel examples by analysing the softmax scores.

An open set version of the closed set softmax classifier is proposed by Bendale and Boult [147]. The proposed approach, named OpenMax, argues that spatial relationships in the class activation space can be used as a measure of novelty. Training set class activation vectors from the output of the final fully connected layer in a CNN are recorded. The correctly classified training examples are used to compute a per class mean activation vector. Due to a lack of uniformity across classes, distances between an example activation vector and mean activation vectors cannot be simply thresholded to detect novelty. The proposed method fits a per class meta-recognition model on the

distances between the mean vectors and the furthest away training examples that were correctly classified. The per class models are used to measure novelty and refine the activation vector by revising class activations and introducing a pseudo-activation representing novel and unknown classes.

An out-of-distribution detector for convolutional neural networks, named ODIN, is proposed by Liang *et al.* [148]. The approach aims to push the softmax scores of a pre-trained CNN from known and novel classes apart. This allows easier detection of novel examples by thresholding on softmax confidence scores. This is achieved through two methods: softmax temperature scaling and input pre-processing. The former involves scaling the softmax scores by a constant prior to normalisation, which has been shown to result in better confidence calibration [146]. The latter method involves using the gradients of the softmax scores to make small perturbations to the input image. The authors find that in general, these perturbations result in greater change in the resultant softmax scores for known class examples than for novel class examples.

Other deep learning approaches to novelty detection include a density-based confidence score [149], rather than one based on softmax scores. Novelty detection via metric learning with contrastive loss has also been investigated [150]. A downside of this approach is that the classification performance of the model is poor, meaning that on its own, the model isn't suitable for open set recognition or simultaneous classification and novelty detection. Kliger and Fleishman [151] use a generative model to generate examples from known and novel distributions, which are used to train a multi-class discriminative model.

A related to problem to novelty detection is classification with abstention [152, 153, 154, 155, 156, 157]. In this problem, a classifier is able to abstain from making a prediction if it believes that prediction is likely to be incorrect. In other words, the classifier is able to say "I don't know". This is important in applications where classifier mistakes can be costly, such as robotics and medical diagnostics. In these fields, it is often more desirable for a classifier to make no prediction than an incorrect prediction.

2.5.2 Active Learning

The process of actively querying a user or oracle for class labels is known as active learning. Compared to a conventional supervised learning setting, in which the starting point is generally a dataset of labelled training examples, in an active learning the starting point is often a large dataset of unlabelled examples. It is assumed that acquiring a label for a piece of data is costly and as such, only the most informative examples should be labelled. In this thesis, active learning is considered an open set problem. This is because the proposed work is concerned with learning the unknown and novel distribution of data in the environment, following initial model training and deployment.

The primary problem for active learning algorithms is the method of selecting examples to be labelled. This process is referred to as query selection. In active learning problems, the aim is to learn from as few labelled examples as possible. As such, the selection of the best and most informative examples for querying is vital. The definition of "best" or "most informative" depends on the query selection approach used. In general, unlabelled examples that have high uncertainty and are representative of many other unlabelled examples, are good candidates for query selection.

A common class of query selection approaches are known as uncertainty methods [158]. These algorithms favour the selection of examples that when classified by the existing model, are done so with high uncertainty or low confidence. The rationale behind such query selection approaches is that acquiring labels for examples that are already well classified is wasteful; examples that the model does not currently know how to classify will be more informative to the model when labelled. This simple approach can be an effective method of query selection and is arguably the most common in the literature [159]. However, there are drawbacks to uncertainty methods. Such approaches assume that the confidence with which a prediction is made is relative to the probability of that prediction being correct. In other words, uncertainty methods assume that the classifier has good confidence calibration. However, this is not always the case [146]. Further, uncertainty approaches do not consider how informative an example is to other unlabelled examples. It may be that a highly uncertain example is an outlier and not representative of the distribution of "normal" data. Such an example may not be particularly informative.

The most simple uncertainty-based method of query selection is to select the example that has the least confidence [158, 160]. In other words, the example with the smallest maximum class probability is selected for labelling. This approach only considers a single class probability for each example, ignoring all other available informative in the probability distributions. Margin-based uncertainty methods [161] consider more information by finding the difference in probability between the most likely and second most likely class label. The example with the smallest margin is selected for querying. All information in the probability distribution is considered by approaches that query examples based on the Shannon entropy [162]. Examples that have the largest entropy are selected for querying.

Another class of query selection algorithms are those based on decision-theoretic methods. Rather than simply selecting examples for querying based only on their individual class probabilities, decision-theoretic approaches consider the expected change if a given example is labelled. An example decision-theoretic query selection algorithm is one based on the expected gradient length [163]. Such an approach aims to measure the expected change to the model that will imparted by a given label query. A labelling that results in a large training gradient has caused a significant change in the model and is likely informative. Another approach is to select the example that results in the largest reduction in the expected error [164]. However, this type of query selection is computationally expensive [159], requiring both estimation of the expected error of the unlabelled examples and iterative retraining of the model. This makes such approaches unsuitable for many problem domains, including robotics.

Several recent works have investigated active learning in the context of deep learning [165, 166, 167, 168, 169]. Wang *et al.* [165] propose a dual labelling approach named *cost-effective active learning* (CEAL). As with conventional active learning, a small number of the most informative examples are selected for labelling. Additionally, the CEAL algorithm automatically assigns labels to examples that are classified with high confidence by the CNN. This allows the training algorithm to fine-tune to network weights using a larger number of labelled examples than would otherwise be available.

Fang *et al.* [168] propose a novel active learning approach for natural language processing problems. The query selection process is framed as a reinforcement learning problem. The policy learned by the reinforcement learning model is a decision policy that determines which examples should be

selected for labelling. A generative-adversarial approach to active learning is proposed by Zhu and Bento [167]. Unlike conventional active learning approaches, which use the outputs of a learner to select queries from a pool of unlabelled data, the generative-adversarial active learning approach generates new examples for labelling. The synthesised examples should be informative, as they are adapted to the current state of the model.

Sener and Savarese [169] propose a query selection approach based on coreset methods. The algorithm aims to find a small subset of examples that allow the model to be trained such that it is competitive over the entire dataset. As the labels of the dataset are unknown, an approximation to the k-centre (facility location) problem is used to find the subset. Stark *et al.* [166] apply active learning to the special case of CAPTCHA recognition with CNNs. The proposed approach requires no human input during label querying, as it takes advantage of the automatically generated signal received by solving a CAPTCHA.

2.6 Structured Prediction with Probabilistic Graphical Models

Although deep learning methods are able to achieve very impressive results on semantic segmentation challenges, they have limited ability to incorporate higher-levels of context into the inference process. The topic of structured prediction aims to include information about the structure of the data into learning and inference algorithms. This is useful for prediction tasks in which the output is not a single scalar but a structured object. One example of structured data is speech. In order for a machine to recognise and understand a sentence spoken by a human, it is important to consider the structure of the sentence and the relationship between the phonemes and the word they make up, as well as between the words and the sentence they produce. Recognising speech on a phoneme by phoneme basis ignores all context of the speech and results in poorer recall. In a similar vein, considering only a local patch in a image in order to determine to which semantic class a pixel belongs, ignores the natural structure of an image and all context of where the information in that pixel sits in the real-world.

Probabilistic graphical models are an approach to structured prediction wherein the data and outputs are arranged in a graph structure and nodes

of the graph represent either measurable quantities from the data or random variables defined over the output space. Edges connect nodes in cases where the interaction between the nodes is considered valuable to the prediction. Each edge in the graphical model has a cost associated with it, defining the importance of that particular interaction. The job of the probabilistic graphical model's inference algorithm is to find a prediction that minimises the cost, or energy, of the entire graph. For example it may be known that neighbouring nodes often have the same desired output value as each other, in which case the cost for short edges will be greater than that of longer edges to non-neighbouring nodes. The inference algorithm will attempt to minimise the energy (cost) of the system by encouraging neighbouring nodes to take the same output value. Minimising the energy is equivalent to maximising the probability, as the probability is formulated as the result of the natural exponential function with the negative energy taken as input. This is discussed in detail in Section 3.3.1. Common probabilistic graphical models are the Markov random field (MRF) [12], which models the joint probability, and the conditional random field (CRF) [13], which is an MRF formulated to model the conditional probability of an output given observed data.

A common graph structure is a pairwise graphical model, which considers only two energy types: unary and pairwise. The unary terms represent the belief at a node when considering no interactions with other nodes. These initial per-node beliefs can be provided by some other classifier, such as a CNN. In the case of semantic segmentation, a pairwise term considers the energy of assigning labels to nodes that are connected by an edge.

2.6.1 Model Inference

Probabilistic graphical model inference refers to the process of finding the output state that minimises the defined energy function. In the case of semantic segmentation, it involves converging on a particular pixel-wise labelling for which the energy is a minimum, that is, the maximum a posteriori labelling. This is a non-trivial problem. For example, a VGA sized image with 307,200 pixels that can each take one of 20 semantic labels has 20^{307200} different possible labellings. Energy minimisation is further complicated by

loops found in graphs that are not linear chains. As such, exact energy minimisation is often intractable and a number of different inference algorithms have been development for approximate energy minimisation.

The first class of energy minimisation to consider is based on graph cuts. Ford and Fulkerson [170] propose a method to find the maximal flow in a network that is far simpler than the naive linear programming approach. Graph cut approaches for energy minimisation that are useful for computer vision problems were introduced by Boykov *et al.* [171]. Two graph cut-based methods that provide a fast and approximate solution are proposed: α-expansion and $\alpha\beta$ swap. In an iteration of α-expansion, a particular label α is able to gain nodes, that is, non-α nodes are able to change their labels to α. The expansion move that results in the greatest decrease in energy is calculated via graph cuts. Since only α and non-α labels are considered at a given iteration, the optimisation is a binary problem. The $\alpha\beta$ swap algorithm allows only nodes currently labelled as α or β to swap their labels to the other, at a given iteration. This algorithm can be used when pairwise terms are not metric, but it provides no guarantees about closeness to an optimal solution.

Messaging passing methods are another popular form of energy minimisation for random field models [172, 173, 174]. One of the simplest and most popular message passing algorithms is belief propagation [173], which has two similar variants: sum-product belief propagation [175] and max-product belief propagation [176]. The former works to find the best labelling at each node, while the latter works to find the best labelling globally. In belief propagation, messages are passed between nodes via edges. Nodes pass messages to other nodes with information about what that node believes about its state based on its own unary terms, connections with other nodes and previous messages received. Messages are passed over a number of iterations until convergence, at which point the belief for each node can be determined from the unary term and the converged incoming messages. Belief propagation will find an exact solution for graphs with no loops and an approximate solution for graphs with loops. When loops are present in the graph, the algorithm is named loopy belief propagation [177].

2.6.2 Conditional Random Fields

A class of probabilistic graphical models that are extremely useful for prediction problems and are widely applied to semantic segmentation are conditional random fields (CRFs) [13]. These random fields model the conditional probability of the output random variables conditioned on the input, or observed data. In general, edges in a conditional random field are undirected. This formulation, comprised of separate sets of nodes relating to the output and observed variables, lends itself well to vision problems. This is because arbitrary functions defined over the observed variables, such as the input image, can be easily incorporated. The CRF framework allows properties such as pixel location, colour and texture to be directly included in the model.

2.6.2.1 Adjacency CRFs

Conventional CRFs for semantic segmentation are of a grid or adjacency-type structure, wherein nodes are connected only to nearby or neighbouring nodes [178, 179, 180, 181]. This has the effect of enforcing local smoothness in labellings. Shotton *et al.* [178] use a CRF model where every pixel is a node with edges arranged in a four-connected grid. A pixel is connected by edges to the four adjacent pixels in the horizontal and vertical directions. Along with the pairwise connections, the CRF model incorporates texture, colour and location cues to obtain an accurate segmentation.

2.6.2.2 Fully Connected CRFs

The adjacency structure conditional random fields of [178] do well in finding a smooth local labelling, however, they do not incorporate high-levels of contextual information from long range connections in the image. A fully connected conditional random field [182, 183, 184] aims to incorporate this information by having an edge between every pair of pixels in the image. That is, every pixel is connected to every other pixel. Such a model incorporates both short range information to enforce local smoothness and long range contextual information to correct erroneous labellings and better delineate object boundaries.

Inference with conventional graph cut or belief propagation methods becomes intractable when so many edges exist in the CRF. Krähenbühl and

Koltun [182] propose a mean field approximation approach to efficient inference in a fully connected CRF. Efficient message passing is achieved through filtering over high-dimensional simplexes, which aims to minimise the KL-divergence between the true energy and an approximate distribution. The CRF includes two pairwise terms: a local spatial term to encourage nearby pixels to take the same label and an appearance term that includes pixel colour information, thereby encouraging pixels with similar appearance to take the same label. In order for the efficient message passing algorithm to work, however, the potentials are constrained to be Gaussian. Campbell *et al.* [184] relax this constraint by learning non-parametric potentials directly from the training data and encoding the desired potentials as Gaussian kernels.

2.6.3 Joint CNN and CRF Models

There has been great interest in combining the strong representational power of a deep convolutional neural network with the context and structure-aware models of conditional random fields [90, 91, 185, 186, 187, 188, 189]. Chen *et al.* [90, 91] refine the segmentation produced by a fully convolutional neural network by use of a conditional random field as a post-processing step. The pixel-wise beliefs from the CNN are used as the unary terms in the CRF, resulting in better object boundary detection and the removal of some spurious labellings. A fully connected, dense CRF [182] approach is taken, wherein each pixel is connected by an edge to every other pixel in the image. The reason for this implementation is that grid or adjacency graphical models have a smoothing effect on the segmentation, which is only desirable when the unary terms contain local variability. Unary terms from deep convolutional networks, however, are generally already smooth. As such, the purpose of the CRF is to include longer range contextual information to aide with object boundary refinement and the correction of larger regions of erroneous classifications.

Similar work by Zheng *et al.* [185] directly incorporates the fully connected CRF as layers in the convolutional neural network architecture, rather than applying the CRF as a post-processing step. The mean field inference algorithm of [182] is formulated as a recurrent neural network, the layers of which are connected to the conventional CNN layers.

The work of [185] is extended in [187] to include higher order potentials in the system. Two types of higher order potentials are used: one to help refine object boundaries and another to assist in scenarios where the unary terms are too poor for a pairwise CRF to be useful. The former is achieved by over segmenting the image into superpixels [190] and defining a higher order clique over each superpixel in the CRF. In other words, since it is likely that all pixels within a superpixel belong to the same semantic object, the CRF encourages pixels belonging to same superpixel to take the same class label. This is implemented only as a soft constraint, as there may be errors in this assumption. In order to improve regions with poor unaries, a state-of-the-art object detector is used to generate object proposals and a rough foreground segmentation of these proposals is obtained via the GrabCut method [191]. These proposals are incorporated into the CRF as higher order cliques, which encourage the underlying pixels to take the label of the object proposal when the detection is deemed to be correct. This process results in an improvement in terms of segmentation performance when compared to the unary and pairwise model of [185]. However, this approach does not address the underlying issues that cause the original spurious results, but rather utilises an additional state-of-the-art recognition model to form an ensemble learning system.

A slightly different approach to joint deep CNN and CRF models is taken by Lin *et al.* [186]. Context is employed in the CRF in a patch-to-patch and patch-to-background manner. The former is concerned with the interaction of two image patches, while the latter is concerned with the relationship between an image patch and an area of background. Unlike other work, the pairwise potentials are learned directly via the CNN, which attempts to model the relationships between semantic image patches. These general pairwise potentials complicate the joint learning of the CNN and CRF parameters. As such, an approximate piecewise training methodology is used for the sake of efficiency.

Chapter 3

Fundamentals

In order to make this thesis as self contained as possible, this chapter details the fundamental concepts, algorithms and models that are used in the subsequent chapters. Firstly, optimisation algorithms for machine learning are presented. This is followed by a detailed discussion on convolutional neural networks, including network architectures and training procedures. Finally, the details of conditional random fields for the problem of semantic segmentation are presented.

3.1 Optimisation for Machine Learning

Machine learning models are trained using optimisation algorithms by updating the model parameters such that a loss function, also known as a cost function or error function, is minimised. Due to the use of complex non-linear models with a large number of parameters, as well as large training datasets and complex target domains, the loss functions generally cannot be minimised using direct methods. As such, iterative optimisation approaches are used to minimise the loss. Figure 3.1 illustrates the operation of an iterative optimisation algorithm. Beginning at some parameter initialisation, the model samples training data and computes an error term, the optimisation algorithm uses this error to compute the updates to the model parameters. After updating the parameters, data is once again sampled and the process repeats until the error converges. The state of the model at each iteration of the algorithm can be considered an approximate solution. Ideally, each iteration will yield an approximate solution that is closer to an optimal solution than at the previous iteration.

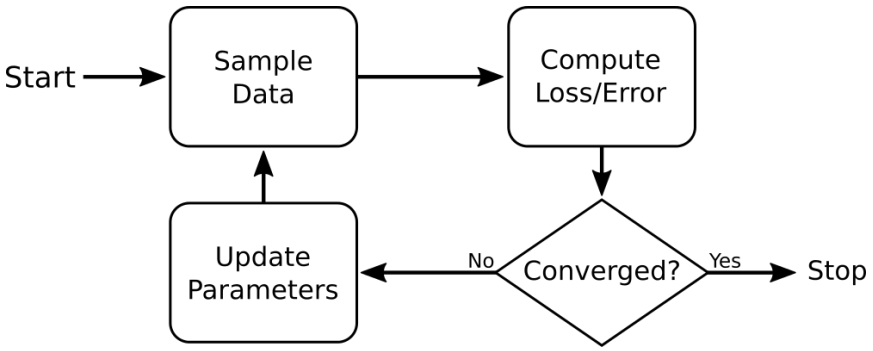


FIGURE 3.1: Overview of iterative optimisation algorithms.

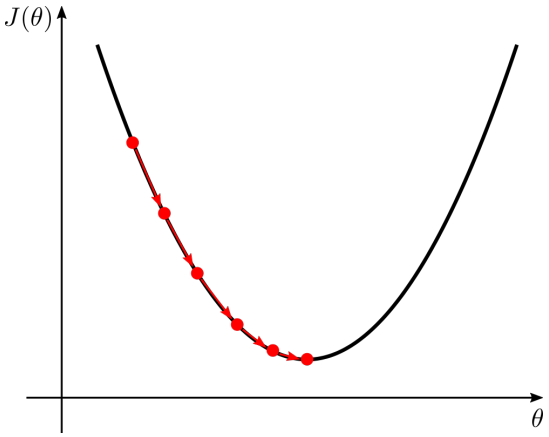
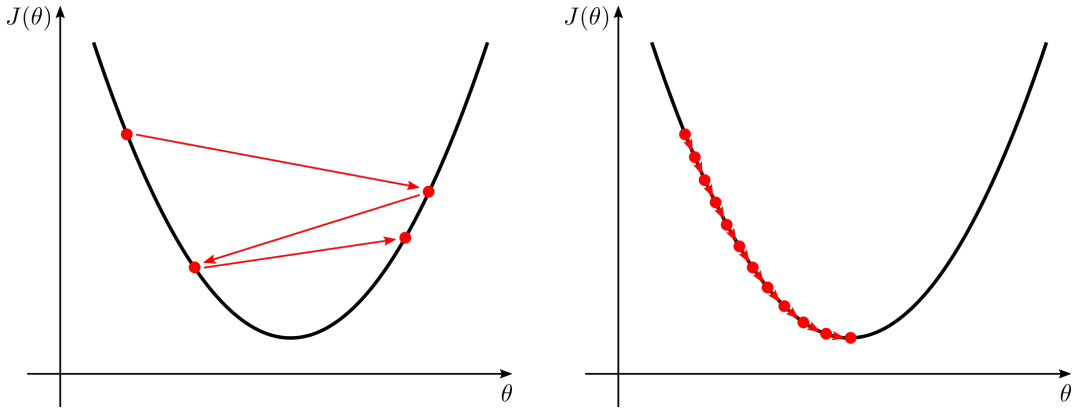


FIGURE 3.2: Gradient descent with a single parameter model.

The remainder of this section focuses on the gradient descent algorithm, which is a commonly used first-order optimisation method. The algorithm is *first-order* as it takes the first-derivatives of the objective function. This is compared to higher-order methods that take higher-order derivatives, such as the second-order Newton's method.

3.1.1 Gradient Descent

The most popular iterative optimisation algorithm in machine learning is gradient descent. This algorithm minimises a loss function by taking steps in the direction of the steepest negative gradient. In other words, the partial derivatives of the loss function with respect to the parameters are used to update the parameters such that the loss moves in the direction of steepest descent. This is illustrated in Figure 3.2 with a simple one parameter example. When minimising a loss function J, the updates made to the model parameters $\boldsymbol{\theta}$ are given by the formula in Equation 3.1. The subscript of the



(A) Learning rate too large. (B) Learning rate too small.

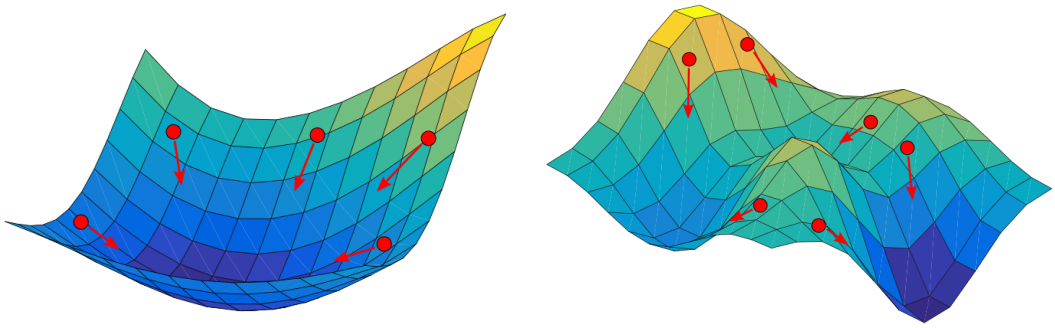
FIGURE 3.3: The importance of careful selection of the learning rate for the gradient descent algorithm.

parameters $\boldsymbol{\theta}$ denotes the time step of the iterative algorithm. For example, the gradient of the function with respect to the parameters at time step t is denoted as $\nabla J(\boldsymbol{\theta}_t)$. The size of the step taken is proportional to the gradient and the learning rate $\alpha \in \mathbb{R}_{>0}$.

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \nabla J(\boldsymbol{\theta}_t) \tag{3.1}$$

Appropriate selection of the learning rate α is vital to the performance of the algorithm. If the learning rate is set too large, the loss may bounce around and diverge. Conversely, if the learning rate is too small, the algorithm will require a large number of iterations and training of the model will take too long. Both of these scenarios are illustrated in Figure 3.3 with simple one parameter functions. Of course, machine learning models such as convolutional neural networks may have millions of trainable parameters.

Note that the examples shown in Figures 3.2 and 3.3 are convex functions, meaning that taking a step in the direction of the negative gradient moves the solution towards the global minimum. Realistic optimisation spaces are not convex and contain local minima. This means that the direction of movement and final location of convergence is highly dependent on the current state of the parameters. This is illustrated in Figure 3.4.



(A) Ideal convex space.

(B) Typical non-convex space.

FIGURE 3.4: In a convex space, any step in the direction of the steepest negative gradient will be towards the global minimum. In a non-convex space, the direction of a step and the final solution are dependent on the current location.

3.1.2 Variants of Gradient Descent

Iterative optimisation objective functions often decompose as the sum over errors of individual training examples. This allows for flexibility in how often, in terms of the amount of data that has been sampled, updates are made to the parameters. Three commonly used variants of the gradient descent algorithm are discussed in this section: *batch gradient descent*, *stochastic gradient descent* and *mini-batch gradient descent*.

3.1.2.1 Batch Gradient Descent

The standard version of the algorithm is batch gradient descent, which makes updates to the weights only after the entire training set has been sampled and contributed to the loss. The advantages of this approach are that the updates made are very stable, meaning that model convergence is consistent. However, this stability also makes the approach susceptible to becoming stuck in local minima. The approach can also be costly in terms of training time, as only a single update is made per pass over the training set. For large datasets and computationally complex models, such an approach is unsuitable.

3.1.2.2 Stochastic Gradient Descent

On the other side of the spectrum to batch gradient descent, stochastic gradient decent performs an update to the parameters from the sampling of a single training example. Training data is generally shuffled after each data

sampling or after the entire set has been sampled. For a model that can process one training example at a time and for a dataset containing n training examples, stochastic gradient descent makes n times more updates to the parameters in the same time period. This can result in significantly faster model training. However, computing gradients based on a single training example can result in noisy parameter updates, causing the loss to jump around. In general, stochastic gradient descent will require more update steps than batch gradient descent to converge. The trade-off between the required number of updates and the time taken to perform an update will depend on the task and the model.

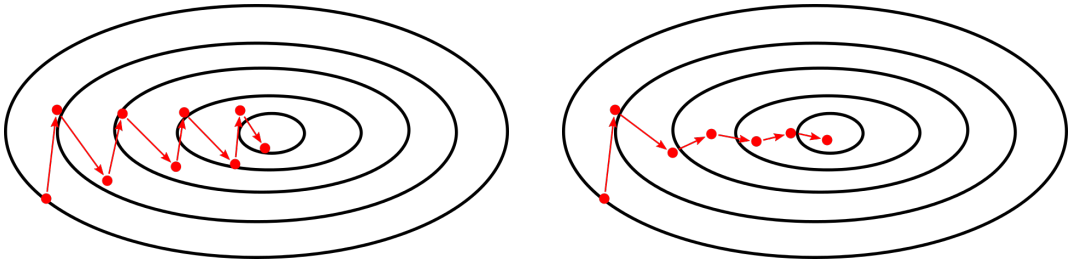
3.1.2.3 Mini-Batch Gradient Descent

A middle ground approach between batch gradient descent and stochastic gradient descent, mini-batch gradient descent performs an update after sampling a subset of β training examples. A subset is referred to as a mini-batch, or often simply a batch. If β is equal to one, the approach is the same as stochastic gradient descent. If β is equal to the size of the training set, it is equivalent to batch gradient descent. Batch sizes are usually selected to be small relative to the training set size. For example, a batch size of 50 for a training set size of 10000 would be reasonable. As with stochastic gradient descent, the training data is generally shuffled after sampling one mini-batch, or after the entire training set has been sampled. The stochastic nature of both stochastic gradient descent and mini-batch gradient descent means that it can be possible to jump out of local minima.

Mini-batch gradient descent provides a balance between the frequency of parameter updates and the stability of gradients used to compute the updates. As such, it is arguably the most widely used method, particularly for larger datasets. The parallel nature of convolutional neural networks trained with GPUs allows all examples in an appropriately sized mini-batch to processed simultaneously, making mini-batch gradient descent an efficient and appropriate approach for such models.

3.1.3 Momentum

Stochastic and mini-batch gradient descent result in noisy updates to parameters and jumpy steps in the loss space. The momentum method considers



(A) Without momentum. (B) With momentum.

FIGURE 3.5: Momentum can dampen oscillations in the updates and reduce the number of iterations required. A two-dimensional parameter space is shown as a contour plot.

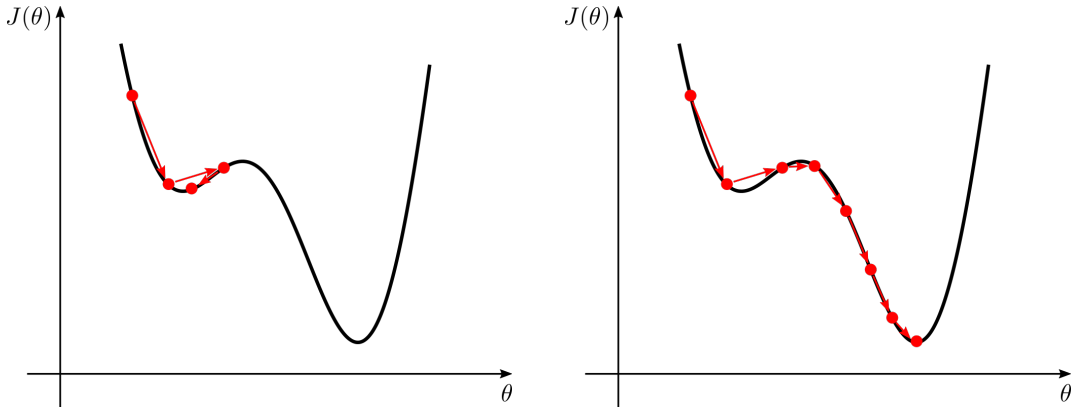
the direction of movement from previous steps in the calculation of the next step. A linear combination of the previous update and the current gradient is used to find the next parameter update. Including momentum in the optimisation algorithm dampens oscillations and can reduce the time take to reach convergence. This is shown in Figure 3.5 with a two parameter example. It can also be useful in avoiding becoming stuck in local minima, as the momentum from previous updates pushes the solution beyond the cusp of the local minimum. This is shown with a single parameter example in Figure 3.6. The gradient descent formula from Equation 3.1 can be extended to include momentum by breaking it into two parts: the update calculation step (Equation 3.2) and the parameter update step (Equation 3.3). The momentum factor γ, where $0 \leq \gamma < 1$, is treated as another hyperparamerer, similar to the learning rate α. The formula for gradient descent is sometimes written with the omission of the $\gamma - 1$ term and with the learning rate scaled by that amount in Equation 3.2.

$$\boldsymbol{\phi}_t = \gamma \boldsymbol{\phi}_{t-1} + (\gamma - 1)\nabla J(\boldsymbol{\theta}_t) \tag{3.2}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \boldsymbol{\phi}_t \tag{3.3}$$

3.1.4 Learning Rate Adaptation

It is common practice to reduce the learning rate during training. This allows the optimisation algorithm to take large steps towards the solution at the early stages of training when the loss is far away from minima and smaller steps at later stages when it is nearer the solution. In other words, the optimisation algorithm should slow down as a good solution is approached.



(A) Without momentum. (B) With momentum.

FIGURE 3.6: Momentum can help the gradient descent algorithm avoid becoming stuck in local minima.

The simplest method is to step down the learning rate by some factor at a set interval. For example, the learning rate may be multiplied by a factor of 0.1 every η training iterations. The AdaGrad algorithm [192] adjusts the base learning rate for each parameter separately at each time step. The previous gradients are summed for each parameter and used to attenuate the learning rate for frequent parameters and increase the learning rate for sparse parameters. Other approaches include AdaDelta [193] and RMSprop [194], which instead of accumulating all previous gradients, compute an exponentially decaying average. Adam [195] extends these approaches by incorporating a running average of the second moments of the gradients.

3.2 Convolutional Neural Networks

Deep learning with convolutional neural networks (CNNs) is all about the extraction of useful features from input data. Unlike other machine learning models that require features to be hand-crafted, CNNs extract features directly from the data by learning the feature extraction parameters simultaneously with the classifier parameters, in an end-to-end fashion. The bulk of the feature extraction work is carried out by convolutional filters that are organised into layers, which are stacked one after the other. A convolutional layer extracts features from the outputs of previous convolutional layers. Features extracted in the shallow layers of the network, such as the first layer that operates directly on the image, are generally low-level structural features, such as edges. The deeper into the network, the more high-level the extracted

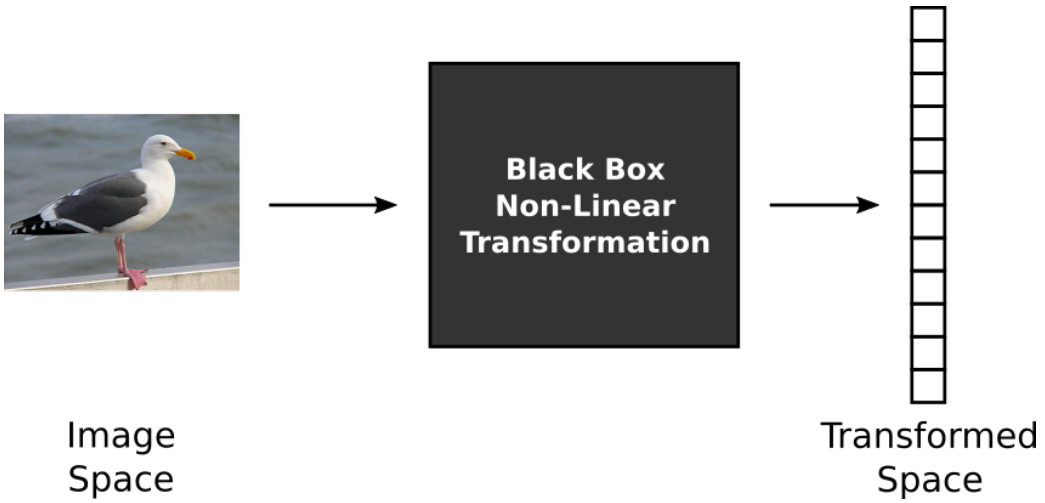


FIGURE 3.7: Black box model of a CNN. The network performs a transformation from the image space to a transformed space.

features become. The parameters of these layers, that is, the convolutional filters, are learned during network training.

A convolutional neural network can also be thought of as black box operation that performs a non-linear transformation from the image space into a reduced feature space, as shown in Figure 3.7. Depending on the problem domain, the transformed space may be a class activation space, in which each dimension corresponds to the likelihood of the input belonging to a particular class. Alternatively, the transformation could be to a more semantically rich space that encodes semantic relationships between different images.

High-level overviews of example convolutional neural networks are shown in Figure 3.8. The network in Figure 3.8A is the type of architecture used for classification problems. A series of convolutional layers extract increasingly meaningful features from the input image. The feature maps are downsampled throughout the network until all spatial information is lost. The final feature is passed into a classifier, which predicts a class label for the input image. The disposal of spatial information is by design; a CNN for image classification should be able to extract semantic information from the salient object in the image, regardless of where that object is located in the image. However, this loss of spatial information is not suitable for segmentation problems, which require the localisation of objects in the image.

Figure 3.8B shows the type of architecture used for segmentation problems. The input dimensions are restored in the output by following the downsampling section of the network with an upsampling section. These network

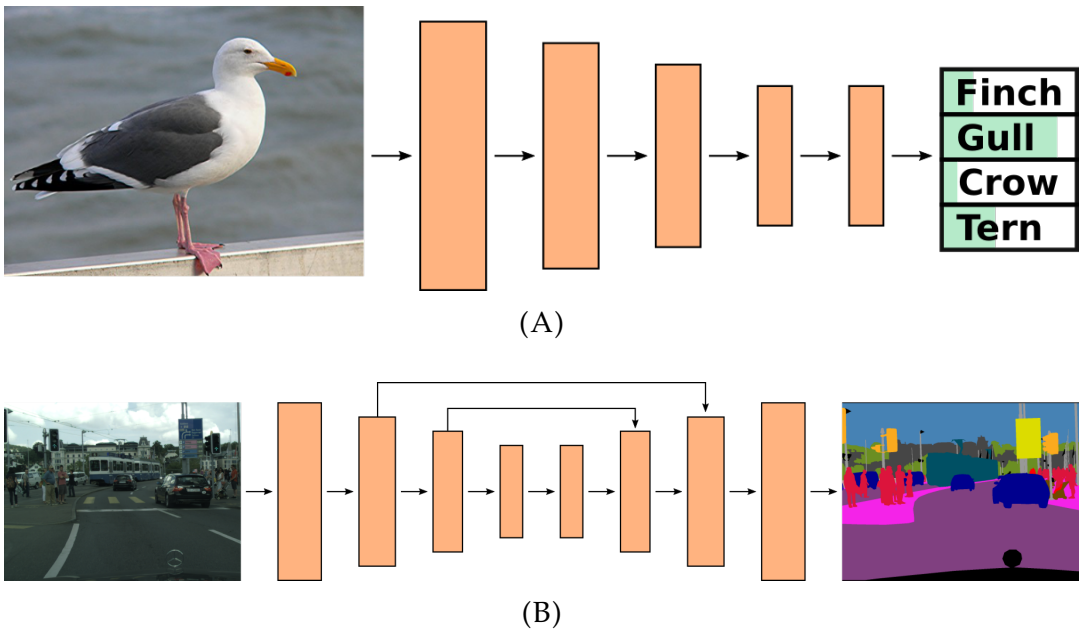


FIGURE 3.8: High-level overviews of network architectures for classification (top) and segmentation (bottom).

sections are known as the encoder and decoder, respectively. The nature of convolutions and the need to include some downsampling operations for memory efficiency and spatial invariance, means that some spatial information will always be lost in the encoder network. However, convolutional and sampling layers can be designed to minimise this effect. This is discussed in Section 3.2.2. Skip connections between encoder layers and decoders layers are also included in order to retain fine-grained spatial information.

3.2.1 Terminology

Commonly used terms relating to deep learning and convolutional neural networks are defined below.

- **Weights**: the trainable parameters of the neural network. These parameters are updated during network training such that some objective function is optimised.
- **Tensor**: a generalisation of a vector or matrix. Data in a CNN is often represented as an d-dimensional tensor.
- **Input data**: the input to the convolutional neural network. In the context of this thesis, the input data is always an image, which can be stored as a tensor. For example, an RGB image can be stored as a

$H \times W \times 3$ tensor, where H and W are the height and width of the image, respectively. Image data is often pre-processed before being input to the network. This is discussed in Section 3.2.4.

- **Labels**: ground-truth target values for training data, used by supervised training algorithms. For image data, there may be a single label per image or a label per pixel.
- **Layer**: the building block of convolutional neural networks. A layer performs an operation on an input tensor to create an output tensor. Common layer types are discussed in Section 3.2.2.
- **Features**: useful representations of the input data extracted by the convolutional neural network. CNN features may describe low-level structural information in the data, such as edges and corners, as well as high-level semantic information. However, features extracted by network layers may not always be interpretable as meaningful image cues, when analysed in isolation. The weights of convolutional layers (Section 3.2.2.1) and fully connected layers (Section 3.2.2.3) are learned such that useful features are extracted from the input tensors.
- **Activations**: the output tensors of activation layers, which apply non-linear functions to input tensors. Activation layers are discussed in Section 3.2.2.4.
- **Hyperparameters**: parameters that are not learned during training. Hyperparameters are set before the learning algorithm is carried out. CNN hyperparameters include the learning rate (Section 3.1.1), momentum factor (Section 3.1.3) and weight decay factor (Section 3.2.5.5).

3.2.2 Layers

Convolutional neural networks are made up of a series of layers. The most common of these layers can be categorised into five types: convolutional, sampling, fully connected, activation and normalisation. Of these five layer categories, convolutional and fully connected layers can be further categorised as feature extraction layers, which are made up of trainable weights. Commonly used layer types within these five categories are explained in this section.

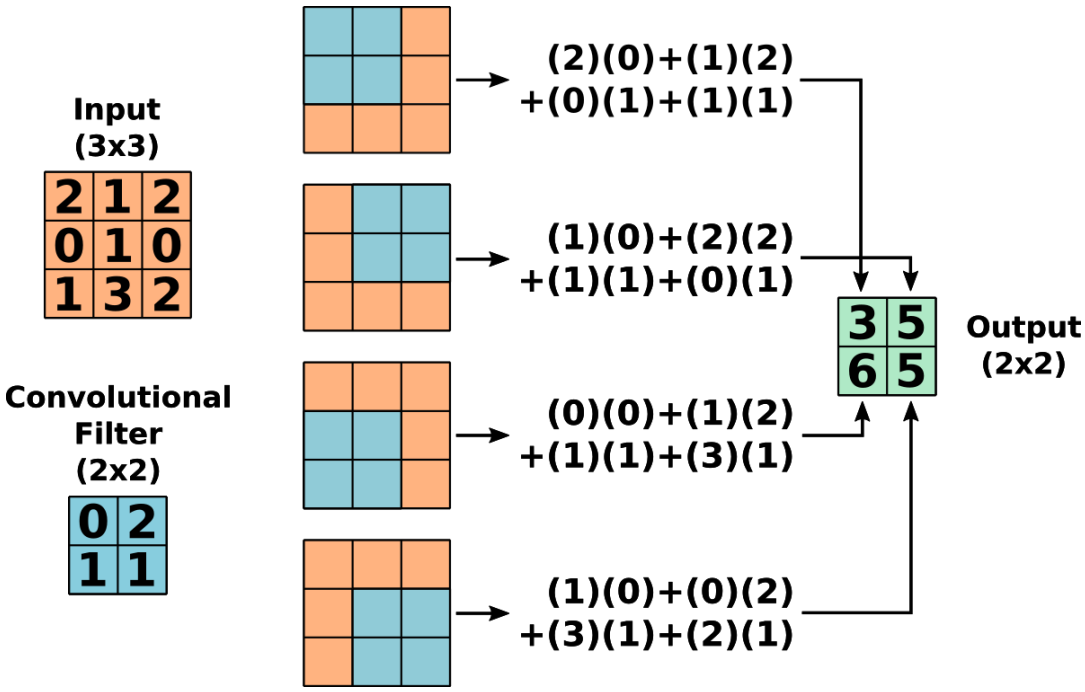


FIGURE 3.9: Image convolution with a 3×3 input and a 2×2 filter.

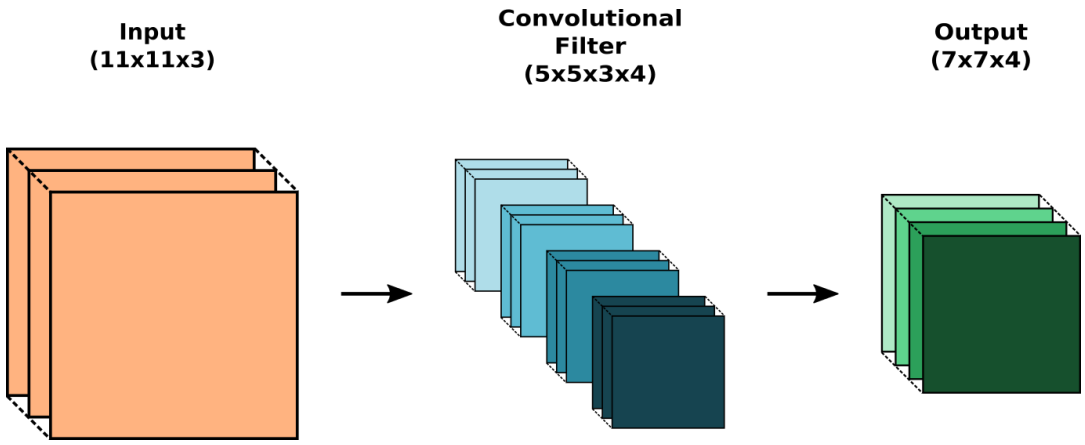


FIGURE 3.10: Convolutional layer with multiple input and output channels.

3.2.2.1 Convolutional Layers

The primary building blocks of CNNs are convolutional layers. These layers are comprised of a stack of convolutional filters that extract features from the input by performing an image convolution. The filter parameters are trainable weights that are tuned during network training such that the objective function is optimised. An image convolution involves sliding the convolutional filter over the input. At each point in the sliding window process, an

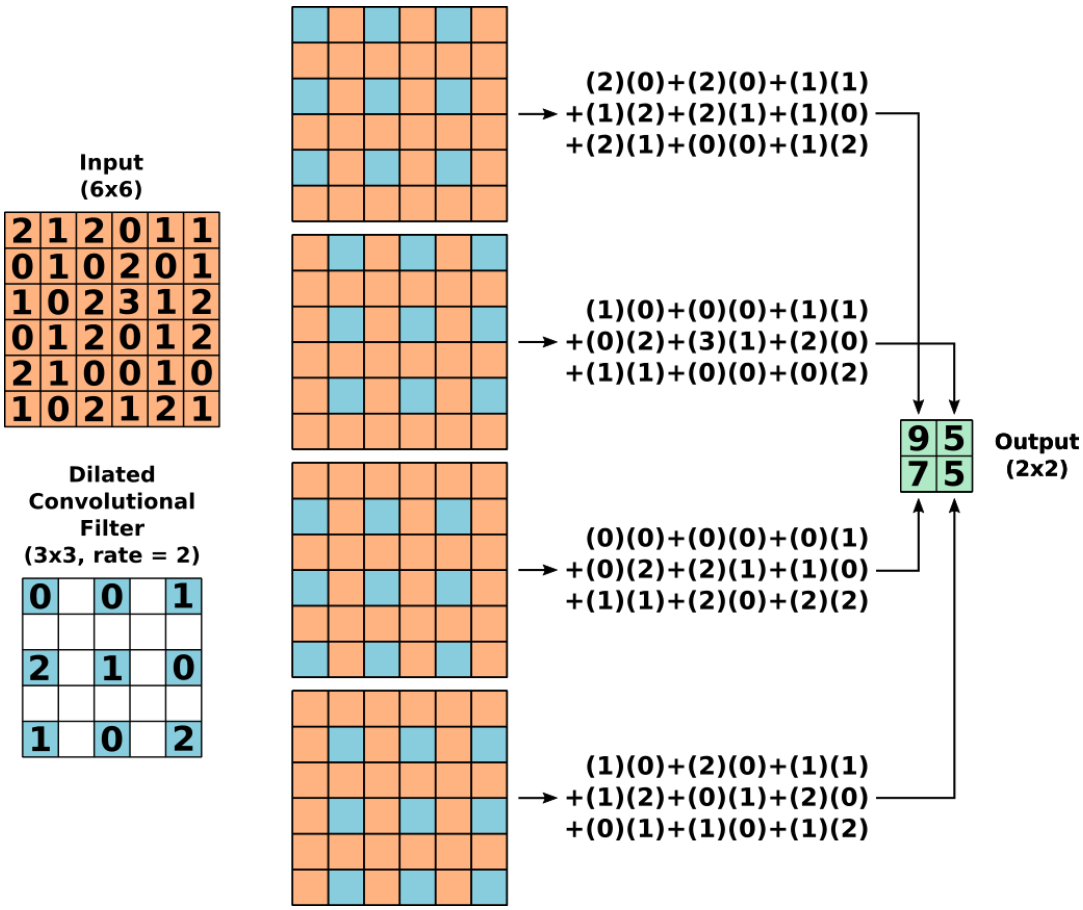


FIGURE 3.11: Image convolution with a dilated convolutional filter.

output value is computed by multiplying the filter element-wise with the underlying section of the input and summing the element-wise products.

A simple 2-dimensional image convolution is shown in Figure 3.9, with a 2×2 filter on a 3×3 image. The stride of a convolution is the amount the kernel is shifted after each computation. In Figure 3.9, the stride is one in both the horizontal and vertical directions. It is important to note that the formal definition of a convolution involves flipping the kernel. For a one-dimensional kernel, this amounts to reversing the order of the kernel elements. However, the examples presented in this section do not employ kernel flipping, which is common practice in computer vision problems. Further, padding is not applied to the inputs in these examples, meaning the output heights and widths are reduced compared to the input. In practice, this can be avoided by zero padding the input dimensions by the appropriate amount.

In CNNs, convolutional layers generally perform higher-dimensional convolutions than the 2-dimensional example in Figure 3.9. The dimensions of

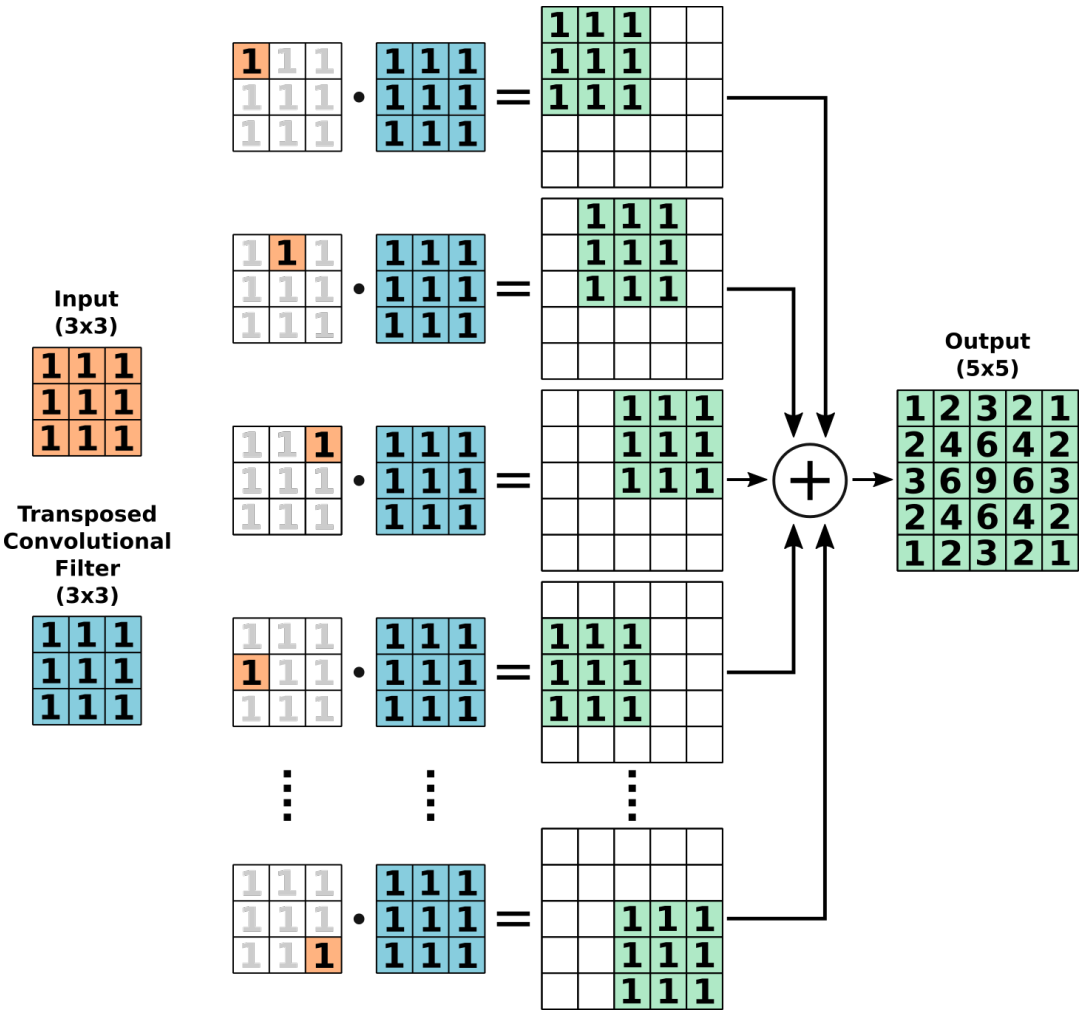


FIGURE 3.12: Transposed convolution with input and filter elements equal to values of one.

a convolutional layer can be expressed as $h \times w \times c_{in} \times c_{out}$, where h is the height of the filter, w is the width of the filter, c_{in} is the number of input channels and c_{out} is the number of output channels. A layer is made up of c_{out} convolutional filters, each of which have the dimensions $h \times w \times c_{in}$. This is illustrated in Figure 3.10. In this example, the input dimensions are $11 \times 11 \times 3$ and the output dimensions are $7 \times 7 \times 4$. This is achieved by stacking four convolutional filters of size $5 \times 5 \times 3$, making the dimensions of the layer $5 \times 5 \times 3 \times 4$.

It is generally desirable for convolutions to have a large receptive field in order to extract semantically rich features. In conventional network architectures with small convolutional filters sizes, such as 3×3, the receptive field is increased by downsampling feature maps using pooling layers (see Section 3.2.2.2). However, pooling operations result in a significantly reduced output resolution, compared to the input resolution. For problems such as semantic

segmentation, this loss of spatial information is undesirable. Dilated convolutions allow the receptive field of the convolution to be increased without increasing the filter size and without the need for pooling layers. In a dilated convolutional filter, the weights are sparsely arranged, rather than densely packed, allowing the same number of parameters to operate over a larger input region. For example, the dilated convolutional filter in Figure 3.11 is 5×5 in size, but with 16 regularly space holes, resulting in the same number of parameters as a densely packed 3×3 filter. Dilated convolutions are also referred to as atrous convolutions.

A transposed convolutional layer, also known as a deconvolutional layer, performs the opposite of a convolutional layer in terms of restoring the resolution. However, a transposed convolution is not the inverse of a convolution and does not undo a convolutional operation. These layers are a method of upsampling and are often found in the decoder section of an encoder-decoder network, such as that shown in Figure 3.8B. The operation performed performed by a transposed convolutional layer is shown in Figure 3.12, with a simple 2-dimensional example. The highlighted input value is multiplied by each of the elements in the filter and the results are place into the appropriate window of the output tensor. This is repeated for all input elements. Overlapping elements in the output tensor are summed. For simplicity in this example, all elements in the input tensor and the filter are set to values of one. This shows how the transposed convolutional filter distributes an input element across the output space.

3.2.2.2 Sampling Layers

A sub-category of sampling layers are those that perform downsampling operations. These are generally referred to as pooling layers. Reducing the resolution of feature maps can be desirable for several reasons. Firstly, successively extracting features from high resolution inputs is expensive in terms of computation and memory requirements. Secondly, downsampling the full image resolution introduces a degree of spatial invariance into the network, as neighbouring pixel locations are pooled together. Finally, pooling layers increase the receptive field of subsequent convolutional layers, allowing the extraction of more context-aware features. However, as discussed in Section

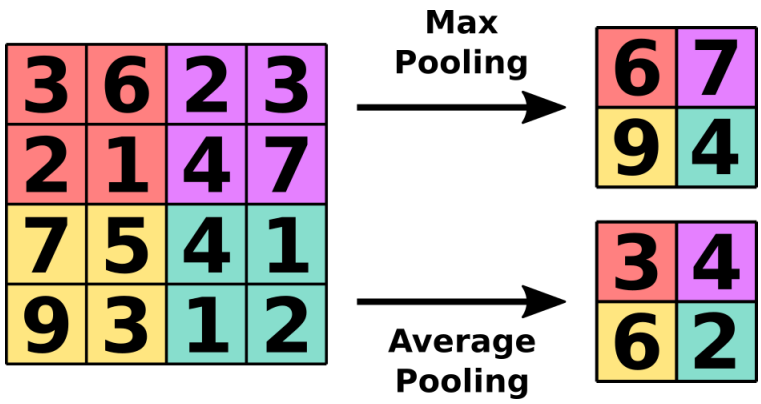


FIGURE 3.13: Max-pooling and average-pooling layers.

3.2.2.1, reducing the resolution can be undesirable for problems such as semantic segmentation, where spatial information is important. An alternative approach to downsampling is convolutional layers with a large stride.

The two most common types of pooling layers are max-pooling and average-pooling, both of which are shown in Figure 3.13. A max-pool operation takes the maximum value within the pooling window as output, while an average-pool operation computes the average of all values within the window. It is common for the stride of the pooling kernel to be set such that there is never an overlap of windows, as shown in Figure 3.13.

Upsampling layers are used in decoder networks to recover the spatial resolution lost during the encoder stage of the network. Simple upsampling methods include nearest neighbour interpolation, bi-linear interpolation and bi-cubic interpolation. Transposed convolutional layers, discussed in Section 3.2.2.1, are also a method of upsampling. A commonly used upsampling layer is the unpooling layer. As the name suggests, unpooling layers attempt to undo the pooling operations from the encoder network. Three approaches to unpooling are shown in Figure 3.14. The most simple upooling technique is to place the input value in the top left corner of the upsampled window, filling the remaining elements with zeros (Figure 3.14A). A second approach involves filling each element of the upsampled window with the input value (Figure 3.14B). A third approach, proposed by Zeiler and Fergus [64], stores the locations of the maximum values from the pooling layers and replaces the values into these locations in the unpooling layers (Figure 3.14C).

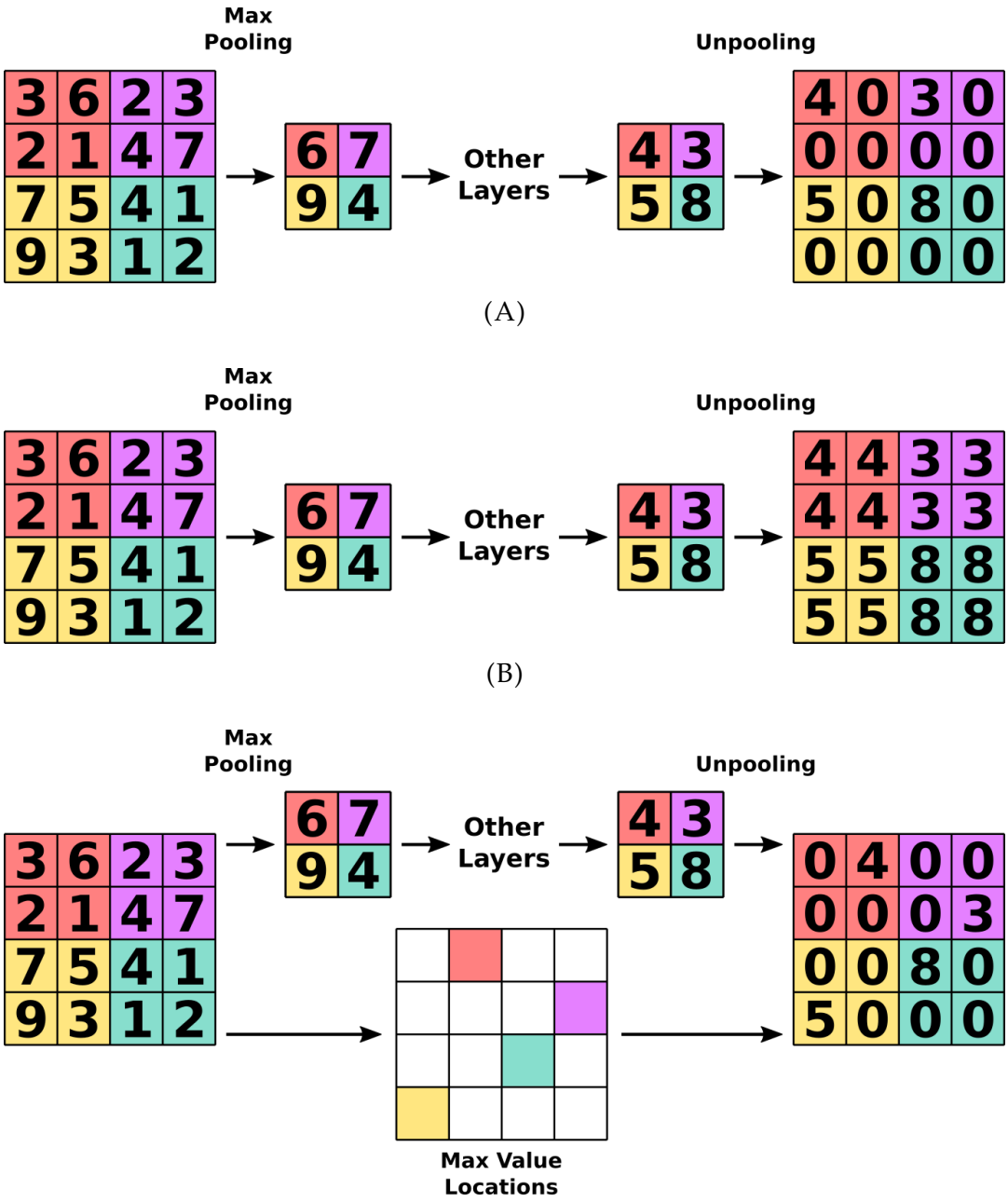


FIGURE 3.14: Three different approaches to unpooling layers.

3.2.2.3 Fully Connected Layers

In CNN architectures, fully connected layers can be used as feature extraction layers and also in classifiers to create a class activation space. The parameters of a fully connected layer are trainable and are learned end-to-end with the other trainable network weights. As the name suggests, fully connected layers connected every input element to every output element, via a trainable

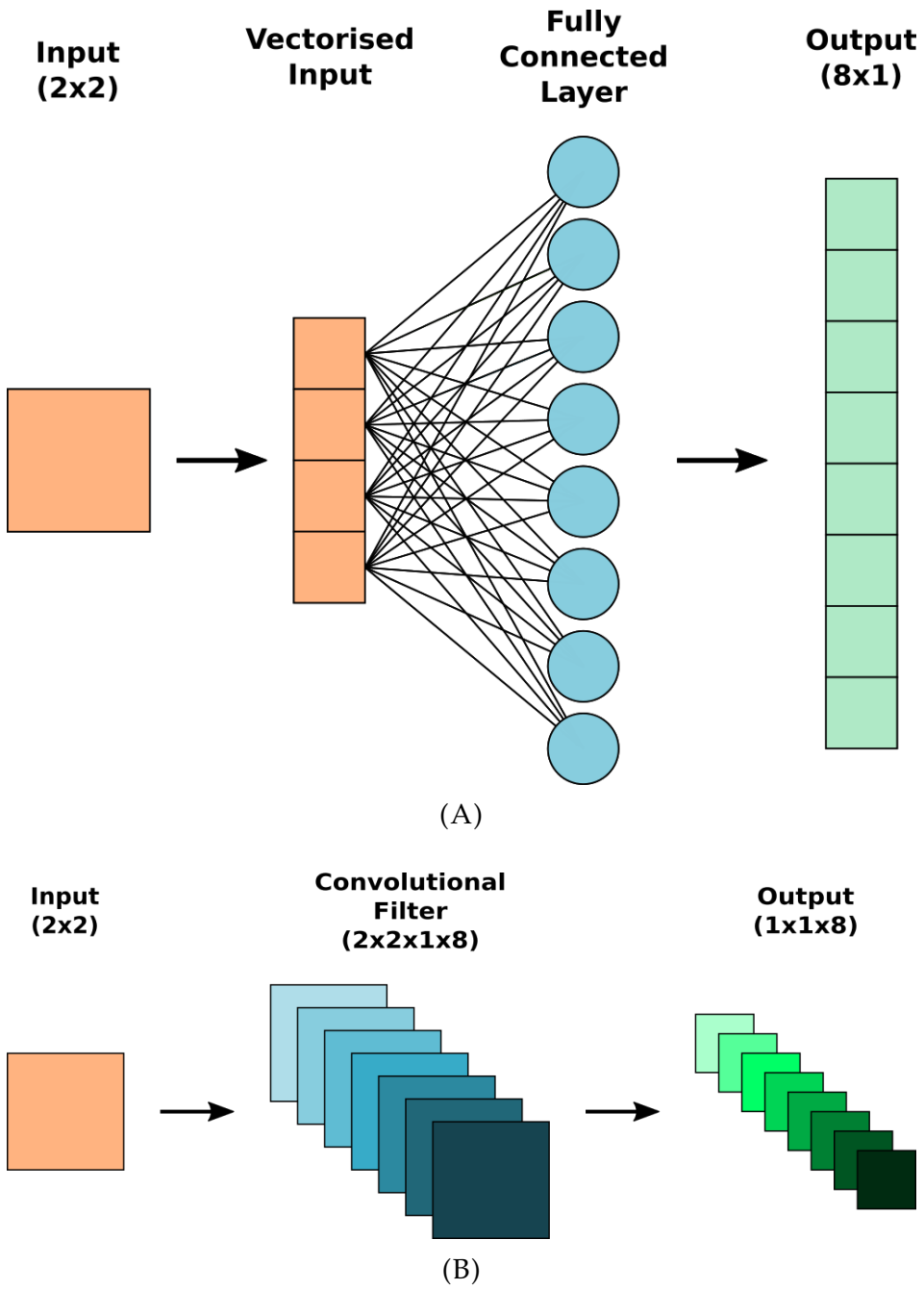
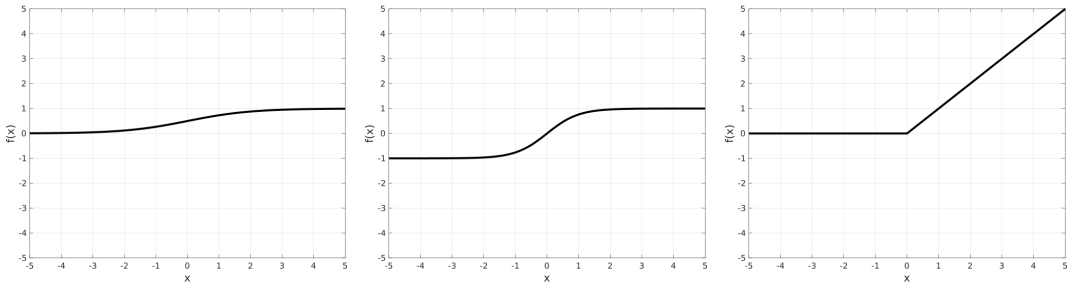


FIGURE 3.15: Top: fully convolutional layer with an 8-dimensional feature vector output. Bottom: the same fully connected layer formulated as a convolutional layer.

weight. These layers output a c_{out}-dimensional feature vector. This arrangement allows the fully connected layer to extract a feature that describes the image in a global sense, since the fully connected layer has a field of view that encompasses the entire input space. This makes such layers suitable for image classification problems, but unsuitable for segmentation problems, which require image structure to be maintained. The number of weights in a



(A) Sigmoid. (B) Hyperbolic tangent. (C) ReLU.

FIGURE 3.16: Example activation functions.

fully connected layer is equal to $H \cdot W \cdot c_{in} \cdot c_{out}$, where $H \times W \times c_{in}$ are the dimensions of the input and c_{out} is the number of output channels. Figure 3.15A shows a fully connected layer with eight output channels.

A fully connected layer can also be constructed as a convolutional layer. For an input of size $H \times W$, a convolutional layer with a filter size of $H \times W \times 1 \times c_{out}$ is equivalent to flattening the input and passing it into a fully connected layer with c_{out} output channels. This is shown in Figure 3.15B.

3.2.2.4 Activation Layers

Convolutions and the dense multiplications performed by fully connected layers are linear operations. The problems that CNNs are applied to are not linearly solvable and require a non-linear mapping from the input space to the desired output space. Activation layers provide this non-linearity by applying element-wise non-linear functions to the outputs of convolutional and fully connected layers. Commonly used activation functions are described graphically in Figure 3.16 and mathematically below.

- Sigmoid (Logistic) Activation Function.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.4}$$

- Hyperbolic Tangent (tanh) Activation Function.

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.5}$$

- Rectified Linear Unit (ReLU) Activation Function.

$$f(x) = \max(0, x) \tag{3.6}$$

ReLUs are the most commonly used activation functions in deep CNNs, generally following convolutional and fully connected layers. The function is efficient to compute, as is the derivative, which is equal to one for positive inputs and zero otherwise. ReLUs are useful in reducing the effect of the vanishing gradient problem, compared to sigmoid and hyperbolic tangent activation functions. This is discussed in detail in Section 3.2.5.7.

The sigmoid function returns a value between zero and one, which makes it useful for models that predict probabilities. The softmax function is a generalisation of the sigmoid function for multi-class scenarios. All output channels are squeezed such that the values are between zero and one, and are normalised by the sum of all channels. This results in the channels summing to one, as required to predict the probability distribution. Softmax functions are commonly used in classification networks, following a fully connected layer that outputs a vector with the number of channels equal to the number of classes. This vector can be referred to a class activation vector, since each dimension corresponds to the likelihood of the input belonging to a particular class. The softmax function scales and normalises these activations in order to predict the probability distribution over class labels. The formula for the softmax function is shown in Equation 3.7, where L is the number of classes.

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{L} e^{x_j}} \quad \text{for } i = 1, \ldots, L \tag{3.7}$$

3.2.2.5 Normalisation Layers

It is often useful to normalise features at various stages of the network. Features extracted by fully connected layers are sometimes normalised using ℓ_2 normalisation, such that all features have the same scale. Other types of normalisation include local contrast normalisation (LCN) and local response normalisation (LRN), both of which perform local normalisations to incite competition between nearby neuron outputs. However, both LCN and LRN have fallen out of favour in more recent network architectures.

Batch normalisation layers [97] are commonly used in current CNN architectures. During network training, examples are passed through the network in mini-batches (see Section 3.2.5.3). Batch normalisation is the process of normalising features within a batch to have a mean of zero and a standard

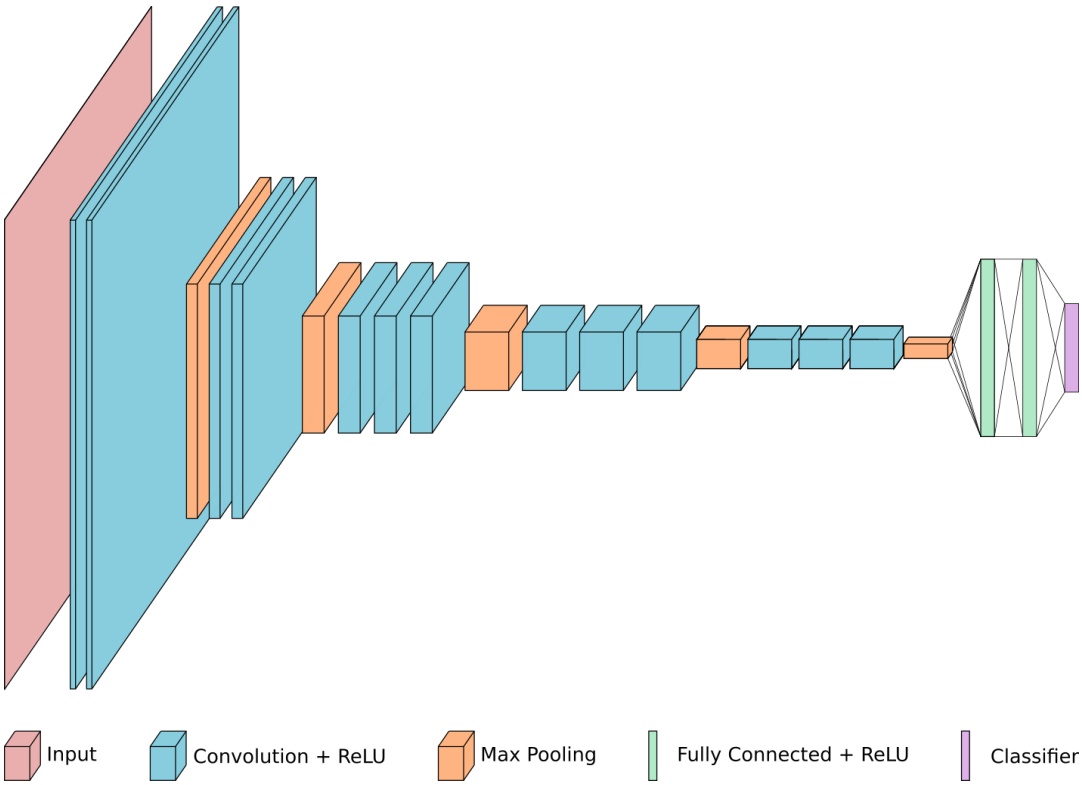


FIGURE 3.17: The VGG16 network architecture [65].

deviation of one. This has the advantage of improving the stability of the network during training and reducing the time taken for the model to reach convergence. Additionally, the inclusion of batch normalisation layers introduces more regularisation into the network. Regularisation is discussed in detail in Section 3.2.5.5. At test time, fixed global training set normalisation parameters are used. These parameters are estimated during training. Batch normalisation layers are generally located after convolutional layers.

3.2.3 Network Architecture

An example network architecture is shown in Figure 3.17. The network is the commonly used VGG16 architecture [65], consisting of 13 convolutional layers and 2 fully connected layers in the feature extraction section of the network. The classifier consists of an additional fully connected layer, to output a class activation vector, and a softmax layer to predict a probability distribution over classes. Each of the convolutional layers, as well as the two fully connected layers in the feature extraction section, are followed by a ReLU activation layer. Pooling layers follow banks of two or three consecutive convolutional and ReLU layers. Compared to some recent network architectures,

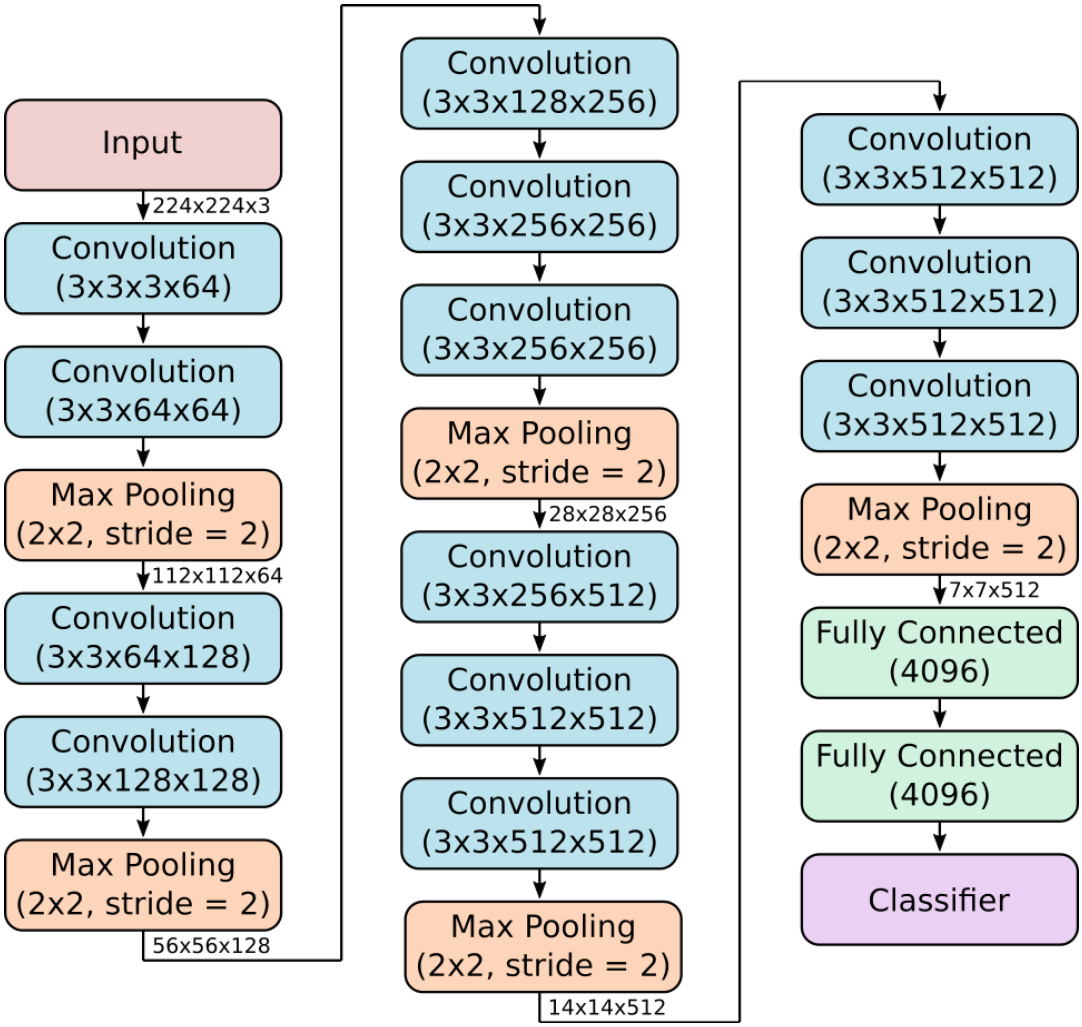


FIGURE 3.18: The VGG16 network architecture [65] in terms of convolutional kernel sizes, pooling window sizes and fully connected layer output sizes. Output feature map dimensions are shown next to the arrows.

the VGG architecture is relatively simple, with no batch normalisation layers or skip connections, like in ResNet [69] or DenseNet [71].

Figure 3.18 shows the VGG16 architecture in terms of the convolutional kernel sizes, pooling window sizes and fully connected layer output sizes. The dimensions shown next to arrows at the layer outputs are the dimensions of the feature maps at that location in the network. All convolutional kernels have a height and width of three and all pooling layers half the resolution in terms of both height and width. The two feature extraction fully connected layers output a 4096-dimension feature vector.

3.2.4 Data Pre-Processing

Before image data is fed into a convolutional neural network, pre-processing steps are usually carried out. Many CNN architectures, particularly those for image classification problems, require that the input is of a specific size. This can be seen in the VGG16 architecture in Figure 3.18, which expects input images to be $224 \times 224 \times 3$. The three channels correspond to the red, green and blue channels of a colour image. If an image is grayscale with a single channel, it can be converted to RGB by replicating the channel. Possible methods of adjusting the image dimensions include rescaling the image to the correct size, which may not preserve the aspect ratio, or resizing the image such that the smallest side is of the correct size and then taking a centre crop.

Image data is usually processed such that the mean across the training set is zero centred. Two methods of zero centring the data are commonly employed. The first involves finding the mean channel values of the training set. In other words, the mean red, green and blue values are found across the training data. Pixel intensities are then shifted by the appropriate channel mean. A second approach involves finding a mean image across the training set, that is, the RGB mean for each pixel location. The mean image is then subtracted from the input data.

It can also be useful to further normalise the input data to be at a uniform scale. This is generally carried out by dividing the input data by the training set standard deviation. Further data pre-processing is often carried out during training in order to artificially expand the size of the training set. This is discussed in Section 3.2.4.

3.2.5 Training Convolutional Neural Networks

In this section, the details of learning the convolutional neural network weights are discussed. The training of the model is treated as an iterative optimisation problem. A simple method of updating the network weights is via the gradient descent algorithm, which is described in detail in Section 3.1.1.

3.2.5.1 Initialisation and Pre-Training

Before training the network on data from the target domain, the weights of the network must first be initialised. Two methods of network initialisation are commonly employed: random initialisation and initialisation with pre-trained weights. Methods of random initialisation include Gaussian, Xavier [196] and He [197]. Random initialisation is desirable over uniform initialisation, as it ensures each weight receives a different signal and gradient, avoiding a bias toward any particular local solution. With random network weights, it is expected that a softmax classifier would have a classification accuracy comparable to random classification.

Training network weights from scratch is costly in terms of time and the amount of training data required. Often, target domains have limited labelled training data available. As such, random initialisation is undesirable. It is common for network weights to be initialised from a model that has been pre-trained on some large dataset, such as ImageNet [61]. The rationale is that the learned weights from such a pre-trained model will generalise to new target domains well enough to be a useful starting point. The weights are then fine-tuned on the target training set. Initialisation from pre-trained weights requires that the network architectures of the pre-trained model and the target model match. In cases where the match isn't perfect, partial initialisation with pre-trained weights can be employed, where only the weights from layers that have matching shapes are copied. It is often the case that the early convolutional layers will match, but modifications have been made to the deeper layers of the target network. The modified layers are initialised with random weights. Prior to the availability of large datasets for pre-training, unsupervised pre-training using auto-encoders was often employed [198].

3.2.5.2 Loss Functions

The training of CNN weights is driven by a loss function, which defines the objective of the training algorithm. Loss functions can also be referred to as objective functions, cost functions or error functions. The value returned by a loss function is a measure of how well the machine learning model is performing on the given data. This measure is obtained by comparing the prediction made by the model to a target ground truth value, in some manner. The network weights can be updated such that the loss is minimised using

a variant of the gradient descent algorithm (see Section 3.1.1). For complex models and objectives, loss functions can often only be optimised to local minimum.

An example loss function used for classification problems is the negative log loss, or cross-entropy loss. A model with parameters $\boldsymbol{\theta}$ that predicts a label $\hat{y}_i$ for an input x_i has the following cross-entropy loss:

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{m} \log\left(\Pr(\hat{y}_i = y_i | x_i)\right), \tag{3.8}$$

where y_i is the true label of x_i and m is the number of examples. For a CNN with a softmax classifier (see Section 3.2.2.4), this is equal to the negative logarithm of the normalised softmax channel pertaining to the true class label.

For regression problems, such as predicting depth, a common cost function is the ℓ_2 loss between the predicted and target values:

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} (\hat{y}_i - y_i)^2, \tag{3.9}$$

where $\hat{\mathbf{y}} = [\hat{y}_1, \ldots, \hat{y}_n]$ a vector of predicted values and $\mathbf{y} = [y_1, \ldots, y_n]$ is a vector of target values.

3.2.5.3 Training Batches, Iterations and Epochs

Training data is passed through the convolutional neural network during training in mini-batches, or simply, batches. A batch is a small subset of the training examples that are grouped together. A training *iteration* consists of a forward and backward pass of a batch through the network, where a forward pass involves propagating data from the input to the output, while a backward pass involves propagating the gradients of the error backwards from the output (see Section 3.2.5.4). A training *epoch* is a full pass over the training set. The number of iterations in an epoch is equal to the number of training examples divided by the batch size. The loss is computed across the entire batch and the weights are generally updated each training iteration. Batch selection usually involves shuffling the training data and sampling without replacement until all training examples have been selected. The data is then shuffled again and the process repeats over the next epoch. In general, the batch size is selected to be the largest that will fit in the GPU memory.

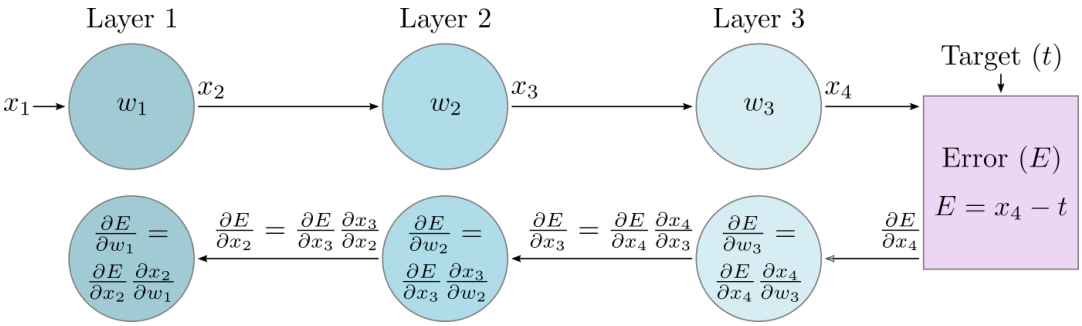
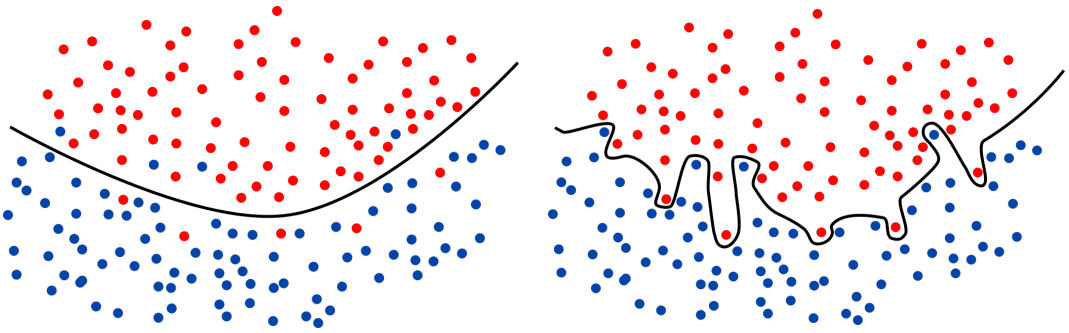


FIGURE 3.19: Overview of the back propagation of errors approach to training neural networks. This example consists of three simple layers that each contain a single trainable weight (w_1, w_2 and w_3). The inputs and outputs of layers are named x_1 through x_4 and the target value is t.

3.2.5.4 Back Propagation of Errors

The weights of the network can be updated using an iterative optimisation method, such as gradient descent (see Section 3.1.1). Such optimisation algorithms require calculation of the partial derivatives of the loss function, or error, with respect to each trainable parameter. This is achieved in convolutional neural networks by back propagation of errors, or simply, back propagation. At each training iteration, a batch is forward propagated through the network, with the objective function producing an error term. Error gradients are back propagated layer-by-layer through the network. At each layer, the derivative of the error with respect to trainable parameters in the layer, as well as with respect to the input to that layer, can be computed by applying the chain rule of differentiation. The derivatives with respect to the layer weights are used to update those weights, while the derivatives with respect to the inputs are back propagated to the previous layer.

Back propagation is illustrated in Figure 3.19 with a simple three layer example. Each of the three layers have a single trainable weight, w_1, w_2 and w_3, and output a single value. Inputs and outputs to layers are denoted as x_1 through x_4. In this example, the error function is simply the difference between the output of the final layer and a target value t. The derivative of the error with respect to the output of layer 3 is computed and passed to that layer. The chain rule is used to find the derivative with respect to weight w_3 and with respect to the input of the layer, x_3. The latter is back propagated to the previous layer and this process repeats.



(A) Good fit. (B) Over-fit.

FIGURE 3.20: Over-fitting demonstrated on a two class classification problem. Training examples are represented as points in the space, where the colour denotes the class label. The model fits a decision boundary to the training data, shown as a black line.

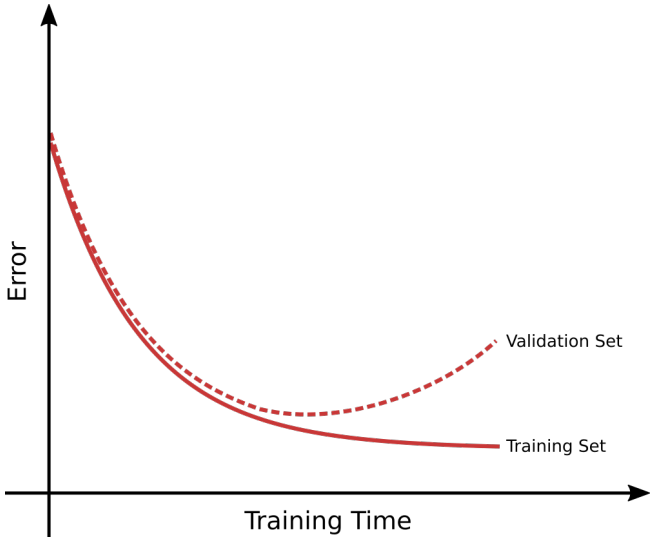


FIGURE 3.21: The validation loss increasing while the training loss continues to decrease is an indication that the model is over-fitting to the training data.

3.2.5.5 Over-Fitting

The problem of over-fitting refers to the scenario when a model learns the noise in the training data, that is, the model has "over-fit" to the training examples. This is illustrated with a two class example problem in Figure 3.20. A model that has over-fit will not generalise well to unseen examples and will likely have poor classification accuracy on a withheld set of data. Over-fitting can be observed during training if the error of a withheld set of examples, named the validation set, begins to increase, while the error of the

training set continues to decrease. This is shown in Figure 3.21

Techniques that aim to reduce the generational error of models by mitigating over-fitting are referred to as regularisation techniques. A simple approach to regularisation is the use of dropout [63]. During training, the outputs of neurons are randomly set to zero, or dropped out. This forces neurons to learn something that is useful without the reliance on specific previous activations and in turn, forces the network to extract more generalised features. Dropout can also be thought of as sampling a different network architecture at each training iteration.

Another approach to regularisation is weight decay [199], which is used to prevent the network weights becoming large too quickly. Each trainable weight in the network results in an additional term being included in the loss function, which incorporates a tunable hyperparameter that controls the dampening factor of the weight decay. The rationale behind constraining weights to be small is that simpler and sparse networks tend to generalise better than complex networks.

The size of the training set plays an important role in avoiding over-fitting. Smaller training sets are more susceptible to over-fitting since it is easier for the network to learn the noise in a small amount of data. Artificial expansion of the training set can be used to reduce over-fitting. This is discussed in Section 3.2.5.6. Batch normalisation, discussed in Section 3.2.2.5, also introduces some regularisation into the training process.

3.2.5.6 Data Augmentation

Training datasets can be artificially expanded by performing data augmentation on the training images. This involves processing training examples in some manner to create "new" training examples. Augmentation can be achieved by taking random crops of the image or by performing mirroring, usually only in the horizontal direction. Other approaches include scaling, adding noise to the image, adjusting the pixel brightness and perspective transformations. It is important to ensure that any data augmentation does not alter the semantics of the underlying image. For example, rotating an image of the digit "6" by 180 degrees results in an image that appears to be of the digit "9".

3.2.5.7 Vanishing Gradients

The problem of vanishing gradients can occur in deep neural networks when saturating activation functions, such as sigmoid and hyperbolic tangent (see Section 3.2.2.4), are used. The problem refers to gradient magnitudes exponentially decaying as they are back propagated through the network, resulting in slow training of early layers. The use of ReLU activation functions, which are linear for inputs greater than zero, largely mitigates this problem. However, vanishing gradients can still occur when training very deep networks. Residual networks with skip connections [69] address this problem by directly adding the input of a residual block to the output of the block. These connections skip the activation functions, resulting in larger gradients to back propagate.

3.3 Conditional Random Fields

Probabilistic graphical models, such as condition random fields (CRFs), allow the modelling of structural and contextual information to be incorporated into the inference process. In the context of this thesis, CRFs are considered for the problem of the semantic segmentation of images. The models work to find the labelling, or segmentation, that maximises a defined probability distribution, or minimises a defined energy definition. A conditional random field is defined by a set of random variables, or nodes, that are arranged in a graphical structure. Interactions and relationships between nodes are represented by edges in the graph. In the case of CRFs for semantic segmentation problems, edges are generally undirected. A node may be defined for every pixel in the image, or for larger image regions. Edges may be defined between nearby nodes only or between every possible pair of nodes in a fully connected dense CRF [182]. A simple pairwise CRF is depicted in Figure 3.22.

3.3.1 Energy Function

The energy that is to be minimised by the graphical model inference algorithm is described in this section. In the probability domain, this corresponds to finding the labelling that maximises a probability expression defined over the random field. A set of random variables $\{X_1, ..., X_N\}$ defines a random

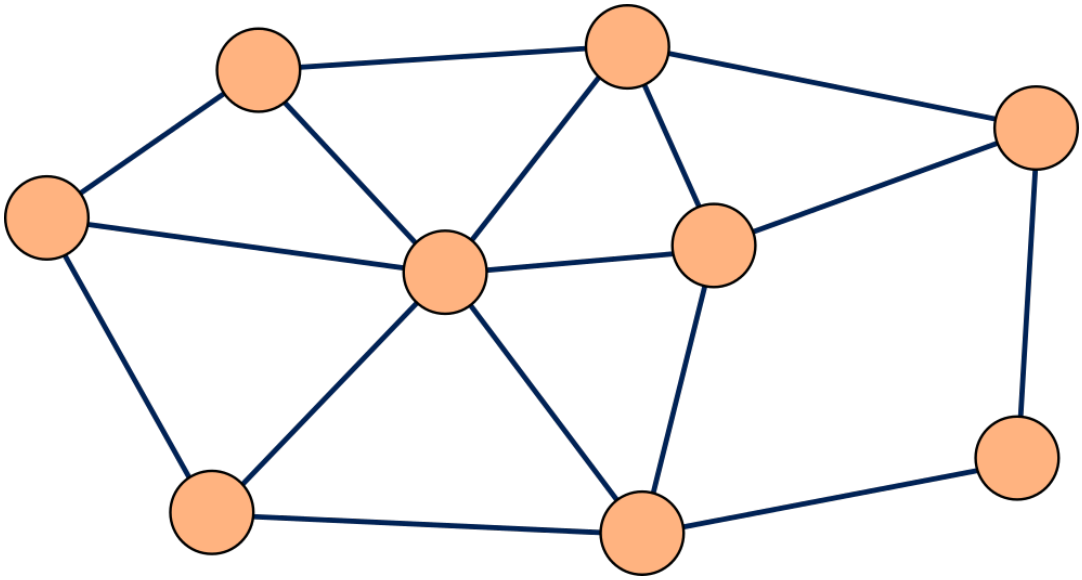


FIGURE 3.22: Example structure of a pairwise CRF.

field $\mathbf{X}$, where N is the number of nodes in the graph. Each random variable is defined over a set of class labels $\mathcal{L} = \{1, 2, ..., L\}$. An assignment $\mathbf{x}$ to $\mathbf{X}$, where all $x_i \in \mathcal{L}$, is a valid segmentation. The probability distribution for a general graph is seen in Equation 3.10.

$$\Pr(\mathbf{X} = \mathbf{x} \mid \mathbf{I}) = \frac{1}{Z} \prod_{\omega \in \Omega} \psi_\omega(\mathbf{X}_\omega \mid \mathbf{I}) \tag{3.10}$$

In the general case, Ω is a set of cliques within a graph that describe the interactions between nodes. Each clique has an associated potential ψ_ω. The partition function, Z, is the function that normalises the distribution. In the context of semantic segmentation, potentials can be defined for individual pixels (*unary terms*), pairs of pixels (*pairwise terms*) or regions of pixels (*higher-order terms*).

3.3.1.1 Pairwise CRF Energy

The most common CRF structure is a pairwise graph, which includes only unary and pairwise clique potentials. This is the structure that is considered for the remainder of this section. The pairwise graph has the probability distribution seen in Equation 3.11, where $u_i(.)$ are the unary terms, $f_{ij}(.,.)$ are the pairwise terms defined over connected pairs of nodes and $\mathcal{E}$ is the set of edges.

$$\Pr(\mathbf{X} = \mathbf{x} \mid \mathbf{I}) = \frac{1}{Z} \prod_{i=1}^{N} u_i(x_i) \prod_{i,j \in \mathcal{E}} f_{ij}(x_i, x_j) \tag{3.11}$$

It is generally more computationally efficient and numerically stable to operate in the log, or energy, domain. The distribution is reformulated in the log domain as:

$$\Pr(\mathbf{X} = \mathbf{x} \mid \mathbf{I}) = \frac{1}{Z} \exp\left(- \left(\sum_{i=1}^{N} U_i(x_i) + \sum_{i,j \in \mathcal{E}} F_{ij}(x_i, x_j) \right) \right), \tag{3.12}$$

where $U_i(.)$ and $F_{ij}(.,.)$ are the negative logarithms of the unary and pairwise terms, respectively. Maximising the distribution $\Pr(\mathbf{X} = \mathbf{x} \mid \mathbf{I})$ is equivalent to minimising the energy, which is defined in Equation 3.13 as $E(\mathbf{x} \mid \mathbf{I})$.

$$E(\mathbf{x} \mid \mathbf{I}) = \sum_{i=1}^{N} U_i(x_i) + \sum_{i,j \in \mathcal{E}} F_{ij}(x_i, x_j) \tag{3.13}$$

3.3.1.2 Unary Potentials

The potential at a given node without considering any interactions from connected nodes is referred to as a unary potential. The unaries can be treated as initial guesses of the likely labelling of each individual node. These initial distributions are generally computed by some other classifier. For example, a fully convolutional network [80] can be used to obtain a probability distribution over classes at every pixel location. The unary potentials can be defined as the logarithms of these distributions. The predictions made by convolutional neural networks are good initial guesses due to a CNN's ability to extract rich semantic features. However, the context incorporated into such predictions is limited to the receptive field of the network. Conditional random fields allow further contextual and structural information to be incorporated into the predictions by defining interactions between nodes in the graphical model.

3.3.1.3 Pairwise Potentials

The cost associated with assigning labels to a pair of nodes that are connected by an edge is referred to as a pairwise potential. These potentials can be

defined based on prior knowledge, such as *a car pixel is more likely to be near road pixels than water pixels*, or they can be learned via iterative optimisation techniques (see Section 3.1). Terms based on the contextual consistency of class labels are often referred to as label compatibility terms. A simple label compatibility function is the Potts model, which is equal to a value of zero for connected nodes that have the same label and a value of one for those with different labels.

In addition to label compatibility, information from the input data can be directly incorporated. For example, the dense CRF in [182] encourages nodes belonging to pixels that are nearby or have a similar appearance to take the same class label. This is achieved by applying Gaussian kernels to the pixel coordinates and RGB intensities. The sum of the Gaussian kernels is multiplied by a label compatibility function.

3.3.2 Energy Minimisation with Belief Propagation

Finding the labelling that minimises the energy described in Section 3.3.1.1 is complex, as the graph is undirected and contains loops. A common method used to find an approximate minimum solution is message passing, wherein nodes communicate information about their own belief to connected nodes. These belief messages propagate around the network until they converge, at which point the belief at each node can be calculated from the unary distribution and incoming messages. If message passing is suited to the problem, the resultant state of the nodes will be an exact minimisation of the energy for tree-like graphs and an approximate minimisation for graphs containing loops. Message passing in graphs with loops is referred to as loopy belief propagation [177].

3.3.2.1 Message Passing

Nodes compute messages based on their current belief and propagate the information to connected nodes, such that the receiver nodes can in turn update their belief states. The message a node propagates is computed based on the unary distribution at that node, the defined pairwise potentials and previously received messages. A caveat of this approach is that a node should not receive information that it has itself propagated. This is illustrated in

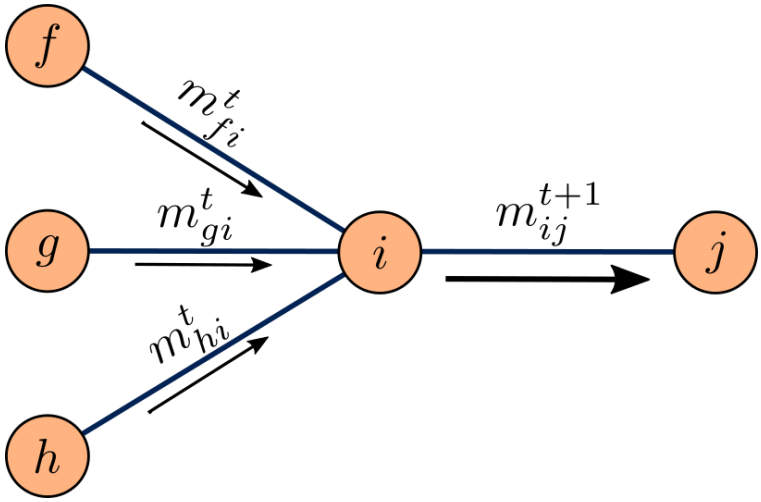


FIGURE 3.23: The message from node i to node j considers previous messages flowing into node i, except for the message propagated from node j.

Figure 3.23. When computing the message from node i to node j, the message includes information propagated to node i from all nodes except j. The message m^t_{ij} refers to the message passed from node i to node j at time t.

The message update formula is shown in Equation 3.14, where i indicates the sender node, j is the receiver node and t is the time step. The edges connecting node i to other nodes are contained in the set $\mathcal{E}_i$, and therefore $\mathcal{E}_i \backslash j$ represents edges to node i excluding the edge connected to node j. The unary distributions at node i and the pairwise potentials are represented by u_i and f respectively, while x_i and x_j represent class labels under consideration at each node.

$$m^{t+1}_{ij}(x_j) = \sum_{x_i} f(x_i, x_j)\, u_i(x_i) \prod_{k \in \mathcal{E}_i \backslash j} m^t_{ki}(x_i) \tag{3.14}$$

Equation 3.14 shows the single scalar message for a particular class label. The message for the entire distribution of class labels is simply a matrix-vector multiplication between the pairwise terms and the product of unaries and incoming messages. For numerical stability, messages are normalised such that $\sum_{x_j} m_{ij}(x_j) = 1$.

3.3.2.2 Belief Calculation

Messages propagate until convergence is reached, at which point the beliefs at nodes can be computed from the converged incoming messages and the node unary potentials. Equation 3.15 describes the belief calculation at a given node, where $b_i(x_i)$ is the belief of label x_i at node i. Once normalised, the belief becomes a probability distribution across the set of class labels at the given node.

$$b_i(x_i) = u_i(x_i) \prod_{k \in \mathcal{E}_i} m_{ki}(x_i) \tag{3.15}$$

In the energy domain, the product operation becomes a sum of energies, as seen in Equation 3.16, avoiding the multiplication of many small probabilities that may result in numerical instability.

$$b_i(x_i) = \exp\left(\ln\left(u_i(x_i)\right) + \sum_{k \in \mathcal{E}_i} \ln\left(m_{ki}(x_i)\right) \right) \tag{3.16}$$

Chapter 4

Deep Metric Learning and Image Classification

This chapter proposes a Gaussian kernel loss function and training algorithm for convolutional neural networks that can be directly applied to both metric learning and image classification problems. The method treats all training feature embeddings from a deep neural network as Gaussian kernel centres and computes the loss by summing the influence of a feature embedding's nearby centres in the feature embedding space. Fast approximate nearest neighbour search is leveraged, in the interest of providing a scalable solution. End-to-end learning of the network weights is made feasible by the introduction of periodic asynchronous updates of the Gaussian centres. Our approach results in a well formed feature embedding space, in which semantically related instances are likely to be located near one another, regardless of whether or not the network was trained on those classes. Our proposed method outperforms state-of-the-art deep metric learning approaches on embedding learning challenges, as well as conventional softmax classification on several datasets.

The problems of classification and metric learning are each important to the application of robotic vision. Classifying observed objects is an important precursor to robotic interaction with the environment, such as object manipulation [4] or safe navigation [2]. Metric learning allows a robot to measure the similarity between an observed example and other observations or known training examples. This is an important step in enabling a robot to reason about unknown observations.

4.1 Motivation

Metric learning is the problem of learning a distance function over inputs. In this work, the inputs are images and the distance function should be a measure of the semantic similarity of given inputs. Since distance measures between examples in the image space are largely meaningless in terms of high-level semantic information, metric learning approaches aim to learn a transformation from the image space to a semantically meaningful feature embedding space. This transformation should be learned such that a standard distance measure, such as the Euclidean distance, between examples in the feature embedding space can be used as a measure of the semantic similarity. In other words, feature embeddings extracted from semantically similar images will be located nearby, while those extracted from semantically dissimilar images will be located further apart. A well learned distance metric should transfer to out-of-distribution examples, that is, examples belonging to semantic classes that are not represented in the training set. This provides a means for a system, such as a robot, to reason about novel observations. Other applications of metric learning include retrieval, zero-shot and few-shot learning, clustering and weakly supervised or self-supervised learning.

Image classification is the task of categorising an image into one out of a set of classes. The classes must be previously known to the model and are generally those that are represented in the training set. Image classification is a fundamental problem in robotic vision, as the ability to categorise and recognise observations is often a necessary precursor to interaction with the environment. Examples of image classification include coarse object recognition [61, 15], fine-grained recognition [200, 201, 202, 203], face recognition [204] and scene recognition [205, 206].

The tasks of classification [60, 207, 66, 69] and metric learning [19, 21, 22] are generally treated as two separate problems. Although the feature embeddings learned by metric learning approaches often transfer well to novel classes, they are generally less amenable to classification. As such, metric learning models struggle to compete with, and may significantly underperform, conventional state-of-the-art classifiers for problems beyond zero-shot and few-shot classification [116]. Likewise, state-of-the-art classification approaches fail to learn feature embedding spaces that represent inter-class and intra-class similarities and variations to the standard of metric learning

approaches. This means that a model trained for a specific classification task may not be suitable for any other task. Outside of zero-shot and few-shot learning, some metric learning algorithms have been applied to classification [116, 20], although approaches that perform well in both domains remain uncommon.

Metric learning and classification should not need to be treated as separate problems. Both tasks require rich representations of the high-level semantic information contained within an input image. One would expect that feature embeddings from metric learning approaches that encode fine-grained information should also be amenable to classification tasks. This suggests that it should be possible to learn a single model that can be successfully applied to problems in both domains.

Such a model is particularly desirable for robotic applications. A robot operating in an unconstrained environment will almost certainly observe objects that are from outside of the training set distribution. The ability of the learned representations to transfer to out-of-distribution classes enables a robotic system to reason about novel observations, rather than silently failing. The ability to classify in-distribution examples allows a robot to understand and interact with the environment.

To this end, we propose a loss function and training algorithm for convolutional neural networks that can be applied to both metric learning and image classification problems. The approach defines training set feature embeddings as Gaussian kernel centres, which are used to push or pull features in a local neighbourhood, depending on the labels of the associated training examples. Fast approximate nearest neighbour search is used to provide an efficient and scalable solution. Our proposed approach differs from kernel or radial basis function neurons [208, 209], as the kernel centres are not learned network parameters, but are defined to be the locations of the training set features in the embedding space. Additionally, kernels are used only in the loss function and classifier, not as activation functions throughout the network. The proposed approach is related to NCA [210, 211], but introduces per exemplar weights, makes training feasible through the introduction of periodic asynchronous updates of the kernel centres and is made scalable for a large number of training examples and a high feature embedding dimension. Additionally, the importance of the feature embedding space dimensionality is explored. Our approach is motivated in terms of metric learning in Section 4.1.1 and in terms of classification in Section 4.1.2.

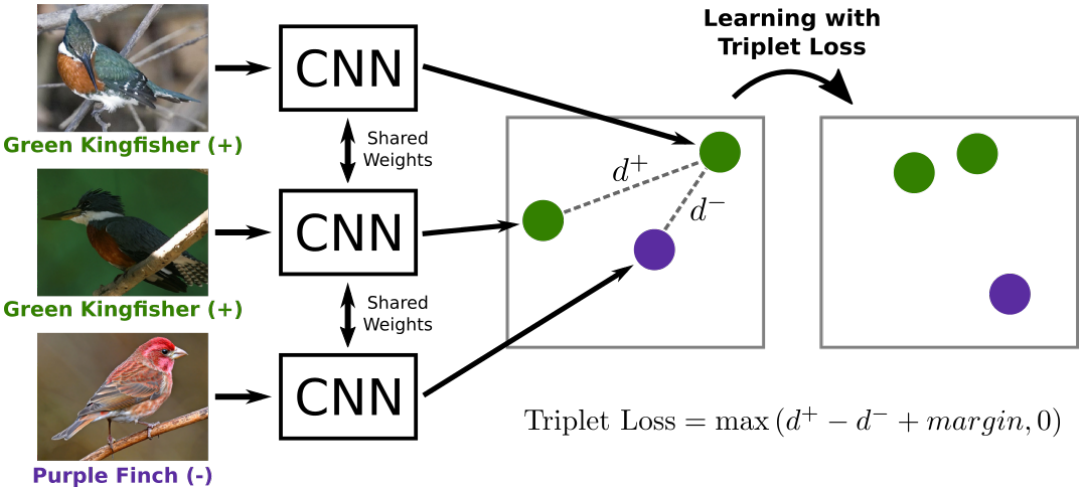


FIGURE 4.1: Training a triplet network. The positive pair distance should be smaller than the negative pair distance, by some margin.

4.1.1 Metric Learning Motivation

The best success on feature embedding learning tasks has been achieved by deep metric learning methods [17, 18, 19, 20, 21, 22, 23, 21], which use deep neural networks to learn the transformation between the image space and the feature embedding space. The majority of these approaches use or in some way generalise a triplet architecture with hinge loss [109]. At a given training iteration, a triplet network takes a trio of inputs: two images of the same class, one of which is called the anchor, and one image of a different class. The training algorithm tries to make the distance between the anchor and the positive example smaller than the distance between the anchor and the negative example, by some margin. This is illustrated in Figure 4.1.

Triplet networks often demand that a single cluster is formed per class. The reason for this is that during training the algorithm may indiscriminately select any semantically similar examples to be pulled together. For example, if the two feature embeddings sampled as the positive pair are located in separate, well formed local clusters of similar examples, the algorithm will still attempt to pull those examples together. In other words, these approaches consider only the global structure of the feature embedding space and not the local structure of the space. Triplet loss can be defined for each example over only a selected subset of target neighbours, however, this may restrict the possible feature space that can be learned by the model.

A single cluster per class is not necessarily the most semantically rich representation and can be limiting in terms of representing inter-class and intra-class variations. Considering, for example, the problem of learning a distance metric over images of different species of birds. One region of the space may contain species-based clusters of birds in flight, while another region may contain clusters of the same species, but on the ground. Allowing multiple clusters to form per class may enable a richer representation of such fine-grained information.

The proposed Gaussian kernel metric learning approach allows multiple clusters to form per class, if that is appropriate. The local structure of the feature embedding space is considered, avoiding the indiscriminate pulling together of examples in well-structured local regions. This allows the proposed method to better represent fine-grained variations that may be present in the data, compared to existing state-of-the-art deep metric learning approaches. Further, experimental results show that the feature embeddings learned by our approach are also amenable to classification, outperforming conventional classification approaches on several datasets.

4.1.2 Classification Motivation

The most common approach to image classification is a convolutional neural network trained with a softmax classifier, which transforms activations into a distribution across class labels [60, 207, 66, 69]. As shown in Figure 4.2, these networks include a fully connected layer that outputs a vector with the number of channels equal to the number of classes. After passing the vector through the softmax layer, cross-entropy loss is used to train the network. This structure is inflexible as the number of classes is baked into the network. As such, new classes cannot be added without restructuring and retraining of the network. The structure is closed-set by design, meaning that the network will silently fail when presented with an example of a novel class by incorrectly categorising the novel example as one of the known classes.

Softmax-based networks are also limited in terms of the inter-class and intra-class variations that can be represented. The loss function and network structure forces classes to be axis-aligned. In other words, the training algorithm aims for the network to output a high value on the channel corresponding to the input's class, and a low value on all other channels. This restricts the

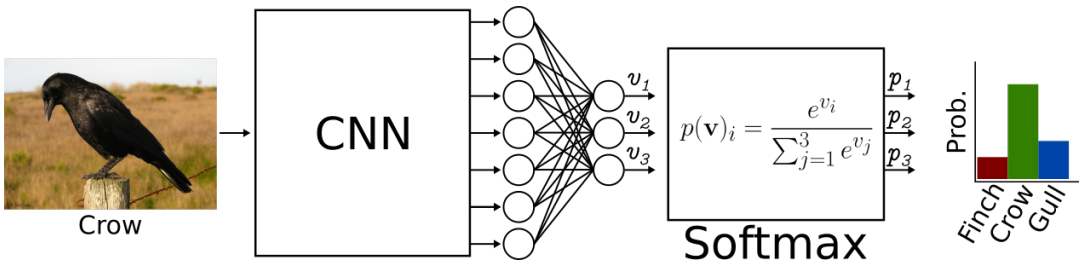
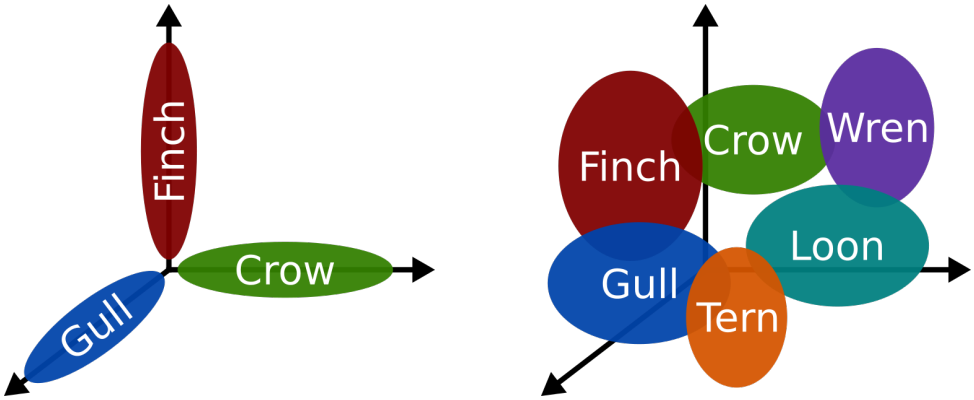


FIGURE 4.2: Classification with a softmax-based convolutional neural network.



(A) Softmax. (B) Metric learning.

FIGURE 4.3: Example arrangement of bird classes in a softmax-based approach (A) and a metric learning approach (B). Softmax approaches are inefficient, as classes must be axis-aligned. Metric learning approaches allow efficient representation of classes and become even more efficient as the feature embedding dimensionality becomes large. We embed features in 64 dimensions up to 4096 dimensions.

representational power of the network, as seen in Figure 4.3, leading to a potential loss of inter-class and intra-class fine-grained variations that are useful for transfer learning and open-set problems.

The proposed Gaussian kernel approach addresses these issues associated with conventional softmax-based networks. As our approach learns a transformation to a space in which distance is a measure of similarity, intra-class and inter-class similarities can be better represented by the model. The approach also allows for new classes to be added on-the-fly, with no updates to the network weights or changes to the network architecture required for the distance metric to remain valid. Importantly, the proposed approach not only learns feature embeddings that are amenable to classification, but also that transfer well to novel classes.

4.2 Contributions

This chapter presents a Gaussian kernel-based metric learning approach. The proposed approached learns feature embeddings that are amenable to both transfer learning and classification problems. The advantages of our proposed approach can be summarised as follows:

- The proposed training algorithm is made feasible by introducing periodic asynchronous updates of the kernel centres (Section 4.7).
- End-to-end learning can be made scalable by leveraging fast approximate nearest neighbour search (Section 4.6).
- Trainable per-kernel weights are introduced, allowing the influence of important training set feature embeddings to be amplified (Section 4.5.3).
- Our approach can be applied to what are usually treated as two separate problems: image classification and metric learning.
- For feature embedding learning problems, the proposed approach outperforms state-of-the-art deep metric learning algorithms on the Stanford Cars196 and CUB Birds200 2011 datasets (Section 4.10.1).
- The proposed approach outperforms a conventional softmax classifier on the CUB Birds200 2011, Stanford Cars196, Oxford 102 Flowers and Leafsnap datasets (Section 4.10.2).
- Further to learning better compact feature embeddings than triplet-based networks, the Gaussian kernel approach is able to take advantage of a larger feature embedding space (Section 4.10.1.4).
- Finally, the proposed approach is able to learn from limited training data, significantly outperforming softmax-based networks in few-shot learning problems (Section 4.10.2.4).

The majority of the research detailed in this chapter has been presented in a peer-reviewed publication [24]. Additional work not found in the publication includes the analysis in Sections 4.7.2, 4.8.2 and 4.9.2.3, as well as the experimental results in Sections 4.10.2.5 through 4.10.2.7.

4.3 Related Work

This section provides a brief summary of previous work that directly relates to the proposed approach, as well as methods that our approach is evaluated against in the experimental results in Section 4.10. A detailed review of metric learning and image classification literature can be found in Chapter 2.

4.3.1 Radial Basis Functions in Neural Networks

A Gaussian kernel is a type of radial basis function (RBF), which have previously been used to varying degrees in neural networks. Radial basis function networks were introduced by Broomhead and Lowe [208]. These networks formulate activation functions as RBFs, resulting in an output that is a sum of radial basis function values between the input and network parameters. In contrast to these radial basis function networks, our proposed Gaussian kernel approach uses RBFs in the classifier and loss function of a deep convolutional neural network and the radial basis function centres are defined to be the locations of the high-dimensional feature embeddings of training examples, rather than being network parameters. Radial basis functions have been used as neural network classifiers in the form of support vector machines. In one such formulation, a neural network is used as a fixed feature extractor and separate support vector machines are trained to classify the features [73, 74]. No joint training occurs between the classifier and network. Such an approach is often used for transfer learning, where the network is trained on vast amounts of data and the support vector machines are trained for problems in which labelled training data is scarce. The work proposed by Tang [212] replaces the typical softmax classifier with linear support vector machines. In this case, the classifier and network are trained jointly, meaning the loss that is minimised is margin based.

4.3.2 Metric Learning

Early methods in the domain of metric learning include those that use Siamese networks [101] and contrastive loss [102, 103]. The objective of these approaches is to pull pairwise examples of the same class together and push pairwise examples of different classes apart. Such methods work on absolute distances, while triplet networks with hinge loss [109] work on

relative distance. Metric learning with deep convolutional neural networks is known as deep metric learning. Several deep metric learning approaches make use of, or generalise, deep triplet neural networks [17, 106, 18, 19, 20, 21].

The selection of informative triplets that will result in useful updates to the network weights is an important problem for triplet networks. A useful triplet is one that is not currently well-learned by the network. For example, a triplet for which the distance between the anchor and the negative example is significantly smaller than the distance between the anchor and the positive example, would be considered a *hard* example. These hard triplets are much more informative than triplets that already satisfy the triplet objective, and therefore allow better updates to be made to the network weights. Selecting good triplets is an important line of research in metric learning, known as hard negative mining. Schroff *et al.* [18] perform semi-hard mining within a mini-batch, while [21] introduces a smart mining technique that mines for triplets over the entire training set.

Other approaches seek to achieve useful and efficient updates to the network weights by performing multiple comparisons for a single anchor image at a single training iteration. Song *et al.* [19] propose a lifted structured embedding with efficient computation of the full distance matrix within a mini-batch. This allows comparisons between all positive and negative pairs in the batch, resulting in more efficient updates at each training iteration. Similarly, Sohn [20] proposes an approach that allows multiple intra-batch distance comparisons, but optimises a generalisation of triplet loss, named N-pair loss, rather than a max-margin based objective, as in [19]. The global embedding structure is considered in [22] by directly minimising a global clustering metric. However, a combination of global and triplet loss is shown to be beneficial in the work by Kumar *et al.* [23].

4.4 Feature Embeddings

Feature embeddings are representations of input data in a space in which properties of the data can be more easily represented. These properties may be the semantic information contained within an input image. Practically, a feature embedding is a fixed-length and continuous vector of real numbers. In this section, the process of extracting feature embeddings by transforming

an input from the image space into a feature embedding space using convolutional neural networks is described. Visualisations of example feature embeddings are then presented and discussed.

4.4.1 Extracting Feature Embeddings

A deep convolutional neural network extracts a hierarchy of features from a given input image. The first layer in a network extracts features directly from the image pixel intensities. These features are passed through an activation function and the subsequent layer extracts features from these activations. This process continues through the network layers, with each layer transforming the features from the previous layer into a higher-level feature representation. In embedding space learning problems, generally only a single feature representation is explicitly considered. In general, the activations of the final layer of network are taken as the feature embeddings. All other feature representations learned by the network are treated as latent. Although no particular structure is demanded by embedding learning problems, feature embeddings are usually $d \times 1$ in shape, where d is the dimensionality of the feature embedding space.

The final layer of the network that produces the feature embeddings is referred to as the embedding layer. Most commonly this layer will be a fully connected or pooling layer. For example, a fully connected layer will output a d-dimensional feature embedding for a given image, where d is simply set using a layer parameter that controls the number of output channels produced. An example configuration with the commonly used VGG16 architecture [207] may involve taking the penultimate fully connected layer (FC7) as the embedding layer, producing a 4096-dimension feature embedding. Extracting a feature embedding from a given image using a VGG16 architecture is illustrated in Figure 4.4.

Formally, an input image $\mathbf{I} \in \mathcal{I}$ is transformed from the image space into a feature embedding space by a convolutional neural network. This transformation is expressed as:

$$\mathbf{x} = g(\mathbf{I}; \boldsymbol{\theta}), \tag{4.1}$$

where $\mathbf{x} = \left[x^{(1)}, \ldots, x^{(d)}\right]$ is a d-dimensional feature embedding for image $\mathbf{I}$, g is the transformation function and $\boldsymbol{\theta}$ denotes the parameters, or weights, of the neural network.

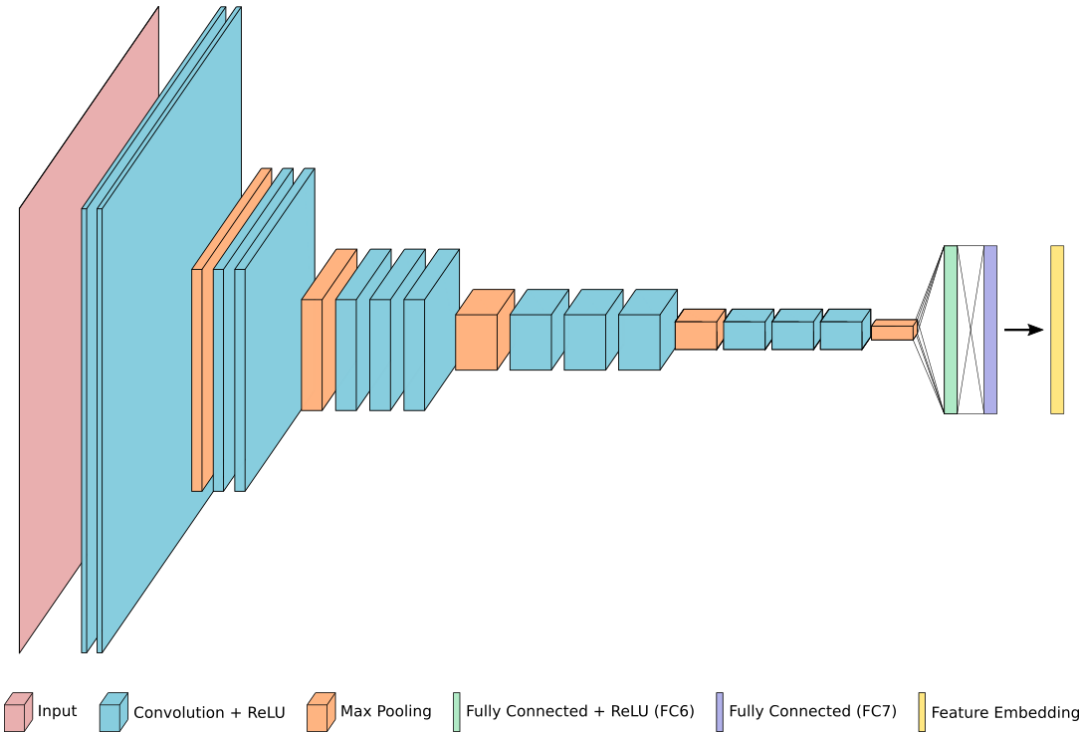


FIGURE 4.4: Extracting a feature embedding using a convolutional neural network with a VGG16 architecture [207]. The embedding layer is taken as FC7, without a subsequent activation layer. This set-up extracts a 4096-dimension feature embedding from an input image.

4.4.2 Visualising Feature Embeddings

In this work, the aim is for feature embeddings to represent the semantic information contained in input images. The feature embeddings of semantically similar images should be located nearby in the embedding space, while those of semantically dissimilar images should be located further apart. Figure 4.5 shows visualisations of example feature embeddings learned by the approach described in the remainder of this chapter. These visualisations illustrate how CNNs can be trained to transform images into a space that allows easier representation of complex, high-level semantic information.

In Figure 4.5, the feature embeddings belonging to different classes of car makes/models/years are visualised by projecting the vector elements to RGB values. It can be seen that examples belonging to the same semantic category have similar representations in the feature embedding space, despite the many input image variations that exist, such as viewpoint, pose and car colour. Further, examples belonging to different but related semantic categories have feature representations that are more similar than those

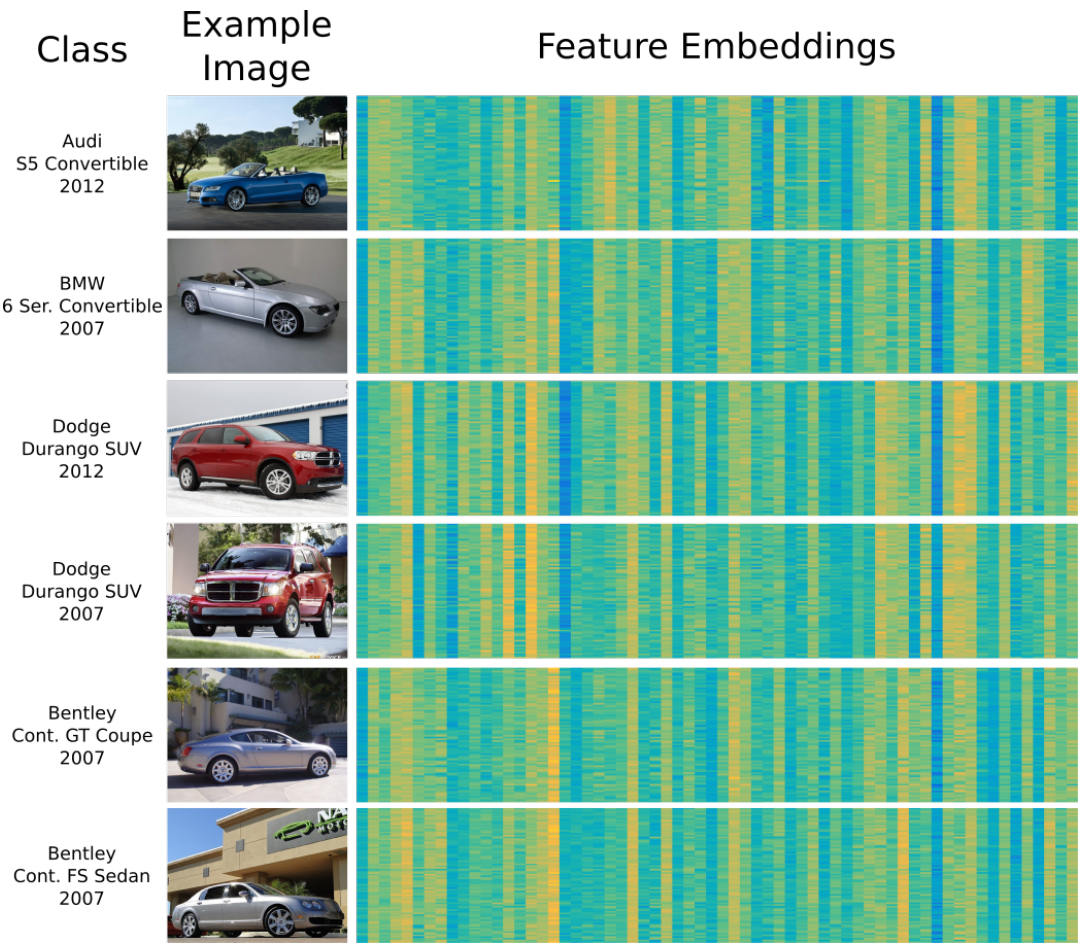


FIGURE 4.5: Visualisation of feature embeddings for different car makes/models/years. Rows of the visualisation correspond to examples and columns correspond to the dimensions of the feature embeddings. For each class, 80 examples each with 64-dimension feature embeddings are shown. Visualisations are obtained by converting each feature element to an RGB vector. It can be seen that the feature embeddings encode more information than simply the class label. Similar classes, such as different makes and models of convertibles, different years of the same model or different body types of the same model, result in similar feature embeddings.

belonging to less semantically related categories. This shows that convolutional neural networks not only have the ability to classify examples into semantic categories, but also allow measurement of the semantic similarity between examples from different categories. The similar semantic categories include two different makes and models of convertibles, two different years of the same car model and two different body types of the same car model.

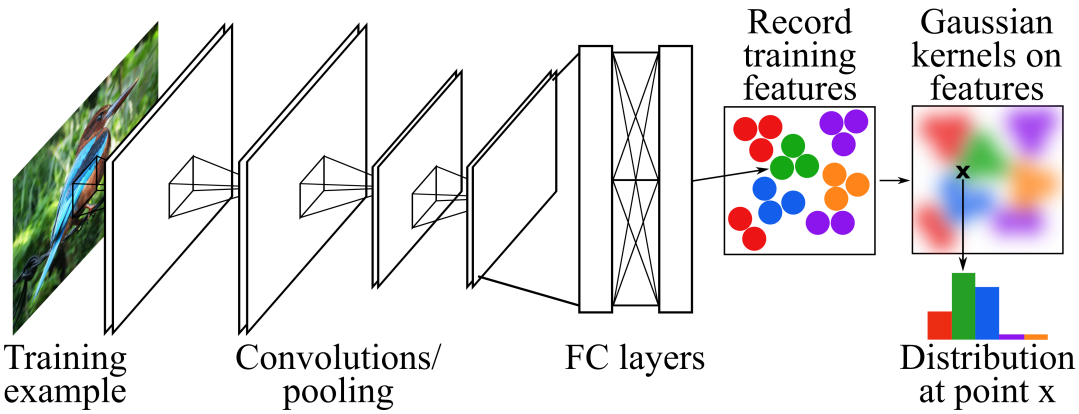


FIGURE 4.6: Overview of our approach. Note that the feature embeddings are high-dimensional.

4.5 Classifier and Objective Function

This section introduces the proposed approach that can be applied to both metric learning and classification problems. Gaussian kernels are described and a classifier based on these kernels is presented. The objective function used to the train the model is also discussed. An overview of the proposed approach seen in Figure 4.6.

4.5.1 Gaussian Kernels

A Gaussian kernel is a type of radial basis function. Given two points, $\mathbf{x}$ and $\mathbf{c}$, a radial basis function f returns a value that depends only on the distance between those points:

$$f(\mathbf{x}, \mathbf{c}) = f\left(\|\mathbf{x} - \mathbf{c}\|\right). \tag{4.2}$$

Although many radial basis functions exist, this work considers only Gaussian kernels, which return values based on the Gaussian function of the distance between the two points:

$$f(\mathbf{x}, \mathbf{c}) = \exp\left(\frac{-\|\mathbf{x} - \mathbf{c}\|^2}{2\sigma^2}\right), \tag{4.3}$$

where σ is the Gaussian standard deviation that sets the width of the Gaussian. The point $\mathbf{c}$ is referred to as the Gaussian centre. As such, the kernel can be thought of as returning a value based on how far some example point $\mathbf{x}$ is from the centre of the Gaussian curve $\mathbf{c}$. The value returned radiates out evenly from $\mathbf{c}$ in all directions.

4.5.2 Defining the Gaussian Kernel Centres

In this work, the feature embeddings of each training example are defined as Gaussian kernel centres. That is, the training set feature embeddings are stored and used in order train the network and classify new examples. Formally, a set of α Gaussian kernel centres is defined as $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_\alpha\}$, where $\mathbf{c}_i = \left[c_i^{(1)}, \ldots, c_i^{(d)}\right]$ is the d-dimensional feature embedding for the i-th training set example. A feature embedding for training example i is obtained by passing the corresponding training image through the convolutional neural network, transforming the example from the image space into the feature embedding space. This is shown in Equation 4.4, where $\mathbf{I}_i$ is the i-th training image and g is the transformation function performed by the network with parameters $\boldsymbol{\theta}$.

$$\mathbf{c}_i = g(\mathbf{I}_i; \boldsymbol{\theta}) \tag{4.4}$$

4.5.3 Gaussian Kernel Classifier

Given the set of Gaussian kernel centres C, a classifier can be formed that allows new examples to be categorised into one out of the set of semantic classes represented in the labelled training data. A classifier is formed by the weighted sum of the kernel distance calculations between an example feature embedding and the Gaussian centres. Classification of an example is achieved by first passing the given input image through the network, resulting in a feature embedding in the same space as the Gaussian centres. A probability distribution over classes is found by summing the influence of each Gaussian centre and normalising. Centres that are nearby the example in the feature embedding space have large influence over the example's predicted class, while centres that are further away from the example have smaller influence. A Gaussian centre contributes only to the probability of the class label associated with the training example coupled to that centre. In other words, a centre inherits its class label from the training example by which it is defined.

The probability that example feature embedding $\mathbf{x}$ has class label ℓ is:

$$\Pr(\mathbf{x} \in \text{class } \ell) = \frac{\sum_{i \in C} w_i f(\mathbf{x}, \mathbf{c}_i) \left[\mathbf{c}_i \in \text{class } \ell\right]}{\sum_{j \in C} w_j f(\mathbf{x}, \mathbf{c}_j)}, \tag{4.5}$$

where f is the kernel function and w_i is a learned weight for centre i. The term $[\mathbf{c}_i \in \text{class } \ell]$ evaluates to a value of 1 if centre $\mathbf{c}_i$ has class label ℓ and a value of 0 otherwise. If the example $\mathbf{x}$ is in the training set, the distance calculation to itself is omitted during the computation of the classification distribution, the objective function and the derivatives of the objective function.

The value of the shared σ defines the radius around a Gaussian centre that is considered to be important. It sets the distance from an example feature embedding at which the influence of a Gaussian centre should become negligible. That is, it defines the radius beyond which the returned kernel values decay to zero. A single global Gaussian σ is shared by all kernels, rather than allowing the value to differ for each centre. The shared σ means that the classifier is a weighted nearest neighbour-based approach. The value of σ can be learned end-to-end with the rest of the model parameters or it can be set to a reasonable value before training and fixed. By setting the value before training, the parameter is being treated as an extra hyperparameter of the model, similar to the learning rate, weight decay and momentum. These implementation details are discussed further in Section 4.9.2.2.

In Equation 4.5, each Gaussian centre has an associated weight, for example w_i for the i-th centre. The per-centre weights are learned end-to-end with the rest of the network weights ($\boldsymbol{\theta}$). By incorporating these weights, the influence of important centres can amplified and that of problematic centres can be diminished. Section 4.9.2.1 describes the implementation details for the per-kernel weights, while the impact that the weights have on the performance is investigated in Section 4.10.2.5.

4.5.4 Objective Function

The objective function used to train the network is simply the summed negative logarithm of the probabilities of the true class labels. For example, the loss for training example $\mathbf{x}$ with ground truth label R is:

$$loss(\mathbf{x}) = -\ln\left(\Pr(\mathbf{x} \in \text{class } R)\right). \tag{4.6}$$

During training, the model parameters should be updated such that the objective function is minimised.

This objective function means that examples are pushed and pulled in the feature embedding space proportional to the gradient of the Gaussian, as

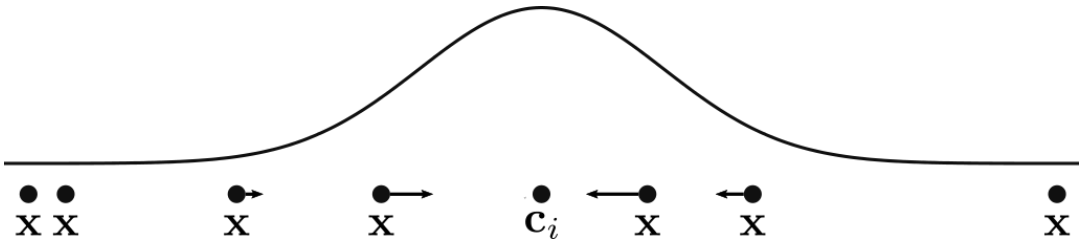


FIGURE 4.7: Gradients with respect to point $\mathbf{x}$ (at various locations) produced by a Gaussian kernel centred on point $\mathbf{c}_i$. Note that in this example $\mathbf{x}$ has the same class label as $\mathbf{c}_i$. The magnitude of the gradient is represented by the size of the arrow. Since the Gaussian decays to zero, the gradient also decays to zero at points that are far from the centre. This allows multiple clusters to form for a single class, if that is appropriate.

seen in Figure 4.7. The gradient decays to zero as an example point $\mathbf{x}$ moves further away from the Gaussian centre $\mathbf{c}$. As such, only the local neighbourhood around a given Gaussian centre is influenced during training. The size of the local neighbourhood is determined by the Gaussian parameter σ. This property of considering the local structure of the space allows multiple clusters to form for a single class, if that is appropriate. Examples in one cluster won't be influenced by examples in another, if the separation between those two clusters is large enough. This allows better representation of inter-class and intra-class similarities and variations in the data, compared to methods that demand a single cluster per class. Such methods include triplet-based approaches, which may select any examples of the same class to pull together, regardless of whether or not those two examples are already well clustered locally. As discussed in Section 4.1.1, this limits the variations in the data that can be represented by the model.

The same objective function is used for both classification and metric learning problems. This is possible since the Gaussian kernel classifier is directly computed from distances between features in the embedding space. This means that a network trained for classification will result in features of the same class being located near one another, and similarly a network trained for metric learning will result in an embedding space in which features can be well classified using the Gaussian kernels.

4.6 Nearest Neighbour Gaussian Kernels

The probability distribution over classes from Equation 4.5 is found by summing the influence of all Gaussian centres in C. Since the centres are defined to be the locations of the training set feature embeddings, and there can be any large number of training examples, computing this sum can become costly. It is also unnecessary to compute the full sum of Gaussian kernels, as most of the kernel values for a given feature embedding $\mathbf{x}$ will be effectively zero. This is because $\mathbf{x}$ will only lie within a subset of the Gaussian windows of the centres in C. The subset of meaningful Gaussian centres for a given example is likely to be significantly smaller than the total number of centres. This assumption is verified in Section 4.10.2.6. Since only those centres nearest an example feature embedding have non-negligible influence, only the local neighbourhood around the example needs to be taken into account. Considering the nearest Gaussian centres to a feature embedding ensures that most of the distance computations are pertinent to the probability and objective calculations.

In the interest of providing a solution that is scalable in terms of both training set size and the dimensionality of the embedding space, approximate nearest neighbour search is used to obtain candidate nearest neighbour lists for example feature embeddings. Compared to exact nearest neighbour search problems, which demand that the returned neighbours are the true nearest neighbours, approximate nearest neighbour methods return what can be considered a good guess of the true neighbourhood. This allows for a trade off between precision and computational efficiency. This is particularly suitable for the Gaussian kernel approach described in this chapter, since the neighbourhood sizes considered will be some order of magnitude larger than one. The relatively large neighbourhood sizes returned, although still small relative to the entire set of centres, means that it becomes very likely that the search algorithm will return the most important centres, that is, those that are nearest the example feature embedding. The precision of the approximate nearest neighbour search approach is evaluated in Section 4.8.2. This is achieved by analysing the variation between the true nearest neighbours of example points and the approximate nearest neighbours returned by the search algorithm.

Specifically, the search algorithm used is a fast approximate nearest neighbour graph (FANNG) [110]. This approach provides the most efficiency

when needing a high probability of finding the true nearest neighbours of a query point. Importantly, FANNG provides scalability in terms of the number of dimensions and the number of training examples. Full details on why this search method is suitable and how it is integrated into the Gaussian kernel metric learning approach can be found in Section 4.8.

4.6.1 Nearest Neighbour Classifier and Objective

Let S be the set of β nearest neighbour Gaussian kernel centres for example $\mathbf{x}$, where S is a subset of the set of all Gaussian kernel centres C, that is, $S \subseteq C$. The number of elements in S may be significantly smaller than the number of elements in C. Using this subset in the classifier, the probability distribution from Equation 4.5 becomes:

$$\Pr(\mathbf{x} \in \text{class } \ell) = \frac{\sum_{i \in C} w_i f(\mathbf{x}, \mathbf{c}_i) \, [\mathbf{c}_i \in \text{class } \ell] \, [\mathbf{c}_i \in S]}{\sum_{j \in C} w_j f(\mathbf{x}, \mathbf{c}_j) \, [\mathbf{c}_j \in S]}, \tag{4.7}$$

where the term $[\mathbf{c}_i \in \text{class } \ell]$ evaluates to a value of 1 if centre $\mathbf{c}_i$ has class label ℓ and a value of 0 otherwise. The term $[\mathbf{c}_i \in S]$ evaluates to a value of 1 if centre $\mathbf{c}_i$ is in the set of nearest neighbours for example $\mathbf{x}$ and a value of 0 otherwise. Again, if the example $\mathbf{x}$ is in the training set, the distance calculation to itself is omitted during the computation of the classification distribution, the objective function and the derivatives. The objective function optimised to train the model is unchanged from Equation 4.6, but uses the nearest neighbour-based probability calculation from Equation 4.7, rather than the calculation from Equation 4.5.

4.7 Making Training Feasible

Training the network should result in the Gaussian centres, that is, the training set feature embeddings, forming clusters according to class label. At a given training iteration, a batch of training examples is passed through the network and the probability distribution over classes is found for each example. The loss is computed for each example using the objective function from Equation 4.6 and the network weights are updated such that the loss for those examples will decrease. This is done by differentiating the objective function and updating the weights using an optimisation algorithm. These

implementation details are discussed in Section 4.9. However, it is infeasible to train the network by computing the probability distribution in Equation 4.5 in an exact manner. In this section, the problems that arise with exact training are discussed and solutions are proposed.

4.7.1 Moving Centres

At each training iteration, the network weights are updated. This means that the locations of the training set feature embeddings are constantly changing during training. Since the Gaussian centres are defined to be the locations of training set examples in the feature embedding space, the centres are also constantly moving. The true locations of an example's nearest centres are required to compute the distribution from Equation 4.7 correctly. However, finding these true locations online at every training iteration is intractable. At a given training iteration, to correctly compute this distribution requires $\beta+1$ images to be forward passed through the network, where β is the number of nearest neighbours considered. These are the images corresponding the β neighbouring centres and the image of the training example itself. For any reasonable neighbourhood size, this is intractable. For example, in the experiments presented in Section 4.10.2.3, neighbourhood sizes of up to 200 are considered. Forward passing 201 images through the network for a single training example, at a single training iteration, is clearly not feasible.

In order to solve this problem, periodic asynchronous updates of the Gaussian centres are introduced. During training, the feature embeddings belonging to each training example are stored. As training progresses, the stored centres are kept static and the true locations of the training feature embeddings are allowed to drift from the stored centres. At some interval, the stored centres are updated with the true training set feature embedding locations by forward passing the training set through the network. As training continues, the true locations again drift from the stored centre locations and the process repeats. Periodic asynchronous updates of the Gaussian centres is illustrated in Figure 4.8. The experiments presented in Section 4.10 show that the periodic asynchronous updates of the Gaussian centres allows meaningful learning to take place and importantly, allows training to be computationally reasonable.

The training process with periodic asynchronous updates of the Gaussian centres is outlined in Algorithm 1. The interval at which the stored centres

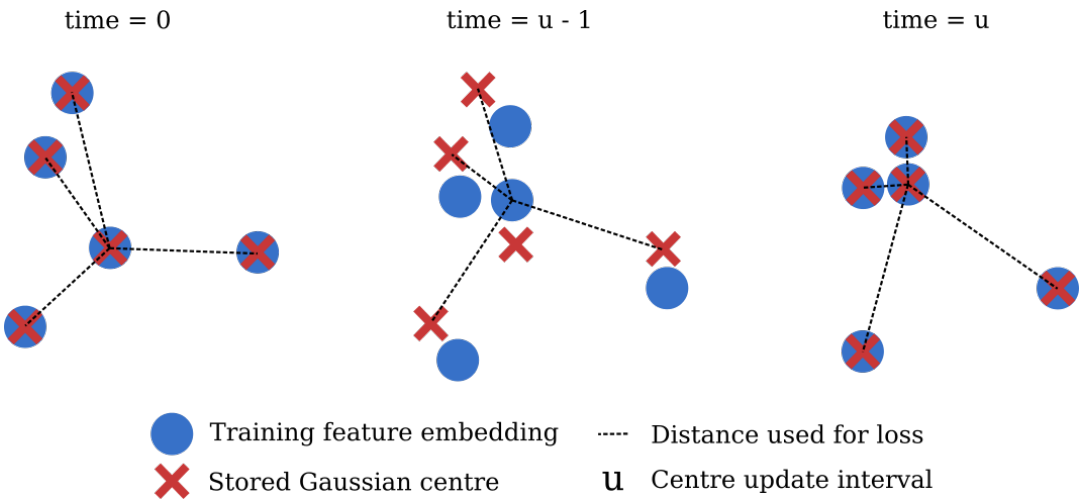


FIGURE 4.8: The true locations of the Gaussian centres are allowed to drift from the stored locations during training. The stored centres are updated periodically.

are updated is referred to as the *update interval* and is denoted as u. The update interval is represented in terms of training epochs, where an epoch is equal to the number of training iterations required to perform a full pass over the training set. For example, for a training set size of 10000 images and a batch size of 10, a training epoch is equal to 1000 training iterations. A static list of approximate nearest neighbours for each training set example is also computed once per update interval, after the stored centres are updated. This is discussed in more detail in Section 4.7.2. The set of stored nearest neighbours for the i-th training example is denoted as $S^{(i)}$, where $S^{(i)} \subseteq C$. The true Gaussian centre locations drift from the stored centres only during training; the true centres are used during testing and deployment, with classification carried out according to Equation 4.7. Note that for network architectures that incorporate dropout [213] during training, the stored Gaussian centres C do not have dropout applied.

Experimental results suggest that the length of the update interval has little impact on the final model performance. However, a larger update interval means that the approximations made during training are greater and therefore training may take longer to converge than with a smaller update interval. The time taken to update the centres and find the training set nearest neighbour lists depends on the training set size. This means that for small datasets, a short update interval can be used, as the update of centres and neighbours is fast to compute and can be carried out often, allowing the model to converge in fewer training epochs. For very large datasets, a longer

Algorithm 1 Pseudo-code for training. A batch size of one is shown for notational convenience only.

Precondition:

Set of training images $\mathcal{I}^{(t)} = \{\mathbf{I}_1, \ldots, \mathbf{I}_\alpha\}$
Vector of labels for training images $\mathbf{y} = [y_1, \ldots, y_\alpha]$
Set of Gaussian centres $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_\alpha\}$
Vector of per centre weights $\mathbf{w} = [w_1, \ldots, w_\alpha]$
Set of per training example nearest neighbour sets $\mathbf{S} = \{S^{(1)}, \ldots, S^{(\alpha)}\}$, where $S^{(i)} \subseteq C$ for all $i \in 1, \ldots, \alpha$
Neural network g with weights $\boldsymbol{\theta}$

while training **do**
  for each image $\mathbf{I}_i \in \mathcal{I}^{(t)}$ **do**
    $\mathbf{c}_i = g(\mathbf{I}_i, \boldsymbol{\theta})$ ▷ Update centres with current training features
  end for
  for u epochs **do** ▷ Update interval for centres
    for each image $\mathbf{I}_i \in \mathcal{I}^{(t)}$ **do**
      $\mathbf{x} = g(\mathbf{I}_i, \boldsymbol{\theta})$
      $p = \frac{\sum_{j \in C, j \neq i} w_j \exp\left(\frac{-|\mathbf{x}-\mathbf{c}_j|^2}{2\sigma^2}\right) [\mathbf{c}_j \in \text{class } y_i] \, [\mathbf{c}_j \in S^{(i)}]}{\sum_{k \in C, k \neq i} w_k \exp\left(\frac{-|\mathbf{x}-\mathbf{c}_k|^2}{2\sigma^2}\right) [\mathbf{c}_j \in S^{(i)}]}$ ▷ Prob. of true label
      $loss = -\log(p)$ ▷ Log prob. of true class
      Update network weights $\boldsymbol{\theta}$ and $\mathbf{w}$
    end for
  end for
end while
for each image $\mathbf{I}_i \in \mathcal{I}^{(t)}$ **do** ▷ Update centres a final time when done
  $\mathbf{c}_i = g(\mathbf{I}_i, \boldsymbol{\theta})$
end for

update interval should be used. Although training will take a greater number of epochs to converge, expensive updates of the centres and neighbours are carried out less often. Further discussion on the implementation details of the update interval is carried out in Section 4.9.2.3. Testing and deployment of the network is efficient regardless of the training set size. This is due to the properties of the approximate nearest neighbour search algorithm that are discussed in Section 4.8.2. Timing information regarding forward propagation through the model, which is used to update the stored centres, can be found in Section 4.10.2.3.

4.7.2 Neighbourhood Structure Changes

As the true Gaussian centres drift from the stored locations, the true nearest neighbours also drift from the computed nearest neighbour lists. Since it is intractable to find the true Gaussian centres online at each training iteration, it is also intractable to find the real nearest neighbour lists each time the network weights are updated. This problem is simply remedied by considering a larger number of nearest neighbours than would be required if all stored Gaussian centres and neighbour lists were up-to-date at all times. This solution is suitable because the feature embedding space changes slowly enough that it is highly likely many of an example's previously neighbouring Gaussian centres will remain relevant in subsequent training iterations. This assumption is verified in this section. Since the Gaussian kernel decays to zero as the distance between an example feature embedding and a Gaussian centre becomes large, it does not matter if a centre that is no longer near the example remains a candidate nearest neighbour. This simple solution increases computational complexity only modestly, as the total number of nearest neighbours required is still significantly smaller than the total training set size. An analysis of the number of nearest neighbours required per example during training is carried out in Section 4.10.2.6.

The proposed solution of simply considering a larger neighbourhood size assumes that the feature embedding space changes slowly enough during training that many of an example's previously neighbouring centres will remain relevant. This assumption is tested in Figure 4.9, which shows the mean fraction of the true nearest neighbours that are found in the 200 stored neighbours, at different points during training. Results are shown at 256 training iterations after the most recent update of the stored centres and nearest neighbour lists. The results in Figure 4.9 verify the assumption that the feature embedding space changes slowly enough during training that the majority of previously found nearest neighbours will remain pertinent in subsequent training iterations. This suggests the proposed simple solution of considering a larger neighbourhood size around training examples is suitable.

4.8 Fast Approximate Nearest Neighbour Graphs

The approximate nearest neighbour search method leveraged by our approach is a fast approximate nearest neighbour graph (FANNG) [110]. In

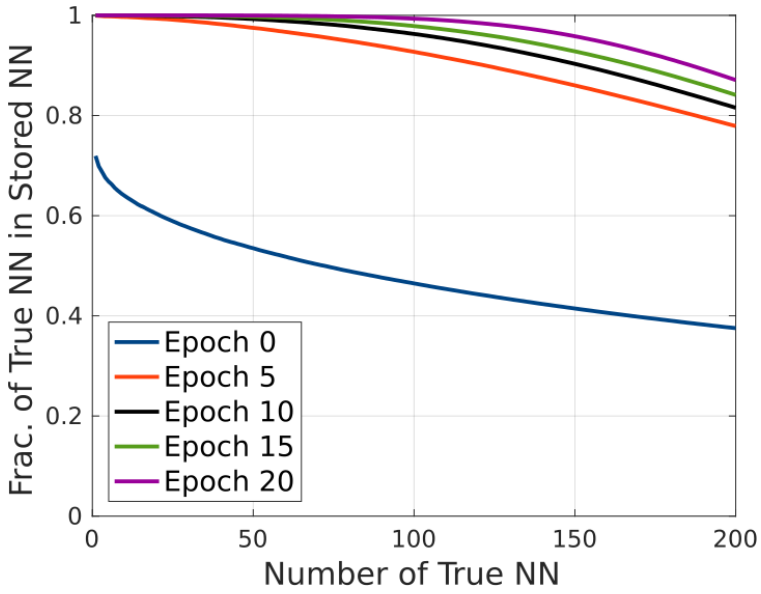


FIGURE 4.9: Mean fraction of the true nearest neighbours (NN) found in the 200 stored nearest neighbours, 256 training iterations after the most recent update of the Gaussian centres and stored neighbours. The number of true nearest neighbours considered is plotted from 1 up to 200, and the results are shown at various stages of training. Despite 256 updates being made to the network weights since the most recent update of the stored centres and neighbours, a large fraction of the true nearest neighbours are still found in the stored neighbour lists. The CUB Birds200 2011 dataset [200] and a VGG16 architecture [207] are used.

this section, building of the graph during training is discussed, as well as searching the graph, which is used both during training and testing. These are discussed only in terms of our proposed approach; full details on the build and search algorithms can be found in [110].

4.8.1 Building the Graph

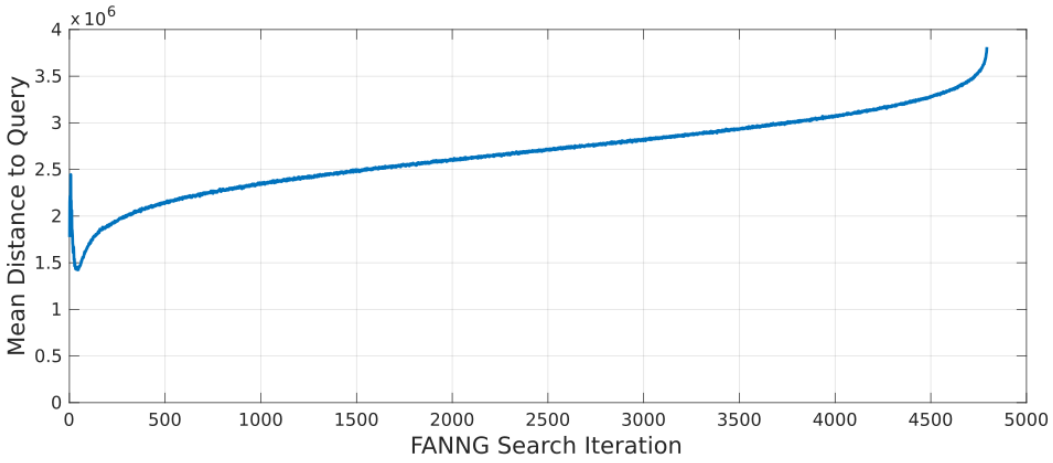
Nodes of the fast approximate nearest neighbour graph are defined for each training set feature embedding. The building process involves the definition of directed edges between nodes to create a graphical structure that is efficient to search. During the training of our model, the nodes of the graph are defined to be the stored feature embeddings, which are allowed to drift from the true locations, as discussed in Section 4.7.1. When the stored centre locations are updated, the nearest neighbour graph is rebuilt to represent these changes. At the conclusion of training, the graph should be rebuilt a final time and deployed with the converged model.

4.8.2 Searching the Graph

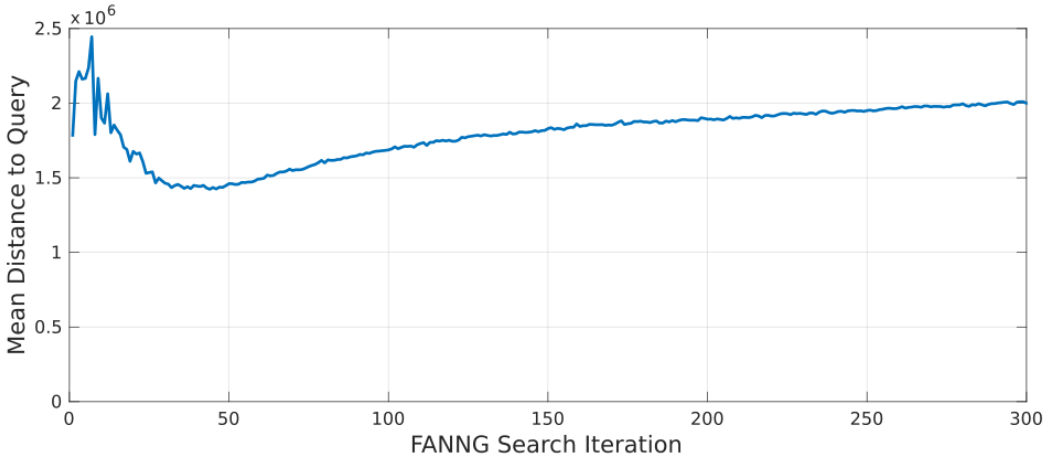
The approximate nearest neighbours of a query point are found by searching the graph. This is used in training to find the neighbourhood of training examples to compute the loss, and during testing to classify a given example. The search algorithm is iterative, with a new node, or training example, returned each iteration. If the query point is a training example, the search can begin at the node corresponding to that feature embedding. For query points that do not have nodes in the graph, such as test examples, the search can begin at any node. We chose to begin the search at a node near the training dataset mean for such examples. Searching of the graph is analysed in Figure 4.10. The training and validation sets of the CUB Birds200 2011 dataset [200] are used as the graph nodes and the query points, respectively.

Our approach requires not only the nearest neighbour to be returned but the local neighbourhood around a given query point. Figures 4.10A and 4.10B show that FANNG is able to achieve this goal. Figure 4.10A shows the mean distance to the query point as a function of the search iteration. The reported distance is the Euclidean distance between the query point and the node visited at a given search iteration. This is averaged across all query points. Figure 4.10B is the same plot, but zoomed into the earlier search iterations to highlight the behaviour of the algorithm when it is in the local neighbourhood of the query point. Once in the local neighbourhood of the query point, the search algorithm continues to return nearby examples before returning examples that are further away. This can be thought of as the algorithm spiralling out from the query point. As seen in Figures 4.10A and 4.10B, the number of search iterations required to return a local neighbourhood around a query point is significantly smaller than the size of the dataset. For example, a search size of around 100 would be reasonable for this dataset containing just under 5000 examples.

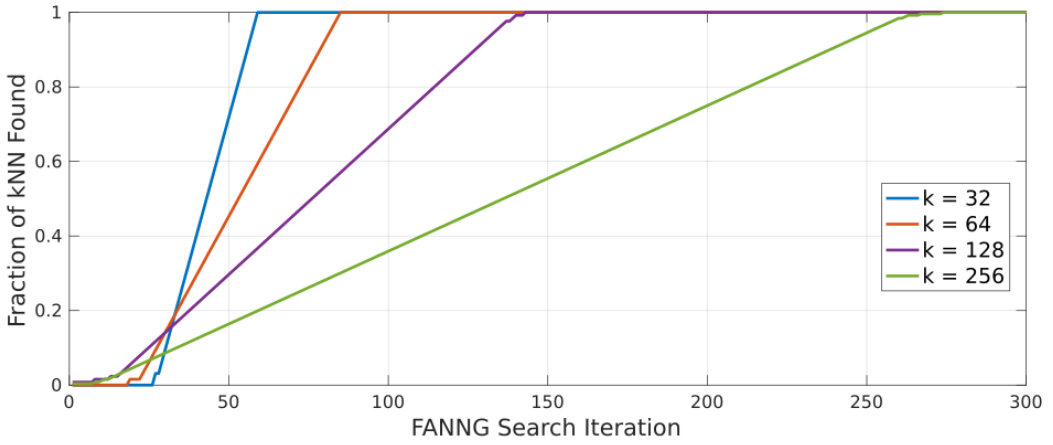
Figure 4.10C shows the fraction of the true k-nearest neighbours found as a function of the search iteration. Four values of k are considered: 32, 64, 128 and 256. It can be seen that the number of search iterations required to find all k true nearest neighbours is only slightly larger than k. For example, to find the true 128 nearest neighbours requires just 143 search iterations. This illustrates how FANNG allows our approach to be efficient and scalable to large datasets.



(A)



(B)



(C)

FIGURE 4.10: Searching a fast approximate nearest neighbour graph. (A) shows the mean distance to the query point as a function of the search iteration. (B) shows the same plot as (A), but zoomed in to the early search iterations. (C) shows the fraction of the true k-nearest neighbours returned as a function of the search iteration.

4.9 Implementation Details

In this section, details regarding the implementation of the proposed approach are presented. Model parameter and training hyperparameter settings are discussed, while the extraction of feature embeddings from various network architectures is detailed.

4.9.1 Optimisation Details

The optimisation algorithm makes updates to the model parameters in order to reduce the training error. A full description of model optimisation is given in Section 3.1. For our approach, the first order optimisation algorithm of mini-batch gradient descent is used. This optimiser uses several well established hyperparameters, including the base learning rate, momentum and weight decay rate. These parameters are set such that validation loss is minimised. Experimental investigation finds that a conventional range of values is suitable for the momentum and weight decay rate, such as 0.9 and 5×10^{-4}, respectively. The magnitudes of the gradients produced by the proposed Gaussian kernel approach are larger than those produced by a standard softmax-based network. As such, a smaller than conventional learning rate is required. The gradients produced by our approach are two orders of magnitude larger than those from the softmax network and as such, experimental results find that a comparatively small base learning rate of the order of 10^{-5} is suitable.

4.9.2 Introduced Parameters

This section discusses the parameters introduced by our proposed approach that are not standard in existing methods.

4.9.2.1 Per Kernel Weights

As seen in Equation 4.7, each Gaussian centre has an associated scalar weight that can be learned end-to-end with the rest of the convolutional neural network weights. These weights allow the influence of certain Gaussian centres

to be adjusted in a manner that reduces the training error. Experimental results suggest that the per kernel weights are largely insensitive to the learning rate. However, there appears to be a small benefit in multiplying the base learning weight by some larger factor, such as 1000, for these parameters. This is likely to offset the smaller than usual base learning rate required by our approach. The importance of learning the per kernel weights to the model performance is analysed in the ablation study in Section 4.10.2.5.

The per kernel weights are denoted as $\mathbf{w} = [w_1, \ldots, w_\alpha]$, where w_i is the weight for the i-th Gaussian centre and $\mathbf{w}$ is defined over the set of real positive numbers $\mathbf{w} \in \{\mathbb{R}_{>0}\}$. The initial value for all kernel weights should be set to one. Since the gradient descent algorithm makes updates to parameters without any regard to the parameters' valid range of values, the per kernel weights must be properly parameterised to avoid the kernel weights becoming negative. To achieve this, the weights are reparamaterised in the log space:

$$\mathbf{w} = \exp(\boldsymbol{\upsilon}), \tag{4.8}$$

where $\boldsymbol{\upsilon} = [\upsilon_1, \ldots, \upsilon_\alpha]$ are the parameters that are directly updated by the optimisation algorithm. This reparameterisation means that the gradient descent algorithm can make updates to the weights $\boldsymbol{\upsilon}$ without regard to the valid range of values for $\mathbf{w}$. The exponential ensures that the values of the weights $\mathbf{w}$ are positive. Since the initial values of $\mathbf{w}$ should be one, the initial values of the components of $\boldsymbol{\upsilon}$ should be set to zero. To update the weights, the partial derivatives of the objective function must be found with respect to the parameters in $\boldsymbol{\upsilon}$, rather than the parameters in $\mathbf{w}$. The effect that this has on the derivatives is shown in Equation 4.9.

$$\frac{\partial loss}{\partial \upsilon_i} = \frac{\partial loss}{\partial w_i} \frac{dw_i}{d\upsilon_i} = \frac{\partial loss}{\partial w_i} w_i \tag{4.9}$$

4.9.2.2 Gaussian Kernel Scale

The scale, or width, of the Gaussian kernel is controlled by the parameter σ from Equation 4.7. This parameter can either be learned end-to-end with the network weights or treated as a model hyperparameter by setting it to a reasonable value before training. Experimental results suggest that there is little, if any, benefit in allowing the value of σ to update during training. Even if the parameter is tuned during training, it must still be set to a reasonable

initial value. For these reasons, the experiments in Section 4.10 are carried out with a fixed σ.

In general, experimental investigation finds that the model is rather robust to the setting of σ. Training will converge at the same performance level over a wide range of σ values. However, if the value is set too small, no meaningful learning will take place as there will be no feature embeddings within a centre's small Gaussian window. Similarly, if the value is set too large, the majority of feature embeddings will be within a centre's Gaussian window and training will progress slowly, if at all. The appropriate range of values for σ will depend heavily on the number of dimensions in the feature embedding space. As the number of dimensions grows larger, the average distance between feature embeddings will also grow. The network architecture also plays a role in the appropriate value range for σ, as some architectures tend to produce feature embeddings with a larger range of values than others. For example, AlexNet [60] and VGG [207] architectures tend to produce a larger range than a ResNet [69] architecture. Further, networks that make use of dropout during training will require a larger σ, as the random zeroing of channels will increase the distance between training examples and stored centres. Finally, the appropriate range is loosely dependant on the dataset, however, this factor appears to be the least important, as the range of appropriate σ values overlaps considerably between datasets. In order to set the value of σ, the parameter should be treated as any other model hyperparameter, such as the learning rate, momentum or weight decay rate. That is, the value should be tuned such that validation or withheld set loss is minimised. For problems with no validation set, tuning the parameter such that the training set error steadily improves, generally leads to acceptable model performance.

If the value of σ is to be learned end-to-end with the network weights, it is important that the value remains a real positive number, that is, $\sigma \in \{\mathbb{R}_{>0}\}$. As with the per kernel weights in Section 4.9.2.1, this is achieved by reparametrising σ in the log space:

$$\sigma = \exp(\phi), \tag{4.10}$$

where ϕ is the parameter to which updates are made by the optimisation algorithm. This impacts the derivatives with respect to the scale term in the same manner as shown in Equation 4.9.

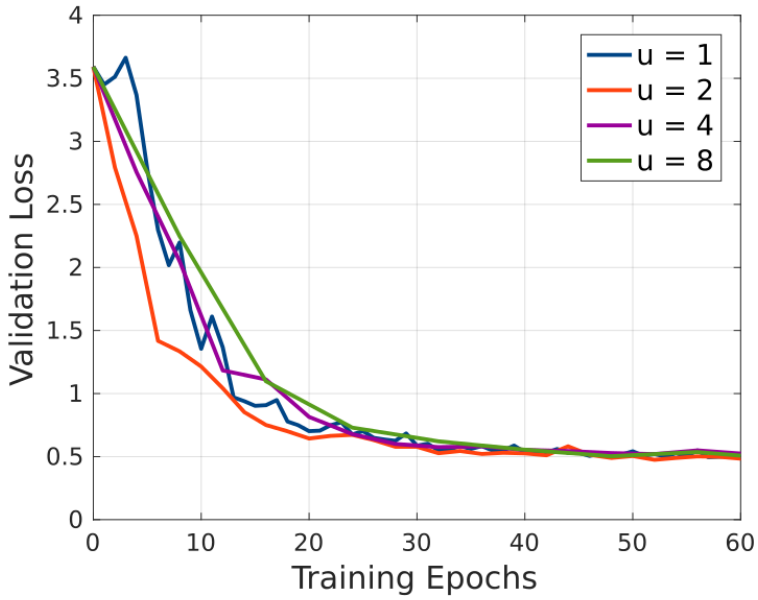


FIGURE 4.11: Validation loss on the Oxford 102 Flowers dataset [202] for different update intervals (u) with a VGG16 architecture [207].

4.9.2.3 Update Interval

As discussed in Section 4.7, introducing periodic asynchronous updates of the stored Gaussian centres allows training to become tractable. The impact that the frequency at which the centres are updated, referred to as the *update interval*, has on the training process is analysed in this section. In general, a larger update interval will result in more training epochs being required in order for the model to converge. The time taken to perform update, which includes finding the approximate nearest neighbours, is dependant on the training set size and the number of dimensions in the feature embedding space. For very large datasets, frequent updates should be avoided, even if it results in more training epochs, as the training duration in terms of overall time will be decreased.

The impact that the update interval has on the model in terms of the validation set error is seen in Figure 4.11. This plot shows that the model is robust to the length of the update interval, with only marginal differences observed in the final validation loss.

4.9.3 Network Architectures and Feature Embeddings

Two methods of extracting feature embeddings from standard convolutional neural network architectures are used in the experiments in Section 4.10. Starting with a conventional softmax-based network, the first method is to simply remove the softmax layer and the final fully connected layer that outputs the class activation vector. After removing those layers, the new final layer is considered the feature embedding layer and its output is treated as a feature embedding. For both AlexNet and VGG16 architectures, this makes the feature embedding layer FC7, which produces a 4096-dimension feature embedding. For GoogLeNet and ResNet architectures, the embedding layer becomes the final average pooling layer in the respective networks. This produces a 1024-dimension feature embedding for GoogLeNet and a 2048-dimension feature embedding for ResNet.

The second method is used if the feature embedding produced by the first method is not of a suitable dimensionality. To change the dimensionality of the feature embedding space, the softmax and classification fully connected layers are removed, as in the first method, but a new fully connected layer is added to the end of the network. The feature embedding dimensionality can be controlled by setting the number of output channels produced by this new fully connected layer.

If the network architecture incorporates dropout, such as in VGG architectures after the FC6 and FC7 layers, it is important to ensure that the stored Gaussian kernel centres do not have dropout applied. The feature embeddings computed at each training iteration for the examples in that batch, however, may have dropout applied. The activation function used in the network architectures considered is the ReLU. Since this function zeros any negative input, ReLUs are not placed after the feature embedding layer. This avoids loss of information in the feature embedding space by considering the full range of possible values. For networks containing batch normalisation layers, we find that using the global statistics during training yields better performance than using batch statistics.

4.10 Experiments

Our proposed approach is quantitatively and qualitatively evaluated in this section. The approach is compared to state-of-the-art deep metric learning methods in Section 4.10.1, by applying it to a transfer learning problem. Additionally, the importance of the feature embedding space dimensionality is explored. Section 4.10.2 investigates the efficacy of the proposed approach to image classification, comparing results to a conventional softmax-based convolution neural network. Ablation studies are also carried out in Section 4.10.2.

4.10.1 Metric Learning

The standard way to evaluate metric learning and feature embedding learning problems in the literature is to apply the learned models to transfer learning problems [18, 19, 20, 23, 22, 21]. This is done by training and evaluating the model on different sets of classes. At test time, the model is presented with images belonging to semantic classes never before seen by the model. The feature embedding space is evaluated by examining how well the notion of similarity is encoded by distance in the space. In other words, a well performing model will produce feature embeddings that are located nearby for semantically similar examples, despite the model not being trained on the test set classes.

4.10.1.1 Datasets

Two datasets are used to evaluate our approach: Stanford Cars196 [201] and CUB Birds200 2011 [200]. The Cars196 dataset that contains 16,185 images belonging to 196 different classes of car, split at the level of make, model and year. Example class labels include *Audi S4 Sedan 2007*, *Audi S4 Sedan 2012*, *BMW 3 Series Sedan 2012* and *BMW 3 Series Wagon 2012*. The Birds200 dataset contains 11,788 images belong to 200 different bird species. Example classes include *Herring Gull*, *Western Gull*, *Belted Kingfisher* and *Green Kingfisher*. Both datasets have object bounding boxes available. However, for fair comparison to existing approaches, these are not used in the metric learning experiments.

4.10.1.2 Experimental Set-Up

The experimental set-up from [19, 20, 22, 21] is followed, allowing fair comparison to the state-of-the-art methods. For each dataset, the first half of classes is used for training and the second half is used for testing. This results in 98 classes each for training and testing for the Cars196 dataset, and 100 each for training and testing for the Birds200 dataset. All images are resized to 256x256 pixels. Training data is augmented using random cropping and random horizontal mirroring.

Mini-batch gradient descent is used as the optimisation algorithm, while GoogLeNet [66] with ImageNet [61] pre-trained weights is used as the model. The softmax and class-specific fully connected layers are removed from the standard GoogLeNet architecture. A new fully connected layer is added as the embedding layer, with the number of output channels determining the number of dimensions in the feature embedding space. No auxiliary losses are used. A neighbourhood size of 100, update interval of 10 epochs, batch size of 20, base learning rate of 10^{-5} and weight decay of 2×10^{-4} is used. The Gaussian σ used depends on the number of dimensions in the feature embedding. Values between 10 and 30 work well for this task.

4.10.1.3 Evaluation Metrics

Two metrics are used to evaluate the learned feature embedding spaces. Both of the metrics consider only the test set feature embeddings, that is, those belonging to novel classes on which the network was not trained. The first, normalised mutual information (NMI) [214], is a clustering metric that essentially measures the homogeneity of cluster assignments. The measure is the ratio of mutual information and average entropy of a set of clusters formed by the feature embeddings and the true labels of the feature embeddings. It evaluates only for the number of clusters equal to the number of classes in the test set. As discussed in Section 4.1.1, a good feature embedding space does not necessarily have only one cluster per class, but may have multiple well formed clusters per class in the space. This means that the mutual information for our method may be higher than reported with this metric. Further, the cluster assignments are found using the k-means algorithm [8], meaning that the evaluation metric is not deterministic, due to the initialisation process. Nevertheless, in the interest of comparing to existing methods that evaluate on this metric, NMI results are reported.

The second metric, Recall@k (R@k), is more appropriate for evaluating feature embedding spaces. This metric reports the percentage of test set feature embeddings that have at least one example of the same class in their nearest k neighbouring feature embeddings. For example, a Recall@1 of 100 means that every test example's nearest neighbour is of the same class. Results are reported for k equal to 1, 2, 4 and 8 nearest neighbours.

4.10.1.4 Feature Embedding Dimensionality

The importance of the feature embedding dimensionality is investigated. The number of dimensions in the feature embedding space is set by the number of output channels in the embedding layer. A similar study in [19] suggests that the number of dimensions is not important for triplet networks, in fact, increasing the number of dimensions can be detrimental to performance. The proposed Gaussian kernel approach with increasing dimensionality is compared to two triplet-based approaches with increasing dimensionality: standard triplet loss [109, 18] and lifted structured embedding [19]. Results for the compared approaches are taken from the study in [19].

Figures 4.12A and 4.12B show the effect of the embedding size on NMI score for Cars196 and Birds200, respectively. Increasing the number of dimensions for triplet-based networks does not result in improved performance. The NMI score oscillates or deteriorates as the dimensionality increases. For our proposed approach, however, the NMI score clearly improves as the number of dimensions grows larger. Similar behaviour is seen in Figures 4.12C and 4.12D, which show the Recall@k metric for the proposed approach on Cars196 and Birds200, respectively. Again, these plots show a clear performance improvement as the dimensionality of the feature embedding space increases.

The likely explanation for this behaviour is that our approach utilises significantly more information at each training iteration compared to triplet-based approaches. For a given training example at a given training iteration, a standard triplet approach considers only two examples in the space: one of the same class and one of a different class. While some approaches select difficult triplets to improve the effectiveness of each training iteration [18, 21], or cleverly structure the data in the batch to allow multiple comparisons [19, 20], the amount of information is still significantly less than in our approach, which considers all examples within the neighbourhood. As a result, triplet-based

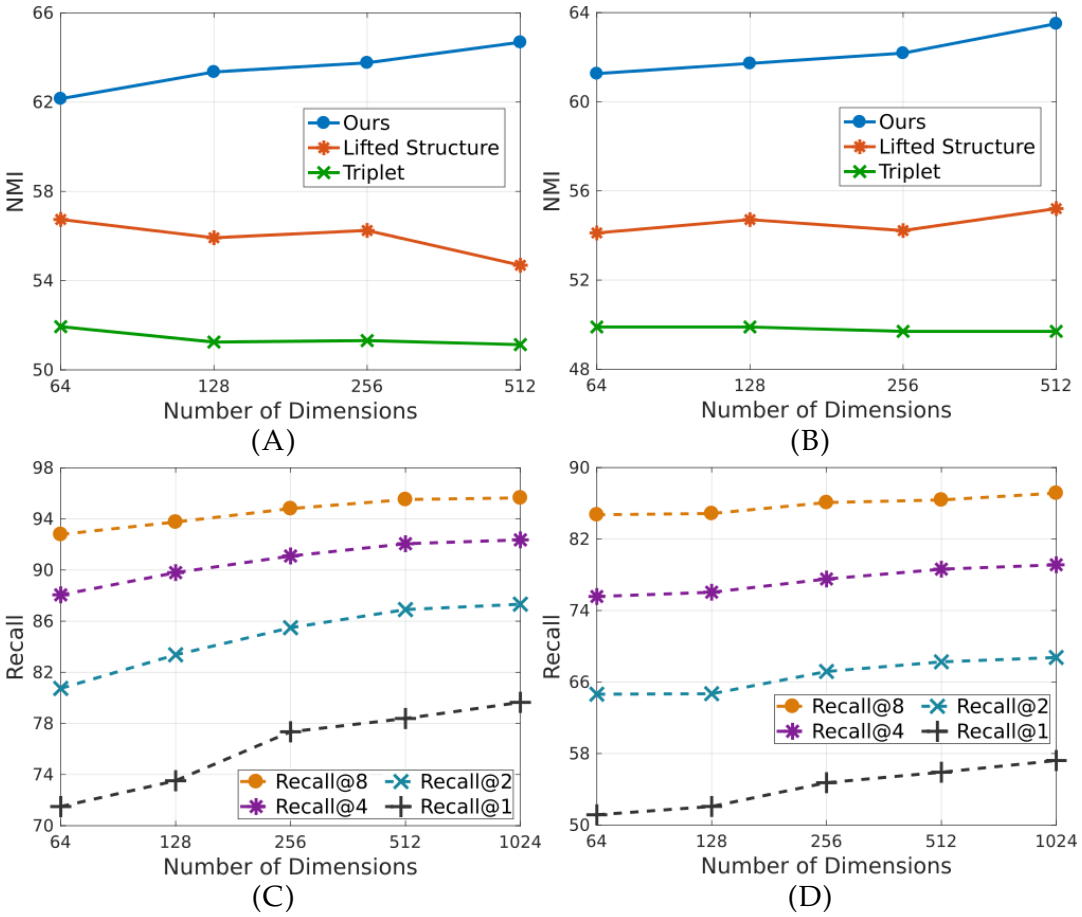


FIGURE 4.12: Top: NMI score with increasing feature embedding dimensionality on Cars196 (A) and Birds200 (B). Bottom: Recall@k nearest neighbours for our proposed approach on Cars196 (C) and Birds200 (D).

approaches are less able to take advantage of a higher-dimensional feature embedding space, compared to our approach.

4.10.1.5 Comparisons to Existing Methods

The proposed method is compared to existing state-of-the-art approaches on the Cars196 dataset in Table 4.1 and the Birds200 dataset in Table 4.2. The compared results are taken from [22] and [21]. As discussed in Section 4.10.1.4, the dimensionality does not have much impact on the compared approaches. As such, all results in [22] and [21] are reported using 64 dimensions. For fair comparison, results for our method are reported at 64 dimensions, but also at the better performing higher dimensions.

At 64 dimensions, our approach outperforms the compared methods on both datasets and across all evaluation metrics. The gain is particularly large on

TABLE 4.1: Feature embedding results on Cars196.

	Dims	R@1	R@2	R@4	R@8	NMI
Semi-hard [18]	64	51.54	63.78	73.52	82.41	53.35
LiftStruct [19]	64	52.98	65.70	76.01	84.27	56.88
N-pairs [20]	64	53.90	66.76	77.75	86.35	57.79
Tripl/Gbl [23]	64	61.41	72.51	81.75	88.39	58.20
Cluster [22]	64	58.11	70.64	80.27	87.81	59.04
SmrtMine [21]	64	64.65	76.20	84.23	90.19	59.50
Ours	64	**71.05**	**80.74**	**88.06**	**92.79**	**62.15**
Ours	128	73.52	83.37	89.80	93.76	63.35
Ours	256	77.35	85.49	91.10	94.81	63.76
Ours	512	78.39	86.91	92.06	95.52	64.68
Ours	1024	**79.65**	**87.33**	**92.36**	**95.65**	**65.30**

TABLE 4.2: Feature embedding results on Birds200.

	Dims	R@1	R@2	R@4	R@8	NMI
Semi-hard [18]	64	42.59	55.03	66.44	77.23	55.38
LiftStruct [19]	64	43.57	56.55	68.59	79.63	56.50
N-pairs [20]	64	45.37	58.41	69.51	79.49	57.24
Tripl/Gbl [23]	64	49.04	60.97	72.33	81.85	58.61
Cluster [22]	64	48.18	61.44	71.83	81.92	59.23
SmrtMine [21]	64	49.78	62.34	74.05	83.31	59.90
Ours	64	**51.15**	**64.64**	**75.57**	**84.72**	**61.26**
Ours	128	52.08	64.69	76.05	84.86	61.72
Ours	256	54.74	67.18	77.53	86.09	62.18
Ours	512	55.91	68.26	78.63	86.38	63.50
Ours	1024	**57.22**	**68.75**	**79.12**	**87.14**	**63.95**

the Cars196 dataset, with a relative improvement of around 10% in the Recall@1 measure over the best performing compared method. Beyond a dimensionality of 64, our approach sees improvement in all evaluation metrics. At a dimensionality of 1024, our approach sees an improvement of 8.6% in the Recall@1 measure (relative improvement of 12%), compared to the 64 dimension version. These results show that the proposed Gaussian kernel approach is not only able to produce better compact feature embeddings than existing methods, but can also take advantage of a larger embedding space.

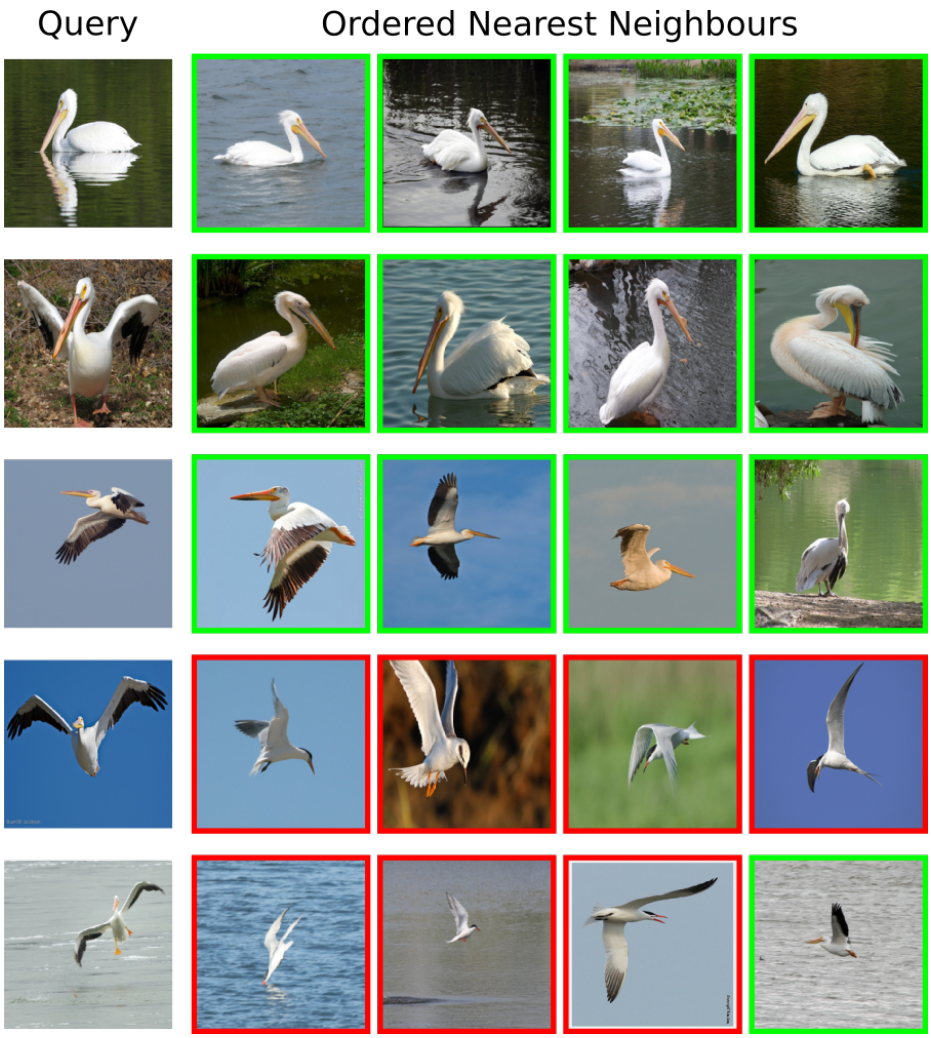


FIGURE 4.13: First through fourth nearest neighbours for example query images from the Birds200 test set. All query images are of the same class and the network was not trained on any classes from the test set. A green border around the neighbour images indicates that it is of the same class as the query image, while a red border indicates that it is of a different class. Interesting success and failure cases are shown. The embedding size is 64 dimensions.

4.10.1.6 Visualising the Feature Embedding Space

Figure 4.13 shows the four nearest neighbours for example query images from the Birds200 test set. All query images are from the same class. The figure shows interesting success and failure cases.

A t-SNE [215] visualisation of the learned feature embedding space for the Birds200 dataset is shown in Figure 4.14. Similarly, Figure 4.15 shows a visualisation for the Cars196 dataset. The t-SNE algorithm is a dimension reduction method that is well suited to visualising high-dimensional data.

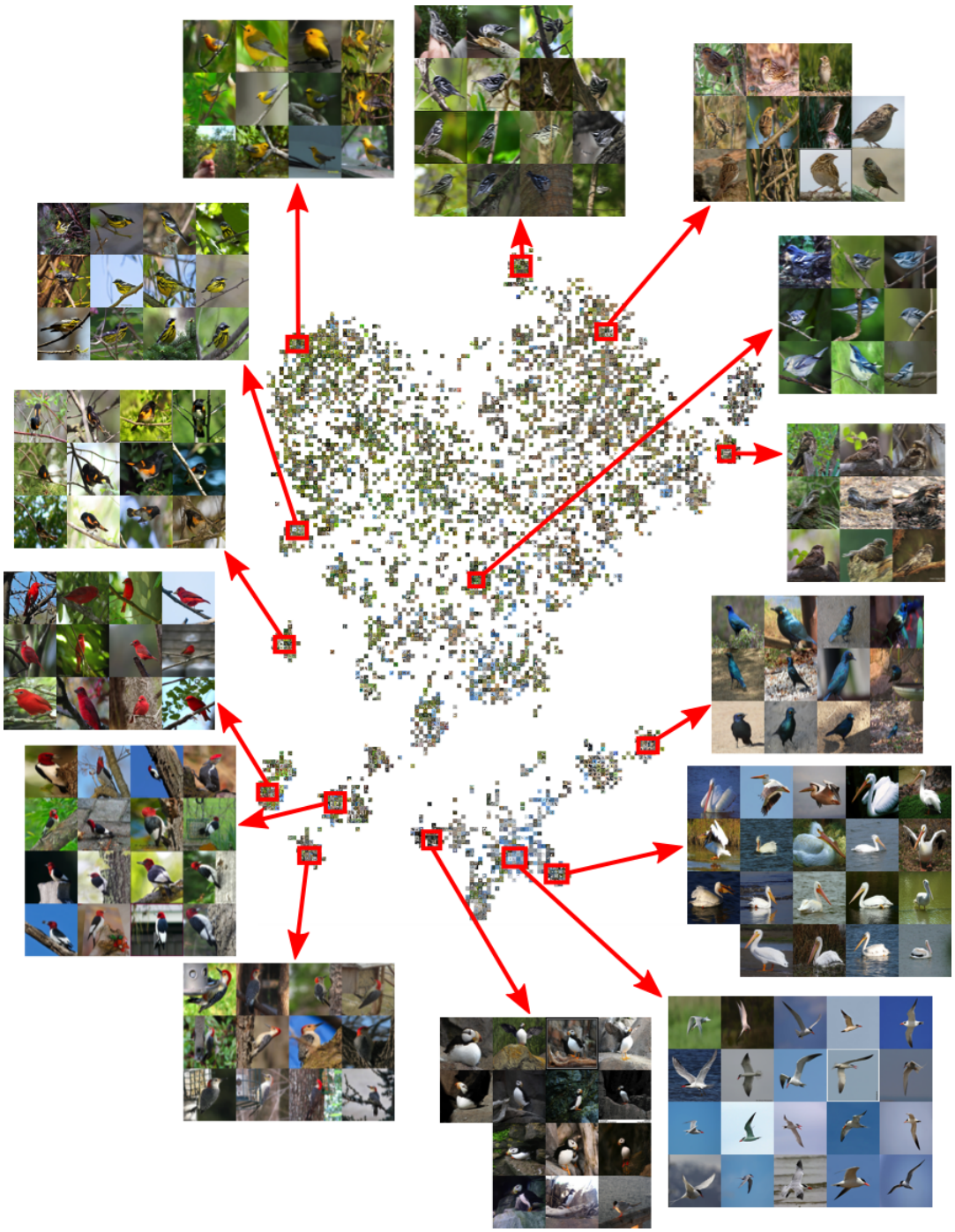


FIGURE 4.14: Visualisation of the Birds200 test set embedding space from Section 4.10.1.5. All species of bird visualised are from withheld classes that were not present during training. Despite this, examples are still well clustered based on species and attributes. The visualisation was obtained using the t-SNE algorithm [215].

In this case, the 64-dimension feature embeddings are mapped to two dimensions. The location of a feature embedding in the two-dimensional space is marked by the image corresponding to that example. The visualised feature

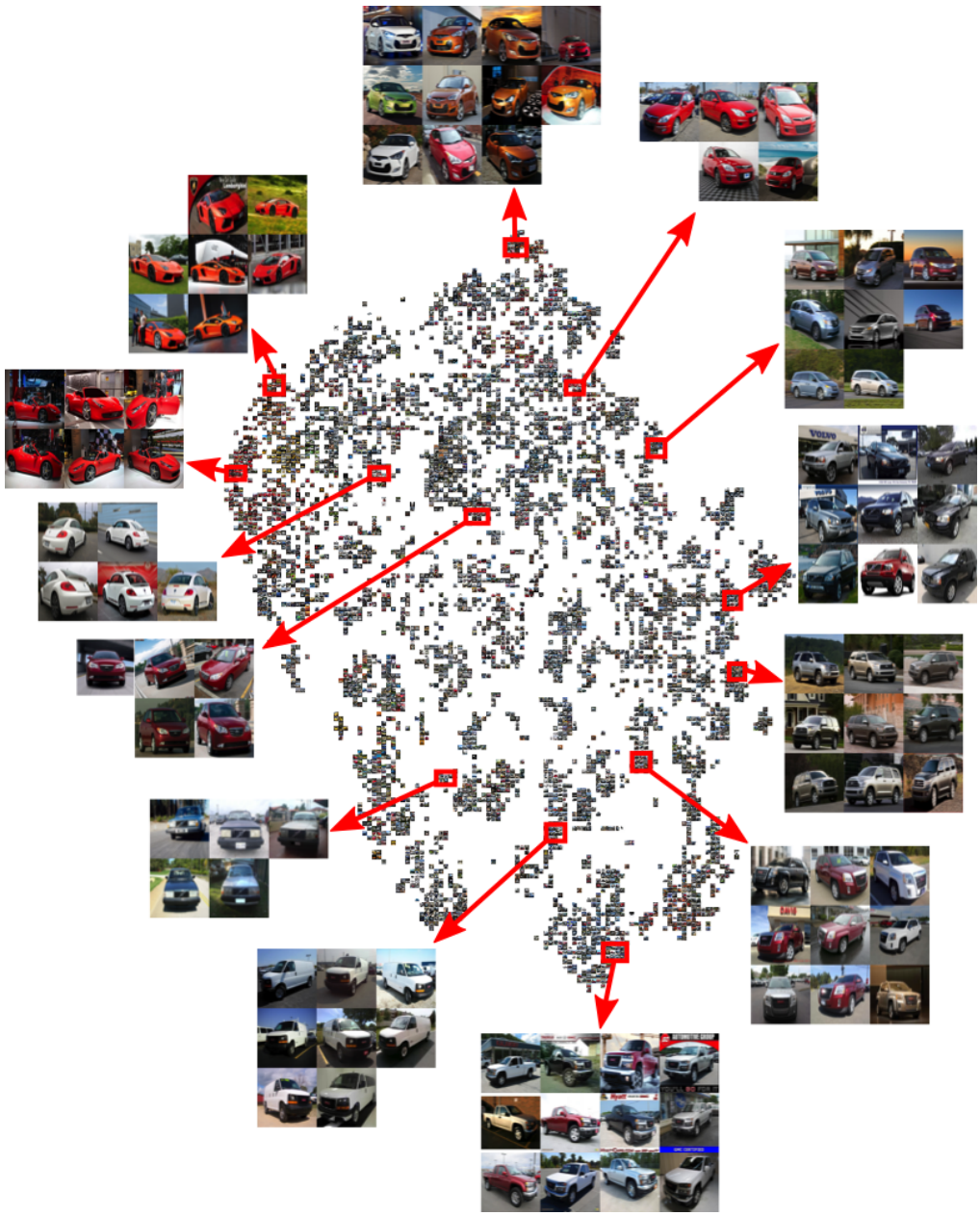


FIGURE 4.15: Visualisation of the Cars196 test set embedding space from Section 4.10.1.5. All models of car visualised are from withheld classes that were not present during training. Despite this, examples are still well clustered based on car model and attributes. The visualisation was obtained using the t-SNE algorithm [215].

embeddings are from the test set, meaning that the network was not trained on any of the classes seen in the visualisations. Despite belonging to withheld novel classes, examples are still well clustered based on class and attributes.

In the Birds200 visualisation in Figure 4.14, the bottom right of the visualised space shows different clusters of water-based bird species located nearby. On the bottom left, different species of woodpecker, which belong to the same

genus, are located nearby. Similar behaviour is seen in the Cars196 visualisation in Figure 4.15. For example, the top left shows clusters of different sports car models located nearby, while the middle right of the visualisation shows co-location of different models of SUV. This fine-grained and hierarchical semantic information is learned implicitly by the metric learning algorithm. Distance in the feature embedding space as a measure of the notion of similarity has transferred to novel classes that have never before been seen by the network. This ability to transfer to unseen classes would allow a vision system to make meaningful inference about novel observations.

4.10.2 Image Classification

The suitability of our approach to image classification is evaluated in this section by comparing performance to conventional softmax-based convolutional neural networks. This is the standard classification problem; the network is tested on withheld examples belonging to the same set of classes on which the network was trained. Additionally, the efficacy of the proposed approach to few-shot learning is investigated and an ablation study is presented. We also investigate how well our proposed approach represents fine-grained semantic information, such as attributes.

4.10.2.1 Datasets

Experiments are carried out on four datasets: CUB Birds200 2011 [200], Stanford Cars196 [201], Oxford 102 Flowers [202] and Leafsnap [203]. All datasets are split into training, validation and test sets. Hyperparameters for both the proposed approach and the compared approach are selected such that the validation loss is minimised. Dataset information and splits can be sumarised as follows:

- The Birds200 dataset contains 11,788 images belonging to 200 different bird species. Example classes include *Herring Gull*, *Western Gull*, *Belted Kingfisher* and *Green Kingfisher*. Images are cropped using the provided bounding boxes. The validation set is created from 20% of the standard training data.
- The Cars196 dataset contains 16,185 images belonging to 196 different classes of car, split at the level of make, model and year. Example class labels include *Audi S4 Sedan 2007*, *Audi S4 Sedan 2012*, *BMW 3 Series*

Sedan 2012 and *BMW 3 Series Wagon 2012*. Images are cropped using the provided bounding boxes. Since no standard validation set exists for this dataset, 30% of the training data is taken as validation.

- The Oxford 102 Flowers dataset is made up of images from 102 different flower species. Example classes include *Sword Lily*, *Water Lily* and *Toad Lily*. The standard splits are used for the Flowers102 dataset.
- Finally, the Leafsnap dataset consists of images of different leaf species, split into those taken in controlled lab conditions and uncontrolled field conditions. Classification is evaluated using only the challenging field images. Example classes include *Acer Ginnala* and *Acer Rubrum*. The field dataset contains 7719 images belonging to 185 classes of leaf species. The data is split into 50%, 20% and 30% for training, validation and testing, respectively.

4.10.2.2 Experimental Set-Up

Three different network architectures are used to evaluate classification performance: AlexNet [60], VGG16 [207] and ResNet50 [69]. For each of these architectures, the softmax layer and final class-specific fully connected layer are removed. No new layers are added, meaning that the final remaining layer is taken as the embedding layer. For both AlexNet and VGG, this makes FC7 (with dropout and no ReLU) the embedding layer, producing a 4096-dimension feature embedding. While for ResNet, the final average pooling layer becomes the embedding layer, producing a 2048-dimension feature embedding. Experiments find that following the ResNet embedding layer with a dropout layer results in a small performance gain for both the proposed approach and the softmax baseline.

Hyperparameters are selected that minimise the validation loss. Mini-batch gradient descent optimisation is used for all approaches. On the Birds200 dataset, a batch size of 20, update interval of 10 epochs and base learning rate of 10^{-5} is used for the proposed Gaussian kernel approach. A Gaussian σ of around 100 is found to be suitable for the 4096-dimension VGG16 embeddings on Birds200 and a neighbourhood size of 200 is used, unless otherwise noted. All networks are initialised with ImageNet [61] pre-trained weights.

4.10.2.3 Softmax Baseline Comparison

The proposed Gaussian kernel approach is compared to a softmax baseline in this section. Experiments investigate the performance with both diversity of dataset and diversity of network architecture.

Diversity of Architecture Evaluation is carried out over three different network architectures, on the same dataset. Table 4.3 shows the classification accuracy of the softmax baseline and our approach on the Birds200 dataset, with an AlexNet, VGG16 and ResNet50 base network architecture. The proposed approach outperforms the softmax counterpart across all three base architectures. The gain is largest for the AlexNet architecture, with our approach resulting in a 4.54% absolute increase in accuracy. This amounts to a reduction of 12% in errors compared to the softmax baseline, that is, $4.54/(100 - 62.41) = 12$. Similarly, for a VGG architecture the absolute gain in classification accuracy is 3.26%, amounting to a reduction of around 13% in errors compared to the softmax-based network.

The performance gain seen for the ResNet architecture is smaller than for the AlexNet and VGG architectures. An absolute improvement of 0.93% in classification accuracy is achieved by the proposed approach, amounting to a reduction of around 4% in errors compared to the softmax baseline. The reason for this is two fold. Firstly, as the baseline performance increases, the room for improvement shrinks. This is seen in the drop in improvement between AlexNet and VGG, as well as between VGG and ResNet. Secondly, a ResNet architecture contains significantly more non-linear activation units than the AlexNet and VGG architectures. This means that there is less improvement seen when using the highly non-linear Gaussian Kernel classifier and loss function. Nevertheless, our approach outperforms a softmax-based network across the three presented architectures. The inference time of the proposed approach, which includes forward passing pre-processed data through the model and performing classification, is approximately 14 milliseconds for a single image and 61 milliseconds for a batch size of 20, when using a ResNet50 architecture. In order to update the stored centres during training, the classifier portion of the model does not need to be utilised. This means that the forward pass time is reduced.

TABLE 4.3: Birds200 test set accuracy with various networks.

Base Network	Softmax	Ours
AlexNet	62.41	**66.95**
VGG16	75.37	**78.63**
ResNet50	78.05	**78.98**

TABLE 4.4: Test accuracy on four classification datasets.

Dataset	Softmax	Ours
Flowers102	82.79	**86.26**
Cars196	85.67	**86.52**
Leafsnap Field	73.80	**75.96**
Birds200	75.37	**78.63**

Diversity of Dataset Our approach is further evaluated on various datasets, with a fixed base network architecture. In addition to the Birds200 dataset, classification accuracy of the softmax baseline and our proposed approach on Cars196, Flowers102 and Leafsnap are seen in Table 4.4. A VGG16 base network architecture is used. Note that the Birds200 results are the same as from Table 4.3. The proposed Gaussian kernel approach outperforms the softmax counterpart on all four datasets. The improvement is largest for the Flowers102 dataset, with an absolute increase of 3.47%, which equates to an approximately 20% reduction in errors over softmax. For the Leafsnap dataset, the absolute improvement is 2.16% and the reduction in errors is approximately 8%. Finally, on the Cars196 dataset, our approach results in an absolute improvement of 0.85% and an approximately 6% reduction in errors, compared to the softmax baseline.

When training with a softmax classifier on the Birds200 dataset, validation loss plateaus at around 7000 iterations. For the proposed Gaussian kernel approach, the number of iterations taken for validation loss to stop improving depends on the update interval, that is, the interval at which the Gaussian centres are updated and the nearest neighbours are computed. For update intervals of 1, 5 and 10, validation loss stops improving at around 8500, 12000 and 15000 iterations, respectively. The softmax network is able to converge in fewer iterations than our approach. This is likely due to the Gaussian centres not being up-to-date at all times, leading to weight updates that are less effective than in the ideal scenario. However, as discussed in Section 4.7, keeping the Gaussian centres up-to-date at all times in intractable.

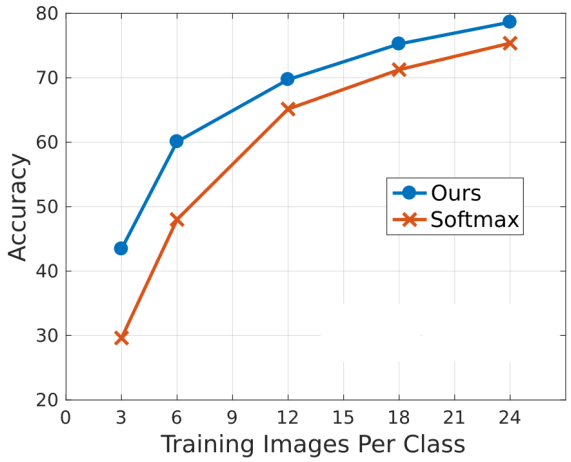


FIGURE 4.16: Effect of the number of training examples per class on the test set accuracy of Birds200, with a VGG16 architecture.

4.10.2.4 Impoverished Training Data

Our proposed approach is further evaluated by analysing the impact that the number of training examples has on the performance. There is a particular interest in the case of impoverished training data, where very few training images exist per class. This is because softmax-based networks generally perform poorly when training data is limited. Learning from a few examples per class is known as few-shot learning. The effect of the number of training examples on the classification performance is analysed across a range from few-shot learning up to full training set size. Figure 4.16 shows the classification accuracy of the softmax baseline and the proposed approach against the training set size.

As seen in Figure 4.16, the proposed Gaussian kernel approach outperforms the softmax baseline at all training set sizes from three images per class up to the standard training set size. Interestingly, the performance gain over softmax is largest when the number of training examples per class is small. For the few-shot classification scenario of three images per class, the proposed approach outperforms the softmax baseline by 13.8%. This is compared to the 3.26% absolute gain seen when the entire training set is used. This is an important result, since labelled training data is often difficult obtain. As such, the ability to learn from impoverished data can often be vital. This is particularly true in robotic applications, where a robot may observe novel objects in the environment and obtain a small number of labels in an active learning setting.

TABLE 4.5: Ablation study for classification on Birds200.

Initial Network Weights	Tune σ	Learn per Kernel Weights	Fine-tune Network Weights	Test Accuracy
Random	Yes	No	No	1.35
ImageNet	Yes	No	No	47.32
ImageNet	Yes	Yes	No	49.22
ImageNet	Yes	No	Yes	77.94
ImageNet	Yes	Yes	Yes	78.63

4.10.2.5 Ablation Study

Table 4.5 shows an ablation study for the proposed approach. The study shows how each of the following impacts the classification performance: network weight pre-training, fine-tuning the network weights on the given dataset with the Gaussian kernel loss and learning the per-kernel weights $\mathbf{w}$. An appropriate Gaussian σ value is required to perform classification with the Gaussian kernels, as such, the value of the Gaussian kernel σ is first tuned to minimise validation loss for each of the five arrangements of settings shown. Note that when *Learn per Kernel Weights* is marked as *No*, the weights in $\mathbf{w}$ are fixed at values of one.

A random classifier on the Birds200 dataset would achieve a classification accuracy of 0.5%, since there are 200 classes. Table 4.5 shows that the Gaussian kernel classifier with random network weights achieves a higher accuracy than random guessing, at 1.35%. Initialising the network with ImageNet pre-trained weights unsurprisingly has a significant impact on the accuracy. Without any fine-tuning of the network weights or learning of the per kernel weights on Birds200, ImageNet initialisation achieves an accuracy of 47.32%. Learning the per kernel weights on the Birds200 dataset, but fixing the network weights at the ImageNet initialisation results in a 1.9% increase in accuracy. Conversely, fixing the per kernel weights at values of one and fine-tuning the network weights using the Gaussian kernel loss on Birds200 results in a 30.62% improvement over the ImageNet initialisation. Finally, learning the per kernel weights together with fine-tuning the network weights with the Gaussian kernel loss results in a 31.31% increase over the ImageNet initialisation.

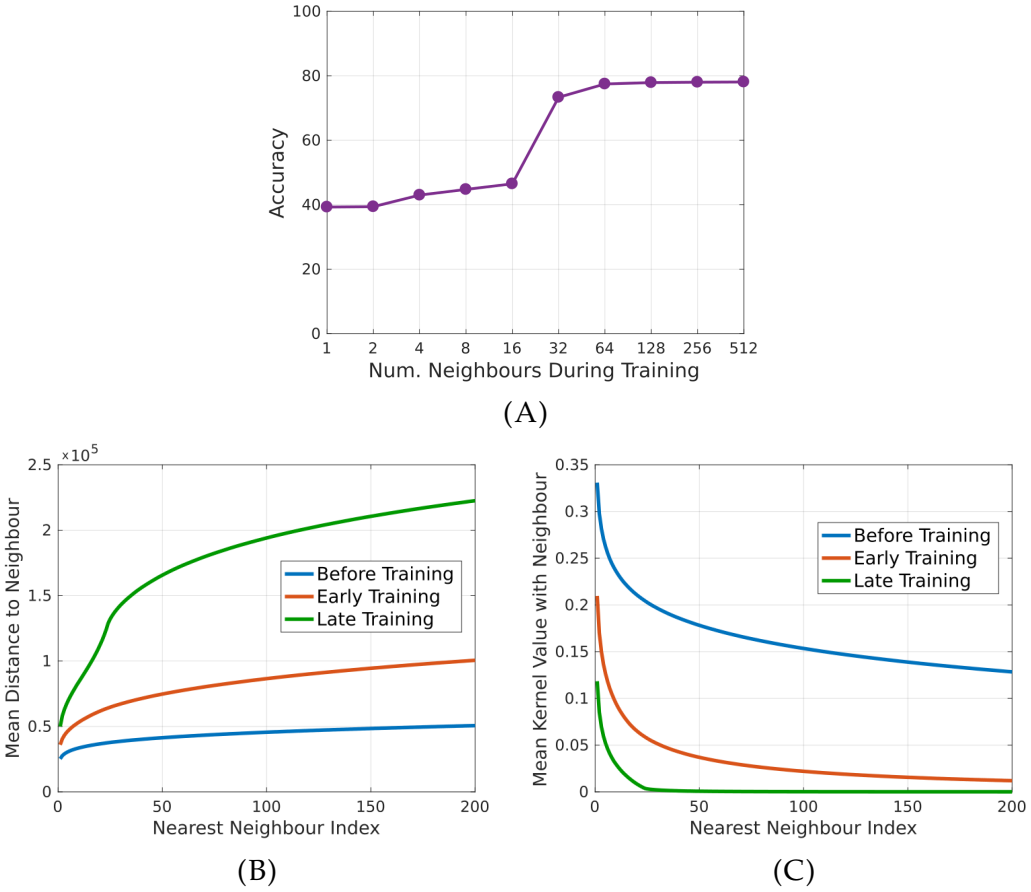


FIGURE 4.17: (A) The effect of the number of nearest neighbours considered while training the network. (B) The average distance from training examples to their nearest Gaussian kernel centres, at different points during training. (C) The Gaussian kernel value (Equation 4.3) between training examples and their nearest kernel centres, at different points during training.

4.10.2.6 Neighbourhood Size Analysis

Figure 4.17A shows the impact of the neighbourhood size used during training, in terms of the classification accuracy achieved. The neighbourhood size is the number of approximate nearest neighbours considered for each training example. The training neighbourhood size is also used for testing. There is a clear lower bound required for good performance. Below a neighbourhood size of 32, the maximum classification accuracy achieved is approximately 46%. At 32 nearest neighbours, the accuracy jumps to approximately 73%, with performance plateauing at around 78%. The reason for this lower bound is because, as discussed in Section 4.7, the network weights are constantly being updated, resulting in the true centre locations moving at each

training iteration. However, the stored Gaussian centres are not updated every time the network weights are updated. As such, a larger neighbourhood needs to be considered, compared to if the centres were always up-to-date.

The local structure of the feature embedding space around training examples and how it changes throughout training is shown in Figures 4.17B and 4.17C. The VGG16 network used for these experiments was trained with a neighbourhood size of 200. Figure 4.17B shows the Euclidean distance from each training example to its 200 nearest Gaussian kernel centres, at different points during training. The distances are averaged over all training examples. The horizontal axis is the ordered nearest neighbour index. That is, the first point on the horizontal axis corresponds to the nearest neighbour and the last point corresponds to the 200th nearest neighbour. Similarly, Figure 4.17C shows the average Gaussian kernel function value (from Equation 4.3) between training examples and their 200 nearest Gaussian centres. As training progresses, the feature embedding space expands and clusters begin to form. This can be clearly seen in the *Late Training* lines on both plots. Interestingly, the Gaussian kernel function value begins to decay to zero at around the 30th nearest neighbour. This lines up with the minimum neighbourhood size required to achieve good classification performance in Figure 4.17A.

4.10.2.7 Implicit Attribute Learning

The proposed Gaussian kernel approach allows clusters to position themselves freely in the feature embedding space, such that the intrinsic structure of the data can be represented. As a result, it is expected that the feature embeddings will be co-located based not only in terms of class, but also in terms of more fine-grained information, such as attributes. We use the 312 binary attributes of Birds200 to confirm this expectation. For each 4096-dimension VGG16 test set embedding, attributes are propagated by computing the density of each attribute label present in the neighbouring test feature embeddings. This is done using Gaussian kernel functions, treating each attribute as a binary classification problem. The best Gaussian σ is found for softmax and Gaussian kernel learned embeddings separately. A precision and recall curve, shown in Figure 4.18, is generated by sweeping the classification discrimination threshold from zero to one. The bottleneck FC7 layer is used as the embedding layer for both the softmax approach and the proposed approach.

TABLE 4.6: Attribute Area Under Precision-Recall curve (AUPR) on the 312 binary attributes of Birds200.

Method	AUPR
Softmax	0.5455
Ours	0.5945

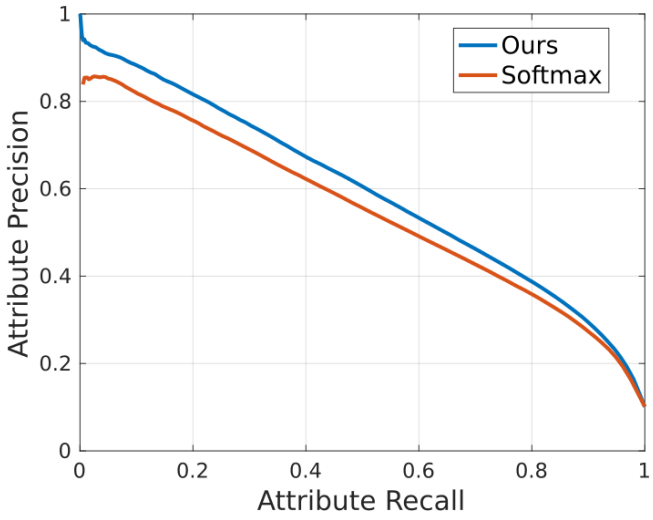


FIGURE 4.18: Attribute precision and recall on the 312 binary attributes of Birds200. The attributes are propagated from neighbouring test embeddings and the curves are generated by sweeping the classification discrimination threshold. The ideal value of σ is found for the Gaussian kernel and softmax approaches separately. No training was carried out on the attribute labels.

At a given precision, the Gaussian kernel approach results in a feature embedding space with better attribute recall than softmax. Note that the networks are not trained using the attribute labels. The Area Under Precision Recall curve (AUPR) of the two approaches is shown in Table 4.6. This measure is threshold independent, allowing approaches to be compared using a single number. These results show that compared to a softmax network, the proposed Gaussian kernel approach learns a feature embedding space that allows better representation of the fine-grained semantic information contained within the data.

4.11 Discussion and Conclusion

In this chapter, we proposed a nearest neighbour Gaussian kernel metric learning approach that can be directly applied to both feature embedding learning problems and classification problems. Our approach is trained in the same manner, regardless of whether the primary goal is classification or feature embedding learning problems, such as knowledge transfer. This means that a model trained for classification will also learn feature embeddings that are suitable for transfer learning. Likewise, a model trained for feature embedding learning problems will result in feature embeddings that can be well classified by the Gaussian kernel classifier. A classification model that also produces a well formed feature embedding space is particularly useful in robotic applications. A robot operating in a dynamic environment will almost certainly observe objects that are outside of the training set distribution. A model that transfers well to out-of-distribution observations allows reasonable inference to be made about previously unknown semantic classes.

Experimental results show that the proposed approach outperforms existing state-of-the-art deep metric learning approaches on feature embedding learning problems. The Gaussian kernel approach also outperforms a conventional softmax-based convolutional neural network on four experimented classification datasets. Further, results show that our approach learns feature embeddings that better capture the fine-grained semantic information contained within images, compared to a softmax network. The importance of dimensionality in the feature embedding space was also investigated. Experimental results show that in addition to learning better compact feature embeddings than a triplet-based metric learning model, our approach can also take advantage a larger, higher-dimensional space. Training of our method is made feasible by the introduction of periodic asynchronous updates of the Gaussian centres and the approach is made scalable through the use of fast approximate nearest neighbour search.

Chapter 5

Open Set Recognition and Active Learning of Novel Classes

State-of-the-art deep neural network recognition systems are designed for a static and closed world. It is usually assumed that the distribution at test time will be the same as the distribution during training. As a result, classifiers are forced to categorise observations into one out of a set of predefined semantic classes. Robotic problems are dynamic and open world; a robot will likely observe objects that are from outside of the training set distribution. Classifier outputs in robotic applications can lead to real-world robotic action and as such, a practical recognition system should not silently fail by confidently misclassifying novel observations. In this chapter, we show how a deep metric learning classification system can be applied to such open set recognition problems, allowing the classifier to label novel observations as unknown. Further to detecting novel examples, we propose an open set active learning approach that allows a robot to efficiently query a user about unknown observations. Our approach enables a robot to improve its understanding of the true distribution of data in the environment, from a small number of label queries. Experimental results show that the proposed approach significantly outperforms comparable methods in both the open set recognition and active learning problems.

5.1 Motivation

Robotic applications demand special considerations when designing a visual recognition system. The open set nature of robotics means that a robot will encounter observations belonging to novel, out-of-distribution classes. Any classifier prediction in a robotic environment can trigger some sort of costly

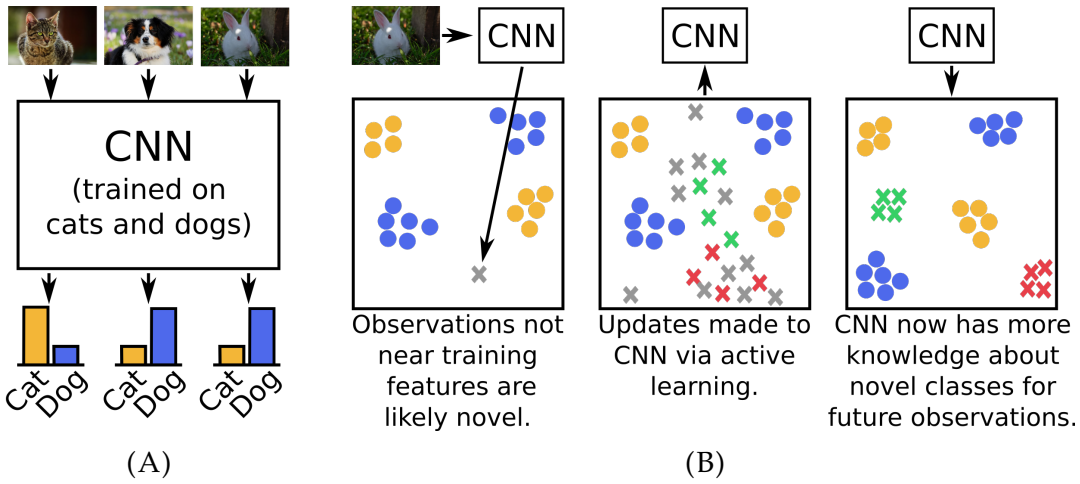


FIGURE 5.1: (A) Motivation for our approach. Conventional classifiers will silently fail when observing a novel example. (B) Overview of our approach.

robotic action. As such, a recognition system must not silently fail when observing a novel example by incorrectly predicting a label from the training set distribution, as shown in Figure 5.1A. Additionally, a robotic vision system should not cease learning after the initial training phase. The distribution of data in the training set will undoubtedly vary from the true distribution of data in the robot's operating environment. By sampling data from the environment and interactively querying a human user about novel observations, a robotic vision system can continue to improve its understanding of the real-world data distribution, as shown in Figure 5.1B.

Detecting out-of-distribution observations is known as novelty detection [135] and the problem of both classifying in-distribution observations with the correct class label and detecting novel examples is known as open set recognition [147]. The task of interactively querying a user for labels is referred to as active learning [159]. If a recognition system can select the most informative observations for labelling, the model can efficiently learn from a small number of labelled examples. This is important for robotics, as the number of observations may be very large but the labelling budget is likely to be small. In this chapter, we focus on the active learning of novel classes, also referred to as open set active learning. A model trained on known classes is deployed in an environment containing both known and novel classes. The active learning algorithm aims to learn about the novel class distribution from as few human-labelled examples as possible.

Conventional classification approaches, such as convolutional neural networks (CNNs) with softmax classifiers [60, 207, 66, 69], are closed set by design. As such, these commonly used methods are limited in their ability to detect novel examples. Softmax-based CNNs are not only forced to predict a known label for out-of-distribution examples, but often do so with high confidence [145]. Approaches that are limited in their ability to detect novel examples, are also limited in their ability to learn from and improve their understanding of the corresponding unknown classes. As a result, these conventional softmax-based approaches are not suitable for open set robotic vision problems.

Deep metric learning algorithms learn a transformation from the image space to a feature embedding space, in which distance is a measure of semantic similarity. State-of-the-art deep metric learning models demonstrate an impressive aptitude for transfer learning [18, 19, 20, 23, 22, 21], meaning that features are likely to be co-located based on class, even when those classes are outside of the training distribution. This not only allows for the reliable detection of novel examples, but also provides a meaningful way of determining an observation's informativeness of the true class distribution. This knowledge enables efficient querying in an active learning setting, allowing the model to learn about novel classes from a small number of labelled examples.

An overview of our proposed approach is shown in Figure 5.1 and the contributions of the work are enumerated in Section 5.2. The proposed approach is a deep metric learning method for the open set problems of novelty detection and active learning of novel classes. Experimental results show that our proposed approach outperforms comparable methods in both problems. The metric learning-based approach is further motivated from the perspective of novelty detection and open set recognition in Section 5.1.1 and from the perspective of novel class active learning in Section 5.1.2.

5.1.1 Open Set Recognition

The converse of open set recognition is closed set recognition. Most work in the field of image classification is in this domain, as it is generally assumed that the distribution at test time will be the same as during training. The majority of classifiers are closed set by design; a classifier is forced to make a prediction for every observation by classifying into a set of predefined training

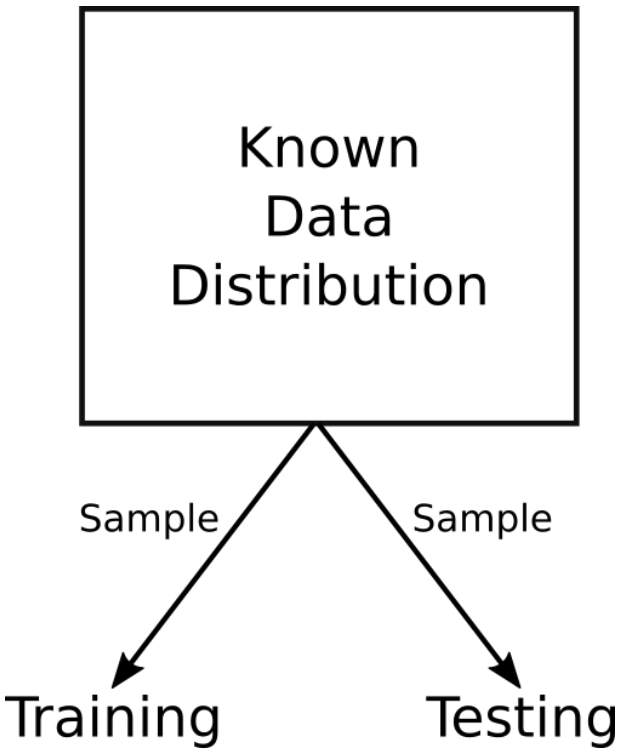


FIGURE 5.2: Closed set problems. During testing and deployment, the trained model assumes all data is sampled from only the distribution of known classes.

classes. This is illustrated in Figure 5.2, wherein the same data distribution, $\mathcal{K}$, is sampled from during training and testing.

The scenario depicted in Figure 5.2 is not representative of many real-world vision applications. This is particularly true of robotic vision, which is, by its very nature, an open set problem. It is unreasonable to expect that a training set could include all possible semantic categories that a robot in an unconstrained and dynamic world may observe. A training set should be the best representation of the true distribution of data in the environment that is possible, given the available prior knowledge and budget of human effort allotted to the task of gathering and labelling training data. It should then be assumed the true distribution varies in some way from the known training distribution $\mathcal{K}$.

An open set recognition setting is illustrated in Figure 5.3. In this scenario, the model is again trained on the distribution of known classes $\mathcal{K}$. Unlike in the closed set problem, the classifier samples data from both the known distribution and some unknown distribution during model testing or deployment. The unknown novel distribution $\mathcal{N}$ contains observations from the environment that belong to semantic classes not represented in the training

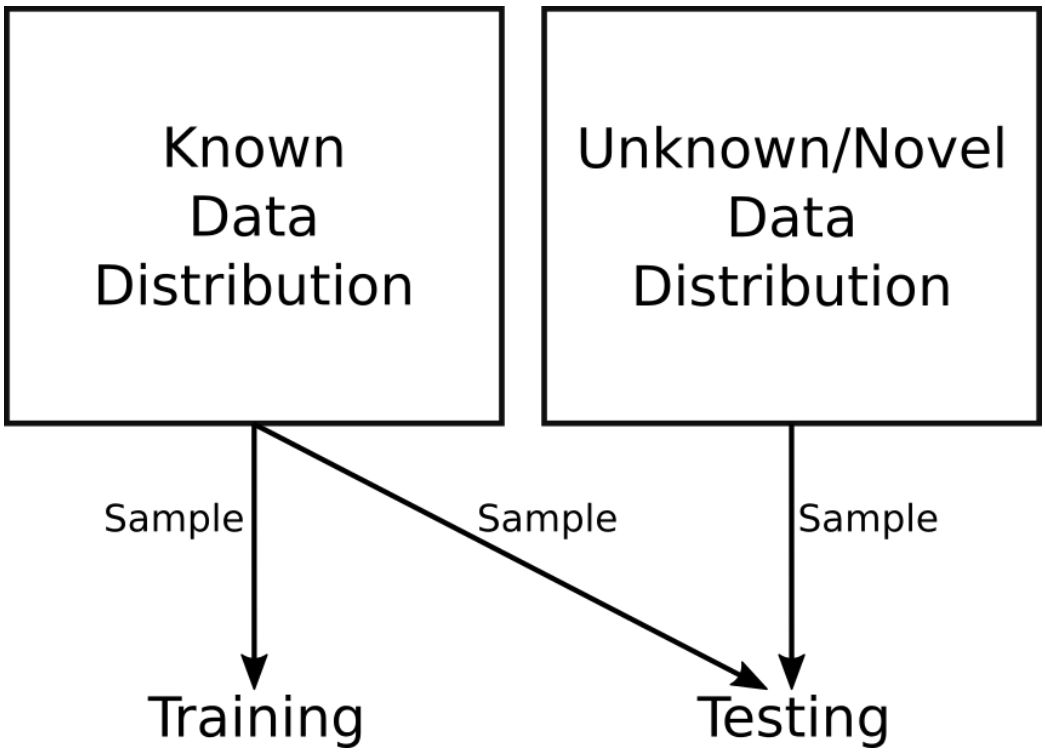


FIGURE 5.3: Open set problems. During testing and deployment, the trained model observes a mixture of data from the distribution of known classes and some distribution of unknown, novel classes.

set. This is the most likely setting in which a robotic vision system will operate. A robot will observe semantic entities belonging to a mixture of known and novel semantic categories. When a model that is designed only for closed set problems is deployed in a open set world, observations belonging to the unknown distribution $\mathcal{N}$ will be incorrectly classified as one of the classes in the known distribution $\mathcal{K}$. As such, closed set classifiers are inappropriate for many robotic applications.

A classifier that is able to perform open set recognition is vital to robotic vision. The predictions that are made by a classifier are not the end goal in robotic applications. Predictions are made to enable some sort of inference with the environment, such as interacting with certain objects, avoiding other objects and navigating the scene. In other words, classifier outputs in robotic applications can trigger costly robotic action. This action may be either physical or computational. As such, the observation and misclassification of novel objects by a closed set classifier does not allow a robot to carry out tasks safely or accurately. To avoid such ill-advised robotic action, an open set classifier is required. This allows a robot to "know when it doesn't

know", rather than silently failing and making confident mistakes by incorrectly classifying novel observations as known.

Open set recognition is included in a range of terms inconsistently used to described related problems. These terms also include novelty detection, anomaly detection, outlier detection and out-of-distribution detection. The definitions of these terms as they are used in this chapter, are described below:

- **Novelty Detection** is the process of detecting when an observed example belongs to a class that is not represented in the training set. Such an example is referred to as novel because the model was not aware of any such class at training time.
- **Anomaly and Outlier Detection** are related to novelty detection, but focus more on the detection of examples that are unusual, relative to some "regular" distribution. This may be in the form of detecting fraudulent financial transactions or malicious web activity.
- **Out-of-Distribution Detection** refers to the detection of any examples that are outside of some training or regular distribution. This is a catch-all term that covers novelty detection, anomaly detection and outlier detection.
- **Open Set Recognition** is the partial goal of the work proposed in this chapter. It is the process of simultaneously classifying observed known examples into the correct class and detecting observed novel examples as unknown. This is shown in Figure 5.4.

5.1.1.1 The Problem with Softmax

Convolutional neural networks with softmax classifiers are closed set by design. The final fully connected layer outputs an activation vector with the number of channels equal to the number of known classes, which is then passed into the softmax layer. The probability distribution over known classes must sum to a value of one. A softmax classifier is forced to make a prediction for every input, by categorising examples into one out of the set of known training classes.

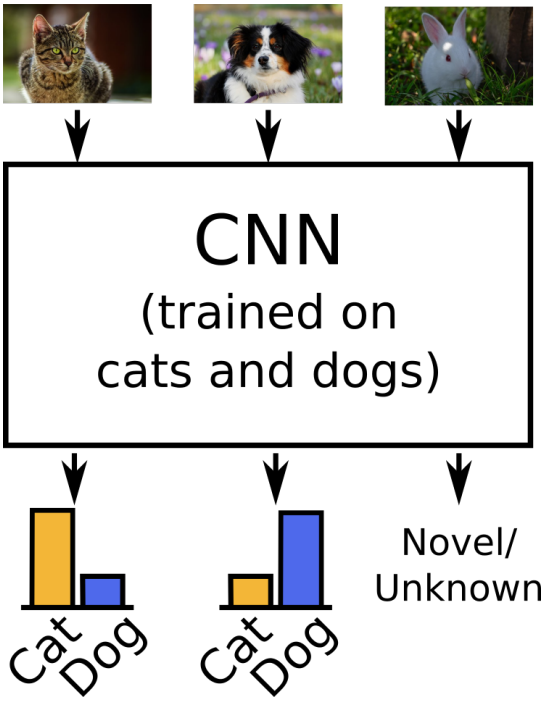


FIGURE 5.4: Open set classifier. Examples from novel classes should be flagged as such, rather than the classifier silently failing by incorrectly classifying the example as from a known class.

A simple novelty detector for softmax-based convolutional neural networks would be to threshold on the softmax certainty [145]. That is, take the maximum softmax probability output by the network, which should represent the certainty/confidence with which the classifier is making its prediction, and if it is too low, reject the input. One would assume that novel examples will be classified as a known class with low certainty, while the classifier will make confident predictions for known examples. However, softmax classifiers tend to make confident predictions for most inputs, regardless of whether the predictions are correct or incorrect [145]. Since the classifier makes mistakes with high confidence, it is difficult to distinguish between known and novel examples by analysing the softmax certainty.

Softmax confidence is a poor measure of the likelihood of a prediction being correct because deep convolutional neural networks are poorly calibrated [146]. A classifier with perfect confidence calibration would output a confidence score that is exactly equal to the probability of that prediction being correct. For example, a perfectly calibrated softmax classifier would be correct 50% of the time when classifying an example with a softmax score of 0.5. As convolutional neural networks have poor confidence calibration, the softmax distribution does not capture the true probability distribution over the

universe of examples. This property of deep softmax-based convolutional neural networks is shown in the work by Guo *et al.* [146]. Analysis shows that on a 100 class classification dataset, a ResNet model has an accuracy of around 70%, but an average confidence of around 90%. In addition to showing that the probability of a prediction being correct does not correspond to the confidence score, the work shows that the probability of a prediction being correct does not necessarily even increase monotonically with the confidence score. Note that the poor calibration is not the result of the softmax function itself, but rather the class activation feature vectors that are extracted from the final layer of a neural network that is trained with cross-entropy loss on a softmax classifier.

5.1.1.2 Opening Softmax

There has been some work focused on making networks with softmax classifiers more appropriate for open set problems [148, 147]. Liang *et al.* [148] introduce *ODIN*, which aims to improve the calibration of softmax networks by performing input pre-processing and softmax temperature scaling. The input pre-processing involves making small perturbations to the input image along the direction of gradients of the softmax confidence. The authors find that these small perturbations generally result in greater change in the softmax space for in-distribution examples compared to out-of-distribution examples. Softmax temperature scaling involves scaling the activations from the final fully connected layer by some constant before performing softmax normalisation. Careful selection of the scaling constant, known as the temperature, results in better confidence calibration [146]. These two techniques push the softmax confidence scores of in-distribution and out-of-distribution examples further apart, allowing for easier detection of novelty by considering the softmax confidence.

Bendale and Boult [147] propose an open set version of softmax known as *OpenMax*. This work introduces a pseudo-activation to estimate the confidence score relating to the notion of novelty. The probability of an example belonging to an unknown class is estimated by fitting per-class models to the activation vectors of the training data. These models are used to refine the raw per class activations into an activation vector that includes a channel for novel examples. The refined activation vector can be normalised in the normal softmax manner to produce an OpenMax distribution. In addition

to directly estimating the probability of an example being novel, the novelty detector also thresholds on the known class confidence scores. We compare the novelty detection and open set recognition performance of our proposed approach to these open versions of softmax in Section 5.7.2.

5.1.1.3 Metric Learning for Open Set Recognition

In Chapter 4, a deep metric learning approach was introduced that not only transfers well to novel classes, but also has good classification performance. A distance metric that transfers to previously unknown classes allows for a meaningful measure of novelty by considering some sort of distance measure between examples in the feature embedding space. The approach from Chapter 4 stores all training set feature embeddings for the purpose of classification. This allows direct measurement of the semantic similarity between observed examples and all training set examples from known classes. In addition to measuring the similarity of observations to training set examples, a metric space also allows the similarity between observed novel examples to be measured. For example, it is possible for a metric learning model to recognise when it is observing a novel object that is similar to a previously observed novel object. This provides the model with information about the distribution of novel classes in the environment, enabling efficient learning of these classes. This active learning of novel classes is discussed further in Section 5.1.2.

In this chapter, we show the importance of deep metric learning to robotic vision through the application of open set recognition. Novelty measures for metric spaces are investigated and evaluated. Experimental results show that the deep metric learning approach to novelty detection and open set recognition significantly outperforms the softmax baseline, as well as the open set variants of softmax discussed in Section 5.1.1.2. Importantly, the proposed metric learning model performs well at both detecting novel examples and classifying known examples into the correct semantic category.

5.1.2 Active Learning of Novel Classes

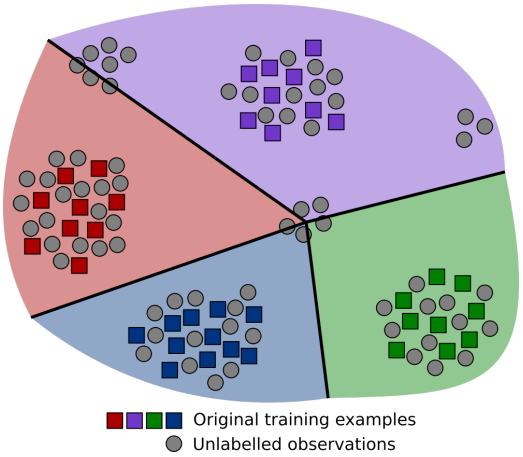
As an open set robotic vision system observes objects in the environment, known objects will be classified into the correct class and novel objects will be flagged as such. The learning phase of a robotic vision system should not

end after the initial training of the model on the training data. A robot that is able to recognise when it is observing novel examples is also able to learn from such examples. This should improve the robot's understanding of the environment and reduce the discrepancy between the known data distribution and the true distribution in the robot's operating environment.

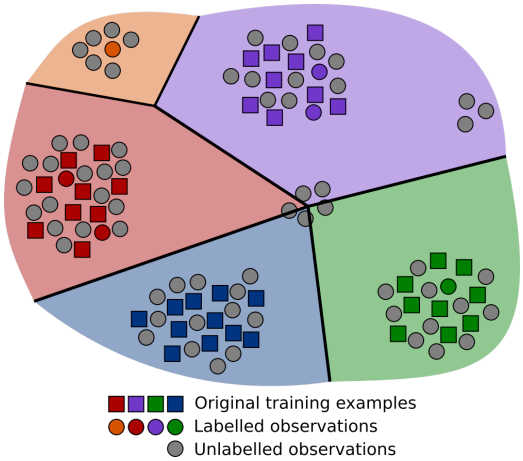
In order to update the model parameters to better represent the true data distribution in a supervised manner, semantic labels must be attached to the novel observations. Such labels could be provided by a human user. The potential number of observations made by a robot is unbounded. As such, it is not reasonable or feasible to expect that labels can be attached to all novel examples that a robot may observe. The best observations should be selected by the model for labelling, adhering to some user labelling budget. In this case "best" refers to the examples which, when labelled, will provide the most information to the system about the true distribution of data and allow the system to make effective and efficient updates to the model parameters.

This problem is known as active learning. It is so named because the algorithm is actively querying a user or oracle for labels, rather than passively receiving a set of labelled examples. Active learning works on the premise that not all data is created equal. Given the current state of the model, some examples will be more informative of the true data distribution than others. In terms of a feature embedding space, an example is generally more informative if it is far from other labelled examples. In addition, an unlabelled example that is surrounded by many other unlabelled examples is more informative than a lone unlabelled example. It is the job of the active learning algorithm to select the examples that will allow the model to learn from as few label queries as possible. In our case, we are concerned with the active learning of novel classes that are outside of the original training set distribution.

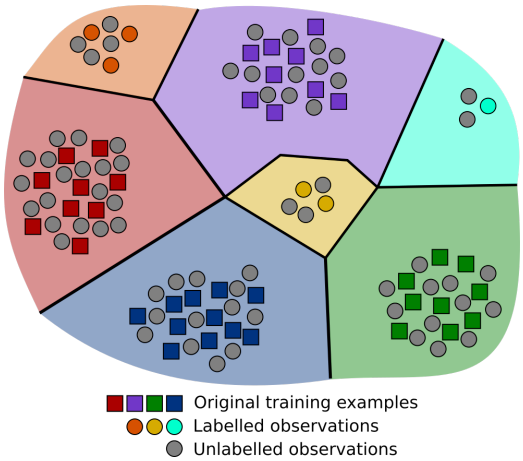
The importance of novel class active learning and careful query selection is shown in Figure 5.5. The example shows an embedding space containing original training data from four different classes and observed data, which contains examples from both the known four classes and additional novel classes. In this example, labels can be obtained for six of the observations. Given this limited labelling budget, care must be taken to select the most informative examples for labelling. Random query selection (Figure 5.5B) fails to discover all novel classes observed, as the majority of the labelling budget is spent acquiring labels for already well learned classes. Careful



(A) Example space.



(B) Random query selection.



(C) Careful query selection.

FIGURE 5.5: The importance of novel class active learning and careful query selection. The colour of examples denotes the class (with grey representing unlabelled observations), black lines represent decision boundaries and the shaded colours represent the class to which new examples will be classified in that region of the space. Note that the decision boundaries are for illustrative purposes only; our approach performs classification by summing the influence of nearby labelled examples in the feature embedding space. The original training data contains four classes and the observed data contains an additional three novel classes. A labelling budget of six is shown.

query selection with an intelligent querying scheme that selects informative examples for labelling (Figure 5.5C) allows for all three novel classes to be discovered.

5.1.2.1 The Problem with Softmax (Again)

As discussed in Section 5.1.1.1, softmax-based networks are limited in their ability to detect novel examples. Consequently, softmax networks are also limited in their ability to learn from and improve their understanding of observed novel classes. Since the classifiers are poorly calibrated, softmax confidence scores are not a good measure of an observed example's potential informativeness to the model. Further, the informativeness of an observation with regards to an entire set of unlabelled observations is difficult to measure since softmax networks do not learn a distance metric. The lack of a meaningful way to measure the similarity between observed novel examples means there is no clear way to select the observation that best represents the entire set of unlabelled observations.

Those approaches that improve the ability of softmax classifiers to discern novel observations [148, 147] still suffer from these problems. Although the ODIN out-of-distribution detector proposed in [148] drives the confidence scores of in-distribution and out-of-distribution examples further apart, the lack of a learned distance metric between observations means that the ability of such an approach to select the most informative observations for labelling is limited. The open set version of softmax, OpenMax [147], does attempt to measure the similarity of observations to training examples of each known class. This is done by fitting a per class model on the training data. However, this does not allow for similarity measurements between different observations and therefore, does not enable the selection of the most informative observations. Further, as the model does not learn a distance metric, there is no guarantee that distances between activation vectors are an effective measure of semantic similarity.

5.1.2.2 Active Learning with Metric Spaces

A distance metric over images allows for direct measurement of the similarity of semantic entities. An observation can be compared to examples in the training set, as well as unlabelled observations, including those from known classes and unknown classes. As a result, metric learning models are a more suitable approach to the active learning of novel classes than conventional softmax-based networks. The learned distance metric provides a meaningful way for the active learning algorithm to select the most informative observations. This is because the metric can measure how similar an observation is to

other labelled examples, including the original training data and previously labelled observations, in order to determine the novelty of the observation. It can also measure the similarity of an observation to other unlabelled observations, allowing the algorithm to select an example that is representative of many unlabelled novel observations.

The ability to select the most informative examples for labelling is not the only requirement for an effective active learning model. Since the goal of novel class active learning is to improve a model's ability to make predictions about observations by enabling a larger range of semantic classes to be categorised, the approach must also be suitable for classification. As discussed throughout Chapter 4, many state-of-the-art deep metric learning algorithms have poor classification performance compared to a conventional network with a softmax classifier. This eliminates the majority of deep metric learning approaches as candidates for this problem. The Gaussian kernel metric learning approach proposed in Chapter 4, however, does meet the criteria for this problem. Experimental results in Section 4.10 show that the Gaussian kernel approach learns a distance metric that transfers well to novel classes, enabling effective selection of informative observations for labelling. Importantly, the results also show that the proposed approach learns feature embeddings that are amenable to classification by the Gaussian kernel classifier, with the approach outperforming softmax networks on several datasets.

5.1.2.3 Informativeness Measures

A well structured feature embedding space provides a means to effectively measure the informativeness of observations. However, constructing an informativeness measure in order to select the best observations for labelling, is non-trivial. A common approach is the *furthest nearest neighbour* (FNN) informativeness measure [216]. This method selects the next observation for querying based on the distance to the nearest labelled example. The observation that is furthest from a labelled example is chosen for querying. The assumption is that the most informative example is that which is the most semantically dissimilar to all labelled examples. Although a reasonable measure of an observation's informativeness in terms of novelty, the measure does not consider the distribution of other unlabelled examples in the feature embedding space. An lone observation is not necessarily informative of the true distribution of data in the environment, as it may just be outlier.

Even if the observation is not an outlier, those observations that are representative of more common novel entities should be selected first. Such an observation would be found in regions of high unlabelled density in the feature embedding space.

Another approach to measuring the informativeness of an observation is kernel density estimation. With labelled feature embeddings as kernel centres, a per class density can be computed. This is akin to the per class classification confidence scores. The observation with the smallest maximum class density is selected for querying. The intuition is that observations with a high maximum confidence score are not informative, since they are in high density regions that are largely homogeneous in terms of class. However, observations that are located in contentious regions, with labelled examples belonging to more than one class, are treated as informative. Observations that are far from any labelled examples are also considered informative. As with the furthest nearest neighbour approach, kernel density estimation does not consider an observation's similarity to other unlabelled observations. This means that the informativeness measure fails to consider the degree to which an observation is representative of the entire set of unlabelled examples.

In this chapter, we propose a novel informativeness measure to be used with deep metric spaces. The proposed approach is a density-based measure that considers the structure of both labelled and unlabelled examples in the feature embedding space. The informativeness measure is simple to compute and fuses well with the Gaussian kernel classifier discussed in Chapter 4, which already computes the Gaussian distances between an observation and labelled examples for the purpose of classification. By considering both labelled and unlabelled density, the approach is able to measure informativeness both in terms of novelty and in terms of how representative an observation is of the unknown true distribution.

5.2 Contributions

In this chapter, the importance of deep metric learning to open set problems is investigated. The main contributions are as follows:

- We present a deep metric learning approach to novelty detection and open set recognition, exploring several distance-based novelty measures (Section 5.5).

- We show that our proposed approach to novelty detection and open set recognition outperforms conventional CNNs and purpose built novelty detectors (Section 5.7.2).
- We present an open set active learning approach for metric spaces using our proposed query selection method, unlabelled to labelled density ratio, which allows efficient learning of observed novel classes (Section 5.6).
- We show that our proposed approach to active learning significantly outperforms comparable methods at small labelling budgets (Section 5.7.3).
- For a labelling budget of zero, we investigate if the representation of observed novel classes can be improved using unsupervised pseudo-labels (Section 5.7.3.7).

The major contributions of this chapter have been presented in a peer-reviewed publication [25]. Further analysis that is not found in the publication includes a portion of the experimental results in Section 5.7.2.4 (Figure 5.8) and Section 5.7.3.4 (Figures 5.10-5.12), as well as the visualisations in Section 5.7.3.5.

5.3 Related Work

In this section, a brief overview of novelty detection, open set recognition and active learning literature is given. A detailed discussion of previous work in these fields can be found in Section 2.5.

5.3.1 Novelty Detection and Open Set Recognition

An in-depth review of classical novelty detection approaches can be found in the survey by Pimentel *et al.* [135]. Common methods include *probabilistic approaches* [127, 128] that estimate the probability density function of the data, *distance approaches* [136, 137] that assume novel examples are located far from known examples, *domain approaches* [124, 140] that treat the task as a binary classification problem and *information-theoretic approaches* [143, 144] that analyse the information content of data. Our metric learning method is in the *distance approaches* category. Novelty detection is related to the task of anomaly and outlier detection [217].

Recent works that make use of deep CNNs include a generative adversarial network approach [151], in which a multi-class discriminator is trained with a generator that creates data from both known and novel distributions. Mandelbaum and Weinshall [149] propose a density-based confidence score that can be applied to novelty detection, as an alternative to confidence scores based on softmax probabilities [145].

Bendale and Boult [147] propose an open set version of a softmax classifier named OpenMax. Class probabilities are revised using a meta-recognition Weibull model fitted on distances between activation vectors and per-class mean activation vectors. A pseudo-class representing unknown classes is introduced, allowing direct measurement of novelty.

Liang *et al.* [148] introduce an out-of-distribution detector called ODIN that operates on pre-trained softmax-based networks. The authors use softmax temperature scaling and input pre-processing to push softmax scores from known and novel classes further apart. This method requires a forward pass, backward pass and second forward pass through the network to perform novelty detection.

A contrastive loss metric learning approach is proposed in [150]. However, the classification performance is poor, making it unsuitable for direct use in open set recognition. Further, the work proposed in [150], along with [218], requires out-of-distribution examples during training.

5.3.2 Active Learning

Classic methods of active learning include uncertainty approaches [158, 161, 219] and decision-theoretic approaches [163, 164, 220]. A comprehensive review of these methods can be found in Settles' survey [159]. Recent works have investigated active learning with CNNs [165, 166, 167, 168, 169]. These approaches include framing active learning as a reinforcement learning problem [168], generative adversarial active learning [167] and a core-set based approach [169]. These methods aim to select a subset of examples for labelling that best represent the entire set of unlabelled examples, for the purpose of initial network training. Our method is focused on learning from observed novel classes that are not present in the existing training set. This means that rather than selecting a subset that best represents the entire observed unlabelled set, we want to select a subset that best represents the

novel classes in the unlabelled set. In other words, we don't want to waste our limited labelling budget acquiring labels for classes that are already well learned.

5.4 Notation and Background

We first introduce the notation used for the remainder of this chapter. The term *observation* is used to refer to an example presented to the system at test time. This is representative of a robotic system observing some object in the environment. For a given observation with true label y, a β-dimensional feature embedding $\mathbf{x} = [x^{(1)}, \ldots, x^{(\beta)}]^T$ is extracted. The feature embedding is defined in a metric space $\mathcal{M}$, in which the Euclidean distance between feature embeddings is a measure of the semantic similarity between the associated images. The set of α labelled training feature embeddings is denoted as $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_\alpha\}$, with $\mathbf{c}_i = [c_i^{(1)}, \ldots, c_i^{(\beta)}]^T$ corresponding to the feature embedding vector of the i-th labelled training image.

The feature embeddings are computed by learning a transformation from the image space $\mathcal{I}$ into the metric space (or feature embedding space) $\mathcal{M}$. Practically, the transformation is performed by a convolutional neural network. The transformation function learned and performed by the network is denoted as g. That is, the convolutional neural network transforms an example image $\mathbf{I} \in \mathcal{I}$ into a feature embedding $\mathbf{x}$ as:

$$\mathbf{x} = g(\mathbf{I}; \boldsymbol{\theta}), \tag{5.1}$$

where $\boldsymbol{\theta}$ denotes the neural network parameters.

5.4.1 Deep Metric Learning Model

In this section, a brief review of deep metric learning and the model used for our proposed approach is presented. For a detailed discussion on metric learning, refer to Chapter 4. Deep metric learning refers to metric learning approaches that make use of deep convolutional neural networks. Unlike conventional classification models, such as a CNN with a softmax classifier, metric learning algorithms aim to learn a transformation from the image space to a feature embedding space $\mathcal{M}$, in which distance is a measure

of semantic similarity. Deep metric learning approaches learn feature embeddings that are amenable to transfer learning [18, 19, 20, 23, 22, 21]. This suggests that such models are suitable for detecting novel examples.

The deep metric learning approach we use in this chapter is described in detail in Chapter 4. A brief recap of the approach is presented in this section. A Gaussian kernel, or radial basis function, is centred on each training feature embedding. The probability that example $\mathbf{x}$ has class label ℓ is computed as:

$$\Pr(y = \ell \mid \mathbf{x}) = \frac{\sum_{j \in C} \exp\left(\frac{-|\mathbf{x}-\mathbf{c}_j|^2}{2\sigma^2}\right) [\mathbf{c}_j \in \text{class } \ell] \, [\mathbf{c}_j \in S]}{\sum_{k \in C} \exp\left(\frac{-|\mathbf{x}-\mathbf{c}_k|^2}{2\sigma^2}\right) [\mathbf{c}_k \in S]}, \tag{5.2}$$

where σ is a shared Gaussian scale term and $S \subseteq C$ is the set of training nearest neighbours for example $\mathbf{x}$. The Iverson brackets $[condition]$ evaluate to values of 1 if $condition$ is true and a value of 0 otherwise.

During training the Gaussian kernels pull examples of the same class together and push examples of different classes apart. The loss for a given training example is the negative logarithm of true class probability. The approach is made scalable to large numbers of classes and examples through the use of fast approximate nearest neighbour search. Training is made feasible and efficient by periodic asynchronous updates of the training embeddings and nearest neighbours, negating the need to do so after every network update.

Although many deep metric learning approaches perform well on transfer learning tasks, the feature embeddings learned from commonly used triplet approaches are not well suited to classification [116]. In contrast, the Gaussian kernel approach performs well for transfer learning as well as classification, outperforming softmax classification on several datasets. This makes the model suitable for our open set recognition and active learning setting.

5.5 Novelty Detection and Open Set Recognition

We now describe the problem of detecting out-of-distribution observations. In this context, *out-of-distribution* refers to examples belonging to semantic classes that are not represented in the training set or a set of known labelled examples. Let $\mathcal{K}$ denote the distribution of the known training data and $\mathcal{N}$ denote the distribution of unknown data that is outside of $\mathcal{K}$. Our open set

recognition system should determine whether an observation $\mathbf{x}$ is from the known distribution $\mathcal{K}$ or the unknown distribution $\mathcal{N}$. If $\mathbf{x}$ is from $\mathcal{K}$, the classifier should predict a known class label $\hat{y} \in \mathcal{K}$ as normal. If $\mathbf{x}$ is deemed to be from outside of the known distribution, the observation should be labelled as unknown/novel.

5.5.1 Rationale

The deep metric learning model used by our approach stores all training set feature embeddings C and computes the Euclidean distance between an example embedding and its set of nearest neighbours for the purpose of classification. Distance between examples in the metric space can be used as a measure of semantic similarity. We expect that most observed examples from a known class will be located nearby training set feature embeddings of the same class. Observed examples that are not located nearby any training set embeddings are novel to the model and are likely from the unknown distribution $\mathcal{N}$. Since it is known that the metric learning model transfers well to novel classes, we expect the model to be well suited to novelty detection. This assumption is evaluated experimentally in Section 5.7.2.

5.5.2 Open Set Classifier

Our open set classifier and novelty detector predicts a class label $\hat{y}$ for example $\mathbf{x}$ as follows:

$$\hat{y} = \begin{cases} \arg\max_{\ell} \Pr(y = \ell \mid \mathbf{x}), & \text{if } n(\mathbf{x}) \leq \delta, \\ \text{unknown/novel}, & \text{if } n(\mathbf{x}) > \delta, \end{cases} \tag{5.3}$$

where n is a novelty function and δ is a threshold.

5.5.3 Novelty Measures

The novelty measure $n(\mathbf{x})$ from Equation 5.3 is a function that for an observation $\mathbf{x}$, returns a score that is proportional to the novelty of the observed

example. A low novelty score corresponds to an observation that is semantically similar to known labelled examples. Such an observation will be classified with high confidence into a known class. A high novelty score corresponds to an observation that is semantically dissimilar to all known labelled examples. Such an example is likely outside of the training set distribution and from an unknown, novel semantic category.

We investigate the effectiveness of several simple distance-based novelty measures for use in a deep metric space. Since the metric learned by the convolutional neural network g transfers to novel classes, distance in the metric space is an appropriate measure of novelty. The presented novelty measures are compared with softmax confidence-based novelty detection and purpose built novelty detectors in Section 5.7.2. The novelty measures for metric spaces that are investigated and evaluated are described below.

1) Nearest neighbour distance (NN dist.) is the distance between an observed example and its nearest training set embedding (Equation 5.4). The Euclidean distance between the embeddings $\mathbf{x}$ and $\mathbf{c}_i$ is denoted as $d(\mathbf{x}, \mathbf{c}_i)$.

$$n\left(\mathbf{x}\right) = \min_{\mathbf{c}_i \in C} d\left(\mathbf{x}, \mathbf{c}_i\right) \tag{5.4}$$

2) Maximum class density (density) is found by computing the per class densities of training set embeddings nearby an example (Equation 5.5), using the Gaussian kernel sum from Equation 5.2. For notational convenience in Equation 5.3, we subtract the class density from one.

$$n\left(\mathbf{x}\right) = 1 - \max_{\ell} \Pr(y = \ell \mid \mathbf{x}) \tag{5.5}$$

3) Entropy is the Shannon entropy of the class density distribution from Equation 5.2.

$$n\left(\mathbf{x}\right) = -\sum_{\ell} \Pr(y = \ell \mid \mathbf{x}) \log\left(\Pr(y = \ell \mid \mathbf{x})\right) \tag{5.6}$$

Since the metric learning model's class probability distribution is computed based on class densities, the *density* measure is equivalent to measuring novelty based on the maximum class probability. Density is suggested as a suitable measure for novelty detection in [149] and we adapt the method for our approach, using the shared Gaussian σ.

5.6 Active Learning of Novel Classes

When deployed, a deep metric learning model trained on the distribution $\mathcal{K}$, observes new unlabelled data from a mixture of the distributions $\mathcal{K}$ and $\mathcal{N}$. Let $U = \{\mathbf{u}_1, \ldots, \mathbf{u}_\gamma\}$ represent the set of feature embeddings from γ unlabelled observations, with $\mathbf{u}_i = [u_i^{(1)}, \ldots, u_i^{(\beta)}]^T$. In addition to detecting observations belonging to $\mathcal{N}$, our system should select the most informative examples in U for labelling by a user. Obtaining a label is referred to as a *query*. The selected examples should be those that allow the model to learn the most about $\mathcal{N}$, from the fewest label queries. Metric spaces that transfer knowledge to novel examples enable such efficient label querying.

5.6.1 Labelling Budget

A robotic system may observe any large number of objects while navigating the environment. Since the number of potential observations is unbounded, it is unreasonable to expect labels for all examples to be supplied by a user. The labelling budget, denoted as b, is defined as the number of labels that can be obtained from a user. Since the labelling budget may be significantly smaller than the total number of observations, it is important that examples are selected for querying based on an informativeness measure. This selection process is known as *query selection* and is generally based on an informativeness measure that describes how useful an observation may potentially be in allowing the model to improve its understanding of the true data distribution. Our approach to query selection is presented in Section 5.6.2. The experimental results in Section 5.7 evaluate over a large range of labelling budgets, from a small a fraction of the total number of observations up to the best case scenario of acquiring labels for all observations.

5.6.2 Query Selection

With a limited labelling budget, it is vital that the observations that are selected for labelling are those that will be most useful in enabling the learning of the true data distribution. Acquiring labels for examples that are clearly from the known distribution is a waste of the labelling budget, since observations that are already well understood provide little new information to the system. It is also wasteful to acquire labels for novel objects that are rare in

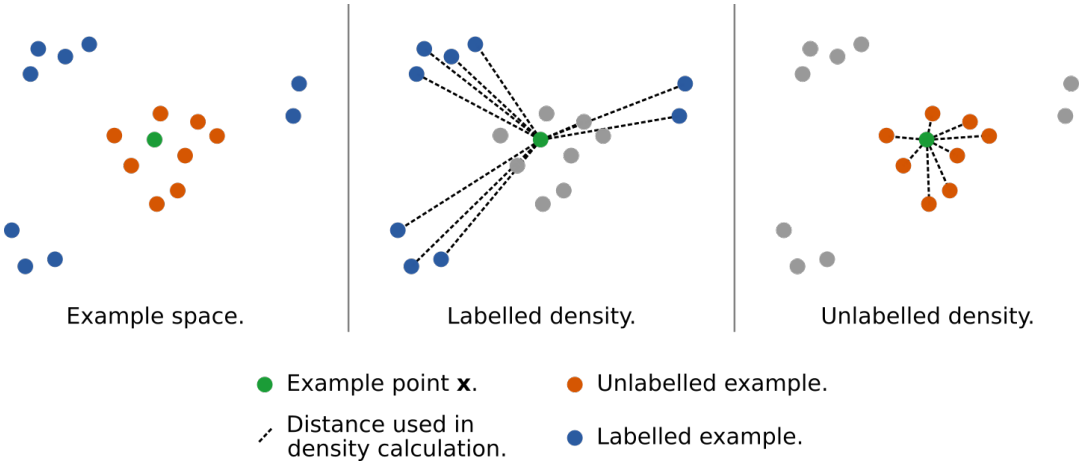


FIGURE 5.6: Distances used when calculating the ULDR for example point $\mathbf{x}$.

the environment, if there are more common novel objects that are not yet understood. Since the learned distance metric transfers to novel classes, it is not only possible to measure the novelty of observations but also how similar they are to other novel observations. This allows the active learning algorithm to first select observations that belong to more common novel classes, before selecting those which are less common.

The proposed informative measure used to rank observations and select queries is the *unlabelled to labelled density ratio* (*ULDR*). This informativeness measure is defined as:

$$\text{ULDR}(\mathbf{x}) = \frac{\sum_{\mathbf{u}_j \in U} \exp\left(\frac{-|\mathbf{x}-\mathbf{u}_j|^2}{2\sigma^2}\right)}{\sum_{\mathbf{c}_k \in C} \exp\left(\frac{-|\mathbf{x}-\mathbf{c}_k|^2}{2\sigma^2}\right)}, \tag{5.7}$$

where U the set of unlabelled feature embeddings and C the set of labelled feature embeddings, which includes the original training data. The informativeness of an example feature embedding $\mathbf{x}$ is the ratio of the density of unlabelled feature embeddings and the density of labelled feature embeddings at the location of $\mathbf{x}$ in the feature embedding space. Note that if the observation $\mathbf{x}$ is in either U or C, the distance to itself is ignored. The value of σ is the same as that in Equation 5.2. The next example selected for querying is that with the largest unlabelled to labelled density ratio. Calculation of the ULDR is visualised in Figure 5.6.

The ULDR query selection method ensures that examples that are novel, as well as good representatives of the distribution of novel observations, are

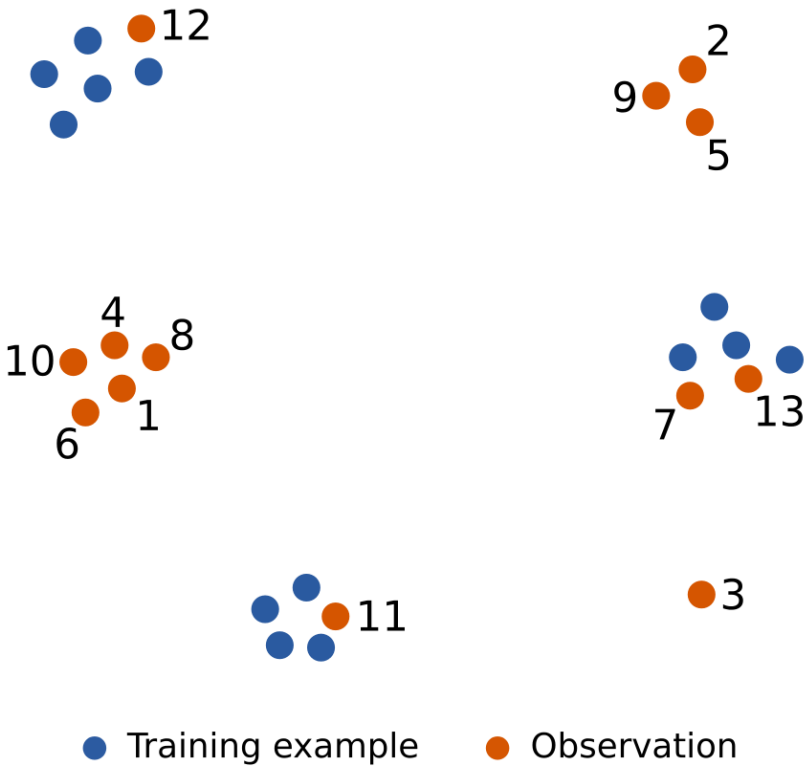


FIGURE 5.7: ULDR query selection on two-dimensional synthetic data. The numbers next to observations represent the order of selection for label querying. Once an observation is labelled, it is included in the set of training examples (blue points).

favoured. Novel observations are selected over known observations as a result of the labelled density (denominator is Equation 5.7). Feature embeddings that are in regions of high labelled density are similar to many known observations and are likely from known classes. While feature embeddings that are far from all labelled examples are likely novel. The favouring of novel examples that are representative of the distribution of novel classes is facilitated by the unlabelled density (numerator in Equation 5.7). A novel example with a small unlabelled density is located in a sparsely populated region of the feature embedding space. An novel example with a large unlabelled density is in a region that is densely populated with other novel observations. Such a cluster of novel observations suggests the presence of a novel class that is common in the environment. That is, the observation likely belongs to a class that is outside of $\mathcal{K}$, but common in $\mathcal{N}$. As such, novel observations that are in regions of high unlabelled density are favoured. This behaviour is shown with synthetic two-dimensional data in Figure 5.7.

5.6.3 Algorithm for Novel Class Active Learning

The active learning of novel classes is described in Algorithm 2. With a set of labelled feature embeddings C, which includes the original training data and any subsequently labelled observations, a set of unlabelled observations U and a labelling budget b, the active learning algorithm first computes the informativeness measure for each observation in U. The informativeness measure is the ULDR, discussed in Section 5.6.2. The formula for the ULDR is reworked in Equation 5.8 (Line 13 of Algorithm 2) to compute density for a single observation in U with index i. After computing the informativeness measure for each unlabelled example, the algorithm selects the observation with the largest ULDR for label querying. A label is then obtained for the selected observation from a user or oracle. The newly labelled example is then removed from the set of unlabelled examples U and included in the set of labelled examples C. This process repeats until the budget is reached.

With b new labelled examples included in the set C, fine-tuning of the network weights can proceed. Gaussian kernels are centred on all new labelled observations; the new entries are now treated the same as any other training example. The model is fine-tuned using all labelled examples in C, including the original training data. This is important to ensure that knowledge about previously learned classes is not lost. Fine-tuning is carried out in the same way as described in Chapter 4. Following fine-tuning, the model should have an improved understanding about the true distribution of data in the environment. Any novel classes that were introduced to the model in the active labelling process are now considered known. The model should correctly classify future observations belonging to such classes.

The proposed process of query selection works based on the knowledge that the learned distance metric transfers to novel classes. The feature embedding space is already well structured for observed novel examples before any fine-tuning with novel classes takes place. Although it is possible to measure the similarity of novel examples, which can be useful in some problems, it is not possible to classify the novel examples, since the model does not know that such novel classes exist. Classifying observations from classes that have no labelled training examples is known as zero-shot classification. Approaches to this problem generally have some sort of description of classes available, such as attributes. As such, the model needs to aware of the existence of specific novel classes, even if there are no labelled examples supplied.

Algorithm 2 Open set active learning with unlabelled to labelled density ratio (ULDR).

Precondition:
Labelling budget b
Set of unlabelled feature embeddings (observations) U
Set of labelled feature embeddings C

for $1 \dots b$ **do**
  Initialise $r(i) = 0$, for all $i \in U$
  for each $i \in U$ **do**
    $r(i) = ULDR(i, U, C)$
  end for
  $q = \arg\max(\mathbf{r})$
  Query user for label of $\mathbf{u}_q$
  C.append($\mathbf{u}_q$) ▷ Add to labelled set
  U.remove(q) ▷ Remove from unlabelled set
end for
Fine-tune deep metric learning model on C

function ULDR(i, U, C) ▷ Query selection

$$r = \frac{\sum_{\mathbf{u}_j \in U, j \neq i} \exp\left(\frac{-|\mathbf{u}_i - \mathbf{u}_j|^2}{2\sigma^2}\right)}{\sum_{\mathbf{c}_k \in C} \exp\left(\frac{-|\mathbf{u}_i - \mathbf{c}_k|^2}{2\sigma^2}\right)} \tag{5.8}$$

  return r
end function

We do not consider zero-shot classification, but rather use active labelling of examples to enable classification of previously unknown classes. Obtaining labels for a subset of the novel observations informs the model of the existence of previously unknown classes. Since the feature embedding space is well structured, reasonable classification performance can be achieved even without model fine-tuning. This is shown in the experiments in Section 5.7.3. Allowing the model to update its weights using the newly obtained information enables some restructuring of the feature embedding space. This new information should not only allow for classification of the newly introduced classes, but also a better metric that can be applied to future observations.

Fast approximate nearest neighbour search and period asynchronous updates of the training embeddings can be utilised to make query selection and network fine-tuning scalable to large numbers of classes and training examples. Nearest neighbours are computed to classify an observation and can be used to consider only a local neighbourhood of training examples for the ULDR computation. These details are discussed in depth in Chapter 4.

5.6.4 Learning with a Zero Labelling Budget

It is known from the experiments in Section 4.10.1 that the learned distance metric transfers well to novel classes. Feature embeddings belonging to classes that the network has not be trained on, are still well clustered based on class. This suggests that it should be possible to update the network weights in such a manner that improves the distance metric, without obtaining any new labels from a user or oracle. In other words, we want to investigate if it is possible to improve the model's understanding of the semantic entities in the environment with a labelling budget of zero. As discussed in Section 5.6.3, without knowledge of the existence of classes, it is not possible to categorise examples into unknown classes. The aim here is not to perform classification of novel observations, but rather to use them to improve the distance metric, or the structure of the feature embedding space, for future observations.

Without the ability to obtain true labels from a user, the algorithm can use spatial relationships in the feature embedding space to obtain pseudo-labels for observations. Since the distance metric transfers to novel classes, a cluster of novel observations likely contains examples belonging to predominantly one unknown class. Pseudo-labels are label assignments that allow supervised learning algorithms to be trained on data for which true class labels are not available. In this case, unique pseudo-labels are supplied to each cluster in the metric space. This can be achieved by using unsupervised clustering techniques, such as k-means [8]. The newly pseudo-labelled examples can be treated as normal training examples, with each pseudo-class treated the same as real classes during network fine-tuning. Gaussian kernels are centred on each pseudo-labelled example and fine-tuning is carried out as normal, with both the original training data and the pseudo-labelled examples.

Of course, the pseudo-label assignments will not be perfect. Although a single cluster in the feature embedding space will likely contain predominately one class, there may be examples of a different class assigned the same pseudo-label. This is akin to learning with noisy labels. Further, since the pseudo-labels are obtained using unsupervised cluster analysis, one true class may be assigned multiple pseudo-classes if it has multiple clusters in the feature embedding space. A single class with multiple clusters suggests a significant semantic difference between the two subsets. Since the Gaussian kernel metric learning model allows multiple clusters to form for a single

class, the assignment of a single class to multiple pseudo-classes may not be detrimental. The ability of the model to learn despite the noisy pseudo-labels is investigated in Section 5.7.3.7.

The pseudo-labels are obtained on the assumption that novel examples are already well clustered based on class. This leads to a question: why would allowing the model to fine-tune with pseudo-labels further improve the distance metric? Improvement seen is likely due to the arrangement of clusters in the feature embedding space being adjusted. Given new observations, the model may push clusters of some classes into better regions of the space. For example, this may allow better representation of a previously known class, given the newly obtained information. It is important that updating the network weights using the noisy pseudo-labels does not degrade the classification performance for known classes. This is investigated in Section 5.7.3.7.

5.7 Experiments

The importance of metric learning to open set problems is investigated in this section. Beginning with novelty detection and open set recognition in Section 5.7.2, several distance-based novelty measures are investigated and compared with softmax baselines and purpose built novelty detectors. In Section 5.7.3, the novel class active learning approach is evaluated. Finally, our proposed approach to learning with a labelling budget of zero is analysed in Section 5.7.3.7.

5.7.1 Datasets

Three datasets are utilised for the open set experiments: Stanford Cars196 [201] (196 classes), Oxford Flowers102 [202] (102 classes) and CUB Birds200 2011 [200] (200 classes). The classes from each dataset are split into known and novel. The first half of classes, that is the first 98, 51 and 100 classes respectively, are taken as known classes (from $\mathcal{K}$). The remaining half are taken as novel classes (from $\mathcal{N}$). The datasets are split into training, observed and test sets. The training sets contain only images belonging to the known classes, while the observed and test sets contain an equal number of known and novel class images. The observed set simulates a model

being deployed in an environment containing both in-distribution and out-of-distribution classes. This is used for evaluating open set recognition and performing active learning. Evaluation of the active learning is performed on the test set.

5.7.2 Novelty Detection and Open Set Recognition

We first evaluate the deep metric learning approach on the problem of open set recognition. In these experiments, the network is trained on the training sets that contain only images from the distribution $\mathcal{K}$. The observed set, which contains images from both the known distribution $\mathcal{K}$ and the distribution of novel classes $\mathcal{N}$, is used for testing. During testing, a perfect system will categorise all examples from $\mathcal{K}$ into the correct known class, while flagging all examples from $\mathcal{N}$ as novel/unknown.

5.7.2.1 Experimental Set-up

A VGG16 [207] architecture is used for all experiments, as we find this network configuration performs particularly well for transfer learning tasks. The second fully connected layer, FC7, is taken as the feature embedding layer. This produces a 4096-dimension feature embedding for a given input image, and therefore, a 4096-dimension metric space. The network is trained on the training set of known classes, following the methodology described in Chapter 4. Training data is augmented using random cropping and horizontal mirroring. For the Gaussian kernel approach, a learning rate of 10^{-5}, weight decay of 5×10^{-4}, momentum of 0.9 and shared Gaussian kernel σ of 91, 75 and 103 is used for Cars196, Flowers102 and Birds200, respectively. All hyperparameters are selected as described in Section 4.10.2.2.

5.7.2.2 Evaluation Metrics

Three metrics are used to evaluate the compared approaches on the task of novelty detection. These evaluation metrics and any necessary intermediate measures are described below. Note that for all measures except the false positive rate (FPR), a higher value corresponds to better performance.

- **True Positives (*TP*)**: Number of novel examples correctly detected as novel.

- **False Positives (*FP*)**: Number of known examples incorrectly detected as novel.
- **True Negatives (*TN*)**: Number of known examples correctly not detected as novel.
- **False Negatives (*FN*)**: Number of novel examples not detected as novel.
- **Recall (*recall*)** = $\frac{TP}{TP+FN}$
- **Precision (*precision*)** = $\frac{TP}{TP+FP}$
- **True Positive Rate (*TPR*)** = *recall*
- **False Positive Rate (*FPR*)** = $\frac{FP}{FP+TN}$
- **Area Under Precision-Recall Curve (*AUPR*)**: Computing the recall and precision requires the selection of a discrimination threshold (i.e. δ from Equation 5.3), which in turn requires a desired level of precision to be selected. To avoid threshold selection, the area under the precision-recall curve, created by sweeping the threshold across an appropriate range of values, is computed. This measure is threshold independent.
- **Area Under ROC Curve (*AUROC*)**: Another threshold independent measure that is the area under the receiver operating characteristic (ROC) curve found by sweeping the threshold δ. The ROC curve is a plot of the true positive rate against the false positive rate.
- **F-measure (*F1*)**: A fixed-threshold metric, the F-measure takes both the precision and recall into account:

$$F1 = 2\,\frac{precision \cdot recall}{precision + recall} \tag{5.9}$$

For open set recognition, that is, the problem of simultaneously detecting novel examples and classifying known examples, performance is measured using the open set recognition accuracy (*Acc.*). This is the standard classification accuracy with a single unknown/novel superclass for all observations from $\mathcal{N}$. That is, for a problem with L known classes, an example is predicted as class $L+1$ when it is detected as novel. Open set accuracy is defined as $Acc. = \frac{1}{M}\sum_{m=1}^{M}[y_m = \hat{y}_m]$, where $\mathbf{y} = [y_1, \ldots, y_M]$ and $\hat{\mathbf{y}} = [\hat{y}_1, \ldots, \hat{y}_M]$ are the true and predicted class labels, respectively, and the term $[y_m = \hat{y}_m]$ evaluates to zero or one based on the logical condition.

5.7.2.3 Compared Methods

We evaluate our deep metric learning approach to novelty detection and open set recognition with varying novelty measures. Full details on the novelty measures can be found in Section 5.5.3. Our methods are:

- ***DML NN dist.***: Our deep metric learning approach with nearest neighbour distance novelty measure.
- ***DML Density***: Our deep metric learning approach with maximum class density novelty measure.
- ***DML Entropy***: Our deep metric learning approach with Shannon entropy novelty measure.

Our approach is compared to baselines and purpose built novelty detectors and open set classifiers, which are discussed in detail in Sections 5.1.1.1 and 5.1.1.2. The compared methods are:

- ***Baseline [145] Max Pr.***: A baseline softmax uncertainty novelty detector with maximum class probability thresholding.
- ***Baseline [145] Entropy***: A baseline softmax uncertainty novelty detector with Shannon entropy thresholding.
- ***OpenMax [147]***: Open set version of softmax classification.
- ***ODIN [148] Max Pr.***: Out-of-distribution detector with maximum class probability thresholding.
- ***ODIN [148] Entropy.***: Out-of-distribution detector with Shannon entropy thresholding.

For both the softmax baseline and ODIN, we utilise two different novelty measures: the confidence score of a classification and the entropy of the classification probability distribution. Entropy is a more informative measure as it considers all class probabilities, rather than simply the maximum class probability. The non-maximum probabilities often contain useful information regarding the certainty of a classification and as such, entropy may be a better measure of novelty than the confidence score for these classifiers.

Both our approach and [145] have only one tunable parameter (the threshold δ), while [147] and [148] each have three. A withheld set of images is used to tune the parameters such that the withheld set F-measure is maximised, as suggested in [147]. Note that no parameter tuning is needed for the AUROC

and AUPR measures for our approach or the softmax baseline [145]. Since OpenMax [147] explicitly includes an estimated novel class probability, AUROC and AUPR measures cannot be computed. As such, we report only the F-measure and open set accuracy for this approach.

5.7.2.4 Results

Results are shown for the Cars196, Flowers102 and Birds200 datasets in Tables 5.1, 5.2 and 5.3, respectively. Our deep metric learning approach outperforms the compared methods on all evaluation measures and datasets, in most cases by a significant margin. The softmax baseline and ODIN both see greater performance when using the entropy novelty measure, compared to the maximum class probability, in most cases. In general, the advantage of using entropy is larger for the softmax baseline. Comparing the best performing novelty measure for each method on the Cars196 dataset, the deep metric learning approach results in a 4.13%, 4.47% and 6.37% increase in the F-measure over the ODIN, OpenMax and softmax baseline methods, respectively. On the threshold-independent AUROC measure, the metric learning method outperforms ODIN and the softmax baseline by 3.6% and 5.12%, respectively, on the Cars196 dataset. Considering the open set accuracy, a 4.18%, 5.12% and 6.46% increase is achieved by the metric learning approach over ODIN, OpenMax and the softmax baseline, respectively. These results show that metric learning is important for open set problems, with a consistent and significant advantage seen over softmax and purpose built novelty detectors and open set classifiers.

All three novelty measures investigated for our metric learning approach outperform the best performing compared method in all cases. Comparing the three distance-based novelty measures, density is almost uniformly the poorest performer. Across the twelve metrics (four evaluation metrics on three datasets), the entropy novelty measure is the best performing on five occasions, while the nearest neighbour distance is the best novelty measure on the other seven occasions.

TABLE 5.1: Novelty detection results on Cars196.

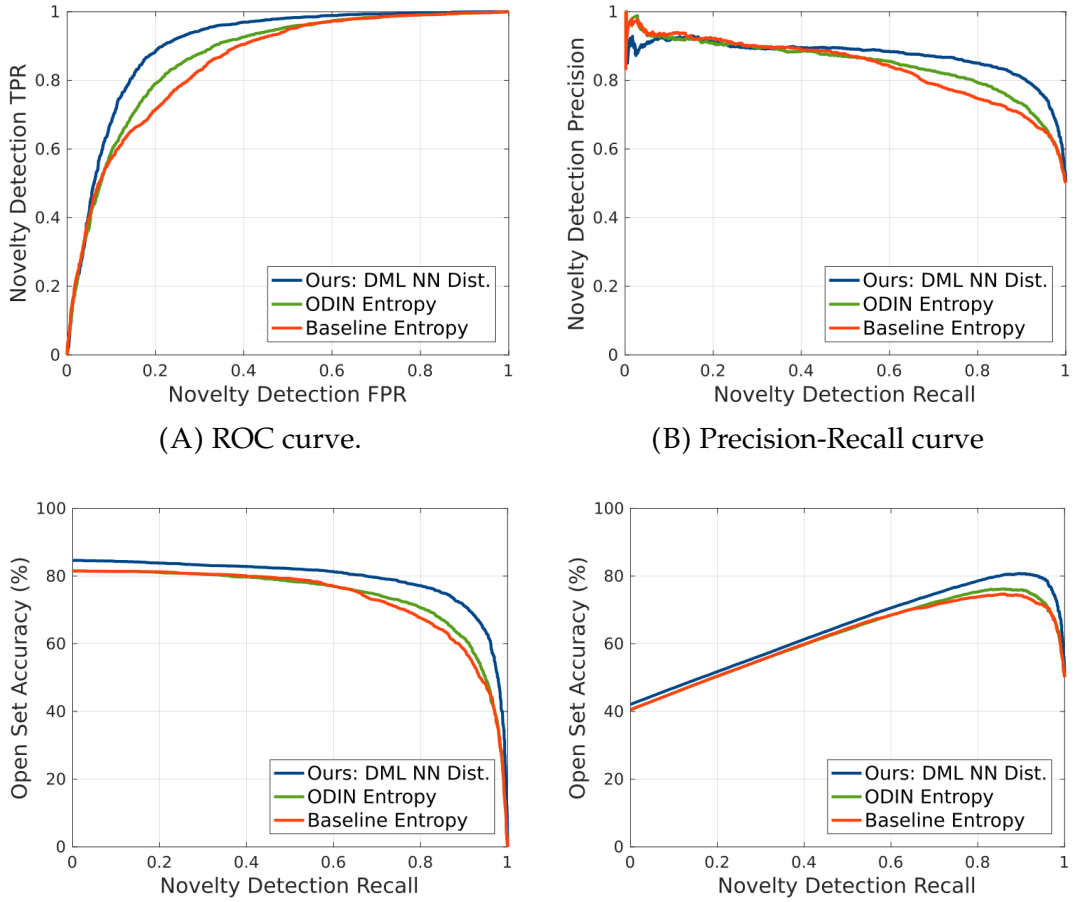
Method	AUROC	AUPR	F1	Acc. (%)
Baseline [145] Max Pr.	0.8331	0.8116	0.7832	73.34
Baseline [145] Entropy	0.8512	0.8374	0.7865	73.95
OpenMax [147]	-	-	0.8055	75.15
ODIN [148] Max Pr.	0.8613	0.8443	0.8021	75.31
ODIN [148] Entropy	0.8668	0.8469	0.8089	76.23
Ours: DML Density	0.8901	0.8671	0.8263	78.78
Ours: DML Entropy	0.9013	**0.8710**	0.8454	80.33
Ours: DML NN Dist.	**0.9028**	0.8706	**0.8502**	**80.41**

TABLE 5.2: Novelty detection results on Flowers102.

Method	AUROC	AUPR	F1	Acc. (%)
Baseline [145] Max Pr.	0.8509	0.8051	0.8004	78.73
Baseline [145] Entropy	0.8559	0.8206	0.8015	79.07
OpenMax [147]	-	-	0.7985	75.88
ODIN [148] Max Pr.	0.8712	0.8471	0.8021	75.31
ODIN [148] Entropy	0.8690	0.8447	0.8085	78.48
Ours: DML Density	0.9043	0.8741	0.8442	82.55
Ours: DML Entropy	**0.9084**	0.8718	0.8477	82.99
Ours: DML NN Dist.	0.9078	**0.8884**	**0.8543**	**83.97**

TABLE 5.3: Novelty detection results on Birds200.

Method	AUROC	AUPR	F1	Acc. (%)
Baseline [145] Max Pr.	0.7311	0.6933	0.7277	64.73
Baseline [145] Entropy	0.7397	0.7017	0.7280	64.22
OpenMax [147]	-	-	0.7628	68.93
ODIN [148] Max Pr.	0.7383	0.7031	0.7271	64.04
ODIN [148] Entropy	0.7400	0.7034	0.7260	63.89
Ours: DML Density	0.7838	0.7475	0.7419	68.95
Ours: DML Entropy	**0.7981**	**0.7601**	0.7559	**70.12**
Ours: DML NN Dist.	0.7961	0.7489	**0.7652**	68.40



(A) ROC curve.

(B) Precision-Recall curve

(C) Open set accuracy (known classes only).

(D) Open set accuracy (known and novel classes).

FIGURE 5.8: Novelty detection on the Cars196 dataset. The best performing novelty measures from Table 5.1 are shown for each compared method. Note that the novelty detection recall and true positive rate (TPR) are the same measure. The open set accuracy in (C) considers only the test examples belonging to the set of known classes. The open set accuracy in (D) considers test examples from both the known and novel class sets, with a single superclass for all novel examples.

Further results are shown in Figure 5.8 for the Cars196 dataset. The best performing novelty measures for each of the Baseline [145], ODIN [148] and DML approaches are shown. Figures 5.8A and 5.8B show the novelty detection ROC curve and precision-recall curve, respectively. Our approach outperforms the compared method across all novelty detection recall values, except when the recall is less than approximately 0.1. In this region, the compared methods result in a better novelty detection precision. However, this range of very low novelty detection recall values is not of great interest. Figure 5.8C shows the open set accuracy as a function of the novelty detection

recall, considering only the test examples from the known set of classes. Figure 5.8D shows the open set accuracy for both known and novel class test examples. The open set accuracy includes a superclass for all novel examples. Our approach outperforms the compared methods at all recall values.

5.7.3 Open Set Active Learning Results

The proposed approach to novel class active learning is evaluated in this section. The advantage of a metric learning approach in general, over a conventional network configuration, is shown. We also investigate the quality of the proposed query selection methodology, comparing it to other commonly used distance-based query selection approaches.

5.7.3.1 Experimental Set-up

Initial training of the network is carried out on the training set of known classes in the same manner as the novelty detection experiments, which is described in Section 5.7.2.1. Query selection is then carried out on the observed set, which contains examples from both known and novel classes. Following the query selection, which expands the training set size, the network is further fine-tuned with the same parameters. This fine-tuning stage includes training examples from the original training sets, as described in Section 5.7.1, as well as any newly labelled examples from the query selection algorithm. In our experiments, labels are provided to the model automatically in response to a query. This simulates the process of a human user providing labels to a robot. Approaches are then evaluated on the test set, containing unseen examples from both the original known class set and the novel class set.

5.7.3.2 Evaluation Metrics

Standard classification accuracy is used to evaluate the approaches. The accuracy on the novel class examples only is reported, as well as the accuracy on the combined known and novel class examples. Note that individual novel classes are used for the accuracy calculation in these experiments, not a single superclass, as in Section 5.7.2.

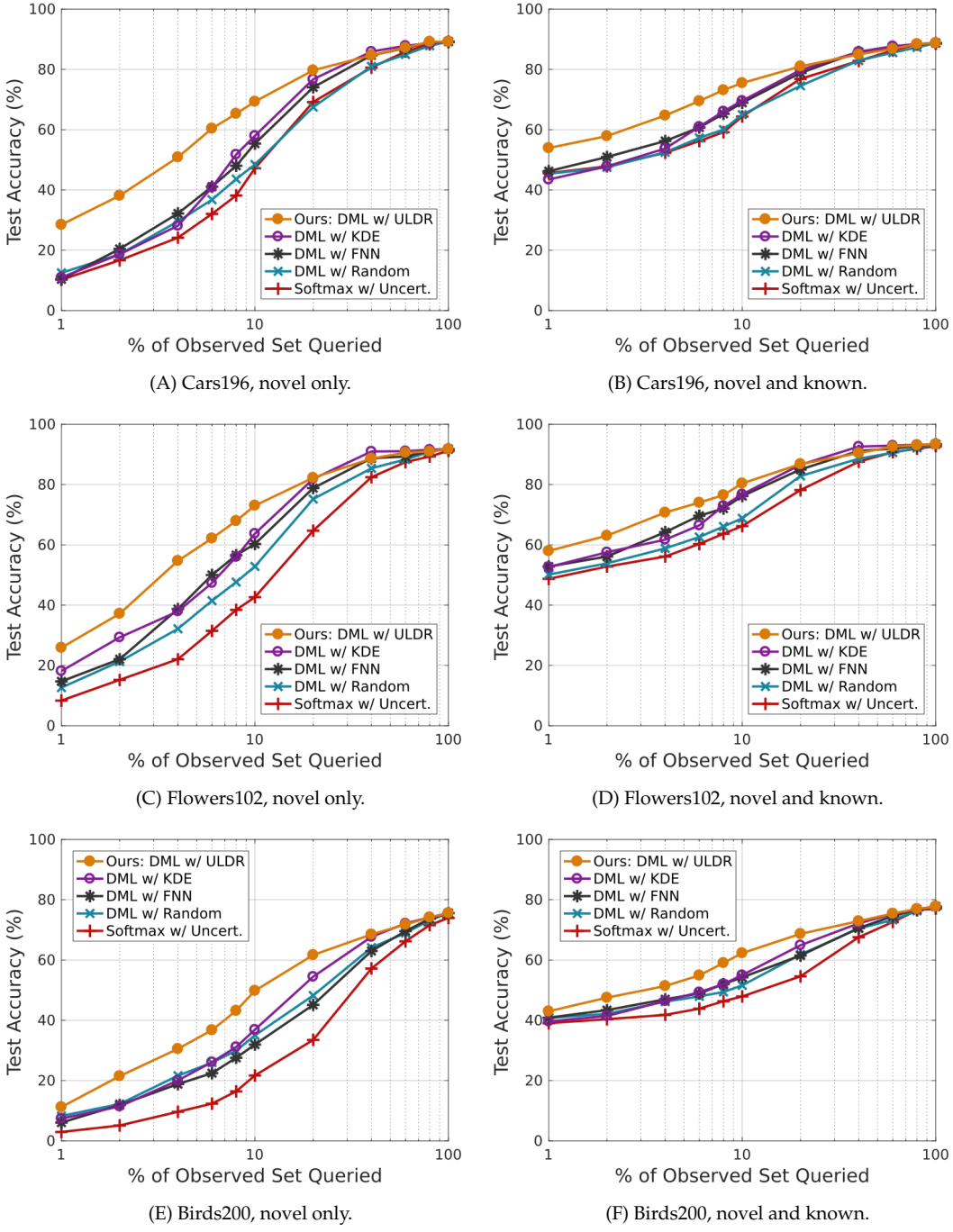
5.7.3.3 Compared Methods

We compare a deep metric learning-based active learning approach to a conventional softmax network approach. The proposed query selection method is also evaluated, by comparing it to other commonly used distance-based informativeness measures. The compared distance measures are discussed in detail in Section 5.1.2.3. Random query selection is also included to provide some context to the results. Approaches that include *DML* in their name are deep metric learning approaches, all of which use the same metric learning algorithm and model, described in Chapter 4. The only difference between these approaches is the method of query selection in the active learning algorithm. The following approaches are compared:

- ***Softmax w/ Uncert.***: A conventional softmax approach with a typical query selection method based on classifier uncertainty. The observation with the largest Shannon entropy is queried.
- ***DML w/ Random***: Deep metric learning approach with random query selection.
- ***DML w/ FNN [216]***: Deep metric learning approach with furthest nearest neighbour (FNN) query selection [216]. The observation with the largest distance to its nearest labelled example is queried.
- ***DML w/ KDE***: Deep metric learning approach with kernel density estimation (KDE) query selection. The observation with the smallest maximum class probability (density) is queried. The probabilities are found using Equation 5.2.
- ***Ours: DML w/ ULDR***: Deep metric learning with our unlabelled to labelled density ratio (ULDR) query selection. The observation with the largest ULDR is queried.

5.7.3.4 Quantitative Results

Novel class active learning results for the three experimented datasets are shown in Figure 5.9. The plots show the test set classification accuracies as a function of the labelling budget. The labelling budget is shown as a percentage of the total observed set queried. A labelling budget of 100% means that the entire observed set is included as training data. This represents the upper bound on performance. Our experiments aim to show two important



(A) Cars196, novel only.

(B) Cars196, novel and known.

(C) Flowers102, novel only.

(D) Flowers102, novel and known.

(E) Birds200, novel only.

(F) Birds200, novel and known.

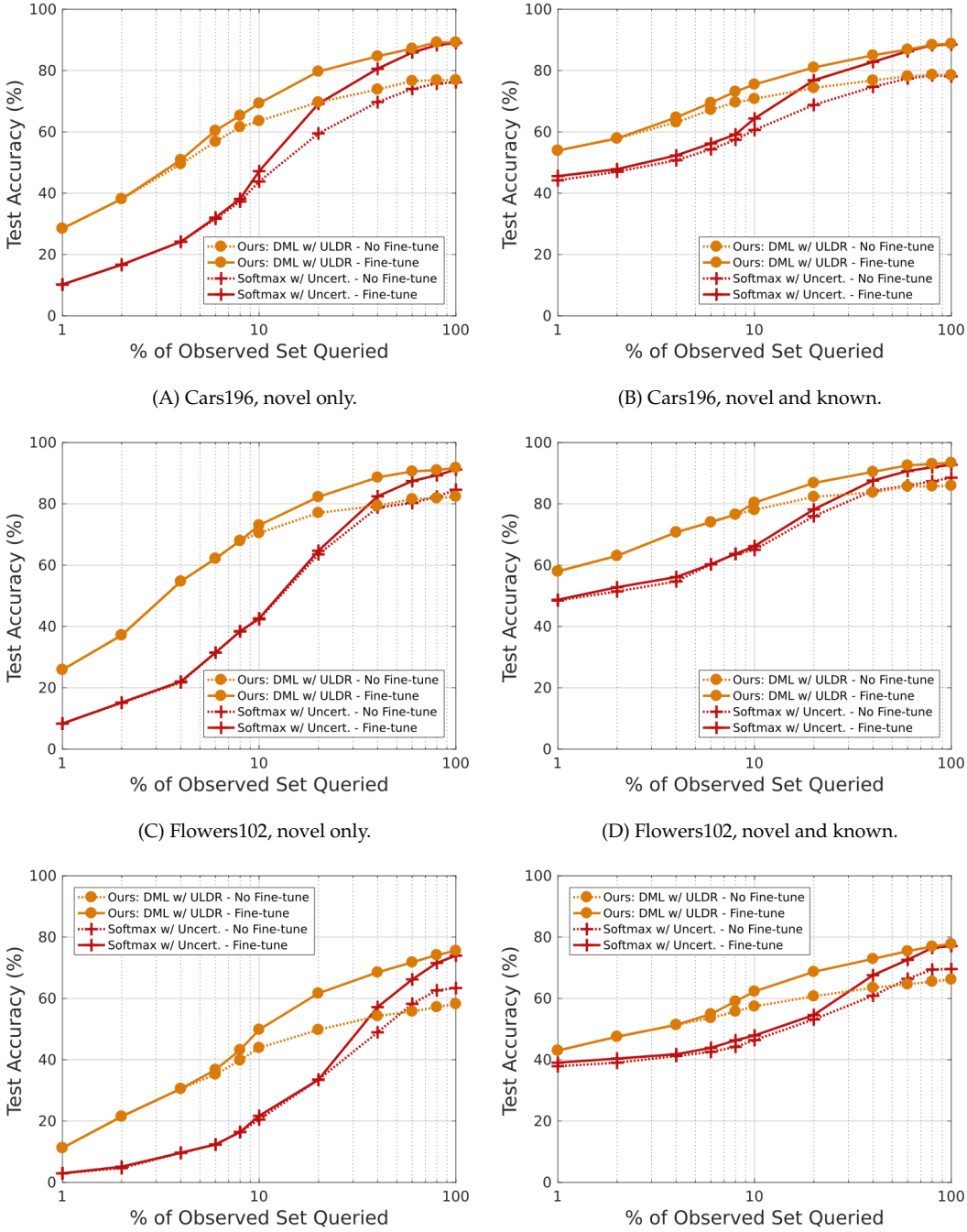
FIGURE 5.9: Active learning results on Cars196 (top), Flowers102 (middle) and Birds200 (bottom). Plots show the test set classification accuracy of novel classes only and the combined novel and known classes at various labelling budgets. The labelling budget is represented as a percentage of the total observed set.

points: that deep metric learning is better suited to open set active learning than softmax-based networks, and that our proposed ULDR approach to query selection is efficient and effective.

All deep metric learning approaches outperform the softmax approach in the majority of cases. Even random selection of queries in the metric learning framework results in an often significant advantage over the softmax-based approach. Comparing the softmax approach with the best performing metric learning approach (our ULDR query selection), the performance gain is significant. The advantage of the deep metric learning approach is largest at small labelling budgets. This is important because the aim is to be able to learn from as few queries as possible. As the labelling budget increases, the advantage shrinks. This is expected as more examples are queried and the methods of measuring informativeness and novelty become less important. Considering the Cars196 dataset, our deep metric learning approach with ULDR query selection outperforms the softmax approach by 18.26%, 26.70% and 22.13% at labelling budgets of 1%, 4% and 10%, respectively, on the novel class accuracy. For combined known and novel class accuracy, the advantage is 8.34%, 12.40% and 11.10%, for the same labelling budgets. These results show the importance of metric learning to open set active learning problems.

Our proposed ULDR query selection method significantly outperforms the compared distance-based query selection approaches. Again, the advantage is largest for small labelling budgets, as query selection is less import as a larger fraction of the observed data is labelled. Compared to the KDE approach, the ULDR query selection results in a novel class accuracy improvement of 17.52%, 22.73% and 11.36% at labelling budgets of 1%, 4% and 10%, respectively. At the same labelling budgets, the advantage over the FNN approach is 18.11%, 18.66% and 13.95%, respectively. Our ULDR method results in the selection of informative examples, enabling efficient learning of the novel class distribution from few label queries.

The impact of fine-tuning the network weights after query selection with both the original training data and the newly labelled examples is shown in Figure 5.10. Results are shown for the DML approach with ULDR query selection and for the softmax approach. The results with no fine-tuning of the network weights are obtained for the DML approach by simply including the newly labelled examples in the set of Gaussian centres, allowing these new examples to contribute to the classification of test examples. No fine-tuning of any network parameters is carried out. For the softmax approach, the results with no fine-tuning are obtained by freezing all network weights, except the final class-dependent fully connected layer. This layer must be trained because it is reshaped when new classes are discovered.



(A) Cars196, novel only.

(B) Cars196, novel and known.

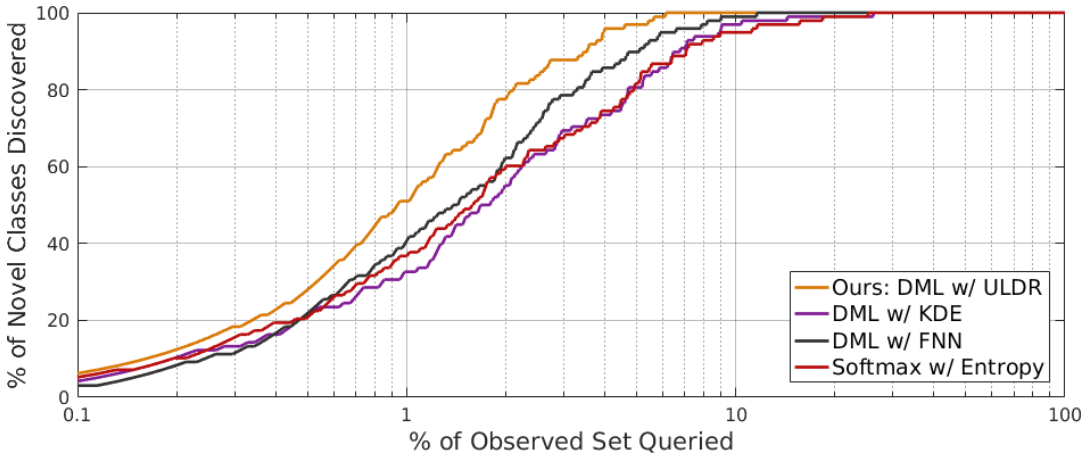
(C) Flowers102, novel only.

(D) Flowers102, novel and known.

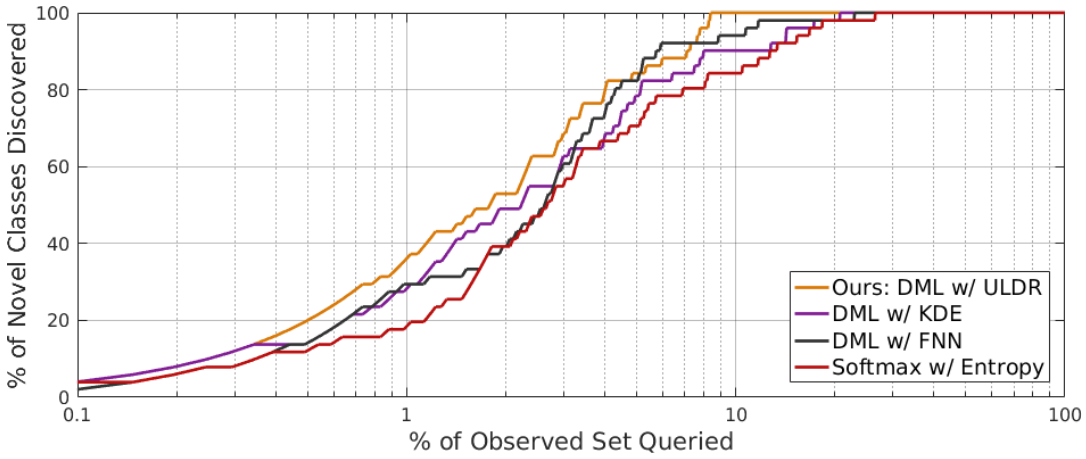
(E) Birds200, novel only.

(F) Birds200, novel and known.

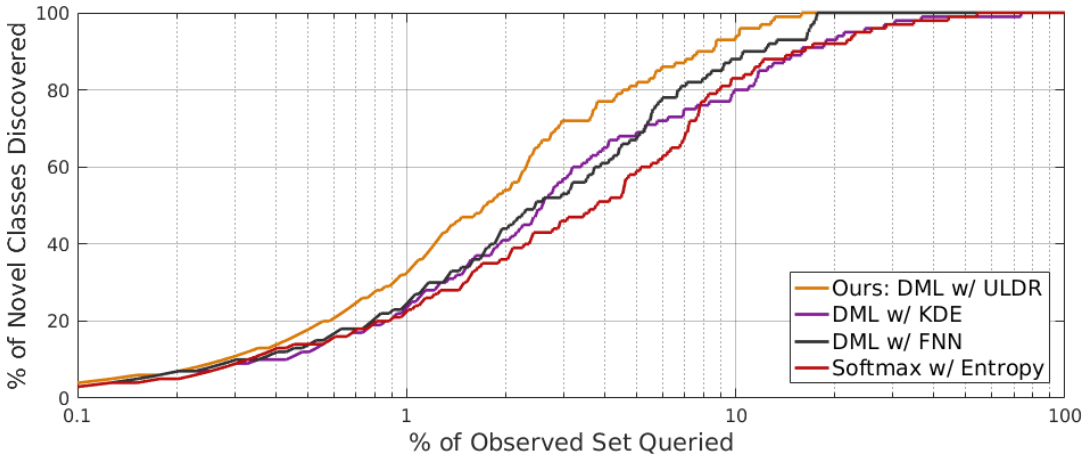
FIGURE 5.10: Impact of fine-tuning the network weights with newly labelled queries for both the deep metric learning (with ULDR query selection) and softmax (with uncertainty-based query selection) approaches. For softmax with no fine-tuning, all network weights are frozen except the final fully connected layer, which must be reshaped and retrained when new classes are introduced. For deep metric learning (DML) with no fine-tuning, all network weights are frozen and newly labelled examples are included as Gaussian centres for the purpose of classification.



(A) Cars196 dataset.

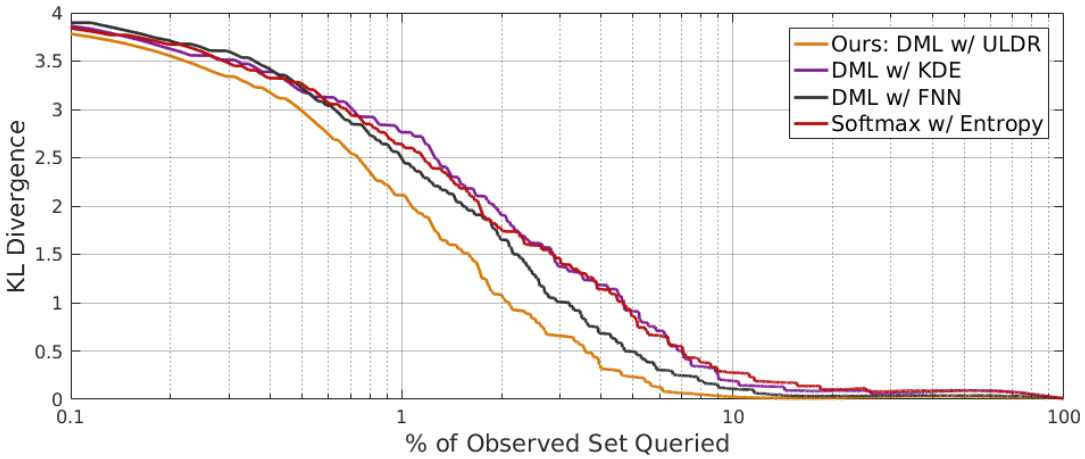


(B) Flowers102 dataset.

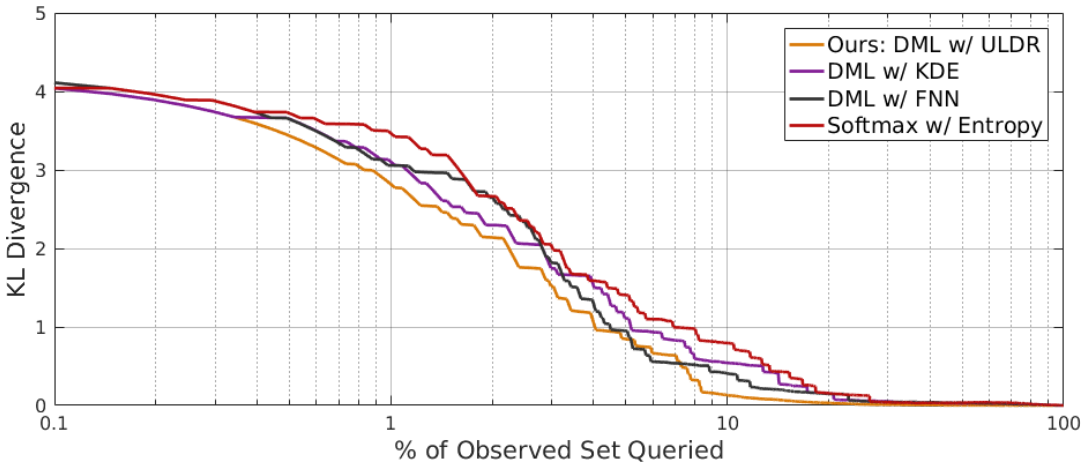


(C) Birds200 dataset.

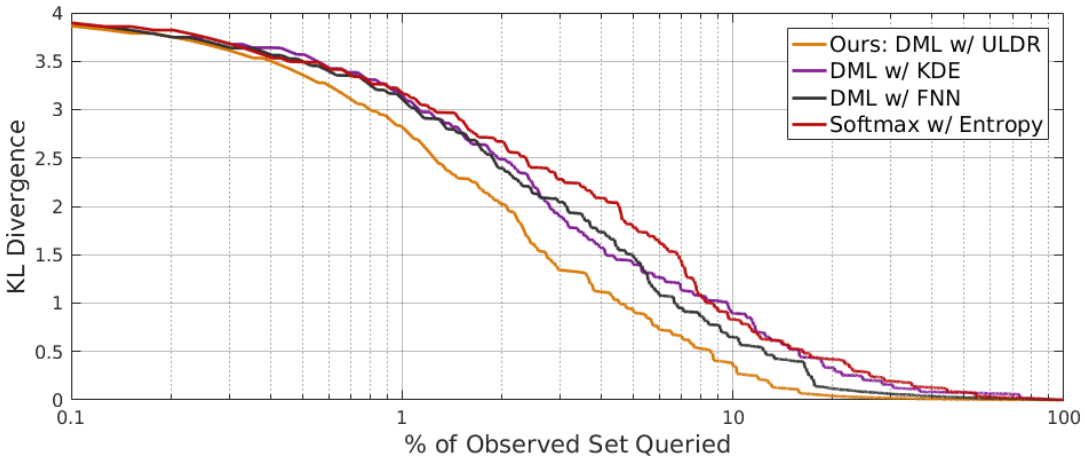
FIGURE 5.11: The number of novel classes in the observed set that have been discovered as a function of the number of queries. A novel class is "discovered" if at least one observation from that class has been selected for querying.



(A) Cars196 dataset



(B) Flowers102 dataset.



(C) Birds200 dataset.

FIGURE 5.12: The Kullback–Leibler divergence (KL divergence) between the true class distribution (all labelled examples in the training and observed sets) and the current class distribution (all labelled training examples and the observed examples queried so far) as a function of the number of queries.

The active learning approaches are further compared by analysing the number of novel classes discovered as a function of the number of label queries in Figure 5.11. A novel class is considered "discovered" if there is at least one newly labelled observation with that class label. A good query selection method for novel class active learning should discover classes with as few label queries as possible. Our approach outperforms the compared methods in the vast majority of cases. For example, at a labelling budget of 2% on the Cars196 dataset, our method outperforms the FNN approach by 15.3% and the KDE approach by 22.5%.

We also investigate the Kullback–Leibler divergence (KL divergence) between the current class distribution and the true class distribution, as a function of the number of label queries, in Figure 5.12. A small KL divergence indicates that the approximated distribution is similar to the true distribution, while a large divergence indicates that the distributions differ significantly. The true class distribution is computed using all training examples and all examples in the observed set. The current approximated distribution is found using all training examples and the observed examples that have been selected for label querying. To avoid zero probabilities for undiscovered classes, small values are added to zero elements. These values affect only the scale of the plots, with negligible impact on the shapes. This allows for reasonable relative comparison between the approaches. Again, our proposed method outperforms the compared methods in the vast majority of cases.

5.7.3.5 Visualising Query Selection

The compared query selection approaches for deep metric spaces (FNN, KDE and ULDR) are visualised for Cars196, Flowers102 and Birds200 in Figures 5.13, 5.14 and 5.15, respectively. The figures show t-SNE [215] visualisations of the metric space, with the original training examples and the examples from the observed set included. The observed examples selected for querying are shown for each of the query selection methods at labelling budgets of 1% and 10% of the entire observed sets. It is important to note that t-SNE visualisations are not faithful representations of the global space. The size and density of clusters and the distance between them may not be meaningful. This is because the data transformations are not consistent across the entire

space, resulting in a notion of distance that differs regionally. The t-SNE algorithm can be helpful in understanding the query selection process in high-dimensional metric spaces, however, the visualisations should be taken with a pinch of salt.

Analysing the Birds200 visualisation in Figure 5.15 with a labelling budget of 1%, it can be seen that the ULDR method selects no observations from known classes for querying. Conversely, the KDE and FNN methods select several known class examples, wasting some of the limited labelling budget on classes that already have a large amount of labelled examples in the training data. For a labelling budget of 10%, the ULDR approach again selects significantly fewer known class examples for querying, particularly compared to the FNN approach. Further, the ULDR method results in better coverage of the novel class examples. The KDE approach results in large regions and clusters of novel examples with no labels, meaning that several novel classes are yet to be discovered by this approach. This result is mirrored in the class discovery plots in Figure 5.11. These visualisations, combined with the class discovery plots, show how our proposed approach is able to perform efficient query selection of novel observations, with examples selected from across the range of novel classes and with few queries wasted on classes that already have a large number of labelled examples.

5.7.3.6 Qualitative Results

The novel class test set feature embedding space is visualised in Figure 5.16. Two-dimensional visualisations of the high dimensional feature space are obtained using the t-SNE algorithm [215]. The initial feature embedding space is shown, that is, the space before any active learning has taken place. Novel test classes are already quite well clustered before any learning has taken place with novel examples. This shows the transfer learning capabilities of deep metric learning that motivate our approach. Also shown is the feature embedding space after fine-tuning on observed examples, with labelling budgets of 10% and 100%. The 100% labelling budget is the best case scenario and is included to show the upper bound of performance. The improvement seen with a labelling budget of 10% over the initial state is significant, though novel examples are less tightly clustered compared to the best case. Note that the visualised feature embeddings are from the test set and unseen during initial training and active learning.

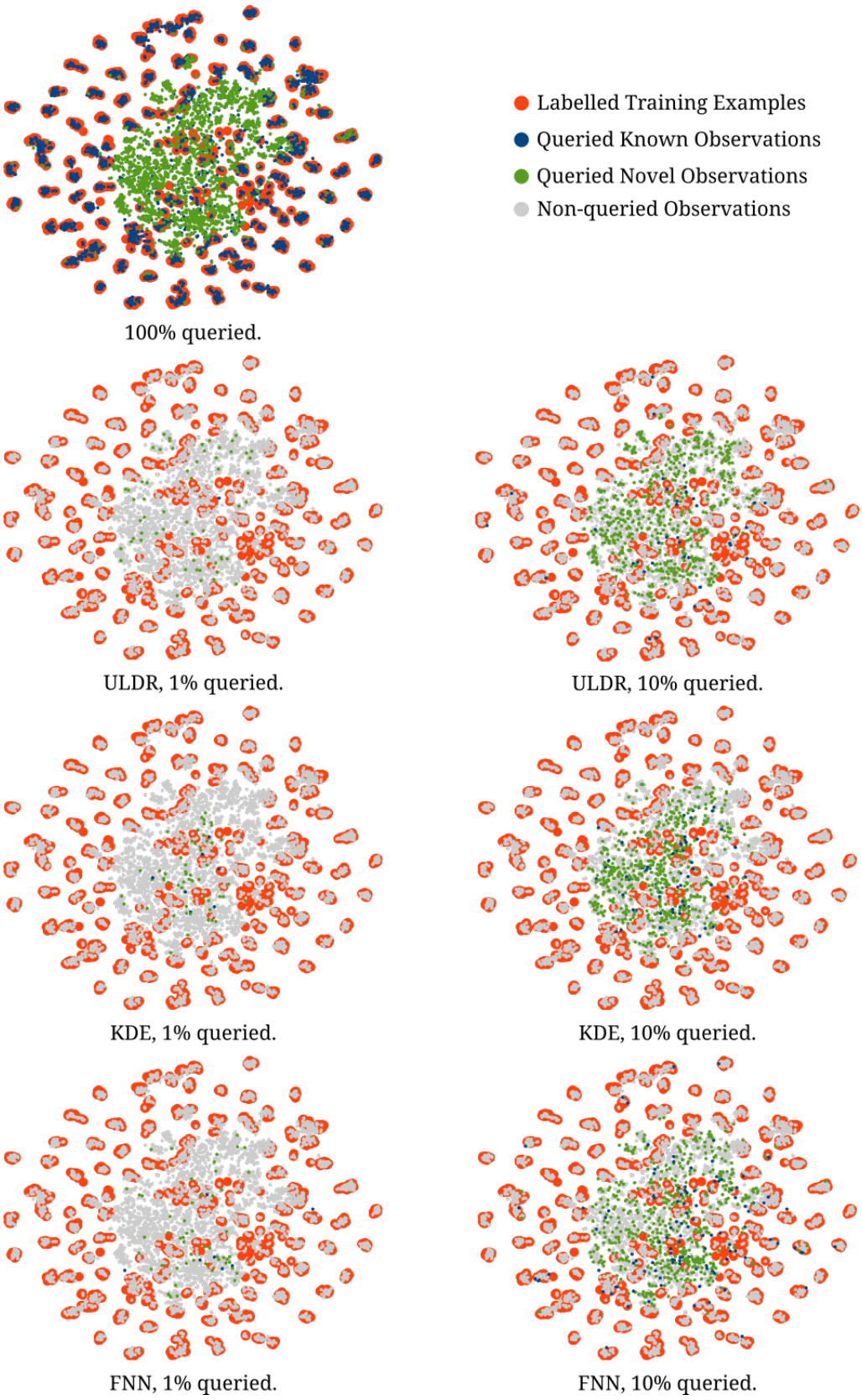


FIGURE 5.13: Cars196 dataset: t-SNE visualisation of query selection approaches for deep metric spaces. Best viewed zoomed in on a monitor.

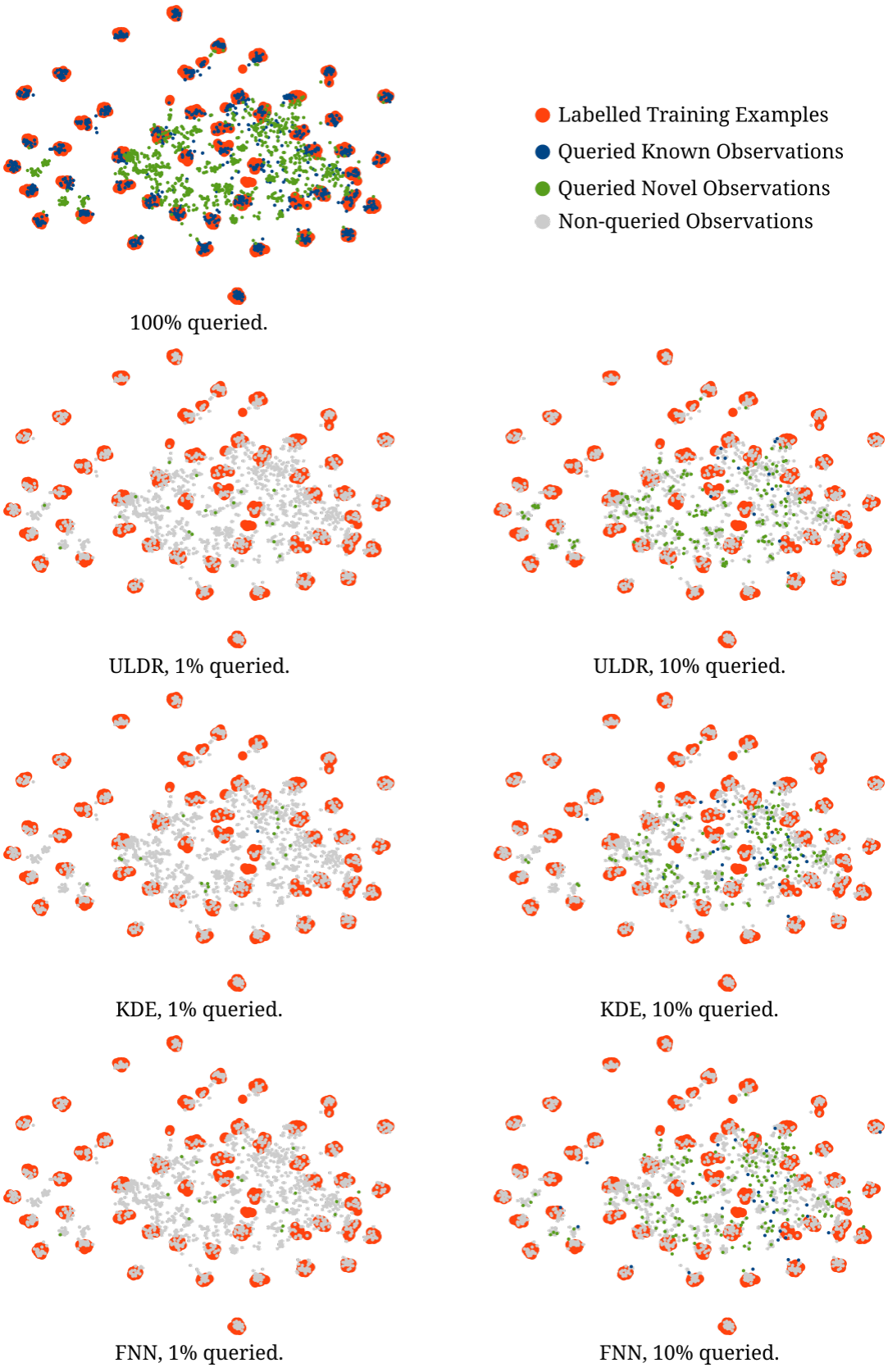


FIGURE 5.14: Flowers102 dataset: t-SNE visualisation of query selection approaches for deep metric spaces. Best viewed zoomed in on a monitor.

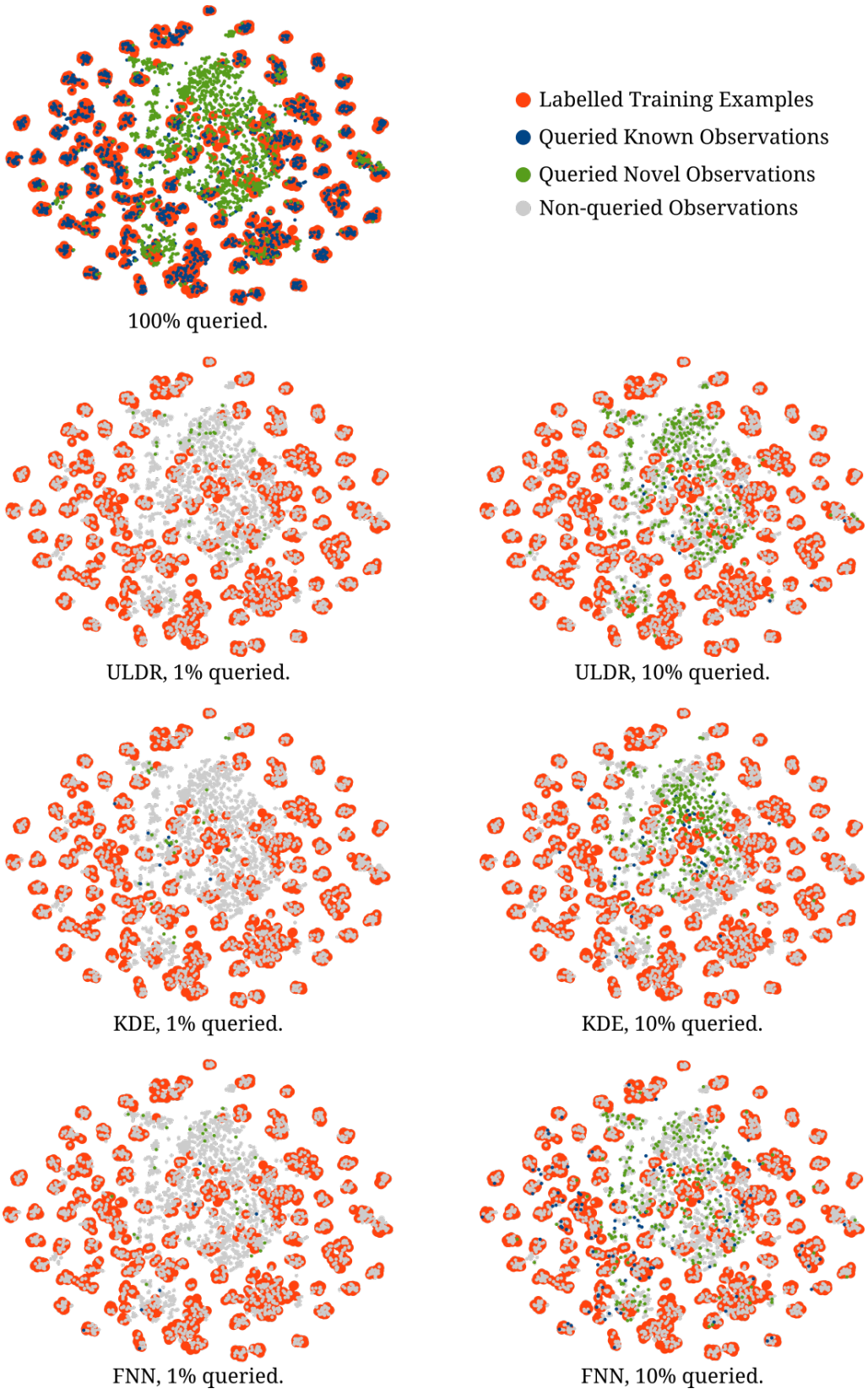


FIGURE 5.15: Birds200 dataset: t-SNE visualisation of query selection approaches for deep metric spaces. Best viewed zoomed in on a monitor.

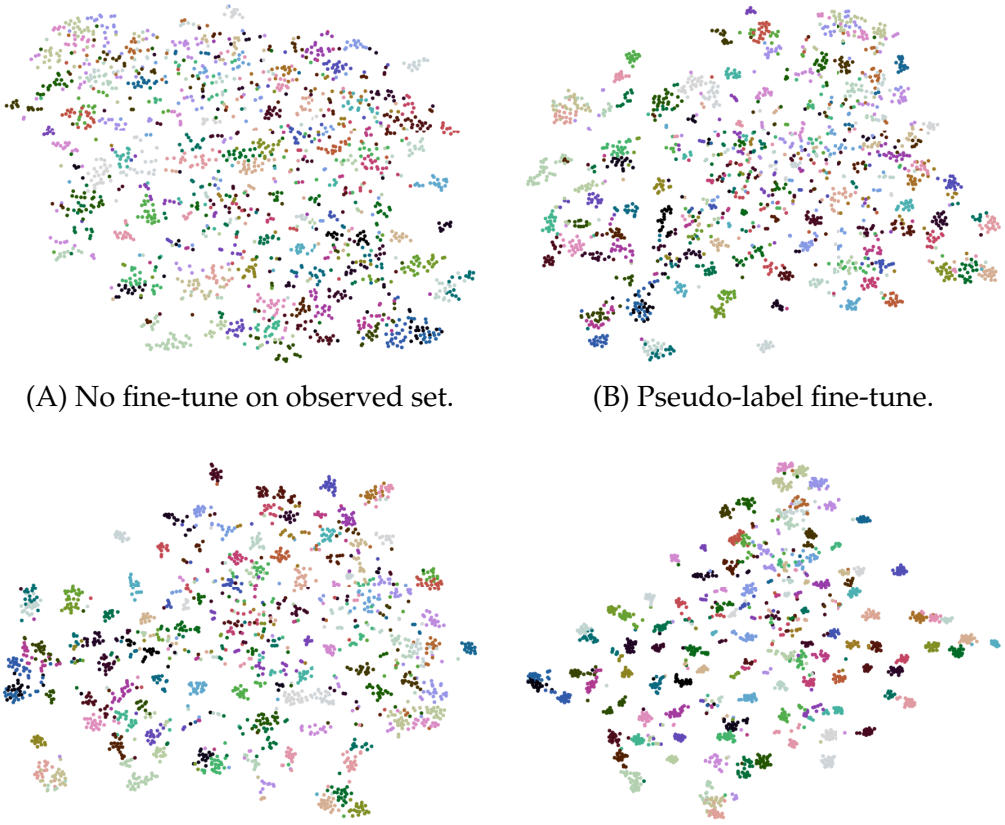
TABLE 5.4: Unsupervised pseudo-label approach compared to no novel class fine-tuning and active learning of novel classes. The mean pseudo-label results with standard deviation values are shown.

	Novel R@1	Novel R@2	Novel R@4	Novel R@8	Known Acc. (%)
Initial	68.93	79.11	86.80	91.61	83.22
Pseudo-labels	75.04± 0.23	83.16± 0.40	89.04± 0.47	93.06± 0.23	83.68± 0.25
AL, b = 10%	77.42	84.47	89.78	93.60	83.87
AL, b = 100%	86.80	91.01	94.84	96.87	89.01

5.7.3.7 Zero Labelling Budget Results

We further investigate whether a model can improve its representation of observed novel classes with a labelling budget of zero. We use spatial relationships in the metric space to generate pseudo-labels for observed examples. In other words, the knowledge that deep metric spaces transfer well to novel classes is used to generate a training signal. We use k-means [8], with k-means++ initialisation [221], to obtain pseudo-labels for each observed example. The network is fine-tuned using the observed examples with pseudo-labels together with the original training examples. The value of k is selected such that the Silhouette Score [222] is maximised, indicating that the cluster assignments are tight. A k value of 240 is used for the Cars196 dataset. Since the true labels of the observed set are not known in this case, we evaluate how well the network has learned to represent the novel set of classes using a recall measure on the test set examples. Recall@n (R@n) is the percentage of test examples that have the same true class label as at least one of their n nearest neighbours in the metric space. Note that this is the same measure as the Recall@k measure from Chapter 4, but renamed to avoid confusion with the k from the k-means algorithm. Experiments are run multiple times and the results are averaged.

Table 5.4 shows the Recall@n of the novel examples in the Cars196 test set. Note that the test set metric space contains examples from both known and novel classes. The classification accuracy of the test set examples from known classes (*Known Acc.*) is also shown. The pseudo-label approach is compared to a model that has not been fine-tuned on the observed set (*initial*). This



(A) No fine-tune on observed set.

(B) Pseudo-label fine-tune.

(C) AL with b = 10%.

(D) AL with b = 100%.

FIGURE 5.16: Visualisations of the metric space of novel class test examples using the t-SNE algorithm [215] on Cars196. Colour represents the class of examples. The initial metric space before fine-tuning on the observed set is shown in (A), while (B) is the metric space after fine-tuning using pseudo-labels only (Section 5.7.3.7). Metric spaces after active learning (*AL*) are shown in (C) and (D), with the labelling budget b as a percentage of the observed set labelled. A budget of 100% is the upper bound on performance. Novel examples are already quite well clustered before any fine-tuning on novel classes, as seen in (A). This demonstrates the ability of deep metric spaces to transfer to novel classes. This property motivates our approach.

is the lower bound on performance. Active learning (*AL*) results are also included, with labelling budgets b of 10% and 100%. The 100% labelling budget is the upper bound on performance, as the entire observed set is labelled. In terms of Recall@1, the pseudo-label approach improves over the initial state by 6.11% and under-performs an active learning approach with 10% labelling budget by just 2.38%. Interestingly, these results indicate that although no true labels are available for the observed set, there is merit in allowing observed examples to be pushed into a better region of the metric space. We

do not expect known class accuracy to improve with this method, but importantly, it does not deteriorate (see final column of Table 5.4).

Figure 5.16 shows a t-SNE visualisation [215] of the novel test set metric space. Compared to the initial feature embedding space, the novel examples are better clustered based on class. This is despite the absence of any true novel class labels being supplied to the algorithm. The visualisation, along with the results in Table 5.4, suggest that spatial relationships in a well structured metric space can be used to generate noisy pseudo-labels to improve the representation of novel classes present in the environment.

5.8 Discussion and Conclusion

Real-world robotic problems are open set and dynamic. Conventional object classifiers are closed set by design and are limited in their ability to detect observations belonging to unknown novel classes. Classifier outputs in robotic problems are not the final goal; predictions made by a classifier can trigger some sort of robotic action. As such, it is often vital that a classifier does not silently fail when observing novel examples. A classifier that knows what it doesn't know, also knows what it needs to learn. Open set recognition enables a vision system to select informative observations for labelling, allowing the model to be updated to better represent the distribution of data in the environment. Conventional closed set classifiers that are limited in their ability to detect novel examples, are also limited in their ability to learn from novel examples.

In this chapter, the suitability of deep metric learning to open-set problems was investigated. We showed how a deep metric learning classification model is well suited to novelty detection and open set recognition. Experimental results show that a deep metric learning approach significantly outperforms conventional softmax-based networks and purpose built novelty detectors in open set recognition problems.

A novel approach to the active learning of previously unknown classes was also proposed. The importance of metric learning to open set active learning problems was demonstrated experimentally. A conventional softmax network with uncertainty-based query selection is outperformed by even random query selection for a metric learning approach, in the majority of cases. A distance-based query selection method was proposed. We showed

how the proposed approach significantly outperforms other commonly used distance-based informativeness measures. The advantage of deep metric learning and the proposed query selection methodology is particularly large with small labelling budgets. As such, this would allow a vision system to efficiently and effectively extend its understanding of the environment beyond the original training distribution. Finally, we showed how spatial relationships in a deep metric space can be used to generate pseudo-labels for observed examples. This enables the model to improve its representation of novel classes with a labelling budget of zero.

Chapter 6

Improved Semantic Segmentation for Robotic Applications

Conventional approaches to the problem of semantic segmentation are inappropriate for robotic applications, as they focus on pixel-level performance and give little significance to spurious object detections. This chapter presents a region-based conditional random field model for semantic segmentation that focuses on object-level performance, recognising that in a robotics context, false object-detections can have costly consequences. We show how optimising at the semantic region-level results in significantly fewer false-positive object detections than conventional approaches. We further show how both object and pixel-level performance can be improved over conventional methods by combining region random fields with dense pixel random fields in a hierarchical manner. An object-aware performance metric is introduced that heavily penalises false positive and false negative object detections, as appropriate for robotic applications. Our approach is evaluated on the challenging NYU v2 and Pascal VOC datasets, outperforming comparable conventional methods in terms of object and pixel-level performance.

Unlike the previous chapters, we do not focus on convolutional neural networks in this chapter. Rather, the CNN is taken as a black box classifier that provides initial pixel-level probability distributions. We propose conditional random field models that refine and significantly improve the initial CNN segmentations by incorporating high-level contextual information. There exists further promising research opportunities in adapting metric learning approaches for pixel-wise feature embedding learning. Additional research opportunities exist in the investigation of open set semantic segmentation

problems. As the initial classifier is treated as a black box, the methodologies proposed in this chapter could also be used in conjunction with such open set metric learning models. These possibilities are discussed in Section 7.2. Note that the research presented in this chapter was conducted and published prior to the research presented in Chapters 4 and 5.

6.1 Motivation

Semantic segmentation is the problem of dividing an image into semantically meaningful regions. This is generally achieved by predicting a pixel-wise labelling, in which each pixel is assigned one out of a set of semantic labels. The semantic labels belong to what are often referred to as "things" or "stuff". Classes that are defined as "things" are usually those with a well defined shape and which can be described as an object. Examples of such classes include chair, table, car or human. On the other hand, classes that are referred to as "stuff" have less well defined, amorphous shapes and often lack objectness. These classes include floor, grass, road and water. Semantic segmentation allows both categories of semantic classes to be predicted together, enabling a rich and detailed representation of the environment.

Semantic segmentation is an important problem in robotic vision, as it simultaneously provides the robot with information about the structure of the environment (e.g. floor, wall, road and grass), as well as information about objects in the environment (e.g. chair, table, car and human), with which the robot may attempt to make affordances. The structural information is often required for the purpose of navigation, for example, manoeuvring through a doorway or avoiding rough terrain, such as grass or gravel. Information about objects is a vital component of robot decision making. For example, interaction with a human begins with the detection and recognition of a human. The same is true for manipulating or avoiding certain objects.

Conventional approaches to semantic segmentation seldom consider the problem from a robotics perspective and as a result give little consideration to robot-specific factors. Additionally, as conventional approaches are not designed with robotic factors in mind, they are not evaluated with metrics that are appropriate for robotic applications. The problems with conventional semantic segmentation approaches, when considered through a robotics lens, are discussed in Section 6.1.1. Details on how the proposed

approach addresses these issues are also presented. Similarly, Section 6.1.2 discusses the problems with conventional evaluation measures and details how our proposed measure addresses these problems.

6.1.1 Semantic Segmentation for Robotic Vision

The design of a semantic segmentation system demands special considerations when examined from a robotics point of view. Conventional approaches generally treat the output segmentation as the end goal and therefore do not consider any requirements of real-world applications in the design process. In robotics, the segmentation itself is not the goal; the segmentation is an enabler of decision-based robotic operation. In a robotics context, any object detection can trigger some sort of costly robotic action. This action may be further computation, as the robot attempts to learn more about the object, such as its pose or instance label. Alternatively, the action may be physical, such as the robot attempting to manipulate an object or navigate over some terrain. In these situations, erroneous object detections can be detrimental to the operation of the robot. As such, approaches for robotics should give a high level of significance to false object detections.

Conventional approaches to semantic segmentation give greatest significance to pixel-level performance. The degree to which an incorrectly labelled image region is penalised is proportional to the size of that region, that is, the number of pixels. In practical applications, the size of an object detection is often irrelevant; a small object may have just as much semantic importance as a large object. Semantic regions of any size can trigger costly robotic action, and as such, a semantic segmentation approach for robotic applications should not optimise performance at the level of pixels. An approach that is suitable for robotics should focus on minimising the number of false positive and false negative object detections, without regard to size.

6.1.1.1 Dense Conditional Random Fields

Many state-of-the-art approaches to semantic segmentation combine the use of a convolutional neural network and a fully connected, or dense, conditional random field (CRF) [182]. A dense prediction convolutional neural network, such as a fully convolutional network (FCN) [80], predicts pixel-wise class probability distributions. The segmentation from the network is

then refined by the conditional random field. A CRF allows the incorporation of contextual and structural information into the inference process, as well as object boundary refinement and the enforcement of local smoothness [90, 185, 186]. An energy function is defined over nodes (such as pixels) in a graphical structure and the CRF inference algorithm aims to find the state (pixel labels) that minimises the energy. The energy function is usually a combination of unary terms, that is, the energy at a single node, and higher order terms, such as the pairwise energy between connected nodes. The interconnection of image subregions in the graph allows the prediction of a smooth and contextually consistent segmentation. A detailed discussion on neural networks for dense prediction can be found in Section 2.3.2 and a discussion on conditional random fields is carried out in Section 2.6.

The conventional densely connected CRF [182] defines a node for every pixel in the input image. A connection, or edge, is defined between every possible pair of pixels. That is, every pixel has an edge to every other pixel in the image. The dense CRF graph is illustrated in Figure 6.1. This structure means that both short range contextual information, which is useful for label smoothness and local consistency, and long range information, such as higher-level contextual information, can be incorporated. An energy function that is defined over a fully connected graph structure is computationally complex to minimise. The authors of [182] make approximate inference tractable and efficient with mean-field approximation and by placing a Gaussian constraint on the pairwise potentials.

Such approaches have shown impressive results in terms of pixel-level performance, particularly with the refinement of object boundaries and other fine-grained details in the image. However, since these CRFs optimise at the level of pixels, the object performance, such as the number of false positive and false negative object detections, is less impressive. Additionally, the effectiveness of such dense approaches on pixel-level performance is diminished when the initial probability distributions, provided by the fully convolutional network, are poor. This may occur when evaluating on challenging datasets with a large number of class labels and a small amount of training data. This is seen in the example segmentation shown in Figure 6.2. The segmentation produced by the neural network contains many regions of spurious labels that the dense CRF struggles to correct. Large regions that are incorrectly labelled by the FCN are particularly difficult for the dense CRF to refine. Since false object detections in a robotics context can result in costly

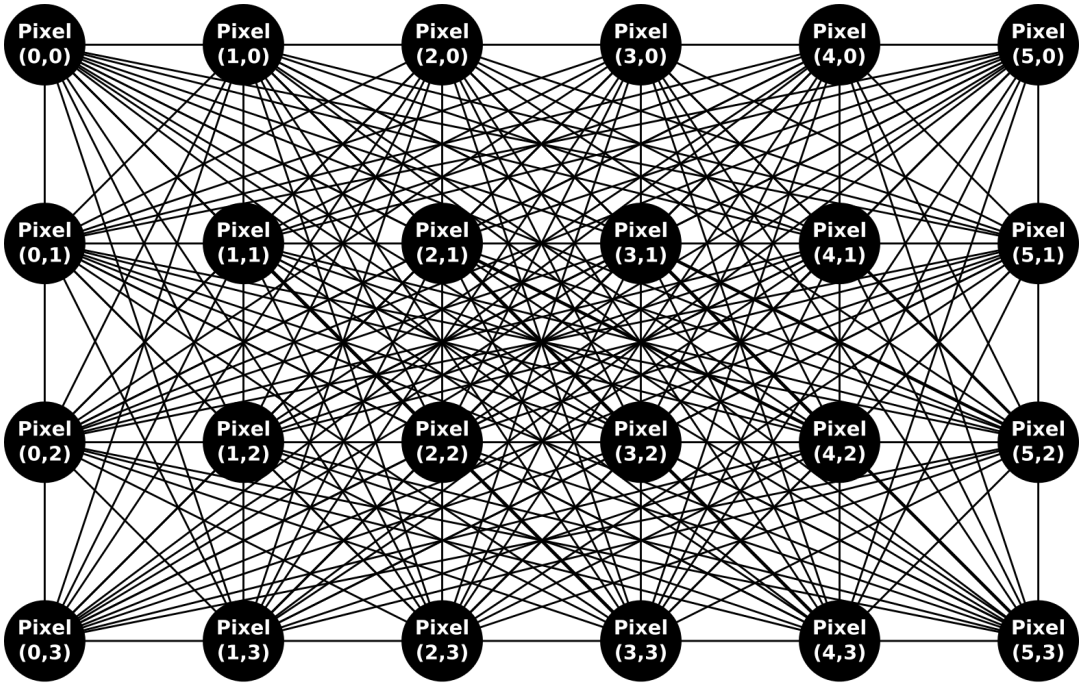


FIGURE 6.1: Structure of a fully connected, dense conditional random field. Each pixel has a node in the CRF and there is a connection from each node to every other node in the graph. For convenience, only a small 6×4 example is shown.

computation or real-world action, we believe that a robotics approach to semantic segmentation should consider object-level performance as important, if not more, than pixel-level performance.

Conditional random fields make a Markov assumption; it is assumed that nodes are independent of other nodes given their neighbours (see Section 3.3.1). This is clearly not the case with CRFs for image segmentation. The errors between pixels or image regions are highly correlated; the factors that lead to an example pixel being incorrectly classified as particular class are very similar to the factors that lead to a nearby pixel also being incorrectly classified as that class. When a large region of an image is incorrectly labelled by a neural network, it is difficult for a pixel-level CRF to correct this mislabelling, as there are a large number of nodes in the graph that believe the same errors. Given enough incorrectly labelled pixels, the nodes consolidate the errors amongst themselves and may become even more certain of the erroneous prediction. The incorrect assumption of independence of errors is a significant contributing factor to the poor performance of densely connected CRFs in terms of the object and region-level performance that is important for robotics.

(A) Image. (B) Ground truth.

(C) Unary (FCN). (D) FCN + dense CRF. (E) Ours.

FIGURE 6.2: (A) Input image from the NYU v2 dataset. (B) Ground truth segmentation. (C) Segmentation produced by the unary classifier used by our models (FCN [80]). (D) Segmentation produced by the commonly used fully connected (dense) CRF [182]. (E) Segmentation produced by our approach.

6.1.1.2 Solution: Region and Hierarchical CRFs

In an effort to limit the amount of non-independent information that is shared between nodes and therefore improve the ability of CRF approaches to refine erroneous labels, we propose a region-based CRF, wherein nodes represent semantically meaningful image regions. Treating each semantically distinct region of an input image as a node allows for easier correction of spurious initial labellings provided by the neural network. This approach places a great deal of significance on higher-order performance, such as object regions, compared to the performance at the pixel-level. Penalising segmentations at the semantic region-level results in a system that favours the reduction of false object detections, over more fine-grained refinements such as object boundary localisation. This is appropriate for robotics, as any false object detection can trigger costly robotic action.

Further to the semantic region CRF, we propose a novel hierarchical CRF that allows additional fine-grained label refinement. We show how a semantic region CRF can be combined with a dense, pixel-level CRF. This formulation combines the best of both CRF structures: minimal false object detections and the conservation of fine-grained image details. Our proposed approach

significantly outperforms a conventional dense CRF in terms of false object detections. The hierarchical CRF also results in an improvement in conventional pixel-level evaluation measures, compared to a standard dense CRF.

A key result of our proposed approach is shown in Figure 6.2. The initial segmentation produced by the fully convolutional network is riddled with false positive object detections. Combing the FCN with a dense CRF results in a small improvement, but the CRF fails to removed many large regions of false positive labellings. Our proposed semantic region CRF significantly reduces the number of false object detections, which we believe makes it more suitable for robotic applications. Further, the inclusion of a pixel-level CRF in a hierarchical model means that fine-grained refinement can occur, if that is required for a particular task.

6.1.2 Evaluating Segmentations for Robotic Vision

Since conventional approaches to semantic segmentation optimise at the level of pixels, performance is also measured and evaluated at the level of pixels. As discussed in Section 6.1.1, pixel-level approaches are not suitable for robotic vision, as any object detection, regardless of size, can trigger some sort of costly robotic action. Therefore, object-aware measures must also be analysed, in addition to conventional evaluation measures, to effectively evaluate an image segmentation for robotic vision. In Section 6.1.2.1, the shortcomings of conventional measures are discussed. These shortcomings motivate our proposed evaluation measure, which is briefly introduced in Section 6.1.2.2.

6.1.2.1 Conventional Evaluation Measures

Commonly used pixel-level measures for evaluating a segmentation include pixel accuracy and mean class intersection-over-union (mIoU). Pixel accuracy is simply the fraction of pixels that are assigned the correct class label. For the mIoU measure, the intersection-over-union is first computed separately for each class label. This is done by dividing the number of pixels that are both classified as a given class and truly belong to that class (intersection), by the number of pixels that are classified as the given class or truly belong to that class (union). The mean class intersection-over-union is found by averaging across class labels. Both of these evaluation metrics lack the ability to

effectively measure the object and region-level performance that is important for robotic applications. In order to demonstrate the limitations of pixel-level evaluation measures, we compute the measures for example segmentations in Section 6.7.4.

6.1.2.2 Solution: Object-Aware Metric

In order to address the problems associated with pixel-level evaluation metrics for robotic vision semantic segmentation, we propose an object-aware evaluation measure. The proposed measure significantly penalises any false positive or false negative object detection. Recognising that object detections of any size can trigger costly robotic action, the number of pixels in an object detection is not considered in the counting of true positives, false negatives and false positives. This avoids the selection of hard thresholds on the overlap amount between predicted and ground truth object regions. The overlap amount, that is, the pixel-level precision and recall, is incorporated in introduced intersection-over-area terms. These terms also ensure that the evaluation measure cannot be gamed by erroneous growing and merging of regions. In Section 6.7.4, we show how our proposed object-aware measure captures information that is important for rigorous evaluation of an image segmentation. We also show how conventional pixel-level measures fail to capture this information.

6.2 Contributions

In this chapter, we present a region-to-pixel hierarchical CRF approach to semantic segmentation for robotic applications. Our approach results in fewer false positive object detections than comparable conventional approaches, as seen in the example result in Figure 6.2. An overview of our approach is shown in Figure 6.3. We also proposed a novel object-aware evaluation measure for semantic segmentation that is appropriate for robotic vision. The main contributions of this chapter have been presented in a peer-review publication [26] and can be summarised as follows:

- We present a performance metric that gives more significance to false positive and false negative object detections compared to pixel-level metrics, as appropriate for robotic applications (Section 6.7).

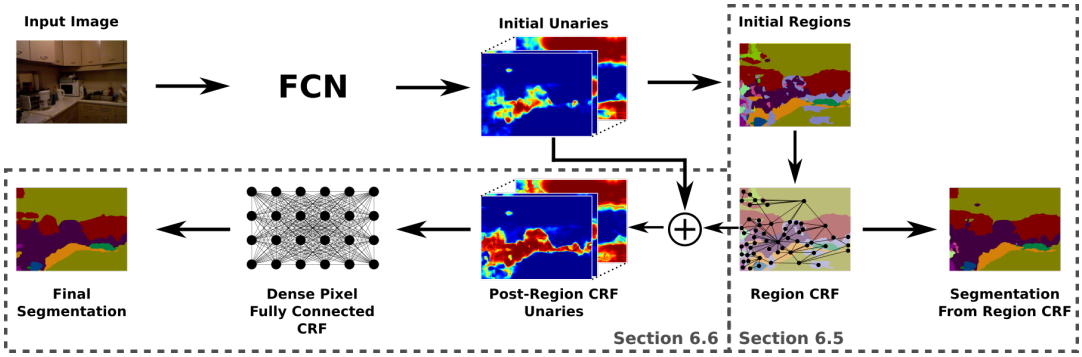


FIGURE 6.3: Overview of our approach. A CNN produces initial pixel-level unaries from which semantic regions can be segmented. Messages from the semantic region-level CRF are propagated to the pixel-level and combined with the original unaries to produce modified unary terms for a dense pixel CRF. The region CRF layer of our hierarchical approach reduces the number of false positive object detections, while the pixel CRF layer refines object boundaries and incorporates fine-grained information to further improve the segmentation.

- We present a conditional random field model that optimises at the semantic region-level, resulting in significantly fewer false positive object detections at a given true positive rate, compared to conventional approaches (Section 6.5).
- We show how a region-level random field can be combined with a pixel-level random field, in a hierarchical manner, to achieve both better object and pixel-level performance than conventional approaches (Section 6.6).
- We show the advantage of using semantically meaningful regions rather than data-driven image regions for improving object-level performance in a region CRF (Section 6.8.3).
- Finally, we demonstrate how our approach is able to perform well on challenging datasets with small numbers of annotated training images (Section 6.8.3).

6.3 Related Work

In this section, we summarise and review conditional random field methods for semantic segmentation. A detailed review of dense and structured prediction literature can be found in Chapter 2.

6.3.1 CNNs and Pixel-level CRFs

Several approaches have been developed to adapt deep convolutional neural networks for dense, pixel-wise prediction. Long *et al.* [80] present a fully convolutional network that replaces the fully connected layers of a traditional CNN with convolutions. This produces a coarse prediction map that is subsequently upsampled in the network, resulting in a dense pixel-level prediction. Extensions of FCN include ones that replace the upsampling layers with a full decoder network [81, 82, 83, 84, 85, 86, 87]. Further approaches incorporate dilated or atrous convolutions [89, 90, 91, 92, 93] into the network to avoid the use of downsampling pooling layers.

Conditional random fields [13] in the context of semantic segmentation allow contextual and structural information, such as class label compatibility and spatial smoothness, to be elegantly incorporated into the inference process. Shotton *et al.* [178] employ a four-connected adjacency CRF to achieve local label smoothness, however, such a formulation is limited in terms of incorporating longer range information. A fully connected, or dense CRF [182], models long range contextual information by defining connections between every pair of pixels. Inference is made tractable by using mean-field approximation. The pairwise potentials are constrained to be Gaussian, however, Campbell *et al.* [184] relax this constraint by learning non-parametric potentials directly from the training data and encoding the desired potentials as Gaussian kernels. These models are able to refine object boundaries and enforce spatial smoothness. However, they are less successful at removing many regions of spurious labels and their effectiveness is further diminished when the unary terms are poor. We address this problem in Section 6.5.

The representational power of CNNs is combined with the contextual modelling of CRFs in [90]. A fully connected CRF is used as a post processing step after a CNN. Zheng *et al.* [185] formulate a CRF as layers in the neural network, allowing end-to-end joint learning of the CNN weights and CRF parameters, while higher order potentials are included by Arnab *et al.* in [187].

A related but different task to semantic segmentation is simultaneous detection and segmentation [78, 223, 224], which aims to detect and densely label individual object instances. In this work, however, we focus on improving CRF technologies for conventional semantic segmentation, which can be a precursor to full instance segmentation.

6.3.2 Region-level CRFs

Previous approaches that incorporate CRFs at the image region-level generally use regular super-pixels as regions [225, 226], or larger data-driven and contour-aware regions [227, 228]. In contrast to these approaches, we use semantically meaningful regions derived from the unary potentials computed by a CNN. We find such an approach, rather than using data-driven regions such as super-pixels or clustered super-pixels, is able to achieve better object-level performance. Furthermore, our use of regions is not in pursuit of computational efficiency, but as an approach to limit the amount of dependant information shared between related pixels; computational efficiency is a welcome by-product of this.

6.3.3 Hierarchical CRFs

We propose combining a region-level CRF with a fully connected pixel-level CRF in a hierarchical manner. Several other pieces of work involve hierarchical, or multi-scale, CRFs but contain key differences to our approach. Kumar and Herbert [229] present a two-layer hierarchical CRF that first performs inference at the pixel-level, before passing information via directed edges to the region layer of the CRF. Unlike [229], our approach first performs inference at the semantic region-level, before refinement takes place in the pixel layer. Unary potentials in [229] are from hand crafted features, while we use deep CNNs to obtain initial distributions. Moreover, the primary purpose of the pixel layer of the CRF in [229] is to enforce local smoothness and as such it is formulated as an adjacency CRF. In contrast, our approach uses a fully connected (dense) CRF at the pixel layer to employ long range contextual information.

The hierarchical CRF proposed by Ladický *et al.* [230] combines features from different image quantisation levels (pixels, segments and super-segments) into a single random field model. Graph cut techniques are used to minimise the energy, taking between 6 and 20 seconds for a single image [231]. The energy is minimised across all layers of the graph jointly. Our approach, however, minimises the energy at the semantic region-level first, with no connections or interactions with the individual pixels. After this first stage of inference, we propagate region-level messages to the original pixel-level unaries, obtaining modified unary terms for the pixel layer of the CRF. Such

an approach gives more significance to spurious regions of labels, regardless of the size of those regions. This greatly reduces the number of false positive object detections, as appropriate for robotic applications, while also refining the pixel-level performance.

6.4 Conditional Random Fields

In this section, we give an overview of conditional random fields for semantic segmentation and introduce the notation used for the remainder of the chapter. Background information on conditional random fields can be found in Section 3.3.

6.4.1 Notation

A set of random variables $\{X_1, \ldots, X_N\}$ defines a random field $\mathbf{X}$, in which each random variable is defined over a set of class labels $\mathcal{L} = \{1, 2, ..., L\}$. In the case of image segmentation, N may be the number of pixels in the image or the number of larger image regions to be represented in the model. An assignment $\mathbf{x}$ to $\mathbf{X}$, where all $x_i \in \mathcal{L}$, is a valid segmentation.

Given a graph structure G that defines the interactions between nodes (given by $\mathbf{X}$), a CRF works to find the assignment of $\mathbf{x}$ to $\mathbf{X}$, or the labelling, that maximises the following distribution:

$$\Pr(\mathbf{X} = \mathbf{x} \mid \mathbf{I}) = \frac{1}{Z(\mathbf{I})} \exp(-E(\mathbf{x} \mid \mathbf{I})), \tag{6.1}$$

where $\mathbf{I}$ is the image, $E(\mathbf{x} \mid \mathbf{I})$ is the energy of the labelling and $Z(\mathbf{I})$ is the partition function that normalises the distribution. The energy $E(\mathbf{x} \mid \mathbf{I})$ is defined as the sum over clique cost functions that describe the cost or potential of a particular labelling.

6.4.2 Pairwise Conditional Random Fields

A commonly used and simple graph structure is a pairwise CRF, in which only two types of cliques are defined: unary and pairwise. Unary potentials are the cost of assigning a label to a particular node when considering no interaction with other nodes. Pairwise potentials are the cost associated

with assigning labels to two nodes that interact via an edge in the graph G. The energy to be minimised in order to find the best labelling in a pairwise random field is:

$$E(\mathbf{x} \mid \mathbf{I}) = \sum_{i \in N} \psi_i(x_i) + \sum_{\substack{i \in N \\ j \in \boldsymbol{\mathcal{E}}_i}} \phi_{ij}(x_i, x_j), \tag{6.2}$$

where $\psi_i(.)$ is the unary potential, or cost, at node i, $\phi_{ij}(.,.)$ is the pairwise cost between nodes i and j, and $\boldsymbol{\mathcal{E}}_i$ is the set of nodes connected to node i via direct edges.

The unary potentials are provided by some classifier that produces distributions over class labels. All unary potentials discussed in the remainder of this chapter are from a fully convolutional network [80], which is often used as a semantic segmentation baseline for comparison. FCN produces pixel-level distributions that result in generally smooth labellings, devoid of fine-grain noise. A more detailed discussion on fully convolutional networks can be found in Section 2.3.2.1. Pairwise potentials can be hand crafted based on prior knowledge of likely label configurations or can be learned by standard optimisation approaches, as discussed in Section 6.5.4.

6.5 Semantic-Region CRF

This section details our proposed region-based conditional random field for semantic segmentation.

6.5.1 Overview and Motivation

Rather than defining a node for each pixel in our proposed region CRF, an initial segmentation is used to extract regions of the image that are each represented in the model by a node. Inference is carried out at the region-level, but the final belief may be propagated back down to the pixels. We define edges between neighbouring regions and use the pairwise energy definition in Equation 6.2. The structure of the region CRF can be seen in Figure 6.4.

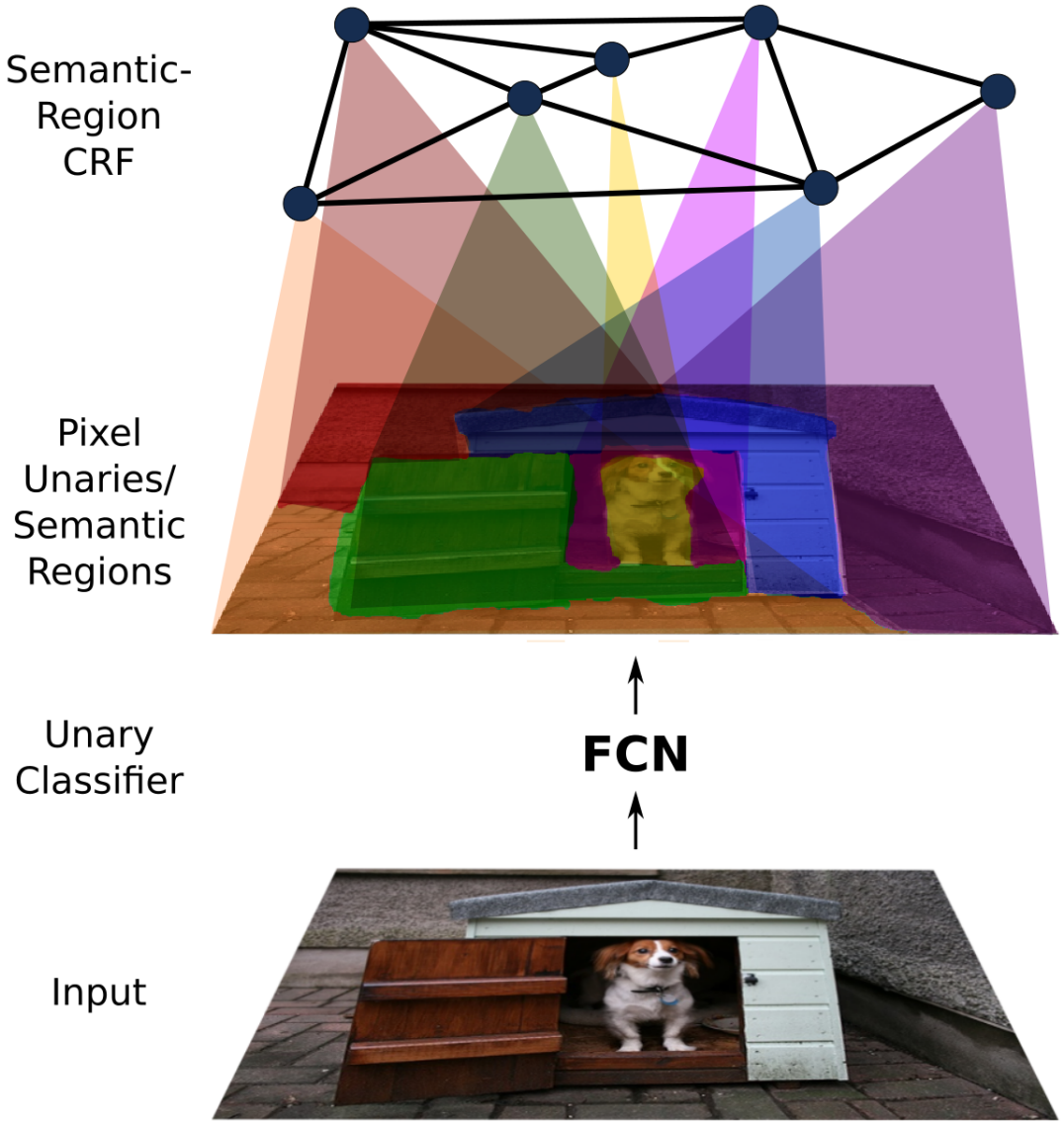


FIGURE 6.4: Overview of the proposed semantic region CRF. A unary classifier, such as a fully convolutional network, produces a per-pixel unary distribution and an initial segmentation. The initial unary regions are regions of semantic significance, according to the unary classifier. Each semantic region has a corresponding node in the region CRF. Region-level unary terms are computed from the underlying pixel unaries.

Such a formulation allows many erroneous regions of pixels to correct their labels, which would otherwise not have been possible in a dense CRF framework due to the propagation and consolidation of errors amongst semantically dependant pixels. Our approach coincides well with the needs of a robotic vision system, as described in Section 6.1.1, since the presence of false positive or false negative object detections can have costly consequences in a robotic application.

6.5.2 Initial Regions and Clique Potentials

An initial segmentation of the image must be obtained in order to construct the random field. This segmentation provides the image regions that will be represented as nodes in the model. The approach can use any method to obtain these initial regions, however, we propose to use the semantic regions from the segmentation produced by the unary classifier (FCN). This is illustrated in Figure 6.4. Such an approach means that the interactions in the model are between semantically distinct image regions, limiting the amount of dependent information shared between nodes. This is in comparison to using a data-driven segmentation that finds image regions based only on pixel intensities, such as super-pixels or clustered super-pixels. As with pixels, using data-driven initial regions that are not based on semantic information, results in propagation of errors between semantically non-independent image regions. This limits the ability of the random field to correct spurious labellings. Section 6.8.3 investigates the effect of using a data-driven approach to obtain an initial segmentation compared to using semantic regions from the unary classifier.

Per region unary distributions are calculated by integrating over the underlying pixel unary distributions (from FCN) in each region. We denote the region-level potentials with superscript (r) and the original pixel unaries from FCN with superscript (FCN). The region-level unary potential for region i is:

$$\psi_i^{(r)}(x_i) = -\ln\left(u_i^{(r)}(x_i)\right), \tag{6.3}$$

where $u_i^{(r)}(.)$ is the normalised region probability distribution for region i, calculated from the underling FCN pixel probabilities.

The region pairwise potential between connected regions i and j is:

$$\phi_{ij}^{(r)}(x_i, x_j) = -\ln\left(f(x_i, x_j)\right), \tag{6.4}$$

where $f(.,.)$ is a transitional probability function between class labels. The parameters of this function enable high-level contextual information to be incorporated into the inference process. Learning of these parameters is discussed in Section 6.5.4. Region-level ground truth distributions are computed by integrating over the underlying pixel ground truths.

6.5.3 Energy Minimisation

Finding the labelling that minimises the pairwise energy defined in Equation 6.2 is intractable. We use loopy belief propagation [177] to find an approximate solution, by passing messages between nodes across edges. The messages contain information from the unary distribution at the sender node, the pairwise cost between the two connected nodes, and previous messages received by the sender node. Messages are propagated until convergence, at which point the belief may be read out. The belief at a node can be normalised to obtain a probability distribution over classes. The message update equation is:

$$m_{ij}^{t+1}(\ell) = \sum_{\ell'=1}^{L} \left(f(\ell, \ell') u_i^{(r)}(\ell') \prod_{k \in \boldsymbol{\mathcal{E}}_i \setminus j} m_{ki}^{t}(\ell') \right), \tag{6.5}$$

where $m_{ij}^{t+1}(\ell)$ is the message from node i to node j for label ℓ, t is the time step, L is the total number of class labels and $\boldsymbol{\mathcal{E}}_i \setminus j$ is the set of nodes directly connected to node i via an edge, excluding the receiver node j. Previous messages sent by the receiver node are ignored such that information that was originally sent from that node isn't received back. The unary and pairwise terms, $u_i(.)$ and $f(.,.)$, are described in Equations 6.3 and 6.4, respectively. Equation 6.5 is the message formula for the sum-product variant of the belief propagation algorithm. We find this approach works well in our experiments, however, it should be noted that a max-product variant is also commonly used.

A caveat of belief propagation between regions is that a very small region can have undue influence over the state of very large regions. Although this disregard for region size is by design, and is indeed part of why our approach performs well at the object-level, excessive influence from very small regions is not desired. A scale factor is introduced to the belief read out in order to quell this effect. The scaling term is $\min\left(\frac{n_k}{n_j}, 1\right)$, where n_k and n_j are the number of pixels belonging to the sender and receiver nodes, respectively. This term ensures that very small regions cannot overly influence very large regions. The saturation at a value of one ensures that larger regions do not dominate the information being propagated around the graph.

The log-belief at region j after message passing is:

$$\ln\left(b_j(\ell)\right) = \psi_i^{(r)}(\ell) + w \sum_{k \in \boldsymbol{\mathcal{E}}_j} \ln\left(m_{kj}^T(\ell)\right) \ \min\left(\frac{n_k}{n_j}, 1\right), \tag{6.6}$$

where b_j is the belief of label ℓ at region j, w dictates the level of influence of incoming messages and T is the final time step of the belief propagation algorithm. Normalising the belief at a region results in the probability distribution over labels.

Since our approach passes messages between regions that are comprised of pixels, the belief can be read out at either the region-level, changing the labels of all underlying pixels together (Equation 6.6), or at the pixel-level, allowing individual pixels to change labels independently (Equation 6.7). Belief read out at the region-level naturally results in fewer false positive object detections, however, propagation to the pixel-level is important for initialising the dense layer of the hierarchical CRF discussed in Section 6.6, allowing refinement of both the pixel and object-level labelling.

Let p be a pixel belonging to region j. The pixel belief, denoted as b_p, is found by combining the FCN pixel unary potential $\psi_p^{(FCN)}$, with the region-level messages. This is seen in Equation 6.7. Again, the probability distribution over class labels at pixel p is found by normalising the belief.

$$\ln\left(b_p(\ell)\right) = \psi_p^{(FCN)} + w \sum_{k \in \boldsymbol{\mathcal{E}}_j} \ln\left(m_{kj}^T(\ell)\right) \ \min\left(\frac{n_k}{n_j}, 1\right) \tag{6.7}$$

6.5.4 Learning Pairwise Potentials

The pairwise terms allow contextual and structural information to be modelled in the random field. In order to obtain good pairwise potentials, we learn the parameters using the first order optimisation approach of gradient descent. Full details on the gradient descent algorithm can be found in Section 3.1.1. We define the objective function $J(\mathbf{F})$ as the log sum error over the nodes (regions) in the graph, where $\mathbf{F}$ is the set of pairwise potential parameters (e.g. $f(\ell, \ell')$). Minimising the objective function involves differentiating the belief from Equation 6.6, after normalisation.

The update step of gradient descent allows the parameters to become either positive or negative. As the matrix-vector multiplication of these parameters with the unary terms (Equation 6.5) is carried out in the probability domain,

and since negative probabilities are meaningless, it does not make sense for the pairwise parameters to become negative. To alleviate this problem, the pairwise terms are reparameterised in the log-space, e.g. $f(\ell, \ell') = e^{\theta(\ell,\ell')}$. The objective function is minimised with respect to $\boldsymbol{\theta}$ and all updates are performed in the log-space. The effect of this on the derivatives of the objective function is seen in Equation 6.8. This reparameterisation allows the optimisation algorithm to operate in a meaningful way, ensuring that all $f(\ell, \ell')$ remain positive.

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta(\ell, \ell')} = \frac{\partial J(\boldsymbol{\theta})}{\partial f(\ell, \ell')} \frac{df(\ell, \ell')}{d\theta(\ell, \ell')} = \frac{\partial J(\boldsymbol{\theta})}{\partial f(\ell, \ell')} f(\ell, \ell') \tag{6.8}$$

6.6 Hierarchical Region-Pixel CRF

In order to further improve the semantic segmentation produced by the region CRF from Section 6.5, we propose a region-to-pixel hierarchical CRF that allows fine-grained pixel refinement after inference in the region CRF.

6.6.1 Structure

Our hierarchical CRF consists of two layers: the first is the region-level CRF model discussed in Section 6.5 and the second is a dense pixel-level, fully connected CRF. Inference is first carried out in the region layer as described in Section 6.5.3 and the pixel beliefs are used to calculate the unary potentials for the dense pixel layer. Inference is then performed in the pixel layer to produce the final labelling. The region layer of the model is used to optimise region and object-level performance, particularly by reducing the number of false positive object detections. A standalone dense CRF is far less effective at this task, as discussed in Section 6.1.1.1. The pixel layer has edges connecting every pixel with every other pixel in the image. This formulation is effective at incorporating rich contextual information and carrying out refinement at a finer level than the region CRF alone. Inference is carried out in the pixel layer using mean-field approximation, as described in [182].

6.6.2 Fully Connected Pixel Layer

The dense CRF model follows the pairwise graph energy described in Equation 6.2. Since a node is defined for every pixel, the subscripts p and q are used for nodes in the dense CRF layer in order to differentiate from the i and j subscripts used for regions in the region CRF layer. To distinguish between the potentials from the region CRF, let the unary and pairwise potentials of the dense layer be $\psi_p^{(d)}(x_p)$ and $\phi_{pq}^{(d)}(x_p, x_q)$, respectively.

6.6.2.1 Pixel Layer Unary Potentials

Region-level messages from Section 6.5.3 are propagated down to pixels, as described in Equation 6.7, to obtain pixel beliefs. The dense layer unary potential is the normalised pixel belief after inference in the region CRF layer, as seen in Equation 6.9.

$$\psi_p^{(d)}(x_p) = -\ln\left(\frac{b_p(x_p)}{\sum_{\ell=1}^{L} b_p(\ell)}\right) \tag{6.9}$$

6.6.2.2 Pixel Layer Pairwise Potentials

Inference in fully connected CRFs with general potentials is intractable, as such the pairwise potentials are constrained to be Gaussian kernels. This allows efficient approximate mean-field inference using high-dimensional convolutions, as detailed in [182]. The pairwise potentials have the following form:

$$\phi_{pq}^{(d)}(x_p, x_q) = \mu(x_p, x_q) \sum_{a=1}^{A} w_a k_a(\mathbf{v}_p, \mathbf{v}_q), \tag{6.10}$$

where $\mu(.,.)$ is a label compatibility function, $k_a(.,.)$ is a Gaussian kernel, $\mathbf{v}_p$ is a feature vector for pixel p and w_a is a weight parameter for kernel k_a. The label compatibility takes the form of a Potts model, $\mu(x_p, x_q) = [x_p \neq x_q]$, which equals one when the connected nodes are assigned different labels. This means that the CRF will encourage pixels p and q to take the same label when the sum of the kernel outputs for that connection is large.

We make use of three kernels, two of which follow the formulation in [182]: a local spatial smoothness kernel (Equation 6.11), which encourages nearby pixels to take the same label, and an appearance kernel (Equation 6.12),

which encourages pixels within a larger radius that have similar appearance to take the same label. These kernels are represented as:

$$k_1(\mathbf{v}_p, \mathbf{v}_q) = \exp\left(-\frac{\left\|\boldsymbol{\rho}_p - \boldsymbol{\rho}_q\right\|^2}{2\sigma_\alpha^2}\right), \tag{6.11}$$

$$k_2(\mathbf{v}_p, \mathbf{v}_q) = \exp\left(-\frac{\left\|\boldsymbol{\rho}_p - \boldsymbol{\rho}_q\right\|^2}{2\sigma_\beta^2} - \frac{\left\|\mathbf{I}_p - \mathbf{I}_q\right\|^2}{2\sigma_\gamma^2}\right), \tag{6.12}$$

where $\boldsymbol{\rho}_p$ is the position and $\mathbf{I}_p$ is the colour vector of pixel p. The parameters σ_α, σ_β and σ_γ control the degree of closeness in terms of spatial or colour distance considered to be important. For example, the value of σ_α is usually small, as label smoothness should only be encouraged between very nearby pixels. The parameter σ_β is generally larger, so that the appearance of pixels within a large radius have some influence over the final labelling of the pixel. Selection of the parameters is discussed in [182].

When the beliefs following the region CRF inference are read out at the region-level, as in Equation 6.6, a new segmentation is produced. These new regions represent contiguous portions of the image that are believed, according to the region CRF, to largely belong to the same object. To further encourage underlying pixels in these new regions to take the same final label, we introduce a higher order term that operates at the level of regions. This term can be implemented as a pairwise kernel:

$$k_3(\mathbf{v}_p, \mathbf{v}_q) = \exp\left(-\frac{\left\|r_p - r_q\right\|^2}{2\sigma_\zeta^2}\right), \tag{6.13}$$

where r_p is the index number of the region to which pixel p belongs. Since this kernel is meant only to encourage pixels from the same region to take the same label, σ_ζ is set to be near zero. This means that the exponentiated term will be a negative number with a large magnitude when the two pixels belong to different regions. As such, the influence of the kernel will disappear. We find that incorporating this soft constraint over regions results in better object-level performance.

6.7 Object-Aware Evaluation Metric

Conventional evaluation metrics for semantic segmentation are based on pixel-level performance. These metrics are insufficient to effectively evaluate an approach for real-world problems, such as robotic vision, as they fail to give significance to object-level performance. Pixel-level evaluation metrics assume that all pixels in an image have equal semantic significance. For example, the correcting of some small number of incorrectly labelled pixels for a given class would result in the same increase in performance, regardless of the context of those pixels. In pixel-level metrics, refining the boundaries of an already well segmented object is given the same importance as recognising some pixels as belonging to a previously undetected object. To a robot, the information gain from detecting the previously unseen object is significantly greater than the information gain from a small refinement of boundary pixels. As such, an evaluation measure for robotic vision should give great significance to object-level false positive and false negative performance.

In this section, we propose an object-aware performance metric for semantic segmentation that is appropriate for robotic vision. Unlike conventional pixel metrics, our object-aware metric heavily penalises all false positive and false negative object detections, regardless of size. We show that the proposed measure captures information that pixel-level evaluation measures do not. The proposed measure should be used in conjunction with pixel-level measures to provide a better overall gauge of the performance of a semantic segmentation system.

6.7.1 Object-Aware False Positives and False Negatives

The first step of computing the proposed evaluation metric is finding the object-aware true positives, false negatives and false positives. In order to compute these values, the ground truth label image, ground truth object instance masks and the predicted label image are required. The ground truth label image provides a pixel-wise true labelling, while the predicted label image provides the pixel-wise label predictions from the semantic segmentation approach that is under evaluation. The ground truth object instance masks label each instance of a semantic class separately, allowing object-level performance to be measured.

The predicted label image is segmented into spatially contiguous regions of the same predicted class. We define $\mathcal{R} = \{R_1, \ldots, R_a\}$ as a family of subsets of the input image space $\mathcal{I}$. Each subset $R_i \in \mathcal{R}$ is a predicted region of pixels and the number of predicted regions is a. The predicted label for each region is defined as $\hat{\mathbf{z}} = [\hat{z}_1, \ldots, \hat{z}_a] \in \mathcal{L}$. Similarly, we define $\mathcal{O} = \{O_1, \ldots, O_b\}$ as another family of subsets of the input image space $\mathcal{I}$, where each subset $O_i \in \mathcal{O}$ is a set of pixels belonging to a ground truth object instance. The number of true object instances is b and the true label for each instance is defined as $\mathbf{z} = [z_1, \ldots, z_b] \in \mathcal{L}$.

In the following equations, the Iverson brackets $[condition]$ evaluate to a value of one if *condition* is true, otherwise, the brackets evaluate to a value of zero. The example condition $\exists\lambda \in \Lambda \colon Q(\lambda)$ is true if there exists at least one λ in the set Λ such that $Q(\lambda)$ is true. The condition $\nexists\lambda \in \Lambda \colon Q(\lambda)$ is true if there does not exist any λ in the set Λ such that $Q(\lambda)$ is true. The operators $\vee$ and $\wedge$ represent *logical or* and *logical and*, respectively. The empty set is denoted as $\emptyset$. Object-aware true positives, false negatives and false positives are defined and calculated as follows:

- *True positives (tp)*: The number of ground truth object instances that have an overlap with a predicted region of the correct class.

$$tp = \sum_{i=1}^{b} [\exists R_j \in \mathcal{R} \colon (O_i \cap R_j \neq \emptyset) \wedge (z_j = \hat{z}_i)] \tag{6.14}$$

- *False negatives (fn)*: The number of ground truth object instances that do not have an overlap with a predicted region of the correct class.

$$fn = \sum_{i=1}^{b} [\nexists R_j \in \mathcal{R} \colon (O_i \cap R_j \neq \emptyset) \wedge (z_i = \hat{z}_j)] = b - tp \tag{6.15}$$

- *False positives (fp)*: The number of prediction regions that do not have any overlap with a ground truth instance of the same class.

$$fp = \sum_{i=1}^{a} [\nexists O_j \in \mathcal{O} \colon (R_i \cap O_j \neq \emptyset) \wedge (\hat{z}_i = z_j)] \tag{6.16}$$

Equations 6.14, 6.15 and 6.16 compute the counts of true positives, false negatives and false positives globally across all class labels. However, the counts can also be computed on a per-class basis. In Section 6.8, we evaluate both

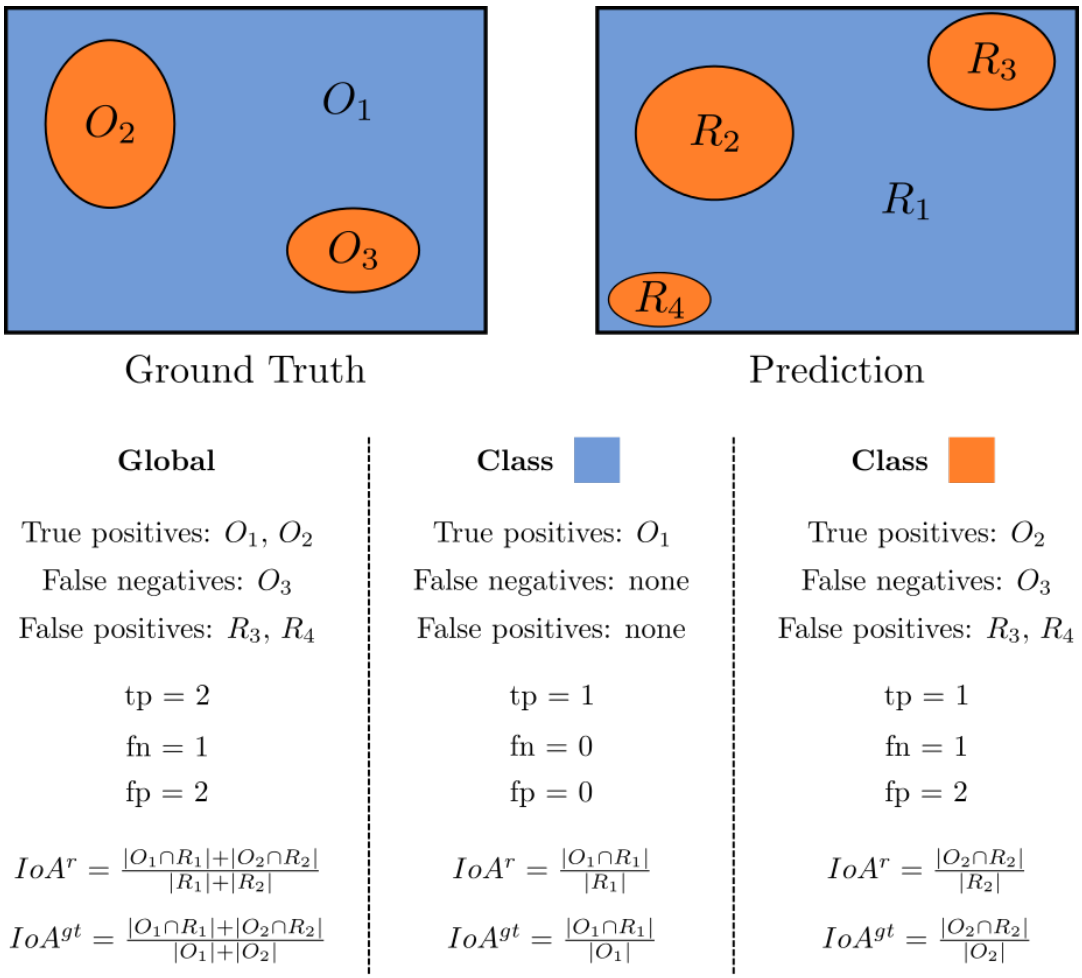


FIGURE 6.5: Computation of the object-aware true positives, false negatives and false positives, as well as the intersection-over-area terms for a simple example. There are three ground truth regions and four predicted regions.

with global and mean class measures. A simple example is shown in Figure 6.5, illustrating how global and per class true positives, false negatives and false positives are found.

6.7.2 Object-Aware Precision and Recall

Precision and recall measures can be computed from the object-aware true positives, false negatives and false positives. However, these measures alone are not sufficient to evaluate performance, as the erroneous growing and merging of predicted regions of the same class is always beneficial or neutral for that class, in that it will potentially reduce false positives with no impact on false negatives. To alleviate this issue and ensure that the measure cannot be gamed by such methods, an intersection-over-area (IoA) term is

introduced to the precision (Equation 6.17) and recall (Equation 6.18) calculations. Again, these measures can be computed globally across all classes or on a per class basis. For notational convenience, Equations 6.17-6.20 describe the global measures. Figure 6.5 illustrates the per class calculation.

$$P^{IoA} = \frac{tp\ IoA^r}{tp + fp} \tag{6.17}$$

$$R^{IoA} = \frac{tp\ IoA^{gt}}{tp + fn} \tag{6.18}$$

The IoA^r term for a given class is the intersection of the predicted region and ground truth instance, divided by the area of the predicted region. The IoA^{gt} term is the intersection of the predicted region and ground truth instance, divided by the area of the ground truth instance. These are calculated as:

$$IoA^r = \frac{\sum_{i=1}^{a}\sum_{j=1}^{b}|R_i \cap O_j|\ \ [z_i = \hat{z}_j]}{\sum_{k=1}^{a}|R_k|\ \ [R_k \text{ is not a false postive}]}, \tag{6.19}$$

$$IoA^{gt} = \frac{\sum_{i=1}^{a}\sum_{j=1}^{b}|R_i \cap O_j|\ \ [z_i = \hat{z}_j]}{\sum_{k=1}^{b}|O_k|\ \ [O_k \text{ is not a false negative}]}, \tag{6.20}$$

where $|R_k|$ is the size of the region R_k, that is, the number of pixels in the region, and $[R_k$ is not a false postive$]$ is equal to one if region R_k is not a false positive and equal to zero if it is a false positive. These IoA measures are calculated over the entire image. P^{IoA} and R^{IoA} are the object-aware precision and recall, marked with IoA to distinguish them from conventional precision and recall. Including the ratio term between the prediction and ground truth overlap area, and the total area of the predicted region (for precision) or ground truth instance (for recall), ensures that the metric cannot be gamed by the erroneous growing and merging of regions. The computation of the intersection-over-area terms is shown for a simple example in Figure 6.5, both globally and on a per class basis.

A simple alternative to the intersection-over-area (IoA) terms would be to set a threshold on the intersection size. True positives would only be counted if the overlap between the predicted region and ground truth instance was larger than some threshold. However, the setting of such thresholds is difficult. Introducing the intersection-over-area terms allows us to avoid the selection of hard thresholds. True and false detections are determined based on any overlap between predicted regions and ground truth instances, with the IoA measures addressing the issue of the size of the overlap.

6.7.3 Object-Aware F-Measure

When evaluating a classification system, it is often useful to combine precision and recall into a single value for easy comparison. The F-measure considers both the precision and recall, allowing the importance of each to weighted. Our general object-aware F-measure is defined as:

$$F_{\beta}^{IoA} = (1+\beta^2)\frac{P^{IoA}R^{IoA}}{(\beta^2 P^{IoA} + R^{IoA})}, \tag{6.21}$$

where β weights the importance of precision and recall. When $\beta > 1$, more emphasis is placed on false negatives, weighting the importance of recall higher than precision. When $\beta < 1$, less influence is placed on false negatives and the importance of precision is weighted higher than recall.

The relative importance of precision and recall can be difficult to determine and is likely to be specific to the application. In Section 6.8, we consider only the equal weighting of precision and recall by analysing the F-measure when β is equal to one. The formula for our object-aware F_1-measure is shown in Equation 6.22. This metric should be considered along with conventional pixel metrics to better understand the performance of an approach.

$$F_{1}^{IoA} = 2\frac{P^{IoA}R^{IoA}}{(P^{IoA} + R^{IoA})} \tag{6.22}$$

6.7.4 Example Evaluations

In this section, we evaluate example segmentations using pixel-level measures and the proposed object-aware measures. These examples illustrate the importance of object-level performance when evaluating a segmentation.

6.7.4.1 Conventional Pixel Measures

We compare our proposed object-aware metric to two conventional pixel-level evaluation metrics. The two measures are described below, where M is the total number of pixels, L is the number of class labels, $\mathbf{y} = [y_1, \ldots, y_M]$ are the true pixel labels and $\hat{\mathbf{y}} = [\hat{y}_1, \ldots, \hat{y}_M]$ are the predicted pixel labels.

- **Pixel Accuracy (Pixel Acc.)**:

$$\text{Pixel Acc.} = \frac{1}{M} \sum_{i=1}^{M} [y_i = \hat{y}_i] \tag{6.23}$$

- **Mean Class Intersection-over-Union (Pixel mIoU)**:

$$\text{Pixel mIoU} = \sum_{\ell=1}^{L} \left(\frac{\sum_{i=1}^{M} [(y_i = \ell) \wedge (\hat{y}_i = \ell)]}{\sum_{j=1}^{M} [(y_j = \ell) \vee (\hat{y}_j = \ell)]} \right) / L \tag{6.24}$$

6.7.4.2 Example Segmentations

Figure 6.6 shows the example segmentations. The scene, which is shown in Figure 6.6A, consists of three semantic instances from two classes: one table and two coffee mugs. A ground truth segmentation is seen in Figure 6.6B. Example segmentations are shown in Figures 6.6C-6.6F. The segmentations in Figures 6.6C and 6.6D contain the same number of correctly classified pixels per class. Similarly, the segmentations in Figures 6.6E and 6.6F contain the same number of correctly classified pixels per class.

Although the number of correctly labelled pixels is the same in Figures 6.6C and 6.6D, the segmentations are significantly different. The segmentation in Figure 6.6C completely misses one of the mugs, while the segmentation in Figure 6.6D has correctly labelled pixels for both coffee mug instances. From a robotic vision perspective, the segmentation in Figure 6.6D is much more informative of the scene, as every semantic instance is captured. Compared to Figure 6.6C, less of the coffee mug on the left-hand-side is correctly labelled. However, the information gained by detecting some of the right-hand-side coffee mug is greater than that of improving the labelling of an already detected instance. This is because the presence or absence of an object in a scene can trigger robotic action.

Again, the segmentations in Figures 6.6E and 6.6F are significantly different, despite the same number of correctly labelled pixels. The segmentation in Figure 6.6E contains one false positive detection of a coffee mug, while the segmentation in Figure 6.6F contains three false positive detections of coffee mugs. Since any false positive object detection can trigger some sort of robotic action, false positives should be minimised for robotic vision applications. As such, the segmentations in Figures 6.6E and 6.6F should not be evaluated to the same performance level.

TABLE 6.1: Conventional pixel-based evaluation metrics compared to the proposed object-aware metrics on the example segmentations from Figure 6.6.

	Pixel Acc.	Pixel mIoU	Gbl. P^{IoA}	Gbl. R^{IoA}	Gbl. F_1^{IoA}	Mean P^{IoA}	Mean R^{IoA}	Mean F_1^{IoA}
Ex. 1	0.9594	0.6411	0.9594	0.6396	0.7675	0.9793	0.5809	0.6287
Ex. 2	0.9594	0.6411	0.9594	0.9594	0.9594	0.9793	0.6618	0.7340
Ex. 3	0.9305	0.5738	0.4652	0.9305	0.6203	0.5591	0.6465	0.5891
Ex. 4	0.9305	0.5738	0.6978	0.9305	0.7975	0.6127	0.6465	0.6283

(A) Image.

(B) Ground truth.

(C) Example segmentation 1 (Ex. 1).

(D) Example segmentation 2 (Ex. 2).

(E) Example segmentation 3 (Ex. 3).

(F) Example segmentation 4 (Ex. 4).

FIGURE 6.6: Example segmentations of an input image.

6.7.4.3 Evaluating the Example Segmentations

Table 6.1 evaluates the four example segmentations from Figure 6.6. Conventional pixel-based metrics are shown, as well as the proposed object-aware measures. Note that the mean class object-aware F-measure is computed by finding the mean of the per class F-measures, not by using the mean class recall and precision. Looking first at Example 1 (Figure 6.6C and *Ex. 1* in Table 6.1) and Example 2 (Figure 6.6D and *Ex. 2* in Table 6.1), it can be seen that the pixel-level evaluation metrics are the same for both segmentations. Any difference in the quality of the segmentations is not recognised by the pixel-level metrics. The proposed object-aware metrics, however, differ for the two segmentations. The object-aware recall measures are greater for Example 2, since Example 1 completely misses one of the coffee mugs, while Example 2 detects both. Consequently, the object-aware F_1 measure is also greater for Example 2. The object-aware precision measures are the same, since there are no false positives in either example.

For Example 3 (Figure 6.6E and *Ex. 3* in Table 6.1) and Example 4 (Figure 6.6F and *Ex. 4* in Table 6.1), the pixel-level metrics again fail to recognise any difference in quality and report the same values for each example. Conversely, the proposed object-aware evaluation measures recognise the difference in the segmentations. The object-aware precision is greater for Example 4, as Example 3 has more object-aware false positives. This results in a larger F_1 measure for Example 4. Since all objects are correctly detected in both examples, the recall measures are unchanged. These examples show that our proposed evaluation metrics place great significance on object-level performance. The proposed measures should be used alongside conventional pixel-level measures to robustly evaluate a semantic segmentation approach for robotic vision.

6.8 Experiments

The proposed semantic region CRF and hierarchical region-to-pixel CRF are evaluated in this section. Performance is compared to conventional approaches on two commonly used semantic segmentation datasets: NYU v2 (Section 6.8.3) and Pascal VOC (Section 6.8.4). The compared approaches are evaluated both quantitatively, by analysing object-aware and pixel-level

performance metrics, and qualitatively, through visual inspection of example segmentations.

6.8.1 Compared Methods

Five methods are evaluated and compared in the following experiments. The two baseline methods are a fully convolutional network (FCN) [80] with no CRF and an FCN with the commonly use dense CRF model [182]. For our proposed approach, we analyse three different variants. The evaluation includes analysing performance of the region-based CRF only, compared to the hierarchical region-to-pixel CRF model. The importance of using semantic regions in the region CRF is also investigated. This is achieved by comparing performance to a model with data-driven initial regions for the region-based CRF. The compared methods are described in detail below. Note that for all methods, the same unary terms from a fully convolutional network are used.

Unary (FCN) [80] The first baseline is simply the unary terms only, with no conditional random field. A fully convolutional neural network is used to provide the unary terms. A detailed description of fully convolutional networks can be found in Section 2.3.2.1.

FCN [80] + Dense CRF [182] The second baseline is the commonly used set-up of a fully convolutional network with the conventional pixel-level fully connected CRF (dense CRF). Refer to Section 6.1.1.1 for full details on this CRF model.

Region CRF with Semantic Regions A fully convolutional network with our proposed region-level CRF. The regions used are the semantically meaningful regions determined by the fully convolutional network. The regions obey the semantic boundaries found by the FCN, as discussed in detail in Section 6.5.2. Following message passing, beliefs are read out at the region-level. This means that entire semantic regions change labels together. This approach includes no pixel-level refinement. Full details can be found in Section 6.5. Semantic regions are referred to as *SemReg* in the results.

FIGURE 6.7: Example data-driven regions generated using the approach from [190] (ImgReg). Unlike in other segmentation figures, colour denotes the image region, not a class label.

Hierarchical CRF with Semantic Regions A fully convolutional network with our proposed region-to-pixel hierarchical CRF. In the region layer of the CRF, the semantically meaningful regions from the FCN are used as the initial regions. Following message passing in the region layer, beliefs are read out at the pixel-level, rather than the region-level. These new pixel distributions are used as the unary terms in the dense pixel CRF layer, which carries out fine-grained pixel-level refinement. Refer to Section 6.6 for full details.

Hierarchical CRF with Image Regions This approach enables analysis of the importance of the proposed semantic regions in the region CRF. Again, this method is a fully convolutional network with our region-to-pixel hierarchical CRF. However, instead of the semantically meaningful regions from the FCN, structurally meaningful data-driven regions are used in the region CRF. The image regions are segmented based only on pixel intensities; connected pixels with similar appearance are grouped together. The data-driven segmentation is produced using the method described in [190]. This results in contour-aware regions that vary significantly in size, from large planar regions to small super-pixel sized regions. Some example regions found using this method are shown in Figure 6.7. The data-driven image regions are referred to as *ImgReg* in the results.

6.8.2 Evaluation Metrics

All compared approaches are evaluated using both conventional pixel-level measures, as well as the proposed object-aware semantic segmentation measures that are appropriate for robotic vision. The evaluation measures that are used in the following experiments can be summarised as:

- ***Gbl.*** $\boldsymbol{F_1^{IoA}}$: Global object-aware F_1 measure for robotic vision, as proposed in Section 6.7.3.
- ***Mean*** $\boldsymbol{F_1^{IoA}}$: Mean per-class object-aware F_1 measure for robotic vision, as proposed in Section 6.7.3.
- ***False Pos., fp***: Total number of object-aware false positive detections, as defined in Section 6.7.1.
- ***False Neg., fn***: Total number of object-aware false negative detections, as defined in Section 6.7.1.
- ***True Pos., tp***: Total number of object-aware true positive detections, as defined in Section 6.7.1.
- ***TPR***: Object-level true positive rate, as defined in Equation 6.25. Can also be referred to as the object-level recall.

$$TPR = \frac{tp}{tp + fn} \tag{6.25}$$

- ***Pixel Acc.***: The fraction of pixels that are correctly classified. See Section 6.7.4.1 for a mathematical definition and discussion.
- ***Pixel mIoU***: Mean per-class intersection-over-union of true pixel labels and predicted pixel labels. See Section 6.7.4.1 for a mathematical definition and discussion.

6.8.3 NYU v2 Dataset

Our proposed approach is first evaluated on the challenging NYU v2 indoor segmentation dataset [16].

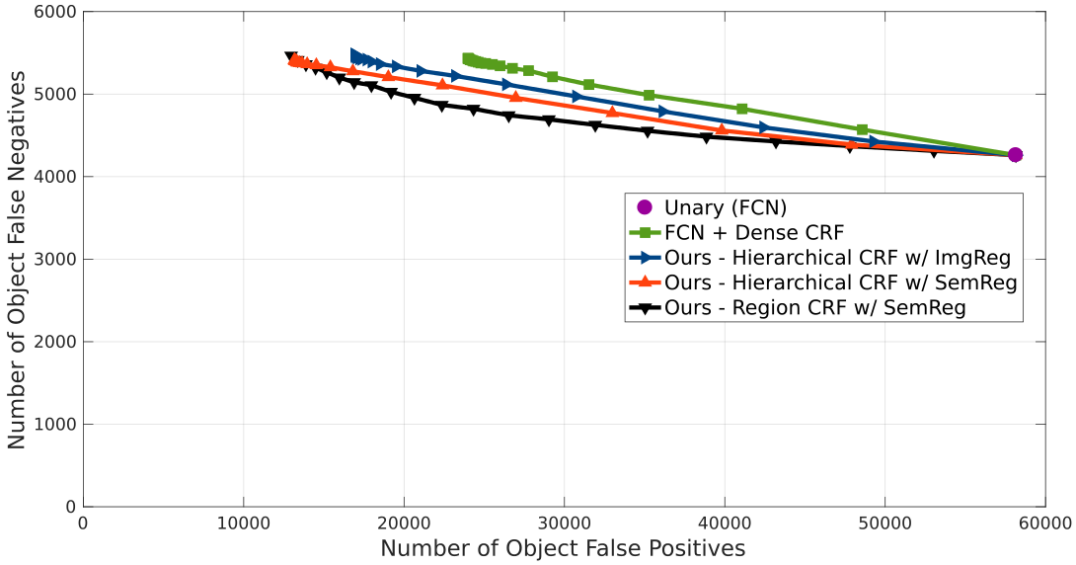


FIGURE 6.8: NYU v2 object-level performance of our approach compared to the unary terms and the commonly used dense CRF. Note the difference in scale of the two axes. All variants of our approach outperform dense CRF, with fewer false positives at a given number of false negatives. The importance of using semantic regions (*SemReg*) over data-driven image regions (*ImgReg*) in the initial segmentation is also shown.

6.8.3.1 Experimental Set-up

The NYU v2 dataset contains 1449 images of indoor scenes, with pixel-level and scene-level labels. In these experiments, only the pixel-level labels are used. The standard training and test sets are used, which split the data into 795 and 654 images for training and testing, respectively. While the dataset contains RGB-D images, in this work we evaluate all approaches using only the RGB channels. There are over 1000 different semantic classes represented in the dataset. The very fine-grained class labels are generally collapsed into coarser classes. We evaluate using the challenging 40 class version put together by Gupta *et al.* [232]. Example class labels include wall, cabinet, table and bed.

6.8.3.2 Quantitative Results

The object-level performance, in terms of the number of object-aware false positives and false negatives, is seen in Figure 6.8. The curves show a varying true positive rate and are generated by sweeping the level of influence of the incoming messages to a node (w in Equation 6.6) from zero to one. The w

TABLE 6.2: NYU v2 global and mean class object and pixel performance at 65% object true positive rate (except unary, which has a fixed TPR).

	Gbl. F_1^{IoA}	Mean F_1^{IoA}	False Pos.	Pixel Acc.	Pixel mIoU
Unary (FCN) [80]	18.9	14.0	58150	60.1	30.2
FCN [80] + Dense CRF [182]	28.8	22.3	24240	61.5	31.5
Ours: Region CRF SemReg	35.8	25.6	13371	61.7	30.7
Ours: Hierar. CRF SemReg	**36.6**	**26.8**	**13159**	62.7	31.7
Ours: Hierar. CRF ImgReg	33.2	25.0	17668	**63.4**	**32.5**

term controls the balance between the initial distributions (unary terms) and the pairwise connections. A value of zero means that the pairwise terms disappear, leaving just the unary terms from the FCN. A value of one means equal weighting between the unary and pairwise terms. Note the scale of the two axes. The unary term results in only 4260 false negatives but 58150 false positives. Unsurprisingly, no approach results in fewer false negatives than the unary term alone. This is because the segmentation produced by the unary classifier is littered with false positives. It would be possible to achieve a very low number of false negatives simply by randomly classifying every pixel. Of course, this would result in an extremely large number of false positives.

All of our approaches outperform the commonly used dense CRF [182]. That is, at a given number of object false negatives, our approaches result in fewer false positive object detections. For a given object true positive rate (or false negative rate) the best performance is achieved by either our region CRF or our hierarchical CRF with semantic regions. The region CRF outperforms the hierarchical approach when the object false negative rate is lower, while the hierarchical CRF is the best performing method when the object false negative rate is higher. The true advantage of the hierarchical approach, however, is that allowing pixel refinement results in superior object and pixel-level performance compared to the dense CRF approach, while the region CRF only outperforms dense CRF at the object-level. Compared to semantically meaningful regions (SemReg), the data-driven image regions (ImgReg) result in poorer object-aware performance. This shows the importance of using semantic regions when aiming to reduce object-aware false positives.

Table 6.2 summarises the object and pixel performance of the evaluated approaches at a given true positive rate (65%). In addition to the proposed

object-aware evaluation measures, performance is also measured by the number of false positives. The best pixel performance is achieved by the hierarchical approaches, with the data-driven (*ImgReg*) variant outperforming the semantic regions (*SemReg*). This is expected, since the data-driven regions are far more contour-aware than the semantic regions from FCN. This results in better object boundary delineation, as seen in Figure 6.9. However, the data-driven regions suffer in terms of object-level performance compared to the semantic regions, with a 34% increase in false positive object detections.

Compared to our hierarchical CRF with semantic regions, the conventional dense CRF results 84% more false positive object detections. Overall, our hierarchical CRF with semantic regions outperforms dense CRF in the global and mean measures at both the object-level and pixel-level. This is significant because, as discussed in Section 6.1.1, all false object detections in robotic applications can be costly.

Including FCN inference, our region CRF approach is 60% faster than dense CRF. Our hierarchical approach, which includes a dense CRF, increases the dense CRF inference time by only 5%. Inference on FCN takes between 50ms and 190ms, depending on the network size used, on a GeForce GTX 1080 GPU. Dense CRF inference takes approximately 70ms per iteration, while region CRF set-up and message passing takes 7ms and a further 23ms to propagate to pixels, on an Intel i7-4770 CPU. We report results using the largest FCN model and five iterations of dense CRF.

6.8.3.3 Qualitative Results

Example segmentations are shown in Figure 6.9 for the five compared approaches. The segmentations produced by the fully convolutional network are poor, with a large number of object-aware false positives of various sizes. Poor unary terms are a challenge for conditional random fields. Compared to the unary terms only and the conventional dense CRF approach, our semantic region CRF results in a significant improvement in the segmentation. Large regions of incorrect labels from the fully convolutional network are unable to be corrected by the dense CRF in many cases. The dense CRF improves the object boundary delineation, however, it does not perform well at the task of correcting incorrect classifications from the FCN. Conversely, the semantic region CRF performs very well at the task of correcting spurious

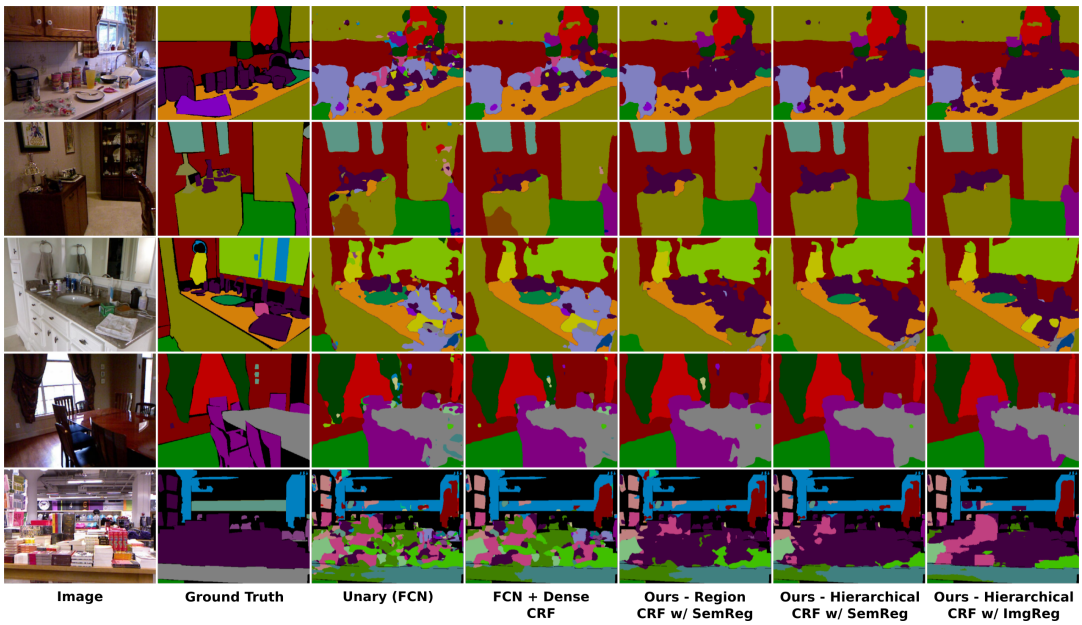


FIGURE 6.9: Qualitative results on NYU v2. Colour denotes class labels, with black used to represent unlabelled regions of the ground truth segmentations.

object-aware detections in the unary terms. No pixel-level refinement occurs in the region CRF segmentations, as entire semantic regions change labels together.

Compared to the region CRF alone, the hierarchical CRF further improves the segmentations. Firstly, the fine-grained refinement performed by the pixel-level layer of the CRF results in better object boundary localisation. Secondly, the hierarchical structure also results in a further reduction in object-aware false detections. This can be seen, for example, on the curtains in the dining room scene (Row 4) and throughout the bookshop scene (Row 5) in Figure 6.9.

Comparing the use of data-driven initial regions (ImgReg) to semantic initial regions (SemReg) in the hierarchical structure, the data-driven regions result in better object boundary localisation. This is expected, as the data-driven regions are based on pixel intensities and result in structurally meaningful regions that better obey object boundaries. This coincides with the pixel-level performance of the data-driven regions outperforming the semantic regions in Table 6.2. However, it is clear from visual inspection that the data-driven regions result in significantly poorer object-level performance, with many more false positive object detections. Again, this coincides with the object-aware measure in Table 6.2. As discussed in detail in Section 6.1, pixel-level performance is often less important than object-level performance for robotic

TABLE 6.3: Object-aware (global and mean class) and pixel performance on the Pascal VOC validation set.

	Gbl. F_1^{IoA}	Mean F_1^{IoA}	Pixel mIoU
Unary (FCN) [80]	55.3	45.3	65.8
FCN [80] + Dense CRF [182]	66.0	55.1	67.3
Ours: Region CRF SemReg	71.6	60.3	67.0
Ours: Hierar. CRF SemReg	**71.9**	**60.5**	**67.5**

vision applications. Therefore, based on both visual inspection and analysis of the object-aware and pixel-level evaluation metrics, the hierarchical CRF with semantic regions is the best overall performing model. This model results in significantly fewer object-aware false positives, as well as superior pixel-level performance, compared to a conventional dense CRF.

6.8.4 Pascal VOC

Our proposed approach to semantic segmentation is further evaluated on the Pascal VOC dataset [15].

6.8.4.1 Experimental Set-up

We evaluate on the Pascal VOC dataset with the additional annotations of the Semantic Boundary Dataset (SBD) [233]. In total, there are 8498 training images with pixel-wise class labels. Since the test set annotations are not publicly available, we evaluate using the 736 images of the validation set that do not overlap with the training set of SBD. The dataset contains 20 object classes and a background class. Compared to the NYU v2 dataset, Pascal VOC contains far more training images and almost half the number of class labels. As such, the unary terms from the fully convolutional network are significantly better than for the NYU v2 dataset in Section 6.8.3.

6.8.4.2 Quantitative Results

Results on the Pascal VOC dataset are summarised in Table 6.3. Both our region and hierarchical approaches outperform the conventional dense CRF in the object-aware metrics, with the hierarchical approach also achieving better pixel-level performance. Our hierarchical model results in a 16.6% and 5.9%

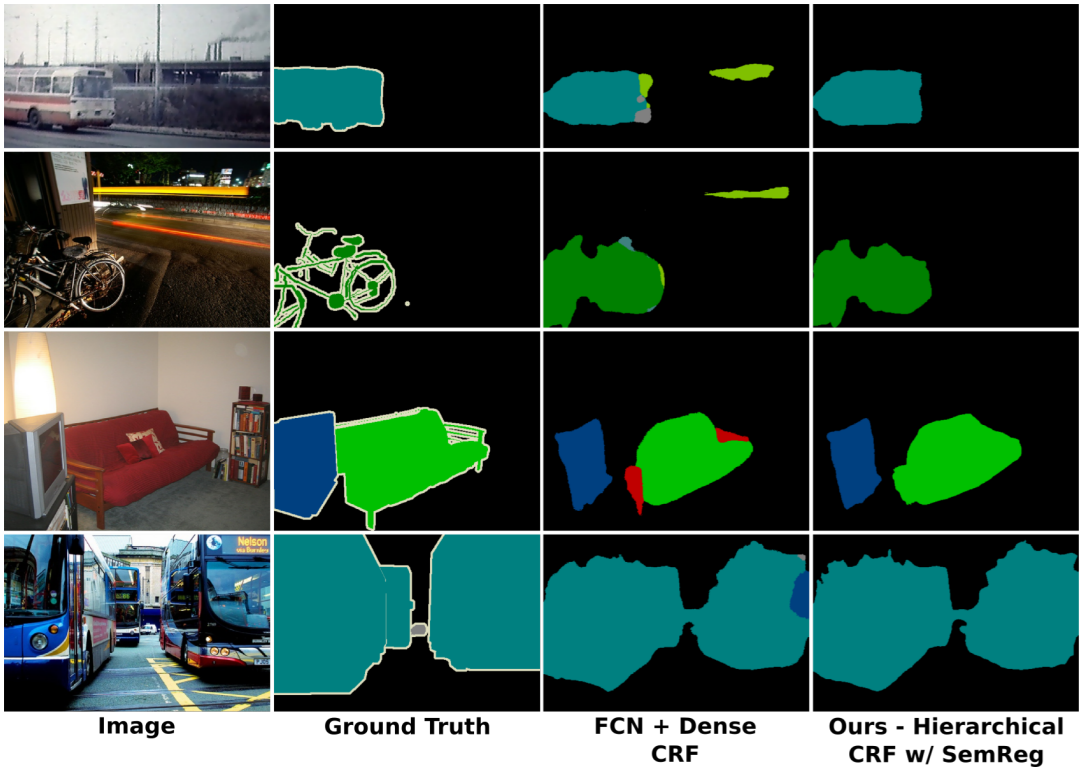


FIGURE 6.10: Qualitative results on Pascal VOC Validation. Ground truth segmentations are unlabelled at the object boundaries (shown as white). Non-white colours represent the class labels.

increase in the global F_1^{IoA} measure, compared to the unary terms only and the dense CRF, respectively. On the mean F_1^{IoA} measure, the improvement is 15.2% and 5.4%, respectively. The semantic region CRF on its own similarly outperforms the baseline methods, but the hierarchical model is required to improve performance on the pixel-level measure.

6.8.4.3 Qualitative Results

Example segmentations are shown in Figure 6.10. Due to the reduced number of classes and increased training set size compared to NYU v2, the dense CRF approach has fewer false positives than the NYU v2 segmentations in Figure 6.9. However, the dense CRF is still unable to refine many regions of erroneous labels. Our hierarchical CRF with semantic regions successfully corrects many of the object-aware false positives that the conventional dense CRF is unable to refine. Again, these results show how our proposed approach is well suited to problems in which any false object detection can be costly.

6.9 Discussion and Conclusion

Semantic segmentation for robotic vision demands special considerations. In robotic applications, the segmentation is not the end goal, but a tool to enable robotic action. As such, any objects that are detected in a scene can trigger some sort of costly robotic action. Conventional approaches to semantic segmentation optimise pixel-level performance, meaning that little significance is given to false object detections if the size of the detected object is small. However, size is often irrelevant in a real-world scenario, as robotic action can be triggered by objects of any scale. As such, semantic segmentation approaches for robotic vision should give a great deal of significance to object-level performance. Conventional pixel-level CRFs not only penalise mistakes based on size, but also struggle to refine large regions of incorrect predictions due to the consolidation of non-independent errors amongst pixels.

In this chapter, we introduced a semantic region CRF that heavily penalises any false object detection, regardless of size. Nodes in the CRF correspond to regions of the image that, according to a fully convolutional neural network, are semantically meaningful. In addition to the region-level CRF, a hierarchical CRF model that allows fine-grained pixel refinement is also proposed. The hierarchical model allows the object-level improvement seen by the region CRF to be combined with the advantages of a pixel-level CRF, such as better object boundary localisation. Finally, an object-aware evaluation measure for semantic segmentation is introduced. Due to the significance placed on false object detections, the measure is more appropriate for robotic vision, compared to conventional pixel-level measures.

Experimental results show both quantitatively and qualitatively that our proposed approaches significantly outperform conventional pixel-level CRFs. The region CRF results in a large reduction in object-aware false positives at a given true positive rate. The hierarchical model further improves on the object-level performance, while also improving on pixel-level performance by allowing fine-grained refinement. The importance of using semantic regions in the region CRF is investigated by comparing performance to a CRF with data-driven regions. Experimental results show that the semantic regions are important in reducing the number object-aware false positives. The results suggest that compared to conventional methods, our proposed approach is well suited to real-world problems such as robotic vision, where classifier outputs can trigger costly action.

Chapter 7

Conclusion

In this thesis, the problems of visual representation learning and semantics have been approached with robotic considerations in mind. Although motivated by the needs of a robotic vision system, the methodologies proposed in this thesis have applications in the wider field of computer vision. In this chapter, the key contributions of the presented research are summarised and future promising research opportunities are discussed.

7.1 Summary of Contributions

A deep metric learning approach for feature embedding learning and image classification problems was proposed in Chapter 4. The feature embeddings of all training set examples are defined to be Gaussian kernel centres, which can be used to classify examples by summing the influence of nearby centres in the feature embedding space. Training of the network was made feasible by the introduction of periodic asynchronous updates of the Gaussian centres, allowing the true centre locations to drift from the stored locations. We showed how to make the approach scalable by use of fast approximate nearest neighbour search.

A thorough experimental evaluation analysed the proposed nearest neighbour Gaussian kernel method in terms of distance metric learning and image classification. Our approach outperforms state-of-the-art deep metric learning methods on transfer learning problems, showing that feature embeddings co-locate in the metric space based on semantic similarity, despite the classes being withheld during training. Finally, we showed that the proposed approach outperforms a conventional softmax-based convolutional

neural network on several image classification datasets, with the advantage particularly large when training data is impoverished.

Novelty detection and open set recognition was investigated in Chapter 5. A deep metric learning approach that allows examples from known classes to be classified and examples from novel classes to be detected as such, was proposed. This ability is a necessity of robotic and other real-world problems, where conventional closed set classifiers that silently fail by classifying novel examples into a known class are unacceptable. Novelty measures for use in deep metric spaces were investigated and compared. Experimental evaluation demonstrated how the proposed approach is able to detect novel examples more reliably than softmax uncertainty baselines and purpose built novelty detectors and open set classifiers.

Open set problems were further explored in Chapter 5 by investigating the active learning of observed novel classes. Recognising that learning should not cease following initial model training, we investigated how a deep metric learning model can efficiently learn from observed novel examples by querying a user or oracle for labels. A query selection method was proposed that favours the selection of examples that are both novel and from classes that are common in the true distribution of data. We showed how the proposed query selection approach can be incorporated into an active learning algorithm. A method to improve the representation of novel classes with a labelling budget of zero was also explored. The proposed method uses spatial relationships in the metric space to generate pseudo-labels for unlabelled observations. An in-depth experimental evaluation showed how our proposed active learning approach significantly outperforms the compared methods at small labelling budgets, enabling a vision system to efficiently learn from observations and improve its understanding of the true distribution of data.

We turned to the problem of semantic segmentation in Chapter 6. The issues with conventional semantic segmentation systems that optimise only at the level of pixels were discussed, highlighting the importance of minimising false object detections for robotic applications. A semantic region conditional random field model was proposed that significantly penalises any false object detection. The importance of using semantically meaningful image regions as conditional random field nodes was discussed. Further, a hierarchical region-to-pixel conditional random field was proposed to allow fine-grained pixel-level refinement.

The shortcomings of conventional semantic segmentation evaluation methods were discussed, recognising that for robotic applications, pixel-level performance is often less important than object-level performance. To that end, an object-aware evaluation metric was proposed that heavily penalises all erroneous object detections, regardless of size. Example segmentations were used to show how the proposed measure addresses the discussed issues with conventional pixel-level evaluation measures. Our proposed conditional random field models were evaluated experimentally. Quantitative results showed that our models outperform the compared methods on both the conventional and proposed evaluation measures, resulting in significantly fewer false positive detections at a given number of false negatives.

7.2 Future Work

The contributions summarised in Section 7.1 lead to a number of promising future research opportunities that are explored below.

- In Chapter 4, we showed how our proposed metric learning approach results in feature embeddings that co-locate based on semantic similarities beyond class label. For the Birds200 dataset, for example, feature embeddings were clustered based on species but also often co-located based on genus. There exists a promising research opportunity in exploring how our metric learning approach can be modified to explicitly model class hierarchy. This could involve multiple Gaussian kernels per example with differing scale terms. Explicit modelling of class hierarchy should improve the ability of the learned distance metric to transfer to novel examples.
- Research opportunities exist in the deployment of our open set recognition and active learning methodologies from Chapter 5 on robotic platforms. An example of this is a mobile robot exploring the environment, categorising known objects into the correct class, detecting novel objects and querying a human user for labels. Deployment on a robot introduces new hurdles. For example, the system would likely require an object proposal and acceptance pipeline. However, new opportunities are also introduced. For example, the agency of the robot can be exploited by allowing further investigation of observed unknown examples, such as viewing the object from different angles.

- Metric learning has many potential benefits for robotic applications and should be further explored from a pixel-level perspective. As discussed in Section 2.4.2.5, existing literature regarding pixel-level metric learning is limited. The transfer learning capabilities and classification advantages of our metric learning approach, demonstrated in Chapter 4, would be equally as useful at the pixel-level as at the image-level. This promising future research opportunity requires investigation into how to manage the potentially astronomical numbers of pixel-wise feature embeddings. This could involve the intelligent selection of key pixels, with region growing used to form region-level feature embeddings.
- Adaptation of our proposed metric learning approach for pixel-wise feature embeddings also opens opportunities to explore open set scenarios in semantic segmentation problems. Of particular interest is active learning for semantic segmentation. An open research question in this domain is how to select pixels for active labelling. Possible solutions include selecting key pixels, object bounding box proposals or object region proposals.
- The semantic segmentation approaches proposed in Chapter 6 can be improved by incorporating geometric information from active depth sensors or stereo cameras into the inference process. Depth information would allow for better contextual modelling compared to using only the pixel coordinates. Contextual relationships such as *a coffee mug is likely to be above a table* could be more accurately modelled as *a coffee mug is likely to be on a table*. The conditional random field models are readily extendible to incorporate this geometric information and the training of such models can be facilitated by the depth data that is attached to the RGB images of the NYU v2 dataset [16].
- Semantic segmentation is often an important part of robotic arm manipulation systems. As such, it would be interesting to investigate the performance of our proposed semantic segmentation systems from Chapter 6 on such applications. Cluttered scenes, as well as impoverished training data scenarios, are of particular interest as these would likely generate poor unary terms with high numbers of false object detections. These false detections could trigger ill-informed robotic action. The importance of optimising object-level performance would be highlighted in such a scenario.

Several interesting broad future research directions exist in the domain of machine learning for robotic vision. The incorporation of temporal information is perhaps one of the most interesting of these directions. Robotics is an inherently temporal problem, which opens up a range of opportunities to improve and extend existing computer vision approaches for static images. A second interesting research direction in robotic vision is the exploration of how the agency of robots can be used to enhance machine learning methods. A robot is not a passive observer of the world; an intelligent agent has the ability to "poke" and explore the environment. When a robot is able to measure the uncertainty of the predictions it makes, it can actively attempt to reduce that uncertainty by interacting with the environment, such as by changing its viewpoint or an object's orientation. Finally, incorporating the structure of the environment, such as depth information, into learning algorithms remains an interesting research topic. Robots exist in a 3D world and require this information to reliably interact with the environment. As such, there are opportunities to better leverage such depth and structure for representation learning and semantics.

7.3 Concluding Remarks

The learning of visual representations and extraction of semantic information are important steps in the development of autonomous robotic systems that can sense and understand the environment. Vision is a key aspect of perception, cognition and learning in humans. As such, there is a strong argument that this should also be the case for robotics. Visual representations and semantic information can play important roles in enabling the safe and effective execution of tasks such as navigation, object manipulation and human-robot interaction. Learning from visual data allows a robot to develop an understanding of its environment. This understanding is an imperative precursor to intelligent robotic action.

Existing literature in the field of learning for vision seldom make any considerations for such robotic applications. In this thesis, methodologies were proposed to bridge part of this gap by designing machine learning systems with a focus on the needs of robotic vision. Although motivation for each contribution of this research was drawn primarily from robotic problems, the developed methodologies have advantages beyond the application to robotics.

Improvements were demonstrated over existing approaches on conventional computer vision problems.

The needs of a robotic vision system are not unique to the application of robotics. In many real-world computer vision problems, such as computer-aided diagnosis, human-computer interaction, security and biodiversity monitoring, the outputs of machine learning models are not the end goal. Model predictions can trigger an action and mistakes in predictions can be costly. The research presented in this thesis has potential in many such applications. Although deployment on such real-world problems was not in the scope of this thesis, it is our hope that this can be demonstrated in future work, with a particular interest in pursuing the field of practical robotic vision.

Bibliography

[1] D. L. Ripley and T. Politzer, "Vision Disturbance after TBI", *NeuroRehabilitation*, vol. 27, no. 3, pp. 215–216, 2010 (cit. on p. 1).

[2] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun, "Towards fully autonomous driving: systems and algorithms", in *IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 163–168 (cit. on pp. 1, 92).

[3] J. Biswas and M. Veloso, "Depth camera based indoor mobile robot localization and navigation", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 1697–1702 (cit. on p. 1).

[4] J. Leitner, A. W. Tow, N. Sünderhauf, J. E. Dean, J. W. Durham, M. Cooper, M. Eich, C. Lehnert, R. Mangels, C. McCool, P. T. Kujala, L. Nicholson, T. Pham, J. Sergeant, L. Wu, F. Zhang, B. Upcroft, and P. Corke, "The acrv picking benchmark: a robotic shelf picking benchmark to foster reproducible research", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 4705–4712 (cit. on pp. 1, 92).

[5] J. Engel, J. Stückler, and D. Cremers, "Large-scale direct slam with stereo cameras", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 1935–1942 (cit. on p. 1).

[6] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for rgb-d cameras", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 2100–2106 (cit. on p. 1).

[7] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120 db 15μs latency asynchronous temporal contrast vision sensor", *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008 (cit. on p. 1).

[8] S. Lloyd, "Least squares quantization in PCM", *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982 (cit. on pp. 5, 25, 123, 165, 185).

[9] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.", *Psychological review*, vol. 65, no. 6, pp. 386–408, 1958 (cit. on pp. 6, 26).

[10] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position", *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980 (cit. on pp. 6, 29).

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998 (cit. on pp. 6, 28, 29, 33).

[12] R. Kindermann and J. L. Snell, *Markov Random Fields and Their Applications*. AMS, 1980 (cit. on pp. 7, 25, 54).

[13] J. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data", in *International Conference on Machine Learning (ICML)*, vol. 8, 2001, pp. 282–289 (cit. on pp. 7, 25, 54, 56, 198).

[14] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016 (cit. on p. 10).

[15] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. [Online]. Available: `http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html` (cit. on pp. 10, 31, 93, 224).

[16] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor Segmentation and Support Inference from RGBD Images", in *European Conference on Computer Vision (ECCV)*, 2012 (cit. on pp. 10, 219, 230).

[17] E. Hoffer and N. Ailon, "Deep metric learning using triplet network", in *International Workshop on Similarity-Based Pattern Recognition*, 2015, pp. 84–92 (cit. on pp. 12, 43, 95, 100).

[18] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 815–823 (cit. on pp. 12, 43, 44, 95, 100, 122, 124, 126, 142, 157).

[19] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep Metric Learning via Lifted Structured Feature Embedding", in *IEEE Conference on*

Computer Vision and Pattern Recognition (CVPR), 2016, pp. 4004–4012 (cit. on pp. 12, 43, 44, 93, 95, 100, 122–124, 126, 142, 157).

[20] K. Sohn, "Improved Deep Metric Learning with Multi-class N-pair Loss Objective", in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 1857–1865 (cit. on pp. 12, 43, 44, 94, 95, 100, 122–124, 126, 142, 157).

[21] V. B. G. Kumar, B. Harwood, G. Carneiro, I. Reid, and T. Drummond, "Smart Mining for Deep Metric Learning", *arXiv preprint arXiv:1704.01285*, 2017 (cit. on pp. 12, 43, 44, 93, 95, 100, 122–126, 142, 157).

[22] H. O. Song, S. Jegelka, V. Rathod, and K. Murphy, "Deep Metric Learning via Facility Location", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2206–2214 (cit. on pp. 12, 45, 93, 95, 100, 122, 123, 125, 126, 142, 157).

[23] V. B. G. Kumar, G. Carneiro, and I. Reid, "Learning Local Image Descriptors with Deep Siamese and Triplet Convolutional Networks by Minimizing Global Loss Functions", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 5385–5394 (cit. on pp. 12, 44, 95, 100, 122, 126, 142, 157).

[24] B. J. Meyer, B. Harwood, and T. Drummond, "Deep metric learning and image classification with nearest neighbour gaussian kernels", in *IEEE International Conference on Image Processing (ICIP)*, 2018, pp. 151–155 (cit. on pp. 20, 98).

[25] B. J. Meyer and T. Drummond, "The Importance of Metric Learning for Robotic Vision: Open Set Recognition and Active Learning", *arXiv preprint arXiv:1902.10363*, 2019 (cit. on pp. 20, 154).

[26] B. J. Meyer and T. Drummond, "Improved semantic segmentation for robotic applications with hierarchical conditional random fields", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 5258–5265 (cit. on pp. 20, 196).

[27] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: a review of classification techniques", *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007 (cit. on p. 23).

[28] D. G. Lowe, "Object recognition from local scale-invariant features", in *International Conference on Computer Vision (ICCV)*, vol. 2, 1999, pp. 1150–1157 (cit. on pp. 23, 29).

[29] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)", *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008 (cit. on pp. 23, 29).

[30] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, vol. 1, 2005, pp. 886–893 (cit. on pp. 24, 29).

[31] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints", in *In Workshop on Statistical Learning in Computer Vision, ECCV*, 2004, pp. 1–22 (cit. on p. 24).

[32] L. Breiman, *Classification and regression trees*. Routledge, 1984 (cit. on p. 24).

[33] J. R. Quinlan, "Induction of decision trees", *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986 (cit. on p. 24).

[34] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993 (cit. on p. 24).

[35] R. E. Schapire, "The strength of weak learnability", *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990 (cit. on p. 24).

[36] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting", *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997 (cit. on p. 24).

[37] L. Mason, J. Baxter, P. L. Bartlett, and M. R. Frean, "Boosting algorithms as gradient descent", in *Advances in Neural Information Processing Systems (NIPS)*, 2000, pp. 512–518 (cit. on p. 24).

[38] L. Breiman, "Bagging predictors", *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996 (cit. on p. 24).

[39] L. Breiman, "Random forests", *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001 (cit. on p. 24).

[40] C. Cortes and V. Vapnik, "Support-vector networks", *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995 (cit. on pp. 25, 29).

[41] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers", in *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, 1992, pp. 144–152 (cit. on pp. 25, 29).

[42] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines", *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, Mar. 2002 (cit. on p. 25).

[43] K.-B. Duan and S. S. Keerthi, "Which is the best multiclass svm method? an empirical study", in *Multiple Classifier Systems*, Springer Berlin Heidelberg, 2005 (cit. on p. 25).

[44] J. C. Dunn, "A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters", *Journal of Cybernetics*, vol. 3, no. 3, pp. 32–57, 1973 (cit. on p. 25).

[45] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, 1981 (cit. on p. 25).

[46] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm", *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977 (cit. on p. 25).

[47] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines", in *International Conference on Machine Learning (ICML)*, 2010, pp. 807–814 (cit. on p. 27).

[48] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks", in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011, pp. 315–323 (cit. on p. 27).

[49] A. G. Ivakhnenko and V. G. Lapa, *Cybernetic Predicting Devices*, ser. JPRS 37, 803. Purdue University School of Electrical Engineering, 1965 (cit. on p. 27).

[50] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors", *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986 (cit. on p. 27).

[51] V. Vapnik, *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics)*. Springer-Verlag New York, Inc., 1982 (cit. on p. 29).

[52] V. Blanz, B. Scholkopf, H. Bultho, C. Burges, V. Vapnik, T. Vetter, B. Schslkopf, and H. B. I, "Comparison of View-Based Object Recognition Algorithms Using Realistic 3D Models", in *International Conference on Artificial Neural Networks (ICANN)*, 1996, pp. 1–6 (cit. on p. 29).

[53] D. G. Lowe, "Distinctive image features from scale-invariant keypoints", *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004 (cit. on p. 29).

[54] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF : Binary Robust Independent Elementary Features", in *European Conference on Computer Vision (ECCV)*, 2010, pp. 778–792 (cit. on p. 29).

[55] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF", in *International Conference on Computer Vision (ICCV)*, 2011, pp. 2564–2571 (cit. on p. 29).

[56] S. Hochreiter and J. Schmidhuber, "Long short-term memory.", *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997 (cit. on p. 29).

[57] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks", *Science*, vol. 313, no. 5786, pp. 504–507, 2006 (cit. on p. 29).

[58] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks", in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1223–1231 (cit. on p. 29).

[59] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, High Performance Convolutional Neural Networks for Image Classification", in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2011, pp. 1237–1242 (cit. on p. 29).

[60] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105 (cit. on pp. 29, 31–33, 36, 93, 96, 119, 131, 142).

[61] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge", *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015 (cit. on pp. 30, 37, 81, 93, 123, 131).

[62] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: a large-scale hierarchical image database", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255 (cit. on p. 30).

[63] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors", *arXiv preprint arXiv:1207.0580*, 2012 (cit. on pp. 32, 85).

[64] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks", in *European Conference on Computer Vision (ECCV)*, 2014, pp. 818–833 (cit. on pp. 33, 73).

[65] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recoginition", in *International Conference on*

Learning Representations (ICLR), 2015, pp. 1–14 (cit. on pp. 33, 36, 78, 79).

[66] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9 (cit. on pp. 33, 41, 93, 96, 123, 142).

[67] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826 (cit. on pp. 34, 39).

[68] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning.", in *AAAI Conference on Artificial Intelligence*, vol. 4, 2017, p. 12 (cit. on pp. 34, 35).

[69] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778 (cit. on pp. 34, 35, 38, 41, 79, 86, 93, 96, 119, 131, 142).

[70] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks", in *European Conference on Computer Vision (ECCV)*, 2016, pp. 630–645 (cit. on p. 35).

[71] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks.", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2017, p. 3 (cit. on pp. 35, 38, 79).

[72] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth", in *European Conference on Computer Vision (ECCV)*, 2016, pp. 646–661 (cit. on p. 35).

[73] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2014, pp. 512–519 (cit. on pp. 35, 99).

[74] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition.", in *International Conference on Machine Learning (ICML)*, vol. 32, 2014, pp. 647–655 (cit. on pp. 35, 99).

[75] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat : Integrated Recognition , Localization and Detection using

Convolutional Networks", *arXiv preprint arXiv:1312.6229*, 2013 (cit. on p. 36).

[76] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images", in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 2843–2851 (cit. on p. 36).

[77] R. Girshick, J. Donahue, T. Darrell, U. C. Berkeley, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 2–9 (cit. on p. 36).

[78] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Simultaneous Detection and Segmentation", in *European Conference on Computer Vision (ECCV)*, 2014, pp. 1–16 (cit. on pp. 36, 198).

[79] S. Gupta, R. Girshick, P. Arbelaez, and J. Malik, "Learning Rich Features from RGB-D Images for Object Detection and Segmentation", in *European Conference on Computer Vision (ECCV)*, 2014 (cit. on p. 36).

[80] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 3431–3440 (cit. on pp. 36–38, 88, 191, 194, 198, 201, 217, 221, 224).

[81] A. Kendall, V. Badrinarayanan, and R. Cipolla, "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding", *arXiv preprint arXiv:1511.02680*, 2015 (cit. on pp. 38, 198).

[82] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017 (cit. on pp. 38, 198).

[83] H. Noh, S. Hong, and B. Han, "Learning Deconvolution Network for Semantic Segmentation", in *International Conference on Computer Vision (ICCV)*, vol. 1, 2015, pp. 1520–1528 (cit. on pp. 38, 198).

[84] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation", in *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, pp. 234–241 (cit. on pp. 38, 198).

[85] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal, "The importance of skip connections in biomedical image segmentation", in *Deep Learning and Data Labeling for Medical Applications*, Springer, 2016, pp. 179–187 (cit. on pp. 38, 198).

[86] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation", in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 1175–1183 (cit. on pp. 38, 198).

[87] G. Lin, A. Milan, C. Shen, and I. D. Reid, "RefineNet: Multi-path Refinement Networks for High-Resolution Semantic Segmentation.", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2017, p. 5 (cit. on pp. 39, 198).

[88] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, "Large kernel matters–improve semantic segmentation by global convolutional network", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4353–4361 (cit. on p. 39).

[89] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions", in *International Conference on Learning Representations (ICLR)*, 2016 (cit. on pp. 40, 41, 198).

[90] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs", in *International Conference on Learning Representations (ICLR)*, 2015, pp. 1–14 (cit. on pp. 40, 57, 192, 198).

[91] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2018 (cit. on pp. 40, 41, 57, 198).

[92] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation", *arXiv preprint arXiv:1706.05587*, 2017 (cit. on pp. 40, 198).

[93] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation", in *European Conference on Computer Vision (ECCV)*, 2018 (cit. on pp. 40, 41, 198).

[94] G. Papandreou, I. Kokkinos, and P. Savalle, "Modeling local and global deformations in Deep Learning: Epitomic convolution, Multiple Instance Learning, and sliding window detection", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 390–399 (cit. on p. 40).

[95] W. Liu, A. Rabinovich, and A. C. Berg, "ParseNet: Looking Wider to See Better", *arXiv preprint arXiv:1506.04579*, 2015 (cit. on p. 41).

[96] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2881–2890 (cit. on p. 41).

[97] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift", *arXiv preprint arXiv:1502.03167*, 2015 (cit. on pp. 41, 77).

[98] P. O. Pinheiro and R. Collobert, "From Image-Level to Pixel-Level Labeling With Convolutional Networks", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015 (cit. on p. 41).

[99] J. Dai, K. He, and J. Sun, "BoxSup: Exploiting Bounding Boxes to Supervise Convolutional Networks for Semantic Segmentation", in *International Conference on Computer Vision (ICCV)*, 2015 (cit. on p. 41).

[100] A. Khoreva, R. Benenson, J. H. Hosang, M. Hein, and B. Schiele, "Simple Does It: Weakly Supervised Instance and Semantic Segmentation.", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2017, p. 3 (cit. on p. 41).

[101] J. Bromley, I. Guyon, Y. Lecun, E. Sackinger, and R. Shah, "Signature verification using a Siamese time delay neural network", in *Advances in Neural Information Processing Systems (NIPS)*, 1993 (cit. on pp. 42, 99).

[102] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality Reduction by Learning an Invariant Mapping", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2006, pp. 1735–1742 (cit. on pp. 42, 99).

[103] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2005, pp. 539–546 (cit. on pp. 42, 99).

[104] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point

descriptors", in *International Conference on Computer Vision (ICCV)*, 2015, pp. 118–126 (cit. on p. 43).

[105] Y. Yuan, K. Yang, and C. Zhang, "Hard-aware deeply cascaded embedding", in *International Conference on Computer Vision (ICCV)*, 2017, pp. 814–823 (cit. on p. 43).

[106] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning Fine-Grained Image Similarity with Deep Ranking", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1386–1393 (cit. on pp. 43, 100).

[107] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition.", in *British Machine Vision Conference (BMVC)*, vol. 1, 2015, p. 6 (cit. on p. 43).

[108] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin, "Deep metric learning with angular loss", in *International Conference on Computer Vision (ICCV)*, 2017, pp. 2612–2620 (cit. on pp. 43, 44).

[109] K. Q. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification", in *Advances in Neural Information Processing Systems (NIPS)*, 2006 (cit. on pp. 43, 95, 99, 124).

[110] B. Harwood and T. Drummond, "FANNG: Fast Approximate Nearest Neighbour Graphs", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 5713–5722 (cit. on pp. 44, 108, 113, 114).

[111] M. T. Law, N. Thome, and M. Cord, "Quadruplet-wise image similarity learning", in *International Conference on Computer Vision (ICCV)*, 2013, pp. 249–256 (cit. on p. 44).

[112] E. Ustinova and V. Lempitsky, "Learning deep embeddings with histogram loss", in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 4170–4178 (cit. on pp. 44, 45).

[113] W. Chen, X. Chen, J. Zhang, and K. Huang, "Beyond triplet loss: a deep quadruplet network for person re-identification", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2017 (cit. on pp. 44, 45).

[114] C. Huang, C. C. Loy, and X. Tang, "Local similarity-aware deep feature embedding", in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 1262–1270 (cit. on pp. 44, 45).

[115] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification", in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 1988–1996 (cit. on p. 45).

[116] O. Rippel, M. Paluri, P. Dollar, and L. Bourdev, "Metric learning with adaptive density discrimination", in *International Conference on Learning Representations (ICLR)*, 2016 (cit. on pp. 46, 93, 94, 157).

[117] A. W. Harley, K. G. Derpanis, and I. Kokkinos, "Learning dense convolutional embeddings for semantic segmentation", *arXiv preprint arXiv:1511.04377*, 2015 (cit. on p. 46).

[118] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy, "Semantic instance segmentation via deep metric learning", *arXiv preprint arXiv:1703.10277*, 2017 (cit. on p. 46).

[119] Y. Chen, J. Pont-Tuset, A. Montes, and L. Van Gool, "Blazingly fast video object segmentation with pixel-wise metric learning", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 1189–1198 (cit. on p. 46).

[120] S. Li, B. Seybold, A. Vorobyov, A. Fathi, Q. Huang, and C.-C. Jay Kuo, "Instance embedding transfer to unsupervised video object segmentation", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6526–6535 (cit. on p. 46).

[121] X. Wang and A. Gupta, "Unsupervised Learning of Visual Representations Using Videos", in *International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 2794–2802 (cit. on p. 47).

[122] B. Sofman, B. Neuman, A. Stentz, and J. A. Bagnell, "Anytime online novelty and change detection for mobile robots", *Journal of Field Robotics*, vol. 28, no. 4, pp. 589–618, 2011 (cit. on p. 48).

[123] L. Tarassenko, P. Hayton, N. Cerneaz, and M. Brady, "Novelty detection for the identification of masses in mammograms", in *International Conference on Artificial Neural Networks*, Jun. 1995, pp. 442–447 (cit. on p. 48).

[124] C. P. Diehl and J. B. Hampshire, "Real-time object classification and novelty detection for collaborative video surveillance", in *International Joint Conference on Neural Networks*, vol. 3, 2002, pp. 2620–2625 (cit. on pp. 48, 49, 154).

[125] M. Markou and S. Singh, "A neural network-based novelty detector for image sequence analysis", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1664–1677, 2006 (cit. on p. 48).

[126] V. Jyothsna, V. V. Rama Prasad, and K. Munivara Prasad, "A review of anomaly based intrusion detection systems", *International Journal of Computer Applications*, vol. 28, pp. 26–35, Aug. 2011 (cit. on p. 48).

[127] F. E. Grubbs, "Procedures for detecting outlying observations in samples", *Technometrics*, vol. 11, no. 1, pp. 1–21, 1969 (cit. on pp. 48, 154).

[128] C. C. Aggarwal and P. S. Yu, "Outlier detection with uncertain data", in *SIAM International Conference on Data Mining*, 2008, pp. 483–493 (cit. on pp. 48, 154).

[129] D. P. Filev and F. Tseng, "Novelty detection based machine health prognostics", in *International Symposium on Evolving Fuzzy Systems*, IEEE, 2006, pp. 193–199 (cit. on p. 48).

[130] P. Paalanen, J.-K. Kamarainen, J. Ilonen, and H. Kälviäinen, "Feature representation and discrimination based on gaussian mixture model probability densities—practices and algorithms", *Pattern Recognition*, vol. 39, no. 7, pp. 1346–1358, 2006 (cit. on p. 48).

[131] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos, "Online outlier detection in sensor data using non-parametric models", in *International conference on Very large data bases*, 2006, pp. 187–198 (cit. on p. 48).

[132] R. Ramezani, P. Angelov, and X. Zhou, "A fast approach to novelty detection in video streams using recursive density estimation", in *4th International IEEE Conference Intelligent Systems*, vol. 2, 2008, pp. 14–2 (cit. on p. 48).

[133] D. Dasgupta and N. S. Majumdar, "Anomaly detection in multidimensional data using negative selection algorithm", in *Proceedings of the 2002 Congress on Evolutionary Computation*, IEEE, vol. 2, 2002, pp. 1039–1044 (cit. on p. 48).

[134] F. A. González and D. Dasgupta, "Anomaly detection using real-valued negative selection", *Genetic Programming and Evolvable Machines*, vol. 4, no. 4, pp. 383–403, 2003 (cit. on p. 48).

[135] M. A. F. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection", *Signal Processing*, vol. 99, pp. 215–249, 2014 (cit. on pp. 48, 49, 141, 154).

[136] F. Angiulli and C. Pizzuti, "Fast Outlier Detection in High Dimensional Spaces", in *Principles of Data Mining and Knowledge Discovery*, 2002, pp. 15–27 (cit. on pp. 48, 154).

[137] V. Hautamaki, I. Karkkainen, and P. Franti, "Outlier detection using k-nearest neighbour graph", in *International Conference on Pattern Recognition (ICPR)*, vol. 3, Aug. 2004, pp. 430–433 (cit. on pp. 48, 154).

[138] S. D. Bay and M. Schwabacher, "Mining distance-based outliers in near linear time with randomization and a simple pruning rule",

in *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2003, pp. 29–38 (cit. on p. 48).

[139] Z. He, X. Xu, and S. Deng, "Discovering cluster-based local outliers", *Pattern Recognition Letters*, vol. 24, no. 9-10, pp. 1641–1650, 2003 (cit. on p. 48).

[140] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection", in *Advances in Neural Information Processing Systems (NIPS)*, 2000, pp. 582–588 (cit. on pp. 49, 154).

[141] A. B. Gardner, A. M. Krieger, G. Vachtsevanos, and B. Litt, "One-class novelty detection for seizure analysis from intracranial eeg", *Journal of Machine Learning Research*, vol. 7, no. Jun, pp. 1025–1044, 2006 (cit. on p. 49).

[142] J. Ma and S. Perkins, "Time-series novelty detection using one-class support vector machines", in *International Joint Conference on Neural Networks*, IEEE, vol. 3, 2003, pp. 1741–1745 (cit. on p. 49).

[143] Z. He, S. Deng, and X. Xu, "An Optimization Model for Outlier Detection in Categorical Data", in *Advances in Intelligent Computing*, 2005, pp. 400–409 (cit. on pp. 49, 154).

[144] S. Ando, "Clustering Needles in a Haystack: An Information Theoretic Analysis of Minority and Outlier Detection", in *IEEE International Conference on Data Mining (ICDM)*, Oct. 2007, pp. 13–22 (cit. on pp. 49, 154).

[145] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks", *arXiv preprint arXiv:1610.02136*, 2016 (cit. on pp. 49, 142, 146, 155, 169–172).

[146] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks", in *International Conference on Machine Learning (ICML)*, 2017 (cit. on pp. 49–51, 146, 147).

[147] A. Bendale and T. E. Boult, "Towards open set deep networks", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1563–1572 (cit. on pp. 49, 141, 147, 151, 155, 169–171).

[148] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks", in *International Conference on Learning Representations (ICLR)*, 2018 (cit. on pp. 50, 147, 151, 155, 169, 171, 172).

[149] A. Mandelbaum and D. Weinshall, "Distance-based Confidence Score for Neural Network Classifiers", *arXiv preprint arXiv:1709.09844*, 2017 (cit. on pp. 50, 155, 159).

[150] M. Masana, I. Ruiz, J. Serrat, J. van de Weijer, and A. M. Lopez, "Metric learning for novelty and anomaly detection", *arXiv preprint arXiv:1808.05492*, 2018 (cit. on pp. 50, 155).

[151] M. Kliger and S. Fleishman, "Novelty Detection with GAN", *arXiv preprint arXiv:1802.10560*, 2018 (cit. on pp. 50, 155).

[152] C. Cortes, G. DeSalvo, and M. Mohri, "Learning with Rejection", in *Algorithmic Learning Theory*, 2016, pp. 67–82 (cit. on p. 50).

[153] C. Cortes, G. DeSalvo, C. Gentile, M. Mohri, and S. Yang, "Online Learning with Abstention", *arXiv preprint arXiv:1703.03478*, 2017 (cit. on p. 50).

[154] P. L. Bartlett and M. H. Wegkamp, "Classification with a reject option using a hinge loss", *Journal of Machine Learning Research*, vol. 9, pp. 1823–1840, Oct. 2008 (cit. on p. 50).

[155] Y. Grandvalet, A. Rakotomamonjy, J. Keshet, and S. Canu, "Support vector machines with a reject option", in *Advances in Neural Information Processing Systems (NIPS)*, 2009, pp. 537–544 (cit. on p. 50).

[156] R. Herbei and M. H. Wegkamp, "Classification with reject option", *Canadian Journal of Statistics*, vol. 34, no. 4, pp. 709–721, 2006 (cit. on p. 50).

[157] M. Yuan and M. Wegkamp, "Classification methods with reject option based on convex risk minimization", *Journal of Machine Learning Research*, vol. 11, pp. 111–130, 2010 (cit. on p. 50).

[158] D. D. Lewis and W. A. Gale, "A Sequential Algorithm for Training Text Classifiers", in *Special Interest Group on Information Retrieval (SIGIR)*, 1994, pp. 3–12 (cit. on pp. 51, 52, 155).

[159] B. Settles, "Active Learning Literature Survey", University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2010 (cit. on pp. 51, 52, 141, 155).

[160] A. Culotta and A. McCallum, "Reducing labeling effort for structured prediction tasks", in *AAAI Conference on Artificial Intelligence*, vol. 5, 2005, pp. 746–751 (cit. on p. 52).

[161] T. Scheffer, C. Decomain, and S. Wrobel, "Active Hidden Markov Models for Information Extraction", in *Advances in Intelligent Data Analysis*, 2001, pp. 309–318 (cit. on pp. 52, 155).

[162] C. E. Shannon, "A mathematical theory of communication", *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948 (cit. on p. 52).

[163] B. Settles, M. Craven, and S. Ray, "Multiple-instance active learning", in *Advances in Neural Information Processing Systems (NIPS)*, 2008, pp. 1289–1296 (cit. on pp. 52, 155).

[164] N. Roy and A. McCallum, "Toward Optimal Active Learning Through Sampling Estimation of Error Reduction", in *International Conference on Machine Learning (ICML)*, 2001, pp. 441–448 (cit. on pp. 52, 155).

[165] K. Wang, D. Zhang, Y. Li, R. Zhang, and L. Lin, "Cost-Effective Active Learning for Deep Image Classification", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 12, pp. 2591–2600, 2017 (cit. on pp. 52, 155).

[166] F. Stark, C. Hazirbas, R. Triebel, and D. Cremers, "CAPTCHA Recognition with Active Deep Learning", in *GCPR Workshop on New Challenges in Neural Computation*, 2015 (cit. on pp. 52, 53, 155).

[167] J.-J. Zhu and J. Bento, "Generative Adversarial Active Learning", *arXiv preprint arXiv:1702.07956*, 2017 (cit. on pp. 52, 53, 155).

[168] M. Fang, Y. Li, and T. Cohn, "Learning how to Active Learn: A Deep Reinforcement Learning Approach", in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017 (cit. on pp. 52, 155).

[169] O. Sener and S. Savarese, "Active Learning for Convolutional Neural Networks: A Core-Set Approach", *arXiv preprint arXiv:1708.00489*, 2017 (cit. on pp. 52, 53, 155).

[170] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network", *Canadian Journal of Mathematics*, vol. 8, pp. 399–404, Jan. 1956 (cit. on p. 55).

[171] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001 (cit. on p. 55).

[172] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization", *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 10, pp. 1568–1583, 2006 (cit. on p. 55).

[173] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision", *International Journal of Computer Vision*, vol. 70, no. 1, pp. 41–54, Oct. 2006 (cit. on p. 55).

[174] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Understanding Belief Propagation and its Generalizations", *Intelligence*, vol. 8, pp. 236–239, 2002 (cit. on p. 55).

[175] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm", *IEEE Transactions on Information Theory*, vol. 47, pp. 498–519, Feb. 2001 (cit. on p. 55).

[176] Y. Weiss and W. T. Freeman, "On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs", *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 736–744, 2001 (cit. on p. 55).

[177] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study", in *Conference on Uncertainty in Artificial Intelligence (UAI)*, vol. 9, 1999, pp. 467–475 (cit. on pp. 55, 89, 204).

[178] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "TextonBoost: Joint Appearance, Shape and Context Modeling for Multi-class Object Recognition and Segmentation", in *European Conference on Computer Vision (ECCV)*, 2006, pp. 1–15 (cit. on pp. 56, 198).

[179] B. Fulkerson, A. Vedaldi, and S. Soatto, "Class segmentation and object localization with superpixel neighborhoods", in *International Conference on Computer Vision (ICCV)*, 2009, p. 4 (cit. on p. 56).

[180] P. Kohli, L. Ladický, and P. H. S. Torr, "Robust higher order potentials for enforcing label consistency", *International Journal of Computer Vision*, vol. 82, no. 3, pp. 302–324, 2009 (cit. on p. 56).

[181] J. Verbeek and B. Triggs, "Scene Segmentation with Conditional Random Fields Learned from Partially Labeled Images", in *Advances in Neural Information Processing Systems (NIPS)*, 2007, pp. 1–8 (cit. on p. 56).

[182] P. Krähenbühl and V. Koltun, "Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials", in *Advances in Neural Information Processing Systems (NIPS)*, 2011, pp. 1–9 (cit. on pp. 56, 57, 86, 89, 191, 192, 194, 198, 206–208, 217, 221, 224).

[183] Y. Zhang and T. Chen, "Efficient Inference for Fully Connected CRFs with Stationarity", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012 (cit. on p. 56).

[184] N. D. F. Campbell, K. Subr, and J. Kautz, "Fully-connected CRFs with non-parametric pairwise potential", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 1658–1665 (cit. on pp. 56, 57, 198).

[185] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr, "Conditional Random Fields as Recurrent Neural Networks", in *International Conference on Computer Vision (ICCV)*, 2015 (cit. on pp. 57, 58, 192, 198).

[186] G. Lin, C. Shen, A. van dan Hengel, and I. Reid, "Efficient piecewise training of deep structured models for semantic segmentation", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016 (cit. on pp. 57, 58, 192).

[187] A. Arnab, S. Jayasumana, S. Zheng, and P. H. S. Torr, "Higher order conditional random fields in deep neural networks", in *European Conference on Computer Vision (ECCV)*, 2016, pp. 524–540 (cit. on pp. 57, 58, 198).

[188] Z. Liu, X. Li, P. Luo, C. Change, and L. X. Tang, "Semantic Image Segmentation via Deep Parsing Network", in *International Conference on Computer Vision (ICCV)*, 2015 (cit. on p. 57).

[189] X. Qi, S. Jianping, S. Liu, R. Liao, and J. Jia, "Semantic Segmentation With Object Clique Potentials", in *International Conference on Computer Vision (ICCV)*, 2015 (cit. on p. 57).

[190] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient Graph-Based Image Segmentation", *International Journal of Computer Vision*, pp. 1–26, 2004 (cit. on pp. 58, 218).

[191] C. Rother, V. Kolmogorov, and A. Blake, ""GrabCut": interactive foreground extraction using iterated graph cuts", *ACM Transactions on Graphics*, vol. 23, no. 3, p. 309, 2004 (cit. on p. 58).

[192] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization", *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011 (cit. on p. 65).

[193] M. D. Zeiler, "Adadelta: an adaptive learning rate method", *arXiv preprint arXiv:1212.5701*, 2012 (cit. on p. 65).

[194] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude", *COURSERA: Neural Networks for Machine Learning*, 2012 (cit. on p. 65).

[195] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014 (cit. on p. 65).

[196] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks", in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010, pp. 249–256 (cit. on p. 81).

[197] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: surpassing human-level performance on imagenet classification", in *International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034 (cit. on p. 81).

[198] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?", *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 625–660, 2010 (cit. on p. 81).

[199] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization", in *Advances in Neural Information Processing Systems (NIPS)*, 1992, pp. 950–957 (cit. on p. 85).

[200] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, "Caltech-UCSD Birds 200", California Institute of Technology, Tech. Rep. CNS-TR-2010-001, 2010 (cit. on pp. 93, 114, 115, 122, 130, 166).

[201] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization", in *International Conference on Computer Vision Workshops*, 2013, pp. 554–561 (cit. on pp. 93, 122, 130, 166).

[202] M.-E. Nilsback and A. Zisserman, "Automated Flower Classification over a Large Number of Classes", in *Indian Conference on Computer Vision, Graphics and Image Processing*, 2008 (cit. on pp. 93, 120, 130, 166).

[203] N. Kumar, P. N. Belhumeur, A. Biswas, D. W. Jacobs, W. J. Kress, I. Lopez, and J. V. B. Soares, "Leafsnap: A Computer Vision System for Automatic Plant Species Identification", in *European Conference on Computer Vision (ECCV)*, 2012 (cit. on pp. 93, 130).

[204] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: a dataset for recognising faces across pose and age", in *International Conference on Automatic Face and Gesture Recognition (IEEE FG)*, 2018 (cit. on p. 93).

[205] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: large-scale scene recognition from abbey to zoo", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 3485–3492 (cit. on p. 93).

[206] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: a 10 million image database for scene recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017 (cit. on p. 93).

[207] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", *arXiv preprint arXiv:1409.1556*, 2014 (cit. on pp. 93, 96, 101, 102, 114, 119, 120, 131, 142, 167).

[208] D. S. Broomhead and D. Lowe, "Radial basis functions, multi-variable functional interpolation and adaptive networks", DTIC Document, Tech. Rep., 1988 (cit. on pp. 94, 99).

[209] J. Xu, X. Zhang, and Y. Li, "Kernel neuron and its training algorithm", in *International Conference on Neural Information Processing (ICONIP)*, 2001, pp. 861–866 (cit. on p. 94).

[210] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov, "Neighbourhood components analysis", in *Advances in Neural Information Processing Systems (NIPS)*, 2005, pp. 513–520 (cit. on p. 94).

[211] R. Salakhutdinov and G. Hinton, "Learning a nonlinear embedding by preserving class neighbourhood structure", in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007, pp. 412–419 (cit. on p. 94).

[212] Y. Tang, "Deep learning using linear support vector machines", *arXiv preprint arXiv:1306.0239*, 2013 (cit. on p. 99).

[213] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.", *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014 (cit. on p. 111).

[214] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1 (cit. on p. 123).

[215] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE", *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008 (cit. on pp. 127–129, 180, 181, 186, 187).

[216] J. Wang, E. Sung, and W. Yau, "Active learning for solving the incomplete data problem in facial age classification by the furthest nearest-neighbor criterion", *IEEE Transactions on Image Processing*, vol. 20, no. 7, pp. 2049–2062, 2011 (cit. on pp. 152, 174).

[217] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey", *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009 (cit. on p. 154).

[218] K. Lee, H. Lee, K. Lee, and J. Shin, "Training confidence-calibrated classifiers for detecting out-of-distribution samples", in *International Conference on Learning Representations (ICLR)*, 2018 (cit. on p. 155).

[219] A. J. Joshi, F. Porikli, and N. Papanikolopoulos, "Multi-class active learning for image classification", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 2372–2379 (cit. on p. 155).

[220] X. Zhu, J. Lafferty, and Z. Ghahramani, "Combining active learning and semi-supervised learning using gaussian fields and harmonic functions", in *ICML 2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, vol. 3, 2003 (cit. on p. 155).

[221] D. Arthur and S. Vassilvitskii, "K-means++: The Advantages of Careful Seeding", in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007, pp. 1027–1035 (cit. on p. 185).

[222] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis", *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987 (cit. on p. 185).

[223] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik, "Hypercolumns for Object Segmentation and Fine-Grained Localization", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015 (cit. on p. 198).

[224] J. Dai, K. He, and J. Sun, "Instance-Aware Semantic Segmentation via Multi-Task Network Cascades", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016 (cit. on p. 198).

[225] M. A. Reza and J. Kosecka, "Object Recognition and Segmentation in Indoor Scenes from RGB-D Images", *Robotics Science and Systems (RSS) conference-5th workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2014 (cit. on p. 199).

[226] Q.-X. Huang, M. Han, B. Wu, and S. Ioffe, "A hierarchical conditional random field model for labeling and segmenting images of street scenes", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 1953–1960 (cit. on p. 199).

[227] M. Reza and J. Kosecka, "Reinforcement Learning for Semantic Segmentation in Indoor Scenes", *arXiv preprint arXiv:1606.01178*, 2016 (cit. on p. 199).

[228] J. Yao, S. Fidler, and R. Urtasun, "Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation.", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 702–709 (cit. on p. 199).

[229] S. Kumar and M. Hebert, "A hierarchical field framework for unified context-based classification", in *International Conference on Computer Vision (ICCV)*, vol. 2, Oct. 2005, 1284–1291 Vol. 2 (cit. on p. 199).

[230] L. Ladický, C. Russell, P. Kohli, and P. H. S. Torr, "Associative hierarchical CRFs for object class image segmentation", in *International Conference on Computer Vision (ICCV)*, 2009, pp. 739–746 (cit. on p. 199).

[231] C. Russell, L. Ladicky, P. Kohli, and P. H. S. Torr, "Exact and Approximate Inference in Associative Hierarchical Networks using Graph Cuts", in *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2010, pp. 501–508 (cit. on p. 199).

[232] S. Gupta, P. Arbelaez, and J. Malik, "Perceptual organization and recognition of indoor scenes from RGB-D images", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 564–571 (cit. on p. 220).

[233] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik, "Semantic Contours from Inverse Detectors", in *International Conference on Computer Vision (ICCV)*, 2011 (cit. on p. 224).