Geometry and Uncertainty in Deep Learning for Computer Vision

Alex Guy Kendall

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Trinity College November 2017

“It is not the mountain we conquer but ourselves.”
Sir Edmund Hillary (1919 - 2008)

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Alex Guy Kendall
November 2017

Acknowledgements

The majority of this thesis was written while travelling across eight countries and three continents. In many ways, this reflects the quantity of experiences I encountered during my doctorate. I feel fortunate enough to not have experienced the stereotypical solitary research life and there are many people I would like to thank for making this work stimulating, challenging and fun.

Firstly, I would like to thank my supervisor, Professor Roberto Cipolla, for his unequivocal guidance and support. He has given my mind substantial room to wander for the last three and a bit years. His enthusiasm for new ideas is relentless, no matter how crazy the idea. Thank you for the wonderful opportunities you have opened for me.

Secondly, I am indebted to the Woolf Fisher Trust for blindly putting faith in my studies and providing the financial support for this research at the University of Cambridge. The community of outstanding New Zealanders they have formed in Cambridge is incredibly inspiring. I hope they continue to support education and push the boundaries of young New Zealanders.

I would like to thank a number of colleagues who I was fortunate to learn from and collaborate with to produce the research in this thesis. In alphabetical order: Abe Bachrach, Vijay Badrinarayanan, Adam Bry, Saumitro Dasgupta, Yarin Gal, Matthew Grimes, Ankur Handa, Peter Henry, Yani Ioannau, Ryan Kennedy, Hayk Martirosyan, Thomas Roddick, Torsten Sattler, Amar Shah and Jamie Shotton.

Thank you to Trinity College, in particular the Trinity College Zoo, for providing a welcoming and supportive community. Most importantly, I would like to thank my family — Tania, Guy, Fiona and Joce — and my friends, who kept the conversation scholarly for only what was necessary. In particular, a big thank you to my fiancée, Ali Stone, for keeping me alive and motivated throughout this whole adventure.

Abstract

Deep learning and convolutional neural networks have become the dominant tool for computer vision. These techniques excel at learning complicated representations from data using supervised learning. In particular, image recognition models now out-perform human baselines under constrained settings. However, the science of computer vision aims to build machines which can see. This requires models which can extract richer information than recognition, from images and video. In general, applying these deep learning models from recognition to other problems in computer vision is significantly more challenging.

This thesis presents end-to-end deep learning architectures for a number of core computer vision problems; scene understanding, camera pose estimation, stereo vision and video semantic segmentation. Our models outperform traditional approaches and advance state-of-the-art on a number of challenging computer vision benchmarks. However, these end-to-end models are often not interpretable and require enormous quantities of training data.

To address this, we make two observations: (i) we do not need to learn everything from scratch, we know a lot about the physical world, and (ii) we cannot know everything from data, our models should be aware of what they do not know. This thesis explores these ideas using concepts from geometry and uncertainty. Specifically, we show how to improve end-to-end deep learning models by leveraging the underlying geometry of the problem. We explicitly model concepts such as epipolar geometry to learn with unsupervised learning, which improves performance. Secondly, we introduce ideas from probabilistic modelling and Bayesian deep learning to understand uncertainty in computer vision models. We show how to quantify different types of uncertainty, improving safety for real world applications.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Motivation

Vision is the process of discovering from images what is present in the world and where it is (Marr, 1982). It is an extremely complicated sense, but provides the most powerful cues of our environment. Our eyes capture ten gigabits per second of information from the world around us (Anderson et al., 2005). The human brain is able to process over three million bits per second of this information (Anderson et al., 2005). Our brain can use this information to learn a remarkably rich representation of the world around us (Barlow, 1989). However, developing an artificial system which can achieve the same performance and robustness as humans has long challenged researchers from fields as diverse as physiology, philosophy, psychology, engineering, computer science and artificial intelligence.

Computer vision is a multidisciplinary science that strives to give machines the ability to see (Szeliski, 2010). This problem is particularly challenging because of the vast complexity and variation in appearance we observe from our visual world. To date, designing a hand-engineered approach has not been able to scale to a satisfactory level of understanding (Papert, 1966; Roberts, 1963). Machine learning techniques (Bishop, 2006) provide the most promising approach for designing systems with human-level understanding of imagery.

As a science, computer vision is having a profound impact on many disruptive areas of technology. The contributions of this thesis are novel machine learning architectures which address many of the core computer vision problems. The architectures proposed in this work are practical, real-time systems which are influencing the development of today's technology, including autonomous vehicles, augmented reality, medical imaging, drones and smart-city infrastructure. Moreover, being able to build intelligent vision models may contribute to our understanding of the neuroscience behind visual intelligence (Sterling and Laughlin, 2015).

1.2 Approach

Deep learning (Goodfellow et al., 2016) is now ubiquitous in the field of computer vision. As a machine learning tool, deep neural networks are very effective at understanding high-dimensional data, such as images. They learn representations by encoding the input through a number of non-linear layers and sub-sampling operations, resulting in powerful image-level understanding and recognition capabilities. Deep learning models were first used in computer vision for image recognition tasks (Krizhevsky et al., 2012; LeCun, 1988). However, the science of computer vision aims to build machines which can see. This requires models which can extract richer information from images and video than recognition. In general, applying these deep learning models from recognition to other problems in computer vision is significantly more challenging.

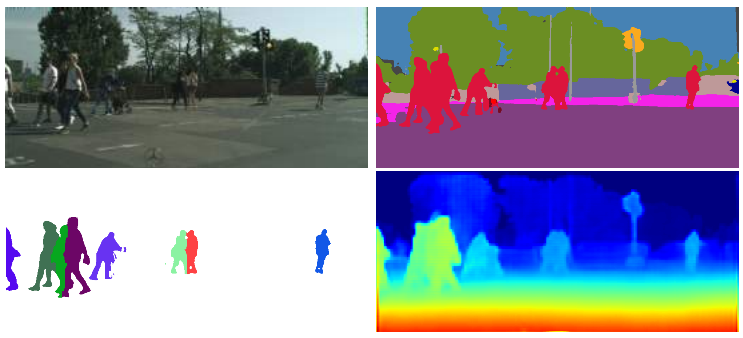
This thesis shows how to formulate deep learning models for many core computer vision tasks, and advances the state-of-the-art of each task with practical, real-time models:

1. Semantic segmentation (what is around us),
2. Instance segmentation (where objects are),
3. Monocular metric depth (how far away objects are),
4. Camera pose (where we are),
5. Stereo disparity (depth from binocular vision),
6. Optical flow (motion of objects in an image),
7. Video semantic segmentation (where objects are in video).

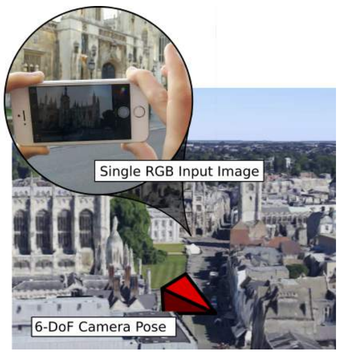
An overview of these results is given in Figure 1.1. Collectively, these new methods advance state-of-the-art, with many out-performing previously published approaches to these problems.

The algorithms we propose rely on deep learning (Hinton et al., 2006). Deep learning has emerged as a powerful paradigm for understanding high dimensional data, such as visual imagery. It is now the core technology behind natural language understanding (Sutskever et al., 2014), image recognition (Krizhevsky et al., 2012), speech recognition (Hinton et al., 2012) and many other challenging, high dimensional problems in biology and physics (Ciodaro et al., 2012; Helmstaedter et al., 2013; Leung et al., 2014).

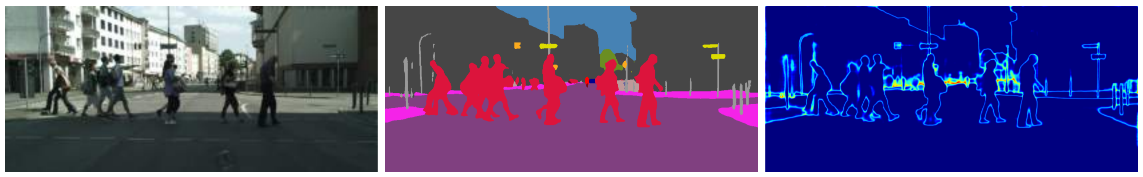
Deep learning models form hierarchical layers of increasing levels of abstraction (Goodfellow et al., 2016). These models are typically optimised by propagating a training signal



(a) Scene Understanding — understanding the geometry and semantics of a scene. Clockwise from top-left: input image, semantic segmentation, depth prediction, instance segmentation.



(b) Localisation — estimating the camera's 3D position and orientation from an image.



(c) Uncertainty — understanding the prediction's confidence and what the model does not know. From left: input image, semantic segmentation, model uncertainty (where blue represents certain and more red colours are uncertain predictions). The model exhibits increased uncertainty in distance objects and around object boundaries.

Fig. 1.1 Examples of the variety of algorithms developed in this thesis. (a) shows a scene understanding system from Chapter 2. (b) shows a localisation system which can determine the camera's 3D position and orientation in space from Chapter 3. (c) shows a semantic segmentation system which is aware of its uncertainty from Chapter 2.

from the output all the way to the input. This is known as end-to-end learning. End-to-end learning models have many advantages. Firstly, they can obtain higher performance by optimising all layers with regards to the end goal. Secondly, they are more scalable and can reduce engineering effort by enabling learning from vast amounts of training data. We show that we can formulate and train models on many computer vision tasks with end-to-end learning, outperforming prior approaches.

But end-to-end learning has problems too. It is often not good for data efficiency, interpretability or safety (McAllister et al., 2017). This thesis makes two observations which challenge the weaknesses of end-to-end learning:

- We do not need to learn everything from scratch, we know many things about the world,
- We cannot explain everything from our data and we need to know what our model does not know.

In this thesis, we focus on these ideas using concepts from ***geometry*** and ***uncertainty***. These ideas are elaborated on in the next few sections.

1.2.1 Machine Learning in Computer Vision

Let us consider two arguments which motivate the use of machine learning approaches, in particular deep learning, to complicated computer vision problems.

First, as the most powerful demonstration of perception, how does the human visual pathway develop? Surprisingly, we are born without the ability to see (Gibson and Walk, 1960). Infant humans learn through a mixture of semi-supervised and unsupervised learning (Kellman and Arterberry, 2006). It typically takes three months to learn colour and depth perception. Object detection can take nine to twelve months. Suppose that an infant experiences one saccade (visual experience) per second of their first year of existence. This amounts to 1 saccade/s $\times$ 3600 sec/hour $\times$ 8 h/day $\times$ 365 days/year $= 10,000,000$ training examples in the first year of their life. Interestingly, this size is of the same order of magnitude as one of the largest computer vision datasets, ImageNet (Deng et al., 2009). It is also the size of the data which is required for today's image recognition models to achieve super-human performance (He et al., 2016). As our best example, the human visual system learns to see — through nurture, not nature.

Secondly, it simply is not scalable to hand-design an algorithm to support and adapt to complicated and dynamic data such as our visual world. Modern computer vision algorithms contain over 100 million parameters. Input data is very high dimensional, typically containing several million pixels being streamed over time. To date, we have been unable to specify hand-engineered rules to design top-performing architectures. To understand large amounts of data at scale we need to learn.

While there are many examples of machine learning models, *deep convolutional neural networks* (Fukushima, 1979; Krizhevsky et al., 2012) are most effective for vision tasks. Convolutional neural networks are a subset of deep learning which are particularly useful for computer vision because they are spatially invariant. Algorithms like stochastic gradient descent (Kiefer et al., 1952) and backpropagation (Rumelhart et al., 1986) can train networks containing millions of parameters. These networks can be optimised from a loss function formulated from supervised labelled training data or unsupervised learning.

Deep learning models have typically been restricted to classification settings. They excel at sub-sampling data and expanding feature dimensions to make classifications at an image level (Krizhevsky et al., 2012). Applying these models to other problems in computer vision is more challenging. This is because other computer vision tasks require representations for recognition, registration or reconstruction (Cipolla et al., 2010). Standard recognition encoder networks (He et al., 2004; Krizhevsky et al., 2012; Simonyan et al., 2013) cannot be naively applied to these domains. This thesis proposes a broader range of deep learning architectures for many other computer vision problems.

Finally, there is an interesting comparison between deep learning models and the human visual system. The human visual cortex processes images from the retina initially through the V1-V4 and IT cortex (Hubel and Wiesel, 1962). Evidence shows that hierarchical representations are formed at increasing levels of abstraction (edges, lines, contours, objects, scenes) (Hubel and Wiesel, 1962). Similar levels of abstraction are seen with increasing convolutional neural network depth (Zeiler and Fergus, 2014).

1.2.2 Geometry in Computer Vision

Geometry is concerned with questions of shape, size, relative position of figures and the properties of space. In computer vision, geometry is used to describe the structure and shape of the world. Specifically, it concerns measures such as depth, volume, shape, pose, disparity, motion or optical flow. We understand the mathematics of geometry very well (Faugeras, 1993; Hartley and Zisserman, 2000; Koenderink, 1990). Consequently, there are a lot of complex relationships, such as depth and motion, which do not need to be learned from scratch with deep learning. By building architectures which use this knowledge, we can simplify the learning problem.

The alternative paradigm to geometry is using semantic representations. Semantic representations use a language to describe relationships in the world. For example, we might describe an object as a ‘cat’ or a ‘dog’. Geometry has two attractive characteristics over semantics:

- Geometry can be directly observed. We see the world’s geometry directly using vision. At the most basic level, we can observe motion and depth directly from a video by following corresponding pixels between frames (Koenderink and Van Doorn, 1991). Other interesting examples include observing shape from shading (Horn and Brooks, 1989) or depth from stereo disparity (Scharstein and Szeliski, 2002). In contrast, semantic representations are often proprietary to a human language, with labels corresponding to a limited set of nouns, which cannot be directly observed from the world.

- Geometry is based on continuous quantities. For example, we can measure depth in meters or disparity in pixels. In contrast, semantic representations are largely discretised quantities or binary labels.

With these properties, we show that we can use geometry to improve performance (Chapter 3 and Chapter 4) and for unsupervised learning, without human-annotated labels (Chapter 4 and Chapter 5).

However, geometry alone cannot form a robust vision system. This is because models must learn representations which are robust to noise and outliers. Deep learning is powerful at learning representations which are robust to outliers and other nuisance variables. Additionally, models also need to be aware of the inherent uncertainty in the data.

1.2.3 Uncertainty in Computer Vision

Understanding what a model does not know is a critical part of many machine learning systems (Ghahramani, 2015). Uncertainty is important to improve trustworthiness and safety of these systems (McAllister et al., 2017). Unfortunately, today's deep learning algorithms are usually unable to understand their uncertainty. These models are often taken blindly and assumed to be accurate, which is not always the case. For example, in two recent situations this has had disastrous consequences.

- In May 2016 the world tragically experienced the first fatality from an assisted driving system. According to the manufacturer's blog, 'Neither Autopilot nor the driver noticed the white side of the tractor trailer against a brightly lit sky, so the brake was not applied' (NHTSA, 2017).
- In July 2015, an image classification system erroneously identified two African American humans as gorillas, raising concerns of racism and discrimination (Guynn, 2015).

If both these algorithms could assign a high level of uncertainty to their erroneous predictions, then each system may have been able to make better decisions and likely avoid disaster. Unfortunately, traditional machine learning approaches to understanding uncertainty, such as Gaussian processes (Rasmussen and Williams, 2006), do not scale to high dimensional inputs like images and videos. To effectively understand this data, we need deep learning. But deep learning struggles to model uncertainty.

One of the techniques used in this thesis to model uncertainty is Bayesian deep learning (MacKay, 1992) which provides a deep learning framework which can model uncertainty. Bayesian deep learning is a field at the intersection between deep learning and Bayesian probability theory. It offers principled uncertainty estimates from deep learning architectures. These deep architectures can model complex tasks by leveraging the hierarchical representation power of deep learning, while also being able to infer complex multi-modal posterior distributions. Probabilistic deep learning models typically form uncertainty estimates by either placing distributions over model weights, or by learning a direct mapping to probabilistic outputs. In this thesis we show how to formulate accurate and scalable computer vision models which can understand the model's uncertainty with Bayesian deep learning.

1.3 Contributions

To summarise, the contributions of this thesis are as follows:

- We demonstrate how to formulate many challenging computer vision problems with end-to-end deep learning. We show the performance of these models significantly improves over traditional approaches.
- We show how to improve performance of these models by leveraging the problem's geometry. We demonstrate that this reduces the amount of training data required and improves the generalisation of these models to novel examples.
- Finally, for practical and safe systems, it is important to understand our model's uncertainty. We show how to quantify uncertainty in deep learning computer vision models with Bayesian deep learning and probabilistic modelling.

1.4 Co-Authored Papers

Some extracts from this thesis appear in the following co-authored publications and preprints. Chapter 2 contains work from:

- Vijay Badrinarayanan, Alex Kendall and Roberto Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.
- Alex Kendall and Yarin Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? Advances in Neural Information Processing Systems, 2017.
- Alex Kendall, Yarin Gal and Roberto Cipolla. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.

Chapter 3 is adapted from:

- Alex Kendall, Matthew Grimes and Roberto Cipolla. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. Proceedings of the International Conference on Computer Vision, 2015.
- Alex Kendall and Roberto Cipolla. Modelling Uncertainty in Deep Learning for Camera Relocalization. Proceedings of the International Conference on Robotics and Automation, 2016.
- Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

Chapter 4 extends:

- Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-End Learning of Geometry and Context for Deep Stereo Regression. Proceedings of the International Conference on Computer Vision, 2017.

And finally, Chapter 5 is adapted from:

- Alex Kendall and Roberto Cipolla. Learning Semantics, Motion and Geometry for Video Scene Understanding. Under Review, 2017.

1.5 Thesis Structure

The outline of the thesis is as follows. The following four chapters discuss core computer vision tasks: scene understanding in Chapter 2, localisation in Chapter 3, stereo vision in Chapter 4, video and motion in Chapter 5. For each chapter's topic, we review the prior art. We introduce formulations for end-to-end deep learning architectures. We discuss how to improve these end-to-end models with notions of the problem's geometry. Finally, we show how to capture model uncertainty. In Chapter 6, we make overall conclusions, discuss the application of this technology and suggest directions for future research.

Chapter 2

Scene Understanding

2.1 Introduction

In this chapter, we address the problem of scene understanding. Scene understanding is a task which is fundamental to computer vision. It requires extracting information about what is where (Marr, 1982). Visual scene understanding algorithms must extract information about objects and their contextual relationships from the observed environment. Scene understanding requires knowledge of semantics and geometry and can be broken down into many subtasks. In this work we focus on the tasks of semantic segmentation, instance segmentation and depth prediction. Collectively, these three representations provide a fine-grained, per-pixel representation of objects and their geometry.

We begin the chapter by proposing a deep convolutional encoder-decoder architecture, SegNet, capable of learning per-pixel output. Motivated by the problem of semantic segmentation, we analyse different architectures for upsampling features and producing dense pixel-wise output. We benchmark the efficacy of SegNet on many datasets on road scene, indoor and object segmentation datasets. Some results are shown in Figure 2.1.

For practical systems, it is important to understand what our models do not know. In Section 2.4, we discuss two types of uncertainty; aleatoric and epistemic uncertainty (Der Kiureghian and Ditlevsen, 2009). We derive deep learning models which can capture both forms of uncertainty using Bayesian deep learning (MacKay, 1992) and probabilistic modelling. We build models for semantic segmentation and per-pixel depth estimation with these ideas. We show modelling uncertainty gives an improvement in performance. We

In this Chapter, Section 2.2 and Section 2.3 was collaborative work with Vijay Badrinarayanan and Roberto Cipolla and was published in (Badrinarayanan et al., 2017). Section 2.4 and Section 2.5 was collaborative work with Yarin Gal and was published in (Kendall and Gal, 2017) and (Kendall et al., 2017b), respectively.

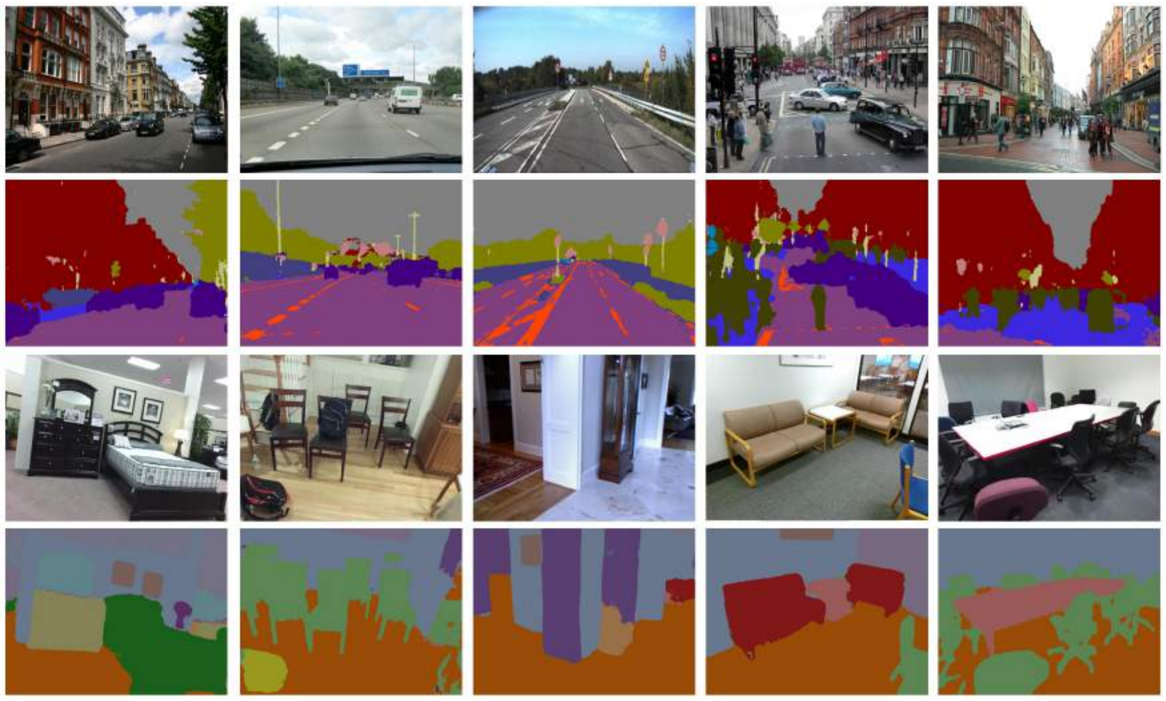


Fig. 2.1 SegNet predictions on indoor and outdoor scene test samples from the wild. To try our system yourself, please see our online web demo at http://mi.eng.cam.ac.uk/projects/segnet/.

empirically evaluate the quality of these uncertainty metrics and their properties with respect to novel examples, with increasing distinction from the training dataset.

However, scene understanding requires joint knowledge of semantics and geometry. In Section 2.5, we present a framework for designing models capable of learning many tasks from a single representation, using multi-task learning. We make the observation that the relative weighting of each task's loss greatly influences performance. We derive a loss function which learns the task weights using probabilistic modelling. The final model jointly predicts semantic segmentation, instance segmentation and depth regression. Interestingly, we show that jointly learning these tasks in a single multi-task model out-performs equivalent models individually trained on each task.

2.2 Semantic Segmentation

We begin by introducing the task of semantic segmentation, which is a core task in scene understanding. Semantic pixel-wise segmentation requires an estimate of each pixel's semantic class (see Figure 2.1). It is an active topic of research, fuelled by challenging datasets (Brostow et al., 2009; Cordts et al., 2016; Everingham et al., 2015; Geiger et al., 2012; Silberman et al., 2012; Song et al., 2015). Before the arrival of deep learning, the best performing methods mostly relied on hand engineered features classifying pixels independently. Typically, an image patch was fed into a classifier, *e.g.* random forests (Brostow et al., 2008; Shotton et al., 2008) or boosting (Ladickỳ et al., 2010; Sturgess et al., 2009), to predict the class probabilities of the centre pixel. Features based on appearance (Shotton et al., 2008) or motion and appearance (Brostow et al., 2008; Ladickỳ et al., 2010; Sturgess et al., 2009) have been explored. These per-pixel noisy predictions (often called *unary* terms) from the classifiers are then smoothed by using pair-wise (or higher order) conditional random fields (CRFs) (Ladickỳ et al., 2010; Sturgess et al., 2009) to improve the accuracy. More recent approaches have aimed to produce high quality unaries by trying to predict the labels for all the pixels in a patch as opposed to only the centre pixel. This improves the results of random forest based unaries (Kontschieder et al., 2011), but reduces performance on thin structures. Dense depth maps have also been used as input for classification using Random Forests (Zhang et al., 2010). Another approach argues for the use of a combination of popular hand designed features and spatio temporal super-pixels to obtain higher accuracy (Tighe and Lazebnik, 2013).

Indoor RGB-D pixel-wise semantic segmentation has also gained popularity since the release of the NYU dataset (Silberman et al., 2012) which contains labelled RGB-D data collected from a Kinect sensor. Many papers demonstrate that inputting depth modality significantly improves segmentation performance (Gupta et al., 2013; Hermans et al., 2014; Ren et al., 2012; Silberman et al., 2012). However, all these methods use hand-engineered features for classifying RGB-D images.

The success of deep convolutional neural networks for object classification has more recently led to researchers to exploit their feature learning capabilities for structured prediction problems such as segmentation. There have also been attempts to apply networks designed for object categorization to segmentation, particularly by replicating the deepest layer features in blocks to match image dimensions (Farabet et al., 2012, 2013; Gatta et al., 2014; Grangier et al., 2009). However, the resulting classification is coarse (Grangier et al., 2009). Another approach using recurrent neural networks (Pinheiro and Collobert, 2014) merges several low resolution predictions to create input image resolution predictions. These techniques are already an improvement over hand engineered features (Farabet et al., 2013) but their ability to delineate boundaries is poor.

Newer deep architectures (Eigen and Fergus, 2015; Hong et al., 2015; Long et al., 2015; Noh et al., 2015; Zheng et al., 2015) particularly designed for segmentation have advanced the state-of-the-art by learning to decode or map low resolution image representations to pixel-wise predictions. These methods rely on pre-trained features from the large ImageNet object classification dataset (Deng et al., 2009). For example, Fully Convolutional Networks

(FCN) (Long et al., 2015) learn to up-sample its input feature map(s) and combines them with the corresponding encoder feature map to produce a dense output. It has a large number of trainable parameters in the encoder network (134M) but a very small decoder network (0.5M). The overall large size of this network makes it hard to train end-to-end on a relevant task. Therefore, the authors use a stage-wise training process. Each decoder in the decoder network is progressively added to an existing trained network. The network is grown until no further increase in performance is observed.

The approach in FCN forms the *core segmentation engine* for a number of other approaches (Chen et al., 2016; Liu et al., 2015c; Schwing and Urtasun, 2015; Zheng et al., 2015). These methods append CRFs as a post-processing method to clean the segmentation. These methods are slow as they require either MAP inference over a CRF (Lin et al., 2015), (Schwing and Urtasun, 2015) or aids such as region proposals (Noh et al., 2015) for inference. We believe the perceived performance increase obtained by using a CRF is due to the lack of good decoding techniques in their core feed-forward segmentation engine.

Multi-scale deep architectures are also being pursued (Eigen and Fergus, 2015; Hariharan et al., 2015; Lin et al., 2015; Liu et al., 2015c). The common idea is to use features extracted at multiple scales to provide both local and global context (Mostajabi et al., 2014). Feature maps from the early encoding layers retain more high frequency detail leading to sharper class boundaries. Some of these architectures are difficult to train due to their large parameter size (Eigen and Fergus, 2015). Again, a multi-stage training process is employed along with data augmentation. Inference is also expensive with multiple convolutional pathways for feature extraction.

All of the latest state-of-the-art semantic segmentation models use supervised deep learning (Badrinarayanan et al., 2017; Long et al., 2015), benefiting from residual architectures (He et al., 2016; Huang et al., 2017). Recent work has focused on improving the receptive field of features and providing them with more context for semantic reasoning, for example using dilated convolutions (Yu and Koltun, 2016) and pyramid spatial pooling (Zhao et al., 2017). We have also seen semantic segmentation combined with other tasks, such as instance segmentation (He et al., 2017) and geometry (see Section 2.5) in multi-task learning settings.

In the next section, we introduce the SegNet architecture, which was one of the first end-to-end deep convolutional neural networks for semantic segmentation. We compare the different decoding techniques to form pixel-wise prediction with deep learning and benchmark our approach on a number of challenging datasets.

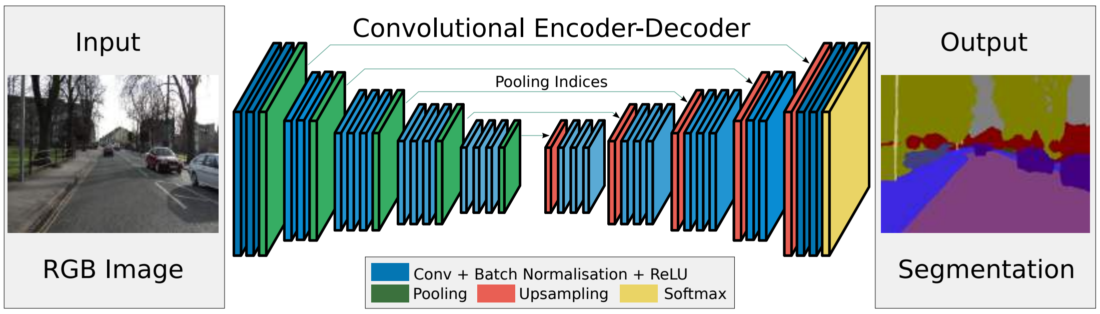


Fig. 2.2 An illustration of the SegNet architecture. There are no fully connected layers and hence it is only convolutional. The decoder upsamples the encoded features using the corresponding pooling indices from the encoder to produce sparse feature maps. It then performs convolution with a trainable filter bank to densify the feature map. The final decoder output feature maps are fed to a softmax classifier for pixel-wise classification.

2.3 SegNet Architecture

The latest semantic segmentation architectures have tried to directly adopt deep architectures designed for category prediction to pixel-wise labelling (Farabet et al., 2013; Long et al., 2015). The results, although very encouraging, appear coarse (Chen et al., 2016). This is primarily because max-pooling and sub-sampling reduce feature map resolution in the encoder for recognition tasks. In this section, we design SegNet, a deep convolutional encoder-decoder architecture capable of outputting pixel-wise labels at full resolution. Our motivation to design SegNet arises from this need to map low resolution features to input resolution for pixel-wise classification. This mapping must produce features which are useful for accurate boundary localization.

Our architecture, SegNet, is designed to be a *core segmentation engine* for pixel-wise semantic segmentation. It is primarily motivated by road scene understanding applications which require the ability to model appearance (road, building), shape (cars, pedestrians) and understand the spatial-relationship (context) between different classes such as road and side-walk. In typical road scenes, the majority of the pixels belong to large classes such as road, building or sky and hence the network must produce smooth segmentations. The engine must also have the ability to delineate moving and other objects based on their shape despite their small size. Hence it is important to retain boundary information in the extracted image representation. From a computational perspective, it is necessary for the network to be efficient in terms of both memory and computation time during inference. It must also be able to train end-to-end in order to jointly optimise all the weights in the network using an efficient weight update technique such as stochastic gradient descent (SGD) (Bottou, 2010). Networks that are trained end-to-end or equivalently those that do not use multi-stage training

(Long et al., 2015) or other supporting aids such as region proposals (Noh et al., 2015) help establish benchmarks that are more easily repeatable. The design of SegNet arose from a need to match these criteria.

SegNet has an encoder network and a corresponding decoder network, followed by a final pixel-wise classification layer. This architecture is illustrated in Figure 2.2. The encoder network consists of 13 convolutional layers which is topologically identical to the convolutional layers in VGG16 (Simonyan and Zisserman, 2014). We can therefore initialize the training process from weights trained for classification on large datasets (Deng et al., 2009). We remove the fully connected layers of VGG16 which makes the SegNet encoder network significantly smaller (from 134 million parameters to 14.7 million) than many other recent architectures (Hong et al., 2015; Liu et al., 2015c; Long et al., 2015; Noh et al., 2015).

The key component of SegNet is the decoder network which consists of a hierarchy of decoders one corresponding to each encoder. Of these, the appropriate decoders use the max-pooling indices received from the corresponding encoder to perform non-linear upsampling of their input feature maps. This idea was inspired from an architecture designed for unsupervised feature learning (Ranzato et al., 2007). Reusing max-pooling indices in the decoding process has several practical advantages; (i) it improves boundary delineation, (ii) it reduces the number of parameters enabling end-to-end training, and (iii) this form of upsampling can be incorporated into any encoder-decoder architecture such as Long et al. (2015) or Zheng et al. (2015) with slight modification. The final decoder output is fed to a multi-class softmax classifier to produce class probabilities for each pixel independently.

Each *encoder* in the encoder network performs convolution with a filter bank to produce a set of feature maps. These are then batch normalized (Ioffe and Szegedy, 2015)). Then an element-wise rectified-linear non-linearity (ReLU) $max(0,x)$ is applied. Following that, max-pooling with a 2×2 window and stride 2 (non-overlapping window) is performed and the resulting output is sub-sampled by a factor of 2. Max-pooling is used to achieve translation invariance over small spatial shifts in the input image. Sub-sampling results in a large receptive field for each pixel in the feature map. However, sub-sampling reduces spatial resolution of the features which is not beneficial for segmentation where boundary delineation is vital. Therefore, it is necessary to *capture and store* boundary information in the encoder feature maps before sub-sampling is performed. If memory during inference is not constrained, then all the encoder feature maps (after sub-sampling) can be stored. This is usually not the case in practical applications and hence we propose a more efficient way to store this information. It involves storing only the max-pooling *indices*, i.e, the locations of the maximum feature value in each pooling window is memorized for each encoder feature map. In principle, this can be done using 2 bits for each 2×2 pooling window and is thus much more efficient to store as compared to memorizing feature map(s) in float precision (such as U-Net (Ronneberger et al., 2015)). As we show later in this work, this lower memory storage results in a slight loss of accuracy but is still suitable for practical applications.

The appropriate *decoder* in the decoder network upsamples its input feature map(s) using the memorized max-pooling indices from the corresponding encoder feature map(s). This step produces sparse feature map(s). This SegNet decoding technique is illustrated in Figure 2.3. These feature maps are then convolved with a trainable decoder filter bank to produce dense feature maps. A batch normalization step is then applied to each of these maps. Note that the decoder corresponding to the first encoder (closest to the input image) produces a multi-channel feature map, although its encoder input has 3 channels (RGB). This is unlike the other decoders in the network which produce feature maps with the same number of size and channels as their encoder inputs. The high dimensional feature representation at the output of the final decoder is fed to a trainable soft-max classifier. This soft-max classifies each pixel independently. The output of the soft-max classifier is a K channel image of probabilities where K is the number of classes. The predicted segmentation corresponds to the class with maximum probability at each pixel.

In the following Sections, we evaluate the performance of SegNet on a number of datasets (Everingham et al., 2015; Hariharan et al., 2011) and scene understanding challenges such as CamVid road scene segmentation (Brostow et al., 2009). We analyse different decoding techniques and the practical trade-offs when designing segmentation architecture. In addition, we present a real-time online demo of road scene segmentation into 11 classes of interest for autonomous driving (see Figure 2.1). Some example test results produced on randomly sampled road scene images from the internet are shown in Figure 2.1.

2.3.1 Decoder Variants

One of the main contributions of this Section is our analysis of decoding techniques for semantic segmentation. Most recent deep architectures for segmentation have identical encoder networks, *i.e.* VGG16 (Simonyan and Zisserman, 2014) or ResNet101 (He et al., 2016), but differ in the form of the decoder network, training and inference. Another common feature is they have trainable parameters in the order of hundreds of millions and thus encounter difficulties in performing end-to-end training (Noh et al., 2015). The difficulty of training these networks has led to multi-stage training (Long et al., 2015), appending networks to a pre-trained core segmentation engine such as FCN (Zheng et al., 2015), use of supporting aids such as region proposals for inference (Noh et al., 2015), disjoint training of classification and segmentation networks (Hong et al., 2015) and use of additional training data for pre-training (Liu et al., 2015c) (Mottaghi et al., 2014) or for full training (Zheng et al.,

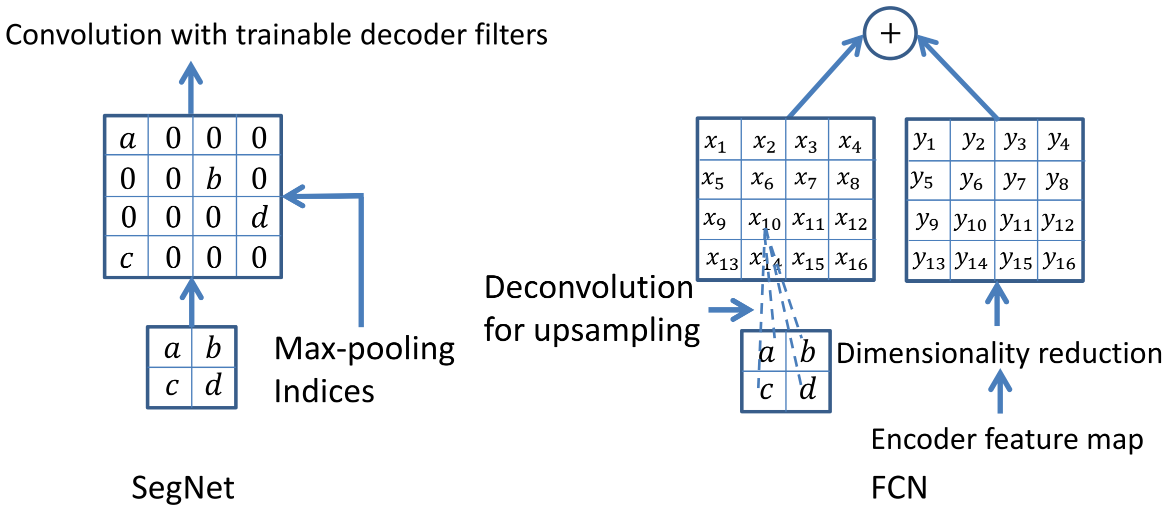


Fig. 2.3 An illustration of SegNet and FCN (Long et al., 2015) decoders. a, b, c, d correspond to values in a feature map. SegNet uses the max pooling indices to upsample (without learning) the feature map(s) and convolves with a trainable decoder filter bank. FCN upsamples by learning to deconvolve the input feature map and adds the corresponding encoder feature map to produce the decoder output. This feature map is the output of the max-pooling layer (includes sub-sampling) in the corresponding encoder. Note that there are no trainable decoder filters in FCN.

2015). In addition, performance boosting post-processing techniques (Chen et al., 2016) have also been popular. Although all these factors improve performance on challenging benchmarks (Everingham et al., 2015), it is unfortunately difficult from their quantitative results to disentangle the key design factors necessary to achieve good performance. We therefore analyse many segmentation decoders in a controlled setting.

In order to analyse SegNet and compare its performance with other decoder variants we use a smaller version of SegNet, termed **SegNet-Basic**, which has 4 encoders and 4 decoders. All the encoders in SegNet-Basic perform max-pooling and sub-sampling and the corresponding decoders upsample its input using the received max-pooling indices. Batch normalization is used after each convolutional layer in both the encoder and decoder network. No biases are used after convolutions and no ReLU non-linearity is present in the decoder network. Further, a constant kernel size of 7×7 over all the encoder and decoder layers is chosen to provide a wide context for smooth labelling *i.e.* a pixel in the deepest layer feature map (layer 4) can be traced back to a context window in the input image of 106×106 pixels. This small size of SegNet-Basic allows us to explore many different variants (decoders) and train them in reasonable time. Similarly we create **FCN-Basic**, a comparable version of FCN for our analysis which shares the same encoder network as SegNet-Basic but with the FCN decoding technique in the corresponding decoders.

On the left in Figure 2.3 is the decoding technique used by SegNet (also SegNet-Basic), where there is no learning involved in the upsampling step. However, the upsampled maps are convolved with trainable multi-channel decoder filters to densify the sparse inputs. Each decoder filter has the same number of channels as the number of upsampled feature maps. A smaller variant is one where the decoder filters are single channel, i.e they only convolve their corresponding upsampled feature map. This variant (**SegNet-Basic-SingleChannelDecoder**) reduces the number of trainable parameters and inference time significantly.

On the right in Figure 2.3 is the FCN (also FCN-Basic) decoding technique (Long et al., 2015). The important design element of the FCN model is the dimensionality reduction step of the encoder feature maps. This *compresses* the encoder feature maps which are then used in the corresponding decoders. Dimensionality reduction of the encoder feature maps, say of 64 channels, is performed by convolving them with $1 \times 1 \times 64 \times K$ trainable filters, where K is the number of classes. The compressed K channel final encoder layer feature maps are the input to the decoder network. In a decoder of this network, upsampling is performed by convolution using a trainable *multi-channel upsampling kernel*. We set the kernel size to 8×8. This manner of upsampling is also termed as *sub-pixel convolution*, *deconvolution* or *transposed convolution*. Note that in SegNet the multi-channel convolution using trainable decoder filters is performed after upsampling to densifying feature maps. The upsampled feature map in FCN is then added to the corresponding resolution encoder feature map to produce the output decoder feature map. The upsampling kernels are initialized using bilinear interpolation weights (Long et al., 2015).

The FCN decoder model requires storing encoder feature maps during inference. This can be memory intensive, for e.g. storing 64 feature maps of the first layer of FCN-Basic at 180×240 resolution in 32 bit floating point precision takes 11MB. This can be made smaller using dimensionality reduction to the 11 feature maps which requires $\approx$ 1.9MB storage. SegNet on the other hand requires almost negligible storage cost for the pooling indices (0.17MB if stored using 2 bits per 2×2 pooling window). We can also create a variant of the FCN-Basic model which discards the encoder feature map addition step and only learns the upsampling kernels (**FCN-Basic-NoAddition**).

In addition to the above variants, we study upsampling using fixed bilinear interpolation weights which therefore requires no learning for upsampling (**Bilinear-Interpolation**). At the other extreme, we can add 64 encoder feature maps at each layer to the corresponding output feature maps from the SegNet decoder to create a more memory intensive variant of SegNet (**SegNet-Basic-EncoderAddition**). Another and more memory intensive FCN-Basic variant (**FCN-Basic-NoDimReduction**) is where there is no dimensionality reduction performed for the encoder feature maps. Finally, please note that to encourage reproduction of our results we release the Caffe (Jia et al., 2014) implementation of all the variants[2].

We also tried other generic variants where feature maps are simply upsampled by *replication* (Farabet et al., 2013), or by using a fixed (and sparse) array of indices for upsampling. These performed quite poorly in comparison to the above variants. A variant without max-pooling and sub-sampling in the encoder network (decoders are redundant) consumes more memory, takes longer to converge and performs poorly.

2.3.2 Training

We use the CamVid road scenes dataset (Brostow et al., 2009) to benchmark the performance of the decoder variants. This dataset is small, consisting of 367 training and 233 testing RGB images (day and dusk scenes) at 360×480 resolution. The challenge is to segment 11 classes such as road, building, cars, pedestrians, signs, poles, side-walk *etc*. We perform local contrast normalization (Jarrett et al., 2009) to the RGB input.

The encoder and decoder weights were all initialized using the technique described in He *et al.* (He et al., 2015). To train all the variants we use stochastic gradient descent (SGD) with a fixed learning rate of 0.1 and momentum of 0.9 (Bottou, 2010) using our Caffe implementation of SegNet-Basic (Jia et al., 2014). We train the variants until the training loss converges. Before each epoch, the training set is shuffled and each mini-batch (12 images) is then picked in order thus ensuring that each image is used only once in an epoch. We select the model which performs highest on a validation dataset.

We use the cross-entropy loss (Long et al., 2015) as the objective function for training the network. The loss is averaged up over all the pixels in a mini-batch which contain a valid label. When there is large variation in the number of pixels in each class in the training set (e.g road, sky and building pixels dominate the CamVid dataset) then there is a need to weight the loss differently based on the true class. This is termed *class balancing*. We use *median frequency balancing* where the weight assigned to a class in the loss function is the ratio of the median of class frequencies computed on the entire training set divided by the class frequency. Therefore, we weight each pixel's loss by

$$\alpha = medianfreq/freq(c), \tag{2.1}$$

where $freq(c)$ is the number of pixels of class c in the dataset, divided by the total number of pixels in images where c is present, and $medianfreq$ is the median of these frequencies. This implies that larger classes in the training set have a weight smaller than 1 and the weights of

[2]See http://mi.eng.cam.ac.uk/projects/segnet/ for our SegNet code and web demo.

				Median frequency balancing			Natural frequency balancing		
Variant	Params (M)	Encoder storage (MB)	Infer time (ms)	G	C	IoU	G	C	IoU
Fixed upsampling									
Bilinear-Interpolation	0.625	0	24.2	77.9	61.1	43.3	82.7	52.5	43.8
Upsampling using max-pooling indices									
SegNet-Basic	1.425	1x	52.6	82.7	62.0	47.7	84.0	54.6	46.3
SegNet-Basic-EncoderAddition	1.425	64x	53.0	83.4	**63.6**	48.5	**84.2**	56.5	**47.7**
SegNet-Basic-SingleChannelDecoder	0.625	1x	33.1	81.2	60.7	46.1	83.5	53.9	45.2
Learning to upsample (bilinear initialisation)									
FCN-Basic	0.65	11x	24.2	81.7	62.4	47.3	83.9	55.6	45.0
FCN-Basic-NoAddition	0.65	n/a	23.8	80.5	58.6	44.1	82.3	53.9	44.2
FCN-Basic-NoDimReduction	1.625	64x	44.8	**84.1**	63.4	**50.1**	83.5	**57.3**	47.0
FCN-Basic-NoAddition-NoDimReduction	1.625	0	43.9	80.5	61.6	45.9	83.7	54.8	45.5

Table 2.1 Comparison of decoder variants on the CamVid dataset. We quantify the performance using global (G), class average (C) and mean of intersection over union (IoU) metrics. The testing accuracies are shown as percentages for both natural frequency and median frequency balanced training loss function. SegNet-Basic performs at the same level as FCN-Basic but requires only storing max-pooling indices and is therefore more memory efficient during inference. Note that the theoretical memory requirement reported is based only on the size of the first layer encoder feature map. Networks with larger decoders and those using the encoder feature maps in full perform best, although they are least efficient.

the smallest classes are the highest. We also experimented with training the different variants without class balancing or equivalently using *natural frequency balancing*.

2.3.3 Analysis

To compare the quantitative performance of the different decoder variants, we use three commonly used performance measures: global accuracy (G) which measures the percentage of pixels correctly classified in the dataset, class average accuracy (C) is the mean of the predictive accuracy over all classes and mean intersection over union (IoU) over all classes as used in the Pascal VOC12 challenge (Everingham et al., 2015). The mean IoU metric is the hardest metric since it penalizes false positive predictions unlike class average accuracy. However, IoU metric is not optimized for directly through the class balanced cross-entropy loss.

We test each variant after each 1000 iterations of optimization on the CamVid validation set until the training loss converges. With a training mini-batch size of 12 this corresponds to testing approximately every 33 epochs (passes) through the training set. We select the iteration wherein the global accuracy is highest amongst the evaluations on the validation set. We report all the three measures of performance at this point on the held-out CamVid test set. Although we use class balancing while training the variants, it is still important to achieve high global accuracy to result in an overall smooth segmentation. Another reason is that the contribution of segmentation towards autonomous driving is mainly for delineating classes such as roads, buildings, side-walk, sky. These classes dominate the majority of the pixels in an image and a high global accuracy corresponds to good segmentation of these important classes. We also observed that reporting the numerical performance when class average is highest can often correspond to low global accuracy indicating a perceptually noisy segmentation output.

In Table 2.1 we report the numerical results of our analysis. We also show the size of the trainable parameters and the highest resolution feature map or pooling indices storage memory, i.e, of the first layer feature maps after max-pooling and sub-sampling. We show the average time for one forward pass with our Caffe implementation, averaged over 50 measurements using a 360×480 input on an NVIDIA Titan GPU with cuDNN v3 acceleration. We note that the upsampling layers in the SegNet variants are not optimised using cuDNN acceleration. We show the results for both testing and training for all the variants at the selected iteration. The results are also tabulated without class balancing (natural frequency) for training and testing accuracies. Below we analyse the results with class balancing.

From the Table 2.1, we see that bilinear interpolation based upsampling without any learning performs the worst based on all the three measures of accuracy. All the other methods which either use learning for upsampling (FCN-Basic and variants) or learning decoder filters after upsampling (SegNet-Basic and its variants) perform significantly better. This emphasizes the need to learn decoders for segmentation. This is also supported by experimental evidence gathered by other authors when comparing FCN with SegNet-type decoding techniques (Noh et al., 2015).

When we compare SegNet-Basic and FCN-Basic we see that both perform equally well on this test over all the three measures of accuracy. The difference is that SegNet uses less **memory** during inference since it only stores max-pooling indices. On the other hand FCN-Basic stores **encoder feature maps** in full which consumes much more memory (11 times more). SegNet-Basic has a decoder with 64 feature maps in each decoder layer. In comparison FCN-Basic, which uses dimensionality reduction, has fewer (11) feature maps in each decoder layer. This reduces the number of convolutions in the decoder network and hence FCN-Basic is faster during inference (forward pass). From another perspective, the decoder network in SegNet-Basic makes it overall a larger network than FCN-Basic. This endows it with more flexibility and hence achieves higher training accuracy than FCN-Basic for the same number of iterations. Overall we see that SegNet-Basic has an advantage over FCN-Basic when memory during inference is constrained but where inference time can be compromised to an extent.

SegNet-Basic is most similar to FCN-Basic-NoAddition in terms of their decoders, although the decoder of SegNet is larger. Both learn to produce dense feature maps, either directly by learning to perform deconvolution as in FCN-Basic-NoAddition or by first upsampling and then convolving with trained decoder filters. The performance of SegNet-Basic is superior, in part due to its larger decoder size. Now, the accuracy of FCN-Basic-NoAddition is also lower as compared to FCN-Basic. This shows that it is important to capture the information present in the encoder feature maps for better performance. This can also explain the reason why SegNet-Basic outperforms FCN-Basic-NoAddition.

The size of the FCN-Basic-NoAddition-NoDimReduction model is slightly larger than SegNet-Basic and this makes it a fair comparison. The performance of this FCN variant is poorer than SegNet-Basic in test but also its training accuracy is lower for the same number of training epochs. This shows that using a larger decoder is not enough but it is also important to capture encoder feature map information to learn better. Here it is also interesting to see that SegNet-Basic has a competitive training accuracy when compared to larger models such FCN-Basic-NoDimReduction.

Another interesting comparison between FCN-Basic-NoAddition and SegNet-Basic-SingleChannelDecoder shows that using max-pooling indices for upsampling and an overall larger decoder leads to better performance. This also lends evidence to SegNet being a good architecture for segmentation, particularly when there is a need to find a compromise between storage cost or accuracy against inference time. In the best case, when both memory and inference time is not constrained, larger models such as FCN-Basic-NoDimReduction and SegNet-EncoderAddition are both more accurate than the other variants. Particularly, discarding dimensionality reduction in the FCN-Basic model leads to the best performance amongst the FCN-Basic variants. This once again emphasizes the trade-off involved between memory and accuracy in segmentation architectures.

The last column of Table 2.1 show the result when no class balancing is used (natural frequency). Here, we can observe that without weighting the results are poorer for all the variants, particularly for class average accuracy and mean IoU metric. The global accuracy is the highest without weighting since the majority of the scene is dominated by sky, road and building pixels. Apart from this all the inference from the comparative analysis of variants holds true for natural frequency balancing too. SegNet-Basic performs as well as FCN-Basic and is better than the larger FCN-Basic-NoAddition-NoDimReduction. The bigger but less efficient models FCN-Basic-NoDimReduction and SegNet-EncoderAddition perform better than the other variants.

We can now summarize the above analysis with the following general points.

1. The best performance is achieved when encoder feature maps are stored in full.
2. When memory during inference is constrained, then compressed forms of encoder feature maps (dimensionality reduction, max-pooling indices) can be stored and used with an appropriate decoder (e.g. SegNet type) to improve performance.
3. Larger decoders increase performance for a given encoder network.

2.3.4 Benchmarking

We quantify the performance of SegNet on three different benchmarks using our Caffe implementation [3]. Through this process we demonstrate the efficacy of SegNet for various scene segmentation tasks which have practical applications. In the first experiment, we test the performance of SegNet on the CamVid road scene dataset (see Sec. 2.3.2 for more information about this data). We use this result to compare SegNet with several methods including Random Forests (Shotton et al., 2008), Boosting (Shotton et al., 2008; Sturgess et al., 2009) in combination with CRF based methods (Ladický et al., 2010). We also trained SegNet on a larger dataset of road scenes collected from various publicly available datasets

[3]Our web demo and Caffe implementation is available for evaluation at http://mi.eng.cam.ac.uk/projects/segnet/

(Brostow et al., 2009; Ros et al., 2015; Torralba et al., 2009) and show that this leads to a large improvement in accuracy.

SUN RGB-D (Song et al., 2015) is a very challenging and large dataset of indoor scenes with 5285 training and 5050 testing images. The images are captured by different sensors and hence come in various resolutions. The task is to segment 37 indoor scene classes including wall, floor, ceiling, table, chair, sofa etc. This task is made hard by the fact that object classes come in various shapes, sizes and in different poses. There are frequent partial occlusions since there are typically many different classes present in each of the test images. These factors make this one of the hardest segmentation challenges. We only use the RGB modality for our training and testing. Using the depth modality would necessitate architectural modifications/redesign (Long et al., 2015). Also the quality of depth images from current cameras require careful post-processing to fill-in missing measurements. They may also require using fusion of many frames to robustly extract features for segmentation.

Pascal VOC12 (Everingham et al., 2015) is a RGB dataset for segmentation with 12031 combined training and validation images of indoor and outdoor scenes. The task is to segment 21 classes such as bus, horse, cat, dog, boat from a varied and large background class. The foreground classes often occupy a small part of an image. The evaluation is performed remotely on 1456 images.

In all three benchmark experiments, we select random 224×224 resolution crops from the images for training. We used SGD with momentum to train SegNet. The learning rate was fixed to 0.001 and momentum to 0.9. The mini-batch size was 4. The optimization was performed for 100 epochs and then tested.

CamVid Road Scenes

A number of outdoor scene datasets are available for semantic parsing (Brostow et al., 2009; Geiger et al., 2012; Gould et al., 2009; Russell et al., 2008). From these, we choose to benchmark SegNet using the CamVid dataset (Brostow et al., 2009) because it contains video sequences. This enables us to compare our proposed architecture with those which use motion and structure (Brostow et al., 2008; Ladický et al., 2010; Sturgess et al., 2009) and video segments (Tighe and Lazebnik, 2013). We also combine (Brostow et al., 2009; Geiger et al., 2012; Gould et al., 2009; Russell et al., 2008) to form an ensemble of 3433 images to train SegNet for an additional benchmark.

The qualitative comparisons of SegNet-Basic and SegNet predictions with several prominent algorithms (unaries, unaries+CRF) are shown in Figure 2.4 along with the input modalities used to train the methods. The qualitative results show the ability of the proposed architectures to segment small (cars, pedestrians, bicyclist) classes while producing a smooth

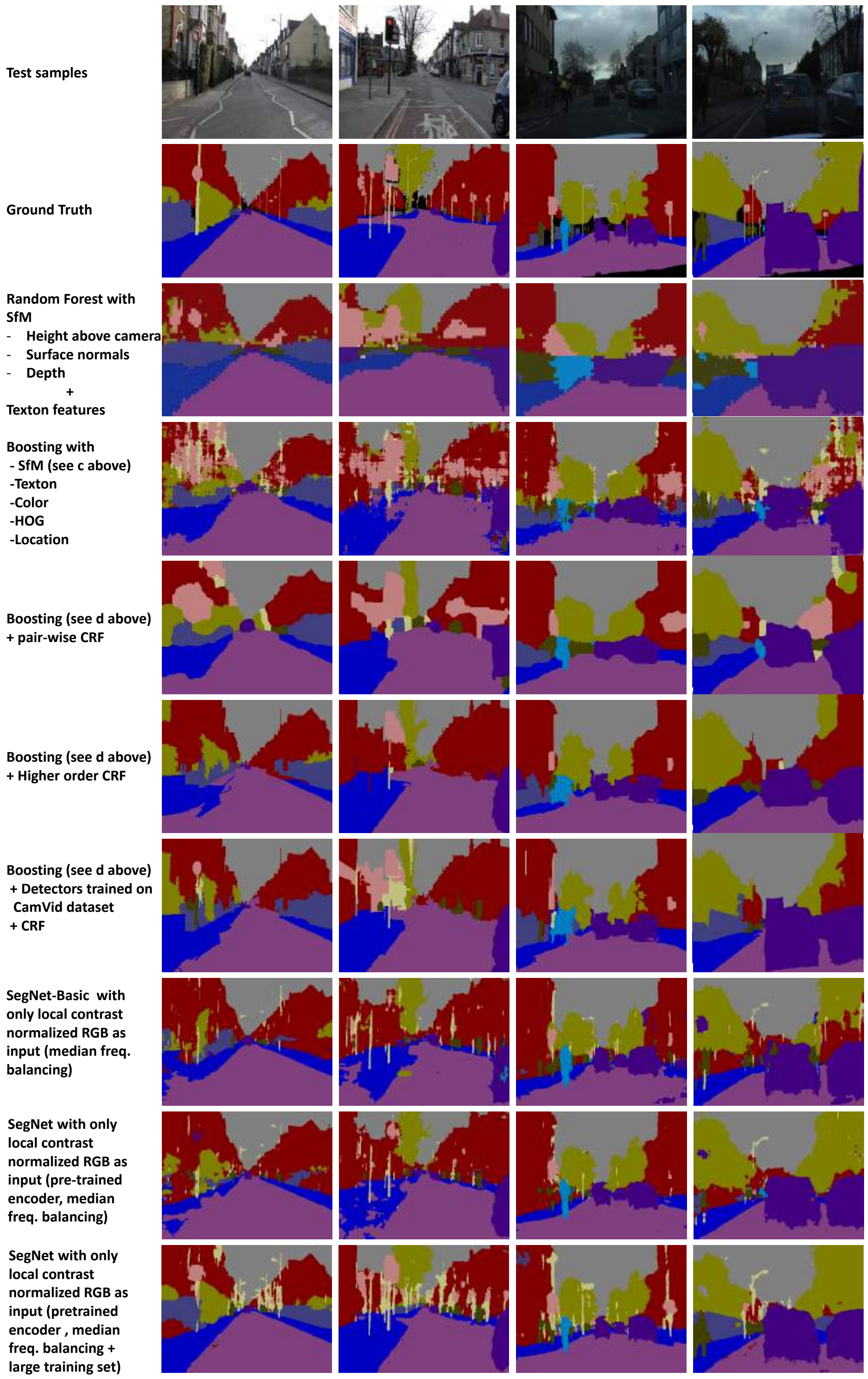

Test samples
Ground Truth
Random Forest with SfM
- Height above camera
- Surface normals
- Depth
+
Texton features
Boosting with
- SfM (see c above)
-Texton
-Color
-HOG
-Location
Boosting (see d above)
+ pair-wise CRF
Boosting (see d above)
+ Higher order CRF
Boosting (see d above)
+ Detectors trained on CamVid dataset
+ CRF
SegNet-Basic with only local contrast normalized RGB as input (median freq. balancing)
SegNet with only local contrast normalized RGB as input (pre-trained encoder, median freq. balancing)
SegNet with only local contrast normalized RGB as input (pretrained encoder , median freq. balancing + large training set)

Fig. 2.4 Results on CamVid day and dusk test samples. The evolution of results from various patch based predictions (Brostow et al., 2008; Shotton et al., 2008), then CRF smoothing models (Ladický et al., 2010; Sturgess et al., 2009) and finally SegNet.

segmentation of the overall scene. The unary-only methods (like Random Forests or Boosting) which are trained to predict the label of the centre-pixel of a small patch produce low quality segmentations. Smoothing unaries with CRF's improve the segmentation quality considerably with higher order CRF's performing best. Although the CRF based results appear smooth, upon close scrutiny, we see that shape segmentation of smaller but important classes such as bicyclists, pedestrians are poor. In addition, natural shapes of classes like trees are not preserved and the details like the wheels of cars are lost. More dense CRF models (Koltun, 2011) can be better but with additional cost of inference. SegNet-Basic and SegNet (without large dataset training) clearly indicate their ability to retain the natural shapes of classes such as bicyclists, cars, trees, poles etc better than CRF based approaches. The overall segmentation quality is also smooth except for the side-walk class. This is because the side-walk class is highly varied in terms of size and texture and this cannot be captured with a small training set. Another explanation could be that the size of the receptive fields of the deepest layer feature units are smaller than their theoretical estimates (Liu et al., 2015c; Zhou et al., 2014a) and hence unable to group all the side-walk pixels into one class. Illumination variations also affect performance on cars in the dusk examples. However, several of these issues can be ameliorated by using larger amounts of training data. In particular, we see that smaller classes such as pedestrians, cars, bicyclists, column-pole are segmented better than other methods in terms of shape retention. The side-walk class is also segmented smoothly.

The quantitative results in Table 2.2 show SegNet-Basic and SegNet obtain competitive results, even without CRF based processing. This shows the ability of the deep architecture to extract meaningful features from the input image and map it to accurate and smooth class segment labels. SegNet is better in performance than SegNet-Basic although trained with the same (small) training set. This indicates the importance of using pre-trained encoder weights and a deeper architecture. Interestingly, the use of the bigger and deeper SegNet architecture improves the accuracy of the larger classes as compared to SegNet-Basic and not the smaller classes as one might expect. We also find that SegNet-Basic (Badrinarayanan et al., 2015) trained in a layer-wise manner using L-BFGS (Nocedal and Wright, 2006) also performs competitively and is better than SegNet-Basic trained with SGD (see Sec. 2.3.2). This is an interesting training approach but needs further research in order for it scale to larger datasets.

The most interesting result is the approximately 15% performance improvement in class average accuracy that is obtained with a large training dataset, obtained by combining Gould et al. (2009), Russell et al. (2008), Brostow et al. (2009) and Geiger et al. (2012). The mean of intersection over union metric is also very high. Correspondingly, the qualitative results of SegNet (see Figure 2.4) are clearly superior to the rest of the methods. It is able to segment both small and large classes well. In addition, there is an overall smooth

Method	Building	Tree	Sky	Car	Sign-Symbol	Road	Pedestrian	Fence	Column-Pole	Side-walk	Bicyclist	Class avg.	Global avg.	Mean IoU
SfM+Appearance (Brostow et al., 2008)	46.2	61.9	89.7	68.6	42.9	89.5	53.6	46.6	0.7	60.5	22.5	53.0	69.1	n/a
Boosting (Sturgess et al., 2009)	61.9	67.3	91.1	71.1	58.5	92.9	49.5	37.6	25.8	77.8	24.7	59.8	76.4	n/a
Dense Depth Maps (Zhang et al., 2010)	85.3	57.3	95.4	69.2	46.5	**98.5**	23.8	44.3	22.0	38.1	28.7	55.4	82.1	n/a
Structured Random Forests (Kontschieder et al., 2011)	n/a											51.4	72.5	n/a
Neural Decision Forests (Bulo et al., 2014)	n/a											56.1	82.1	n/a
Local Label Descriptors (Yang et al., 2012)	80.7	61.5	88.8	16.4	n/a	98.0	1.09	0.05	4.13	12.4	0.07	36.3	73.6	n/a
Super Parsing (Tighe and Lazebnik, 2013)	87.0	67.1	96.9	62.7	30.1	95.9	14.7	17.9	1.7	70.0	19.4	51.2	83.3	n/a
SegNet-Basic	81.3	72.0	93.0	81.3	14.8	93.3	62.4	31.5	36.3	73.7	42.6	62.0	82.7	47.7
SegNet-Basic (layer-wise training (Badrinarayanan et al., 2015))	75.0	84.6	91.2	82.7	36.9	93.3	55.0	37.5	44.8	74.1	16.0	62.9	84.3	n/a
SegNet	**88.8**	87.3	92.4	82.1	20.5	97.2	57.1	49.3	27.5	84.4	30.7	65.2	**88.5**	55.6
SegNet (3.5K dataset training)	73.9	**90.6**	90.1	**86.4**	**69.8**	94.5	**86.8**	**67.9**	**74.0**	**94.7**	**52.9**	**80.1**	86.7	**60.4**
CRF based approaches														
Boosting + pairwise CRF (Sturgess et al., 2009)	70.7	70.8	94.7	74.4	55.9	94.1	45.7	37.2	13.0	79.3	23.1	59.9	79.8	n/a
Boosting+Higher order (Sturgess et al., 2009)	84.5	72.6	**97.5**	72.7	34.1	95.3	34.2	45.7	8.1	77.6	28.5	59.2	83.8	n/a
Boosting+Detectors+CRF (Ladický et al., 2010)	81.5	76.6	96.2	78.7	40.2	93.9	43.0	47.6	14.3	81.5	33.9	62.5	83.8	n/a

Table 2.2 Quantitative results on CamVid (Brostow et al., 2009) consisting of 11 road scene categories. SegNet outperforms all the other methods, including those using depth, video and/or CRF's. In comparison with the CRF based methods SegNet predictions are more accurate in 8 out of the 11 classes. It also shows a good $\approx 15\%$ improvement in class average accuracy when trained on a large dataset of 3.5K images and this sets a new benchmark for the majority of the individual classes. Particularly noteworthy are the significant improvements in accuracy for the smaller/thinner classes.

quality of segmentation much like what is typically obtained with CRF post-processing. Although the fact that results improve with larger training sets is not surprising, the percentage improvement obtained using a pre-trained encoder network and this training set indicates that this architecture can potentially be deployed for practical applications. Our random testing on urban and highway images from the internet (see Figure 2.1) demonstrates that SegNet can *absorb* a large training set and generalize well to unseen images. It also indicates the contribution of the prior (CRF) can be lessened when sufficient amount of training data is made available.

SUN RGB-D Indoor Scenes

Road scene images have limited variation, both in terms of the classes of interest and their spatial arrangements, especially when captured in a single sequence from a moving vehicle. In comparison, images of indoor scenes are more complex since the view points can vary significantly and there is less regularity in both the number of classes present in a scene and their spatial arrangement. Another difficulty is caused by the widely varying sizes of the object classes in the scene. Some test samples from the recent SUN RGB-D dataset (Song et al., 2015) are shown in Figure 2.5. We observe some scenes with few large classes and some others with dense clutter (bottom row and right). The appearance (texture and shape) can also widely vary in indoor scenes. Therefore, we believe this is the hardest challenge for segmentation architectures and methods in computer vision. Other challenges such as Pascal VOC12 (Everingham et al., 2015) salient object segmentation have occupied researchers, but indoor scene segmentation is more challenging and has more practical applications such as in robotics. Our model advances state-of-the-art on the large SUN RGB-D dataset.

The qualitative results of SegNet on some images of indoor scenes of different types such as bedroom, kitchen, bathroom, classroom etc. are shown in Figure 2.5. We see that SegNet obtains sharp boundaries between classes when the scene consists of reasonable sized classes but even when viewpoint changes are present (see bed segmentation from different view points). This is particularly interesting since the input modality is only RGB. It indicates the ability of SegNet to extract features from RGB images which are useful for view-point invariant segmentation provided there is sufficient training data (here 5285 images). RGB images are also useful to segment thinner structures such as the legs of chairs and tables, lamps which is difficult to achieve using depth images from currently available sensors. It is also useful to segment decorative objects such as paintings on the wall.

In Table 2.3 we report the quantitative results on the 37 class segmentation task. We first note here that the other methods that have been benchmarked are not based on deep architectures and they only report class average accuracy. The existing top performing

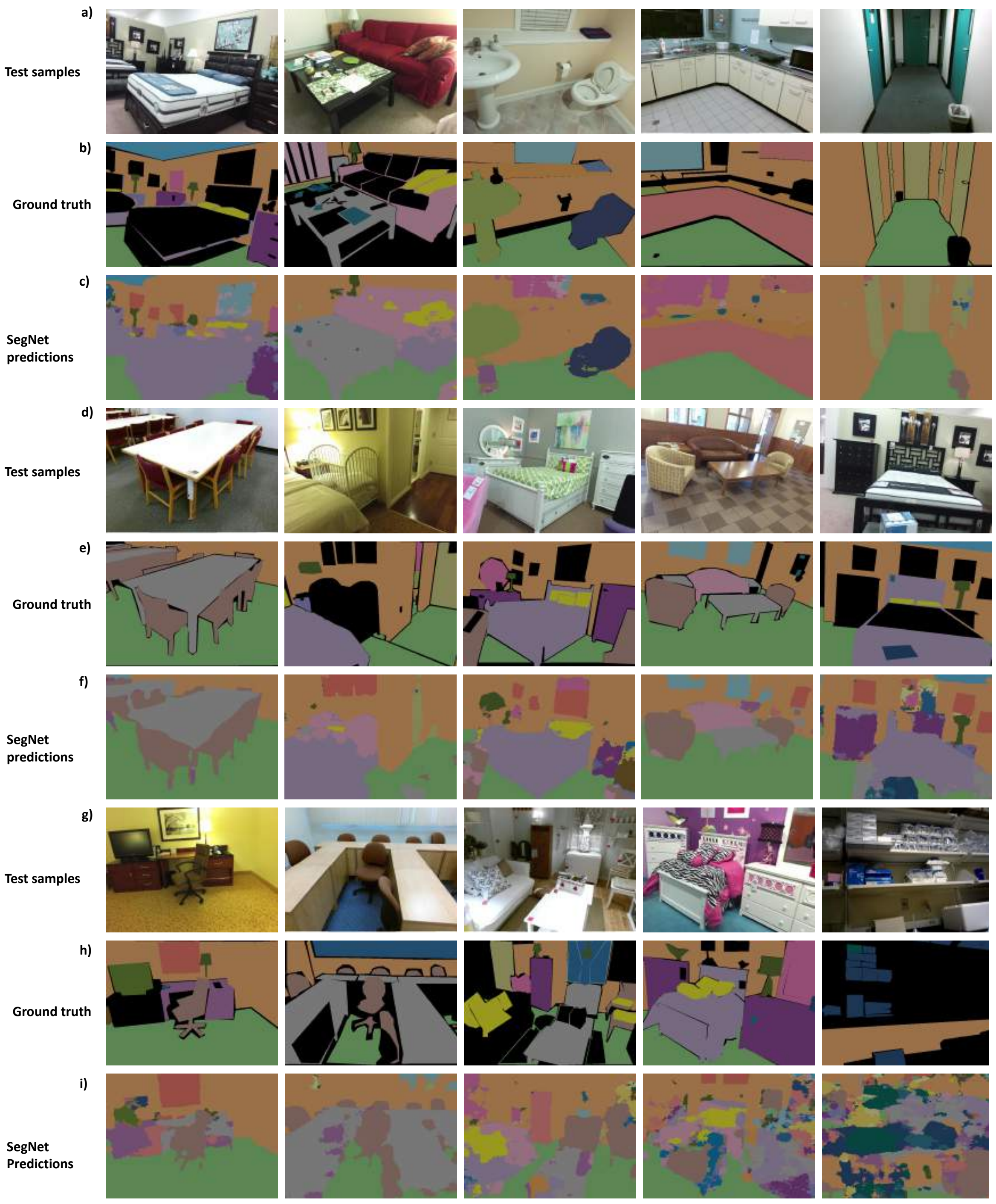


Fig. 2.5 Qualitative assessment of SegNet predictions on RGB indoor test scenes from the recently released SUN RGB-D dataset (Song et al., 2015). In this hard challenge, SegNet predictions delineate inter class boundaries well for object classes in a variety of scenes and their view-points. The segmentation quality is good when object classes are reasonably sized (rows (c,f)) but suffers when the scene is more cluttered (last two samples in row (i)). Unlabelled regions are shown as black in the ground truth.

Method	Global avg.	Class avg.	Mean IoU
RGB			
Liu *et al.* (Liu et al., 2008)	n/a	9.3	n/a
SegNet	**70.3**	35.6	**26.3**
RGB-D			
Liu *et al.* (Liu et al., 2008)	n/a	10.0	n/a
Ren et. al (Ren et al., 2012)	n/a	**36.3**	n/a

Table 2.3 Quantitative comparison on the SUN RGB-D dataset which consists of 5050 test images of indoor scenes with 37 classes. SegNet RGB-based predictions have a high global accuracy and also match the RGB-D based predictions (Ren et al., 2012) in terms of class average accuracy.

Wall	Floor	Cabinet	Bed	Chair	Sofa	Table	Door	Window	Bookshelf	Picture	Counter	Blinds
86.6	92.0	52.4	68.4	76.0	54.3	59.3	37.4	53.8	29.2	49.7	32.5	31.2
Desk	Shelves	Curtain	Dresser	Pillow	Mirror	Floor mat	Clothes	Ceiling	Books	Fridge	TV	Paper
17.8	5.3	53.2	28.8	36.5	29.6	0.0	14.4	67.7	32.4	10.2	18.3	19.2
Towel	Shower curtain	Box	Whiteboard	Person	Night stand	Toilet	Sink	Lamp	Bathtub	Bag		
11.5	0.0	8.9	38.7	4.9	22.6	55.6	52.7	27.9	29.9	8.1		

Table 2.4 Class average accuracy of SegNet predictions for the 37 indoor scene classes in the SUN RGB-D benchmark dataset.

method (Ren et al., 2012) relies on hand engineered features using colour, gradients and surface normals for describing super-pixels and then smooths super-pixel labels with a CRF. For SegNet we achieve a high global accuracy which correlates with an overall smooth segmentation. This also suggests that largest classes such as wall, floor, bed, table, sofa are segmented well in spite of view-point changes and appearance variations. However, the class average accuracy and mean IoU metric are poor, but at the same level as the hand engineered method which also includes the depth channel as input. This shows that smaller and thinner classes which have lesser training data are not segmented well. The individual class accuracies are reported in Table 2.4. From these we see that there is a clear correlation between the size and natural frequency of occurrence of classes and their individual accuracies. It is also informative to note RGB input is useful to segment widely varying (shape, texture) categories such as wall, floor, ceiling, table, chair, sofa with reasonable accuracy.

Pascal VOC12 Segmentation Challenge

The Pascal VOC12 segmentation challenge (Everingham et al., 2015) consists of segmenting a few salient object classes from a widely varying background class. It is unlike the segmentation for scene understanding benchmarks described earlier which require learning both classes and their spatial context. A number of techniques have been proposed based on this

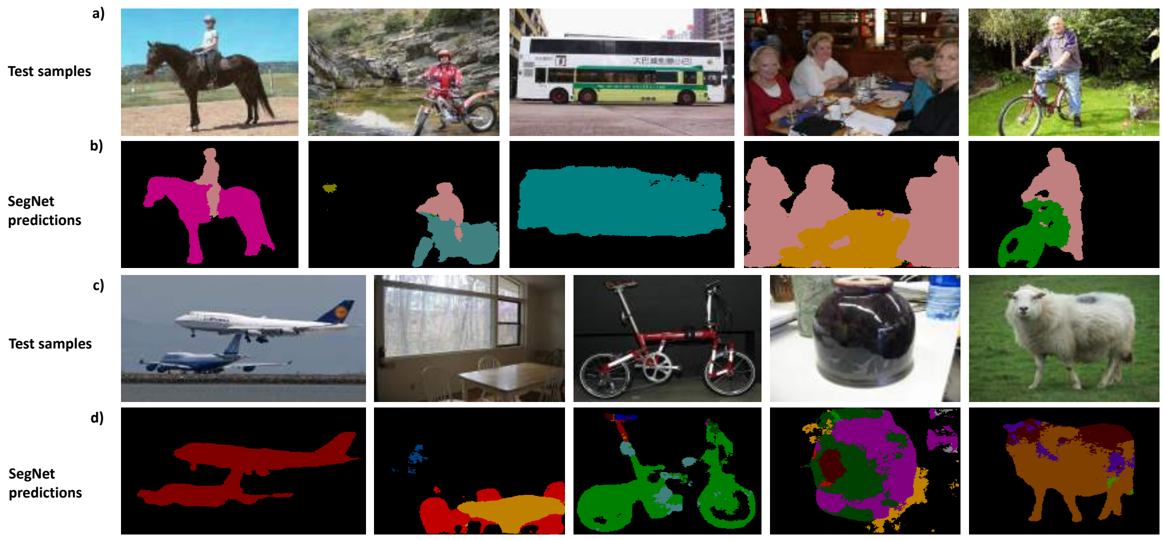


Fig. 2.6 Qualitative assessment of SegNet predictions on test samples from the Pascal VOC12 (Everingham et al., 2015) dataset. SegNet performs competitively (row (b) on several object classes of varying shape and appearance. However, it lacks smoothness particularly on large objects (see row(d)). This can be perhaps be attributed to the smaller empirical size of the receptive field of the feature units in the deepest encoder layer size (Zhou et al., 2014a).

challenge which are increasingly more accurate and complex.[4] Our efforts in this benchmarking experiment have not been diverted towards attaining the top rank by either using multi-stage training (Long et al., 2015), other datasets for pre-training such as MS-COCO (Lin et al., 2014; Zheng et al., 2015), training and inference aids such as object proposals (Noh et al., 2015; Zitnick and Dollár, 2014) or post-processing using CRF based methods (Chen et al., 2016; Noh et al., 2015). Although these supporting techniques clearly have value towards increasing the performance it unfortunately does not reveal the true performance of the deep architecture which is the *core segmentation engine*. It however does indicate that some of the large deep networks are difficult to train end-to-end on this task even with pre-trained encoder weights. Therefore, to encourage more controlled benchmarking, we trained SegNet end-to-end without other aids and report this performance.

In Table 2.5 we show the class average accuracy for some recent methods based on deep architectures. To the best of our ability, we have tried to gather the performance measures of the competing methods for their runs using minimum supporting techniques. We also specify when a method reports the performance of its core engine on the smaller validation set of 346 images (Chen et al., 2016). We find the performance on the full test set to be approximately 1% less as compared to the smaller validation set.

[4] See the leader board at http://host.robots.ox.ac.uk:8080/leaderboard

Method	Encoder size (M)	Decoder size (M)	Total size (M)	Class avg. acc.	Inference 500×500 pixels	Inference 224×224 pixels
chen2016deeplab (Chen et al., 2016) (validation set)	n/a	n/a	< 134.5	58	n/a	n/a
FCN-8 (Long et al., 2015) (multi-stage training)	134	**0.5**	134.5	62.2	210ms	n/a
Hypercolumns (Hariharan et al., 2015) (object proposals)	n/a	n/a	> 134.5	62.6	n/a	n/a
DeconvNet (Noh et al., 2015) (object proposals)	138.35	138.35	276.7	69.6	n/a	92ms ($\times$ 50)
CRF-RNN (Zheng et al., 2015) (multi-stage training)	n/a	n/a	> 134.5	**69.6**	n/a	n/a
SegNet	**14.725**	14.725	**29.45**	59.1	**94ms**	**28ms**

Table 2.5 Quantitative comparison on Pascal VOC12 dataset. The accuracies for the competing architectures are gathered for their inference run using the least number of supporting training and inference techniques. However, since they are not trained end-to-end like SegNet and use aids such as object proposals, we have added corresponding qualifying comments. The first three columns show the number of trainable parameters in the encoder, decoder and full network. Many of the models are approximately the same size as FCN. In comparison, SegNet is considerably smaller but achieves a competitive accuracy without resorting to supporting training or inference aids. This results in SegNet being significantly faster than other models in terms of inference time.

Aeroplane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Dining table
74.5	30.6	61.4	50.8	49.8	76.2	64.3	69.7	23.8	60.8	54.7
Dog	Horse	Motor bike	Person	Potted plant	Sheep	Sofa	Train	TV	Background	
62.0	66.4	70.2	74.1	37.5	63.7	40.6	67.8	53.0	88.6	

Table 2.6 Individual class accuracies of SegNet predictions on the Pascal VOC12 segmentation benchmark consisting of 21 object classes.

From the results in Table 2.5 we can see that the best performing networks are either very large (and slow) (Noh et al., 2015) and/or they use a CRF (Zheng et al., 2015). The CRF encourages large segments with a single label and this suits the Pascal challenge wherein there are one or two salient objects in the centre of the image. This prior also has a larger impact when training data is limited. This is shown in the experiments using CRF-RNN (Zheng et al., 2015) wherein the core FCN-8 model predictions are less accurate without extra training data.

It is interesting that the DeepLab (Chen et al., 2016) architecture which is simply upsampling the FCN encoder features using bilinear interpolation performs reasonably well (on the validation set). The fact that a coarse segmentation is enough to produce this performance shows that this challenge is unlike scene understanding wherein many classes of varying shape and size need to be segmented.

Methods using object proposals during training and/or inference (Hariharan et al., 2015; Noh et al., 2015) are very slow in inference time and it is hard to measure their true performance. These aids are necessitated by the very large size of their deep network (Noh et al., 2015) and also because the Pascal data can also be processed by a detect and segment approach. In comparison, SegNet is smaller by virtue of discarding the fully connected layers in the VGG16 (Simonyan and Zisserman, 2014). The authors of DeepLab (Chen et al., 2016) have also reported little loss in performance by reducing the size of the fully connected layers. The smaller size of SegNet makes end-to-end training possible for benchmarking. Although it may be argued that larger networks perform better, it is at the cost of a complex training mechanism, increased memory and inference time. This makes them unsuitable for real-time applications such as road scene understanding.

The individual class accuracies for SegNet predictions on Pascal VOC12 are shown in Table 2.6. From this we once again see larger and more frequently occurring classes such as aeroplane, bus, cat *etc.* have higher accuracy and smaller/thinner classes such as potted plant, bicycle are poorly segmented. We believe more training data (Lin et al., 2014) can help improve the performance of SegNet.

In the following section, we investigate techniques to improve our scene understanding algorithm by modelling uncertainty with deep learning.

2.4 Modelling Uncertainty

The algorithms presented so far in this chapter produce accurate semantic segmentation predictions, however they are unaware of their uncertainty. Understanding what a model does not know is a critical part of many machine learning systems (Ghahramani, 2015).

Quantifying uncertainty in computer vision applications can be largely divided into regression settings such as depth regression, and classification settings such as semantic segmentation. Existing approaches to model uncertainty in such settings in computer vision include particle filtering and conditional random fields (Blake et al., 1993; He et al., 2004). However, many modern applications mandate the use of *deep learning* to achieve state-of-the-art performance (He et al., 2016). Unfortunately, most deep learning models are not able to represent uncertainty. For example, deep learning typically does not allow for uncertainty representation in regression settings. Deep learning classification models often give normalised score vectors, which do not necessarily capture model uncertainty. For both settings uncertainty can be captured with *Bayesian deep learning* (Denker and LeCun, 1991; MacKay, 1992; Neal, 1995) approaches – which offer a practical framework for understanding uncertainty with deep learning models (Gal, 2016).

In Bayesian modelling, there are two main types of uncertainty one can model (Der Kiureghian and Ditlevsen, 2009). *Aleatoric* uncertainty captures noise inherent in the observations. This could be, for example, sensor noise or motion noise, resulting in uncertainty which cannot be reduced even if more data were to be collected. On the other hand, *epistemic* uncertainty accounts for uncertainty in the model parameters – uncertainty which captures our ignorance about which model generated our collected data. This uncertainty can be explained away given enough data, and is often referred to as *model uncertainty*. Aleatoric uncertainty can further be categorized into *homoscedastic* uncertainty, uncertainty which stays constant for different inputs, and *heteroscedastic* uncertainty. Heteroscedastic uncertainty depends on the inputs to the model, with some inputs potentially having more noisy outputs than others. Heteroscedastic uncertainty is especially important for computer vision applications. For example, for depth regression, highly textured input images with strong vanishing lines are expected to result in confident predictions, whereas an input image of a featureless wall is expected to have very high uncertainty.

In this section we make the observation that in many big data regimes (such as the ones common to deep learning with image data), it is most effective to model aleatoric uncertainty, uncertainty which cannot be explained away. This is in comparison to epistemic uncertainty which is mostly explained away with the large amounts of data often available in machine vision. We further show that modelling aleatoric uncertainty alone comes at a cost. Out-of-data examples, which can be identified with epistemic uncertainty, cannot be identified with aleatoric uncertainty alone.

For this we present a unified Bayesian deep learning framework which allows us to learn mappings from input data to aleatoric uncertainty and compose these together with epistemic uncertainty approximations. We derive our framework for both regression and classification applications and present results for per-pixel depth regression and semantic segmentation tasks (see Figure 2.7 for examples). We show how modelling aleatoric uncertainty in regression can be used to learn loss attenuation, and develop a complimentary approach for the classification case. This demonstrates the efficacy of our approach on difficult and large scale tasks.

The main contributions of this section are;

1. We capture an accurate understanding of aleatoric and epistemic uncertainties, in particular with a novel approach for classification,
2. We improve model performance by $1-3\%$ over non-Bayesian baselines by reducing the effect of noisy data with the implied attenuation obtained from explicitly representing aleatoric uncertainty,
3. We study the trade-offs between modelling aleatoric or epistemic uncertainty by characterizing the properties of each uncertainty and comparing model performance and inference time.

2.4.1 Bayesian Deep Learning

Existing approaches to Bayesian deep learning capture either epistemic uncertainty alone, or aleatoric uncertainty alone (Gal, 2016). These uncertainties are formalised as probability distributions over either the model parameters, or model outputs, respectively. Epistemic uncertainty is modelled by placing a prior distribution over a model's weights, and then trying to capture how much these weights vary given some data. Aleatoric uncertainty on the other hand is modelled by placing a distribution over the output of the model. For example, in regression our outputs might be modelled as corrupted with Gaussian random noise. In this case we are interested in learning the noise's variance as a function of different inputs (such noise can also be modelled with a constant value for all data points, but this is of less practical interest). These uncertainties, in the context of Bayesian deep learning, are explained in more detail in this section.

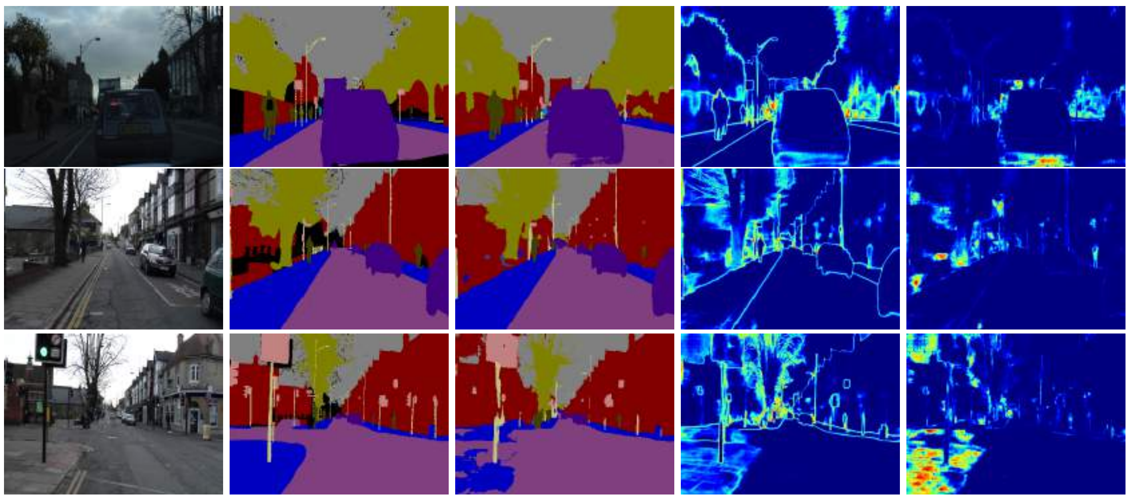

(a) Input Image
(b) Ground Truth
(c) Semantic Segmentation
(d) Aleatoric Uncertainty
(e) Epistemic Uncertainty

Fig. 2.7 **Illustrating the difference between aleatoric and epistemic uncertainty** for semantic segmentation on the CamVid dataset (Brostow et al., 2009). We observe aleatoric uncertainty captures object boundaries where labels are noisy. The bottom row shows a failure case of the segmentation model, when the model fails to segment the footpath, and the corresponding increased epistemic uncertainty.

2.4.2 Epistemic Uncertainty in Bayesian Deep Learning

To capture epistemic uncertainty in a neural network (NN) we put a prior distribution over its weights, for example a Gaussian prior distribution: $\mathbf{W} \sim \mathcal{N}(0, I)$.

Such a model is referred to as a Bayesian neural network (BNN) (Denker and LeCun, 1991; MacKay, 1992; Neal, 1995). Bayesian neural networks replace the deterministic network's weight parameters with distributions over these parameters, and instead of optimising the network weights directly we average over all possible weights (referred to as *marginalisation*). Denoting the random output of the BNN as $\mathbf{f}^{\mathbf{W}}(\mathbf{x})$, we define the model likelihood $p(\mathbf{y}|\mathbf{f}^{\mathbf{W}}(\mathbf{x}))$. Given a dataset $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$, with labels $\mathbf{Y} = \{\mathbf{y}_1, ..., \mathbf{y}_N\}$, Bayesian inference is used to compute the posterior over the weights $p(\mathbf{W}|\mathbf{X}, \mathbf{Y})$. This posterior captures the set of plausible model parameters, given the data.

For regression tasks we often define our likelihood as a Gaussian with mean given by the model output: $p(\mathbf{y}|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) = \mathcal{N}(\mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma^2)$, with an observation noise scalar σ. For classification, on the other hand, we often squash the model output through a softmax function, and sample from the resulting probability vector: $p(\mathbf{y}|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) = \text{Softmax}(\mathbf{f}^{\mathbf{W}}(\mathbf{x}))$.

BNNs are easy to formulate, but difficult to perform inference in. This is because the marginal probability $p(\mathbf{Y}|\mathbf{X})$, required to evaluate the posterior, cannot be evaluated analytically. Different approximations exist (Blundell et al., 2015; Gal and Ghahramani, 2016;

Graves, 2011; Hernández-Lobato et al., 2016). In these approximate inference techniques, the posterior $p(\mathbf{W}|\mathbf{X},\mathbf{Y})$ is fitted with a simple distribution $q^*_\theta(\mathbf{W})$, parametrised by θ. This replaces the intractable problem of averaging over all weights in the BNN with an optimisation task, where we seek to optimise over the *parameters of the simple distribution* instead of optimising the original neural network's parameters.

Dropout variational inference is a practical approach for approximate inference in large and complex models (Gal and Ghahramani, 2016). This inference is done by training a model with dropout before every weight layer, and by also performing dropout at test time to sample from the approximate posterior (stochastic forward passes, referred to as Monte Carlo dropout). More formally, this approach is equivalent to performing approximate variational inference where we find a simple distribution $q^*_\theta(\mathbf{W})$ in a tractable family which minimises the Kullback-Leibler (KL) divergence to the true model posterior $p(\mathbf{W}|\mathbf{X},\mathbf{Y})$. Dropout can be interpreted as a variational Bayesian approximation, where the approximating distribution is a mixture of two Gaussians with small variances and the mean of one of the Gaussians is fixed at zero.

Epistemic uncertainty in the weights can be reduced by observing more data. This uncertainty induces prediction uncertainty by marginalising over the (approximate) weights posterior distribution. For classification this can be approximated using Monte Carlo integration as follows:

$$p(y=c|\mathbf{x},\mathbf{X},\mathbf{Y}) \approx \frac{1}{T}\sum_{t=1}^{T} \text{Softmax}(\mathbf{f}^{\widehat{\mathbf{W}}_t}(\mathbf{x})) \tag{2.2}$$

with T sampled masked model weights $\widehat{\mathbf{W}}_t \sim q^*_\theta(\mathbf{W})$, where $q_\theta(\mathbf{W})$ is the Dropout distribution (Gal, 2016). The uncertainty of this probability vector $\mathbf{p}$ can then be summarised using the entropy of the probability vector: $H(\mathbf{p}) = -\sum_{c=1}^{C} p_c \log p_c$. For regression this epistemic uncertainty is captured by the predictive variance, which can be approximated as:

$$\text{Var}(\mathbf{y}) \approx \sigma^2 + \frac{1}{T}\sum_{t=1}^{T} \mathbf{f}^{\widehat{\mathbf{W}}_t}(\mathbf{x})^T \mathbf{f}^{\widehat{\mathbf{W}}_t}(\mathbf{x}_t) - E(\mathbf{y})^T E(\mathbf{y}) \tag{2.3}$$

with predictions in this epistemic model done by approximating the predictive mean: $E(\mathbf{y}) \approx \frac{1}{T}\sum_{t=1}^{T} \mathbf{f}^{\widehat{\mathbf{W}}_t}(\mathbf{x})$. The first term in the predictive variance, σ^2, corresponds to the amount of noise inherent in the data (heteroscedastic noise – which will be explained in more detail soon). The second part of the predictive variance measures how much the model is uncertain about its predictions – this term will vanish when we have zero parameter uncertainty (i.e. when all draws $\widehat{\mathbf{W}}_t$ take the same constant value).

2.4.3 Heteroscedastic Aleatoric Uncertainty

In the previous section we captured model uncertainty – uncertainty over the model parameters – by approximating the distribution $p(\mathbf{W}|\mathbf{X},\mathbf{Y})$. To capture aleatoric uncertainty in regression, we would have to tune the observation noise parameter σ.

Homoscedastic regression assumes constant observation noise σ for every input point $\mathbf{x}$. Heteroscedastic regression, on the other hand, assumes that observation noise can vary with input $\mathbf{x}$ (Le et al., 2005; Nix and Weigend, 1994). Heteroscedastic models are useful in cases where parts of the observation space might have higher noise levels than others. In non-Bayesian neural networks, this observation noise parameter is often fixed as part of the model's weight decay, and ignored. However, when made data-dependent, it can be learned as a function of the data.

We derive loss functions for both Gaussian and Laplacian priors. The probability density function for the Gaussian distribution is given by:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2 \big/ 2\sigma^2}. \tag{2.4}$$

We wish our neural network to learn to estimate the distribution function. To construct the optimisation objective we take the negative log likelihood of this distribution,

$$-\log p(\mathbf{y}_i|\mathbf{x}_i) \propto \frac{1}{2\sigma^2}||\mathbf{y}_i - \mathbf{x}_i||^2 + \frac{1}{2}\log\sigma^2. \tag{2.5}$$

Similarly for the Laplacian distribution,

$$-\log p(\mathbf{y}_i|\mathbf{x}_i) \propto \frac{\sqrt{2}}{\sigma}||\mathbf{y}_i - \mathbf{x}_i|| + \log\sigma. \tag{2.6}$$

We can see that using a Gaussian prior results in an L2 norm and a Laplacian prior results in an L1 norm to form the residuals. This choice can be informed by the task, often in vision models it is better to use a L1 norm as it is more robust to large, outlying residuals. However, for now we consider the loss with a Gaussian distribution. We can formulate a loss to train our deep learning models (with a Gaussian distribution) by:

$$\mathscr{L}_{\text{NN}}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2\sigma(\mathbf{x}_i)^2}||\mathbf{y}_i - \mathbf{f}(\mathbf{x}_i)||^2 + \frac{1}{2}\log\sigma(\mathbf{x}_i)^2 \tag{2.7}$$

with added weight decay parametrized by λ (and similarly for l_1 loss). Note that here, unlike the above, variational inference is *not* performed over the weights, but instead we perform maximum a posteriori (MAP) inference – finding a single value for the model parameters θ.

This approach *does not* capture epistemic model uncertainty, as epistemic uncertainty is a property of the model and not of the data.

In the next section we will combine these two types of uncertainties together in a single model. We will see how heteroscedastic noise can be interpreted as model attenuation, and develop a complimentary approach for the classification case.

2.4.4 Combining Aleatoric and Epistemic Uncertainty in One Model

In the previous section we described existing Bayesian deep learning techniques. In this section we present novel contributions which extend this existing literature. We develop models that will allow us to study the effects of modelling either aleatoric uncertainty alone, epistemic uncertainty alone, or modelling both uncertainties together in a single model. This is followed by an observation that aleatoric uncertainty in regression tasks can be interpreted as learned loss attenuation – making the loss more robust to noisy data. We follow that by extending the ideas of heteroscedastic regression to classification tasks. This allows us to learn loss attenuation for classification tasks as well.

We wish to capture both epistemic and aleatoric uncertainty in a vision model. For this we turn the heteroscedastic neural network in Section 2.4.3 into a Bayesian neural network by placing a distribution over its weights, with our construction in this section developed specifically for the case of vision models[5].

We need to infer the posterior distribution for a BNN model $\mathbf{f}$ mapping an input image, $\mathbf{x}$, to a unary output, $\hat{\mathbf{y}} \in \mathbb{R}$, and a measure of aleatoric uncertainty given by variance, σ^2. We approximate the posterior over the BNN with a dropout variational distribution using the tools of Section 2.4.2. As before, we draw model weights from the approximate posterior $\widehat{\mathbf{W}} \sim q(\mathbf{W})$ to obtain a model output, this time composed of both predictive mean as well as predictive variance:

$$[\hat{\mathbf{y}}, \hat{\sigma}^2] = \mathbf{f}^{\widehat{\mathbf{W}}}(\mathbf{x}) \tag{2.8}$$

where $\mathbf{f}$ is a Bayesian convolutional neural network parametrised by model weights $\widehat{\mathbf{W}}$. We can use a single network to transform the input $\mathbf{x}$, with its head split to predict both $\hat{\mathbf{y}}$ as well as $\hat{\sigma}^2$.

We fix a Gaussian likelihood to model our aleatoric uncertainty. This induces a minimisation objective given labelled output points x:

$$\mathscr{L}_{BNN}(\theta) = \frac{1}{D}\sum_i \frac{1}{2}\hat{\sigma}_i^{-2}||\mathbf{y}_i - \hat{\mathbf{y}}_i||^2 + \frac{1}{2}\log\hat{\sigma}_i^2 \tag{2.9}$$

[5]Although this construction can be generalised for any heteroscedastic neural network architecture.

where D is the number of output pixels $\mathbf{y}_i$ corresponding to input image $\mathbf{x}$, indexed by i (additionally, the loss includes weight decay which is omitted for brevity). For example, we may set $D = 1$ for image-level regression tasks, or D equal to the number of pixels for dense prediction tasks (predicting a unary corresponding to each input image pixel). $\hat{\sigma}_i^2$ is the BNN output for the predicted variance for pixel i.

This loss consists of two components; the residual regression obtained with a stochastic sample through the model – making use of the uncertainty over the parameters – and an uncertainty regularization term. We do not need 'uncertainty labels' to learn uncertainty. Rather, we only need to supervise the learning of the regression task. We learn the variance, σ^2, implicitly from the loss function. The second regularization term prevents the network from predicting infinite uncertainty (and therefore zero loss) for all data points.

In practice, we train the network to predict the log variance, $s_i := \log \hat{\sigma}_i^2$:

$$\mathscr{L}_{BNN}(\theta) = \frac{1}{D}\sum_i \frac{1}{2}\exp(-s_i)||\mathbf{y}_i - \hat{\mathbf{y}}_i||^2 + \frac{1}{2}s_i. \tag{2.10}$$

This is because it is more stable than regressing the variance, σ^2, as the loss avoids a potential division by zero. The exponential mapping also allows us to regress unconstrained scalar values, where $\exp(-s_i)$ is resolved to the positive domain giving valid values for variance.

To summarize, the predictive uncertainty for pixel $\mathbf{y}$ in this combined model can be approximated using:

$$\text{Var}(\mathbf{y}) \approx \frac{1}{T}\sum_{t=1}^{T} \hat{\mathbf{y}}_t^2 - \left(\frac{1}{T}\sum_{t=1}^{T} \hat{\mathbf{y}}_t\right)^2 + \frac{1}{T}\sum_{t=1}^{T} \hat{\sigma}_t^2 \tag{2.11}$$

with $\{\hat{\mathbf{y}}_t, \hat{\sigma}_t^2\}_{t=1}^{T}$ a set of T sampled outputs: $\hat{\mathbf{y}}_t, \hat{\sigma}_t^2 = \mathbf{f}^{\widehat{\mathbf{W}}_t}(\mathbf{x})$ for randomly masked weights $\widehat{\mathbf{W}}_t \sim q(\mathbf{W})$.

2.4.5 Heteroscedastic Uncertainty as Learned Loss Attenuation

We observe that allowing the network to predict uncertainty, allows it effectively to temper the residual loss by $\exp(-s_i)$, which depends on the data. This acts similarly to an intelligent robust regression function. It allows the network to adapt the residual's weighting, and even allows the network to learn to attenuate the effect from erroneous labels. This makes the model more robust to noisy data: inputs for which the model learned to predict high uncertainty will have a smaller effect on the loss.

The model is discouraged from predicting high uncertainty for all points – in effect ignoring the data – through the $\log \sigma^2$ term. Large uncertainty increases the contribution of this term, and in turn penalizes the model: The model *can* learn to ignore the data – but is penalised for that. The model is also discouraged from predicting very low uncertainty for points with high residual error, as low σ^2 will exaggerate the contribution of the residual and will penalize the model. It is important to stress that this learned attenuation is not an ad-hoc construction, but a consequence of the probabilistic interpretation of the model.

This learned loss attenuation property of heteroscedastic neural networks in regression is a desirable effect for classification models as well. However, heteroscedastic neural networks in classification are peculiar models because technically any classification task has input-dependent uncertainty. Nevertheless, the ideas above can be extended from regression heteroscedastic networks to classification heteroscedastic networks, which we discuss in the next section.

2.4.6 Heteroscedastic Uncertainty in Classification Tasks

We extend the results above to classification models, allowing us to get the equivalent of the learned loss attenuation property in classification as well. For this we adapt the standard classification model to marginalise over intermediate heteroscedastic regression uncertainty placed over the *logit space*. We therefore explicitly refer to our proposed model adaptation as a *heteroscedastic* classification neural network.

For classification tasks our model predicts a vector of unaries $\mathbf{f}_i$ for each pixel i, which when passed through a softmax operation, forms a probability vector $\mathbf{p}_i$. We change the model by placing a Gaussian distribution over the unaries vector:

$$\begin{aligned} \hat{\mathbf{x}}_i|\mathbf{W} &\sim \mathcal{N}(\mathbf{f}_i^{\mathbf{W}}, (\sigma_i^{\mathbf{W}})^2) \\ \hat{\mathbf{p}}_i &= \text{Softmax}(\hat{\mathbf{x}}_i). \end{aligned} \tag{2.12}$$

Here $\mathbf{f}_i^{\mathbf{W}}, \sigma_i^{\mathbf{W}}$ are the network outputs with parameters $\mathbf{W}$. This vector $\mathbf{f}_i^{\mathbf{W}}$ is corrupted with Gaussian noise with variance $(\sigma_i^{\mathbf{W}})^2$ (a diagonal matrix with one element for each logit value), and the corrupted vector is then squashed with the softmax function to obtain $\mathbf{p}_i$, the probability vector for pixel i.

Our expected log likelihood for this model is given by:

$$E_{\mathcal{N}(\hat{\mathbf{x}}_i;\mathbf{f}_i^{\mathbf{W}},(\sigma_i^{\mathbf{W}})^2)}[\log \hat{\mathbf{p}}_{i,c}] \tag{2.13}$$

with c the observed class for input i, which gives us our loss function. Ideally, we would want to analytically integrate out this Gaussian distribution, but no analytic solution is known. We therefore approximate the objective through Monte Carlo integration, and sample unaries through the softmax function. We note that this operation is extremely fast because we perform the computation once (passing inputs through the model to get logits). We only need to sample from the logits, which is a fraction of the network's compute, and therefore does not significantly increase the model's test time. We can rewrite the above and obtain the following numerically-stable stochastic loss:

$$\begin{aligned} \hat{\mathbf{x}}_{i,t} &= \mathbf{f}_i^{\mathbf{W}} + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, (\sigma_i^{\mathbf{W}})^2) \\ \mathcal{L}_x &= \frac{1}{T}\sum_{i,t}(-\hat{x}_{i,t,c} + \log\sum_{c'}\exp\hat{x}_{i,t,c'}) \end{aligned} \tag{2.14}$$

with $x_{i,t,c'}$ the c' element in the logit vector $\mathbf{x}_{i,t}$.

This objective can be interpreted as learning loss attenuation, similarly to the regression case. To understand this objective, we concentrate on a single pixel i and reformulate the objective as $\sum_t \log\sum_{c'}\exp(\hat{x}_{t,c'} - \hat{x}_{t,c})$ with c the observed class and t Gaussian samples. We shall analyse what this objective behaves like for various settings. When the model gives the observed class a high logit value f_c (compared to the logit values of other classes) and a low noise value σ_c, this loss will be near zero – the ideal case. When the model attempts to give the observed class a low logit value (for example if the label is noisy and $f_{c'}$ is the highest logit for some incorrect $c' \neq c$), there are two cases of interest. If the observed class logit has a low noise value, then the loss will be penalised by approximately $f_{c'} - f_c$[6]. However, if the model increases the noise value for this last case, then some noise samples will take a high value and the penalisation will be decreased from this last quantity. Lastly, the model is discouraged from increasing the noise when the observed class is given a high logit value. This is because large noise would lead some logit samples to take high negative values, and these samples will increase the loss.

We next assess the above ideas empirically.

2.4.7 Experiments

In this section we evaluate our methods with pixel-wise depth regression and semantic segmentation. An analysis of these results is given in the following section. To show the robustness of our learned loss attenuation – a side-effect of modelling uncertainty – we present results on an array of popular datasets, CamVid (Brostow et al., 2009), Make3D (Saxena et al., 2009), and NYUv2 Depth (Silberman et al., 2012), where we advance state-of-the-art.

[6]To see this, pull the term $f_{c'} - f_c$ out of the log-sum-exp; the corresponding exponent will now be $\exp(0) = 1$, and since this was the largest exponent, the remaining exp terms in the sum will be near zero.

For the following experiments we use the DenseNet architecture (Huang et al., 2016) which has been adapted for dense prediction tasks by (Jégou et al., 2016). We use our own independent implementation of the architecture using TensorFlow (Abadi et al., 2016) (which slightly outperforms the original authors' implementation on CamVid by 0.2%, see Table 2.7a). For all experiments we train with 224×224 crops of batch size 4, and then fine-tune on full-size images with a batch size of 1. We train with RMS-Prop with a constant learning rate of 0.001 and weight decay 10^{-4}.

We model the benefit of combining both epistemic uncertainty as well as aleatoric uncertainty using our developments presented in Section 2.4.4.

Semantic Segmentation

To demonstrate our method for semantic segmentation, we use two datasets, CamVid (Brostow et al., 2009) and NYU v2 (Silberman et al., 2012). CamVid is a road scene understanding dataset and has been introduced in Section 2.3.4. In Table 2.7a we present results for our architecture. Our method sets a new state-of-the-art on this dataset with mean intersection over union (IoU) score of 67.5%. We observe that modelling both aleatoric and epistemic uncertainty improves over the baseline result. The implicit attenuation obtained from the aleatoric loss provides a larger improvement than the epistemic uncertainty model. However, the combination of both uncertainties improves performance even further. This shows that for this application it is more important to model aleatoric uncertainty, suggesting that epistemic uncertainty can be mostly explained away in this large data setting.

Secondly, NYUv2 (Silberman et al., 2012) is a challenging indoor segmentation dataset with 40 different semantic classes (similar to SUN RGB-D). It has 1449 images with resolution 640×480 from 464 different indoor scenes. Table 2.7b shows our results. This dataset is much harder than CamVid because there is significantly less structure in indoor scenes compared to street scenes, and because of the increased number of semantic classes. We use DeepLabLargeFOV (Chen et al., 2016) as our baseline model. We improve baseline performance by giving the model flexibility to estimate uncertainty and attenuate the loss. The effect is more pronounced, perhaps because the dataset is more difficult. Additional qualitative results are given in Figure 2.8, Figure 2.9 and Figure 2.10.

Pixel-wise Depth Regression

We demonstrate the efficacy of our method for regression using two popular monocular depth regression datasets, Make3D (Saxena et al., 2009) and NYUv2 Depth (Silberman et al., 2012). The Make3D dataset consists of 400 training and 134 testing images, gathered

CamVid	IoU
SegNet (Badrinarayanan et al., 2017)	46.4
FCN-8 (Long et al., 2015)	57.0
chen2016deeplab-LFOV (Chen et al., 2016)	61.6
Bayesian SegNet (Kendall et al., 2017a)	63.1
Dilation8 (Yu and Koltun, 2016)	65.3
Dilation8 + FSO (Kundu et al., 2016)	66.1
DenseNet (Jégou et al., 2016)	66.9
This work:	
DenseNet (Our Implementation)	67.1
+ Aleatoric Uncertainty	67.4
+ Epistemic Uncertainty	67.2
+ Aleatoric & Epistemic	**67.5**

(a) CamVid dataset for road scene segmentation.

NYUv2 40-class	Accuracy	IoU
Gupta et al. (2014)	60.3	28.6
SegNet (Badrinarayanan et al., 2017)	66.1	23.6
FCN-8 (Long et al., 2015)	61.8	31.6
Bayesian SegNet (Kendall et al., 2017a)	68.0	32.4
Eigen and Fergus (2015)	65.6	34.1
This work:		
chen2016deeplabLargeFOV	70.1	36.5
+ Aleatoric Uncertainty	70.4	37.1
+ Epistemic Uncertainty	70.2	36.7
+ Aleatoric & Epistemic	**70.6**	**37.3**

(b) NYUv2 40-class dataset for indoor scenes.

Table 2.7 **Semantic segmentation performance.** Modelling both aleatoric and epistemic uncertainty gives a notable improvement in segmentation accuracy over state of the art baselines.

using a 3-D laser scanner. We evaluate our method using the same standard as (Laina et al., 2016), resizing images to 345×460 pixels and evaluating on pixels with depth less than $70m$. NYUv2 Depth is taken from the same dataset used for classification above. It contains RGB-D imagery from 464 different indoor scenes. We compare to previous approaches for Make3D in Table 2.8a and NYUv2 Depth in Table 2.8b, using standard metrics (for a description of these metrics please see (Eigen et al., 2014)).

These results show that aleatoric uncertainty is able to capture many aspects of this task which are inherently difficult. For example, in the qualitative results in Figure 2.11 and 2.12 we observe that aleatoric uncertainty is greater for large depths, reflective surfaces and occlusion boundaries in the image. These are common failure modes of monocular depth algorithms (Laina et al., 2016). On the other hand, these qualitative results show that epistemic uncertainty captures difficulties due to lack of data. For example, we observe larger uncertainty for objects which are rare in the training set such as humans in the third example of Figure 2.11.

In summary, we have demonstrated that our model can improve performance over non-Bayesian baselines by implicitly learning attenuation of systematic noise and difficult concepts. For example we observe high aleatoric uncertainty for distant objects and on object and occlusion boundaries.

Make3D	rel	rms	$\log_{10}$
Karsch et al. (2012)	0.355	9.20	0.127
Liu et al. (2014)	0.335	9.49	0.137
(Liu et al., 2015a)	0.314	8.60	0.119
Li et al. (2015)	0.278	7.19	0.092
Laina et al. (2016)	0.176	4.46	0.072
This work:			
DenseNet Baseline	0.167	3.92	0.064
+ Aleatoric Uncertainty	**0.149**	3.93	**0.061**
+ Epistemic Uncertainty	0.162	**3.87**	0.064
+ Aleatoric & Epistemic	**0.149**	4.08	0.063

(a) Make3D depth dataset (Saxena et al., 2009).

NYU v2 Depth	rel	rms	$\log_{10}$	δ_1	δ_2	δ_3
Karsch et al. (2012)	0.374	1.12	0.134	-	-	-
Ladicky et al. (2014)	-	-	-	54.2%	82.9%	91.4%
Liu et al. (2014)	0.335	1.06	0.127	-	-	-
Li et al. (2015)	0.232	0.821	0.094	62.1%	88.6%	96.8%
Liu et al. (2015a)	0.230	0.824	0.095	61.4%	88.3%	97.1%
Eigen et al. (2014)	0.215	0.907	-	61.1%	88.7%	97.1%
Eigen and Fergus (2015)	0.158	0.641	-	76.9%	95.0%	98.8%
Laina et al. (2016)	0.127	0.573	0.055	81.1%	95.3%	98.8%
This work:						
DenseNet Baseline	0.117	0.517	0.051	80.2%	95.1%	98.8%
+ Aleatoric Uncertainty	0.112	0.508	0.046	81.6%	95.8%	98.8%
+ Epistemic Uncertainty	0.114	0.512	0.049	81.1%	95.4%	98.8%
+ Aleatoric & Epistemic	**0.110**	**0.506**	**0.045**	**81.7%**	**95.9%**	**98.9%**

(b) NYUv2 depth dataset (Silberman et al., 2012).

Table 2.8 **Monocular depth regression performance.** Comparison to previous approaches on depth regression dataset NYUv2 Depth. Modelling the combination of uncertainties improves accuracy.

2.4.8 What Do Aleatoric and Epistemic Uncertainties Capture?

In Section 2.4.7 we showed that modelling aleatoric and epistemic uncertainties improves prediction performance, with the combination performing even better. In this section we wish to study the effectiveness of modelling aleatoric and epistemic uncertainty. In particular, we wish to quantify the performance of these uncertainty measurements and analyse what they capture.

Qualitative observations

Figure 2.8 shows segmentations and model uncertainty results from Bayesian SegNet (SegNet evaluated with Monte Carlo dropout (Kendall et al., 2017a)) on CamVid Road Scenes (Brostow et al., 2009). Figure 2.9 shows SUN RGB-D Indoor Scene Understanding (Song et al., 2015) results and Figure 2.10 has Pascal VOC (Everingham et al., 2015) results. These figures show the qualitative performance of Bayesian SegNet. We observe that segmentation predictions are smooth, with a sharp segmentation around object boundaries. Also, when the model predicts an incorrect label, the model uncertainty is generally very high. More generally, we observe that a high model uncertainty is predominantly caused by three situations.

Firstly, at class boundaries the model often displays a high level of uncertainty. This reflects the ambiguity surrounding the definition of defining where these labels transition. The Pascal results clearly illustrate this in Figure 2.10.

Secondly, objects which are visually difficult to identify often appear uncertain to the model. This is often the case when objects are occluded or at a distance from the camera.

The third situation causing model uncertainty is when the object appears visually ambiguous to the model. As an example, cyclists in the CamVid results (Figure 2.8) are visually similar to pedestrians, and the model often displays uncertainty around them. We observe similar results with visually similar classes in SUN (Figure 2.9) such as chair and sofa, or bench and table. In Pascal this is often observed between cat and dog, or train and bus classes.

Quality of Uncertainty Metric

To understand what causes the model to be uncertain, we have plotted the relationship between uncertainty and accuracy in Figure 2.13a and between uncertainty and the frequency of each class in the dataset in Figure 2.13b. Uncertainty is calculated as the mean uncertainty value for each pixel of that class in a test dataset. We observe an inverse relationship between uncertainty and class accuracy or class frequency. This shows that the model is

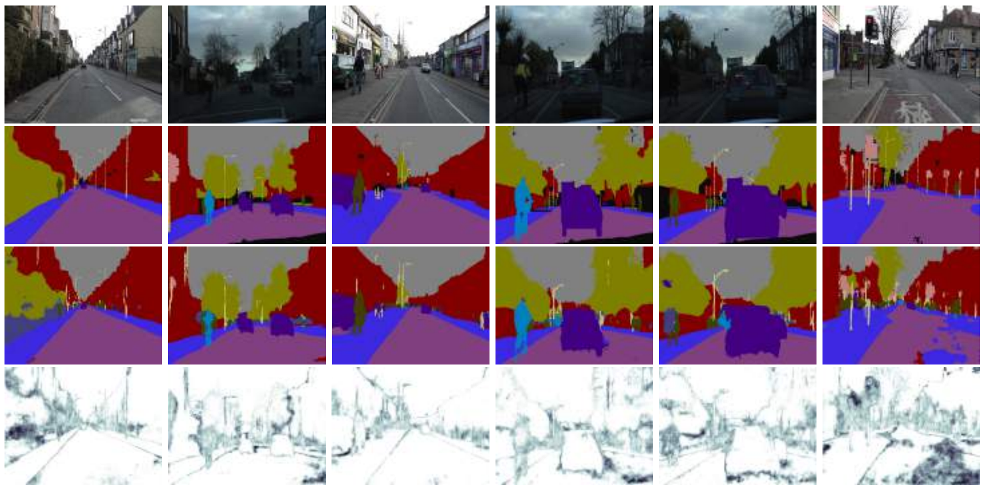


Fig. 2.8 **Bayesian SegNet results on CamVid dataset.** From top: input image, ground truth, Bayesian SegNet's segmentation prediction, and overall model uncertainty averaged across all classes.

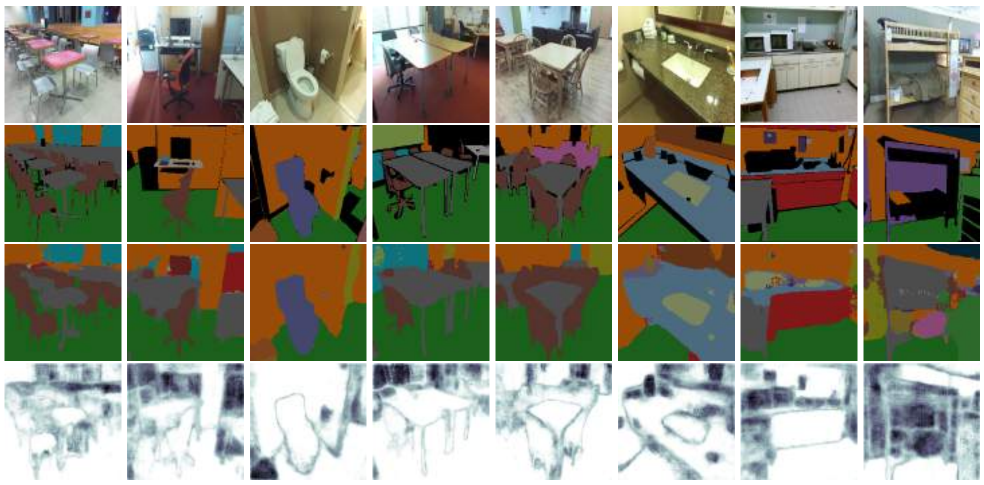


Fig. 2.9 **Bayesian SegNet results on the SUN RGB-D dataset.** From top: input image, ground truth, Bayesian SegNet's segmentation prediction, and overall model uncertainty averaged across all classes.

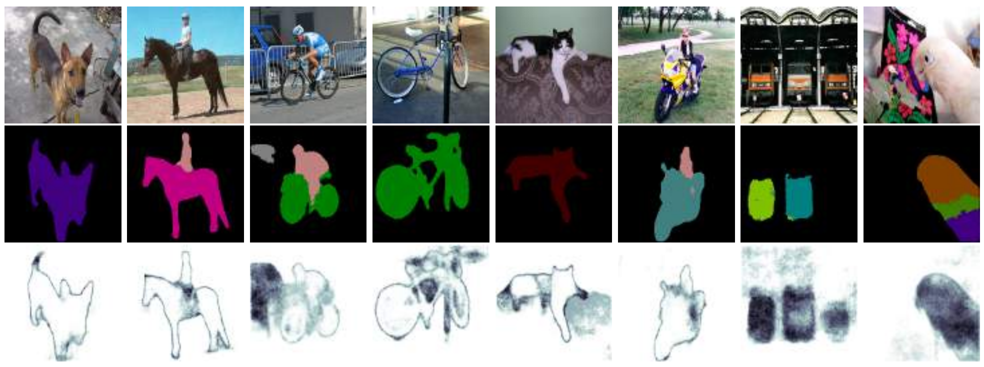


Fig. 2.10 **Pascal VOC 2012 dataset.** From top: input image, Bayesian SegNet's segmentation prediction, and overall model uncertainty averaged across all classes. Ground truth segmentation not available for Pascal test images.

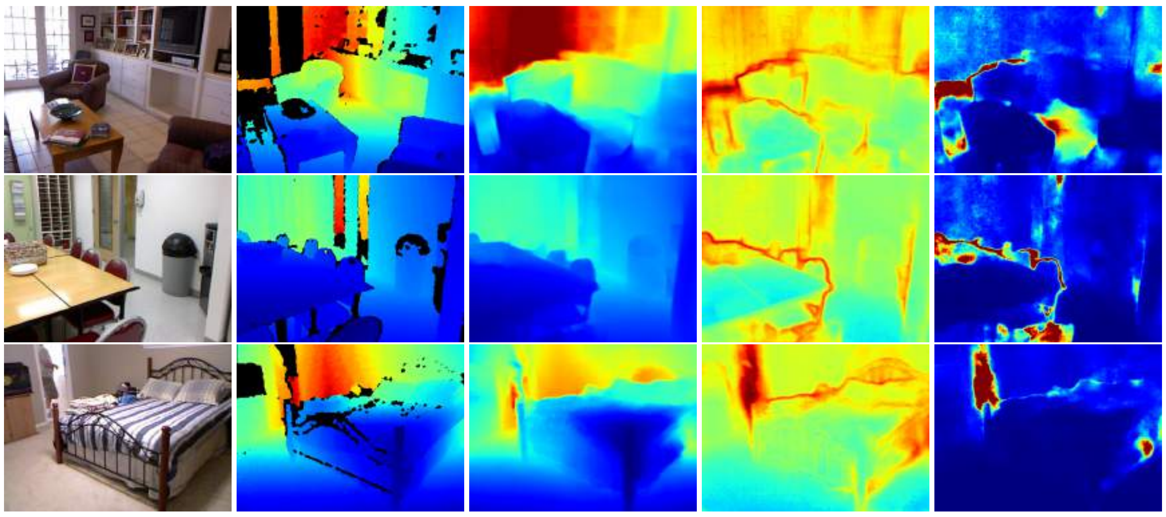


Fig. 2.11 NYUv2 Depth results. From left: input image, ground truth, depth regression, aleatoric uncertainty, and epistemic uncertainty.

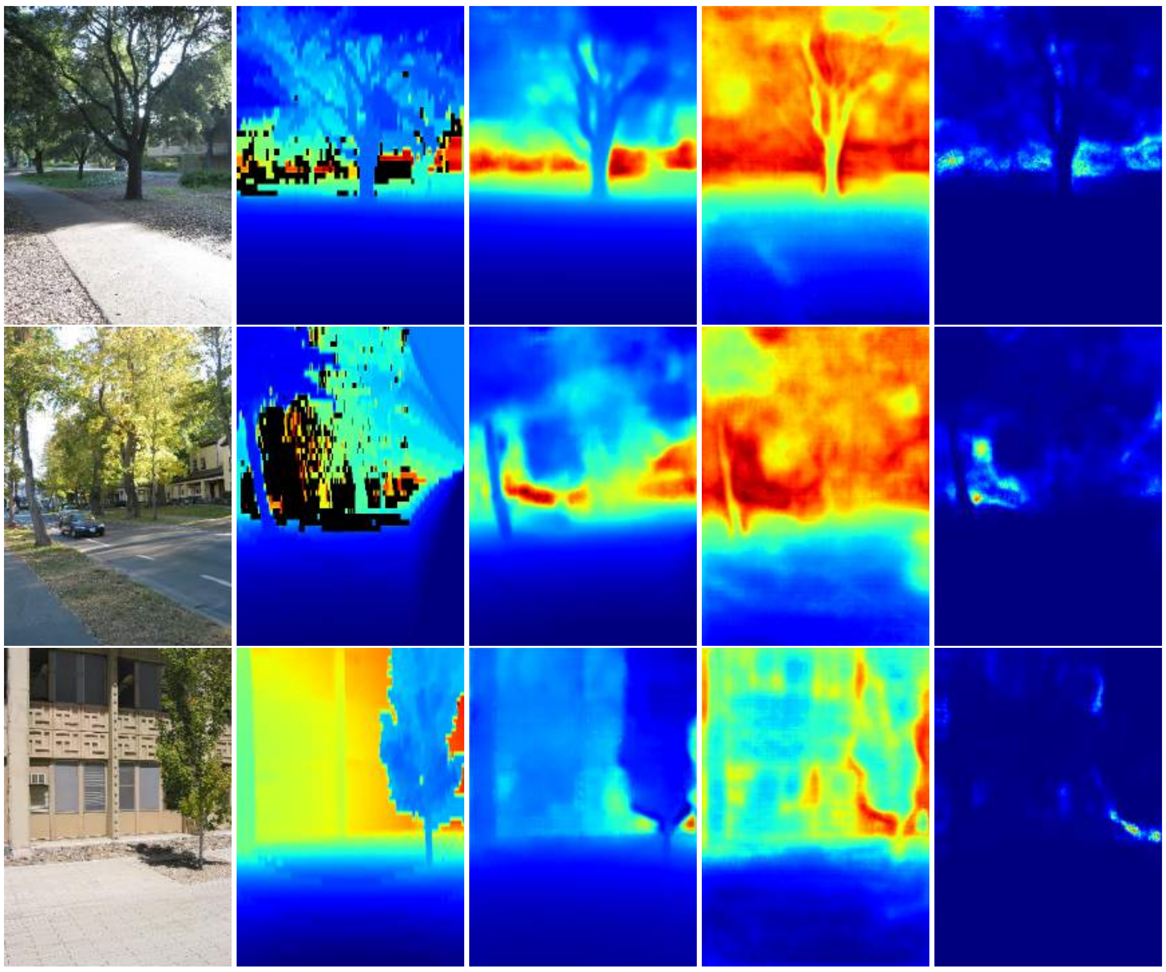
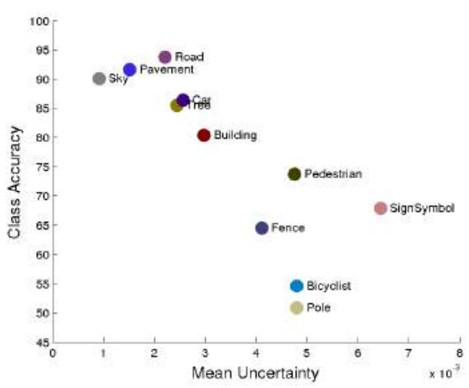
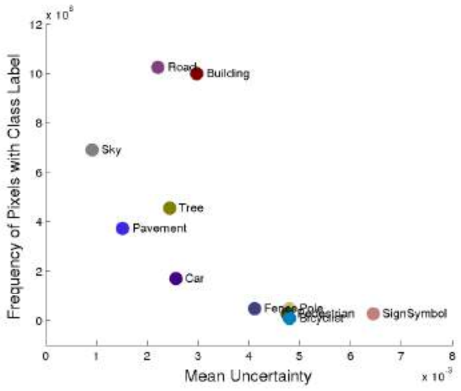


Fig. 2.12 Qualitative results on the Make3D depth regression dataset. Left to right: input image, ground truth, depth prediction, aleatoric uncertainty, epistemic uncertainty.



(a) Performance vs. mean model uncertainty



(b) Class frequency vs. mean model uncertainty

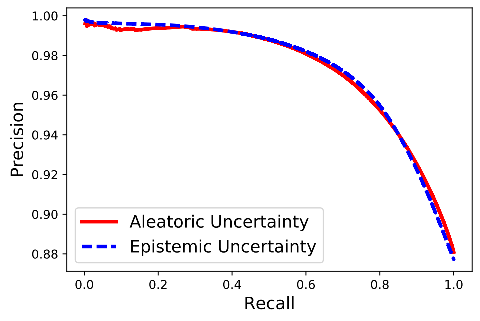
Fig. 2.13 **Bayesian SegNet performance and frequency compared to mean model uncertainty** for each class in CamVid road scene understanding dataset. These figures show a strong inverse relationships. We observe in (a) that the model is more confident with accurate classes. (b) shows classes that Bayesian SegNet is confident at are more prevalent in the dataset. Conversely, for the rare classes such as Sign Symbol and Bicyclist, Bayesian SegNet has a much higher model uncertainty.

more confident about classes which are easier or occur more often, and less certain about rare and challenging classes.

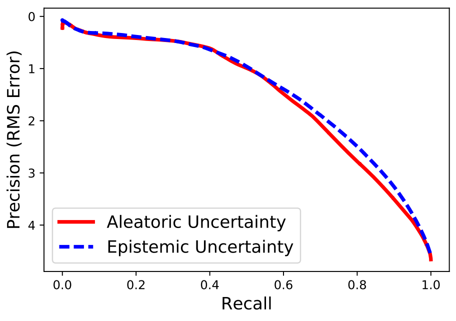
In Section 2.4.7 we showed that modelling aleatoric and epistemic uncertainty improves prediction performance, with the combination performing even better. In this section we show similar trade-offs for the performance of the uncertainty estimate. We compare aleatoric and epistemic uncertainty quantitatively and again show that aleatoric is more effective in big-data settings, however the combination performs best.

Firstly, in Figure 2.14 we show precision-recall curves for regression and classification models. They show how our model performance improves by removing pixels with uncertainty larger than various percentile thresholds. This illustrates two behaviours of aleatoric and epistemic uncertainty measures. Firstly, it shows that the uncertainty measurements are able to correlate well with accuracy, because all curves are strictly decreasing functions. We observe that precision is lower when we have more points that the model is not certain about. Secondly, the curves for epistemic and aleatoric uncertainty models are very similar. This shows that each uncertainty ranks pixel confidence similarly to the other uncertainty, in the absence of the other uncertainty. This suggests that when only one uncertainty is explicitly modelled, it attempts to compensate for the lack of the alternative uncertainty when possible.

Secondly, in Figure 2.15 we analyse the quality of our uncertainty measurement using calibration plots from our model on the test set. To form calibration plots for classification

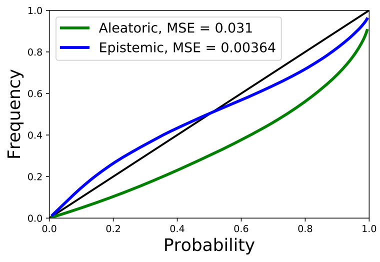


(a) Classification (CamVid)

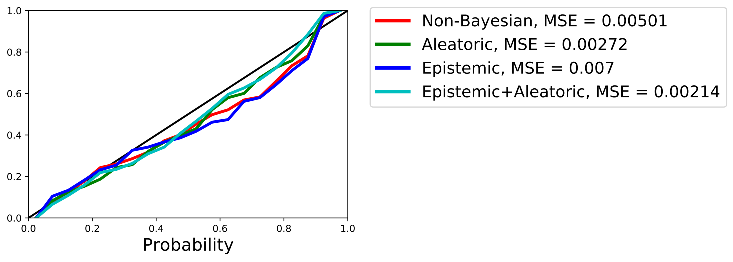


(b) Regression (Make3D)

Fig. 2.14 Precision Recall plots demonstrating both measures of uncertainty can effectively capture accuracy for examples similar to the training dataset, as precision decreases with increasing uncertainty.



(a) Regression (Make3D)



(b) Classification (CamVid)

Fig. 2.15 Uncertainty calibration plots. This plot shows how well uncertainty is calibrated, where perfect calibration corresponds to the line $y = x$, shown in black. We observe an improvement in calibration mean squared error with aleatoric, epistemic and the combination of uncertainties.

models, we discretize our model's predicted probabilities into a number of bins, for all classes and all pixels in the test set. We then plot the frequency of correctly predicted labels for each bin of probability values. Better performing uncertainty estimates should correlate more accurately with the line $y = x$ in the calibration plots. For regression models, we can form calibration plots by comparing the frequency of residuals lying within varying thresholds of the predicted distribution. Figure 2.15 shows the calibration of our classification and regression uncertainties.

Uncertainty with Distance from Training Data

In this section we show two results:

1. Aleatoric uncertainty cannot be explained away with more data,

Train dataset	Test dataset	RMS	Aleatoric variance	Epistemic variance
Make3D / 4	Make3D	5.76	0.506	7.73
Make3D / 2	Make3D	4.62	0.521	4.38
Make3D	Make3D	3.87	0.485	2.78
Make3D / 4	NYUv2	-	0.388	15.0
Make3D	NYUv2	-	0.461	4.87

(a) Regression

Train dataset	Test dataset	IoU	Aleatoric entropy	Epistemic logit variance ($\times 10^{-3}$)
CamVid / 4	CamVid	57.2	0.106	1.96
CamVid / 2	CamVid	62.9	0.156	1.66
CamVid	CamVid	67.5	0.111	1.36
CamVid / 4	NYUv2	-	0.247	10.9
CamVid	NYUv2	-	0.264	11.8

(b) Classification

Table 2.9 Accuracy of aleatoric and epistemic uncertainties for a range of different train and test dataset combinations. We show aleatoric and epistemic uncertainty as the mean value of all pixels in the test dataset. We compare reduced training set sizes (1, ½, ¼) and unrelated test datasets. This shows that aleatoric uncertainty remains approximately constant, while epistemic uncertainty decreases the closer the test data is to the training distribution, demonstrating that epistemic uncertainty can be explained away with sufficient training data (but not for out-of-distribution data).

2. Aleatoric uncertainty does not increase for out-of-data examples (situations different from training set), whereas epistemic uncertainty is required to capture uncertainty from situations different from training set.

In Table 2.9 we give accuracy and uncertainty for models trained on increasing sized subsets of datasets. This shows that epistemic uncertainty decreases as the training dataset gets larger. It also shows that aleatoric uncertainty remains relatively constant and cannot be explained away with more data. Testing the models with a different test set (bottom two lines) shows that epistemic uncertainty increases considerably on those test points which lie far from the training sets.

These results reinforce the case that epistemic uncertainty can be explained away with enough data, but is required to capture situations not encountered in the training set. This is particularly important for safety-critical systems, where epistemic uncertainty is required to detect situations which have never been seen by the model before.

Real-Time Application

Our model based on DenseNet (Jégou et al., 2016) can process a 640×480 resolution image in $150ms$ on a NVIDIA Titan X GPU. The aleatoric uncertainty models add negligible compute. However, epistemic models require expensive Monte Carlo dropout sampling. For models such as ResNet (He et al., 2004), this is possible to achieve economically because only the last few layers contain dropout. Other models, like DenseNet, require the entire architecture to be sampled. This is difficult to parallelise due to GPU memory constraints, and often results in a $50\times$ slow-down for 50 Monte Carlo samples.

2.5 Jointly Learning Geometry and Semantics

Scene understanding requires knowledge of both geometry and semantics. In this section, we wish to jointly learn both geometry and semantics with a single representation. This is a form of multi-task learning, which aims to improve learning efficiency and prediction accuracy by learning multiple objectives from a shared representation (Caruana, 1998). Multi-task learning is prevalent in many applications of machine learning – from computer vision (Kokkinos, 2016) to natural language processing (Collobert and Weston, 2008) to speech recognition (Huang et al., 2013).

We explore multi-task learning within the setting of visual scene understanding in computer vision. Scene understanding algorithms must understand both the geometry and semantics of the scene at the same time. This forms an interesting multi-task learning problem because scene understanding involves joint learning of various regression and classification tasks with different units and scales. Multi-task learning of visual scene understanding is of crucial importance in systems where long computation run-time is prohibitive, such as the ones used in robotics. Combining all tasks into a single model reduces computation and allows these systems to run in real-time.

Prior approaches to simultaneously learning multiple tasks use a naïve weighted sum of losses, where the loss weights are uniform, or manually tuned (Eigen and Fergus, 2015; Kokkinos, 2016; Sermanet et al., 2013). However, we show that performance is highly dependent on an appropriate choice of weighting between each task's loss. Searching for an optimal weighting is prohibitively expensive and difficult to resolve with manual tuning. We observe that the optimal weighting of each task is dependent on the measurement scale (e.g. meters, centimeters or millimeters) and ultimately the magnitude of the task's noise. In this

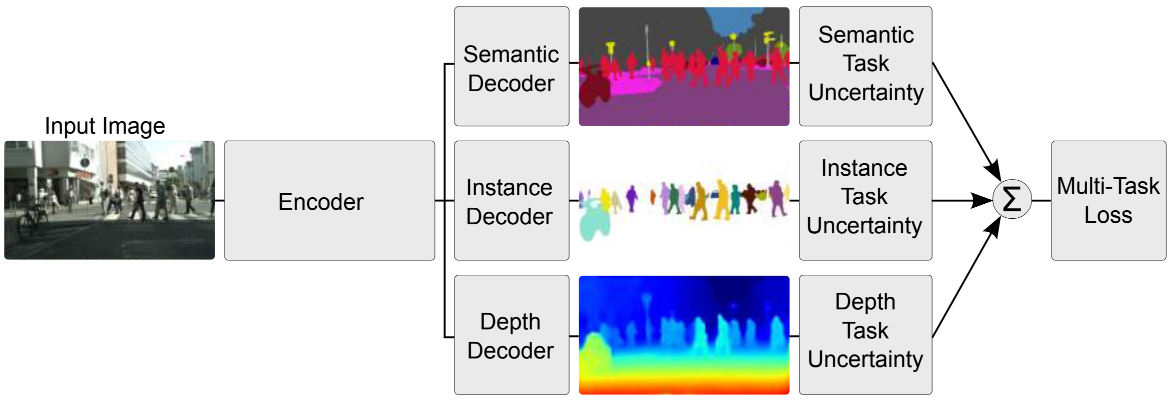


Fig. 2.16 **Multi-task deep learning.** We derive a principled way of combining multiple regression and classification loss functions for multi-task learning. Our architecture takes a single monocular RGB image as input and produces a pixel-wise classification, an instance semantic segmentation and an estimate of per pixel depth. Multi-task learning can improve accuracy over separately trained models because cues from one task, such as depth, are used to regularize and improve the generalization of another domain, such as segmentation.

work we propose a principled way of combining multiple loss functions to simultaneously learn multiple objectives using homoscedastic uncertainty. We interpret homoscedastic uncertainty as task-dependent weighting and show how to derive a principled multi-task loss function which can learn to balance various regression and classification losses. Our method can learn to balance these weightings optimally, resulting in superior performance, compared with learning each task individually.

Specifically, we demonstrate our method in learning scene geometry and semantics with three tasks. Firstly, we learn semantic segmentation. Secondly, our model performs instance segmentation, which is the harder task of segmenting separate masks for each individual object in an image (for example, a separate, precise mask for each individual car on the road) (Bai and Urtasun, 2017; Dai et al., 2016; De Brabandere et al., 2017; Hariharan et al., 2015; Pinheiro et al., 2015). This is a more complicated task than semantic segmentation, as it requires not only an estimate of each pixel's class, but also which object that pixel belongs to. It is also more complicated than object detection, which often predicts object bounding boxes alone (Girshick et al., 2014). Finally, our model predicts pixel-wise metric depth. Depth by recognition has been demonstrated using dense prediction networks with supervised (Eigen and Fergus, 2015) and unsupervised (Garg et al., 2016) deep learning. However it is very hard to estimate depth in a way which generalises well. We show that we can improve our estimation of geometry *and* depth by using semantic labels and multi-task deep learning.

In existing literature, separate deep learning models would be used to learn depth regression, semantic segmentation and instance segmentation to create a complete scene understanding system. Given a single monocular input image, our system is the first to produce a semantic segmentation, a dense estimate of metric depth and an instance level segmentation jointly (Figure 2.16). While other vision models have demonstrated multi-task learning, we show how to learn to combine semantics and geometry. Combining these tasks into a single model ensures that the model agrees between the separate task outputs while reducing computation. Finally, we show that using a shared representation with multi-task learning improves performance on various metrics, making the models more effective.

In summary, the key contributions of this Section are:

1. a novel and principled multi-task loss to simultaneously learn various classification and regression losses of varying quantities and units using homoscedastic task uncertainty,
2. a unified architecture for semantic segmentation, instance segmentation and depth regression,
3. demonstrating the importance of loss weighting in multi-task deep learning and how to obtain superior performance compared to equivalent separately trained models.

2.5.1 Multi-task Learning

We first review related work from the field of multi-task learning. Multi-task learning aims to improve learning efficiency and prediction accuracy for each task, when compared to training a separate model for each task (Baxter et al., 2000; Thrun and Bücken, 1996). It can be considered an approach to inductive knowledge transfer which improves generalisation by sharing the domain information between complimentary tasks. It does this by using a shared representation to learn multiple tasks – what is learned from one task can help learn other tasks (Caruana, 1998).

Fine-tuning (Agrawal et al., 2015; Oquab et al., 2014) is a basic example of multi-task learning, where we can leverage different learning tasks by considering them as a pre-training step. Other models alternate learning between each training task, for example in natural language processing (Collobert and Weston, 2008). Multi-task learning can also be used in a data streaming setting (Thrun and Bücken, 1996), or to prevent forgetting previously learned tasks in reinforcement learning (Kirkpatrick et al., 2017). It can also be used to learn unsupervised features from various data sources with an auto-encoder (Ngiam et al., 2011).

In computer vision there are many examples of methods for multi-task learning. Many focus on semantic tasks, such as classification and semantic segmentation (Liao et al., 2016) or classification and detection (Roddick et al., 2018; Sermanet et al., 2013). MultiNet (Teichmann et al., 2016) proposes an architecture for detection, classification and semantic segmentation. CrossStitch networks (Misra et al., 2016) explore methods to combine multi-task neural activations. Uhrig et al. (Uhrig et al., 2016) learn semantic and instance segmentations under a classification setting. SpineNet (Jamaludin et al., 2016) predicts multiple radiological scores in MRI images for medical imaging. Multi-task deep learning has also been used for geometry and regression tasks. (Eigen and Fergus, 2015) show how to learn semantic segmentation, depth and surface normals. PoseNet (Chapter 3) is a model which learns camera position and orientation. UberNet (Kokkinos, 2016) learns a number of different regression and classification tasks under a single architecture. In this work we are the first to propose a method for jointly learning depth regression, semantic and instance segmentation. Like the model of Eigen and Fergus (2015), our model learns both semantic and geometry representations, which is important for scene understanding. However, our model learns the much harder task of instance segmentation which requires knowledge of both semantics and geometry. This is because our model must determine the class and spatial relationship for each pixel in each object for instance segmentation.

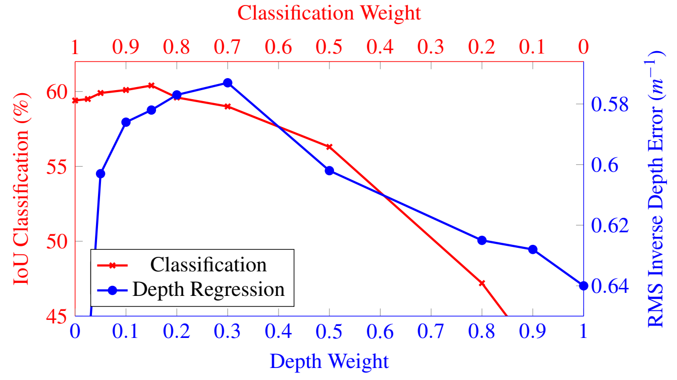
More importantly, all previous methods which learn multiple tasks simultaneously use a naïve weighted sum of losses, where the loss weights are uniform, or crudely and manually tuned. An example is UberNet (Kokkinos, 2016), which is unable to obtain the performance of separately trained models for each task. In this work we give an explaination and solution for this problem. We propose a principled way of combining multiple loss functions to simultaneously learn multiple objectives using homoscedastic task uncertainty. We illustrate the importance of appropriately weighting each task in deep learning to achieve good performance and show that our method can learn to balance these weightings optimally.

2.5.2 Multi Task Learning with Homoscedastic Uncertainty

Multi-task learning concerns the problem of optimising a model with respect to multiple objectives. It is prevalent in many deep learning problems. The naive approach to combining multi-objective losses would be to simply perform a weighted linear sum of the losses for each individual task:

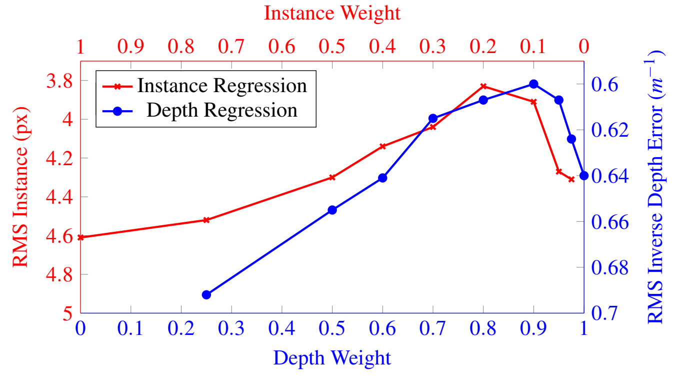
$$L_{total} = \sum_i w_i L_i. \tag{2.15}$$

This is the dominant approach used by prior work (Liao et al., 2016; Sermanet et al., 2013; Teichmann et al., 2016; Uhrig et al., 2016), for example for dense prediction tasks (Kokkinos, 2016), for scene understanding tasks (Eigen and Fergus, 2015) and for rotation (in quaternions) and translation (in meters) for camera pose (Chapter 3). However, there are



Task Weights		Class	Depth
Class	Depth	IoU [%]	Err. [*px*]
1.0	0.0	59.4	-
0.975	0.025	59.5	0.664
0.95	0.05	59.9	0.603
0.9	0.1	60.1	0.586
0.85	0.15	60.4	0.582
0.8	0.2	59.6	0.577
0.7	0.3	59.0	0.573
0.5	0.5	56.3	0.602
0.2	0.8	47.2	0.625
0.1	0.9	42.7	0.628
0.0	1.0	-	0.640
Learned weights with task uncertainty (this work, Section 2.5.4)		**62.7**	**0.533**

(a) Comparing loss weightings when learning **semantic classification and depth regression**



Task Weights		Instance	Depth
Instance	Depth	Err. [*px*]	Err. [*px*]
1.0	0.0	4.61	
0.75	0.25	4.52	0.692
0.5	0.5	4.30	0.655
0.4	0.6	4.14	0.641
0.3	0.7	4.04	0.615
0.2	0.8	3.83	0.607
0.1	0.9	3.91	0.600
0.05	0.95	4.27	0.607
0.025	0.975	4.31	0.624
0.0	1.0		0.640
Learned weights with task uncertainty (this work, Section 2.5.4)		**3.54**	**0.539**

(b) Comparing loss weightings when learning **instance regression and depth regression**

Fig. 2.17 **Learning multiple tasks improves the model's representation and individual task performance**. These figures and tables illustrate the advantages of multi-task learning for (a) semantic classification and depth regression and (b) instance and depth regression. Performance of the model in individual tasks is seen at both edges of the plot where $w = 0$ and $w = 1$. For some balance of weightings between each task, we observe improved performance for both tasks. All models were trained with a learning rate of 0.01 with the respective weightings applied to the losses using the loss function in (2.15). Results are shown using the Tiny CityScapes validation dataset using a down-sampled resolution of 128×256.

a number of issues with this method. Namely, model performance is extremely sensitive to weight selection, w_i, as illustrated in Figure 2.17. These weight hyper-parameters are expensive to tune, often taking many days for each trial. Therefore, it is desirable to find a more convenient approach which is able to learn the optimal weights.

More concretely, let us consider a network which learns to predict pixel-wise depth and semantic class from an input image. In Figure 2.17 the two boundaries of each plot show models trained on individual tasks, with the curves showing performance for varying weights w_i for each task. We observe that at some optimal weighting, the joint network performs better than separate networks trained on each task individually (performance of the model on individual tasks is seen at both edges of the plot: $w = 0$ and $w = 1$). At near-by values to the optimal weight the network performs worse on one of the tasks. However, searching for these optimal weightings is expensive and increasingly difficult with large models with numerous tasks. Figure 2.17 also shows a similar result for two regression tasks; instance segmentation and depth regression. We next show how to learn optimal task weightings using ideas from probabilistic modelling.

2.5.3 Homoscedastic uncertainty as task-dependent uncertainty

In Bayesian modelling, there are two main types of uncertainty one can model (Der Kiureghian and Ditlevsen, 2009).

- *Epistemic uncertainty* is uncertainty in the model, which captures what our model does not know due to lack of training data. It can be explained away with increased training data.

- *Aleatoric uncertainty* captures our uncertainty with respect to information which our data cannot explain. Aleatoric uncertainty can be explained away with the ability to observe all explanatory variables with increasing precision.

Aleatoric uncertainty can again be divided into two sub-categories.

- *Data-dependent* or *Heteroscedastic* uncertainty is aleatoric uncertainty which depends on the input data and is predicted as a model output.

- *Task-dependent* or *Homoscedastic* uncertainty is aleatoric uncertainty which is not dependent on the input data. It is not a model output, rather it is a quantity which stays constant for all input data and varies between different tasks. It can therefore be described as task-dependent uncertainty.

In a multi-task setting, we show that the task uncertainty captures the relative confidence between tasks, reflecting the uncertainty inherent to the regression or classification task. It will also depend on the task's representation or unit of measure. We propose that we can use homoscedastic uncertainty as a basis for weighting losses in a multi-task learning problem.

2.5.4 Multi-task likelihoods

In this section we derive a multi-task loss function based on maximising the Gaussian likelihood with homoscedastic uncertainty. This extends the derivation in Section 2.4.3 to multi-task likelihoods. Let $\mathbf{f}^{\mathbf{W}}(\mathbf{x})$ be the output of a neural network with weights $\mathbf{W}$ on input $\mathbf{x}$. We define the following probabilistic model. For regression tasks we define our likelihood as a Gaussian with mean given by the model output:

$$p(\mathbf{y}|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) = \mathcal{N}(\mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma^2) \tag{2.16}$$

with an observation noise scalar σ. For classification we often squash the model output through a softmax function, and sample from the resulting probability vector:

$$p(\mathbf{y}|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) = \text{Softmax}(\mathbf{f}^{\mathbf{W}}(\mathbf{x})). \tag{2.17}$$

In the case of multiple model outputs, we often define the likelihood to factorise over the outputs, given some sufficient statistics. We define $\mathbf{f}^{\mathbf{W}}(\mathbf{x})$ as our sufficient statistics, and obtain the following multi-task likelihood:

$$p(\mathbf{y}_1, ..., \mathbf{y}_K|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) = p(\mathbf{y}_1|\mathbf{f}^{\mathbf{W}}(\mathbf{x}))...p(\mathbf{y}_K|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) \tag{2.18}$$

with model outputs $\mathbf{y}_1, ..., \mathbf{y}_K$ (such as semantic segmentation, depth regression, etc).

In *maximum likelihood* inference, we maximise the log likelihood of the model. In regression, for example, the log likelihood can be written as

$$\log p(\mathbf{y}|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) \propto -\frac{1}{2\sigma^2}||\mathbf{y} - \mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2 - \log\sigma \tag{2.19}$$

for a Gaussian likelihood (or similarly for a Laplace likelihood) with σ the model's observation noise parameter – capturing how much noise we have in the outputs. We then maximise the log likelihood with respect to the model parameters $\mathbf{W}$ and observation noise parameter σ.

Let us now assume that our model output is composed of two vectors $\mathbf{y}_1$ and $\mathbf{y}_2$, each following a Gaussian distribution:

$$\begin{aligned} p(\mathbf{y}_1, \mathbf{y}_2 | \mathbf{f}^{\mathbf{W}}(\mathbf{x})) &= p(\mathbf{y}_1 | \mathbf{f}^{\mathbf{W}}(\mathbf{x})) \cdot p(\mathbf{y}_2 | \mathbf{f}^{\mathbf{W}}(\mathbf{x})) \\ &= \mathcal{N}(\mathbf{y}_1; \mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma_1^2) \cdot \mathcal{N}(\mathbf{y}_2; \mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma_2^2). \end{aligned} \tag{2.20}$$

This leads to the *minimisation* objective, $\mathcal{L}(\mathbf{W}, \sigma_1, \sigma_2)$, (our loss) for our multi-output model:

$$\begin{aligned} &= -\log p(\mathbf{y}_1, \mathbf{y}_2 | \mathbf{f}^{\mathbf{W}}(\mathbf{x})) \\ &\propto \frac{1}{2\sigma_1^2} ||\mathbf{y}_1 - \mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2 + \frac{1}{2\sigma_2^2} ||\mathbf{y}_2 - \mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2 + \log \sigma_1 \sigma_2 \\ &= \frac{1}{2\sigma_1^2} \mathcal{L}_1(\mathbf{W}) + \frac{1}{2\sigma_2^2} \mathcal{L}_2(\mathbf{W}) + \log \sigma_1 \sigma_2 \end{aligned} \tag{2.21}$$

Where we write $\mathcal{L}_1(\mathbf{W}) = ||\mathbf{y}_1 - \mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2$ for the loss of the first output variable, and similarly for $\mathcal{L}_2(\mathbf{W})$.

We interpret minimising this last objective with respect to σ_1 and σ_2 as learning the relative weight of the losses $\mathcal{L}_1(\mathbf{W})$ and $\mathcal{L}_2(\mathbf{W})$ adaptively, based on the data. As σ_1 – the noise parameter for the variable $\mathbf{y}_1$ – increases, we have that the weight of $\mathcal{L}_1(\mathbf{W})$ decreases. On the other hand, as the noise decreases, we have that the weight of the respective objective increases. The noise is discouraged from increasing too much (effectively ignoring the data) by the last term in the objective, which acts as a regulariser for the noise terms.

This construction can be trivially extended to multiple regression outputs. However, the extension to classification likelihoods is more interesting. We adapt the classification likelihood to squash a *scaled* version of the model output through a softmax function:

$$p(\mathbf{y} | \mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma) = \text{Softmax}\left(\frac{1}{\sigma^2} \mathbf{f}^{\mathbf{W}}(\mathbf{x})\right) \tag{2.22}$$

with a positive scalar σ. This can be interpreted as a Boltzmann distribution (also called Gibbs distribution) where the input is scaled by σ^2 (often referred to as *temperature*). This scalar is either fixed or can be learnt, where the parameter's magnitude determines how 'uniform' (flat) the discrete distribution is. This relates to its uncertainty, as measured in entropy. The log likelihood for this output can then be written as

$$\begin{aligned}\log p(\mathbf{y}=c|\mathbf{f}^{\mathbf{W}}(\mathbf{x}),\sigma) = &\frac{1}{\sigma^2} f_c^{\mathbf{W}}(\mathbf{x}) \\ &- \log \sum_{c'} \exp\left(\frac{1}{\sigma^2} f_{c'}^{\mathbf{W}}(\mathbf{x})\right)\end{aligned} \tag{2.23}$$

with $f_c^{\mathbf{W}}(\mathbf{x})$ the c'th element of the vector $\mathbf{f}^{\mathbf{W}}(\mathbf{x})$.

Next, assume that a model's multiple outputs are composed of a continuous output $\mathbf{y}_1$ and a discrete output $\mathbf{y}_2$, modelled with a Gaussian likelihood and a softmax likelihood, respectively. Like before, the joint loss, $\mathscr{L}(\mathbf{W},\sigma_1,\sigma_2)$, is given as:

$$\begin{aligned}
&= -\log p(\mathbf{y}_1,\mathbf{y}_2=c|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) \\
&= -\log \mathcal{N}(\mathbf{y}_1;\mathbf{f}^{\mathbf{W}}(\mathbf{x}),\sigma_1^2)\cdot \mathrm{Softmax}(\mathbf{y}_2=c;\mathbf{f}^{\mathbf{W}}(\mathbf{x}),\sigma_2) \\
&= \frac{1}{2\sigma_1^2}||\mathbf{y}_1-\mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2 + \log\sigma_1 - \log p(\mathbf{y}_2=c|\mathbf{f}^{\mathbf{W}}(\mathbf{x}),\sigma_2) \\
&= \frac{1}{2\sigma_1^2}\mathscr{L}_1(\mathbf{W}) + \frac{1}{\sigma_2^2}\mathscr{L}_2(\mathbf{W}) + \log\sigma_1 \\
&\qquad + \log \frac{\sum_{c'}\exp\left(\frac{1}{\sigma_2^2}f_{c'}^{\mathbf{W}}(\mathbf{x})\right)}{\left(\sum_{c'}\exp\left(f_{c'}^{\mathbf{W}}(\mathbf{x})\right)\right)^{\frac{1}{\sigma_2^2}}} \\
&\approx \frac{1}{2\sigma_1^2}\mathscr{L}_1(\mathbf{W}) + \frac{1}{\sigma_2^2}\mathscr{L}_2(\mathbf{W}) + \log\sigma_1 + \log\sigma_2,
\end{aligned} \tag{2.24}$$

where again we write $\mathscr{L}_1(\mathbf{W}) = ||\mathbf{y}_1-\mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2$ for the Euclidean loss of $\mathbf{y}_1$, write $\mathscr{L}_2(\mathbf{W}) = -\log \mathrm{Softmax}(\mathbf{y}_2,\mathbf{f}^{\mathbf{W}}(\mathbf{x}))$ for the cross entropy loss of $\mathbf{y}_2$ (with $\mathbf{f}^{\mathbf{W}}(\mathbf{x})$ not scaled), and optimise with respect to $\mathbf{W}$ as well as σ_1, σ_2. In the last transition we introduced the explicit simplifying assumption $\frac{1}{\sigma_2}\sum_{c'}\exp\left(\frac{1}{\sigma_2^2}f_{c'}^{\mathbf{W}}(\mathbf{x})\right) \approx \left(\sum_{c'}\exp\left(f_{c'}^{\mathbf{W}}(\mathbf{x})\right)\right)^{\frac{1}{\sigma_2^2}}$ which becomes an equality when $\sigma_2 \rightarrow 1$. This has the advantage of simplifying the optimisation objective, as well as empirically improving results.

This last objective can be seen as learning the relative weights of the losses for each output. Large scale values σ_2 will decrease the contribution of $\mathscr{L}_2(\mathbf{W})$, whereas small scale σ_2 will increase its contribution. The scale is regulated by the last term in the equation. The objective is penalised when setting σ_2 too large.

This construction can be trivially extended to arbitrary combinations of discrete and continuous loss functions, allowing us to learn the relative weights of each loss in a principled and well-founded way. This loss is smoothly differentiable, and is well formed such that the task weights will not converge to zero. In contrast, directly learning the weights using a simple linear sum of losses (2.15) would result in weights which quickly converge to zero. In the following sections we introduce our experimental model and present empirical results.

In practice, we train the network to predict the log variance, $s := \log \sigma^2$. This is because it is more stable than regressing the variance, σ^2, as the loss avoids any division by zero. The exponential mapping also allows us to regress unconstrained scalar values, where $\exp(-s)$ is resolved to the positive domain giving valid values for variance.

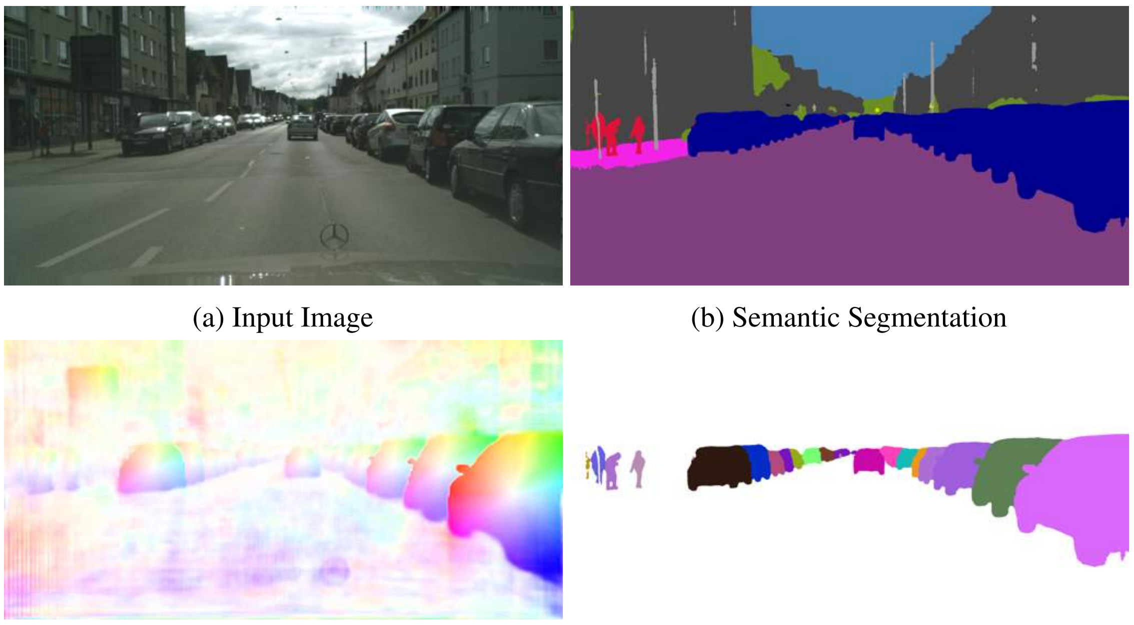
2.5.5 Scene Understanding Tasks

To understand semantics and geometry we first propose an architecture which can learn regression and classification outputs, at a pixel level. Our architecture is a deep convolutional encoder decoder network (Badrinarayanan et al., 2017). Our model consists of a number of convolutional encoders which produce a shared representation, followed by a number of task-specific convolutional decoders. A high level summary is shown in Figure 2.16. The purpose of the encoder is to learn a deep mapping to produce rich, contextual features, using domain knowledge from a number of related tasks. The purpose of the decoder is to learn a mapping from the shared features to an output. This section describes how we learn each of the scene understanding tasks.

Semantic Segmentation. We use the cross-entropy loss to learn pixel-wise class probabilities, averaging the loss over the pixels with semantic labels in each mini-batch.

Instance Segmentation. An intuitive method for defining which instance a pixel belongs to is an association to the instance's centroid. We use a regression approach for instance segmentation (Liang et al., 2015). This approach is inspired by (Leibe et al., 2008) which identifies instances using Hough votes from object parts. In this work we extend this idea by using votes from individual pixels using deep learning. We learn an instance vector, $\hat{x}_n$, for each pixel coordinate, c_n, which points to the centroid of the pixel's instance, i_n, such that $i_n = \hat{x}_n + c_n$. We train this regression with an L_1 loss using ground truth labels x_n, averaged over all labelled pixels, N_I, in a mini-batch: $\mathscr{L}_{Instance} = \frac{1}{|N_I|} \sum_{N_I} \|x_n - \hat{x}_n\|_1$.

Figure 2.18 details the representation we use for instance segmentation. Figure 2.18(a) shows the input image and a mask of the pixels which are of an instance class (at test time inferred from the predicted semantic segmentation). Figure 2.18(b) and Figure 2.18(c) show the ground truth and predicted instance vectors for both x and y coordinates. We then cluster these votes using OPTICS (Ankerst et al., 1999), resulting in the predicted instance segmentation output in Figure 2.18(d).

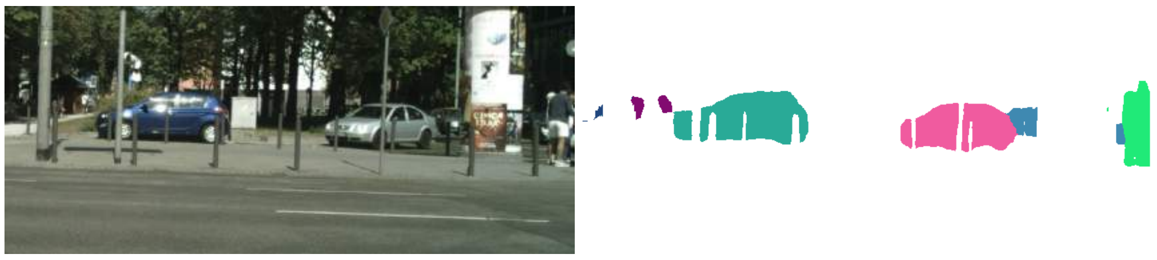

(a) Input Image (b) Semantic Segmentation

(c) Instance vector regression (d) Instance Segmentation

Fig. 2.18 **Instance centroid regression method.** For each pixel, we regress a vector pointing to the instance's centroid. The loss is only computed over pixels which are from instances. We visualise (c) by representing colour as the orientation of the instance vector, and intensity as the magnitude of the vector.

One of the most difficult cases for instance segmentation algorithms to handle is when the instance mask is split due to occlusion. Figure 2.19 shows that our method can handle these situations, by allowing pixels to vote for their instance centroid with geometry. Methods which rely on watershed approaches (Bai and Urtasun, 2017), or instance edge identification approaches fail in these scenarios.

To obtain segmentations for each instance, we now need to estimate the instance centres, $\hat{i}_n$. We propose to consider the estimated instance vectors, $\hat{x}_n$, as votes in a Hough parameter space and use a clustering algorithm to identify these instance centres. OPTICS (Ankerst et al., 1999), is an efficient density based clustering algorithm. It is able to identify an unknown number of multi-scale clusters with varying density from a given set of samples. We chose OPTICS for two reasons. Crucially, it does not assume knowledge of the number of clusters like algorithms such as k-means (MacQueen et al., 1967). Secondly, it does not assume a canonical instance size or density like discretised binning approaches (Comaniciu and Meer, 2002). Using OPTICS, we cluster the points $c_n + \hat{x}_n$ into a number of estimated instances, $\hat{i}$. We can then assign each pixel, p_n to the instance closest to its estimated instance vector, $c_n + \hat{x}_n$.



(a) Input Image (b) Instance Segmentation

Fig. 2.19 This example shows two cars which are occluded by trees and lampposts, making the instance segmentation challenging. Our instance segmentation method can handle occlusions effectively. We can correctly handle segmentation masks which are split by occlusion, yet part of the same instance, by incorporating semantics and geometry.

Depth Regression. We train with supervised labels using pixel-wise metric inverse depth using a L_1 loss function: $\mathscr{L}_{Depth} = \frac{1}{|N_D|} \sum_{N_D} \left\| d_n - \hat{d}_n \right\|_1$. Our architecture estimates inverse depth, $\hat{d}_n$, because it can represent points at infinite distance (such as sky). We can obtain inverse depth labels, d_n, from a RGBD sensor or stereo imagery. Pixels which do not have an inverse depth label are ignored in the loss.

2.5.6 Model Architecture

We base our model on the recently introduced DeepLabV3 (Chen et al., 2017) segmentation architecture. We use ResNet101 (He et al., 2016) as our base feature encoder, with dilated convolutions, resulting in a feature map which is downsampled by a factor of 8 compared with the original input image. We then append dilated (atrous) convolutional ASPP module (Chen et al., 2017). This module is designed to improve the contextual reasoning of the network. We use an ASPP module comprised of four parallel convolutional layers, with 256 output channels and dilation rates (1, 12, 24, 36), with kernel sizes (1^2, 3^2, 3^2, 3^2). Additionally, we also apply global average pooling to the encoded features, and convolve them to 256 dimensions with a 1×1 kernel. We apply batch normalisation to each of these layers and concatenate the resulting 1280 features together. This produces the shared representation between each task.

We then split the network, to decode this representation to a given task output. For each task, we construct a decoder consisting of two layers. First, we apply a 1×1 convolution, outputting 256 features, followed by batch normalisation and a non-linear activation. Finally, we convolve this output to the required dimensions for a given task. For classification, this will be equal to the number of semantic classes, otherwise the output will be 1 or 2 channels for depth or instance segmentation respectively. Finally, we apply bilinear upsampling to scale the output to the same resolution as the input.

The majority of the model's parameters and depth is in the feature encoding, with very little flexibility in each task decoder. This illustrates the attraction of multi-task learning; most of the compute can be shared between each task to learn a better shared representation.

Optimisation.

For all experiments, we use an initial learning rate of 2.5×10^{-3} and polynomial learning rate decay $(1 - \frac{iter}{max\ iter})^{0.9}$. We train using stochastic gradient descent, with Nesterov updates and momentum 0.9 and weight decay 10^{-4}. We conduct all experiments in this paper using PyTorch.

For the experiments on the Tiny CityScapes validation dataset (using a down-sampled resolution of 128×256) we train over $50,000$ iterations, using 256×256 crops with batch size of 8 on a single NVIDIA 1080Ti GPU. We apply random horizontal flipping to the data.

For the full-scale CityScapes benchmark experiment, we train over $100,000$ iterations with a batch size of 16. We apply random horizontal flipping (with probability 0.5) and random scaling (selected from 0.7 - 2.0) to the data during training, before making a 512×512 crop. The training data is sampled uniformly, and is randomly shuffled for each epoch. Training takes five days on a single computer with four NVIDIA 1080Ti GPUs.

2.5.7 Experiments

We demonstrate the efficacy of our method on CityScapes (Cordts et al., 2016), a large dataset for road scene understanding. It comprises of stereo imagery, from automotive grade stereo cameras with a $22cm$ baseline, labelled with instance and semantic segmentations from 20 classes. Depth images are also provided, labelled using SGM (Hirschmüller, 2008), which we treat as pseudo ground truth. Additionally, we assign zero inverse depth to pixels labelled as sky. The dataset was collected from a number of cities in fine weather and consists of 2,975 training and 500 validation images at 2048×1024 resolution. 1,525 images are withheld for testing on an online evaluation server.

In Table 2.10 we compare individual models to multi-task learning models using a naïve weighted loss or the task uncertainty weighting we propose in this paper. To reduce the computational burden, we train each model at a reduced resolution of 128×256 pixels, over $50,000$ iterations. When we downsample the data by a factor of four, we also need to scale the disparity labels accordingly. Table 2.10 clearly illustrates the benefit of multi-task

Loss	Task Weights Seg.	Inst.	Depth	Segmentation IoU [%]	Instance Mean Error [px]	Inverse Depth Mean Error [px]
Segmentation only	1	0	0	59.4%	-	-
Instance only	0	1	0	-	4.61	-
Depth only	0	0	1	-	-	0.640
Unweighted sum of losses	0.333	0.333	0.333	50.1%	3.79	0.592
Approx. optimal weights	0.89	0.01	0.1	62.8%	3.61	0.549
2 task uncertainty weighting	✓	✓		61.0%	**3.42**	-
2 task uncertainty weighting	✓		✓	62.7%	-	0.533
2 task uncertainty weighting		✓	✓	-	3.54	0.539
3 task uncertainty weighting	✓	✓	✓	**63.4%**	3.50	**0.522**

Table 2.10 Quantitative improvement when learning semantic segmentation, instance segmentation and depth with our multi-task loss. Experiments were conducted on the Tiny CityScapes dataset (sub-sampled to a resolution of 128×256). Results are shown from the validation set. We observe an improvement in performance when training with our multi-task loss, over both single-task models and weighted losses. Additionally, we observe an improvement when training on all three tasks ($3 \times \checkmark$) using our multi-task loss, compared with all pairs of tasks alone (denoted by $2 \times \checkmark$). This shows that our loss function can automatically learn a better performing weighting between the tasks than the baselines.

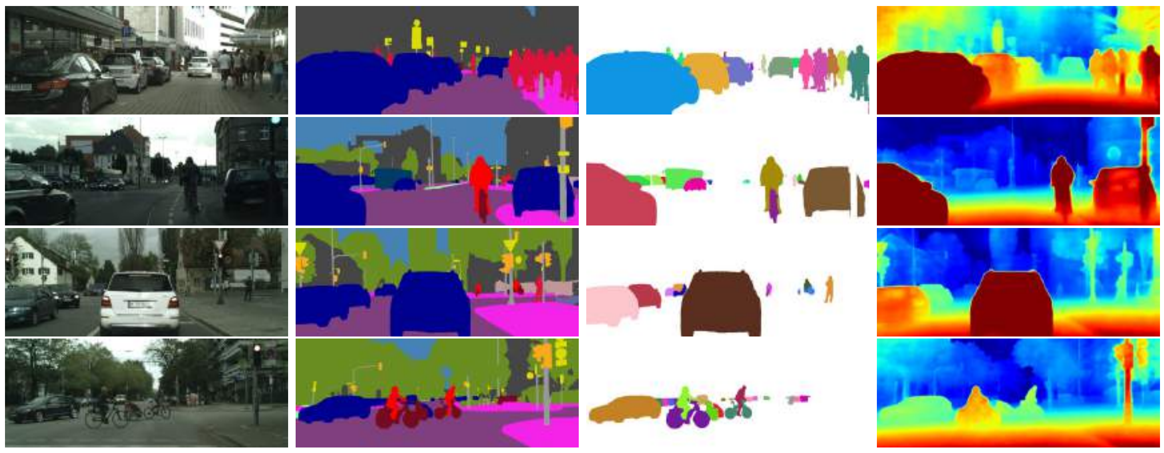

(a) Input image (b) Segmentation output (c) Instance output (d) Depth output

Fig. 2.20 **Qualitative results for multi-task learning of geometry and semantics for road scene understanding with a single network trained on all tasks**. We observe that multi-task learning improves the smoothness and accuracy for depth perception because it learns a representation that uses cues from other tasks, such as segmentation (and vice versa).

learning, which obtains significantly better performing results than individual task models. For example, using our method we improve classification results from 59.4% to 63.4%.

We also compare to a number of naïve multi-task losses. We compare weighting each task equally and using approximately optimal weights. Using a uniform weighting results in poor performance, in some cases not even improving on the results from the single task model. Obtaining approximately optimal weights is difficult with increasing number of tasks as it requires an expensive grid search over parameters. However, even these weights perform worse compared with our proposed method. Figure 2.17 shows that using task uncertainty weights can even perform better compared to optimal weights found through fine-grained grid search. We believe that this is due to two reasons. First, grid search is restricted in accuracy by the resolution of the search. Second, optimising the task weights using a homoscedastic noise term allows for the weights to be dynamic during training. In general, we observe that the uncertainty term decreases during training which improves the optimisation process.

We benchmark our model using the full-size CityScapes dataset. In Table 2.11 we compare to a number of other state of the art methods in all three tasks. Our method is the first model which completes all three tasks with a single model. We compare favourably with other approaches, outperforming many which use comparable training data and inference tools. Figure 2.20 shows some qualitative examples of our model.

This task uncertainty loss is also robust to the value we use to initialise the task uncertainty values. One of the attractive properties of our approach to weighting multi-task losses is that

	Semantic Segmentation				Instance Segmentation				Monocular Disparity Estimation	
Method	IoU class	iIoU class	IoU cat	iIoU cat	AP	AP 50%	AP 100m	AP 50m	Mean Error [px]	RMS Error [px]
Semantic segmentation, instance segmentation and depth regression methods (this work)										
Multi-Task Learning	78.5	57.4	89.9	77.7	21.6	39.0	35.0	37.0	2.92	5.88
Semantic segmentation and instance segmentation methods										
Uhrig et al. (Uhrig et al., 2016)	64.3	41.6	85.9	73.9	8.9	21.1	15.3	16.7	-	-
Instance segmentation only methods										
Mask R-CNN (He et al., 2017)	-	-	-	-	26.2	49.9	37.6	40.1	-	-
Deep Watershed (Bai and Urtasun, 2017)	-	-	-	-	19.4	35.3	31.4	36.8	-	-
R-CNN + MCG (Cordts et al., 2016)	-	-	-	-	4.6	12.9	7.7	10.3	-	-
Semantic segmentation only methods										
DeepLab V3 (Chen et al., 2017)	81.3	60.9	91.6	81.7	-	-	-	-	-	-
PSPNet (Zhao et al., 2017)	81.2	59.6	91.2	79.2	-	-	-	-	-	-
Adelaide (Lin et al., 2015)	71.6	51.7	87.3	74.1	-	-	-	-	-	-

Table 2.11 **CityScapes Benchmark (Cordts et al., 2016).** We show results from the test dataset using the full resolution of 1024×2048 pixels. For the full leaderboard, please see www.cityscapes-dataset.com/benchmarks. The disparity (inverse depth) metrics were computed against the CityScapes depth maps, which are sparse and computed using SGM stereo (Hirschmuller, 2005). Note, these comparisons are not entirely fair, as many methods use ensembles of different training datasets. Our method is the first to address all three tasks with a single model.

it is robust to the initialisation choice for the homoscedastic noise parameters. Figure 2.21 shows that for an array of initial choices of $\log \sigma^2$ from -2.0 to 5.0 the homoscedastic noise and task loss is able to converge to the same minima. Additionally, the homoscedastic noise terms converges after only 100 iterations, while the network requires $30,000+$ iterations to train. Therefore our model is robust to the choice of initial value for the weighting terms.

Figure 2.22 shows losses and uncertainty estimates for each task during training of the final model on the full-size CityScapes dataset. At a point 500 iterations into training, the model estimates task variance of 0.60, 62.5 and 13.5 for semantic segmentation, instance segmentation and depth regression, respectively. Becuase the losses are weighted by the inverse of the uncertainty estimates, this results in a task weighting ratio of approximately 23 : 0.22 : 1 between semantics, instance and depth, respectively. At the conclusion of training, the three tasks have uncertainty estimates of 0.075, 3.25 and 20.4, which results in effective weighting between the tasks of 43: 0.16 : 1. This shows how the task uncertainty estimates evolve over time, and the approximate final weightings the network learns. We observe they are far from uniform, as is often assumed in previous literature.

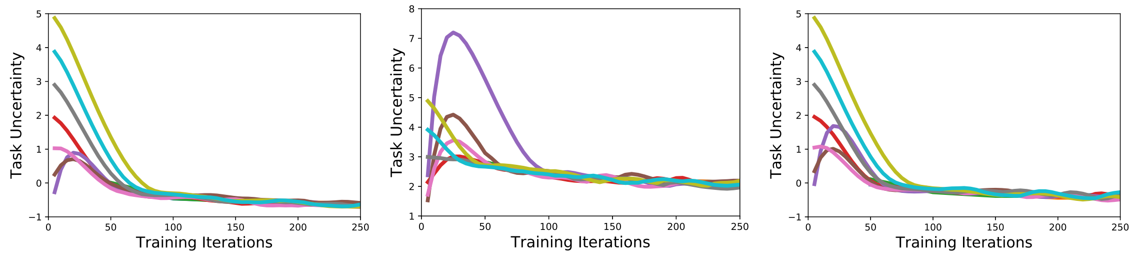
Interestingly, we observe that this loss allows the network to dynamically tune the weighting. Typically, the homoscedastic noise terms decrease in magnitude as training progresses. This makes sense, as during training the model becomes more effective at a task. Therefore the error, and uncertainty, will decrease. This has a side-effect of increasing the effective learning rate – because the overall uncertainty decreases, the weight for each task's loss increases. In our experiments we compensate for this by annealing the learning rate with a power law.

2.5.8 Discussion

There are many interesting questions left unanswered. Firstly, our results show that there is usually not a single optimal weighting for all tasks. Therefore, what is the optimal weighting? Is multi-task learning an ill-posed optimisation problem without a single higher-level goal?

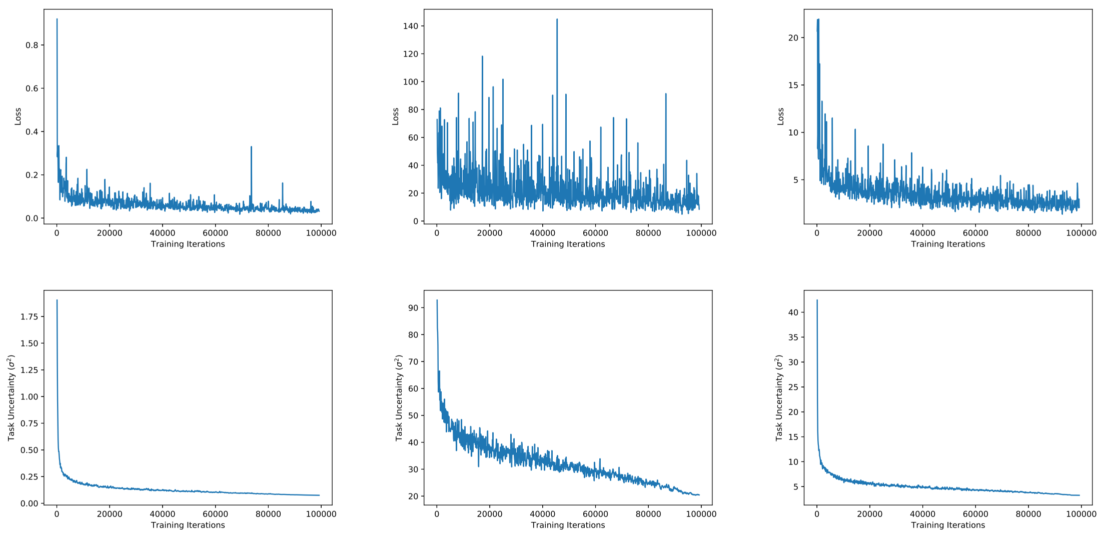
A second interesting question is where the optimal location is for splitting the shared encoder network into separate decoders for each task? What network depth is best for the shared multi-task representation? We leave this to future work.

Why do the semantics and depth tasks out-perform the semantics and instance tasks results in Table 2.10? Clearly the three tasks explored in this paper are complimentary and useful for learning a rich representation about the scene. It would be beneficial to be able to quantify the relationship between tasks and how useful they would be for multi-task representation learning. (Zamir et al., 2018) is some excellent initial work towards this goal.



(a) Semantic segmentation task (b) Instance segmentation task (c) Depth regression task

Fig. 2.21 **Training plots showing convergence of homoscedastic noise and task loss** for an array of initialisation choices for the homoscedastic uncertainty terms for all three tasks. Each plot shows the the homoscedastic noise value optimises to the same solution from a variety of initialisations. Despite the network taking 10,000+ iterations for the training loss to converge, the task uncertainty converges very rapidly after only 100 iterations.



(a) Semantic segmentation task (b) Instance segmentation task (c) Depth regression task

Fig. 2.22 **Learning task uncertainty.** These training plots show the losses and task uncertainty estimates for each task during training. Results are shown for the final model, trained on the fullsize CityScapes dataset.

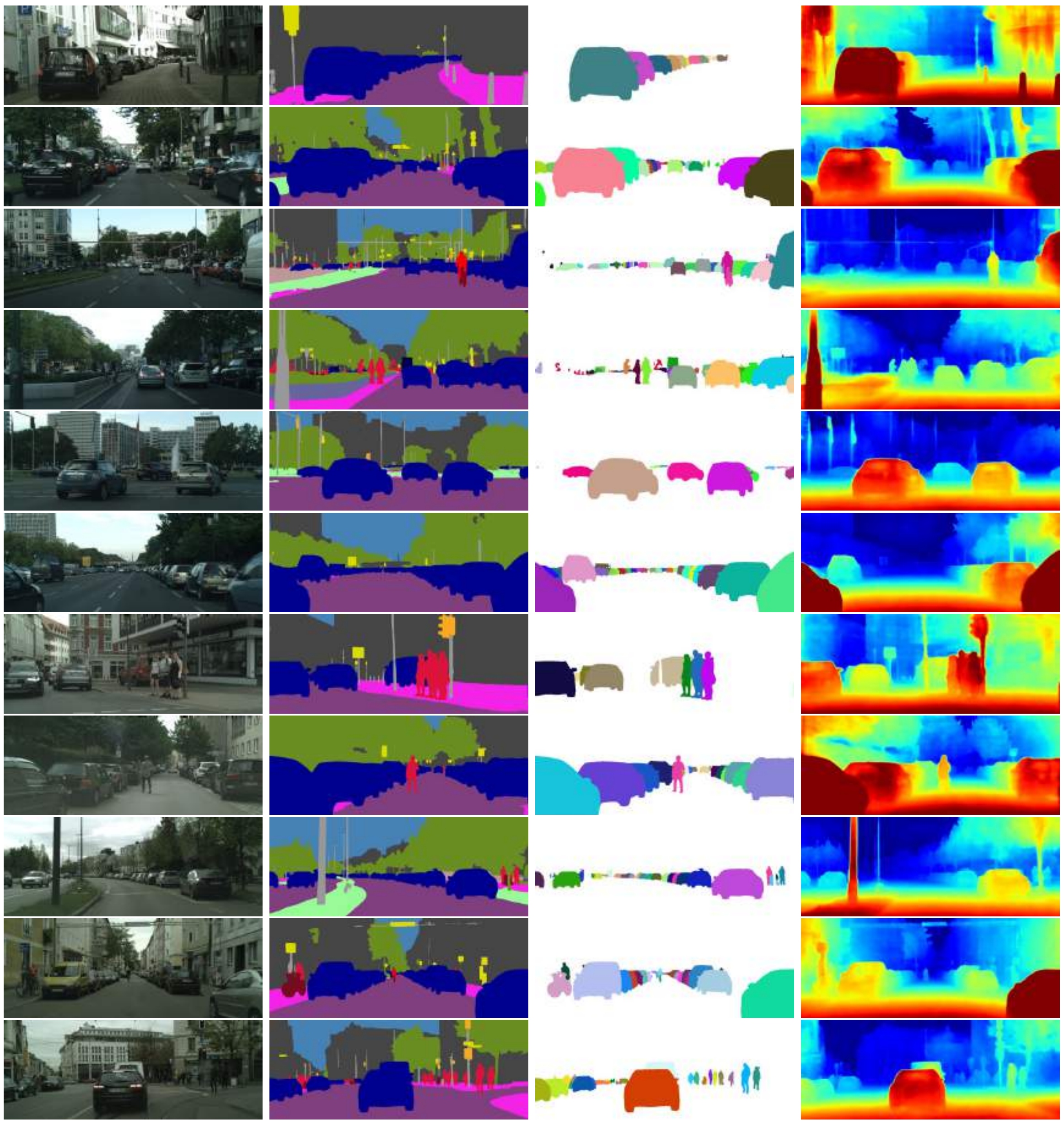

(a) Input image (b) Semantic segmentation (c) Instance segmentation (d) Depth regression

Fig. 2.23 **More qualitative results on test images from the CityScapes dataset.**

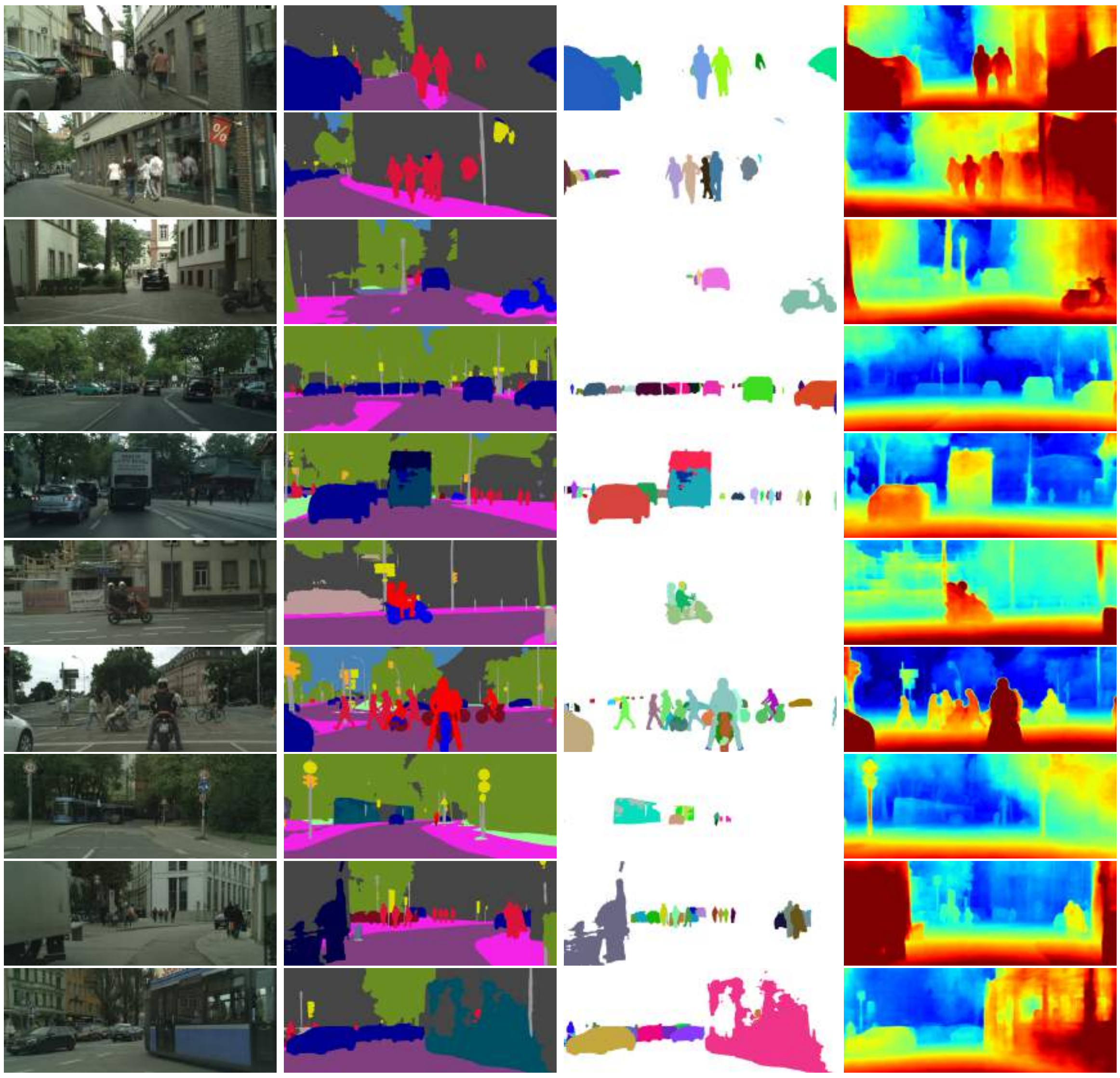

(a) Input image (b) Semantic segmentation (c) Instance segmentation (d) Depth regression

Fig. 2.24 **Example where our model fails on the CityScapes test data.** The first two rows show examples of challenging visual effects such as reflection, which confuse the model. Rows three and four show the model incorrectly distinguishing between road and footpath. This is a common mistake, which we believe is due to a lack of contextual reasoning. Rows five, six and seven demonstrate incorrect classification of a rare class (bus, fence and motorbike, respectively). Finally, the last two rows show failure due to occlusion and where the object is too big for the model's receptive field. Additionally, we observe that failures are highly correlated between the modes, which makes sense as each output is conditioned on the same feature vector. For example, in the second row, the incorrect labelling of the reflection as a person causes the depth estimation to predict human geometry.

Finally, a comment on the model's failure modes. The model exhibits similar failure modes to state-of-the-art single-task models. For example, failure with objects out of the training distribution, occlusion or visually challenging situations. However, we also observe our multi-task model tends to fail with similar effect in all three modalities. For example, an erroneous pixel's prediction in one task will often be highly correlated with error in another modality. Some examples can be seen in Figure 2.23.

2.6 Summary

In this chapter, we investigated the problem of scene understanding and estimating the semantics and geometry of a scene from a single image. We briefly summarise the main conclusions within the three main themes of this dissertation.

End-to-end deep learning. First, we presented SegNet, a deep convolutional network architecture for per-pixel output, capable of being trained end-to-end. The main motivation behind SegNet was the need to design an efficient architecture for road scene understanding which is efficient both in terms of memory and computational time. We analysed SegNet and compared it with other important variants to reveal the trade-offs involved in designing architectures for segmentation. Those which store the encoder network feature maps in full perform best but consume more memory during inference time. SegNet on the other hand is more efficient since it only stores the max-pooling indices of the feature maps and uses them in its decoder network to achieve good performance.

Uncertainty. We presented a novel Bayesian deep learning framework to learn a mapping to aleatoric uncertainty from the input data, which is composed on top of epistemic uncertainty models based on Monte Carlo dropout. We derived a framework for both regression and classification applications. We showed that it is important to model *aleatoric* uncertainty for:

- Large data situations, where epistemic uncertainty is explained away,
- Real-time applications, because we can form aleatoric models without expensive Monte Carlo samples.

And *epistemic* uncertainty is important for:

- Safety-critical applications, because epistemic uncertainty is required to understand examples which are different from training data,
- Small datasets where the training data is sparse.

However aleatoric and epistemic uncertainty models are not mutually exclusive. We showed that the combination is able to achieve new state-of-the-art results on depth regression and semantic segmentation benchmarks.

Geometry. Finally, in order to construct a scene understanding system, algorithms must learn both semantics and geometry. We showed how to formulate this as a multi-task learning problem. We construct a single model which is capable of learning a single representation of semantics and geometry.

We showed that correctly weighting loss terms is of paramount importance for multi-task learning problems. We demonstrated that homoscedastic (task) uncertainty is an effective way to weight losses. We derived a principled loss function which can learn a relative weighting automatically from the data and is robust to the weight initialization. We showed that this can improve performance for scene understanding tasks with a unified architecture for semantic segmentation, instance segmentation and per-pixel depth regression. We demonstrated modelling task-dependent homoscedastic uncertainty improves the model's representation and each task's performance when compared to separate models trained on each task individually.

Chapter 3

Localisation

3.1 Introduction

In this chapter we address the problem of localisation — estimating the camera's position and orientation in three-dimensional space. This is also colloquially known as the *kidnapped robot problem* and is essential for many applications including mobile robotics, navigation and augmented reality.

Designing a system for reliable large scale localisation is a challenging problem. The discovery of the positioning system in mammalian brains, located in the hippocampus, was awarded the 2014 Nobel prize in Physiology or Medicine (Moser et al., 2008; O'Keefe and Nadel, 1978). State of the art computer vision localisation systems perform very well within controlled environments (Engel et al., 2014; Klein and Murray, 2007; Mur-Artal et al., 2015; Newcombe et al., 2011; Sattler et al., 2014). However, we are yet to see their wide-spread use in the wild because of their inability to cope with novel viewpoints or large environmental changes.

Many of the visual localisation systems use point landmarks such as corners (Robertson and Cipolla, 2004), SIFT (Lowe, 2004) or ORB (Rublee et al., 2011) to localise. These features perform well for incremental tracking and estimating ego-motion (Mur-Artal et al., 2015). However, point features cannot encode context and are not able to create a representation which is sufficiently robust to challenging real-world scenarios. For example, they often fail under varying weather, lighting or environmental conditions. Additionally, they lack the ability to capture global context and require robust aggregation of hundreds of points to form a consensus to predict pose (Zeisl et al., 2015).

In this Chapter, Section 3.4 and Section 3.5 was collaborative work with Matthew Grimes and Roberto Cipolla and was published in (Kendall et al., 2015).

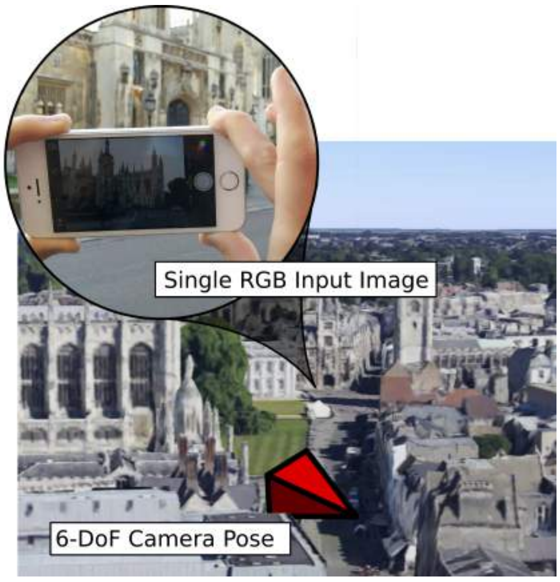


Fig. 3.1 **PoseNet** (Kendall et al., 2015) is trained end-to-end to estimate the camera's six degree of freedom pose from a single monocular image.

This chapter proposes a framework which learns to localise from a single image using deep learning. Our localisation system, PoseNet (Kendall and Cipolla, 2016, 2017; Kendall et al., 2015), takes a single RGB image and regresses the camera's 6-DoF pose relative to a scene. Figure 3.1 illustrates an example. The algorithm is simple in the fact that it consists of a convolutional neural network trained end-to-end to regress the camera's orientation and position. We show that we can localise more robustly using deep learning, compared with point features such as SIFT (Lowe, 2004). PoseNet learns a representation using the entire image context based on appearance and shape. These features generalise well and can localise across challenging lighting and appearance changes. It operates in real time, taking 5ms to run, and obtains approximately 2m and 3 degrees accuracy for large scale outdoor scenes (covering a ground area of up to 50,000m^2). It is very scalable as it does not require a large database of landmarks. Rather, it learns a mapping from pixels to a high dimensional space linear with pose.

We introduce novel techniques and loss functions to design the deep convolutional neural network camera pose regressor. We leverage transfer learning from recognition to relocalisation with very large scale classification datasets. Additionally we use structure from motion to automatically generate training labels (camera poses) from a video of the scene. This reduces the human labour in creating labelled video datasets to just recording the video. We show that the system learns to compute feature vectors which are easily mapped to pose, and which also generalize to unseen scenes with a few additional training samples.

The remainder of the chapter discusses two improvements to PoseNet, using *geometry* and *uncertainty*.

The main weakness of PoseNet in its initial form was that, despite its scalability and robustness, it does not produce metric accuracy which is comparable to other geometric methods (Sattler et al., 2014; Svarm et al., 2014). We argue that a contributing factor to this was because PoseNet naively applied a deep learning model end-to-end to learn camera pose. Therefore, we reconsider this problem with a grounding in geometry. We wish to build upon the decades of research into multi-view geometry (Hartley and Zisserman, 2000) to improve our ability to use deep learning to regress camera pose.

We then improve the performance of PoseNet with geometrically-formed loss functions. It is not trivial to simply regress position and rotation quantities using supervised learning. PoseNet required a weighting factor to balance these two properties, but it was not tolerant to the selection of this hyperparameter. In Section 3.4.3 we explore loss functions which remove this hyperparameter, or optimise it directly from the data. In Section 3.4.3 we show how to train directly from the scene geometry using the reprojection error.

In Section 3.5 we demonstrate our system on an array of datasets, ranging from individual indoor rooms, to the Dubrovnik city dataset (Li et al., 2012). We show that our geometric approach can improve PoseNet's efficacy across many different datasets – narrowing the deficit to traditional SIFT feature-based algorithms. For outdoor scenes ranging from 50,000m^2 to 2km^2 we can achieve relocalisation accuracies of a few meters and a few degrees. In small rooms we are able to achieve accuracies of 0.2-0.4m.

Finally, in Section 3.6 we show how to formulate Bayesian deep learning models for localisation. We introduce Bayesian PoseNet which can estimate model uncertainty. We show that these models produce well-calibrated measures of uncertainty which is useful for practical applications. For example, we show that this uncertainty can recognise novel images that are from new environments the model has not seen before during training. This application is directly useful for addressing the loop-closure problem.

3.2 Localisation

Large scale localisation research can be divided into two categories; place recognition and metric localisation. Place recognition discretises the world into a number of landmarks and attempts to identify which place is visible in a given image. Traditionally, this has been modelled as an image retrieval problem (Chen et al., 2011; Cummins and Newman, 2008; Schindler et al., 2007; Torii et al., 2013) enabling the use of efficient and scalable retrieval approaches (Nister and Stewenius, 2006; Philbin et al., 2007) such as Bag-of-Words

(BoW) (Sivic et al., 2003), VLAD (Delhumeau et al., 2013; Jégou et al., 2010), and Fisher vectors (Jegou et al., 2012). Deep learning models have also been shown to be effective for creating efficient descriptors. Many approaches leverage classification networks (Babenko and Lempitsky, 2015; Gong et al., 2014; Razavian et al., 2014b; Tolias et al., 2016), and fine tune them on localisation datasets (Babenko et al., 2014). Other work of note is PlaNet (Weyand et al., 2016) which trained a classification network to localise images on a world scale. However, all these networks must discretise the world into places and are unable to produce a fine grained estimate of 6-DOF pose.

In contrast, metric localisation techniques estimate the metric position and orientation of the camera. Traditionally, this has been approached by computing the pose from correspondences between two-dimensional (2-D) features in the query image and three-dimensional (3-D) points in the model, which are determined through descriptor matching (Choudhary and Narayanan, 2012; Li et al., 2012, 2010; Sattler et al., 2012; Svarm et al., 2014). This assumes that the scene is represented by a 3-D structure-from-motion model. The full 6 degree-of-freedom pose of a query image can be estimated very precisely (Sattler et al., 2014). However these methods require a 3-D model with a large database of features and efficient retrieval methods. They are expensive to compute, often do not scale well, and are often not robust to changing environmental conditions (Walch et al., 2016).

In this work, we address the more challenging problem of metric localisation with deep learning. In PoseNet (Kendall et al., 2015), we introduced the technique of training a convolutional neural network to regress camera pose. It combines the strengths of place recognition and localisation approaches: it can globally relocalise without a good initial pose estimate, and produces a continuous metric pose. Rather than building a map (or database of landmark features), the neural network learns features whose size, unlike a map, does not require memory linearly proportional to the size of the scene.

Our method removes several issues faced by typical SLAM pipelines, such as the need to store densely spaced key-frames, the need to maintain separate mechanisms for appearance-based localization and landmark-based pose estimation, and a need to establish frame-to-frame feature correspondence. We do this by mapping monocular images to a high-dimensional representation that is robust to nuisance variables. We empirically show that this representation is a smoothly varying injective (one-to-one) function of pose, allowing us to regress pose directly from the image without need of tracking. The scene may include multiple objects and need not be viewed under consistent conditions. For example the scene may include dynamic objects like people and cars or experience changing weather conditions.

Later work has extended PoseNet to use RGB-D input (Li et al., 2017), learn relative ego-motion (Melekhov et al., 2017), improve the context of features (Walch et al., 2016), localise over video sequences (Clark et al., 2017) and interpret relocalisation uncertainty with Bayesian Neural Networks (Kendall and Cipolla, 2016). Additionally, (Walch et al., 2016) demonstrate PoseNet's efficacy on featureless indoor environments, where they demonstrate that SIFT based structure from motion techniques fail in the same environment.

Although PoseNet is scalable and robust (Kendall et al., 2015), it does not produce sufficiently accurate estimates of Pose compared to traditional methods (Sattler et al., 2014). It was designed with a naive regression loss function which trains the network end-to-end without any consideration for geometry. This problem is the focus of this chapter – we do not want to throw away the decades of research into multi-view geometry (Hartley and Zisserman, 2000). We improve PoseNet's performance by learning camera pose with a fundamental treatment of scene geometry.

3.3 Relocalisation Datasets

In this section, we describe the datasets we use in this chapter to demonstrate our localisation algorithms. Deep learning often requires very large datasets. However annotating ground truth labels on these datasets is often expensive or very labour intensive. We can leverage structure from motion (Snavely et al., 2008), or similar algorithms (Shotton et al., 2013), to autonomously generate training labels (camera poses) from image data (Kendall et al., 2015). We use three datasets to benchmark our approach. These datasets are summarised in Table 3.1 and example imagery is shown in Figure 3.2. We use these datasets to demonstrate our method's performance across a range of settings and scales. We endeavour to demonstrate the general applicability of the approach.

Cambridge Landmarks (Kendall et al., 2015) was collected for this thesis and is publicly released for use[2]. It provides labelled video data to train and test pose regression algorithms in an outdoor urban setting. It was collected using a smart phone and structure from motion was used to generate the pose labels (Wu, 2013). Significant urban clutter such as pedestrians and vehicles were present and data was collected from many different points in time representing different lighting and weather conditions. Train and test images are taken from distinct walking paths and not sampled from the same trajectory making the regression challenging.

7 Scenes (Shotton et al., 2013) is an indoor dataset which was collected with a Kinect RGB-D sensor. Ground truth poses were computed using Kinect Fusion (Shotton et al., 2013). The dataset contains seven scenes which were captured around an office building.

[2]The Cambridge Landmarks dataset is publicly available for download at http://mi.eng.cam.ac.uk/projects/localisation/

(a) 7 Scenes Dataset - 43,000 images from seven scenes in small indoor environments (Shotton et al., 2013).

(b) Cambridge Landmarks Dataset - over 10,000 images from six scenes around Cambridge, UK (Kendall et al., 2015).

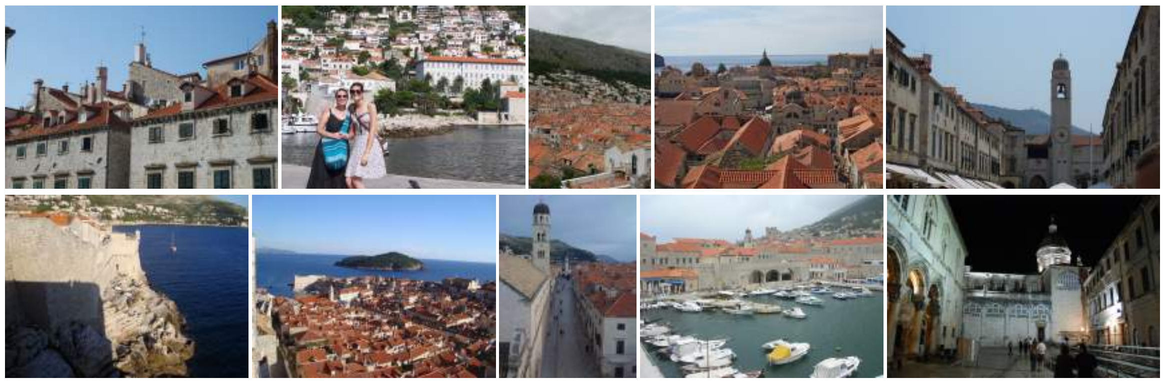

(c) Dubrovnik 6K Dataset - 6,000 images from a variety of camera types in Dubrovnik, Croatia (Li et al., 2010).

Fig. 3.2 **Example images randomly chosen from each dataset.** This illustrates the wide variety of settings and scales and the challenging array of environmental factors such as lighting, occlusion, dynamic objects and weather which are captured in each dataset.

Dataset	Type	Scale	Imagery	Scenes	Train Images	Test Images	3-D Points	Spatial Area
7 Scenes	Indoor	Room	RGB-D sensor (Kinect)	7	26,000	17,000	-	4×3m
Cambridge Landmarks	Outdoor	Street	Mobile phone camera	6	8,380	4,841	2,097,191	100×500m
Dubrovnik 6K	Outdoor	Small town	Internet images (Flikr)	1	6,044	800	2,106,456	1.5×1.5km

Table 3.1 **Summary of the localisation datasets used in this chapter's experiments.** We compare 7 Scenes (Shotton et al., 2013), Cambridge Landmarks (Kendall et al., 2015), Dubrovnik (Li et al., 2012) and the San Francisco (Chen et al., 2011) datasets. These datasets are all publicly available. They demonstrate our method's performance over a range of scales for both indoor and outdoor applications.

Each scene typically consists of a single room. The dataset was originally created for RGB-D relocalisation. It is extremely challenging for purely visual relocalisation using SIFT-like features, as it contains many ambiguous textureless features.

Dubrovnik 6K (Li et al., 2012) is a dataset consisting of 6,044 train and 800 test images which were obtained from the internet. They are taken from Dubrovnik's old town in Croatia which is a UNESCO world heritage site. The images are predominantly captured by tourists with a wide variety of camera types. Ground truth poses for this dataset were computed using structure from motion.

3.4 Model for Camera Pose Regression

In this section we describe the details of the convolutional neural network model we train to estimate camera pose directly from a monocular image, I. Our network outputs an estimate, $\hat{\mathbf{p}}$, for pose, $\mathbf{p}$, given by a 3-D camera position $\hat{\mathbf{x}}$ and orientation $\hat{\mathbf{q}}$. We use a quaternion to represent orientation, for reasons discussed in Section 3.4.2. Pose $\mathbf{p}$ is defined relative to an arbitrary global reference frame. In practice we centre this global reference frame at the mean location of all camera poses in the training dataset. We train the model with supervised learning using pose labels, $\mathbf{p} = [\mathbf{x}, \mathbf{q}]$, obtained through structure from motion, or otherwise (Section 3.3).

3.4.1 Architecture

Our pose regression formulation is capable of being applied to any neural network trained through back propagation. For the experiments in this chapter we adapt state of the art deep neural network architectures for classification, such as GoogLeNet (Szegedy et al., 2015) and ResNet (He et al., 2016), as a basis for developing our pose regression network. This allows us to use pretrained weights, such as those from a model trained to classify images in the ImageNet dataset (Deng et al., 2009). We observe that these pretrained features regularise and improve performance in PoseNet through transfer learning (Oquab et al., 2014). Although, to generalise PoseNet, we may apply it to any deep architecture designed for image classification as follows:

1. Remove the final linear regression and softmax layers used for classification
2. Append a linear regression layer. This fully connected layer is designed to output a seven dimensional pose vector representing position (3 dimensions) and orientation (4 dimensional quaternion)

3. Insert a normalisation layer to normalise the four dimensional quaternion orientation vector to unit length

3.4.2 Pose Representation

An important consideration when designing a machine learning system is the representation space of the output. We can easily learn camera position in Euclidean space (Kendall et al., 2015). However, learning orientation is more complex. In this section we compare a number of different parametrisations used to express rotational quantities; Euler angles, axis-angle, $SO(3)$ rotation matrices and quaternions (Altmann, 2005). We evaluate their efficacy for deep learning.

Firstly, Euler angles are easily understandable as an interpretable parametrisation of 3-D rotation. However, they have two problems. Euler angles wrap around at 2π radians, having multiple values representing the same angle. Therefore they are not injective, which causes them to be challenging to learn as a uni-modal scalar regression task. Additionally, they do not provide a unique parametrisation for a given angle and suffer from the well-studied problem of gimbal lock (Altmann, 2005). The axis-angle representation is another three dimensional vector representation. However like Euler angles, it too suffers from a repetition around the 2π radians representation.

Rotation matrices are a over-parametrised representation of rotation. For 3-D problems, the set of rotation matrices are 3×3 dimensional members of the special orthogonal Lie group, $SO(3)$. These matrices have a number of interesting properties, including orthonormality. However, it is difficult to enforce the orthogonality constraint when learning a $SO(3)$ representation through back-propagation.

In this work, we chose quaternions as our orientation representation. Quaternions are favourable because arbitrary four dimensional values are easily mapped to legitimate rotations by normalizing them to unit length. This is a simpler process than the orthonormalisation required of rotation matrices. Quaternions are a continuous and smooth representation of rotation. They lie on the unit manifold, which is a simple constraint to enforce through back-propagation. Their main downside is that they have two mappings for each rotation, one on each hemisphere. However, in Section 3.4.3 we show how to adjust the loss function to compensate for this.

3.4.3 Loss Function

This far, we have described the structure of the pose representation we would like our network to learn. Next, we discuss how to design an effective loss function to learn to estimate the camera's 6 degree of freedom pose. This is a particularly challenging objective because it involves learning two distinct quantities - rotation and translation - with different units and scales.

This section defines a number of loss functions and explores their efficacy for camera pose regression. We begin in Section 3.4.3 by describing a basic weighted loss function which we proposed in (Kendall et al., 2015). We improve on this in Section 3.4.3 by introducing a novel loss function which can learn the weighting between rotation and translation automatically, using an estimate of the *homoscedastic* task uncertainty. Further, in Section 3.4.3 we describe a loss function which combines position and orientation as a single scalar using the reprojection error geometry. In Section 3.5.1 we compare the performance of these loss functions, and discusses their trade-offs.

Learning Position and Orientation

We can learn to estimate camera position by forming a smooth, continuous and injective regression loss in Euclidean space, $\mathscr{L}_x(I) = \|\mathbf{x} - \hat{\mathbf{x}}\|_\gamma$, with norm given by γ. In Kendall et al. (2015) we used the L_2 Euclidean norm.

However, learning camera orientation is not as simple. In Section 3.4.2 we described a number of options for representing orientation. Quaternions are an attractive choice for deep learning because they are easily formulated in a continuous and differentiable way. The set of rotations lives on the unit sphere in quaternion space. We can easily map any four dimensional vector to a valid quaternion rotation by normalising it to unit length. Kendall et al. (2015) demonstrates how to learn to regress quaternion values:

$$\mathscr{L}_q(I) = \left\| \mathbf{q} - \frac{\hat{\mathbf{q}}}{\|\hat{\mathbf{q}}\|} \right\|_\gamma \tag{3.1}$$

Using a distance norm, γ, in Euclidean space makes no effort to keep $\mathbf{q}$ on the unit sphere. We find, however, that during training, $\mathbf{q}$ becomes close enough to $\hat{\mathbf{q}}$ such that the distinction between spherical distance and Euclidean distance becomes insignificant. For simplicity, and to avoid hampering the optimization with unnecessary constraints, we chose to omit the spherical constraint. The main problem with quaternions is that they are not injective because they have two unique values (from each hemisphere) which map to a single rotation. This is because quaternion, $\mathbf{q}$, is identical to $-\mathbf{q}$. To address this, we constrain all quaternions to one hemisphere such that there is a unique value for each rotation.

Simultaneously Learning Position and Orientation

The challenging aspect of learning camera pose is designing a loss function which is able to learn both position and orientation. Initially, we proposed a method to combine position and orientation into a single loss function with a linear weighted sum (Kendall et al., 2015), shown in (3.2):

$$\mathscr{L}_\beta(I) = \mathscr{L}_x(I) + \beta \mathscr{L}_q(I) \tag{3.2}$$

Because **x** and **q** are expressed in different units, a scaling factor, β, is used to balance the losses. This hyper-parameter attempts to keep the expected value of position and orientation errors approximately equal.

Interestingly, we observe that a model which is jointly trained to regress the camera's position and orientation performs better than separate models trained on each task individually. Figure 3.3 shows that with just position, or just orientation information, the network was not able to determine the function representing camera pose with as great accuracy. The model learns a better representation for pose when supervised with both translation and orientation labels. We also experimented with branching the network lower down into two separate components to regress position and orientation. However, we found that it too was less effective, for similar reasons: separating into distinct position and orientation features denies each the information necessary to factor out orientation from position, or vice versa.

However the consequence of this was that the hyper-parameter β required significant tuning to get reasonable results. In the loss function (3.2) a balance β must be struck between the orientation and translation penalties (Figure 3.3). They are highly coupled as they are regressed from the same model weights. We found β to be greater for outdoor scenes as position errors tended to be relatively greater. Following this intuition it is possible to fine tune β using grid search. For the indoor scenes it was between 120 to 750 and outdoor scenes between 250 to 2000. This is an expensive task in practice, as each experiment can take days to complete. It is desirable to find a loss function which removes this hyperparameter. Therefore, the remainder of this section explores different loss functions which aim to find an optimal weighting automatically.

Learning an Optimal Weighting

Ideally, we would like a loss function which is able to learn position and orientation optimally, without including any hyper parameters. For this reason, we use the multi-task loss function from Section 2.5, which is able to learn a weighting between the position and orientation objective functions. We formulate it using *homoscedastic uncertainty* which we can learn using probabilistic deep learning (as described in Section 2.5). Homoscedastic uncertainty is a

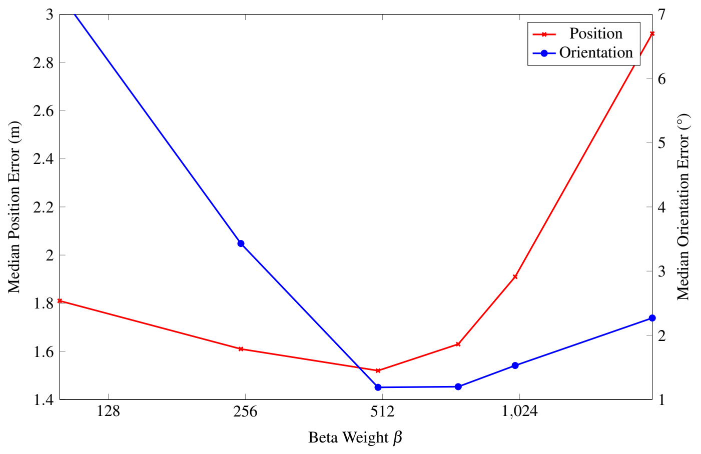


Fig. 3.3 Relative performance of position and orientation regression on **a single model with a range of scale factors** for an indoor scene from the King's College scene in Cambridge Landmarks, using the loss function in (3.2). This demonstrates that learning with the optimum scale factor leads to the network uncovering a more accurate pose function.

measure of uncertainty which does not depend on the input data, as opposed to heteroscedastic uncertainty which is a function of the input data (see Section 2.4). Rather, it captures the uncertainty of the task itself. In Section 2.5 we show how to use this insight to combine losses for different tasks in a probabilistic manner. Here we show how to apply this to learn camera position and orientation (with a Gaussian likelihood):

$$\mathcal{L}_\sigma(I) = \mathcal{L}_x(I)\hat{\sigma}_x^{-2} + \log\hat{\sigma}_x^2 + \mathcal{L}_q(I)\hat{\sigma}_q^{-2} + \log\hat{\sigma}_q^2 \tag{3.3}$$

where we optimise the homoscedastic uncertainties, $\hat{\sigma}_x^2$, $\hat{\sigma}_q^2$, through backpropagation with respect to the loss function. These uncertainties are free scalar values, not model outputs. They represent the homoscedastic (task) noise.

This loss consists of two components; the residual regressions and the uncertainty regularization terms. We learn the variance, σ^2, implicitly from the loss function. As the variance is larger, it has a tempering effect on the residual regression term; larger variances (or uncertainty) results in a smaller residual loss. The second regularization term prevents the network from predicting infinite uncertainty (and therefore zero loss). As we expect quaternion values to have much smaller values (they are constrained to the unit manifold), their noise, σ_q^2 should be much smaller than the position noise, σ_x^2, which can be many meters in magnitude. As σ_q^2 should be much smaller than σ_x^2, orientation regression should be weighted much higher than position – with a similar effect to β in (3.2).

In practice, we learn $\hat{s} := \log \hat{\sigma}^2$,:

$$\mathscr{L}_\sigma(I) = \mathscr{L}_x(I)\exp(-\hat{s}_x) + \hat{s}_x + \mathscr{L}_q(I)\exp(-\hat{s}_q) + \hat{s}_q. \tag{3.4}$$

This is more stable than regressing the variance, σ^2, because the loss avoids a potential division by zero. The exponential mapping also allows us to regress unconstrained scalar values, where $\exp(-s_i)$ is resolved to the positive domain giving valid values for variance. We find that this loss is very robust to our initialisation choice for the homoscedastic task uncertainty values. Only an approximate initial guess is required, we arbitrarily use initial values of $\hat{s}_x = 0.0, \hat{s}_q = -3.0$, for all scenes.

Learning from Geometric Reprojection Error

Perhaps a more desirable loss is one that does not require balancing of rotational and positional quantities at all. Reprojection error of scene geometry is a representation which combines rotation and translation naturally in a single scalar loss (Hartley and Zisserman, 2000). Reprojection error is given by the residual between 3-D points in the scene projected onto a 2-D image plane using the ground truth and predicted camera pose. It therefore converts rotation and translation quantities into image coordinates. This naturally weights translation and rotation quantities depending on the scene and camera geometry.

All losses in this chapter so far assume images and pose labels are provided. In this section, we formulate a reprojection error loss that assumes we have images and the corresponding 3-D points from the scene geometry. First, we define a function, π, which maps a 3-D point, $\mathbf{g}$, to 2-D image coordinates, $(u, v)^T$:

$$\pi(\mathbf{x}, \mathbf{q}, \mathbf{g}) \mapsto \begin{pmatrix} u \\ v \end{pmatrix} \tag{3.5}$$

where $\mathbf{x}$ and $\mathbf{q}$ represent the camera position and orientation. This function, π, is defined as:

$$\begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} = \mathsf{K}(\mathsf{R}\mathbf{g} + \mathbf{x}), \quad \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} u'/w' \\ v'/w' \end{pmatrix} \tag{3.6}$$

where K is the intrinsic calibration matrix of the camera, and R is the mapping of $\mathbf{q}$ to its $SO(3)$ rotation matrix, $\mathbf{q}_{4\times1} \mapsto \mathsf{R}_{3\times3}$.

We formulate this loss by taking the norm of the reprojection error between the predicted and ground truth camera pose. We take the subset, $\mathscr{G}'$, of all 3-D points in the scene, $\mathscr{G}$, which are visible in the image I. The final loss (3.7) is given by the mean of all the residuals from points, $g_i \in \mathscr{G}'$:

$$\mathscr{L}_g(I) = \frac{1}{|\mathscr{G}'|} \sum_{g_i \in \mathscr{G}'} \|\pi(\mathbf{x}, \mathbf{q}, \mathbf{g_i}) - \pi(\hat{\mathbf{x}}, \hat{\mathbf{q}}, \mathbf{g_i})\|_\gamma \tag{3.7}$$

where $\hat{\mathbf{x}}$ and $\hat{\mathbf{q}}$ are the predicted camera poses from PoseNet, with $\mathbf{x}$ and $\mathbf{q}$ the ground truth label, with norm, γ, which is discussed in Section 3.4.3.

Note that because we are projecting 3-D points using both the ground truth and predicted camera pose we can apply any arbitrary camera model, as long as we use the same intrinsic parameters for both cameras. Therefore for simplicity, we set the camera intrinsics, K, to the identity matrix – camera calibration is not required.

This loss implicitly combines rotation and translational quantities into image coordinates. Minimising reprojection error is often the most desirable balance between these quantities for many applications, such as augmented reality. The key advantage of this loss is that it allows the model to vary the weighting between position and orientation, depending on the specific geometry in the training image. For example, training images with geometry which is far away would balance rotational and translational loss differently to images with geometry very close to the camera.

Interestingly, when experimenting with the original weighted loss in (3.2) we observed that the hyperparameter β was an approximate function of the scene geometry. We observed that it was a function of the landmark distance and size in the scene. Our intuition was that the optimal choice for β was approximating the reprojection error in the scene geometry. For example, if the scene is very far away, then rotation is more significant than translation and vice versa. This function is not trivial to model for complex scenes with a large number of landmarks. It will vary significantly with each training example in the dataset. By learning with reprojection error we can use our knowledge of the scene geometry more directly to automatically infer this weighting.

Projecting geometry through a projection model is a differentiable operation involving matrix multiplication. Therefore we can use this loss to train our model with stochastic gradient descent. It is important to note that we do not need to know the intrinsic camera parameters to project this 3-D geometry. This is because we apply the same projection to both the model prediction and ground truth measurement, so we can use arbitrary values.

A practical consideration, when training with stochastic gradient descent, is that it is desirable to have uniform tensors for each data-point in a mini-batch, in order to parallelise computation. Therefore, at training time we sample a random 100 points from the subset of the scene's geometry which is visible by the camera's view. This ensures a uniformly shaped

Loss function	Median Error x [m]	q [°]	Accuracy $< 2m, 5°$ [%]
Linear sum, $\beta = 500$ (3.2)	1.52	1.19	65.0%
Learn weighting with homoscedastic uncertainty (3.3)	0.99	1.06	85.3%
Reprojection loss	does not converge		
Learn weighting pretraining $\mapsto$ Reprojection loss (3.7)	0.88	1.04	90.3%

(a) Cambridge Landmarks, King's College

Loss function	Median Error x [m]	q [°]	Accuracy $< 2m, 5°$ [%]
Linear sum, $\beta = 500$ (3.2)	13.1	4.68	30.1%
Learn weighting with homoscedastic uncertainty (3.3)	9.88	4.73	41.7%
Reprojection loss	does not converge		
Learn weighting pretraining $\mapsto$ Reprojection loss (3.7)	7.90	4.40	48.6%

(b) Dubrovnik 6K

Table 3.2 **Comparison of different loss functions.** We use an L_1 distance for the residuals in each loss. *Linear sum* combines position and orientation losses with a constant scaling parameter β (Kendall and Cipolla, 2016) and is defined in (3.2). Learn weighting is the loss function in (3.3) which learns to combine position and orientation using homoscedastic uncertainty. Reprojection error implicitly combines rotation and translation by using the reprojection error of the scene geometry as the loss (3.7). We find that homoscedastic uncertainty is able to learn an effective weighting between position and orientation quantities. The reprojection loss was not able to converge from random initialisation. However, when used to fine-tune a network pretrained with (3.3) it yields the best results.

mini-batch, while having enough points to sample. If less than 100 points are visible, then we repeat the points until 100.

It should be noted that we need to have some knowledge of the scene's geometry in order to have 3-D points to reproject. The geometry is often known; if our data is obtained through structure from motion, RGBD data or other sensory data (see Section 3.3). Only points from the scene which are visible in the image I are used to compute the loss. We also found it was important for numerical stability to ignore points which are projected outside the image bounds.

Regression Norm

An important choice for these losses is the regression norm, $\| \, \|_\gamma$. Typically, deep learning models use an $L_1 = \| \, \|_1$ or $L_2 = \| \, \|_2$. We can also use robust norms such as Huber's loss (Huber, 2011) and Tukey's loss (Hoaglin et al., 1983), which have been successfully applied to deep learning (Belagiannis et al., 2015). For camera pose regression, we found that they negatively impacted performance by over-attenuating difficult examples. We suspect that for more noisy datasets these robust regression functions might be beneficial. With the datasets used in this chapter, we found the L_1 norm to perform best and therefore use $\gamma = 1$. It does not increase quadratically with magnitude or over-attenuate large residuals.

3.5 Localisation Experiments

To train and benchmark our model on a number of datasets we rescale the input images such that the shortest side is of length 256. We normalise the images so that input pixel intensities range from -1 to 1. We train our PoseNet architecture using an implementation in TensorFlow (Abadi et al., 2016). All models are optimised end-to-end with ADAM (Kingma and Ba, 2014) using the default parameters and a learning rate of 1×10^{-4}. We train each model until the training loss converges. We use a batch size of 64 on a NVIDIA Titan X (Pascal) GPU, training takes approximately 20k - 100k iterations, or 4 hours - 1 day.

3.5.1 Comparison of Loss Functions

In Table 3.2 we compare different combinations of losses and regression norms. We compare results for a scene in the Cambridge Landmarks dataset (Kendall et al., 2015) and the Dubrovnik 6K dataset (Li et al., 2012), which has imagery from a range of cameras.

We find that modelling homoscedastic uncertainty with the loss in (3.3) is able to effectively learn a weighting between position and orientation. It outperforms the constant weighting used in loss (3.2). The reprojection loss in (3.7) is unable to train the model from a random initialisation. We observe that the model gets stuck in a poor local minima, when using any of the regression norms. However, the reprojection loss is able to improve localisation performance when using weights pretrained with any of the other losses. For example, we can take the best performing model using the loss from (3.3) and fine tune with the reprojection loss (3.7). We observe that this loss is then able to converge effectively. This shows that the reprojection loss is not robust to large residuals. This is because reprojected points can be easily placed far from the image centre if the network makes a poor pose prediction. Therefore, we recommend the following two-step training scheme:

1. Train the model using the loss in (3.3), learning the weighting between position and orientation.

2. If the scene geometry is known (for example from structure from motion or RGBD camera data) then fine-tune the model using the reprojection loss in (3.7).

Scene	Area or Volume	Active Search (SIFT) (Sattler et al., 2016)	PoseNet (β weight) (Kendall et al., 2015)	Bayesian PoseNet (Kendall and Cipolla, 2016)	PoseNet Spatial LSTM (Walch et al., 2016)	PoseNet Learn σ^2 Weight	PoseNet Geometric Reprojection
Great Court	$8000m^2$	–	–	–	–	7.00m, 3.65°	6.83m, 3.47°
King's College	$5600m^2$	0.42m, 0.55°	1.66m, 4.86°	1.74m, 4.06°	0.99m, 3.65°	0.99m, 1.06°	0.88m, 1.04°
Old Hospital	$2000m^2$	0.44m, 1.01°	2.62m, 4.90°	2.57m, 5.14°	1.51m, 4.29°	2.17m, 2.94°	3.20m, 3.29°
Shop Façade	$875m^2$	0.12m, 0.40°	1.41m, 7.18°	1.25m, 7.54°	1.18m, 7.44°	1.05m, 3.97°	0.88m, 3.78°
St Mary's Church	$4800m^2$	0.19m, 0.54°	2.45m, 7.96°	2.11m, 8.38°	1.52m, 6.68°	1.49m, 3.43°	1.57m, 3.32°
Street	$50000m^2$	0.85m, 0.83°	–	–	–	20.7m, 25.7°	20.3m, 25.5°
Chess	$6m^3$	0.04m, 1.96°	0.32m, 6.60°	0.37m, 7.24°	0.24m, 5.77°	0.14m, 4.50°	0.13m, 4.48°
Fire	$2.5m^3$	0.03m, 1.53°	0.47m, 14.0°	0.43m, 13.7°	0.34m, 11.9°	0.27m, 11.8°	0.27m, 11.3°
Heads	$1m^3$	0.02m, 1.45°	0.30m, 12.2°	0.31m, 12.0°	0.21m, 13.7°	0.18m, 12.1°	0.17m, 13.0°
Office	$7.5m^3$	0.09m, 3.61°	0.48m, 7.24°	0.48m, 8.04°	0.30m, 8.08°	0.20m, 5.77°	0.19m, 5.55°
Pumpkin	$5m^3$	0.08m, 3.10°	0.49m, 8.12°	0.61m, 7.08°	0.33m, 7.00°	0.25m, 4.82°	0.26m, 4.75°
Red Kitchen	$18m^3$	0.07m, 3.37°	0.58m, 8.34°	0.58m, 7.54°	0.37m, 8.83°	0.24m, 5.52°	0.23m, 5.35°
Stairs	$7.5m^3$	0.03m, 2.22°	0.48m, 13.1°	0.48m, 13.1°	0.40m, 13.7°	0.37m, 10.6°	0.35m, 12.4°

Table 3.3 **Median localisation results for the *Cambridge Landmarks* (Kendall et al., 2015) and *7 Scenes* (Shotton et al., 2013) datasets.** We compare the performance of various RGB-only algorithms. Active search (Sattler et al., 2016) is a state-of-the-art traditional SIFT keypoint based baseline. We demonstrate a notable improvement over PoseNet's (Kendall et al., 2015) baseline performance using the learned σ^2 and reprojection error proposed in this chapter, narrowing the margin to the state of the art SIFT technique.

3.5.2 Benchmarking Localisation Accuracy

In Table 3.3 we show that our geometry based loss outperforms the original PoseNet's naive loss function (Kendall et al., 2015). We observe a consistent and significant improvement across both indoor *7 Scenes* outdoor *Cambridge Landmarks* datasets. We conclude that we can simultaneously learn both position and orientation more effectively by considering scene geometry. The improvement is notably more pronounced for the 7Scenes dataset. We believe this is due to the significantly larger amount of training data for each scene in this dataset, compared with Cambridge Landmarks. We also outperform the improved PoseNet architecture with spatial LSTMs (Walch et al., 2016). However, this method is complimentary to the loss functions in this chapter, and it would be interesting to explore the union of these ideas.

We observe a difference in relative performance between position and orientation when optimising with respect to reprojection error (3.7) or homoscedastic uncertainty (3.3). Overall, optimising reprojection loss improves rotation accuracy, sometimes at the expense of some positional precision.

3.5.3 Comparison to SIFT-Feature Approaches

Table 3.3 also compares to a state-of-the-art traditional SIFT feature based localisation algorithm, Active Search (Sattler et al., 2016). This method outperforms PoseNet, and is effective in feature-rich outdoor environments. However, in the 7Scenes dataset this deficit is less pronounced. The indoor scenes contain much fewer point features and there is significantly more training data. As an explanation for the deficit in these results, PoseNet only uses 256×256 pixel images, while SIFT based methods require images of a few mega-pixels in size (Sattler et al., 2016). Additionally, PoseNet is able to localise an image in $5ms$, scaling constantly with scene area, while traditional SIFT feature approaches require over $100ms$, and scale with scene size (Sattler et al., 2016).

In Table 3.4 we compare our approach on the Dubrovnik dataset to other geometric techniques which localise by registering SIFT features (Lowe, 2004) to a large 3-D model (Li et al., 2012). Although our method improves significantly over the original PoseNet model, it is still yet to reach the fine grained accuracy of these methods (Li et al., 2010; Sattler et al., 2011; Svarm et al., 2014; Zeisl et al., 2015). We hypothesise that this is due to a lack of training data, with only 6k images across the town. However, our algorithm is significantly faster than these approaches. Furthermore, it is worth noting that PoseNet only sees a 256×256 resolution image, while these methods register the full size images, often with a few million pixels.

	Position		Orientation	
Method	Mean [m]	Median [m]	Mean [°]	Median [°]
PoseNet (this work)	40.0	7.9	11.2	4.4
APE (Svarm et al., 2014)	-	0.56	-	-
Voting (Zeisl et al., 2015)	-	1.69	-	-
Sattler, et al. (Sattler et al., 2011)	14.9	1.3	-	-
P2F (Li et al., 2010)	18.3	9.3	-	-

Table 3.4 **Localisation results on the Dubrovnik dataset** (Li et al., 2012), comparing to a number of state-of-the-art point-feature techniques. Our method is the first deep learning approach to benchmark on this challenging dataset. We achieve comparable performance, while our method only requires a 256×256 pixel image and is much faster to compute.

We show that PoseNet is able to effectively localize across both the indoor *7 Scenes* dataset and outdoor *Cambridge Landmarks* dataset in Table 3.3. To validate that the model is regressing pose beyond that of the training examples we show the performance for finding the nearest neighbour representation in the training data from the feature vector produced by the localisation network. As our performance exceeds this we conclude that the network is successfully able to regress pose beyond training examples (see Figure 3.4). We also compare our algorithm to the RGB-D SCoRe Forest algorithm (Shotton et al., 2013).

Figure 3.7 shows cumulative histograms of localisation error for two indoor and two outdoor scenes. We note that although the SCoRe forest is generally more accurate, it requires depth information, and uses higher-resolution imagery. The indoor dataset contains many ambiguous and textureless features which make relocalisation without this depth modality extremely difficult. We note our method often localizes the most difficult testing frames, above the 95th percentile, more accurately than SCoRe across all the scenes. We also observe that dense cropping only gives a modest improvement in performance. It is most important in scenes with significant clutter like pedestrians and cars, for example King's College, Shop Façade and St Mary's Church.

We explored the robustness of this method beyond what was tested in the dataset with additional images from dusk, rain, fog, night and with motion blur and different cameras with unknown intrinsics. Figure 3.6 shows the convolutional neural network generally handles these challenges well. SfM with SIFT fails in all these cases so we were not able to generate a ground truth camera pose, however we infer the accuracy by viewing the 3-D reconstruction from the predicted camera pose, and overlaying this onto the input image.

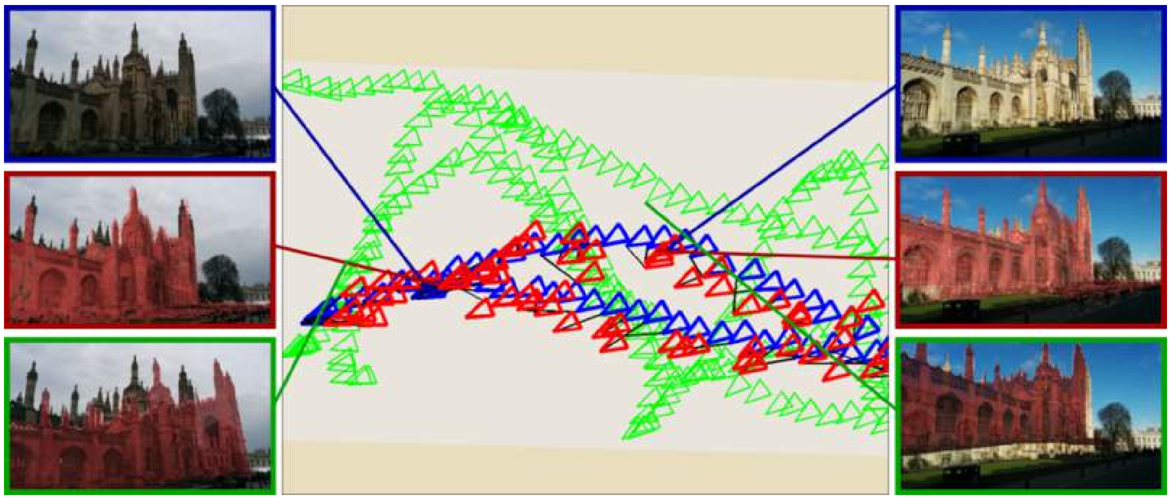


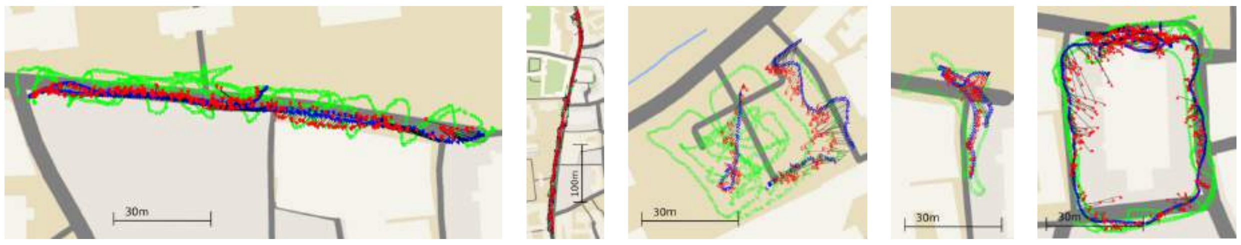
Fig. 3.4 Magnified view of a sequence of **training (green)** and **testing (blue)** cameras for King's College. We show the **predicted camera pose in red** for each testing frame. The images show the test image (top), the predicted view from our model overlaid in red on the input image (middle) and the nearest neighbour training image overlaid in red on the input image (bottom). This shows our system can interpolate camera pose effectively in space between training frames.

3.5.4 Robustness Against Training Image Spacing

We demonstrate in Figure 3.8 that, for an outdoor scale scene, we gain little by spacing the training images more closely than 4m. The system is robust to very large spatial separation between training images, achieving reasonable performance even with only a few dozen training samples. The pose accuracy deteriorates gracefully with increased training image spacing, whereas SIFT-based SfM sharply fails after a certain threshold as it requires a small baseline (Lowe, 2004).

3.5.5 Importance of Transfer Learning

In general deep networks require large amounts of training data. We sidestep this problem by starting our pose training from a network pretrained on giant datasets such as *ImageNet* and *Places*. Similar to what has been demonstrated for classification tasks, Figure 3.9 shows how transfer learning can be utilised effectively between classification and complicated regression tasks. Such 'transfer learning' has been demonstrated elsewhere for training classifiers (Bengio et al., 2013; Oquab et al., 2014; Razavian et al., 2014a), but here we demonstrate transfer learning from classification to the qualitatively different task of pose regression. It is not immediately obvious that a network trained to output pose-invariant classification labels would be suitable as a starting point for a pose regressor. We find, however, that this



King's College *Street* *Old Hospital* *Shop Façade* *St Mary's Church*

Fig. 3.5 **Map of dataset** showing training frames (green), testing frames (blue) and their predicted camera pose (red). The testing sequences are distinct trajectories from the training sequences and each scene covers a very large spatial extent.

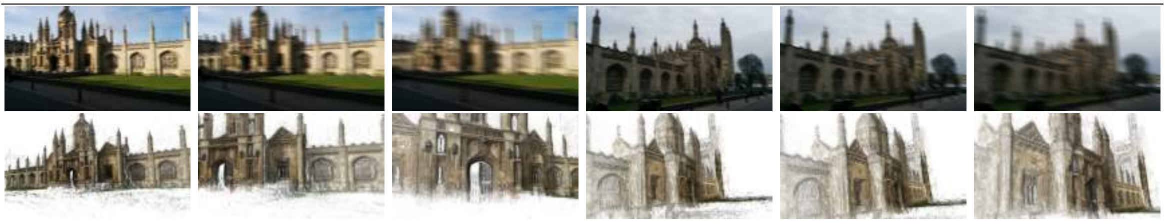
is not a problem in practice. A possible explanation is that, in order for its output to be invariant to pose, the classifier network must keep track of pose, to better factor its effects away from identity cues. This would agree with our own findings that a network trained to output position and orientation outperforms a network trained to output only position. By preserving orientation information in the intermediate representations, it is better able to factor the effects of orientation out of the final position estimation. Transfer learning gives not only a large improvement in training speed, but also end performance.

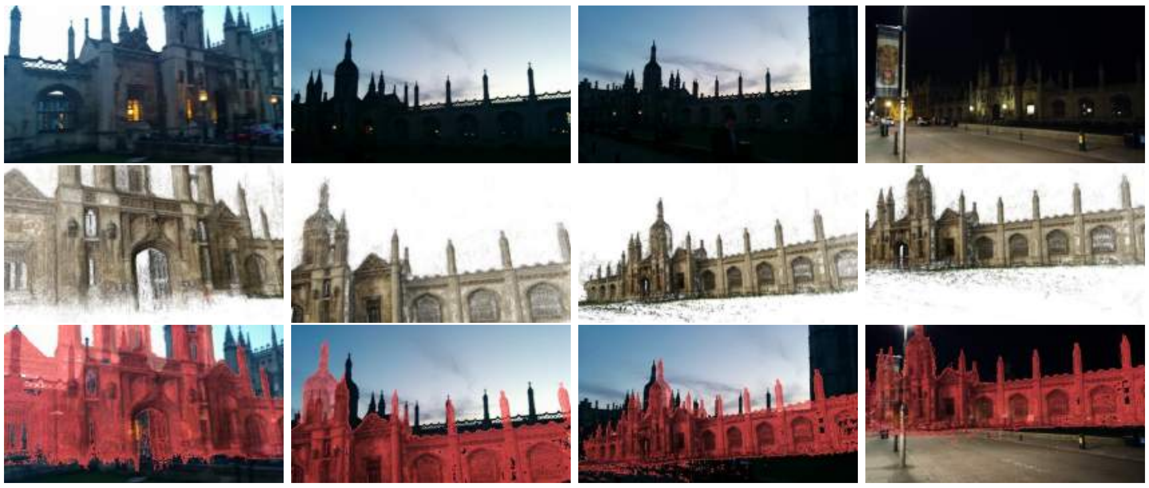
The relevance of data is also important. In Figure 3.9 the *Places* and *ImageNet* curves initially have the same performance. However, ultimately the *Places* pretraining performs better due to being a more relevant dataset to this localisation task.

3.5.6 Visualising Features Relevant to Pose

Figure 3.10 shows example saliency maps produced by PoseNet. The saliency map, as used in (Simonyan et al., 2013), is the magnitude of the gradient of the loss function with respect to the pixel intensities. This uses the sensitivity of the pose with respect to the pixels as an indicator of how important the network considers different parts of the image.

These results show that the strongest response is observed from higher-level features such as windows and spires. However a more surprising result is that PoseNet is also very sensitive to large textureless patches such as road, grass and sky. These textureless patches may be more informative than the highest responding points because the effect of a group of pixels on the pose variable is the sum of the saliency map values over that group of pixels. This evidence points to the net being able to localize off information from these textureless surfaces, something which interest-point based features such as SIFT or SURF fail to do.


(a) Relocalisation with increasing levels of motion blur. The system is able to recognize the pose as high level features such as the contour outline still exist. Blurring the landmark increases apparent contour size and the system believes it is closer.


(b) Relocalisation under difficult dusk and night lighting conditions. In the dusk sequences, the landmark is silhouetted against the backdrop however again the model seems to recognize the contours and estimate pose.

(c) Relocalisation with different weather conditions. PoseNet is able to effectively estimate pose in fog and rain.

(d) Relocalisation with significant people, vehicles and other dynamic objects.

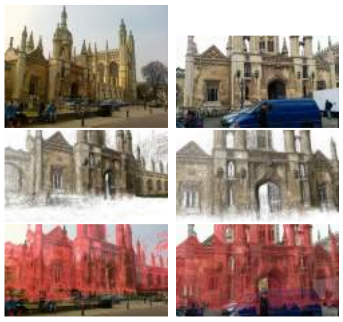

(e) Relocalisation with unknown camera intrinsics: SLR with focal length 45mm (left), and iPhone 4S with focal length 35mm (right) compared to the dataset's camera which had a focal length of 30mm.

Fig. 3.6 **Robustness to challenging real life situations.** Registration with point based techniques such as SIFT fails in examples (a-c), therefore ground truth measurements are not available. None of these types of challenges were seen during training. As convolutional neural networks are able to understand objects and contours they are still successful at estimating pose from the building's contour in the silhouetted examples (b) or even under extreme motion blur (a). Many of these quasi invariances were enhanced by pretraining from the scenes dataset.

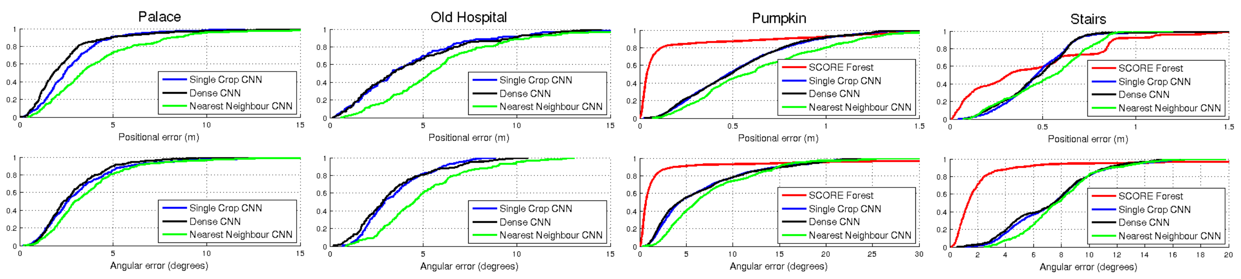


Fig. 3.7 **Localization performance.** These figures show our localization accuracy for both position and orientation as a cumulative histogram of errors for the entire training set. The regression network outperforms the nearest neighbour feature matching which demonstrates we regress finer resolution results than given by training. Comparing to the RGB-D SCoRe Forest approach shows that our method is competitive, but outperformed by a more expensive depth approach. Our method does perform better on the hardest few frames, above the 95th percentile, with our worst error lower than the worst error from the SCoRe approach.

The last observation is that PoseNet has an attenuated response to people and other noisy objects, effectively masking them. These objects are dynamic, and the model has identified them as not appropriate for localisation.

3.5.7 Viewing the Internal Representation

t-SNE (Van der Maaten and Hinton, 2008) is an algorithm for embedding high-dimensional data in a low dimensional space, in a way that tries to preserve Euclidean distances. It is often used, as we do here, to visualize high-dimensional feature vectors in two dimensions. In Figure 3.11 we apply t-SNE to the feature vectors computed from a sequence of video frames taken by a pedestrian. As these figures show, the feature vectors are a function that smoothly varies with, and is largely one-to-one with, pose. This ‘pose manifold’ can be observed not only on networks trained on other scenes, but also networks trained on classification image sets without pose labels. This further suggests that classification networks preserve pose information up to the final layer, regardless of whether it’s expressed in the output. However, the mapping from feature vector to pose becomes more complicated for networks not trained on pose data. Furthermore, as this manifold exists on scenes that the model was not trained on, the model must learn some generic representation of the relationship between landmarks, geometry and camera motion. This demonstrates that the feature vector that is produced from regression is able to generalize to other tasks in the same way as classification networks.

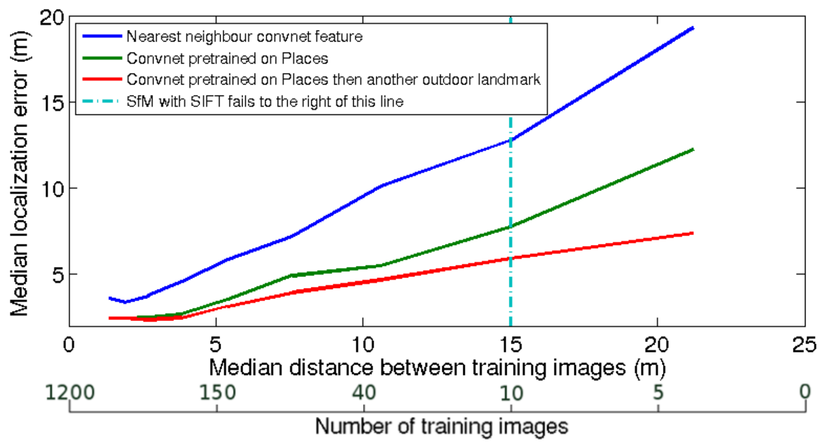


Fig. 3.8 **Robustness to a decreasing training baseline** for the King's College scene. Our system exhibits graceful decline in performance as fewer training samples are used.

3.5.8 System Efficiency

Figure 3.12 compares system performance of PoseNet on a modern desktop computer. Our network is very scalable, as it only takes 50 MB to store the weights, and $5ms$ to compute each pose, compared to the gigabytes and minutes for metric localisation with SIFT. These values are independent of the number of training samples in the system while metric localisation scales $\mathcal{O}(n^2)$ with training data size (Wu, 2013). For comparison, matching to the network's nearest neighbour is also shown. This requires storing feature vectors for each training frame, then perform a linear search to find the nearest neighbour for a given test frame.

3.6 Localisation Uncertainty

In this section, we extend the PoseNet framework to a Bayesian deep learning model which is able to determine the uncertainty of localisation using the ideas from Section 2.4. Our Bayesian convolutional neural network requires no additional memory, and can relocalise in under 50ms per frame on a GPU. By leveraging this probabilistic approach, we achieve 10 - 15% improvement on state of the art performance on *Cambridge Landmarks*, a vary large urban relocalisation dataset, and *7 Scenes*, a challenging indoor relocalisation dataset. Furthermore, our approach qualitatively improves the system by producing a measure of model uncertainty. We leverage this uncertainty value to estimate:

- metric relocalisation error for both position and orientation,

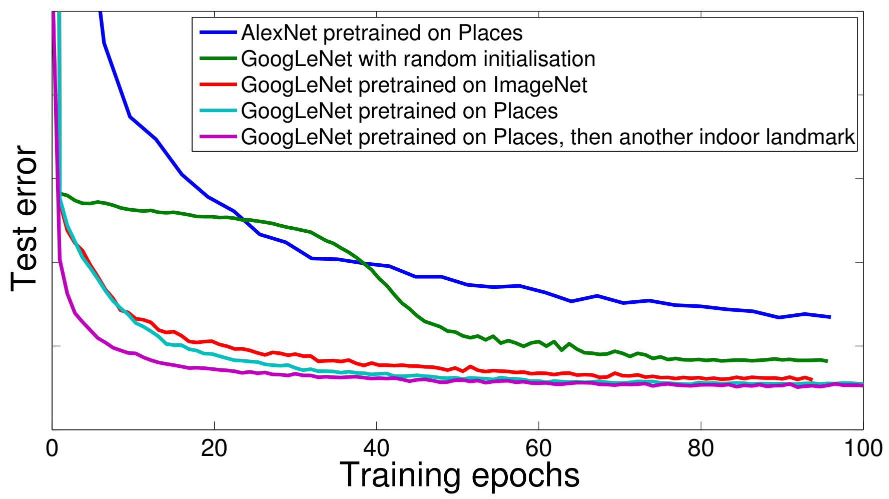


Fig. 3.9 **Importance of transfer learning.** Shows how pretraining on large datasets gives an increase in both performance and training speed.

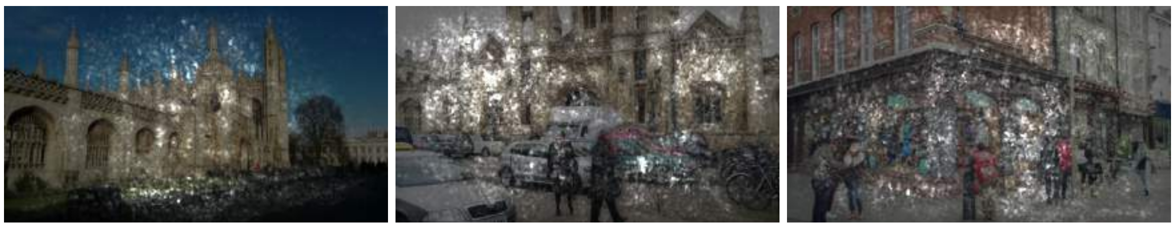


Fig. 3.10 **Saliency maps.** This figure shows the saliency map superimposed on the input image. The saliency maps suggest that the convolutional neural network exploits not only distinctive point features (à la SIFT), but also large textureless patches, which can be as informative, if not more so, to the pose. This, combined with a tendency to disregard dynamic objects such as pedestrians, enables it to perform well under challenging circumstances. (Best viewed electronically.)

- the confidence of modelling the data (detect if the scene is actually present in the input image).

Understanding model uncertainty is an important feature for a localisation system. A non-Bayesian system which outputs point estimates does not interpret if the model is making sensible predictions or just guessing at random. By measuring uncertainty we can understand with what confidence we can trust the prediction.

Secondly, it is easy to imagine visually similar landmarks and we need to be able to understand with what confidence can we trust the result. For example, there are many examples of buildings with visually ambiguous structures, such as window, which are tessellated along a wall.

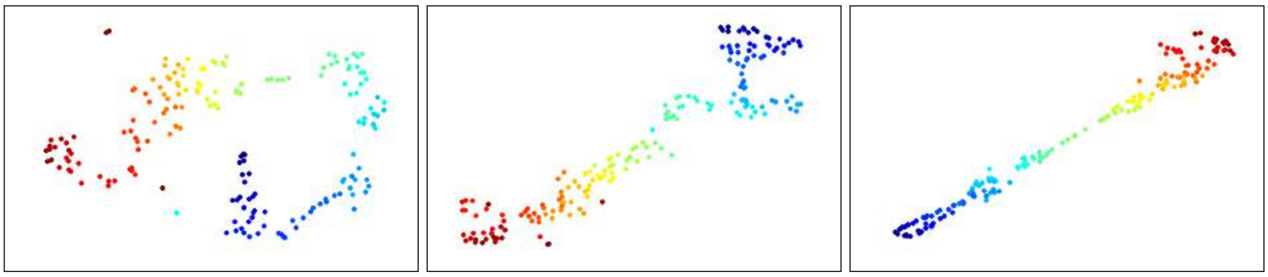


Fig. 3.11 **Feature vector visualisation.** t-SNE visualisation of the feature vectors from a video sequence traversing an outdoor scene (King's College) in a straight line. Colour represents time. The feature representations are generated from the model with weights trained on *Places* (a), *Places* then another outdoor scene, St Mary's Church (b), *Places* then this outdoor scene, King's College (c). Despite (a,b) not being trained on this scene, these visualizations suggest that it is possible to compute the pose as a simple, if non-linear, function of these representations.

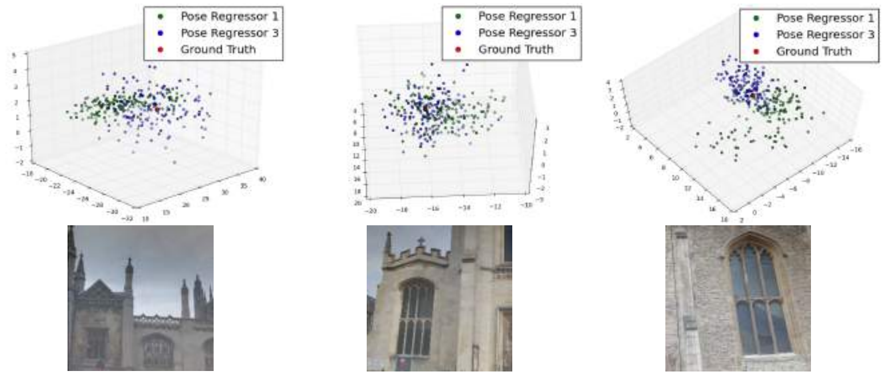
Fig. 3.12 **Implementation efficiency.** Experimental speed and memory use of the regression network, nearest neighbour network feature vector baseline and SIFT relocalisation methods.

Following Chapter 2, we conclude that it is more important to model *epistemic* uncertainty for localisation. This is because training data is often sparse. Furthermore, it is critical to detect if the input image is one from the scene which the model is trained to localise with. This is also known as *loop-closure*. It requires a strong ability to detect novel input images which are outside the training data distribution. In Section 2.4 we show that this is something aleatoric uncertainty cannot model, therefore in this section we focus on modelling epistemic uncertainty.

3.6.1 Modelling Localisation Uncertainty

Although many of the modern SLAM algorithms do not consider localisation uncertainty (Engel et al., 2014; Klein and Murray, 2007; Li et al., 2012), previous probabilistic algorithms have been proposed. Bayesian approaches include extended Kalman filters and particle filter approaches such as FastSLAM (Thrun et al., 2005). However these approaches estimate uncertainty from sensor noise models, not the uncertainty of the model to represent the data. Our proposed framework does not assume any input noise but measures the model uncertainty for localisation.

Neural networks which consider uncertainty are known as Bayesian neural networks (Denker and LeCun, 1991; MacKay, 1992). They offer a probabilistic interpretation of deep learning models by inferring distributions over the networks' weights (see Section 2.4 for a



(a) King's College (b) St Mary's Church (c) St Mary's Church.

Fig. 3.13 3-D scatter plots of **Monte Carlo pose samples from the Bayesian convolutional neural network** (top row) from an input image (bottom row) from the posterior distribution. We show typical examples from two scenes (a,b) and a visually ambiguous example (c). In green are the results from the first auxiliary pose regressor and in blue are samples from the final pose regressor. It shows that the auxiliary pose predictions (from the shallower sub-net) are typically multimodal however results from the final regressor are unimodal.

detailed introduction). They are often very computationally expensive, increasing the number of model parameters without increasing model capacity significantly. Performing inference in Bayesian neural networks is a difficult task, and approximations to the model posterior are often used, such as variational inference (Graves, 2011).

In Section 2.4, we explained that we can consider sampling with dropout as a way of getting samples from the posterior distribution of models. We leverage this method to obtain probabilistic inference of our pose regression model, forming a Bayesian PoseNet. *Dropout* is commonly used as a regularizer in convolutional neural networks to prevent over-fitting and co-adaption of features (Srivastava et al., 2014). During training with stochastic gradient descent, *dropout* randomly removes connections within a network. By doing this it samples from a number of thinned networks with reduced width. At test time, standard dropout approximates the effect of averaging the predictions of all these thinned networks by using the weights of the unthinned network. This can be thought of as sampling from a distribution over models.

we briefly summarise the method to obtain a Bayesian convolutional neural network introduced by (Gal and Ghahramani, 2016). Gal and Ghahramani (Gal and Ghahramani, 2016) show that dropout can be used at test time to impose a Bernoulli distribution over the convolutional net filter's weights, without requiring any additional model parameters. This is achieved by sampling the network with randomly dropped out connections at test time. We can consider these as Monte Carlo samples which sample from the posterior distribution of models. This is significantly different to the 'probabilities' obtained from a softmax classifier in classification networks. The softmax function provides relative probabilities between the class labels, but not an overall measure of the model's uncertainty.

We are interested in finding the posterior distribution over the convolutional weights, **W**, given our observed training data **X** and labels **Y**.

$$p(\mathbf{W} \mid \mathbf{X}, \mathbf{Y}) \tag{3.8}$$

In general, this posterior distribution is not tractable, we must use some learning method to approximate the distribution of these weights (Denker and LeCun, 1991). We use the approximation of Monte Carlo dropout (Gal and Ghahramani, 2015), introduced in Section 2.4. Monte Carlo dropout places a Bernoulli distribution over the model's weights. Sampling from this model, with stochastic dropout masks at test time, estimates the posterior distribution. The dropout probabilities, p_i, could be optimised (Gal et al., 2017). However we leave them at the standard probability of dropping a connection as 50%, i.e. $p_i = 0.5$ (Srivastava et al., 2014). Training the network with stochastic gradient descent will encourage the model to learn a distribution of weights which explains the data well while preventing over-fitting.

As a result of this dropout interpretation of Bayesian convolutional neural networks, a dropout layer should be added after every convolutional layer in the network. However in practice this is not the case as is explained in section 3.6.4. Using standard libraries such as (Jia et al., 2014) we can obtain an efficient implementation of a Bernoulli Bayesian convolutional neural network. At test time we perform inference by averaging stochastic samples from the dropout network.

Therefore the final algorithm for the probabilistic pose net is as follows:

Algorithm 1 Probabilistic PoseNet

```
Require: image, learned weights W, number of samples
1: for sample = 1 to number of samples do
2:     set network's weights to learned values
3:     for each weight in network do
4:         with probability $p$ set neuron activation to zero
5:     end for
6:     sample ← evaluate network
7: end for
8: compute pose as average of all samples
9: compute uncertainty as a function of the samples' variance
Ensure: 6-DOF pose, uncertainty
```

We also explored the possibility of using dense sliding window evaluation of the convolutional pose regressor over the input image to obtain a distribution of poses. This was done by taking 224×224 crops at regular intervals from the 455×256 pixel input image. This is equivalent to the densely evaluated PoseNet introduced in Section 3.4. The variance of these pose samples also correlates with localisation error, however not as strongly as sampling from a Bernoulli distribution over the weights.

3.6.2 Estimating Uncertainty

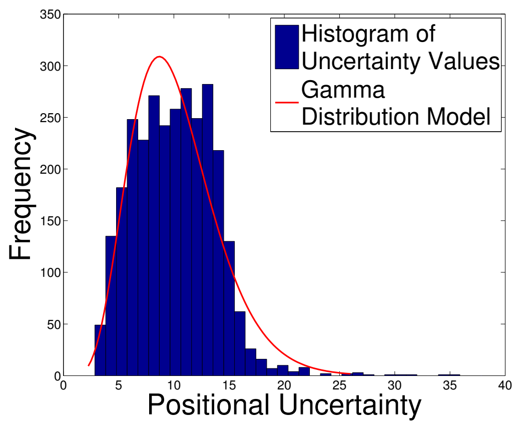
We can evaluate the posterior pose distribution from the Bayesian convolutional network by integrating with Monte Carlo sampling. Figure 3.13 shows a plot of Monte Carlo samples from the output of the posterior network in blue. We observe that the samples vary by a few metres, with less variance in the vertical dimension than the horizontal plane.

Additionally, the green points in 3.13 show the output from the first auxiliary pose regressor from the GoogLeNet architecture (see figure 3 of (Szegedy et al., 2015)). This output regresses pose from the representation after the inception (sub-net) layer 3. This result is at a much shallower depth and provides an insight as to what the network learns with additional depth. A similar result can be observed for the quaternion samples for the rotational component of pose.

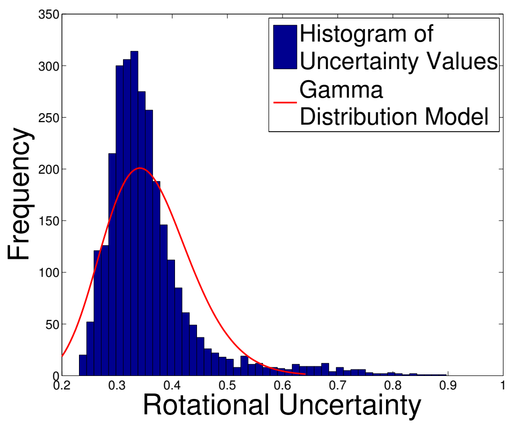
For the full network's output (blue) we obtain a distribution that appears to be drawn from both an isotropic and single-modal Gaussian. The network appears to be very certain about the specific pose. By sampling with dropout over the distribution of models we observe some isotropic scatter around a single pose estimate.

At a shallower depth, with the first auxiliary pose regressor (green), the results are multimodal. This is especially true for visually ambiguous images such as (c) in figure 3.13. The window in image (c) is repeated along the face of St Mary's Church. Using dropout to sample the distribution of models at this shallower depth produces distributions which have components drawn from multiple pose hypotheses. This suggests that this extra depth in the network is able to learn a representation that is sufficiently discriminative to distinguish visually similar landmarks.

Therefore, we fit a unimodal Gaussian to the samples from the network's final pose regressor. We treat the mean of these samples as the point estimate for pose. For an uncertainty measurement we take the trace of the unimodal Gaussian's covariance matrix. We have found the trace to be an effective scalar measure of uncertainty. The trace is a sum of the eigenvalues, which is rotationally invariant and represents the uncertainty that the Gaussian contains effectively. Figure 3.19 empirically shows this uncertainty measure is strongly correlated with metric error in relocalisation.



(a) Translational uncertainty



(b) Rotational uncertainty

Fig. 3.14 **Histograms of uncertainty values from the testing images in the Street scene.** In red we show the Gamma distribution used to model these populations. The Gamma distribution is a reasonable fit of the positively distributed, right skewed data.

We also considered using the determinant, which is a measure of the Gaussian's volume. However the determinant takes the product of the eigenvalues which means that if some are large and others are small then the resulting value will be small. This was the case as the resulting Gaussian often had a strong elongated component to it, as can be observed in figure 3.13. We found that using the determinant resulted in a numerically poorer measure of uncertainty.

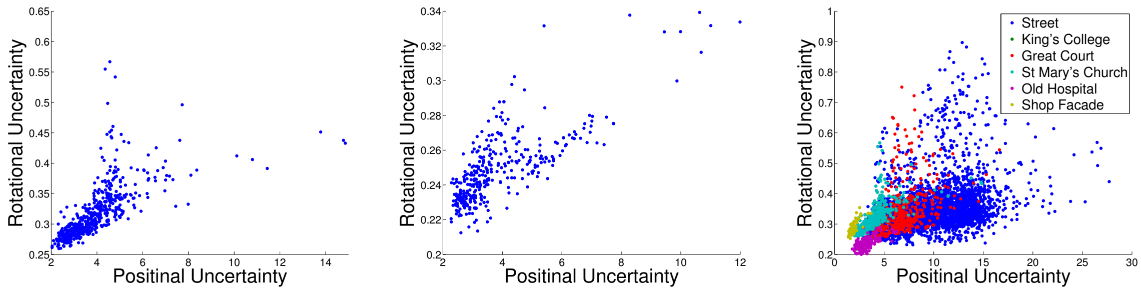
We tried other models which accounted for multi-modality in the data:

- taking the geometric median instead of the mean as a point prediction,
- fitting a mixture of Gaussians model to the data using the Dirichlet process (Blei and Jordan, 2006),
- clustering the samples using k-means and taking the mean of the largest cluster.

However we found that all of these methods produced poorer localisation uncertainty than fitting a single unimodal Gaussian to the data.

3.6.3 Creating a Comparable Uncertainty Statistic

In order to compare the uncertainty values we obtained from a model, we propose the following method to create a normalized measure, or Z-score. This is an uncertainty value which is able to be directly compared between models.



(a) St Mary's Church (b) King's College (c) All Scenes

Fig. 3.15 **Plot of translational uncertainty against rotational uncertainty** for test images in the St Mary's Church and King's College scene and for all scenes. This shows that the model uncertainty values are very strongly correlated for both rotation and translation. This suggests that we can form a single uncertainty value which represents the overall model uncertainty.

To achieve this, firstly we evaluate the test dataset and record the predicted camera poses and associated uncertainties for each scene. Typical distribution of uncertainty results for the Street scene can be viewed in figure 3.14. Examining this distribution, we chose to model it with a Gamma distribution for three reasons; it requires only two parameters, the distribution is constrained to strictly positive values only and is right skewed.

Obtaining an estimate for the distribution of model uncertainty values for images from a scene's test set allows us to evaluate where a new image's uncertainty values sit compared to the population. We can now assign a percentile to both the translational and rotational uncertainty values by using the cumulative distribution function for the Gamma distribution. We treat this percentile as a Z-score and generate this from a separate distribution for both the translational and rotational uncertainties, as well as separately for each scene.

Figure 3.15 shows that the rotational and translational uncertainties are highly correlated. We can therefore compute an overall localisation uncertainty by averaging the Z-score for translational and rotational uncertainty. This gives us a final single percentile score which we assign as the confidence of the pose prediction for a given model.

3.6.4 Architecture

To obtain a fully Bayesian model we should perform dropout sampling after every convolutional layer. However we found in practice this was not empirically optimal. In (Kendall et al., 2015) we discovered that fine tuning from pretrained filters trained on a large scale dataset such as *Places* (Zhou et al., 2014b) enhanced localisation accuracy significantly. This is again true with the probabilistic network. However these pretrained filters were trained without the use of dropout.

Fine-tuning from weights pretrained on the *Places* (Zhou et al., 2014b) dataset, we experimented with adding dropout throughout the model at different depths. We observe a performance drop in localisation when using the fully Bayesian convolutional neural network. Using dropout after every convolutional layer throughout the network acted as too strong a regularizer and degraded performance by approximately 10% . We obtained the optimal result when we included it only before convolutions which had randomly initialized weights. Therefore we add dropout after inception (sub-net) layer 9 and after the fully connected layers in the pose regressor.

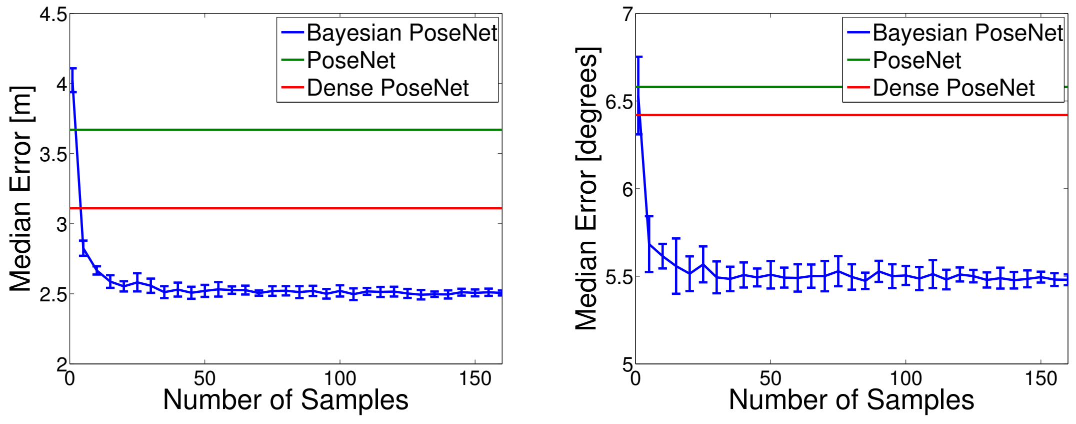
Yosinski et al. (Yosinski et al., 2014) argues that transferring weights can cause performance to drop in two situations. Firstly when the representation is too specific. However this is unlikely to be the case as we found the weights could successfully generalize to the new task (Kendall et al., 2015). The second explanation was that features may co-adapt fragilely and that transferring them breaks these co-adaptions. We believe this may be the case. The local minima that the weights were optimised to without dropout requires complex co-adaptions that are not able to optimise to a network with the same performance when using dropout.

We did not experiment with changing the dropout probability, or attempt to optimise this hyperparameter. We leave this to future work.

With this architecture we can then sample from the probabilistic model at test time to obtain an estimate of pose. We can improve localisation performance by averaging the Monte Carlo dropout samples (Gal and Ghahramani, 2016). Figure 3.16 gives empirical results suggesting that 40 samples are enough to achieve convergence of Monte Carlo samples. We show that less than five samples are typically required to surpass the performance of using a single pose regression convolutional net. After approximately 40 samples no more increase in localisation accuracy is observed.

3.6.5 Uncertainty Experiments

We evaluate the performance of the Bayesian convolutional neural network pose regressor on the localisation dataset, *Cambridge Landmarks*, which was introduced in (Kendall et al., 2015). Additionally we present results on an indoor relocalisation dataset, *7 Scenes* (Shotton et al., 2013). Table 3.17 presents the experimental results of localisation accuracy, averaging 100 Monte Carlo dropout samples from the probabilistic PoseNet. We compare this to PoseNet introduced in Section 3.4 and to a nearest neighbour baseline (Kendall et al., 2015) which finds the nearest pose from the training set in feature vector space. We also compare



(a) Translation (b) Rotation

Fig. 3.16 **localisation accuracy in the St Mary's Church scene for different number of Monte Carlo samples.** Results are averaged over 8 repetitions, with 1 standard deviation error bars shown. Horizontal lines are shown representing the performance of PoseNet (green) and densely evaluated PoseNet (red) (Kendall et al., 2015). This shows that Monte Carlo sampling provides significant improvement over both these point estimate models after a couple of samples. Monte Carlo sampling converges after around 40 samples and no more significant improvement is observed with more samples.

to the SCORE Forest algorithm which is state-of-the-art for relocalisation with depth data, however the need for RGB-D data constrains it to indoor use.

The results in table 3.17 show that using Monte Carlo dropout (Gal and Ghahramani, 2016) results in a considerable improvement in localisation accuracy, improving performance from Section 3.4 by $10-15\%$. Allowing the model to take into account the uncertainty of model selection, by placing a Bernoulli distribution over the weights, results in more accurate localisation. The Monte Carlo samples allow us to obtain estimates of poses probabilistically over the distribution of models. Taking the mean of these samples results in a more accurate solution.

Figure 3.18 shows a cumulative histogram of errors for two scenes. This shows that our probabilistic PoseNet performs consistently better than the non-probabilistic PoseNet for all error thresholds.

3.6.6 Uncertainty as an Estimate of Error

Figure 3.19 shows that the uncertainty estimate is very strongly correlated with metric relocalisation error. This shows that we can use the uncertainty estimate to predict relocalisation error. The plot shows that this relationship is linear for both translational and rotational

Scene	Spatial Extent	SCORE Forest (Uses RGB-D)	Dist. to Conv. Nearest Neighbour	PoseNet	Dense PoseNet	Bayesian PoseNet
=|=|====== King’s College	140 × 40m	N/A	3.34m, 2.96°	2.41m, 2.57°	1.82m, 2.38°	1.74m, 2.03°
Street	500 × 100m	N/A	1.95m 4.51°	4.92m, 4.99°	3.69m, 3.93°	3.36m, 3.06°
Old Hospital	50 × 40m	N/A	5.38m, 4.51°	3.32m, 2.93°	3.33m, 2.83°	2.57m, 2.57°
Shop Façade	35 × 25m	N/A	2.10m, 5.20°	1.69m, 4.84°	1.41m, 4.51°	1.25m, 3.77°
St Mary’s Church	80 × 60m	N/A	4.48m, 5.65°	3.67m, 6.58°	3.11m, 6.42°	2.54m, 5.46°
Average		N/A	3.45m, 4.57°	3.20m, 4.38°	2.67m, 4.02°	2.29m, 3.38°
Chess	3×2×1m	0.03m, 0.66°	0.41m, 5.60°	0.34m, 4.06°	0.32m, 3.76°	0.37m, 3.62°
Fire	2.5×1× 1m	0.05m, 1.50°	0.54m, 7.77°	0.57m, 7.33°	0.57m, 7.02°	0.43m, 6.84°
Heads	2×0.5×1m	0.06m, 5.50°	0.28m, 7.00°	0.29m, 6.00°	0.30m, 6.09°	0.31m, 6.01°
Office	2.5×2×1.5m	0.04m, 0.78°	0.49m, 6.02°	0.52m, 5.33°	0.48m, 5.09°	0.48m, 4.02°
Pumpkin	2.5×2×1m	0.04m, 0.68°	0.58m, 6.08°	0.47m, 4.33°	0.49m, 4.32°	0.61m, 3.54°
Red Kitchen	4×3×1.5m	0.04m, 0.76°	0.58m, 5.65°	0.63m, 4.32°	0.64m, 4.17°	0.58m, 3.77°
Stairs	2.5×2×1.5m	0.32m, 1.32°	0.56m, 7.71°	0.47m, 7.45°	0.48m, 7.49°	0.48m, 6.95°
Average		0.08m, 1.60°	0.49m, 6.55°	0.47m, 5.55°	0.47m, 5.42°	0.47m, 4.96°

Fig. 3.17 **Median localisation results for the *Cambridge Landmarks* (Kendall et al., 2015) and *7 Scenes* (Shotton et al., 2013) datasets.** We compare the performance of the probabilistic PoseNet to PoseNet and a nearest neighbour baseline (Kendall et al., 2015). Additionally we compare to SCORE Forests (Shotton et al., 2013) which requires depth input, limiting it to indoor scenes. The performance of the uncertainty model is shown with 100 Monte Carlo dropout samples. In addition to the qualitative improvement of obtaining an uncertainty metric, we also observe a consistent improvement in relocalisation accuracy of 10-15% over Dense PoseNet.

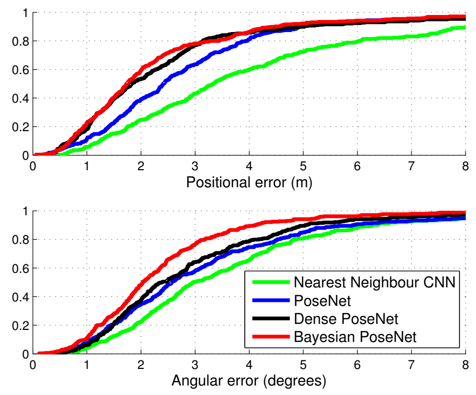
uncertainty. However the proportionality gradient between error and uncertainty varies significantly between scenes.

Figure 3.15 shows that metric error and uncertainty values are correlated between rotational and translational values. This supports the assumptions in our method of generating an overall uncertainty estimate as an ‘average’ of these normalized values. We observe relocalisation error and uncertainty are strongly correlated between both position and orientation.

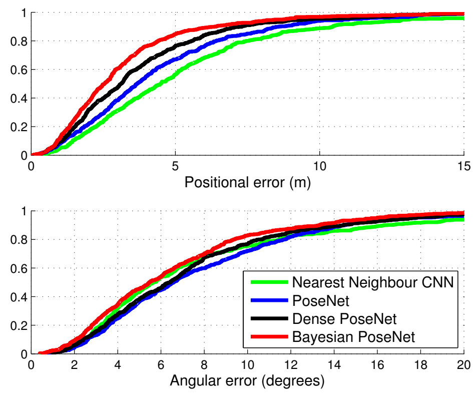
3.6.7 Uncertainty as a Landmark Detector

We show that the uncertainty metric can also be used to determine if the image is from the scene or landmark that the pose regressor was trained on. For a given scene in a dataset we test each image on all of the models. We then compute the uncertainty metric using the normalization method proposed in section 3.6.3. The image should have the lowest uncertainty value with the model which was trained on the scene that the image was taken from.

In figure 3.20 we present a confusion matrix showing this result for the *Cambridge Landmarks* and *7 Scenes* datasets. We exclude the Street scene as it contains many of the



(a) King's College



(b) St Mary's Church

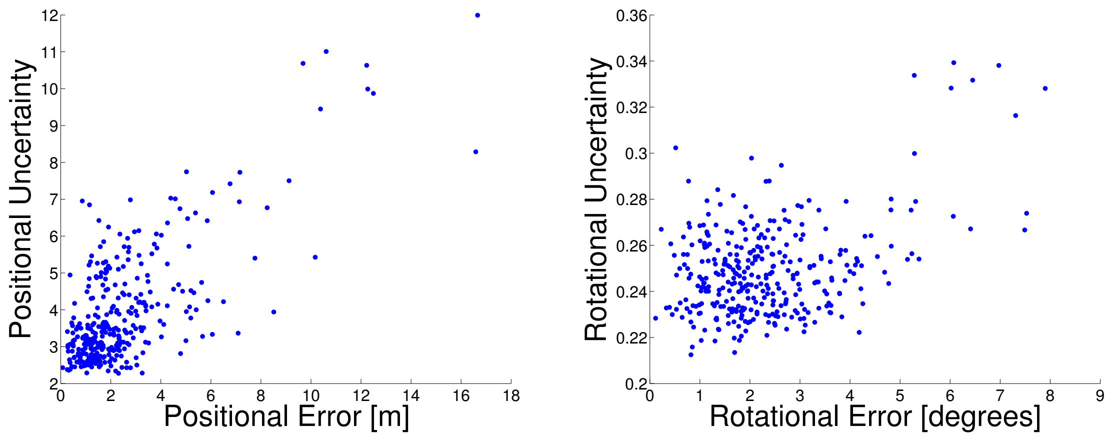
Fig. 3.18 **localisation accuracy for both position and orientation as a cumulative histogram of errors for the entire test set.** This shows that our probabilistic PoseNet performs consistently better than the non-probabilistic PoseNet for all error thresholds.

landmarks in the other scenes. We show the confusion matrix when using the combined normalized uncertainty. We observed that combining the rotation and translation metrics often provides a superior and more robust error metric.

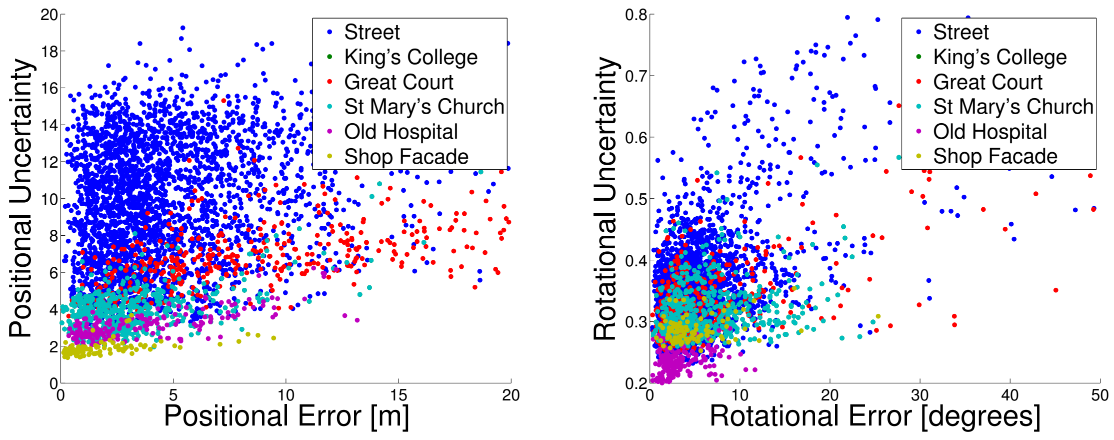
Note that the network has not been trained to classify the landmark it is observing. This is obtained as a by-product of the probabilistic architecture. If the convolutional net was trained to classify landmarks we are confident that it would perform significantly better. The purpose of this was to validate that the uncertainty measurement can reflect whether or not the network is trained on what it is presented with. The results show that the system can not only estimate the accuracy of the prediction, but also correctly identify when the landmark is not present at all.

3.6.8 What Makes the Model Uncertain About a Pose?

An initial hypothesis may be that test images which are far from training examples give very uncertain results, because they are more unknown to the network. To study this we plot, for each test image in a scene, the uncertainty against the Euclidean distance between the test image and the nearest training image. This plot shows a very slight increasing trend but is not sufficiently clear to draw any conclusions. However Euclidean distance to the nearest training image is not a comprehensive measure of similarity between images. It excludes other variables such as orientation, weather, pedestrian activity and lighting.



(a) King's College



(b) All Scenes

Fig. 3.19 **Plot of translational and rotational errors against their respective estimated uncertainty** for test images in the King's College scene and for all scenes. These plots show that the uncertainty is very strongly correlated with error and provides a good estimate of metric relocalisation error. It also shows that the scale of uncertainty values that each model learns varies significantly, suggesting they should be normalized for each model, as proposed in section 3.6.3.

	King's College	St Mary's	Old Hospital	Shop Facade
King's College	**75.22**	5.54	11.66	7.58
St Mary's	2.45	**79.43**	5.28	12.83
Old Hospital	11.54	0	**80.22**	8.24
Shop Facade	4.85	4.85	14.56	**75.73**

(a) Confusion matrix for *Cambridge Landmarks* dataset

	Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs
Chess	57.5	10.5	1.6	9.4	7.2	6.9	6.9
Fire	0.4	59.8	6.8	1.6	1.7	9.9	19.8
Heads	5.9	2.8	52.4	14	3.7	7.8	13.4
Office	12.2	11.4	6.2	42.4	6.6	14.9	6.3
Pumpkin	13.1	11.2	6.6	2.1	45.6	5.4	16
Red Kitchen	7.9	5.2	9.3	5.1	6.5	57.9	8.1
Stairs	11.1	4.5	2.8	11.1	18.4	1.9	50.2

(b) Confusion matrix for *7 Scenes* dataset

Fig. 3.20 **Scene recognition confusion matrices.** For each dataset (row) we computed the Z-score for both rotation and translation uncertainties. Dataset images were classified to the model (column) with the lowest uncertainty. Note that the Street scene is excluded as it contains many of the other landmarks in *Cambridge Landmarks*. This shows that the uncertainty metric is able to recognize correctly the landmark that it was trained to relocalise from. The network outputs large model uncertainty when it is presented with an unknown scene. The average scene detection accuracy is approximately 78% for *Cambridge Landmarks*. The indoor dataset is a far more challenging problem, as many scenes are very visually ambiguous. For example the pumpkin scene is the same room as the kitchen, with a different arrangement. Despite this, our system still performs modestly with 52% accuracy.

PoseNet produces a 2048 dimensional feature vector (see Section 3.4). This feature vector contains a high dimensional representation of instantiation parameters in the scene, such as weather, lighting, and object pose. Therefore we use this feature vector as a representation of the image. To compute similarity between two images, we evaluate the pose regressor's feature vector for each image and take the Euclidean distance between each feature vector. Therefore we can use this as a measure of similarity between a test image and the dataset's training image by finding the distance to the nearest neighbour training image in this feature space. This is the same measure used to compute the nearest neighbour results in table 3.17.

Figure 3.21 shows a plot of model uncertainty against this distance for all test images in the Street scene. The strong relationship indicates that the model is more uncertain for

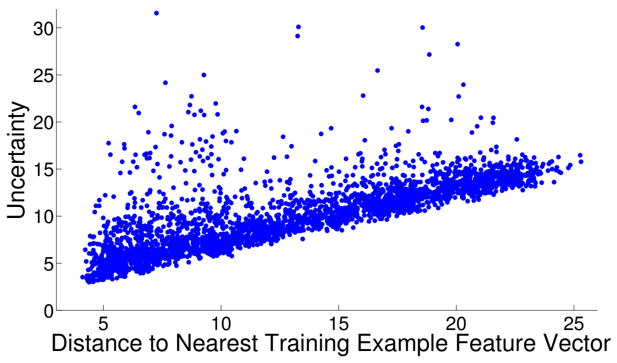


Fig. 3.21 **Uncertainty value for test images in the Street scene, plotted against Euclidean distance to the nearest neighbour training image feature vector.** The feature vector is a 2048 dimensional vector obtained from the final layer in PoseNet before the pose regression. This shows that having similar training examples lowers model uncertainty in test images.

images which are less similar (in this localisation feature space) to those in the training dataset.

The points which deviate from this trend, with larger uncertainty values, are typically the difficult images to localise. Some examples are shown in figure 3.22 These images have challenges such as heavy vehicle occlusion or strong silhouette lighting which result in inaccurate and uncertain prediction.

3.6.9 System Efficiency

We now compare the performance of our probabilistic PoseNet to our non-probabilistic PoseNet introduced in Section 3.4. The probabilistic approach shares all the same benefits of PoseNet, being scalable as its memory and computational requirements do not scale with map size or training data. Introducing dropout uncertainty does not require any more parametrisation and the weight file remains constant at 50 MB. This is still much more efficient than the gigabytes required for metric relocalisation systems with point features (Li et al., 2012).

Drawing stochastic samples however comes at an additional time cost. As figure 3.16 shows, the optimal samples to take is approximately 40 as any more samples than this does not significantly improve performance. When operating on a parallel processor, such as a GPU, this extra computation is manageable by treating it as a mini-batch of operations. This is no different to using the densely evaluated network introduced in Section 3.4. For example, computing pose by averaging 40 Monte Carlo dropout samples in practice takes $50ms$ while 128 samples takes $95ms$. For comparison, a single PoseNet evaluation takes $5ms$ per image.

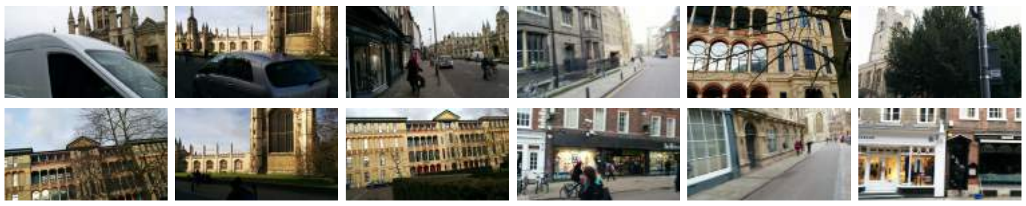


Fig. 3.22 **Images with the largest uncertainty values and largest localisation errors.** All of these images contain one of the following situations that cause difficult and uncertain localisation: strong occlusion from vehicles, pedestrians or other objects, motion blur, are taken from an area at the edge of the scene or are distant from a training example.

3.7 Summary

In this chapter, we investigated the problem of localisation and estimating the camera's 3-D position and orientation from a single image. We briefly summarise the main conclusions within the three main themes of this dissertation.

End-to-end deep learning. We show how to formulate an algorithm for localisation with an end-to-end deep neural network. We find this approach is more robust than traditional point-based feature approaches, being able to deal with significant lighting and pose variation. The algorithm is fast, and as the map is stored within the neural network's weights, scales very well with map size. We demonstrate effective relocalisation ability across large scale street scenes and indoor environments.

Geometry. We have investigated a number of loss functions for learning to regress position and orientation simultaneously with scene geometry. We present an algorithm for training PoseNet which does not require any hyper-parameter tuning. We achieve this by training using the reprojection error of 3-D scene geometry. We demonstrate PoseNet's efficacy on three large scale datasets. We observe a large improvement of results compared to the original loss proposed by PoseNet, narrowing the performance gap to traditional point-feature approaches.

Uncertainty. We show how to successfully apply an uncertainty framework to the convolutional neural network pose regressor, PoseNet. This improves relocalisation accuracy by $10-15\%$. We do this by averaging *Monte Carlo dropout* samples from the posterior Bernoulli distribution of the Bayesian convolutional network's weights. We show that the trace of the sample's covariance matrix provides an appropriate model-uncertainty estimate. We show that this uncertainty estimate accurately reflects the metric relocalisation error and can be used to detect the presence of a previously observed landmark. We present evidence that shows that the model is more uncertain about images which are dissimilar to the training examples, with application for exploratory loop closure detection.

Chapter 4

Stereo Vision

4.1 Introduction

Accurately estimating three dimensional geometry from stereo imagery is a core problem for many computer vision applications, including autonomous vehicles and UAVs (Achtelik et al., 2009). Stereo algorithms typically estimate the difference in the horizontal position of an object between a rectified pair of stereo images. This is known as *disparity*, which is inversely proportional to the scene depth at the corresponding pixel location. In this chapter we are specifically interested in computing the disparity of each pixel between a rectified stereo pair of images.

To achieve this, the core task of a stereo algorithm is computing the correspondence of each pixel between two images. This is very challenging to achieve robustly in real-world scenarios. Current state-of-the-art stereo algorithms often have difficulty with textureless areas, reflective surfaces, thin structures and repetitive patterns. Many stereo algorithms aim to mitigate these failures with pooling or gradient based regularization (Geiger et al., 2010; Hirschmuller, 2005). However, this often requires a compromise between smoothing surfaces and detecting detailed structures.

In contrast, deep learning models have been successful in learning powerful representations directly from the raw data in object classification (Krizhevsky et al., 2012), detection (Girshick et al., 2014) and semantic segmentation (Badrinarayanan et al., 2017; Long et al., 2015). These examples demonstrate that deep convolutional networks are very effective for understanding semantics. They excel at classification tasks when supervised with large

In this Chapter, Section 4.2, Section 4.3 and Section 4.4 was collaborative work with Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry and was published in (Kendall et al., 2017c)

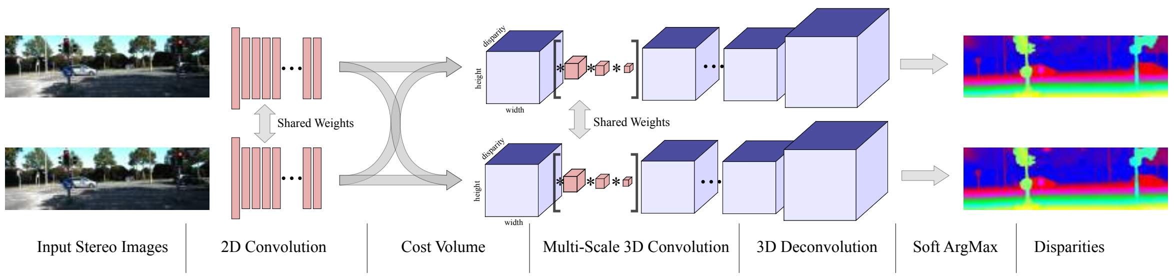


Fig. 4.1 **Our end-to-end deep stereo regression architecture, GC-Net** (Geometry and Context Network).

training datasets. We observe that a number of these challenging problems for stereo algorithms would benefit from knowledge of global semantic context, rather than relying solely on local geometry. For example, given a reflective surface of a vehicle's wind-shield, a stereo algorithm is likely to be erroneous if it relies solely on the local appearance of the reflective surface to compute geometry. Rather, it would be advantageous to understand the semantic context of this surface (that it belongs to a vehicle) to infer the local geometry. In this chapter we show how to learn a stereo regression model end-to-end, with the capacity to understand wider contextual information.

Stereo algorithms which leverage deep learning representations have so far been largely focused on using them to generate unary terms (Luo et al., 2016; Zbontar and LeCun, 2015). Applying cost matching on the deep unary representations performs poorly when estimating pixel disparities (Luo et al., 2016; Zbontar and LeCun, 2015). Traditional regularization and post processing steps are still used, such as semi global block matching and left-right consistency checks (Hirschmuller, 2005). These regularization steps are severely limited because they are hand-engineered, shallow functions, which are still susceptible to the aforementioned problems.

This chapter asks the question, can we formulate the entire stereo vision problem with deep learning using our understanding of stereo geometry? The main contribution of this chapter is an end-to-end deep learning method to estimate per-pixel disparity from a single rectified image pair. Our architecture is illustrated in Figure 4.1. It explicitly reasons about geometry by forming a cost volume, while also reasoning about semantics using a deep convolutional network formulation. We achieve this with two key ideas:

- We learn to incorporate context directly from the data, employing 3-D convolutions to learn to filter the cost volume over $height \times width \times disparity$ dimensions,
- We use a soft argmin function, which is fully differentiable, and allows us to regress sub-pixel disparity values from the disparity cost volume.

Section 4.3 introduces this model. In Section 4.4 we evaluate our model on the synthetic Scene Flow dataset (Mayer et al., 2016) and set a new state-of-the-art benchmark on the KITTI 2012 and 2015 datasets (Geiger et al., 2012; Menze and Geiger, 2015). Finally, in Section 4.4.4 we present evidence that our model has the capacity to learn semantic reasoning and contextual information.

In the remainder of this chapter, we demonstrate how to jointly learn depth from labelled and unlabelled data. We get the best of both worlds, leveraging labels to learn accurate disparities and large cohorts of unlabelled data for robustness. We do this by approaching the problem with a thorough probabilistic treatment. We make the observation that unsupervised learning can provide a strong signal for learning in many regions that supervised models find difficult – such as occlusion boundaries and thin structures, where there is a strong photometric discontinuity. However, regions which suffer from the aperture problem, such as sky and other flat, texture-less regions, are easier to solve with supervised learning. We show how to use recent advances in probabilistic deep learning to leverage the most informative signal from each training mode, and attenuate more uncertain areas.

To achieve this, we model uncertainty in stereo vision using *probabilistic deep learning*, which provides a framework for understanding uncertainty with deep learning models (Gal, 2016; Kendall and Gal, 2017) and was introduced in Chapter 2. In Section 4.5.1 we show how to form an architecture which learns to regress stereo disparities and *heteroscedastic* (data dependent) uncertainty (Der Kiureghian and Ditlevsen, 2009) from a rectified stereo pair of images. Our method does not require labels for uncertainty, rather it is learned implicitly from the data.

In summary, the main contributions of this chapter are:

1. forming an end-to-end model for stereo disparity regression which explicitly uses geometry,

2. advancing state-of-the-art on the Kitti stereo benchmark (Geiger et al., 2012),

3. demonstrating how to model uncertainty in stereo vision with probabilistic deep learning,

4. showing how to combine labelled and unlabelled data with semi-supervised learning.

4.2 Literature Review

The problem of computing depth from stereo image pairs has been studied for quite some time (Barnard and Fischler, 1982). Stereo algorithms typically estimate the difference in the horizontal position of an object between a rectified pair of stereo images. This is known as *disparity*, which is inversely proportional to the scene depth at the corresponding pixel location. A survey by Scharstein and Szeliski (Scharstein and Szeliski, 2002) provides a taxonomy of stereo algorithms as performing some subset of: matching cost computation, cost support aggregation, disparity computation and optimization, or disparity refinement. This survey also described the first Middlebury dataset and associated evaluation metrics, using structured light to provide ground truth. An improved higher resolution Middlebury dataset was presented in (Scharstein et al., 2014). The KITTI dataset (Geiger et al., 2012; Menze and Geiger, 2015) is a larger dataset from data collected from a moving vehicle with LIDAR ground truth. These datasets first motivated improved hand-engineered techniques for all components of stereo, of which we mention a few notable examples.

The matching cost is a measure of pixel dissimilarity for potentially corresponding objects across stereo images (Hirschmüller and Scharstein, 2007). The matching cost does not provide a measure of uncertainty, rather it predicts the relative likelihood between various disparity solutions. Traditionally, local descriptors based on gradients (Geiger et al., 2010) or binary patterns, such as CENSUS (Zabih and Woodfill, 1994) or BRIEF (Calonder et al., 2010; Heise et al., 2015), have been used. More recently, machine learning techniques have been applied to estimate stereo correspondences; Markov random fields (Zhang and Seitz, 2007), conditional random fields (Scharstein and Pal, 2007), support vector machines (Li and Huttenlocher, 2008) and deep learning (Zagoruyko and Komodakis, 2015; Zbontar and LeCun, 2015) have all been shown to be increasingly effective. Recent deep learning advances have improved performance by matching image patches using a Siamese network (Luo et al., 2016), multi-scale embeddings (Chen et al., 2015b).

Local matching costs often require post processing or regularization, which attempts to incorporate knowledge of the global context (Bleyer et al., 2011; Klaus et al., 2006; Kolmogorov and Zabih, 2001). A common technique is *Semi-Global Matching* (SGM) (Hirschmüller, 2008) which uses dynamic programming to optimize the path-wise form of the energy function in many directions.

In addition to providing a basis for comparing stereo algorithms, the ground truth depth data from these datasets provides the opportunity to use machine learning for improving stereo algorithms in a variety of ways. Zhang and Seitz (Zhang and Seitz, 2007) alternately optimized disparity and Markov random field regularization parameters. Scharstein and Pal (Scharstein and Pal, 2007) learn conditional random field (CRF) parameters, and Li and Huttenlocher (Li and Huttenlocher, 2008) train a non-parametric CRF model using the structured support vector machine. Learning can also be employed to estimate the confidence of a traditional stereo algorithm, such as the random forest approach of Haeusler et al. (Haeusler et al., 2013). Such confidence measures can improve the result of SGM as shown by Park and Yoon (Park and Yoon, 2015).

Deep convolutional neural networks can be trained to match image patches (Zagoruyko and Komodakis, 2015). A deep network trained to match 9×9 image patches, followed by non-learned cost aggregation and regularization, was shown by Žbontar and LeCun (Zbontar and LeCun, 2015, 2016) to produce then state-of-the-art results. Luo et al. presented a notably faster network for computing local matching costs as a multi-label classification of disparities using a Siamese network (Luo et al., 2016). A multi-scale embedding model from Chen et al. (Chen et al., 2015b) also provided good local matching scores. Also noteworthy is the *DeepStereo* work of Flynn et al. (Flynn et al., 2016), which learns a cost volume combined with a separate conditional colour model to predict novel viewpoints in a multi-view stereo setting.

Mayer et al. created a large synthetic dataset to train a network for disparity estimation (as well as optical flow) (Mayer et al., 2016), improving the state-of-the-art. As one variant of the network, a 1-D correlation was proposed along the disparity line which is a multiplicative approximation to the stereo cost volume. In contrast, our work does not collapse the feature dimension when computing the cost volume and uses 3-D convolutions to incorporate context.

Though the focus of this work is on binocular stereo, it is worth noting that the representational power of deep convolutional networks also enables depth estimation from a single monocular image (Eigen et al., 2014). Deep learning is combined with a continuous CRF by Liu et al. (Liu et al., 2015b). Instead of supervising training with labelled ground truth, unlabelled stereo pairs can be used to train a monocular model (Garg et al., 2016).

In our work, we apply no post-processing or regularization. We explicitly reason about geometry by forming a fully differentiable cost volume and incorporate context from the data with a 3-D convolutional architecture. We don't learn a probability distribution, cost function, or classification result. Rather, our network is able to directly regress a sub-pixel estimate of disparity from a stereo image pair.

4.3 Learning End-to-End Disparity Regression

Rather than design any step of the stereo algorithm by hand, we would like to learn an end-to-end mapping from an image pair to disparity maps using deep learning. We hope to learn a more optimal function directly from the data. Additionally, this approach promises to reduce much of the engineering design complexity. However, our intention is not to naively construct a machine learning architecture as a black box to model stereo. Instead, we advocate the use of the insights from many decades of multi-view geometry research

(Hartley and Zisserman, 2000) to guide architectural design. Therefore, we form our model by developing differentiable layers representing each major component in traditional stereo pipelines (Scharstein and Szeliski, 2002). This allows us to learn the entire model end-to-end while leveraging our geometric knowledge of the stereo problem.

Our architecture, GC-Net (Geometry and Context Network) is illustrated in Figure 4.1, with a more detailed layer-by-layer definition in Table 4.1. In the remainder of this section we discuss each component in detail. Later, in Section 4.4.1, we present quantitative results justifying our design decisions.

4.3.1 Unary Features

First we learn a deep representation to use to compute the stereo matching cost. Rather than compute the stereo matching cost using raw pixel intensities, it is common to use a feature representation. The motivation is to compare a descriptor which is more robust to the ambiguities in photometric appearance and can incorporate local context.

In our model we learn a deep representation through a number of 2-D convolutional operations. Each convolutional layer is followed by a batch normalization layer and a rectified linear non-linearity. To reduce computational demand, we initially apply a 5×5 convolutional filter with stride of two to sub-sample the input. Following this layer, we append eight residual blocks (He et al., 2016) which each consist of two 3×3 convolutional filters in series. Our final model architecture is shown in Table 4.1. We form the unary features by passing both left and right stereo images through these layers. We share the parameters between the left and right towers to more effectively learn corresponding features.

4.3.2 Cost Volume

We use the deep unary features to compute the stereo matching cost by forming a cost volume. While a naive approach might simply concatenate the left and right feature maps, forming a cost volume allows us to constrain the model in a way which preserves our knowledge of the geometry of stereo vision. For each stereo image, we form a cost volume of dimensionality *height*×*width*×(*max disparity + 1*)×*feature size*. We achieve this by concatenating each unary feature with their corresponding unary from the opposite stereo image across each disparity level, and packing these into the 4D volume. This is shown in Figure 4.2 for a single disparity line.

Crucially, we retain the feature dimension through this operation, unlike previous work which uses a dot product style operation which decimates the feature dimension (Luo et al., 2016). This allows us to learn to incorporate context which can operate over feature unaries

	Layer Description	Output Tensor Dim.
	Input image	H×W×C
	Unary features (section 4.3.1)	
1	5×5 conv, 32 features, stride 2	½H×½W×F
2	3×3 conv, 32 features	½H×½W×F
3	3×3 conv, 32 features	½H×½W×F
	add layer 1 and 3 features (residual connection)	½H×½W×F
4-17	(repeat layers 2,3 and residual connection) × 7	½H×½W×F
18	3×3 conv, 32 features, (no ReLu or BN)	½H×½W×F
	Cost volume (section 4.3.2)	
	Cost Volume	½D×½H×½W×2F
	Learning regularization (section 4.3.3)	
19	3-D conv, 3×3×3, 32 features	½D×½H×½W×F
20	3-D conv, 3×3×3, 32 features	½D×½H×½W×F
21	From Cost Volume: 3-D conv, 3×3×3, 64 features, stride 2	¼D×¼H×¼W×2F
22	3-D conv, 3×3×3, 64 features	¼D×¼H×¼W×2F
23	3-D conv, 3×3×3, 64 features	¼D×¼H×¼W×2F
24	From 21: 3-D conv, 3×3×3, 64 features, stride 2	⅛D×⅛H×⅛W×2F
25	3-D conv, 3×3×3, 64 features	⅛D×⅛H×⅛W×2F
26	3-D conv, 3×3×3, 64 features	⅛D×⅛H×⅛W×2F
27	From 24: 3-D conv, 3×3×3, 64 features, stride 2	1⁄16D×1⁄16H×1⁄16W×2F
28	3-D conv, 3×3×3, 64 features	1⁄16D×1⁄16H×1⁄16W×2F
29	3-D conv, 3×3×3, 64 features	1⁄16D×1⁄16H×1⁄16W×2F
30	From 27: 3-D conv, 3×3×3, 128 features, stride 2	1⁄32D×1⁄32H×1⁄32W×4F
31	3-D conv, 3×3×3, 128 features	1⁄32D×1⁄32H×1⁄32W×4F
32	3-D conv, 3×3×3, 128 features	1⁄32D×1⁄32H×1⁄32W×4F
33	3×3×3, 3-D transposed conv, 64 features, stride 2	1⁄16D×1⁄16H×1⁄16W×2F
	add layer 33 and 29 features (residual connection)	1⁄16D×1⁄16H×1⁄16W×2F
34	3×3×3, 3-D transposed conv, 64 features, stride 2	⅛D×⅛H×⅛W×2F
	add layer 34 and 26 features (residual connection)	⅛D×⅛H×⅛W×2F
35	3×3×3, 3-D transposed conv, 64 features, stride 2	¼D×¼H×¼W×2F
	add layer 35 and 23 features (residual connection)	¼D×¼H×¼W×2F
36	3×3×3, 3-D transposed conv, 32 features, stride 2	½D×½H×½W×F
	add layer 36 and 20 features (residual connection)	½D×½H×½W×F
37	3×3×3, 3-D trans conv, 1 feature (no ReLu or BN)	D×H×W×1
	Soft argmin (section 4.3.4)	
	Soft argmin	H×W

Table 4.1 **Summary of our end-to-end deep stereo regression architecture, GC-Net.** Each 2-D or 3-D convolutional layer represents a block of convolution, batch normalization and ReLU non-linearity (unless otherwise specified).

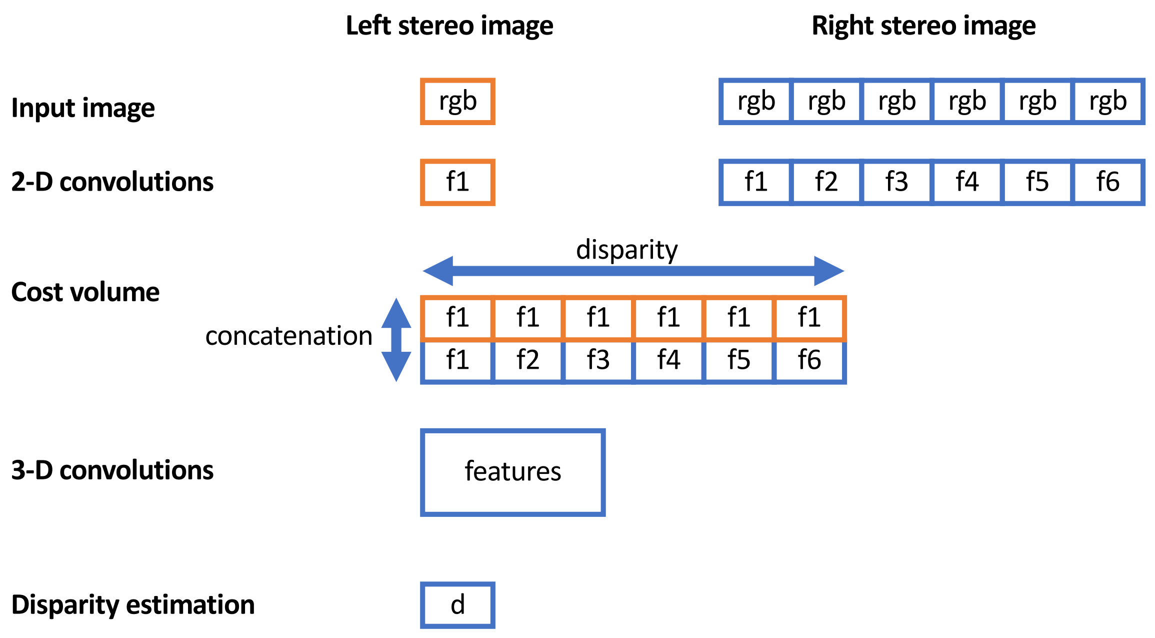


Fig. 4.2 **An explaination of the GC-Net architecture along a single disparity line**. This figure shows the transformations each layer imposes on the features.

(Section 4.3.3). We find that forming a cost volume with concatenated features improves performance over subtracting features or using a distance metric. Our intuition is that by maintaining the feature unaries, the network has the opportunity to learn an absolute representation (because it is not a distance metric) and carry this through to the cost volume. This gives the architecture the capacity to learn semantics. In contrast, using a distance metric restricts the network to only learning relative representations between features, and cannot carry absolute feature representations through to cost volume.

4.3.3 Learning Context

Given this disparity cost volume, we would now like to learn a regularization function which is able to take into account context in this volume and refine our disparity estimate. The matching costs between unaries can never be perfect, even when using a deep feature representation. For example, in regions of uniform pixel intensity (for example, sky) the cost curve will be flat for any features based on a fixed, local context. We find that regions like this can cause multi modal matching cost curves across the disparity dimension. Therefore we wish to learn to regularize and improve this volume.

We propose to use three-dimensional convolutional operations to filter and refine this representation. 3-D convolutions are able to learn feature representations from the height, width and disparity dimensions. Because we compute the cost curve for each unary feature, we can learn convolutional filters from this representation. In Section 4.4.1 we show the importance of these 3-D filters for learning context and significantly improving stereo performance.

The difficulty with 3-D convolutions is that the additional dimension is a burden on the computational time for both training and inference. Deep encoder-decoder tasks which are designed for dense prediction tasks get around their computational burden by encoding sub-sampled feature maps, followed by up-sampling in a decoder (Badrinarayanan et al., 2017). We extend this idea to three dimensions. By sub-sampling the input with stride two, we also reduce the 3-D cost volume size by a factor of eight. We form our 3-D regularization network with four levels of sub-sampling. As the unaries are already sub-sampled by a factor of two, the features are sub-sampled by a total factor of 32. This allows us to explicitly leverage context with a wide field of view. We apply two $3\times3\times3$ convolutions in series for each encoder level. To make dense predictions with the original input resolution, we employ a 3-D transposed convolution to up-sample the volume in the decoder. The full architecture is described in Table 4.1.

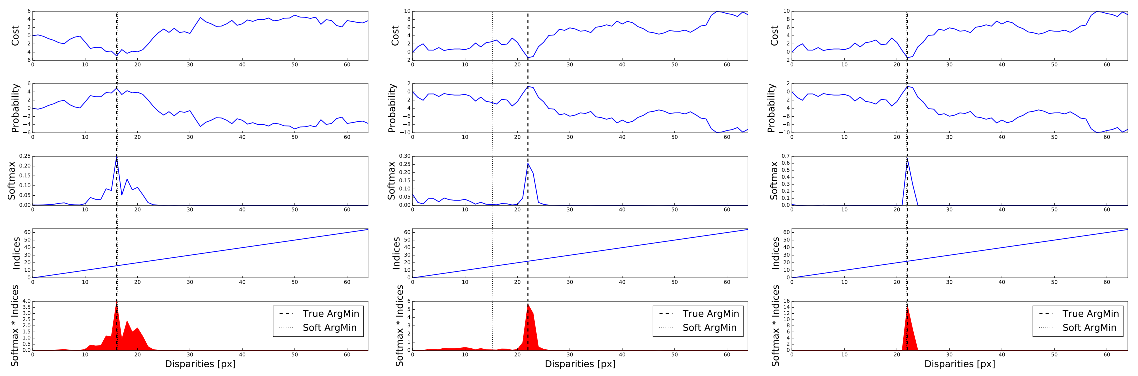
Sub-sampling is useful to increase each feature's receptive field while reducing computation. However, it also reduces spatial accuracy and fine-grained details through the loss of resolution. For this reason, we add each higher resolution feature map before up-sampling. These residual layers have the benefit of retaining higher frequency information, while the up-sampled features provide an attentive feature map with a larger field of view.

Finally, we apply a single 3-D transposed convolution (deconvolution), with stride two and a single feature output. This layer is necessary to make dense prediction in the original input dimensions because the feature unaries were sub-sampled by a factor of two. This results in the final, regularized cost volume with size $\mathrm{H}\times\mathrm{W}\times\mathrm{D}$.

4.3.4 Differentiable ArgMin

Typically, stereo algorithms produce a final cost volume from the matching cost unaries. From this volume, we may estimate disparity by performing an argmin operation over the cost volume's disparity dimension. However, this operation has two problems:

- it is discrete and is unable to produce sub-pixel disparity estimates,
- it is not differentiable and therefore unable to be trained using back-propagation.



(a) Soft ArgMin (b) Multi-modal distribution (c) Multi-modal distribution with prescaling

Fig. 4.3 **A graphical depiction of the soft argmin operation** (Section 4.3.4) which we propose in this work. It is able to take a cost curve along each disparity line and output an estimate of the argmin by summing the product of each disparity's softmax probability and its disparity index. (a) demonstrates that this very accurately captures the true argmin when the curve is uni-modal. (b) demonstrates a failure case when the data is bi-modal with one peak and one flat region. (c) demonstrates that this failure may be avoided if the network learns to pre-scale the cost curve, because the softmax probabilities will tend to be more extreme, producing a uni-modal result.

To overcome these limitations, we define a *soft argmin*[2] which is both fully differentiable and able to regress a smooth disparity estimate. First, we convert the predicted costs, c_d (for each disparity, d) from the cost volume to a probability volume by taking the negative of each value. We normalize the probability volume across the disparity dimension with the softmax operation, $\sigma(\cdot)$. We then take the sum of each disparity, d, weighted by its normalized probability. A graphical illustration is shown in Figure 4.3 and defined mathematically in (4.1):

$$soft\ argmin := \sum_{d=0}^{D_{max}} d \times \sigma(-c_d) \tag{4.1}$$

This operation is fully differentiable and allows us to train and regress disparity estimates. We note that a similar function was first introduced by (Bahdanau et al., 2014) and referred to as a soft-attention mechanism. Here, we show how to apply it for the stereo regression problem.

However, compared to the argmin operation, its output is influenced by all values. This leaves it susceptible to multi-modal distributions, as the output will not take the most likely.

[2]Note that if we wished for our network to learn probabilities, rather than cost, this function could easily be adapted to a soft argmax operation.

Rather, it will estimate a weighted average of all modes. To overcome this limitation, we rely on the network's regularization to produce a disparity probability distribution which is predominantly unimodal. The network can also pre-scale the matching costs to control the peakiness (sometimes called temperature) of the normalized post-softmax probabilities (Figure 4.3). We explicitly omit batch normalization from the final convolution layer in the unary tower to allow the network to learn this from the data.

4.3.5 Loss

We train our entire model end-to-end from a random initialization. We train our model with supervised learning using ground truth depth data. In the case of using LIDAR to label ground truth values (e.g. KITTI dataset (Geiger et al., 2012; Menze and Geiger, 2015)) these labels may be sparse. Therefore, we average our loss over the labelled pixels, N. We train our model using the absolute error between the ground truth disparity, d_n, and the model's predicted disparity, $\hat{d}_n$, for pixel n. This supervised regression loss is defined in (4.2):

$$Loss = \frac{1}{N} \sum_{n=1}^{N} \left\| d_n - \hat{d}_n \right\|_1 \tag{4.2}$$

In the following section we show that formulating our model as a regression problem allows us to regress with sub-pixel accuracy and outperform classification approaches. Additionally, formulating a regression model makes it possible to leverage unsupervised learning losses based on photometric reprojection error (Garg et al., 2016).

4.4 Experimental Evaluation

In this section we present qualitative and quantitative results on two datasets, Scene Flow (Mayer et al., 2016) and KITTI (Geiger et al., 2012; Menze and Geiger, 2015). Firstly, in Section 4.4.1 we experiment with different variants of our model and justify a number of our design choices using the Scene Flow dataset (Mayer et al., 2016). In Section 4.4.2 we present results of our approach on the KITTI dataset and set a new state-of-the-art benchmark. Finally, we measure our model's capacity to learn context in Section 4.4.4.

For the experiments in this chapter, we implement our architecture using TensorFlow (Abadi et al., 2016). All models are optimized end-to-end with RMSProp (Tieleman and Hinton, 2012) and a constant learning rate of 1×10^{-3}. We train with a batch size of 1 using a 256×512 randomly located crop from the input images. Before training we normalize each image such that the pixel intensities range from -1 to 1. We trained the network (from a

Model	> 1 px	> 3 px	> 5 px	MAE (px)	RMS (px)	Param.	Time (ms)
1. Comparison of architectures							
Unaries only (omitting all 3-D conv layers 19-36) w Regression Loss	97.9	93.7	89.4	36.6	47.6	0.16M	0.29
Unaries only (omitting all 3-D conv layers 19-36) w Classification Loss	51.9	24.3	21.7	13.1	36.0	0.16M	0.29
Single scale 3-D context (omitting 3-D conv layers 21-36)	34.6	24.2	21.2	7.27	20.4	0.24M	0.84
Hierarchical 3-D context (all 3-D conv layers)	16.9	9.34	7.22	2.51	12.4	3.5M	0.95
2. Comparison of loss functions							
GC-Net + Classification loss	19.2	12.2	10.4	5.01	20.3	3.5M	0.95
GC-Net + Soft classification loss (Luo et al., 2016)	20.6	12.3	10.4	5.40	25.1	3.5M	0.95
GC-Net + Regression loss	16.9	9.34	7.22	2.51	12.4	3.5M	0.95
GC-Net (final architecture with regression loss)	16.9	9.34	7.22	2.51	12.4	3.5M	0.95

Table 4.2 **Results on the Scene Flow dataset** (Mayer et al., 2016) which contains $35,454$ training and $4,370$ testing images of size 960×540px from an array of synthetic scenes. We compare different architecture variants to justify our design choices. The first experiment shows the importance of the 3-D convolutional architecture. The second experiment shows the performance gain from using a regression loss.

random initialization) on Scene Flow for approximately 150k iterations which takes two days on a single NVIDIA Titan-X GPU. For the KITTI dataset we fine-tune the models pre-trained on Scene Flow for a further 50k iterations. For our experiments on Scene Flow we use F=32, H=540, W=960, D=192 and on the KITTI dataset we use F=32, H=388, W=1240, D=192 for feature size, image height, image width and maximum disparity, respectively.

4.4.1 Model Design Analysis

In Table 4.2 we present an ablation study to compare a number of different model variants and justify our design choices. We wish to evaluate the importance of the key ideas in this section; using a regression loss over a classification loss, and learning 3-D convolutional filters for cost volume regularization. We use the synthetic Scene Flow dataset (Mayer et al., 2016) for these experiments, which contains $35,454$ training and $4,370$ testing images. We use this dataset for two reasons. Firstly, we know perfect, dense ground truth from the synthetic scenes which removes any discrepancies due to erroneous labels. Secondly, the dataset is large enough to train the model without over-fitting. In contrast, the KITTI dataset only contains 200 training images, making the model is susceptible to over-fitting to this very small dataset. With tens of thousands of training images we do not have to consider over-fitting in our evaluation.

The first experiment in Table 4.2 shows that including the 3-D filters performs significantly better than learning unaries only. We compare our full model (as defined in Table 4.1) to a model which uses only unary features (omitting all 3-D convolutional layers 19-36) and a model which omits the hierarchical 3-D convolution (omitting layers 21-36). We observe that the 3-D filters are able to regularize and smooth the output effectively, while learning to retain sharpness and accuracy in the output disparity map. We find that the hierarchical 3-D model outperforms the vanilla 3-D convolutional model by aggregating a much large context, without significantly increasing computational demand.

The second experiment in Table 4.2 compares our regression loss function to baselines classifying disparities with hard or soft classification as proposed in (Luo et al., 2016). Hard classification trains the network to classify disparities in the cost volume as probabilities using cross entropy loss with a ‘one hot’ encoding. Soft classification (used by (Luo et al., 2016)) smooths this encoding to learn a Gaussian distribution centred around the correct disparity value. In Table 4.2 we show our regression approach outperforms both hard and soft classification. This is especially noticeable for the pixel accuracy metrics and the percentage of pixels which are within one pixel of the true disparity, because the regression loss allows the model to predict with sub-pixel accuracy.

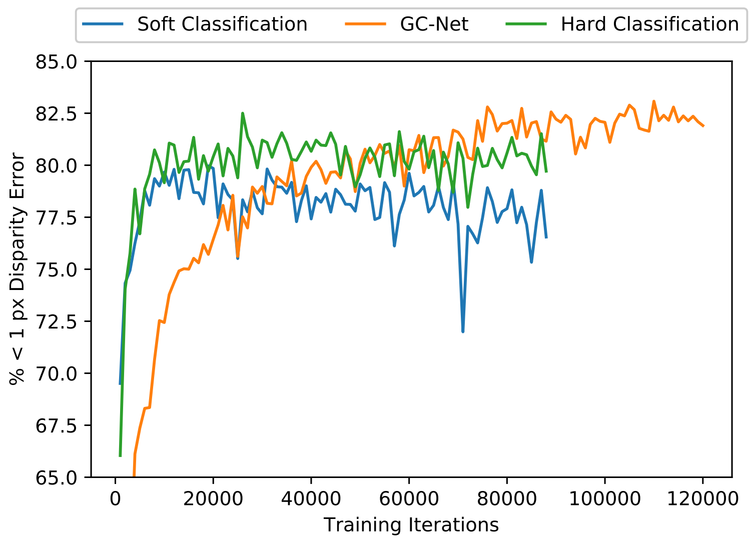


Fig. 4.4 **Validation error** (percentage of disparities with error less than 1 px) during training with the Scene Flow dataset. Classification loss trains faster, however using a regression loss results in better performance.

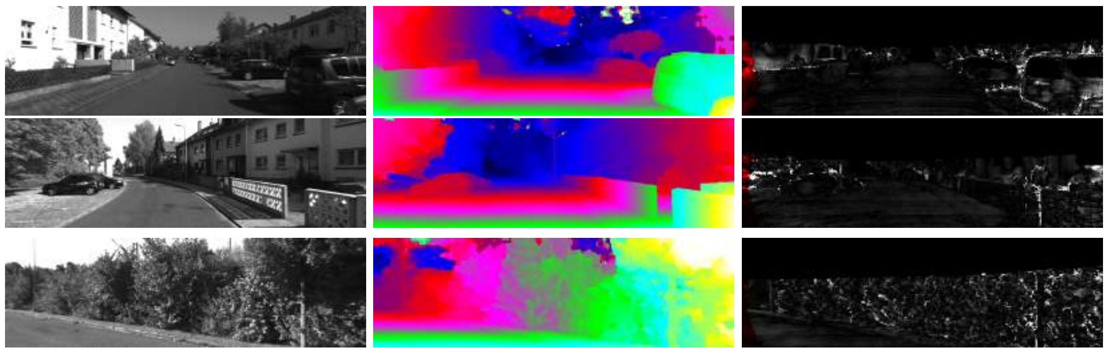
Figure 4.4 plots validation error during training for each of the networks compared in this section. We observe that the classification loss converges faster, however the regression loss performs best overall.

4.4.2 KITTI Benchmark

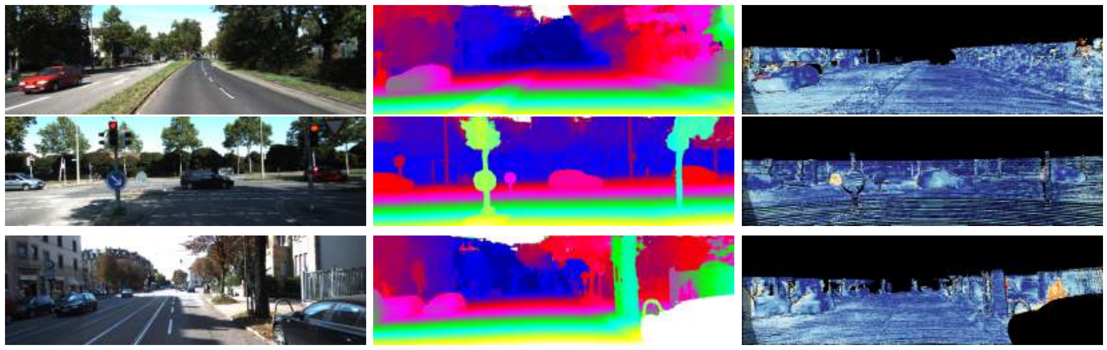
In Table 4.3 we evaluate the performance of our model on the KITTI 2012 and 2015 stereo datasets (Geiger et al., 2012; Menze and Geiger, 2015). These consist of challenging and varied road scene imagery collected from a test vehicle. Ground truth depth maps for training and evaluation are obtained from LIDAR data. KITTI is a prominent dataset for benchmarking stereo algorithms. The downside is that it only contains 200 training images, which handicaps learning algorithms. for this reason, we pre-train our model on the large synthetic dataset, Scene Flow (Mayer et al., 2016). This helps to prevent our model from over-fitting the very small KITTI training dataset. We hold out 40 image pairs as our validation set.

Table 4.3a and 4.3b compare our method, GC-Net (Geometry and Context Network), to other approaches on the KITTI 2012 and 2015 datasets, respectively[3]. Our method achieves state of the art results for both KITTI benchmarks, by a notable margin. We improve on state-of-the-art by 9% and 22% for KITTI 2015 and 2012 respectively. Our method is also notably

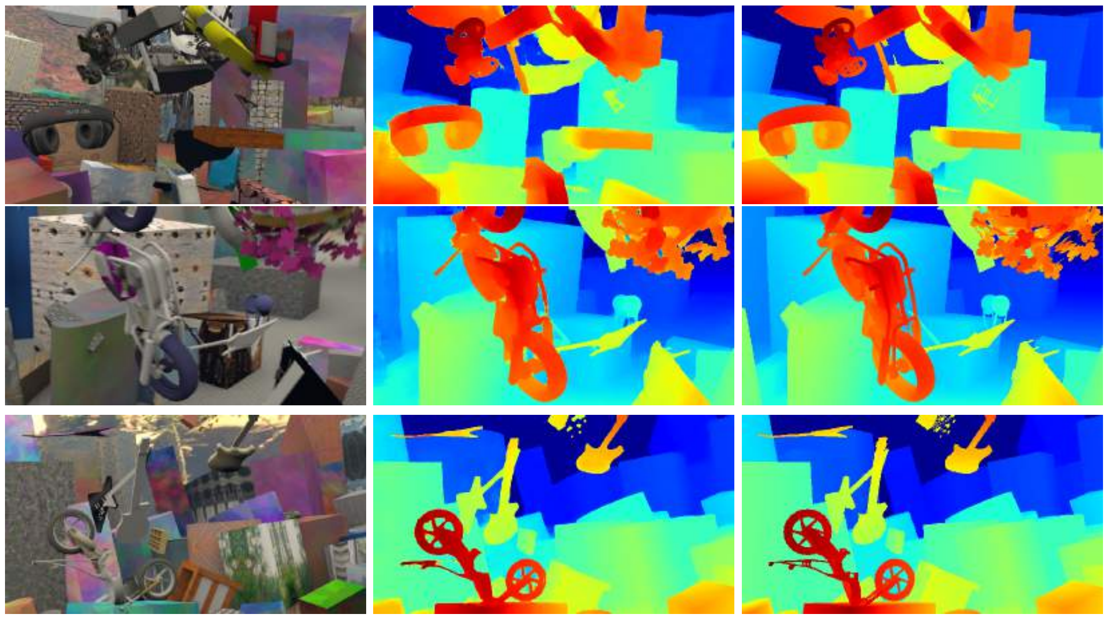
[3]Full leaderboard: www.cvlibs.net/datasets/kitti/



(a) KITTI 2012 test data qualitative results. From left: left stereo input image, disparity prediction, error map.



(b) KITTI 2015 test data qualitative results. From left: left stereo input image, disparity prediction, error map.



(c) Scene Flow test set qualitative results. From left: left stereo input image, disparity prediction, ground truth.

Fig. 4.5 **Qualitative results.** By learning to incorporate wider context our method is often able to handle challenging scenarios, such as reflective, thin or texture-less surfaces. By explicitly learning geometry in a cost volume, our method produces sharp results and can also handle large occlusions.

	>2 px		>3 px		>5 px		Mean Error		Runtime
	Non-Occ	All	Non-Occ	All	Non-Occ	All	Non-Occ	All	(s)
SPS-st (Yamaguchi et al., 2014)	4.98	6.28	3.39	4.41	2.33	3.00	0.9 px	1.0 px	2
Deep Embed (Chen et al., 2015b)	5.05	6.47	3.10	4.24	1.92	2.68	0.9 px	1.1 px	3
Content-CNN (Luo et al., 2016)	4.98	6.51	3.07	4.29	2.03	2.82	0.8 px	1.0 px	**0.7**
MC-CNN (Zbontar and LeCun, 2016)	3.90	5.45	2.43	3.63	1.64	2.39	0.7 px	0.9 px	67
PBCP (Seki and Pollefeys, 2016)	3.62	5.01	2.36	3.45	1.62	2.32	0.7 px	0.9 px	68
Displets v2 (Guney and Geiger, 2015)	3.43	4.46	2.37	3.09	1.72	2.17	0.7 px	0.8 px	265
GC-Net (this work)	**2.71**	**3.46**	**1.77**	**2.30**	**1.12**	**1.46**	**0.6 px**	**0.7 px**	0.9

(a) **KITTI 2012 test set results** (Geiger et al., 2012). This benchmark contains 194 train and 195 test gray-scale image pairs.

	All Pixels			Non-Occluded Pixels			Runtime
	D1-bg	D1-fg	D1-all	D1-bg	D1-fg	D1-all	(s)
MBM (Einecke and Eggert, 2015)	4.69	13.05	6.08	4.33	12.12	5.61	0.13
ELAS (Geiger et al., 2010)	7.86	19.04	9.72	6.88	17.73	8.67	0.3
Content-CNN (Luo et al., 2016)	3.73	8.58	4.54	3.32	7.44	4.00	1.0
DispNetC (Mayer et al., 2016)	4.32	**4.41**	4.34	4.11	**3.72**	4.05	**0.06**
MC-CNN (Zbontar and LeCun, 2016)	2.89	8.88	3.89	2.48	7.64	3.33	67
PBCP (Seki and Pollefeys, 2016)	2.58	8.74	3.61	2.27	7.71	3.17	68
Displets v2 (Guney and Geiger, 2015)	3.00	5.56	3.43	2.73	4.95	3.09	265
GC-Net (this work)	**2.21**	6.16	**2.87**	**2.02**	5.58	**2.61**	0.9

(b) **KITTI 2015 test set results** (Menze and Geiger, 2015). This benchmark contains 200 training and 200 test color image pairs. The qualifier ‘bg’ refers to background pixels which contain static elements, ‘fg’ refers to dynamic object pixels, while ‘all’ is all pixels (fg+bg). The results show the percentage of pixels which have greater than three pixels or 5% disparity error from all 200 test images.

Table 4.3 Comparison to other stereo methods on the test set of **KITTI 2012 and 2015 benchmarks** (Geiger et al., 2012; Menze and Geiger, 2015). Our method sets a new state-of-the-art on these two competitive benchmarks, out performing all other approaches.

faster than most competing approaches which often require expensive post-processing. In Figure 4.5 we show qualitative results of our method on KITTI 2012, KITTI 2015 and Scene Flow.

Our approach outperforms previous deep learning patch based methods (Luo et al., 2016; Zbontar and LeCun, 2015) which produce noisy unary potentials and are unable to predict with sub-pixel accuracy. For this reason, these algorithms do not use end-to-end learning and typically post-process the unary output with SGM regularization (Einecke and Eggert, 2015) to produce the final disparity maps.

The closest method to our architecture is DispNetC (Mayer et al., 2016), which is an end-to-end regression network pre-trained on SceneFlow. However, our method outperforms this architecture by a notable margin for *all* test pixels. DispNetC uses a 1-D correlation layer along the disparity line as an approximation to the stereo cost volume. In contrast, our architecture more explicitly leverages geometry by formulating a full cost volume by using 3-D convolutions and a soft argmin layer, resulting in an improvement in performance.

4.4.3 Cross Dataset Generalization

In Table 4.4 we investigate our model's ability to generalize across unseen datasets. We observe that our model is able to perform amicably on KITTI test data, even when only trained on Scene Flow – a dissimilar synthetic dataset.

Train data	Test data	>3 px	Mean Error
Scene Flow	KITTI 2015 Val	18.6	2.23
KITTI 2015 Train	KITTI 2015 Val	1.50	0.59

Table 4.4 Cross dataset performance. These results show that our method is able to generalize reasonably well across unseen datasets.

4.4.4 Model Saliency

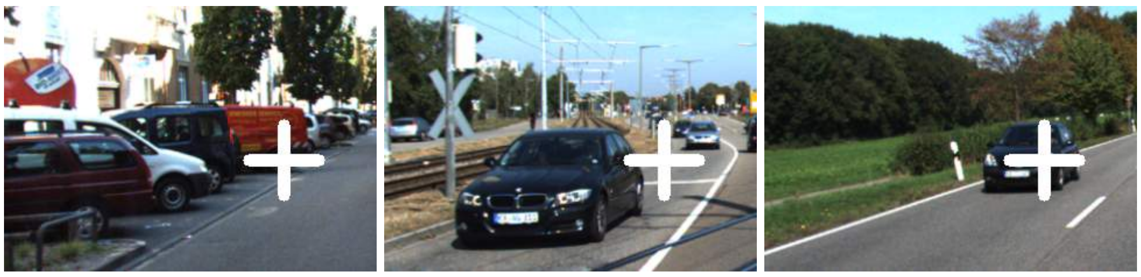
In this section we present evidence which shows our model can reason about local geometry using wider contextual information. In Figure 4.6 we show some examples of the model's saliency with respect to a predicted pixel's disparity. Saliency maps (Simonyan et al., 2013) shows the sensitivity of the output with respect to each input pixel. We use the method from (Zeiler and Fergus, 2014) which plots the predicted disparity as a function of systematically occluding the input images. We offset the occlusion in each stereo image by the point's disparity.

These results show that the disparity prediction for a given point is dependent on a wide contextual field of view. For example, the disparity on the front of the car depends on the input pixels of the car and the road surface below. This demonstrates that our model is able to reason about wider context, rather than simply 9×9 local patches like previous deep learning patch-similarity stereo methods (Luo et al., 2016; Zbontar and LeCun, 2016).

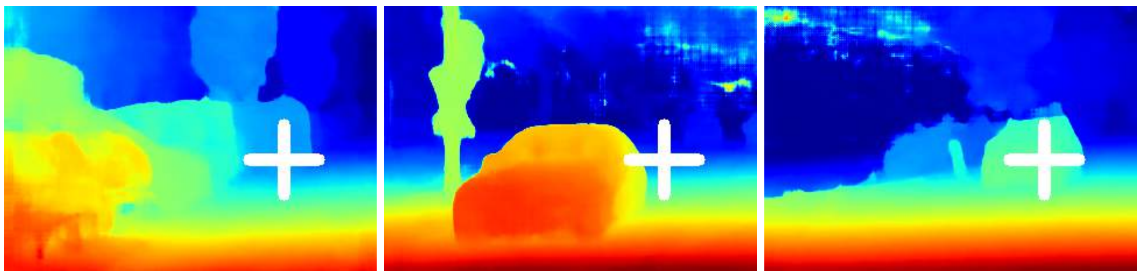
Next, we explore modelling uncertainty in stereo vision and show how to learn to regress depth with unsupervised learning.

4.5 Uncertainty in Stereo Vision

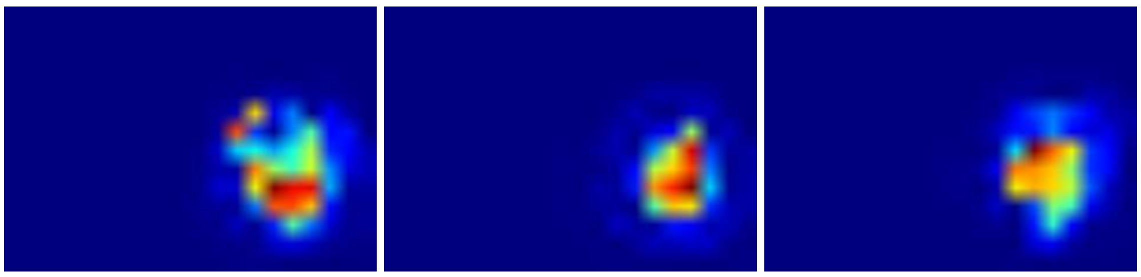
In this section, we present a method to model uncertainty in stereo vision using *probabilistic deep learning*, which provides a framework for understanding uncertainty with deep learning



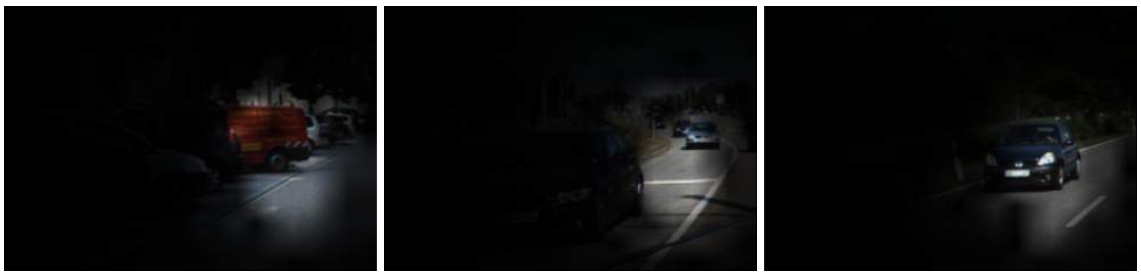
(a) Left stereo input image



(b) Predicted disparity map

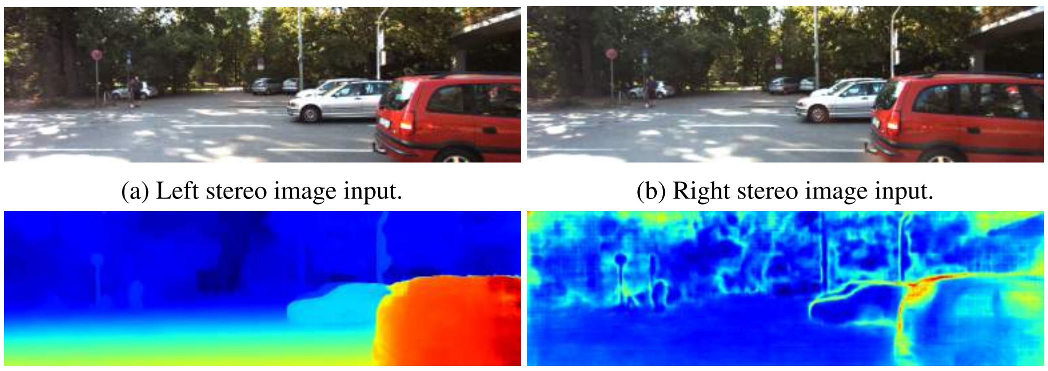


(c) Saliency map (red = stronger saliency)



(d) What the network sees (input attenuated by saliency)

Fig. 4.6 **Saliency map visualization** which shows the model's effective receptive field for a selected output pixel (indicated by the white cross). This shows that our architecture is able to learn to regress stereo disparity with a large field of view and significant contextual knowledge of the scene, beyond the local geometry and appearance. For example, in the example on the right we observe that the model considers contextual information from the vehicle and surrounding road surface to estimate disparity.



(a) Left stereo image input. (b) Right stereo image input.

(c) Disparity output. (d) Heteroscedastic uncertainty output.

Fig. 4.7 **End-to-end probabilistic deep learning model for stereo disparity regression.** We observe increased uncertainty on occlusion boundaries and difficult surfaces, such as the red car's reflective and transparent windows.

models (Chapter 2). In Section 4.5.1 we show how to form an architecture which learns to regress stereo disparities and *heteroscedastic* (data dependent) uncertainty (Der Kiureghian and Ditlevsen, 2009) from a rectified stereo pair of images. Our method does not require labels for uncertainty, rather it is learned implicitly from the data. Unlike other probabilistic deep learning approaches (Gal, 2016), our method does not depend on drop-out sampling and is real-time.

We first review related work in stereo vision which considers uncertainty. Hand-engineered measures of uncertainty in matching costs are often used in stereo to improve disparity map prediction (Egnal et al., 2004; Hu and Mordohai, 2012). Most uncertainty measures are designed to detect occlusions, surface discontinuities and texture-less regions. Hand designed approaches are common, for example using statistical approaches (Sabater et al., 2012), measures of distinctiveness (Manduchi and Tomasi, 1999; Yoon and Kweon, 2007) or entropy (Scharstein and Szeliski, 1998). Random forests have also be used to estimate matching cost confidence (Haeusler et al., 2013; Park and Yoon, 2015). However, the use of uncertainty in stereo has predominantly been limited to estimating the confidence of matching costs to assist post-processing and regularization. In this section, we estimate the uncertainty of the final predicted disparity map.

4.5.1 Modelling Uncertainty in Stereo Vision

In this section we introduce probabilistic deep learning, and show how to model uncertainty in stereo vision. In Chapter 2, we explained there are two main types of uncertainty that one can model in Bayesian modelling (Der Kiureghian and Ditlevsen, 2009). *Aleatoric* uncertainty captures noise inherent in the observations. This could be due to sensor noise or variables the sensor is unable to capture, resulting in uncertainty which cannot be reduced even if more data were to be collected. On the other hand, *epistemic* uncertainty accounts for uncertainty in the model parameters – uncertainty which captures our ignorance about which model generated our collected data. This uncertainty can be explained away given enough data, and is often referred to as *model uncertainty*. Aleatoric uncertainty can further be categorized into *homoscedastic* uncertainty, uncertainty which stays constant for different inputs, and *heteroscedastic* uncertainty. Heteroscedastic uncertainty depends on the inputs to the model, with some inputs potentially having more noisy outputs than others.

Chapter 2 makes the observation that heteroscedastic uncertainty is especially important for computer vision applications. This is because epistemic uncertainty is often explained away by the availability of large datasets or unsupervised learning (Section 4.6). However, heteroscedastic uncertainty, which is uncertainty which is inherent to the data, is important to understand. For stereo vision, one can imagine the types of appearances which should have large heteroscedastic uncertainty, such as reflective surfaces, transparent objects, texture-less areas and over exposed image regions.

Importantly, many applications of stereo vision require algorithms to run in real-time. One of the attractive properties of modelling heteroscedastic uncertainty is that we can formulate probabilistic deep learning models which run in real-time (Kendall and Gal, 2017). In contrast, epistemic uncertainty is often intractable without sampling (Gal, 2016), which drastically increases the computational requirements of the model.

4.5.2 Learning Heteroscedastic Uncertainty with Deep Learning

In this section, we describe how to modify the GC-Net model (Section 4.3) into a probabilistic deep learning architecture to estimate heteroscedastic uncertainty in addition to disparity.

Homoscedastic regression (introduced in Chapter 2) assumes constant observation noise, σ, for all input data. Heteroscedastic regression, on the other hand, assumes that observation noise can vary with input data (Le et al., 2005; Nix and Weigend, 1994). Heteroscedastic models can be useful for stereo vision because parts of the observation space might have higher noise levels than others, such as reflective or textureless regions. Heteroscedastic uncertainty can be learned as a function of the data (as described in Chapter 2). Therefore, variational inference is *not* performed over the weights, but instead we perform maximum a posteriori inference – finding a single value for the model parameters. This approach *does not* allow us to capture epistemic model uncertainty, but has the advantage of being real-time and does not require Monte Carlo sampling.

To train a heteroscedastic neural network, we need to infer the posterior distribution by learning a function, f, which maps a pair of stereo input images, I_L and I_R, to a disparity estimate, $\hat{d}$, and a measure of heteroscedastic uncertainty given by variance, $\hat{\sigma}^2$.

$$[\hat{d}, \hat{\sigma}^2] = f(I_L, I_R) \tag{4.3}$$

The GC-Net model is designed to output disparity estimates, $\hat{d}$, by forming a cost volume of H×W×D×1 dimensions, representing a cost for each disparity and pixel location. To adapt the architecture to regress an uncertainty measurement, $\hat{\sigma}^2$, we modify the final 3-D convolutional layer to regress a cost volume of H×W×D×2 dimensions. The final dimension's first value still represents the cost, with the second value representing uncertainty. To form the final output, the first values are put through a soft argmin layer across the disparity dimension (see Section 4.3.4 for a description) to form the disparity estimates, $\hat{d}$. We average the second values across the disparity dimension to form uncertainty estimates, $\hat{\sigma}^2$. Both these outputs are of size H×W.

In the following section we explain the loss function we use to learn to estimate disparity and heteroscedastic uncertainty. We use the losses derived in Section 2.4.3. We fix a Gaussian likelihood to model our heteroscedastic uncertainty. This induces a minimization objective given labelled output points, d:

$$\mathscr{L}_{heteroscedastic} = \frac{1}{2N} \sum_i ||d_i - \hat{d}_i|| \hat{\sigma}_i^{-2} + \frac{1}{2} \log \hat{\sigma}_i^2 \tag{4.4}$$

where N is the number of output pixels $\hat{d}_i$ corresponding to input image I, indexed by i. $\hat{\sigma}^2$ is the probabilistic neural network output for the predicted variance.

This loss consists of two components; the residual regression and an uncertainty regularization term. We do not need 'uncertainty labels' to learn uncertainty. Rather, we only need to supervise the learning to predict disparity. We learn the variance, $\hat{\sigma}^2$, implicitly from the loss function. The second regularization term prevents the network from predicting infinite uncertainty (and therefore zero loss) for all data points.

In practice, we train the network to predict the log variance, $\hat{s}_i := \log \hat{\sigma}_i^2$:

$$\mathscr{L}_{heteroscedastic} = \frac{1}{2N} \sum_i ||d_i - \hat{d}_i|| \exp(-\hat{s}_i) + \frac{1}{2} \hat{s}_i. \tag{4.5}$$

This is because it is more stable than regressing the variance, $\hat{\sigma}^2$, as the loss avoids a potential division by zero. The exponential mapping also allows us to regress unconstrained scalar values, where $\exp(-\hat{s}_i)$ is resolved to the positive domain giving valid values for variance.

In Table 4.5 and Table 4.6 we show that modelling heteroscedastic uncertainty improves performance on the SceneFlow and KITTI datasets (experimental details are described in Section 4.7). We observe that allowing the network to predict uncertainty allows it effectively to temper the residual loss by $\exp(-\hat{s}_i)$, which depends on the data. This acts similarly to an intelligent robust regression function. It allows the network to adapt the residual's weighting, and even allows the network to learn to attenuate the effect from erroneous labels. This makes the model more robust to noisy data – inputs for which the model learned to predict high uncertainty for will have a smaller effect on the loss.

The model is discouraged from predicting high uncertainty for all points – in effect ignoring the data – through the $\log \hat{\sigma}^2$ term. Large uncertainty increases the contribution of this term, and in turn penalizes the model. The model is also discouraged from predicting very low uncertainty for points with high residual error, as low $\hat{\sigma}^2$ will exaggerate the contribution of the residual and will penalize the model.

4.6 Unsupervised Learning

Stereo vision is a critical component for the success of many animal and robotic vision systems. However, there are significant challenges to develop robust stereo algorithms which can be trusted in the wild. Today, the best performing stereo methods are based on end-to-end deep learning (Kendall et al., 2017c; Luo et al., 2016; Mayer et al., 2016; Zbontar and LeCun, 2016). These methods are, however, very data-hungry, requiring large datasets to achieve top performance (Mayer et al., 2016). In the real world, getting access to large quantities of high quality labelled depth data is extremely challenging. For example, one of the most prominent stereo datasets, KITTI (Geiger et al., 2012), only contains a few hundred labelled training images. Getting labelled data is challenging because structured light sensors only work indoors and LIDAR sensors are expensive and produce sparse labels. Large synthetic datasets have been proposed (Mayer et al., 2016) however these do not always transfer to real world applications.

In the previous section we formulated a model to learn stereo disparities, with uncertainties, from labelled training data. In this section we show how to formulate stereo regression as an unsupervised learning problem using deep learning.

Recently, Garg et al. (Garg et al., 2016) showed how to learn monocular depth without labelled depth data by using photometric reprojection error. Their method was able to train on $10,000's$ of unlabelled KITTI images, much more than the 200 images with ground truth labels. This work has catalysed many of the recent advances in geometry with deep learning. For example, in monocular depth regression (Garg et al., 2016; Godard et al., 2017), optical flow (Jason et al., 2016; Ren et al., 2017), localisation (Kendall and Cipolla, 2017) and ego-motion (Zhou et al., 2017). This demonstrates that photometric reprojection loss has emerged as the dominant technique for learning geometry with unsupervised learning.

However, unsupervised learning methods do not perform as well as supervised regression (Garg et al., 2016; Kendall and Gal, 2017). This is because unsupervised learning suffers from the aperture problem – where correspondence is ambiguous due to lack of context (Hartley and Zisserman, 2000). For example in stereo vision, it is impossible to uniquely determine correspondences between featureless regions like blank walls or sky without considering the wider context (Hirschmüller and Scharstein, 2007). Methods which learn from photometric reprojection error typically enforce a smoothing prior which results in smooth prediction across these regions with no training signal (Garg et al., 2016; Godard et al., 2017; Zhou et al., 2017). However this also results in equally blurred regions where there is good structure and training signal in the data. For this reason, naively combining supervised depth regression and unsupervised learning with reprojection error typically worsens performance.

Unsupervised learning for *monocular* depth prediction using deep learning was first demonstrated by Garg et al (Garg et al., 2016). They showed how unlabelled stereo pairs can be used to train a monocular depth estimation model. In this work, we believe we are the first to demonstrate unsupervised deep learning for stereo vision. One of the reasons why this is now possible is because of new models which can be trained end-to-end to regress stereo disparity (Kendall et al., 2017c; Mayer et al., 2016). Previous deep learning models learned to classify disparities with a probability vector (Luo et al., 2016; Zbontar and LeCun, 2016). This representation classifies the output in discrete disparity steps making it is less suitable for unsupervised learning.

Unlike other unsupervised approaches to learning depth (Garg et al., 2016), we show that our probabilistic model does not need any smoothing terms in the loss. This is because the probabilistic regression loss is able to learn to attenuate noisy data points which previously would have required the smoothing regularization term during training. This results in a sharper depth prediction image which is able to capture thinner structures. Additionally, we explore the use of combining unsupervised and supervised learning as a semi-supervised loss when sparse data labels are available. We demonstrate an improvement in disparity accuracy over non-probabilistic deep learning baselines.

Using the end-to-end stereo regression model described in Section 4.3, we formulate an unsupervised regression loss. We can use the model's disparity estimate to align the left image, I_L, with the right stereo image, I_R. If the predicted disparity is equal to the true disparity then the corresponding pixel locations in the left and right stereo image should have the same intensity. This loss is known as the *photometric reprojection error* (Hartley and Zisserman, 2000).

More formally, we obtain this loss by resampling the right stereo image, I_R, with estimated disparities, $\hat{d}$. Using the sampling method proposed by spatial transformer networks (Jaderberg et al., 2015), this entire process is differentiable and able to be trained using back propagation. The photometric reprojection loss is given by:

$$\mathscr{L}_{photometric} = \frac{1}{N}\sum_{u,v} \left\| I_L(u,v) - I_R(u - \hat{d}_{u,v}, v) \right\|_1 \tag{4.6}$$

where $I(u,v)$ represents the pixel intensity of image I and $\hat{d}_{u,v}$ represents the estimated disparity, at pixel coordinate (u,v).

However, this loss alone is noisy. The photometric reprojection error function is non-informative in homogeneous regions of the scene (Hartley and Zisserman, 2000) – referred to as the aperture problem. For example, multiple disparities can generate equally good loss for repetitive or texture-less surfaces. For this reason, a prior is often applied to the loss function, typically a smoothing function, to overcome the aperture problem. Previous literature (Garg et al., 2016) uses some form of the total variation (TV) loss (Rudin et al., 1992):

$$\mathscr{L}_{smooth} = \frac{1}{N}\sum_{u,v} \left\| \nabla \hat{d}_{u,v} \right\|_1 \tag{4.7}$$

Combining this regularization loss with the photometric reprojection loss, yields the approach used by Garg et al. (Garg et al., 2016) for unsupervised monocular depth regression:

$$\mathscr{L}_{unsupervised} = \mathscr{L}_{photometric} + \gamma \mathscr{L}_{smooth} \tag{4.8}$$

with weight $\lambda = 0.01$. While this TV-regularized loss does indeed alleviate the aperture problem, and converge to a reasonable solution, it does over-smooth the output. In the monocular models, the predicted depth maps in the unsupervised case are over-smoothed and lose edge detail (Garg et al., 2016), compared to supervised regression models (Eigen et al., 2014). Figure 4.8c shows a similar result with stereo.

4.6.1 Attenuating the Loss with Heteroscedastic Uncertainty

In this section, we demonstrate that modelling heteroscedastic uncertainty can provide another solution to the aperture problem. Ideally, we would like to be able to learn to attenuate the photometric reprojection loss in areas of the image which are non-informative and suffer from the aperture problem. We would like to not regularize areas of the image which provide a good signal. This is in contrast to the TV-loss approach which smooths all areas equally.

Learning heteroscedastic uncertainty allows us to achieve this. We can interpret the loss in (4.4) as learned attenuation (Kendall and Gal, 2017). Because the residuals are annealed by the learned variance, $\hat{\sigma}^2$, the model has the flexibility to recognize noisy or non-informative areas of the signal, and attenuate the training loss accordingly.

We can apply the heteroscedastic regression loss in (4.4) to the photometric reprojection error loss in (4.5) as follows:

$$\mathscr{L}_{heteroscedastic\ unsupervised} = \frac{1}{2N}\sum_{u,v}\left\|I_L(u,v) - I_R(u-\hat{d}_{u,v},v)\right\|_1 \exp(-\hat{s}_{u,v}) + \frac{1}{2}\hat{s}_{u,v} \quad (4.9)$$

Table 4.5 and Table 4.6 compare the performance of supervised learning against unsupervised learning with our heteroscedastic model on the Scene Flow and KITTI datasets (experimental details are described in Section 4.7). Unsupervised learning is more effective on synthetic data, because in real world data it is susceptible to camera noise. However, these results show that we can learn monocular depth unsupervised without a regularization or smoothing term. Empirically, we demonstrate an improved performance using the learned attenuation loss in (4.9) over the regularized loss in (4.8). Qualitatively, we observe a sharper and more refined depth prediction.

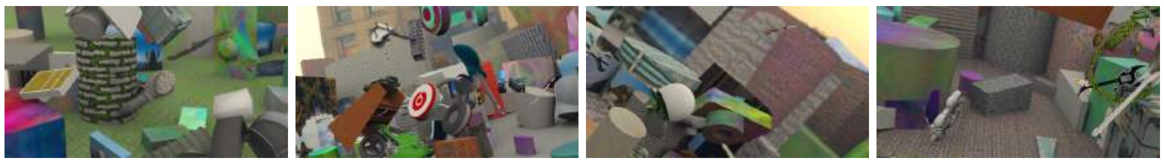
4.6.2 Semi-Supervised Learning

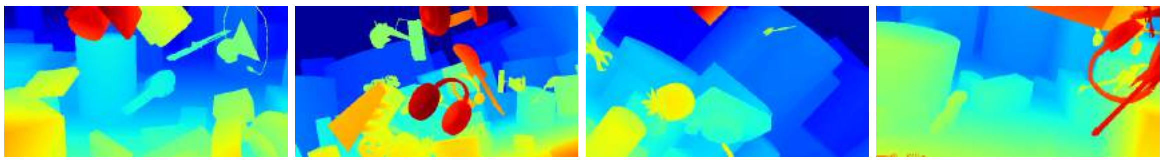
We can also train models in a semi-supervised fashion when we have some data with labels, and some data without labels. For example, the KITTI dataset provides 200 labelled training images, with some hundred thousand unlabelled frames. These losses can be combined:

$$\mathscr{L}_{semi-supervised} = \mathscr{L}_{supervised} + \beta\mathscr{L}_{unsupervised} \quad (4.10)$$

with weighting factor, $\beta = 0.1$. During training, we sample equally from the labelled and unlabelled datasets to form training mini-batches.

Table 4.5 compares models with no labels (unsupervised), various fractions of the labels, through to all labels (supervised). We observe an increase in performance as the amount of labels increases. Table 4.6 shows that the effect is less pronounced on KITTI with real-world data, however semi-supervised learning reduces the need for pretraining on the synthetic SceneFlow dataset.


(a) Input left stereo image.


(b) Ground truth disparity labels.

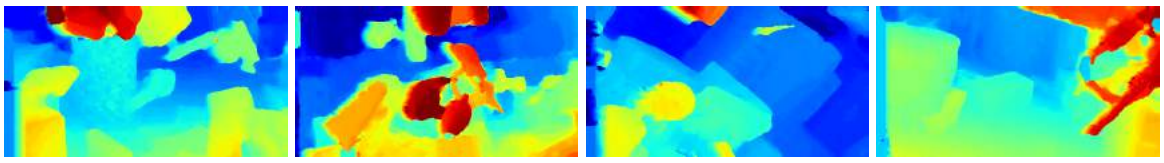
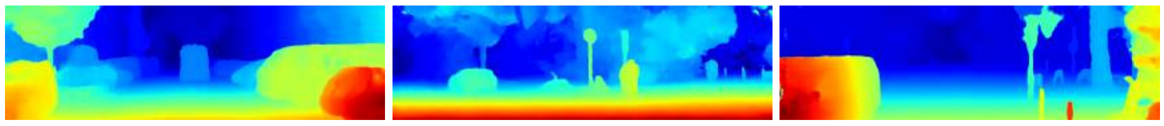

(c) Unsupervised regression with probabilistic modelling (this work).

Fig. 4.8 Qualitative results of our unsupervised probabilistic deep learning model on the Scene Flow dataset. We observe that our model can accurately learn the geometry of the scene, and learns accurate disparity estimates, without labelled training data.

(a) Input left stereo image.


(b) Estimated disparity.

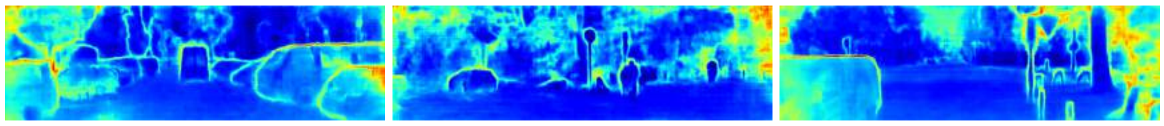

(c) Estimated heteroscedastic uncertainty.

Fig. 4.9 Qualitative results on the KITTI 2015 test set using supervised learning. Qualitatively, the heteroscedastic uncertainty captures noise due to occlusion boundaries, texture-less surfaces and erroneous regions of the disparity map.

Model Variant	Uncertainty	Unsupervised	Smooth Loss	Error	RMS	$> 1px$	$> 3px$	$> 5px$
Modelling heteroscedastic uncertainty improves performance:								
GC-Net Baseline (Kendall et al., 2017c)				2.51	12.4	16.9	9.34	7.22
+Heteroscedastic Uncertainty (this work)	✓			**2.16**	**10.5**	**15.0**	**8.76**	**6.85**
+Heteroscedastic (grey-scale input)	✓			2.84	14.9	16.7	9.82	7.62
Uncertainty can be used instead of a smoothing prior for unsupervised learning:								
+Unsupervised		✓		9.99	26.0	50.3	25.8	23.2
+Unsupervised+TV Loss		✓	✓	8.98	**23.3**	39.2	25.1	21.5
+Unsupervised+Heteroscedastic	✓	✓		8.99	24.9	**38.6**	25.0	21.4
+Unsupervised+Heteroscedastic+TV Loss	✓	✓	✓	**8.97**	24.5	40.4	**24.1**	**20.8**
Semi-supervised learning can achieve reasonable performance with few labels:								
+Semi-supervised (50% labels)+Heteroscedastic	✓	✓		2.99	15.1	19.8	12.1	9.86
+Semi-supervised (25% labels)+Heteroscedastic	✓	✓		3.97	15.6	21.5	13.2	10.8
+Semi-supervised (10% labels)+Heteroscedastic	✓	✓		4.77	16.2	26.3	15.8	12.7
+Semi-supervised (5% labels)+Heteroscedastic	✓	✓		5.44	16.7	27.2	17.3	14.4

Table 4.5 Results on the Scene Flow dataset (Mayer et al., 2016). Our probabilistic model improves mean disparity error by approximately 15% over the baseline by modelling heteroscedastic uncertainty. For unsupervised learning, we observe that our probabilistic loss is able to improve metrics which test fine-grained accuracy, showing that it can lead to accurate results. With semi-supervised learning, our model can still achieve accurate results, even with reduced availability of ground truth labels. By leveraging probabilistic modelling, we learn from both supervised and unsupervised losses without needing smoothing priors.

4.7 Benchmarking Uncertainty

In this section we quantitatively analyse the effectiveness of the measure of heteroscedastic uncertainty. We show that it is well calibrated and correlates strongly with metric error.

Firstly, Figure 4.10 compares precision recall curves for supervised and unsupervised models on the Scene Flow dataset. These curves plot the disparity accuracy against varying percentiles of uncertainty measurements. This shows that the estimate for uncertainty accurately reflects the prediction error of the heteroscedastic neural network model, because both curves are strictly decreasing functions. Practically speaking, we observe a very strong correlation between metric error and uncertainty.

Secondly, Figure 4.11 shows the calibration of our supervised and unsupervised models on the Scene Flow dataset. To form calibration plots for probabilistic regression models, we plot the frequency of residuals against probabilities given by the predicted heteroscedastic distribution. So, given N predicted disparities, $\hat{d_i}$, predicted variance, $\hat{\sigma}_i^2$, and ground truth labels, d_i, we can compute the residuals, $r_i = |d_i - \hat{d_i}|$. We then calculate the frequency of these residuals which lie within probability, p, using the predicted distribution (in our case, with the Gaussian distribution) with variance $\hat{\sigma}_i^2$. A perfectly calibrated model should behave such that $\text{freq}(p) = p$ which reasons that for a given probability tolerance of p, p predictions will be correct.

Probabilistic		Pretrained	Training Data		Disparity Error				
Modelling	Loss	SceneFlow	Labelled	Unlabelled	MAE	RMS	$> 1px$	$> 3px$	$> 5px$
✗	Supervised	✗	160	0	0.572	1.57	10.6	1.81	1.00
✗	Supervised	✓	160	0	**0.567**	1.44	10.2	1.67	0.89
✓	Supervised	✗	160	0	0.584	1.56	10.2	1.73	0.89
✓	Supervised	✓	160	0	0.579	1.56	**10.1**	**1.62**	**0.86**
	Unsupervised	✗	0	43,552	3.20	7.81	50.4	16.3	11.1
✓	Unsupervised	✗	0	43,552	2.55	6.50	45.1	15.0	9.44
✓	Unsupervised	✓	0	43,552	1.77	3.77	46.1	12.6	6.91
✗	Semi-supervised	✗	160	43,552	0.647	1.62	12.0	1.99	0.99
✓	Semi-supervised	✗	160	43,552	0.590	1.59	11.5	1.70	0.94
✓	Semi-supervised	✓	160	43,552	0.572	**1.43**	10.9	1.66	0.90

Table 4.6 **KITTI 2015 validation set results** (Geiger et al., 2012). The KITTI 2015 stereo dataset dataset contains 200 labelled training images, of which we randomly hold out 40 as a validation set, and report these results here. We also use the KITTI odometry dataset, which contains 43,552 unlabelled stereo frames. Modelling uncertainty improves results. Naively combining supervised and unsupervised training worsens performance. However using semi-supervised learning with our probabilistic model achieves better results, even without pretraining on the SceneFlow dataset (Mayer et al., 2016).

We observe that our models are well calibrated with mean square errors of 0.0060 and 0.0075 for supervised and unsupervised, respectively. The uncertainty for the supervised model is the raw, uncalibrated heteroscedastic uncertainty predicted by the model. For the unsupervised model, we need to calibrate it to estimate variance of the disparity prediction (measured in pixels), because the unsupervised heteroscedastic uncertainty estimates the variance of the photometric reprojection residuals (measured in pixel intensity). To achieve this, we apply a linear scaling factor which we tune empirically.

4.8 Conclusions

In this chapter, we investigated the problem of stereo vision and estimating disparity from a rectified pair of stereo images. We briefly summarise the main conclusions within the three main themes of this dissertation.

End-to-end learning. We propose a novel end-to-end deep learning architecture for stereo vision. It is able to learn to regress disparity directly, without any additional post-processing or regularization. We demonstrate the efficacy of our method on the KITTI dataset, setting a new state-of-the-art benchmark.

Geometry. We show how to efficiently learn context in the disparity cost volume using 3-D convolutions. We show how to formulate it as a regression model using a soft argmin layer. This allows us to learn disparity as a regression problem, rather than classification,

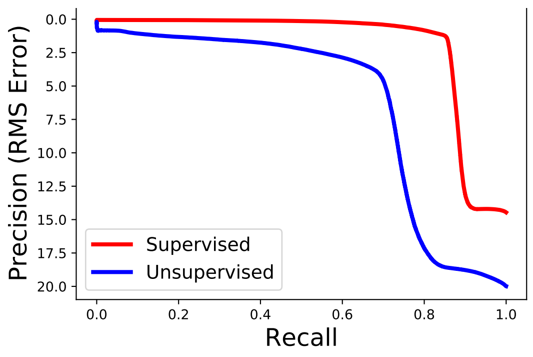


Fig. 4.10 Precision-recall plots for supervised and unsupervised models on the Scene Flow dataset. These curves demonstrate our heteroscedastic uncertainty can effectively capture accuracy, as precision decreases with increasing uncertainty.

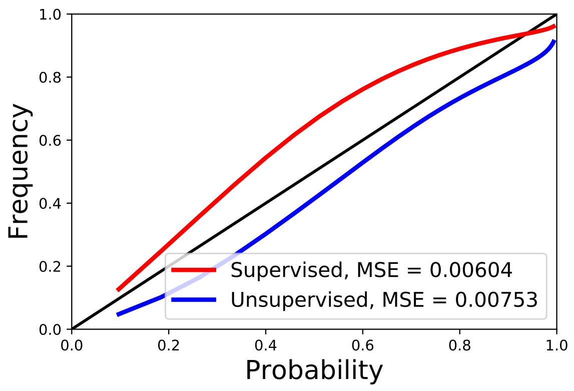


Fig. 4.11 Uncertainty calibration plots for supervised and unsupervised models on the Scene Flow dataset. This plot shows how well uncertainty is calibrated, where perfect calibration corresponds to the line $y = x$, shown in black. We observe our estimates of uncertainty are well calibrated with low mean squared error of 0.0060 and 0.0075 for supervised and unsupervised models, respectively.

improving performance and enabling sub-pixel accuracy. We demonstrate that our model learns to incorporate wider contextual information.

We show that we can use stereo geometry to learn depth with unsupervised learning. We formulate a loss using the photometric reprojection error. We show how to use uncertainty to combine supervised and unsupervised depth losses.

Uncertainty. We have shown how to form a stereo architecture with probabilistic deep learning. In summary, our approach is:

- Safe. Our probabilistic neural network can reason about heteroscedastic uncertainty, which we quantitatively show is well-calibrated.
- Accurate. Our approach improves over the performance of the baseline model.

- Fast. We do not require sampling for probabilistic neural network inference and can estimate uncertainty while adding negligible extra compute.
- Scalable. We show how to use unsupervised and semi-supervised learning to train our model with unlabelled stereo data, without requiring a smoothing or regularization loss term.

Chapter 5

Motion

5.1 Introduction

Contemporary computer vision models are extremely effective at understanding individual images. Given enough training data, deep convolutional neural network architectures can learn powerful representations using end-to-end supervised learning (He et al., 2016; Krizhevsky et al., 2012). For scene understanding, models can extract rich information at a pixel level such as semantics (Badrinarayanan et al., 2017; He et al., 2017; Long et al., 2015) and geometry (Eigen and Fergus, 2015; Garg et al., 2016). However, there is a limit to what we can infer from static images. To effectively understand complex and dynamic scenes we need to reason over video.

Designing models which can learn motion and reason over temporal sequences is of critical importance for computer vision systems. Video and temporal information allows models to encode motion, reason over occlusions and improve temporal consistency and stability. Additionally, many scene understanding systems are required to inform decision making processes. Usually, these decision-making processes cannot be made in temporal isolation and, particularly in robotics, knowledge of scene dynamics is essential.

State of the art video scene understanding systems typically process individual frames independently (Eigen and Fergus, 2015; Gadde et al., 2017; He et al., 2017; Patraucean et al., 2015; Valipour et al., 2017; Zhao et al., 2017; Zhou et al., 2017). Additional temporal reasoning can be performed by filtering (Miksik et al., 2013) or with graphical models (Chen and Corso, 2011; de Nijs et al., 2012; Hur and Roth, 2016; Tripathi et al., 2015), however these systems are unable to be trained jointly and have significantly higher computational requirements. Many papers have proposed to add temporal components such as recurrent neural networks (Hochreiter and Schmidhuber, 1997) on top of semantic segmentation encoders (Patraucean et al., 2015; Valipour et al., 2017). However, to date, none have

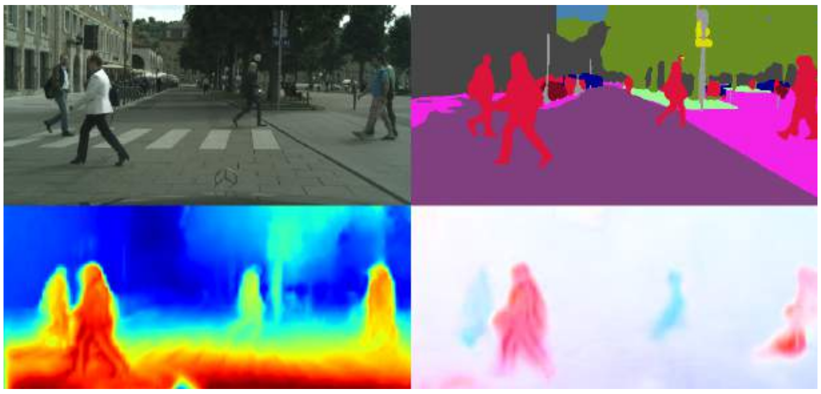


Fig. 5.1 **Video scene understanding.** Our model jointly learns to estimate scene motion (bottom right), depth (bottom left) and semantic segmentation (top right), in addition to estimating ego-motion over video input. We learn semantics with supervised learning and are able to learn geometry and motion using self-supervised learning, without labelled training data.

been able to demonstrate improvement over equivalent per-frame models at scale. In this chapter, we observe the same result; naive sequence-to-sequence modelling from video with recurrent convolutional models harms semantic segmentation performance. We argue that this is because the recurrent units propagate their state from a static location in pixel space over time. However, in practice objects will move significantly in pixel location between consecutive frames due to camera ego-motion and scene dynamics. These models are not aware of this geometry and motion.

In this chapter, we introduce an end-to-end deep learning architecture for video semantic segmentation. We jointly learn optical flow, ego-motion and depth with self-supervised learning and semantic segmentation with supervised learning. Example outputs from the model are shown in Figure 5.1. We propose a motion-aware gated recurrent unit (motion-GRU) which is able to use motion and geometry to account for ego-motion and scene dynamics when propagating state information temporally. Since labelling video frames with semantic labels is prohibitively expensive, we show that learning geometry with unsupervised learning can provide a powerful dense training signal for video. In summary, the novel contributions of this chapter are;

1. showing how to account for motion when propagating recurrent states over time by using motion recurrent units,
2. demonstrating how to provide training signal for each video frame with self-supervised learning of motion and geometry,
3. using temporal augmentation of video data in order to learn a stable representation over time.

In the remainder of this chapter, we discuss these ideas and show how to train large temporal models over long video sequences at scale. We show that our model is able to perform video semantic segmentation with higher accuracy and temporal consistency than equivalent per-frame models.

5.2 Video Scene Understanding

Scene understanding is the task of extracting information about the objects and their contextual relationships from the observed environment. Supervised deep learning is a very effective solution to this problem from single images. In particular, many deep convolutional encoder-decoder architectures have been shown to produce accurate and real-time solutions to problems such as semantic segmentation, optical flow, depth and geometry estimation.

Semantic segmentation is the task of estimating the semantic class of each pixel in an image. State of the art models use supervised deep learning (Badrinarayanan et al., 2017; Long et al., 2015), benefiting from residual architectures (He et al., 2016; Huang et al., 2017). Recent work has focused on improving the receptive field of features and providing them with more context for semantic reasoning, for example using dilated convolutions (Yu and Koltun, 2016) and pyramid spatial pooling (Zhao et al., 2017). We have also seen semantic segmentation combined with other tasks, such as instance segmentation (He et al., 2017) and geometry (Section 2.5) in multi-task learning settings. Probabilistic modeling and Bayesian deep learning has also been used to understand model uncertainty in scene understanding algorithms (Chapter 2), improving safety in practical applications.

Depth and geometry models use similar architectures to semantic segmentation, but in a regression setting. Typically they estimate per-pixel metric depth or surface normals (Eigen and Fergus, 2015). In (Ummenhofer et al., 2017) the authors learn geometry and motion using supervised learning. Deep learning models can also be trained using unsupervised learning without explicit depth labels using reprojection into stereo (Garg et al., 2016) or temporal (Zhou et al., 2017) frames. In Chapter 4 we also show that end-to-end deep learning can be used to estimate depth from stereo vision.

The problem of estimating motion in video is known as optical flow. Specifically, optical flow is a measurement of each pixel's 2D motion between subsequent video frames. It captures motion due to the ego-motion of the camera (which can be represented by the fundamental matrix for two frames (Hartley and Zisserman, 2000)) and motion due to scene dynamics. State-of-the-art real-time models use deep learning (Dosovitskiy et al., 2015; Ilg et al., 2016) and are often trained on large synthetic datasets.

Video scene understanding, and video semantic segmentation, requires these tasks to be performed over video. Video and temporal information allows models to encode motion, reason about occlusion and improves temporal consistency and stability. Initial methods approached temporal modelling as a filtering problem (Miksik et al., 2013). However, the most popular approach to date has been to construct large graphical models that connect different video pixels to achieve temporal consistency across frames (Chen and Corso, 2011; de Nijs et al., 2012; Hur and Roth, 2016; Tripathi et al., 2015). For example,(Chen and Corso, 2011) used dynamic temporal links between the frames but optimized for a 2D CRF with temporal energy terms. A 3D dense CRF across video frames is constructed in (Tripathi et al., 2015) and optimized using mean-field approximate inference. In (Hur and Roth, 2016) a joint model to infer flow and semantic segmentation was proposed.

Geometry, in the form of structure from motion, has been used to aid video segmentation with random forests (Brostow et al., 2008) and Markov random fields (Tighe and Lazebnik, 2013). More recent works look at jointly modelling 2D semantics and 3D reconstruction of scenes from video (Kundu et al., 2014; Sengupta et al., 2013).

Deep learning approaches for understanding video have been largely constrained to video level understanding tasks. For example, 3D convolutions have been used for off-line video classification (Karpathy et al., 2014) or activity recognition (Ji et al., 2013). Additionally, LSTMs have been used for video recognition and captioning (Donahue et al., 2015). However these tasks require representations at a video level, and do not need to consider scene geometry and dynamics like in this work.

Single image semantic segmentation systems have also been adapted to video segmentation. For example, (Gadde et al., 2017; Zhu et al., 2017) both learn a representation warping module to form a two-frame video segmentation model. *Clockwork* convolutional networks (Shelhamer et al., 2016) and (Zhu et al., 2017) both propose a way of reducing computation for video segmentation by reusing representations from single images across time.

To date, there have been a few proposals for video segmentation models over long video sequences with deep learning (Patraucean et al., 2015; Valipour et al., 2017). They typically append a RNN or LSTM layer after a convolutional encoder-decoder model. However, so far every model has decreased model performance, compared to single frame non-video baseline models (Patraucean et al., 2015; Valipour et al., 2017). We believe this work is the first to demonstrate an improvement in performance using end-to-end deep learning over many-frame video sequences.

Video segmentation has also been considered in an off-line setting. This is a very different problem setting to what we consider in this chapter, as algorithms have access to future video frames, and are not constrained to real-time inference. For example, these methods can afford

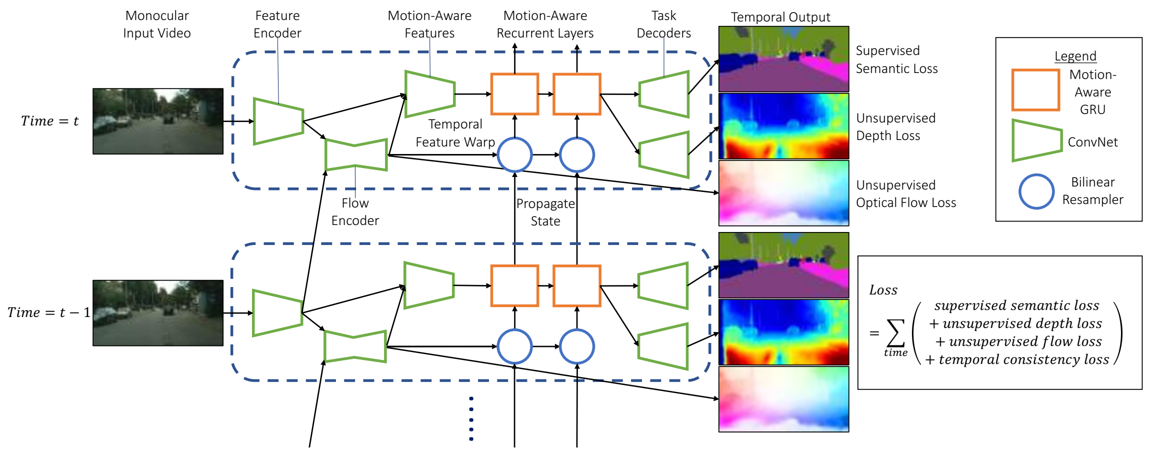


Fig. 5.2 **Overview of our video scene understanding architecture.** The model efficiently propagates information over time, outputting semantic segmentation, depth, ego-motion and optical flow estimates from monocular video.

to use computationally expensive spatial-temporal CRFs (Kundu et al., 2016). In computer graphics and animation this is known as rotoscoping (Miksik et al., 2017). Other related works of note are future video frame prediction (Luc et al., 2017) and label propagation (Budvytis et al., 2010).

Other related work is in video object segmentation and video motion segmentation. These works are largely driven by datasets like DAVIS (Perazzi et al., 2016). This problem focuses on segmenting a single object, or moving objects (Tokmakov et al., 2017; Tsai et al., 2016; Vertens et al., 2017) in video. Unlike our work, explicit knowledge of semantics is not required. Rather, algorithms commonly use one-shot learning for masks and propagate them through video (Voigtlaender and Leibe, 2017). Similar to scene understanding, the best approaches today segment single frames independently (Caelles et al., 2017; Khoreva et al., 2016).

5.3 Temporal Scene Understanding Model

In this section we describe our model for video scene understanding, motion-aware recurrent neural network and the video loss functions.

5.3.1 Architecture

Consider a sequence of images, $I_0, I_1, ..., I_t$. We wish to produce a representation of the scene's semantics, motion and geometry in real-time for time-step, t. In order to reduce the high dimensionality of the input, we pass the image through a deep convolutional encoder to obtain a powerful feature representation,

$$\mathbf{x}_t = encoder(I_t). \tag{5.1}$$

We choose to use the E-Net architecture for our encoder (Paszke et al., 2016) because it achieves good segmentation performance while being computationally light-weight.

Using the feature representations from the previous and current time steps, we can then learn to regress optical flow (the motion of features between frames). We feed both these features through a network inspired by FlowNet simple (Dosovitskiy et al., 2015):

$$\mathbf{y}^{flow}_{(t-1)\to t} = flownet(\mathbf{x}_t, \mathbf{x}_{(t-1)}). \tag{5.2}$$

This results in an estimate of optical flow for each feature from the encoder, $\mathbf{y}^{flow}_{(t-1)\to t}$. We save computation by estimating flow using features shared by the semantic encoder, rather than the raw image. This is because initial filters in flow and semantic encoders will be very similar (Zeiler and Fergus, 2014). We also improve the representation by leveraging multi-task learning (Caruana, 1998).

In order to improve the feature representation for the current frame, and to incorporate motion, we concatenate the flow feature map (from the layer immediately prior to the flow regression), $\mathbf{y}'^{flow}_{(t-1)\to t}$, and the image features at the current time $\mathbf{x}_t$. We pass these features through some convolutional layers to learn a representation with motion-aware features:

$$\mathbf{z}_t = features(\mathbf{x}_t, \mathbf{y}'^{flow}_{(t-1)\to t}) \tag{5.3}$$

We implement $features$ as two residual layers (He et al., 2016) with feature size 64.

This results in a set of features, $\mathbf{z}_t$, at each time-step which encode image semantics and first order motion. Note that these features only have access to input signal from time t and $t-1$. We now would like to learn long term dynamics over a sequence, to model higher-order motion and improve temporal consistency. A common approach to sequence modelling is to use a recurrent neural network (RNN). RNNs use an internal state to propagate memory over sequences. In particular, long-short-term-memory (LSTMs) units (Hochreiter and Schmidhuber, 1997) have been shown to more effectively learn long dependencies. Gated recurrent units (GRU) (Cho et al., 2014) are a variant of LSTMs which retain the gated structure, but have a simpler internal structure making them faster to train.

Many papers have tried to place recurrent modules over features from a convolutional encoder (Patraucean et al., 2015; Valipour et al., 2017). However, every attempt has not improved over per-frame baseline models. We make the observation that each recurrent module propagates its state forward in time for each spatial location in its feature map. However, objects in the scene rarely remain in the same location in pixel coordinates across video frames. This is due to the dynamic nature of real world environments, and ego-motion of the camera. This poses a problem for the recurrent state. This is because the state at time $t-1$, for a given pixel coordinate (u,v), is unlikely to contain information about the same object which likely moved to a new pixel coordinate (u',v') at the next time-step, t. Our hypothesis is that the propagation of misaligned features over time causes this decrease in performance with temporal models.

We propose to solve this problem by introducing a motion-aware GRU module. We know the motion of each pixel from our estimate of the optical flow, $\mathbf{y}^{flow}_{(t-1)\to t}$. Therefore we can align features over time, such that they account for motion of the object in pixel space, by warping the recurrent state and features using the estimate of optical flow. The full equations of our motion-aware GRU are;

$$\mathbf{g}_t = \text{sigmoid}(W_g * \mathbf{z}_t + U_g * \mathbf{h}^{warped}_{t-1} + b_g) \tag{5.4}$$

$$\mathbf{r}_t = \text{sigmoid}(W_r * \mathbf{z}_t + U_r * \mathbf{h}^{warped}_{t-1} + b_r) \tag{5.5}$$

$$\widetilde{\mathbf{h}}_\mathbf{t} = \tanh(W_h * \mathbf{z}_t + U_h * \mathbf{r}_t \cdot \mathbf{h}^{warped}_{t-1}) \tag{5.6}$$

$$\mathbf{h}_t = (1-\mathbf{g}_t) \cdot \mathbf{h}^{warped}_{t-1} + \mathbf{g}_t \cdot \widetilde{\mathbf{h}}_t, \tag{5.7}$$

with convolutional weights, W,U, and biases, b. We use a convolutional kernel size of 3 and add batch-normalisation after each convolution, which we omit for clarity. To form a motion-GRU, our model is going to use its estimate of optical flow to propagate the features temporally over the video sequence, such that the features correspond in pixel space. We do this by resampling features according to the optical flow map using bilinear interpolation, inspired by spatial transformer networks (STNs) (Jaderberg et al., 2015). We define this function which warps the recurrent state, $\mathbf{h}_{(t-1)}$, from time $t-1$ to time t, by resampling using the optical flow vectors, $\mathbf{y}^{flow}_{(t-1)\to t}$, as follows:

$$\mathbf{h}^{warped}_t = warp(\mathbf{h}_{(t-1)}, \mathbf{y}^{flow}_{(t-1)\to t}). \tag{5.8}$$

The motion-GRU then uses $\mathbf{h}^{warped}_t$ rather than $\mathbf{h}_t$ as input. We backpropagate gradients smoothly by implementing this warping with bilinear interpolation (Jaderberg et al., 2015). We can stack multiple GRU layers in series to model long term temporal information. Note, we can also form motion-RNNs and motion-LSTMs with the same change. We choose GRUs because of their ability to gate inputs and simple implementation.

Warping the state vectors between time-steps attempts to align features with their correspondences. However, due to occlusion, it is not possible to achieve this completely, as some features will appear and disappear with motion. We rely on the model to learn to understand this phenomena.

Finally, we learn separate decoders to estimate pixel-wise semantic segmentation and depth regression from the output of the motion-aware GRUs, $\mathbf{h}_t$.

$$\mathbf{y}_t^{class} = decoder_{class}(\mathbf{h}_t), \tag{5.9}$$

$$\mathbf{y}_t^{depth} = decoder_{depth}(\mathbf{h}_t), \tag{5.10}$$

$$\mathbf{y}_t^{egomotion} = decoder_{egomotion}(\mathbf{h}_t), \tag{5.11}$$

where class and depth decoders are comprised of a single 3×3 convolutional layer with feature size 32 followed by a 1×1 convolutional layer regressing the required output dimensions (1 for depth and number of semantic classes for class). The ego-motion decoder uses global average pooling to pool the features, followed by a fully connected layer with feature size 32 and a final fully connected layer regressing a 6-DoF ego-pose vector, similar to (Zhou et al., 2017).

All layers are followed by batch normalisation and ReLU non-linearities, except for the prediction layers. For the depth prediction layer, we constrain the output to a reasonable positive range with $1/(\alpha * sigmoid(x) + \beta)$, with $\alpha = 0.99$ and $\beta = 0.01$ to constrain the depth values to reasonable limits.

This model can be run continuously over video in real-time to estimate per-frame optical flow, depth and semantic segmentation. This model is efficient and real-time because each image is only encoded once, and recurrent states are propagated through time. It is motion-aware from the two-frame optical flow feature input, which provides first-order motion information, and temporally aligns features. The recurrent state memory allows the model to remember higher order motion and reason over multiple views and deal with occlusion.

5.3.2 Loss Function

In this section, we explain how to learn semantic segmentation using supervised learning and optical flow, ego-motion and depth with self-supervised learning[1]. The geometric outputs can

[1]We refer to this as self-supervised learning because we train the model with a regression target using our knowledge of geometry, in the absence of human-annotated labels. However, this has also been referred to as *unsupervised learning* (Garg et al., 2016; Godard et al., 2017; Zhou et al., 2017) or *natural supervision* (Koltun, 2017) because of the lack of labels and the inability to evaluate the performance of the model against the data.

be considered as intermediate auxiliary outputs. They assist learning semantic segmentation by making our model motion and geometry aware, improving the representation.

Large semantic segmentation datasets are available with thousands of densely labelled images (Cordts et al., 2016; Lin et al., 2014). Obtaining accurate depth and flow labels is much harder. Therefore, we use a self-supervised learning regime to learn flow and depth, based on the photometric reprojection error and multi-view geometry. We can learn depth by learning correspondence to stereo images (Garg et al., 2016) or subsequent temporal frames (Zhou et al., 2017). We can learn flow by learning correspondences between temporal frames.

One possibility to learn depth is to use the estimated disparities, $\mathbf{y}_{depth}$, from input image, I_t, to warp the corresponding stereo image, $I_{t,stereo}$. We train on the photometric reconstruction error to form a loss function for depth estimation:

$$\mathscr{L}_{stereo\ depth,\ t} = \frac{1}{N} \sum_{i,j} |I_t(i,j) - I_{t,stereo}(i, j + \mathbf{y}_{depth,t}(i,j))|, \tag{5.12}$$

where indices i and j correspond to pixel locations in the image. The loss is averaged across all temporal frames and pixels, N. Resampling pixel indices is performed by bilinear interpolation (Jaderberg et al., 2015). Minimising this loss causes the model to learn disparity. We omit the conversion from disparity to metric depth for clarity.

Alternatively, if stereo imagery is not avaliable, we can learn depth and ego-motion from monocular video (Zhou et al., 2017). We assume the camera's intrinsic matrix, K, is known. To estimate pixel correspondences in monocular video, we first back-project each homogeneous pixel coordinate, p_{ijt}, using the predicted depth:

$$X = \mathbf{y}_{depth,t}(p_{ijt}) K^{-1} p_{ijt}. \tag{5.13}$$

We then transform these coordinates with the predicted ego-motion camera transformation from the current frame to the previous frame using the pose vector, $\hat{y}_t^{pose}$ which is represented as a 3×4 projection matrix, $\hat{T}_{t \to (t-1)}$,

$$\hat{p}_{ij(t-1)} = K \hat{T}_{t \to (t-1)} X. \tag{5.14}$$

We use these coordinates to resample I_{t-1} and train on the photometric loss:

$$\mathscr{L}_{mono\ depth,\ t} = \frac{1}{N} \sum_{i,j} |I_t(p_{ijt}) - I_{(t-1)}(\hat{p}_{ij(t-1)})|, \tag{5.15}$$

to learn geometry from monocular video. Although, this only allows us to estimate depth and ego-motion up to some unknown scale-factor.

To learn optical flow, we can warp the previous frame to the current frame, except now our disparity estimates have two-degrees of freedom to represent optical flow. Again, the loss is formed using the temporal, photometric reconstruction error:

$$\mathscr{L}_{flow,\ t} = \frac{1}{N}\sum_{i,j} |I_t(i,j) - I_{(t-1)}(i + \mathbf{y}_{flow\ i,t}(i,j), j + \mathbf{y}_{flow\ j,t}(i,j)))|. \tag{5.16}$$

In (5.12), (5.15) and (5.16) the photometric reprojection error function is non-informative in homogeneous regions of the scene – referred to as the aperture problem (Hartley and Zisserman, 2000). For example, in repetitive or texture-less surfaces, multiple disparities can generate equally good losses. Only when considering wider context can we reason about these regions with sufficient information. For this reason, a prior is often applied to the loss function, typically a smoothing function, to overcome the aperture problem. Previous literature uses some form of the total variation (TV) loss (Rudin et al., 1992) and minimises the norm of the gradients for the predicted depth maps (Garg et al., 2016; Zhou et al., 2017). In this work we regularise the output of both flow and depth estimates with the L_1 norm of the second order gradients:

$$\mathscr{L}_{smooth,\ t} = \frac{\lambda}{N}\sum_{i,j} |\nabla^2 \mathbf{y}_{depth,t}| + |\nabla^2 \mathbf{y}_{flow,t}|, \tag{5.17}$$

Following (Zhou et al., 2017), we use the weighting factor of $\lambda = 0.5$.

We learn semantic segmentation with the cross entropy loss between semantic segmentation prediction and ground truth label for all semantic classes, c. Typically, frames in video sequences are only sparsely labeled (Cordts et al., 2016). We therefore average the loss across only pixels and frames with ground truth labels, ignoring all others. However, cross entropy treats each pixel independently in space and time. Additionally, we would like to encourage the network to learn a temporally consistent solution. We define a temporal consistency loss, which penalises temporal differences in segmentation prediction, after accounting for motion using optical flow:

$$\mathscr{L}_{consist,\ t} = \frac{1}{N}\sum_{i,j} |\mathbf{y}_{class,t}(i,j) - \mathbf{y}_{class,(t-1)}(i + \mathbf{y}_{flow\ i,t}(i,j), j + \mathbf{y}_{flow\ j,t}(i,j))|, \tag{5.18}$$

where $\mathbf{y}(i,j)$ represents the semantic logit at pixel location i,j. The intuition here is that, if the optical flow estimate is correct, then corresponding pixel locations should represent the same semantic class.

The total loss combines the individual losses: depth (5.12), flow (5.16), classification and consistency (5.18).We average the loss over all timesteps. Because we are optimising multiple losses, this is a multi-task learning problem (Caruana, 1998). A naive choice of overall loss would be to sum each individual loss:

$$\mathcal{L}_{total} = \frac{1}{T}\sum_{t=0}^{T}\mathcal{L}_{class,t} + \mathcal{L}_{flow,t} + \mathcal{L}_{depth,t} + \mathcal{L}_{smooth,t} + \mathcal{L}_{consist,t}. \tag{5.19}$$

In Chapter 2 we showed that the choice of weights between the losses in deep learning models has a very strong effect on overall performance. We use the method from Section 2.5 to automatically weigh the losses by learning weights automatically by considering each task's uncertainty:

$$\mathcal{L} = \sum_{\mathcal{L}_i \in [\mathcal{L}_0, \mathcal{L}_1, \ldots, \mathcal{L}_N]} \frac{1}{2\sigma_i^2}\mathcal{L}_i + \log\sigma_i, \tag{5.20}$$

where we estimate the task (homoscedastic) uncertainty of each loss, i, by optimising a variable, σ_i. We *do not* learn a weight for the consistency and smoothing regularisation losses. This formulation decreases the weight for tasks which have higher variance in their residuals. This is important because we are learning outputs with many different units: semantics as probability distributions, depth as a distance and optical flow in pixels. This formulation is able to balance the losses and learn a representation which is effective for each task: flow, depth and semantic segmentation.

5.4 Experiments

In this section we perform a number of experiments to evaluate the importance of compensating for motion in our architecture in addition to other design decisions. All experiments include all features of the model described in Section 5.3, unless otherwise stated. By default we train over four time-steps, with a warm-up of four steps. The entire model is randomly initialised before training.

Dataset. We use the CityScapes dataset for all experiments (Cordts et al., 2016) which is a road scene understanding dataset. The CityScapes dataset was taken from a vehicle in various locations around Germany. It provides 5000 stereo images with 1024×2048 resolution (of which $2,975$ are for training) with dense semantic segmentation labels. It also provides 19 preceding and 10 future frames around each labelled frame. In order to train our video model in GPU memory over long sequences, we sub-sample CityScapes images by a factor of two, to 1024×512 resolution for training and testing. The original E-Net, which we use as our encoder, is also trained on this resolution.

The CityScapes dataset provides disparity labels from a stereo algorithm (Cordts et al., 2016) which we use as pseudo-ground truth to compute depth error metrics. We measure depth error in disparity, converting from depth using the provided CityScapes stereo baseline. We use the output of FlowNet 2.0 (Ilg et al., 2016) as a reference measure of optical flow, and the output of ORB-SLAM (Mur-Artal et al., 2015) for ego-motion, in order to evaluate our self-supervised predictions. We evaluate ego-motion with absolute trajectory error over a 5-frame sequence.

Metrics. We evaluate semantic segmentation performance with three metrics. First, mean intersection over union (IoU) measures the pixel wise semantic performance. It penalises false negatives and false positives. We also evaluate the *temporal consistency* (TC) of the predicted semantic labels to measure if our video model produces more consistent and coherent results. We define temporal consistency as the percentage of pixels along an optical flow track which have the same class prediction. We evaluate depth metrics using inverse depth error in pixel disparity, scaled by the stereo baseline. We evaluate optical flow metrics in pixels and ego-motion error in meters.

Optimisation. For all experiments, we use a batch size of sixteen on a machine with four NVIDIA Titan Xp GPUs. We train with stochastic gradient descent, with 0.9 momentum and a base learning rate of 0.1. We decay this learning rate using a polynomial schedule defined by (Chen et al., 2015a) over $100,000$ training iterations. With these learning hyper-parameters our per-frame baseline of E-Net significantly outperforms the author's original implementation of the architecture (Paszke et al., 2016).

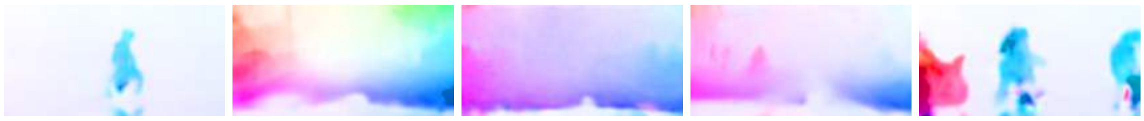
We describe three main challenges for training video scene understanding models, which have largely prevented prior models achieving superior performance to date. We discuss three tricks which address these challenges which we show are critical to achieving good performance when training video scene understanding models.

5.4.1 Trick #1: Account for Motion when Propagating States.

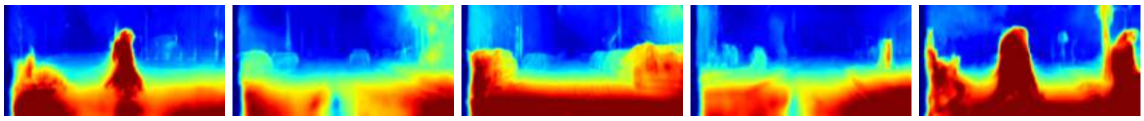
In Section 5.3 we presented the hypothesis that naive temporal convolutional models using recurrent units (e.g. (Patraucean et al., 2015; Valipour et al., 2017)) degraded performance compared to equivalent per-frame models, because they are unable to account for motion. In this section, we show that our motion-GRU improves performance over a per-frame baseline. We use self-supervised learning with photometric reprojection error to learn a representation of optical flow.

We also found that it was beneficial to initialise the recurrent connections such that they are close to identity connections (He et al., 2016), similar to the work of (Gadde et al., 2017). To achieve this, we offset the bias in the motion-GRU gates to 4.0 so that the initialisation of

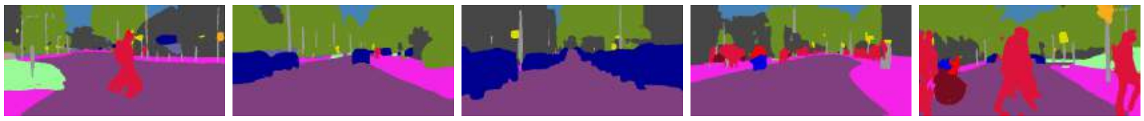
(a) Input test images from video



(b) Optical flow estimation



(c) Depth estimation



(d) Semantic segmentation

Fig. 5.3 **Qualitative results** for video scene understanding using our model. We observe good performance at semantic segmentation, optical flow, ego-motion and depth estimation. Examples are shown from validation sequences.

the sigmoid function evaluates to near 1.0. Consequently, the state will be almost entirely forgotten, and the signal almost entirely propagated from the current time-step. The model incrementally learns to incorporate temporal information by back-propagation.

Table 5.1 confirms that adding a standard GRU degrades both segmentation accuracy and temporal consistency, compared to a per-frame baseline. This experiment also shows a significant improvement in performance when using our motion-GRU. This suggests that the motion-GRU is able to effectively propagate information temporally, by aligning features and accounting for dynamic scene motion. In addition, our temporal consistency loss further improves performance by explicitly enforcing temporal consistency during training.

5.4.2 Trick #2: Learn Stable Representations with Temporal Data Augmentation.

In order to train effective video models with good temporal consistency, it is important to train over long video sequences. It is essential to have a number of frames so that the model can learn higher order dynamics and motion. Previous work only considers training over small sequences, often with only two frames (Gadde et al., 2017), which makes it impossible for the model to learn higher-order motion.

However, we find two problems when attempting to train on long video sequences: (1) training over a fixed sequence length results in an unstable model, and (2) we cannot back-

Recurrent Model	Segmentation IoU	Segmentation Consistency	Depth Err. (px)	Flow Err. (px)
per-frame baseline (no motion)	63.9%	82.3%	11.2	-
GRU	63.5%	87.4%	9.3	14.7
motion-GRU	65.6%	91.9%	**9.1**	12.3
motion-GRU + consistency loss	**65.9%**	**94.2%**	9.4	**12.1**

Table 5.1 **Importance of modelling motion.** We compare our motion-GRU to baseline models and quantify the improvement due to modelling motion and increasing sequence length. Naively adding recurrent units to a segmentation model degrades performance. The motion-aware recurrent architecture improves performance. Our temporal consistency loss improves results further.

propagate through long sequences given our available GPU memory. To overcome these problems, we use *temporal data augmentation*.

To explain the first problem, let us consider training a model using sequences of fixed length N, with only the frame at $t = N$ labelled. We observe the model learns to develop a representation which is able to make a prediction very effectively at time $t = N$. However, leading up to this time, and beyond if we run the recurrent model continuously, the state becomes unstable and diverges. Our intuition is this is because the model has not been taught to make predictions continuously, it is only optimised to predict at time $t = N$. Therefore to address this issue, we propose to randomly vary the sequence length during training, and the position of the labelled frame, which we term *temporal data augmentation*.

Regarding the second issue, back-propagating gradients over video requires significant amounts of GPU memory as feature representations from previous time-steps must be retained in memory. To overcome this, we propose to forward propagate over long (potentially infinite) video sequences, but only back-propagate over a shorter fixed time horizon. We refer to the forward-only data as a *warm-up* as it produces a recurrent state which is used to train over the remaining sequence data. This reduces GPU memory requirements as we can discard feature activations from memory for all time-steps during the warm-up phase. Our final approach is to train using a different random warm-up length for each mini-batch during training.

In Table 5.2, we compare the effect of training over different length time-steps. We observe an improvement with training over longer time sequences, including with a random warm-up when the sequence is too long to back-propagate in memory.

5.4.3 Trick #3: Provide a Dense Loss at Each Time-Step.

Finally, we find that it is important to provide a rich training signal over large amounts of data to learn video scene understanding models. However, labelling frames densely in video

	Segmentation		Depth	Flow
Time-steps	IoU	Consistency	Err. (px)	Err. (px)
1	63.9%	82.3%	11.2	-
2	64.6%	89.1%	9.8	14.5
4	65.9%	92.3%	9.5	12.7
4 + 4 random warm-up	**65.9%**	**94.2%**	**9.4**	**12.1**

Table 5.2 **Method and sequence used for training video models** has a massive impact on model performance. We observe training over longer sequences improves performance. In addition to improving performance, we also find that randomly varying training sequence length forces the model to learn a temporally stable representation, allowing the model to be run recurrently for indefinite time-sequences.

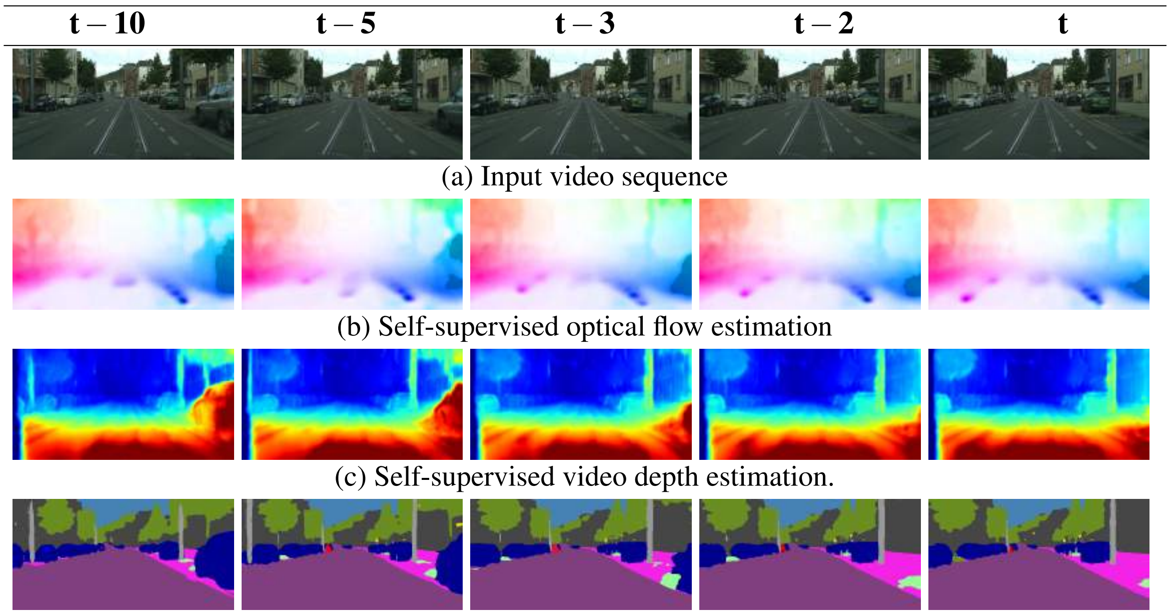
	Segmentation		Depth	Flow	Egomotion
Tasks	IoU	Consistency	Err. (px)	Err. (px)	Err. (m)
segmentation	63.8%	82.7%	-	-	-
segmentation+flow	65.1%	91.3%	-	14.1	-
segmentation+flow+mono depth	65.6%	93.8%	21.8**	12.3	0.39**
segmentation+flow+stereo depth	**65.9%**	**94.2%**	**9.4**	**12.1**	-

Table 5.3 **Benefit of self-supervised and multi-task learning.** These experiments illustrate the improvement our approach receives from multi-task learning. Learning representations of motion and geometry significantly improves performance of semantic segmentation. **We can only resolve the monocular depth and ego-motion prediction up to some scale-ambiguity. To compute accuracy, we optimise the scale factor using the validation data.

is prohibitively expensive. Therefore our third trick is to utilise self-supervised learning to provide a training signal for each frame.

We learn optical flow using self-supervised learning with the loss in (5.16). We show results for training depth using stereo reprojection error with the loss given by (5.12). However, since this requires stereo data, we also show results using the loss in (5.15) to learn depth and ego-motion using monocular temporal reprojection. For all models, we use the probabilistic multi-task loss from Chapter 2 to balance the individual task losses.

In Table 5.3 we train models with different combinations of task losses, over a sequence of four frames and a warm-up of four frames. We observe a large quantitative improvement in results by modelling geometry. Learning with motion and depth features significantly improves performance further. Providing stereo data marginally outperforms monocular depth supervision. However, the monocular depth and ego-motion model is competitive, if only monocular data is available for training.



(a) Input video sequence

(b) Self-supervised optical flow estimation

(c) Self-supervised video depth estimation.

(d) Video semantic segmentation.

Fig. 5.4 **Video semantic segmentation over a 10 frame sequence.** We observe that by learning motion and geometry and leveraging motion cues over video results in temporally consistent segmentation with less flickering between classes in the video output. We observe an increased ability to learn thin structures and difficult classes compared to the baseline.

Figure 5.3 shows qualitative performance. The self-supervised learning produces smooth and robust estimates of geometry, and the model is noticeably more temporally consistent than the per-frame E-Net baseline. Figure 5.4 shows a single 10-frame sequence.

5.5 Conclusions

In this chapter, we investigated the problem of video scene understanding. We briefly summarise the main conclusions within the three main themes of this dissertation.

End-to-end deep learning. In this chapter, we have demonstrated an end-to-end deep learning model to learn semantic segmentation, optical flow and scene depth from video sequences in real-time. We show how to design recurrent units which can compensate for ego-motion and align features over time. Through a number of experiments, we present insights for training temporal models to form temporally consistent representations from long video sequences. We find that end-to-end learning over long video sequences outperforms models trained on individual frames.

Geometry. We show that explicitly modelling geometry, in the form of depth and motion, is critical to obtain accurate feature representations over video sequences. We demonstrate how to use geometry to align features over time, improving segmentation performance. This allows us to improve on the per-frame segmentation model by leveraging motion features. Jointly representing depth, in addition to semantics, further improves model performance.

We find that in order to train over long video sequences, it is important to provide a training signal for each frame. Labelling the semantics of each frame is too expensive. Therefore we demonstrate that learning motion and depth with unsupervised learning is an effective method to provide a training signal for each image in a long video sequence.

Uncertainty. While the probabilistic techniques from the previous chapters can be applied to this model, this chapter does not explicitly explore the application of uncertainty to the problem of video scene understanding. However, we utilise uncertainty for multi-task learning using the technique from Section 2.5.2 to simultaneously learn depth, motion and semantics.

Chapter 6

Conclusions

While this thesis explores a number of core computer vision problems — scene understanding, localisation, stereo and motion — we can observe some general conclusions.

Firstly, end-to-end deep learning has emerged as the prevailing paradigm for modelling computer vision problems. For all of the tasks in this thesis, end-to-end learning is able to outperform hand-engineered approaches. It reduces engineering effort and performs very well by optimising the model with respect to the end goal.

In general, we find using representations of geometry improves the representational power of vision models. It improves performance of models by simplifying the learning task and allows models to learn from unsupervised learning, greatly reducing the dependence on labelled training data.

Finally, probabilistic or Bayesian deep learning is a practical framework for quantifying model uncertainty for vision tasks. We showed that it is important to model uncertainty for safety-critical applications, to understand examples which are different from training data, and small datasets where the training data is sparse.

6.1 Findings and Limitations

We briefly summarise the findings and limitations of each chapter in this thesis.

Chapter 2 presented SegNet, a deep learning framework for dense pixel-wise output. We showed its efficacy for semantic segmentation, instance segmentation and depth regression. We compared different techniques for upsampling and show the advantage of learning filters and using information from the encoder. However, the model often fails to capture objects at different scales and reason within the context of the whole image. We derived a practical Bayesian deep learning formulation to capture aleatoric and epistemic uncertainty. We demonstrate how to use uncertainty to weight multiple tasks to jointly learn semantics and geometry.

Chapter 3 introduces PoseNet, a convolutional neural network for camera pose regression. We show that it is faster and more scalable than traditional structure from motion methods for localisation. While it is more robust to challenging appearance changes, it cannot produce the same level of fine-grained accuracy as classical geometry. However, explicitly using geometric and reprojection error loss functions can improve results. Modelling uncertainty is useful to determine metric relocalisation error and estimate loop closure. Although, this method is only capable of relocalisation, and does not address the problem of online-learning or localisation over video sequences.

Chapter 4 shows an end-to-end deep learning architecture for stereo disparity regression. We show it can outperform prior hand-engineered models while being significantly more accurate. However, the model forms the full cost volume and is not real-time. Further work is needed, perhaps with hierarchical models, for efficient inference. Although, we find we can use geometry for unsupervised learning by learning from photometric reprojection error. Finally, we show uncertainty is useful to attenuate noisy training labels and to combine semi-supervised and unsupervised learning. However, our probabilistic model assumes uni-modal Gaussian distributions which is not always accurate.

Chapter 5 introduces deep learning models which can learn video scene understanding over long video sequences. We show the benefits of explicitly modelling depth and motion. In order to train models over video we find there are a number of important factors to consider. We introduce temporal loss functions and show how to jointly learn semantic segmentation with supervised learning, and depth and optical flow with unsupervised learning.

6.2 Applications

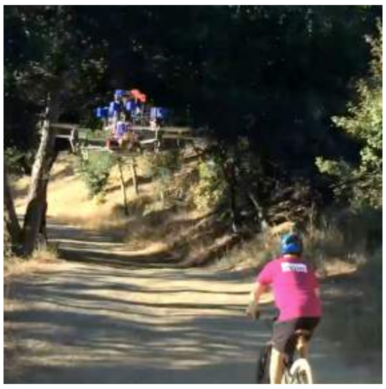
This work has an array of practical applications. Figure 6.1 shows technology which directly use the ideas and software developed in this thesis, from flying drones to self-driving cars. We briefly describe four general application areas of this work.

6.2.1 Robotics and Autonomous Vehicles

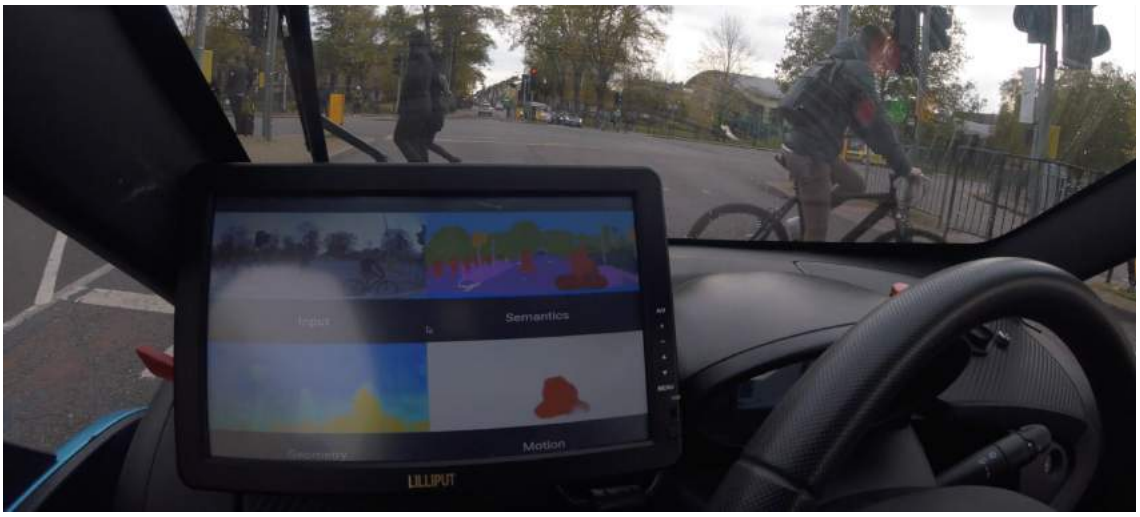
In order for autonomous robots to make their own intelligent decisions, which are trustworthy and useful, we need to give them the ability to understand their environment. This thesis shows that scene understanding with computer vision can provide this knowledge.

(a) Autonomous vehicles.

(b) Smart cities and intelligent surveillance.



(c) Autonomous unmanned aerial vehicles.



(d) Demonstration of the algorithms from Chapter 5 on a self-driving car in Cambridge, UK.

Fig. 6.1 **Applications of the technology developed in this thesis.** Algorithms and software which was written as part of this thesis are inside many of today's products, including these four examples. (a) shows a prototype autonomous vehicle running SegNet for real-time scene understanding. (b) shows a smart-city sensor by Vivacity (Vivacity, 2017) running software from this thesis. (c) shows a prototype drone from Skydio (Skydio, 2017) using algorithms and software from Chapter 2 to understand scene semantics and geometry. Finally, (d) demonstrates the algorithms from Chapter 5 operating in real-time on a self-driving car in Cambridge, UK.

Understanding the scene's geometry and semantics is a fundamental task for autonomous robots. This provides necessary information to navigate, avoid obstacles and to interact intelligently with the world. The ideas from Chapter 2 and Chapter 5 allow us to build such a system with end-to-end deep learning.

Cameras are attractive sensors because they are cheap, they are passive and require low-power. The algorithms presented in this thesis can operate in real-time on robotic platforms with relatively cheap cameras and computation. For example, on autonomous vehicles, self-driving cars, drones and domestic robots (Kendall et al., 2018, 2014; McAllister et al., 2017).

In Figure 6.1 (d), we demonstrate the algorithms from Chapter 5 operating in real-time on a prototype self-driving car on public UK roads in Cambridge. The algorithm is computed at 25Hz on a NVIDIA Drive PX2 computer using a 256×512 pixel image. We observe stable predictions of semantics, motion and geometry across a wide variety of road scenes in varying weather conditions.

The challenges for this application are developing a scene understanding representation which is robust, and generalises world-wide. For self-driving cars, it is paramount that the computer vision system is reliable in all conditions and across all edge-cases. The ideas about using geometry will be beneficial here, to leverage self-supervision to learn more effective representations from larger amounts of data. It is also critical these systems understand their uncertainty to make safe policy decisions based on well-founded information.

6.2.2 Smart-Cities and Surveillance

In addition to robotics, computer vision algorithms are useful to understand scenes from fixed cameras. One can imagine smart city infrastructure, or internet-of-things (IOT) devices which would benefit from visual scene understanding. Semantic and instance video segmentation are important technologies. These could have applications for security monitoring, collecting behaviour statistics and providing analytics of the world in real-time.

6.2.3 Medical Imaging

Computer vision is proving successful in advanced diagnosis of medical images (Ronneberger et al., 2015). However, obtaining training data is difficult and it is often biased against rare conditions and diseases. Therefore, it is important to account for uncertainty when making a diagnosis, with many of the ideas in this thesis very useful for this task. Medical imaging also often involves 3-D data with images obtained in voxels. Geometry may assist deep neural networks to learn more efficiently and be more effective in this domain.

6.2.4 Augmented Reality

Augmented reality provides an improved interface to data by overlaying it on the world we see. For example, wearable technology like glasses can overlay information such as advertising on shop façades, directions on the street or Pokémon to play with.

Augmented reality systems require accurate depth and semantic scene understanding. Global camera pose estimation is also of interest to overlay virtual worlds. Ideas from Chapter 3 for localisation and other ideas in this thesis about learning geometry can be used. Applications include entertainment, education or improving the accessibility of information.

6.3 Future Work

To conclude this thesis, there are many aspects of important future work which we would like to highlight. As is typical with any research, this thesis raises more questions than answers. Improvements to the individual algorithms have been discussed within the body of this work. However, at a high level we would like to highlight the following themes for future research which are particularly exciting.

6.3.1 Learning Geometry with Memory

A great challenge in artificial intelligence research is learning representations with memory (Graves et al., 2014). Today, this is typically approached in language and sequence modelling fields (Weston et al., 2014). However, we believe geometry offers an excellent setting to learn memory, because we can learn geometry with unsupervised learning. An example, is the problem of simultaneous localisation and mapping (Durrant-Whyte and Bailey, 2006), where a model must learn to build a map while experiencing a world. This is a task the human brain performs very well (Moser et al., 2008; O'Keefe and Nadel, 1978). Learning this with a computer vision system remains a great challenge for the best geometry and memory models.

6.3.2 More Flexible Uncertainty Distributions

This thesis has argued that understanding uncertainty is essential for safe computer vision systems. However, Bayesian deep learning often underestimates uncertainty (Chapter 2). It is important we improve the calibration of this uncertainty. In this thesis, one assumption all probabilistic models made was that the probability distribution is Gaussian. In reality, this may not be a valid claim. One important area of future research is learning more flexible distributions and multi-modal distributions which can represent the real-world more effectively. For example, multi-modal predictions are particularly important for predicting the future, which we describe next.

6.3.3 Predicting the Future

This thesis has addressed algorithms for understanding semantics and geometry from both individual images and video sequences. However, another large problem which has not been mentioned so far is, *what is going to happen next?* Predicting scene dynamics is an important problem to solve in computer vision, especially for applications in robotics. While there is some initial work in this area (Luc et al., 2017), this problem would benefit from the three themes of this thesis; end-to-end deep learning, geometry and uncertainty.

6.3.4 Learning to Act

Finally, the purpose of a brain is for movement (Wolpert and Ghahramani, 2000). Therefore, an interesting question is what is the point of learning these representations of semantics, motion and geometry? Ultimately, these algorithms are useful to learn representations for visuomotor control (systems that reason from sensory input to control output). Today, most visuomotor algorithms are naive and do not consider geometry and uncertainty (Kendall et al., 2018; Levine et al., 2016). Benefits could be observed from leveraging the ideas in this thesis about geometry and uncertainty, while learning end-to-end from perception to action.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA*.

Achtelik, M., Bachrach, A., He, R., Prentice, S., and Roy, N. (2009). Stereo vision and laser odometry for autonomous helicopters in gps-denied indoor environments. In *SPIE Defense, security, and sensing*, pages 733219–733219. International Society for Optics and Photonics.

Agrawal, P., Carreira, J., and Malik, J. (2015). Learning to see by moving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 37–45.

Altmann, S. L. (2005). *Rotations, quaternions, and double groups*. Courier Corporation.

Anderson, C. H., Van Essen, D. C., and Olshausen, B. A. (2005). Directed visual attention and the dynamic control of information flow.

Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). Optics: ordering points to identify the clustering structure. In *ACM Sigmod record*, volume 28, pages 49–60. ACM.

Babenko, A. and Lempitsky, V. (2015). Aggregating deep convolutional features for image retrieval. In *International Conference on Computer Vision (ICCV)*.

Babenko, A., Slesarev, A., Chigorin, A., and Lempitsky, V. (2014). Neural codes for image retrieval. In *European Conference on Computer Vision*.

Badrinarayanan, V., Handa, A., and Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *CoRR*, abs/1505.07293.

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for scene segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Bai, M. and Urtasun, R. (2017). Deep watershed transform for instance segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2858–2866. IEEE.

Barlow, H. B. (1989). Unsupervised learning. *Neural computation*, 1(3):295–311.

Barnard, S. T. and Fischler, M. A. (1982). Computational stereo. *ACM Computing Surveys*, 14(4):553–572.

Baxter, J. et al. (2000). A model of inductive bias learning. *J. Artif. Intell. Res.(JAIR)*, 12(149-198):3.

Belagiannis, V., Rupprecht, C., Carneiro, G., and Navab, N. (2015). Robust optimization for deep regression. In *International Conference on Computer Vision (ICCV)*, pages 2830–2838. IEEE.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Blake, A., Curwen, R., and Zisserman, A. (1993). A framework for spatiotemporal control in the tracking of visual contours. *International Journal of Computer Vision*, 11(2):127–145.

Blei, D. M. and Jordan, M. I. (2006). Variational inference for dirichlet process mixtures. *Bayesian analysis*, 1(1):121–143.

Bleyer, M., Rhemann, C., and Rother, C. (2011). PatchMatch Stereo-Stereo Matching with Slanted Support Windows. *Bmvc*, i(1):14.1–14.11.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural network. In *ICML*.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Brostow, G. J., Fauqueur, J., and Cipolla, R. (2009). Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97.

Brostow, G. J., Shotton, J., Fauqueur, J., and Cipolla, R. (2008). Segmentation and recognition using structure from motion point clouds. In *European conference on computer vision*, pages 44–57. Springer.

Budvytis, I., Badrinarayanan, V., and Cipolla, R. (2010). Label propagation in complex video sequences using semi-supervised learning. In *BMVC*, volume 2257, pages 2258–2259.

Bulo, Rota, S., and Kontschieder, P. (2014). Neural decision forests for semantic image labelling. In *CVPR*.

Caelles, S., Maninis, K.-K., Pont-Tuset, J., Leal-Taixé, L., Cremers, D., and Van Gool, L. (2017). One-shot video object segmentation. In *CVPR 2017*. IEEE.

Calonder, M., Lepetit, V., and Strecha, C. (2010). BRIEF : Binary Robust Independent Elementary Features. In *European Conference on Computer Vision (ECCV)*.

Caruana, R. (1998). Multitask learning. In *Learning to learn*, pages 95–133. Springer.

Chen, A. Y. and Corso, J. J. (2011). Temporally consistent multi-class video-object segmentation with the video graph-shifts algorithm. In *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, pages 614–621. IEEE.

Chen, D. M., Baatz, G., Köser, K., Tsai, S. S., Vedantham, R., Pylvänäinen, T., Roimela, K., Chen, X., Bach, J., Pollefeys, M., et al. (2011). City-scale landmark identification on mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 737–744. IEEE.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2015a). Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2016). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*.

Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*.

Chen, Z., Sun, X., Wang, L., Yu, Y., and Huang, C. (2015b). A deep visual correspondence embedding model for stereo matching costs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 972–980.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Choudhary, S. and Narayanan, P. J. (2012). Visibility probability structure from sfm datasets and applications. In *European Conference on Computer Vision*.

Ciodaro, T., Deva, D., De Seixas, J., and Damazio, D. (2012). Online particle detection with neural networks based on topological calorimetry information. In *Journal of physics: conference series*, volume 368, page 012030. IOP Publishing.

Cipolla, R., Battiato, S., and Farinella, G. M. (2010). *Computer Vision: Detection, recognition and reconstruction*, volume 285. Springer.

Clark, R., Wang, S., Markham, A., Trigoni, N., and Wen, H. (2017). Vidloc: 6-dof video-clip relocalization. *arXiv preprint arXiv:1702.06521*.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cummins, M. and Newman, P. (2008). FAB-MAP: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665.

Dai, J., He, K., and Sun, J. (2016). Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158.

De Brabandere, B., Neven, D., and Van Gool, L. (2017). Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*.

de Nijs, R., Ramos, S., Roig, G., Boix, X., Van Gool, L., and Kühnlenz, K. (2012). On-line semantic perception using uncertainty. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4185–4191. IEEE.

Delhumeau, J., Gosselin, P.-H., Jégou, H., and Pérez, P. (2013). Revisiting the VLAD image representation. In *ACM Multimedia*, Barcelona, Spain.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE.

Denker, J. and LeCun, Y. (1991). Transforming neural-net output levels to probability distributions. In *Advances in Neural Information Processing Systems 3*. Citeseer.

Der Kiureghian, A. and Ditlevsen, O. (2009). Aleatory or epistemic? does it matter? *Structural Safety*, 31(2):105–112.

Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634.

Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., and Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766.

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110.

Egnal, G., Mintz, M., and Wildes, R. P. (2004). A stereo confidence metric using single view imagery with comparison to five alternative approaches. *Image and vision computing*, 22(12):943–957.

Eigen, D. and Fergus, R. (2015). Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658.

Eigen, D., Puhrsch, C., and Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374.

Einecke, N. and Eggert, J. (2015). A multi-block-matching approach for stereo. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 585–592. IEEE.

Engel, J., Schöps, T., and Cremers, D. (2014). LSD-SLAM: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer.

Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111:98–136.

Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2012). Scene parsing with multiscale feature learning, purity trees, and optimal covers. In *ICML*.

Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2013). Learning hierarchical features for scene labeling. *IEEE PAMI*, 35(8):1915–1929.

Faugeras, O. (1993). *Three-dimensional computer vision: a geometric viewpoint*. MIT press.

Flynn, J., Neulander, I., Philbin, J., and Snavely, N. (2016). DeepStereo: Learning to Predict New Views from the World's Imagery. *CVPR*.

Fukushima, K. (1979). Neural network model for a mechanism of pattern recognition unaffected by shift in position - Neocognitron. *Trans. IECE*, J62-A(10):658–665.

Gadde, R., Jampani, V., and Gehler, P. V. (2017). Semantic video cnns through representation warping. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE.

Gal, Y. (2016). *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge.

Gal, Y. and Ghahramani, Z. (2015). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *arXiv:1506.02142*.

Gal, Y. and Ghahramani, Z. (2016). Bayesian convolutional neural networks with Bernoulli approximate variational inference. *ICLR workshop track*.

Gal, Y., Hron, J., and Kendall, A. (2017). Concrete dropout. In *Advances in Neural Information Processing Systems (NIPS)*.

Garg, R., Kumar, B. V., Carneiro, G., and Reid, I. (2016). Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer.

Gatta, C., Romero, A., and van de Weijer, J. (2014). Unrolling loopy top-down semantic feedback in convolutional deep networks. In *CVPR Workshop on Deep Vision*.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Geiger, A., Roser, M., and Urtasun, R. (2010). Efficient large-scale stereo matching. In *Asian conference on computer vision*, pages 25–38. Springer.

Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459.

Gibson, E. J. and Walk, R. D. (1960). The "visual cliff". *Scientific American*, 202(4):64–71.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Godard, C., Mac Aodha, O., and Brostow, G. J. (2017). Unsupervised monocular depth estimation with left-right consistency. In *CVPR*.

Gong, Y., Wang, L., Guo, R., and Lazebnik, S. (2014). Multi-scale orderless pooling of deep convolutional activation features. In *European Conference on Computer Vision*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Gould, S., Fulton, R., and Koller, D. (2009). Decomposing a scene into geometric and semantically consistent regions. In *ICCV*, pages 1–8. IEEE.

Grangier, D., Bottou, L., and Collobert, R. (2009). Deep convolutional networks for scene parsing. In *ICML Workshop on Deep Learning*.

Graves, A. (2011). Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Guney, F. and Geiger, A. (2015). Displets: Resolving stereo ambiguities using object knowledge. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4165–4175.

Gupta, S., Arbelaez, P., and Malik, J. (2013). Perceptual organization and recognition of indoor scenes from rgb-d images. In *CVPR*, pages 564–571. IEEE.

Gupta, S., Girshick, R., Arbeláez, P., and Malik, J. (2014). Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer.

Guynn, J. (2015). Google photos labeled black people 'gorillas'. *USA Today*.

Haeusler, R., Nair, R., and Kondermann, D. (2013). Ensemble Learning for Confidence Measures in Stereo Vision. *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 305–312.

Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., and Malik, J. (2011). Semantic contours from inverse detectors. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 991–998. IEEE.

Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. (2015). Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, pages 447–456.

Hartley, R. and Zisserman, A. (2000). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, first edition.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

He, X., Zemel, R. S., and Carreira-Perpiñán, M. Á. (2004). Multiscale conditional random fields for image labeling. In *Computer vision and pattern recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE computer society conference on*, volume 2, pages II–II. IEEE.

Heise, P., Jensen, B., Klose, S., and Knoll, A. (2015). Fast Dense Stereo Correspondences by Binary Locality Sensitive Hashing. *ICRA*, pages 1–6.

Helmstaedter, M., Briggman, K. L., Turaga, S. C., Jain, V., Seung, H. S., and Denk, W. (2013). Connectomic reconstruction of the inner plexiform layer in the mouse retina. *Nature*, 500(7461):168.

Hermans, A., Floros, G., and Leibe, B. (2014). Dense 3D Semantic Mapping of Indoor Scenes from RGB-D Images. In *ICRA*.

Hernández-Lobato, J. M., Li, Y., Hernández-Lobato, D., Bui, T., and Turner, R. E. (2016). Black-box alpha divergence minimization. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1511–1520.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Hirschmuller, H. (2005). Accurate and efficient stereo processing by semi-global matching and mutual information. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 807–814. IEEE.

Hirschmüller, H. (2008). Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341.

Hirschmüller, H. and Scharstein, D. (2007). Evaluation of Cost Functions for Stereo Matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*.

Hoaglin, D. C., Mosteller, F., and Tukey, J. W. (1983). *Understanding robust and exploratory data analysis*, volume 3. Wiley New York.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hong, S., Noh, H., and Han, B. (2015). Decoupled deep neural network for semi-supervised semantic segmentation. *CoRR*, abs/1506.04924.

Horn, B. K. and Brooks, M. J. (1989). *Shape from shading*. MIT press.

Hu, X. and Mordohai, P. (2012). A quantitative evaluation of confidence measures for stereo vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2121–2133.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Huang, G., Liu, Z., Weinberger, K. Q., and van der Maaten, L. (2016). Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*.

Huang, J.-T., Li, J., Yu, D., Deng, L., and Gong, Y. (2013). Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7304–7308. IEEE.

Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154.

Huber, P. J. (2011). *Robust statistics*. Springer.

Hur, J. and Roth, S. (2016). Joint optical flow and temporally consistent semantic segmentation. In *European Conference on Computer Vision*, pages 163–177. Springer.

Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., and Brox, T. (2016). Flownet 2.0: Evolution of optical flow estimation with deep networks. *CoRR*, abs/1612.01925.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167.

Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025.

Jamaludin, A., Kadir, T., and Zisserman, A. (2016). Spinenet: automatically pinpointing classification evidence in spinal mris. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 166–175. Springer.

Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? In *ICCV*, pages 2146–2153.

Jason, J. Y., Harley, A. W., and Derpanis, K. G. (2016). Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *Computer Vision–ECCV 2016 Workshops*, pages 3–10. Springer.

Jégou, H., Douze, M., Schmid, C., and Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3304–3311.

Jegou, H., Perronnin, F., Douze, M., Sánchez, J., Perez, P., and Schmid, C. (2012). Aggregating local image descriptors into compact codes. *IEEE transactions on pattern analysis and machine intelligence*, 34(9):1704–1716.

Jégou, S., Drozdzal, M., Vazquez, D., Romero, A., and Bengio, Y. (2016). The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. *arXiv preprint arXiv:1611.09326*.

Ji, S., Xu, W., Yang, M., and Yu, K. (2013). 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732.

Karsch, K., Liu, C., and Kang, S. B. (2012). Depth extraction from video using non-parametric sampling. In *European Conference on Computer Vision*, pages 775–788. Springer.

Kellman, P. J. and Arterberry, M. E. (2006). Infant visual perception. *Handbook of child psychology*.

Kendall, A., Badrinarayanan, V., and Cipolla, R. (2017a). Bayesian SegNet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Kendall, A. and Cipolla, R. (2016). Modelling uncertainty in deep learning for camera relocalization. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

Kendall, A. and Cipolla, R. (2017). Geometric loss functions for camera pose regression with deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems (NIPS)*.

Kendall, A., Gal, Y., and Cipolla, R. (2017b). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *arXiv preprint arXiv:1705.07115*.

Kendall, A., Grimes, M., and Cipolla, R. (2015). Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

Kendall, A., Hawke, J., Janz, D., Mazur, P., Reda, D., Allen, J.-M., Lam, V.-D., Bewley, A., and Shah, A. (2018). Learning to drive in a day. *arXiv preprint arXiv:1807.00412*.

Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., Kennedy, R., Bachrach, A., and Bry, A. (2017c). End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

Kendall, A. G., Salvapantula, N. N., and Stol, K. A. (2014). On-board object tracking control of a quadcopter with monocular vision. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 404–411. IEEE.

Khoreva, A., Perazzi, F., Benenson, R., Schiele, B., and Sorkine-Hornung, A. (2016). Learning video object segmentation from static images. *arXiv preprint arXiv:1612.02646*.

Kiefer, J., Wolfowitz, J., et al. (1952). Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, page 201611835.

Klaus, A., Sormann, M., and Karner, K. (2006). Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. *Proceedings - International Conference on Pattern Recognition*, 3:15–18.

Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, IEEE and ACM International Symposium on*, pages 225–234. IEEE.

Koenderink, J. J. (1990). *Solid shape*. MIT Press.

Koenderink, J. J. and Van Doorn, A. J. (1991). Affine structure from motion. *JOSA A*, 8(2):377–385.

Kokkinos, I. (2016). Ubernet: Training auniversal'convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. *arXiv preprint arXiv:1609.02132*.

Kolmogorov, V. and Zabih, R. (2001). Computing visual correspondences with occlusions using graph cuts. In *International Conference on Computer Vision (ICCV)*.

Koltun, V. (2011). Efficient inference in fully connected crfs with gaussian edge potentials. In *In: NIPS (2011*.

Koltun, V. (2017). Learning to act with natural supervision.

Kontschieder, P., Bulo, S. R., Bischof, H., and Pelillo, M. (2011). Structured class-labels in random forests for semantic image labelling. In *ICCV*, pages 2190–2197. IEEE.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kundu, A., Li, Y., Dellaert, F., Li, F., and Rehg, J. M. (2014). Joint semantic segmentation and 3d reconstruction from monocular video. In *European Conference on Computer Vision*, pages 703–718. Springer.

Kundu, A., Vineet, V., and Koltun, V. (2016). Feature space optimization for semantic video segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3168–3175.

Ladicky, L., Shi, J., and Pollefeys, M. (2014). Pulling things out of perspective. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 89–96.

Ladickẙ, L., Sturgess, P., Alahari, K., Russell, C., and Torr, P. H. (2010). What, where and how many? combining object detectors and crfs. In *European conference on computer vision*, pages 424–437. Springer.

Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., and Navab, N. (2016). Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE.

Le, Q. V., Smola, A. J., and Canu, S. (2005). Heteroscedastic Gaussian process regression. In *Proceedings of the 22nd international conference on Machine learning*, pages 489–496. ACM.

LeCun, Y. (1988). A theoretical framework for back-propagation. In Touretzky, D., Hinton, G., and Sejnowski, T., editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 21–28, CMU, Pittsburgh, Pa. Morgan Kaufmann.

Leibe, B., Leonardis, A., and Schiele, B. (2008). Robust object detection with interleaved categorization and segmentation. *International journal of computer vision*, 77(1-3):259–289.

Leung, M. K., Xiong, H. Y., Lee, L. J., and Frey, B. J. (2014). Deep learning of the tissue-regulated splicing code. *Bioinformatics*, 30(12):i121–i129.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40.

Li, B., Shen, C., Dai, Y., van den Hengel, A., and He, M. (2015). Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1119–1127.

Li, R., Liu, Q., Gui, J., Gu, D., and Hu, H. (2017). Indoor relocalization in challenging environments with dual-stream convolutional neural networks. *IEEE Transactions on Automation Science and Engineering*.

Li, Y. and Huttenlocher, D. P. (2008). Learning for stereo vision using the structured support vector machine. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*.

Li, Y., Snavely, N., Huttenlocher, D., and Fua, P. (2012). Worldwide pose estimation using 3d point clouds. In *European Conference on Computer Vision*, pages 15–29. Springer.

Li, Y., Snavely, N., and Huttenlocher, D. P. (2010). Location recognition using prioritized feature matching. In *European Conference on Computer Vision*, pages 791–804. Springer.

Liang, X., Wei, Y., Shen, X., Yang, J., Lin, L., and Yan, S. (2015). Proposal-free network for instance-level object segmentation. *arXiv preprint arXiv:1509.02636*.

Liao, Y., Kodagoda, S., Wang, Y., Shi, L., and Liu, Y. (2016). Understand scene categories by objects: A semantic regularized scene classifier using convolutional neural networks. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 2318–2325. IEEE.

Lin, G., Shen, C., Reid, I., et al. (2015). Efficient piecewise training of deep structured models for semantic segmentation. *arXiv preprint arXiv:1504.01013*.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Liu, C., Yuen, J., Torralba, A., and Sivic, J. (2008). Sift flow: Dense correspondence across different scenes. In *ECCV, Marseille*.

Liu, F., Shen, C., and Lin, G. (2015a). Deep convolutional neural fields for depth estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5162–5170.

Liu, F., Shen, C., Lin, G., and Reid, I. (2015b). Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields. *Pattern Analysis and Machine Intelligence*, page 15.

Liu, M., Salzmann, M., and He, X. (2014). Discrete-continuous depth estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 716–723.

Liu, W., Rabinovich, A., and Berg, A. C. (2015c). Parsenet: Looking wider to see better. *CoRR*, abs/1506.04579.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Luc, P., Neverova, N., Couprie, C., Verbeek, J., and Lecun, Y. (2017). Predicting deeper into the future of semantic segmentation. In *ICCV 2017-International Conference on Computer Vision*, page 10.

Luo, W., Schwing, A. G., and Urtasun, R. (2016). Efficient deep learning for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5695–5703.

MacKay, D. J. (1992). A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Manduchi, R. and Tomasi, C. (1999). Distinctiveness maps for image matching. In *Image Analysis and Processing, 1999. Proceedings. International Conference on*, pages 26–31. IEEE.

Marr, D. (1982). Vision. 1982. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.*

Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., and Brox, T. (2016). A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048.

McAllister, R., Gal, Y., Kendall, A., van der Wilk, M., Shah, A., Cipolla, R., and Weller, A. (2017). Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning. In *International Joint Conference on Artificial Intelligence (IJCAI).*

Melekhov, I., Kannala, J., and Rahtu, E. (2017). Relative camera pose estimation using convolutional neural networks. *arXiv preprint arXiv:1702.01381.*

Menze, M. and Geiger, A. (2015). Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR).*

Miksik, O., Munoz, D., Bagnell, J. A., and Hebert, M. (2013). Efficient temporal consistency for streaming video scene analysis. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 133–139. IEEE.

Miksik, O., Perez-Rua, J.-M., Torr, P. H., and Perez, P. (2017). Roam: a rich object appearance model with application to rotoscoping. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR).*

Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. (2016). Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003.

Moser, E. I., Kropff, E., and Moser, M.-B. (2008). Place cells, grid cells, and the brain's spatial representation system. *Annu. Rev. Neurosci.*, 31:69–89.

Mostajabi, M., Yadollahpour, P., and Shakhnarovich, G. (2014). Feedforward semantic segmentation with zoom-out features. *arXiv preprint arXiv:1412.0774.*

Mottaghi, R., Chen, X., Liu, X., Cho, N.-G., Lee, S.-W., Fidler, S., Urtasun, R., et al. (2014). The role of context for object detection and semantic segmentation in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 891–898. IEEE.

Mur-Artal, R., Montiel, J., and Tardós, J. D. (2015). Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163.

Neal, R. M. (1995). *Bayesian learning for neural networks*. PhD thesis, University of Toronto.

Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. (2011). DTAM: Dense tracking and mapping in real-time. In *International Conference on Computer Vision (ICCV)*, pages 2320–2327. IEEE.

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., and Ng, A. Y. (2011). Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696.

NHTSA (2017). PE 16-007. Technical report, U.S. Department of Transportation, National Highway Traffic Safety Administration. Tesla Crash Preliminary Evaluation Report.

Nister, D. and Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Nix, D. A. and Weigend, A. S. (1994). Estimating the mean and variance of the target probability distribution. In *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference On*, volume 1, pages 55–60. IEEE.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, New York, 2nd edition.

Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. *CoRR*, abs/1505.04366.

O'Keefe, J. and Nadel, L. (1978). *The hippocampus as a cognitive map*, volume 3. Clarendon Press Oxford.

Oquab, M., Bottou, L., Laptev, I., and Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1717–1724. IEEE.

Papert, S. A. (1966). The summer vision project.

Park, M. G. and Yoon, K. J. (2015). Leveraging stereo matching with learning-based confidence measures. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:101–109.

Paszke, A., Chaurasia, A., Kim, S., and Culurciello, E. (2016). Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*.

Patraucean, V., Handa, A., and Cipolla, R. (2015). Spatio-temporal video autoencoder with differentiable memory. *arXiv preprint arXiv:1511.06309*.

Perazzi, F., Pont-Tuset, J., McWilliams, B., Van Gool, L., Gross, M., and Sorkine-Hornung, A. (2016). A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision and Pattern Recognition*.

Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2007). Object Retrieval with Large Vocabularies and Fast Spatial Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Pinheiro, P. and Collobert, R. (2014). Recurrent convolutional neural networks for scene labeling. In *ICML*, pages 82–90.

Pinheiro, P. O., Collobert, R., and Dollár, P. (2015). Learning to segment object candidates. In *Advances in Neural Information Processing Systems*, pages 1990–1998.

Ranzato, M., Huang, F. J., Boureau, Y., and LeCun, Y. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR*.

Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian processes for machine learning*, volume 1. MIT press Cambridge.

Razavian, A. S., Azizpour, H., Sullivan, J., and Carlsson, S. (2014a). Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE Conference on*, pages 512–519. IEEE.

Razavian, A. S., Sullivan, J., Maki, A., and Carlsson, S. (2014b). A baseline for visual instance retrieval with deep convolutional networks. In *arXiv:1412.6574*.

Ren, X., Bo, L., and Fox, D. (2012). Rgb-(d) scene labeling: Features and algorithms. In *CVPR*, pages 2759–2766. IEEE.

Ren, Z., Yan, J., Ni, B., Liu, B., Yang, X., and Zha, H. (2017). Unsupervised deep learning for optical flow estimation. In *AAAI*, pages 1495–1501.

Roberts, L. G. (1963). *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology.

Robertson, D. P. and Cipolla, R. (2004). An image-based system for urban navigation. In *BMVC*, volume 19, page 165.

Roddick, T., Kendall, A., and Cipolla, R. (2018). Orthographic feature transform for monocular 3d object detection. *arXiv preprint arXiv:1811.08188*.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer.

Ros, G., Ramos, S., Granados, M., Bakhtiary, A., Vazquez, D., and Lopez, A. (2015). Vision-based offline-online perception paradigm for autonomous driving. In *WACV*.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *International conference on computer vision (ICCV)*, pages 2564–2571. IEEE.

Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533.

Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. (2008). Labelme: a database and web-based tool for image annotation. *IJCV*, 77(1-3):157–173.

Sabater, N., Almansa, A., and Morel, J.-M. (2012). Meaningful matches in stereovision. *IEEE transactions on pattern analysis and machine intelligence*, 34(5):930–942.

Sattler, T., Leibe, B., and Kobbelt, L. (2011). Fast image-based localization using direct 2d-to-3d matching. In *International Conference on Computer Vision*, pages 667–674. IEEE.

Sattler, T., Leibe, B., and Kobbelt, L. (2012). Improving Image-Based Localization by Active Correspondence Search.

Sattler, T., Leibe, B., and Kobbelt, L. (2016). Efficient & effective prioritized matching for large-scale image-based localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Sattler, T., Sweeney, C., and Pollefeys, M. (2014). On sampling focal length values to solve the absolute pose problem. In *European Conference on Computer Vision*, pages 828–843. Springer.

Saxena, A., Sun, M., and Ng, A. Y. (2009). Make3d: Learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):824–840.

Scharstein, D., Hirschmüller, H., Kitajima, Y., Krathwohl, G., Nešić, N., Wang, X., and Westling, P. (2014). High-resolution stereo datasets with subpixel-accurate ground truth. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8753(1):31–42.

Scharstein, D. and Pal, C. (2007). Learning conditional random fields for stereo. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Scharstein, D. and Szeliski, R. (1998). Stereo matching with nonlinear diffusion. *International journal of computer vision*, 28(2):155–174.

Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42.

Schindler, G., Brown, M., and Szeliski, R. (2007). City-scale location recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–7. IEEE.

Schwing, A. G. and Urtasun, R. (2015). Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*.

Seki, A. and Pollefeys, M. (2016). Patch based confidence prediction for dense disparity map. In *British Machine Vision Conference (BMVC)*.

Sengupta, S., Greveson, E., Shahrokni, A., and Torr, P. H. (2013). Urban 3d semantic modelling using stereo vision. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 580–585. IEEE.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*.

Shelhamer, E., Rakelly, K., Hoffman, J., and Darrell, T. (2016). Clockwork convnets for video semantic segmentation. In *Computer Vision–ECCV 2016 Workshops*, pages 852–868. Springer.

Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., and Fitzgibbon, A. (2013). Scene coordinate regression forests for camera relocalization in RGB-D images. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 2930–2937. IEEE.

Shotton, J., Johnson, M., and Cipolla, R. (2008). Semantic texton forests for image categorization and segmentation. In *CVPR*.

Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sivic, J., Zisserman, A., et al. (2003). Video google: A text retrieval approach to object matching in videos. In *International Conference on Computer Vision (ICCV)*, volume 2, pages 1470–1477.

Skydio (2017). https://www.skydio.com/.

Snavely, N., Seitz, S. M., and Szeliski, R. (2008). Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210.

Song, S., Lichtenberg, S. P., and Xiao, J. (2015). Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 567–576.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Sterling, P. and Laughlin, S. (2015). *Principles of neural design.* MIT Press.

Sturgess, P., Alahari, K., Ladicky, L., and H.S.Torr, P. (2009). Combining appearance and structure from motion features for road scene understanding. In *BMVC*.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Svarm, L., Enqvist, O., Oskarsson, M., and Kahl, F. (2014). Accurate localization and pose estimation for large 3d models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 532–539.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*.

Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.

Teichmann, M., Weber, M., Zoellner, M., Cipolla, R., and Urtasun, R. (2016). Multinet: Real-time joint semantic reasoning for autonomous driving. *arXiv preprint arXiv:1612.07695*.

Thrun, S. and Bücken, A. (1996). Learning maps for indoor mobile robot navigation. Technical report, Computer Science Department, Carnegie Mellon University.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. MIT press.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2).

Tighe, J. and Lazebnik, S. (2013). Superparsing. *International Journal of Computer Vision*, 101(2):329–349.

Tokmakov, P., Alahari, K., and Schmid, C. (2017). Learning video object segmentation with visual memory. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

Tolias, G., Sicre, R., and JÃ©gou, H. (2016). Particular object retrieval with integral max-pooling of cnn activations. In *ICLR*.

Torii, A., Sivic, J., Pajdla, T., and Okutomi, M. (2013). Visual place recognition with repetitive structures. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 883–890.

Torralba, A., Russell, B. C., and Yuen, J. (2009). Labelme: online image annotation and applications. Technical report, MIT CSAIL Technical Report.

Tripathi, S., Belongie, S., Hwang, Y., and Nguyen, T. (2015). Semantic video segmentation: Exploring inference efficiency. In *SoC Design Conference (ISOCC), 2015 International*, pages 157–158. IEEE.

Tsai, Y.-H., Yang, M.-H., and Black, M. J. (2016). Video segmentation via object flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3899–3908.

Uhrig, J., Cordts, M., Franke, U., and Brox, T. (2016). Pixel-level encoding and depth layering for instance-level semantic labeling. In *German Conference on Pattern Recognition*, pages 14–25. Springer.

Ummenhofer, B., Zhou, H., Uhrig, J., Mayer, N., Ilg, E., Dosovitskiy, A., and Brox, T. (2017). Demon: Depth and motion network for learning monocular stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Valipour, S., Siam, M., Jagersand, M., and Ray, N. (2017). Recurrent fully convolutional networks for video segmentation. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pages 29–36. IEEE.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605):85.

Vertens, J., Valada, A., and Burgard, W. (2017). Smsnet: Semantic motion segmentation using deep convolutional neural networks. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS).*

Vivacity (2017). http://www.vivacitylabs.com/.

Voigtlaender, P. and Leibe, B. (2017). Online adaptation of convolutional neural networks for video object segmentation. In *BMVC*.

Walch, F., Hazirbas, C., Leal-Taixé, L., Sattler, T., Hilsenbeck, S., and Cremers, D. (2016). Image-based localization with spatial lstms. *arXiv preprint arXiv:1611.07890.*

Weston, J., Chopra, S., and Bordes, A. (2014). Memory networks. *arXiv preprint arXiv:1410.3916.*

Weyand, T., Kostrikov, I., and Philbin, J. (2016). Planet-photo geolocation with convolutional neural networks. In *European Conference on Computer Vision*, pages 37–55. Springer.

Wolpert, D. M. and Ghahramani, Z. (2000). Computational principles of movement neuroscience. *Nature neuroscience*, 3:1212–1217.

Wu, C. (2013). Towards linear-time incremental structure from motion. In *3D Vision-3DV 2013, 2013 International Conference on*, pages 127–134. IEEE.

Yamaguchi, K., McAllester, D., and Urtasun, R. (2014). Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *European Conference on Computer Vision*, pages 756–771. Springer.

Yang, Y., Li, Z., Zhang, L., Murphy, C., Ver Hoeve, J., and Jiang, H. (2012). Local label descriptor for example based semantic image labeling. In *European Conference on Computer Vision*, pages 361–375. Springer.

Yoon, K.-J. and Kweon, I. S. (2007). Stereo matching with the distinctive similarity measure. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–7. IEEE.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

Yu, F. and Koltun, V. (2016). Multi-scale context aggregation by dilated convolutions. In *ICLR*.

Zabih, R. and Woodfill, J. (1994). Non-parametric Local Transforms for Computing Visual Correspondence. *In Proceedings of European Conference on Computer Vision*, pages 151–158.

Zagoruyko, S. and Komodakis, N. (2015). Learning to compare image patches via convolutional neural networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June(i):4353–4361.

Zamir, A. R., Sax, A., Shen, W., Guibas, L., Malik, J., and Savarese, S. (2018). Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3712–3722.

Zbontar, J. and LeCun, Y. (2015). Computing the stereo matching cost with a convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1592–1599.

Zbontar, J. and LeCun, Y. (2016). Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17:1–32.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer.

Zeisl, B., Sattler, T., and Pollefeys, M. (2015). Camera pose voting for large-scale image-based localization. In *International Conference on Computer Vision (ICCV)*.

Zhang, C., Wang, L., and Yang, R. (2010). Semantic segmentation of urban scenes using dense depth maps. In *European Conference on Computer Vision*, pages 708–721. Springer.

Zhang, L. and Seitz, S. M. (2007). Estimating optimal parameters for MRF stereo from a single image pair. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):331–342.

Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. (2017). Pyramid scene parsing network. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. (2015). Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2014a). Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*.

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. (2014b). Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems*, pages 487–495.

Zhou, T., Brown, M., Snavely, N., and Lowe, D. G. (2017). Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Zhu, X., Xiong, Y., Dai, J., Yuan, L., and Wei, Y. (2017). Deep feature flow for video recognition. In *CVPR*, volume 1, page 3.

Zitnick, C. L. and Dollár, P. (2014). Edge boxes: Locating object proposals from edges. In *Computer Vision–ECCV 2014*, pages 391–405. Springer.