Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2018

Active Learning for Road Segmentation using Convolutional Neural Networks

Michael Sörsäter

LINKÖPING UNIVERSITY

Master of Science Thesis in Electrical Engineering

Active Learning for Road Segmentation using Convolutional Neural Networks

Michael Sörsäter

LiTH-ISY-EX–18/5176–SE

Supervisors: **Erik Wernholt**
Veoneer, Linköping
Dennis Lundström
Veoneer, Linköping
Mikael Persson
ISY, Linköping University

Examiner: **Fahad Khan**
ISY, Linköping University

Division of Computer Vision
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Abstract

In recent years, development of Convolutional Neural Networks has enabled high performing semantic segmentation models. Generally, these deep learning based segmentation methods require a large amount of annotated data. Acquiring such annotated data for semantic segmentation is a tedious and expensive task.

Within machine learning, active learning involves in the selection of new data in order to limit the usage of annotated data. In active learning, the model is trained for several iterations and additional samples are selected that the model is *uncertain* of. The model is then retrained on additional samples and the process is repeated again. In this thesis, an active learning framework has been applied to road segmentation which is semantic segmentation of objects related to road scenes.

The uncertainty in the samples is estimated with Monte Carlo dropout. In Monte Carlo dropout, several dropout masks are applied to the model and the variance is captured, working as an estimate of the model's uncertainty. Other metrics to rank the uncertainty evaluated in this work are: a baseline method that selects samples randomly, the entropy in the default predictions and three additional variations/extensions of Monte Carlo dropout.

Both the active learning framework and uncertainty estimation are implemented in the thesis. Monte Carlo dropout performs slightly better than the baseline in 3 out of 4 metrics. Entropy outperforms all other implemented methods in all metrics. The three additional methods do not perform better than Monte Carlo dropout.

An analysis of what kind of uncertainty Monte Carlo dropout capture is performed together with a comparison of the samples selected by baseline and Monte Carlo dropout. Future development and possible improvements are also discussed.

Acknowledgments

First off I would like to thank my supervisors at Veoneer, Erik Wernholt and Dennis Lundström who have been a great resource to me. Thank you for your help and guidance, both with direction and focus in the thesis work as well as helping out with proofreading and feedback of the thesis.

I would like to thank my supervisor Mikael Persson and examiner Fahad Khan at ISY, Linköping University, for your time and help. I would also like to thank my opponent Victor Tranell for your comments and feedback on the thesis.

I direct my thanks to the classification team at Veoneer who have been both friendly and helpful. Finally, I would like to thank Felicia André for your love and support.

Linköping, October 2018
Michael Sörsäter

Contents

1 Introduction **1**
- 1.1 Aim . . . 2
- 1.2 Research Questions . . . 2
- 1.3 Delimitations . . . 3
- 1.4 Background . . . 3
- 1.5 Outline . . . 3

2 Related Work **5**
- 2.1 Convolutional Neural Networks . . . 5
- 2.2 Semantic Segmentation . . . 8
- 2.3 Uncertainty in Neural Networks . . . 11
- 2.4 Active Learning . . . 13

3 Method **17**
- 3.1 Problem Description . . . 18
- 3.2 Dataset . . . 18
- 3.3 Performance Metrics . . . 19
- 3.4 Dataset Size . . . 21
- 3.5 Active Learning Framework . . . 23
- 3.6 Model . . . 25
- 3.7 Uncertainty . . . 28
- 3.8 Selection Criteria . . . 33

4 Experiments and Results **35**
- 4.1 Experiments . . . 36
- 4.2 Results Description . . . 37
- 4.3 Baseline Results . . . 38
- 4.4 Monte Carlo Dropout Results . . . 39
- 4.5 Entropy Results . . . 40
- 4.6 Dropout 50/50 Results . . . 41
- 4.7 Dropout Distribution Results . . . 42
- 4.8 Dropout 50/50 + Distribution Results . . . 43
- 4.9 All Results . . . 44

4.10 Subset Results 46
4.11 Network Improvements 47

5 Analysis **51**
5.1 Network Performances 51
5.2 Dropout Subset Distribution 52
5.3 t-SNE Analysis 55

6 Discussion **57**
6.1 Results Discussion 58
6.2 Informative Samples 58
6.3 Uncertainty Aggregation 60
6.4 Is Active Learning Worth it? 60
6.5 Uncertainty Applications 61
6.6 Thesis Reflection 61

7 Conclusion **63**
7.1 Research Questions 63
7.2 Future Work 64

A ENet Model **67**

Bibliography **69**

Acronyms

CNN Convolutional Neural Network. 1–3, 5–9, 11, 15, 16, 18, 24–26, 36, 60

FCN Fully Convolutional Network. 9, 10

ILSVRC ImageNet Large Scale Visual Recognition Challenge. 1, 5, 9

SVM Support Vector Machine. 14

1

Introduction

A Convolutional Neural Network (CNN) is a neural network developed for analyzing and understanding visual data. The interest, popularity, and applications of CNNs have grown exponentially in recent years. In 2015, a CNN outperformed humans [21] in the visual recognition competition named ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [39].

Semantic segmentation, the task of assigning each pixel in an image to a class, has achieved state-of-the-art results with CNNs. Semantic segmentation is important for *scene understanding*, that is, interpreting the content and meaning of images. For autonomous vehicles, it is crucial to have knowledge of the vehicle's surroundings. This can be achieved with CNNs trained for semantic segmentation. The market for autonomous vehicles is exploding and expected to be worth $7 trillion by 2050 [31].

The availability of raw input data is often abundant in visual recognition problems. In supervised machine learning, the model is trained on correctly annotated samples called *ground truth*. Annotating samples is often done manually which is an expensive task. It takes a long time and requires trained annotators. Difficulties lie in acquiring annotated training data that are diverse and represent the full spectra of potential situations. In regular training, this problem is usually undertaken by using a lot of training data. Leading to increased expenses for image annotation.

Active learning is a technique that aims at limiting the amount of annotated data by involving the learning algorithm in the selection of new training samples. Active learning works by training in iterations. After each active learning iteration is the most *informative* samples selected, annotated and used to retrain the network. This process is repeated until either all samples are used or a satisfactory

performance of the network is obtained. By reducing the amount of annotated data samples required, the annotation cost will decrease as well. Compared to regular training, active learning trains for a longer time as the training is done in multiple iterations. A relatively low performance of active learning is needed for it to be worth it, the annotation cost is high compared to prolonged GPU time.

When working with a CNN, the network is trained on a training set. The performance of the network can be verified by predicting unseen images from a test set and comparing them to the ground truth. However, the *uncertainty* in a network is usually unknown and can be hard to assess. If a model has a way to select uncertain (informative) samples for its training data, the model is expected to reach the same performance as regular training but with less amount of annotated data.

1.1 Aim

This thesis aims at investigating and evaluating the technique active learning on road segmentation using CNNs. Several metrics to rank the uncertainty in samples will be implemented. The network will train for a number of active learning iterations. After each active learning iteration, the most informative samples of the unannotated training and validation sets are selected. These samples are then annotated and added to the training and validation sets.

To evaluate the capabilities and potential improvement of active learning, a method which selects new samples randomly will be used as a baseline. The performance of the network will be measured on a separate test set after each active learning iteration.

1.2 Research Questions

The thesis will be based on the following research questions:

1. Can model uncertainty, estimated by dropout and Monte Carlo methods, be used for informative sample selection in Active Learning?
   (a) What is a good way to aggregate pixel uncertainty to sample uncertainty?
2. Is Active Learning applicable for Semantic Segmentation?
3. Can Active Learning be used to limit the amount of annotated data in Semantic Segmentation?

1.3 Delimitations

A CNN is developed for the problem *road segmentation*, that is, semantic segmentation in a road environment with classes related to road objects. An implementation of the deep neural network Efficient Neural Network (ENet) [38] is available at Veoneer and is the architecture used in the thesis. An internal dataset at Veoneer will be used in the thesis which contains around 54 000 images divided into training, validation, and test sets.

1.4 Background

This thesis is requested by Veoneer in Linköping, Sweden. Veoneer is a company working with automotive safety where active safety is the main focus at the site in Linköping. Veoneer has worked with machine learning for more than a decade. Automotive vision systems, radar systems, and LiDAR systems are a few of the products developed by Veoneer.

1.5 Outline

Structure of the thesis:

- Chapter 2 presents related work in the field.
- Chapter 3 explains how the thesis was carried out.
- Chapter 4 defines the networks to be trained and presents the results obtained from them.
- Chapter 5 analyses the results.
- Chapter 6 discusses the findings, their relevance, and importance.
- Chapter 7 concludes the thesis.

2

Related Work

This chapter presents related work for the thesis. The chapter is divided into four separate parts:

1. **Convolutional Neural Networks** - Explanation of CNNs, dropout, and Monte Carlo dropout.
2. **Semantic Segmentation** - Explanation of the concept and examples of key semantic segmentation CNN models.
3. **Uncertainty in Neural Networks** - What uncertainty is in neural networks and variations of uncertainty.
4. **Active Learning** - What active learning is and example usages of active learning in machine learning.

2.1 Convolutional Neural Networks

CNNs are often used for visual recognition tasks. CNNs are mentioned as early as 1989 by Le Cun et al. [32] who classified handwritten ZIP codes with a CNN. The popularity of CNNs has grown in recent years. In 2012, Krizhevsky et al. [29] participated with a CNN model in the competition ILSVRC [39] and outperformed the previous state-of-the-art results. In 2015, a CNN achieved better results than humans [21] in the same competition.

CNNs usually consists of multiple layers, with varying functionality. The first layers detect lower-level features such as edges, shapes, and colors. Subsequent layers learn more complex features and the final layer output predictions. A typical structure of a CNN is shown in Figure 2.1. Modern networks consist of

a high number of learnable parameters with SegNet [7] around 30 million and VGG-16 [42] around 138 million. With so many parameters, training requires much computational power, long training time, and large datasets.

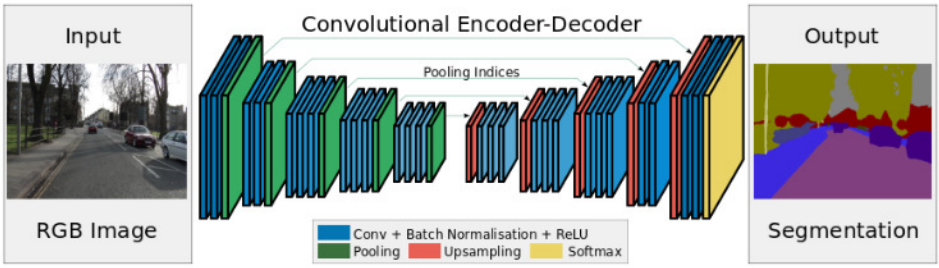


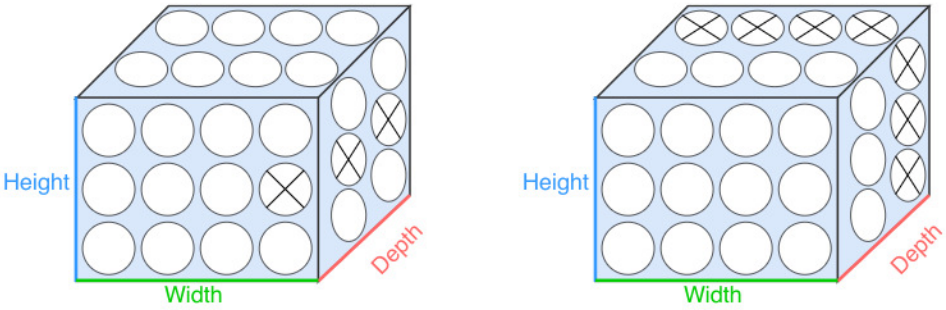
Figure 2.1: *SegNet [7] layer architecture. The encoder and decoder are VGG-16 [42] models with the fully connected layers removed. Image from SegNet [7], used with permission.*

2.1.1 Dropout

During training of neural networks, there is always a risk of overfitting to the training data [43]. Dropout is a regularization technique developed to prevent overfitting. Srivastava et al. [43] apply dropout to different types of neural networks. With dropout, the network train on a sampled version of the original network, meaning some units are ignored (*dropped*). Because the presence of each unit is unreliable, the impact/importance of the individual units is lowered. This has the effect that the co-adaptations to certain features in the training data are reduced, resulting in a lower generalization error.

Dropout works by *dropping* some units with probability p. At test time the units are included with their weights multiplied by the same probability p. This is done for each training case in a mini-batch. Dropout can be seen as adding noise to the network. Training networks with dropout take longer time than regular networks [43].

In the context of images as input, *regular dropout* works by dropping *pixels* in an image and setting the activation for all channels to 0. Tompson et al. [44] presents a new dropout technique, *spatial dropout*. They discovered that the performance did not improve for their CNN model by dropping pixels (regular dropout), due to the high correlation between adjacent pixels in images. In spatial dropout, entire features (channels) are dropped instead of pixels. Regular dropout and spatial dropout are visualized in Figure 2.2.



(a) Regular dropout *(b) Spatial dropout*

Figure 2.2: *Visualization of regular dropout and spatial dropout in Convolutional Neural Networks. Regular dropout drops pixels and spatial dropout channels.*

2.1.2 Monte Carlo Dropout

Aside from preventing the network from overfitting, dropout can also be used to estimate a model's average [43]. Model averaging is a way to combine multiple independently trained models. One way to create an *averaged model* is by averaging the models' unique parameters. The purpose of model averaging is to avoid local maxima and creating a more robust and reliable model that generalizes to test data better. A model's *true* average is derived by explicitly trying all possible parameter configurations of the model. With CNNs this is not feasible due to the complexity of the models and the evaluation time required. With a method called Monte Carlo dropout, a number of *dropout networks* are sampled and the mean of their predictions approximates the model's average. As the number of dropout networks $n, n \to \infty$, Monte Carlo dropout gets close to the true model average [43].

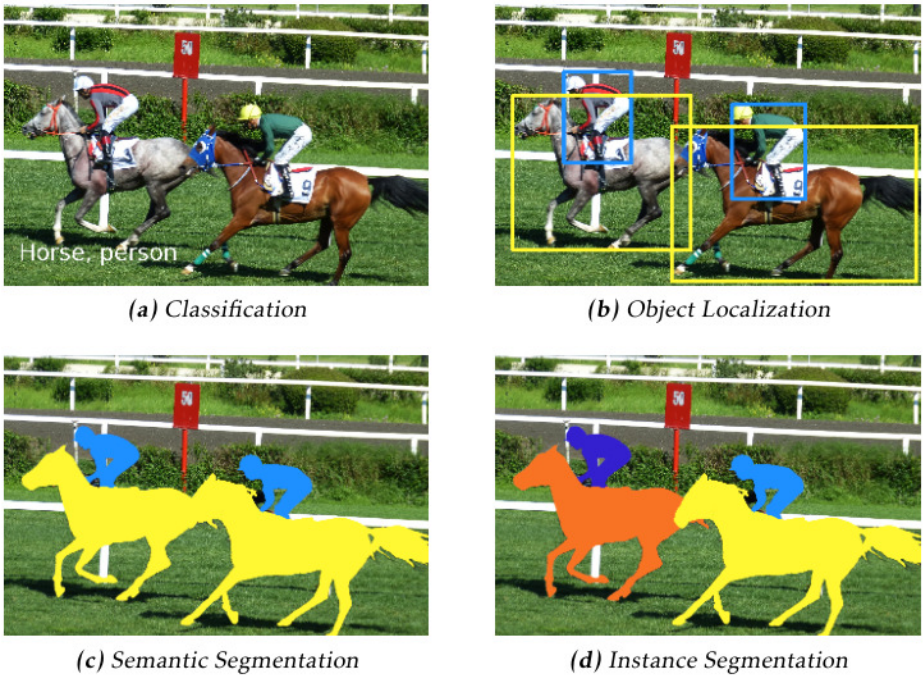
Instead of using the model's predictions to determine the class, the mean of Monte Carlo dropout can be used [16, 17, 25, 26]. Gal et al. [16] use a CNN for classification on the CIFAR-10 [27] dataset. With the mean of 100 sampled dropout networks, they compare the result to the default model, only trained with dropout. In their paper, Monte Carlo dropout performs significantly better than the default model predictions. However, Monte Carlo dropout is more expensive than the default model. Each sample is passed through multiple dropout networks, increasing both the processing time and the usage of hardware resources. In practical applications, this approach is not feasible as this takes too long time.

2.2 Semantic Segmentation

Semantic segmentation is one technique for *scene understanding*. There are several ways to *understand* or *interpret* the meaning of an image. As shown in Figure 2.3, scene understanding can be divided into four distinct categories [34]:

- **Classification:** Identify *which* class/classes that are present in the image.
- **Object Localization:** As classification with the addition to deciding *where* the object/objects are located. Locations are defined by rectangular boxes. Also known as *Object Detection*.
- **Semantic Segmentation:** Each pixel in the image is assigned a class. No notion of object/objects is present.
- **Instance Segmentation:** As semantic segmentation but identify instances of objects.

Supervised machine learning models, and in particular CNN models, are commonly used for semantic segmentation. The CNN model trains with annotated images and learns features to understand the images' content. State-of-the-art results for semantic segmentation are achieved by using CNNs [4, 35].



(a) Classification *(b)* Object Localization

(c) Semantic Segmentation *(d)* Instance Segmentation

Figure 2.3: *Four variations of scene understanding. Original image from ABSFreePic [3], public domain.*

2.2.1 Encoder-decoder Architecture

In CNNs for semantic segmentation, networks are commonly divided into two distinct parts, encoder and decoder [7, 22, 25, 38]. The encoder takes the input image and downsamples the image into a high-dimensional representation while the decoder upsamples the image to the original image size with pixel predictions. One popular network with a distinct encoder-decoder structure is SegNet, Section 2.2.3, Figure 2.1.

2.2.2 FCN - Fully Convolutional Networks

Long et al. [35] develops a Fully Convolutional Network (FCN) for semantic segmentation. FCN does not contain any fully connected layers, which are common in classification networks. In a fully connected layer, every neuron in the previous layer is connected to every neuron in the following layer. This can be seen as regular convolution but with a kernel that covers all neurons. Fully connected layers are not suitable for semantic segmentation as the spatial information of objects is ignored. Instead of fully connected layers, Long et al. use regular convolutional layers followed by a *backwards convolutional* layer (in the literature also called *deconvolutional* or *transposed convolution*). The backwards convolutional layer works like a regular convolutional layer with the addition that they upsample the image [35].

In FCN are *skip connections* [8] used, where some connections skip one or more layers. Passing information through a layer modifies it and can destroy/distort valuable information. Skip connections add the output from previous layers to the current, keeping this information intact. All their evaluated models improve with skip connections [35].

2.2.3 SegNet Architecture

Badrinarayanan et al. [7] base their semantic segmentation network on the VGG-16 model [42]. VGG-16 consists of 16 layers where the 13 first are convolutional layers and the 3 following are fully connected layers. SegNet is built with two VGG-16 models, with the fully connected layers removed. The first model acts as the encoder and the second as the decoder. They are attached to each other with the decoder reversed to the encoder, as seen in Figure 2.1.

2.2.4 ENet - Efficient Neural Network

Paszke et al. [38] present a computationally efficient network (ENet) for semantic segmentation. ENet is based on the network architecture ResNet (Residual Networks) [22], created for image classification. In 2015, ResNet won *ImageNet detection* and *ImageNet localization* in the competition ILSVRC [39] as well as *COCO detection* and *COCO segmentation* in the competition COCO [34].

ENet and ResNet utilize *bottleneck* modules, see Figure 2.4. The bottleneck module consists of two branches. The first branch has: a projection layer, a convolutional layer, and a projection layer again. The second branch is a skip connection

from the previous bottleneck module. At the end is the two branches element-wise added together. The purpose of the projection layers is to reduce dimensionality before the convolutional layer in the middle and then restore the dimensionality again. As an effect of the reduced dimensionality, each bottleneck module consists of relatively few parameters. This lowers the total amount of learnable parameters and allows both deeper network architectures and a speedup in training and inference time. The number of parameters in ENet, FCN, and SegNet are shown in Table 2.1.

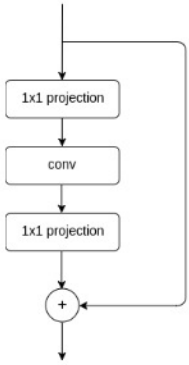


Figure 2.4: *Bottleneck module with skip connections, used in ENet [38] and ResNet [22].*

Name	# weight layers	Input size (w x h)	Parameters (millions)
FCN	16	500 x 500	134.5
SegNet	26	480 x 360	29.45 [25]
ENet	29	512 x 512	0.37

Table 2.1: *Model comparison of FCN [35], SegNet [7] and ENet [38]. All models are developed for semantic segmentation.*

A more thorough explanation of the ENet architecture is found in Appendix A. ENet is the network architecture used in the thesis.

2.2.5 Road Segmentation

Road segmentation or *road scene segmentation* is a variation of semantic segmentation. Images in these datasets are captured as viewed from a vehicle and contain classes related to road scenes. Road segmentation is a crucial part of the development of autonomous vehicles. Famous datasets for this task are CamVid [9], KITTI [19], Cityscapes [12], and Apollo [23]. Examples of classes in road segmentation can be *road*, *lane markings*, *pedestrians*, *vehicles* and *traffic signs*.

2.3 Uncertainty in Neural Networks

Machine learning models are trained to solve complex and abstract tasks. The model's prediction is often interpreted in terms of performance; how well does the model's prediction compare to the ground truth. A pixel's class is determined from the class probabilities, without taking into account the potential uncertainty in the prediction. For example, model A predicts the probabilities: 98%, 1%, 1% and model B: 40%, 30%, 30%. The first class is chosen by both models. However, more information is available, the *certainty* in the prediction. One could argue that model A is more certain. Incorporating a notion of uncertainty could improve the reliability of the predictions, aiding the decision making.

In general models, many types of uncertainties exist. Most uncertainties can be categorized as either *aleatoric* or *epistemic* [14]. Aleatoric uncertainty is the internal uncertainty originating from noise in the observations. Aleatoric uncertainty cannot be solved by using more training data. Epistemic uncertainty, also called *model uncertainty*, is uncertainty stemming from the model of choice. The epistemic uncertainty can be modeled by observing how much the model's parameters vary given different data. The epistemic uncertainty comes from either a lack of data or insufficient knowledge of the problem. A too simplified model can have problems with representing the data, requiring a more complex model. Too little data can also induce epistemic uncertainty, which can be explained away with more training data [26].

2.3.1 Prediction Uncertainty

Prediction uncertainty is estimated by analyzing the softmax probabilities from the final layer. No further calculations need to be performed, making prediction uncertainty a quick method.

Wang et al. [46] train a CNN for classification. They use their model, W, to rank unseen samples by the most *informative*. To represent informativity, they evaluate three methods, which analyzes the model's prediction. The sample is denoted x, the prediction y, and the class j.

- **Least Confidence:** For the most probable class, how sure is the model on sample i. Ranked in ascending order:

$$lc_i = \max_j p(y_i = j \mid x_i; W) \tag{2.1}$$

- **Margin Sampling:** In the prediction, the top two class probabilities are called j_1 and j_2. Ranked in ascending order:

$$ms_i = p(y_i = j_1 \mid x_i; W) - p(y_i = j_2 \mid x_i; W) \tag{2.2}$$

- **Entropy:** Measures the entropy by using all class probabilities. Ranked in descending order:

$$en_i = -\sum_{j=1}^{J} p(y_i = j \mid x_i; W)\, log\, p(y_i = j \mid x_i; W) \tag{2.3}$$

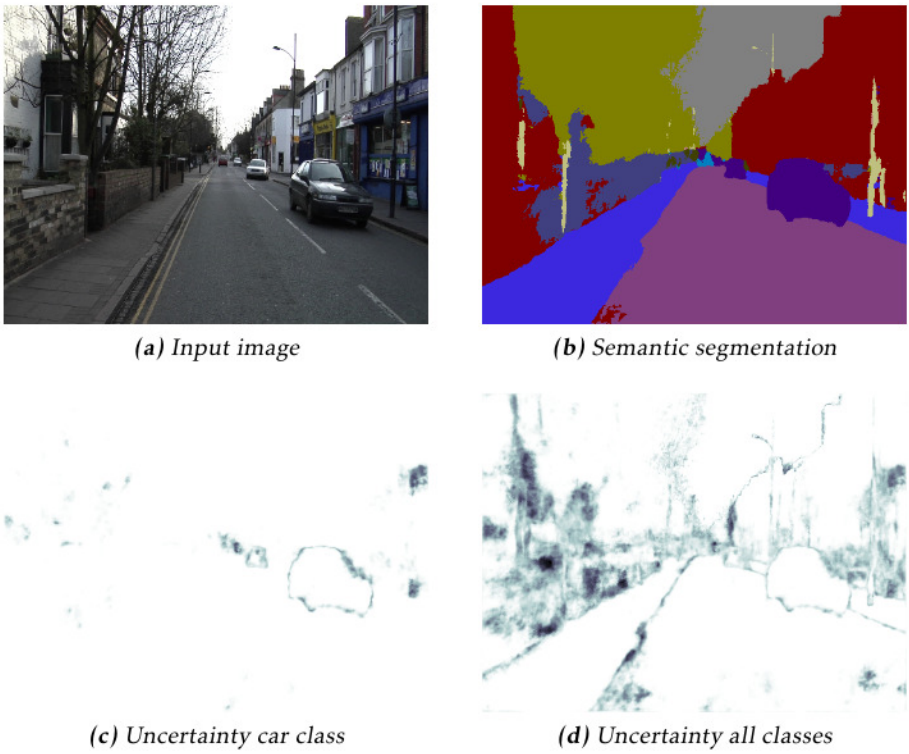
2.3.2 Monte Carlo Dropout Uncertainty

An image is passed through a *dropout network*, where a dropout mask is applied to the network. With Monte Carlo sampling, collecting a number of predictions, the epistemic uncertainty can be approximated.

Kendall et al. [25] analyze epistemic and aleatoric uncertainty in Bayesian deep learning. Their network, Bayesian SegNet, is based on their previous model SegNet [7], described in Section 2.2.3. Modeling the epistemic uncertainty analytically is not possible, and therefore, an approximation is used. The model's epistemic uncertainty is approximated by using Monte Carlo dropout [43] during inference.

At test time, the unseen samples are passed through several *dropout networks* and the softmax predictions are analyzed. Each pixel's class is determined by the mean of the predictions.

The uncertainty from Monte Carlo dropout is visualized in Figure 2.5. For each pixel and class, the variance in the predictions is calculated, working as an estimate of the model's uncertainty. Figure 2.5 **c** shows the uncertainty for the car class. By averaging over the classes, the uncertainty for each pixel is determined, Figure 2.5 **d** shows the uncertainty for all classes.



(a) Input image

(b) Semantic segmentation

(c) Uncertainty car class

(d) Uncertainty all classes

Figure 2.5: *Uncertainty visualized in Bayesian SegNet [25] using the dataset CamVid [9]. Images from Bayesian SegNet tutorial [1], used with permission.*

2.4 Active Learning

Active learning is a technique where the learning algorithm participates in the selection of its own training data. In a literature survey presented by Settles [41], the main motivation for active learning is presented. In supervised machine learning, annotated data is required. In typical applications is the amount of unannotated data huge and the process of annotating data is expensive and takes a long time. Therefore, active learning has emerged, trying to limit the amount of annotated data by letting the algorithm select its own training samples.

Active learning is an iterative process where a machine learning model is trained on an initial set of annotated data. With the trained model and selection criteria, new unannotated samples are identified and added to the current training set. The additional samples are annotated by an *Oracle* and the model retrains on the updated dataset. Generally, the Oracle is a human annotator that is presumed to always annotate samples correctly. The active learning loop is repeated until either all samples are used in the model or the model performs well enough.

Figure 2.6 visualize the active learning loop. A successful implementation of active learning achieves at least the same performance as random selection, but with fewer annotated training samples.

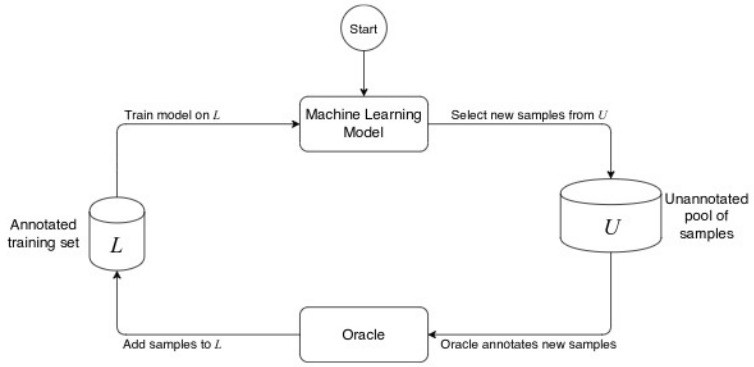


Figure 2.6: *Active learning loop. After training the model on the labeled training set L, the model select new samples from U, the Oracle annotate them and add them to the training set L.*

2.4.1 Querying Samples

The selection of new samples is called *queries*. Pool-based sampling is often used in supervised machine learning [13, 45, 46]. In pool-based sampling is the remainder of the pool analyzed and ranked according to some selection criteria and the most informative samples are chosen [41].

In theory, new samples are added individually and the model is retrained with the additional sample. However, due to the time required to train many machine learning models and the usually large number of samples, new samples are added in batches.

2.4.2 Multi-Class Active Learning for Image Classification

Joshi et al. [24] created an active learning framework for a Support Vector Machine (SVM). Three different tasks are considered: character classification, object recognition and scene recognition. Informative samples are selected by either *entropy* or *margin sampling* (in their paper called *Best-versus-Second-Best*), defined in Section 2.3.1.

In character classification, the SVM trains on three different datasets, Pendigits, USPS and Letter [15]. Margin sampling significantly outperforms entropy and random selection in all three cases. In both digit datasets, entropy demonstrates higher accuracy than random selection.

They also evaluate object recognition on the Caltech-101 [30] dataset. Entropy and random selection are indifferent while margin sampling quickly achieves much higher accuracy.

2.4.3 CEAL - Cost-Effective Active Learning

Wang et al. [46] developed a framework that applies active learning to an image classification task using a CNN. Their network is trained on the datasets Caltech-256 [20] and CACD [10]. After initializing their CNN model with 10% of the training data, it goes through 20 active learning iterations. After each active learning iteration, the algorithm selects new samples according to one of the criteria: *least confidence*, *margin sampling* or *entropy*, defined in Section 2.3.1. These samples are added to the training set which the network is fine-tuned on, completing the active learning iteration.

In CEAL, they also utilize *pseudo-labels*, shown in Definition 2.4. During inference, samples with high confidence and an entropy lower than a threshold δ are picked out. These samples are annotated with the model's prediction, called a pseudo-label. In the following active learning iteration, these high confidence samples are treated as correctly annotated samples. At the end of the active learning iteration, they are returned to the unannotated pool. With pseudo-labels, the unannotated samples contribute to the training. This method, with using both annotated and unannotated data, is called weakly supervised machine learning. The threshold parameter δ decays after each active learning iteration, increasing the performance requirement of high confidence samples.

$$
\begin{aligned}
j^* &= arg\max_j \, p(y_i = j \mid x_i \,;\, W) \\
y_i &= \begin{cases} j^*, & en_i < \delta \\ 0, & \text{otherwise} \end{cases}
\end{aligned}
\tag{2.4}
$$

Wang et al. [46] state that all their mentioned uncertainty criteria perform better than random selection. Sener and Savarese [40] (Section 2.4.5) include CEAL in their analysis, where the model performs worse than random.

2.4.4 DBAL - Deep Bayesian Active Learning for Image Data

Gal et al. [18] evaluate active learning with CNNs on the digit classification dataset MNIST [33]. They discuss the problem with combining active learning and CNNs. CNN models usually take a long time to train, limiting the number of active learning iterations, leading to large batches when selecting samples. In theory, active learning should train on only one additional sample each active learning iteration. For the MNIST dataset, they ran the active learning framework for 100 iterations, selecting 10 additional samples every time. Monte Carlo dropout based methods with varying acquisition functions quickly outperform random selection.

Their model, the default MNIST implementation in Keras [11] contains only 2 convolutional layers and 2 fully connected layers, operating on an input size of 28x28 pixels. With more advanced problems, like semantic segmentation on

large input images, is it not feasible to selects so few samples in each active learning batch.

2.4.5 Core-Set for Convolutional Neural Networks

Sener and Savarese [40] apply active learning to CNNs. They try to find a core-set, the smallest set of samples that cover all of the sample space. A small initial dataset is annotated. The remaining pool is analyzed with an unsupervised subset selection algorithm, in this paper based on a greedy k-center algorithm.

K-center is a clustering algorithm that tries to find a set of k core-points and strives for minimizing the maximum distance from all points to their closest core-point [6]. In each active learning iteration is b additional core points selected by iteratively, greedily removing the other points based on an excluding criteria. The *Oracle* is queried for the new samples' annotation and the samples are then added to the current training set.

Their CNN model is trained on the CIFAR-10 [27], CIFAR-100 [28], SVHN [37], and Caltech-256 [20] datasets. They compare their model against several others, including random selection, CEAL (Section 2.4.3), DBAL (Section 2.4.4), and other uncertainty based methods, achieving state-of-the-art results for all datasets.

They demonstrate that random selection is in many cases much more effective than the uncertainty based methods and they provide a hypothesis for why that is the case: Due to the long training time, samples are added in batches instead of individually. This has the effect that a batch of uncertain samples has a high correlation between them. With uncertainty selection, the samples do not cover the complete sample space, resulting in low generalization to the test set.

It is worth noting that in their model evaluation they only investigate 5 dataset sizes: the initial size, final size with all samples, and three intermediate points. It would be interesting to see how all models perform with smaller steps in data selection sizes.

3
Method

This chapter describes how the work in the thesis was carried out. Divided into eight parts:

1. **Problem Description** - the task in the thesis explained in more detail.
2. **Dataset** - description of the dataset in the thesis.
3. **Performance Metrics** - metrics to evaluate the networks' performance are defined.
4. **Dataset Size** - motivation for reducing the dataset.
5. **Active Learning Framework** - explanation of the how active learning is used in the thesis and description of the active learning framework.
6. **Model** - model description and implementation details.
7. **Uncertainty** - how uncertainty is used for informative sample selection.
8. **Selection Criteria** - a summary of the selection criteria that will be part of the results.

3.1 Problem Description

In the thesis, active learning is applied to *road segmentation*. A variation of semantic segmentation with images related to a road environment. The CNN model is trained on a small initial dataset. Every *active learning iteration*, new samples are added to the training and validation sets and the model is then re-trained. Samples are added by using a metric which ranks their *informativity*. Finding/implementing an effective and suitable informativity metric is one of the main tasks in the thesis.

Monte Carlo dropout will be utilized to try to identify informative samples. At the end of an active learning iteration, the remaining samples in the pool are passed through the CNN model with a dropout mask applied. By analyzing the predictions, the model's uncertainty on the samples is assessed. Other methods to estimate model uncertainty will also be investigated and evaluated.

Active learning is compared to a baseline method that selects new samples randomly. If active learning achieves (at least) the same performance as baseline, but with fewer annotated samples, active learning is deemed successful.

3.2 Dataset

An internal dataset at Veoneer is used, containing in total around 54 000 annotated images. The dataset is beforehand divided into training, validation, and test sets with sizes shown in Table 3.1.

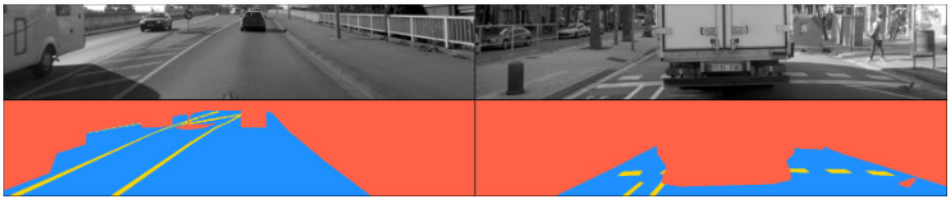
Type	# samples	Size
Training	37 515	68.6%
Validation	9 785	17.9%
Test	7 361	13.5%
Total	54 661	100%

Table 3.1: *Sizes of the training, validation and test set.*

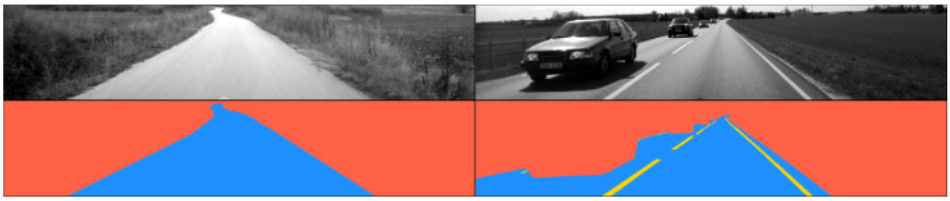
All images in the dataset have the size: 1024x208. Four example images, along with their ground truth annotation, are shown in Figure 3.1. The available classes in the dataset are listed in Table 3.2.

Name	Size	Description
Road	43.4%	Drivable surface in front of the car
Lane	2.2%	All forms of lane markings
Void	54.4%	Pixels that neither belongs to road or lane

Table 3.2: *Classes in the dataset with their respective size and description.*



(a) Image 1 *(b) Image 2*



(c) Image 3 *(d) Image 4*

Figure 3.1: *Four example images in the dataset with their ground truth annotation. Blue: road, yellow: lane, red: void.*

3.2.1 Dataset Subset Partitioning

The samples in the dataset have metadata about the conditions when the samples were recorded. Three of these variables are:

- **Road Type** - sample recorded in a city or rural area.
- **Time of Day** - sample recorded during daytime or nighttime.
- **Road Cover** - sample recorded on a dry or wet (rainy/snowy) road.

From these 3 binary variables, 8 subsets are derived, shown in Table 3.3. The training, validation, and test set all follow this subset distribution. It is expected that the network performs better/worse on some of these subsets. In general are nighttime, cities and wet roads harder than their opposites.

3.3 Performance Metrics

All results presented in the thesis are calculated on the test set. To evaluate the networks' performance, two metrics are used in the thesis: *F1 score* and *FreeSpace TP*.

3.3.1 F1 score

The F1 score is the main metric used in the thesis. A binary image is produced for each class in the dataset. The true positive (TP), true negative (TN), false positive (FP) and false negative (FN) rates are determined. The F1 score is the harmonic mean of precision and recall:

#	Size	Road Type	Time of Day	Road Cover
1	14.9%	City	Day	Dry
2	9.6%	City	Day	Wet
3	4.4%	City	Night	Dry
4	2.9%	City	Night	Wet
5	36.8%	Rural	Day	Dry
6	14.2%	Rural	Day	Wet
7	11.8%	Rural	Night	Dry
8	5.4%	Rural	Night	Wet

Table 3.3: *Distribution in the dataset. The training, validation, and test set all follow this distribution.*

$$\mathrm{F1} = 2 * \frac{\mathrm{Precision} * \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{3.1}$$

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \quad \mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3.2}$$

Precision: of all pixels classified as class c, how large part of them are correct. **Recall**: of all pixels that belong to class c, how large part of them were classified as class c.

3.3.2 FreeSpace TP

Measures the amount of *free space* in front of the car. The input image is evenly divided into a number of sectors, where each sector contains a number of image columns. From the bottom of the image and for each sector, the ground truth marks where the first void pixel is located. In the prediction, if the first void pixel in a sector is within a certain pixel distance from the ground truth, that sector is marked true positive (TP). The FreeSpace TP value of an image is the number of true positive sectors divided by the total number of sectors as stated in Definition 3.3.

$$\mathrm{FreeSpace\ TP} = \frac{\#\ \mathrm{TP\ sectors}}{\#\ \mathrm{sectors}} \tag{3.3}$$

In the thesis, the images are split into 64 sectors and a distance less than, or equal to, 10 pixels is counted as TP.

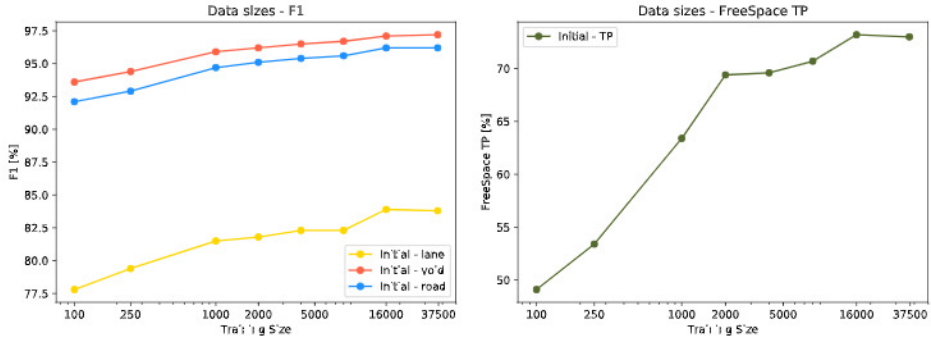
3.4 Dataset Size

An assumption in the thesis is that the network's performance gradually increases with more training data. Having a too small increase in performance, it will be hard to compare sample selection methods to each other. Especially determine if one is better than the other. During early exploration in the thesis, a number of networks are trained with varying sizes of the training set. The size of the validation set is roughly 26% of the training set, following the ratio in Table 3.1. All networks trained in the thesis follow this ratio.

The networks are trained in parallel with the training and validation samples defined in advance. The samples in the previous networks are included in the succeeding ones. Every network takes around 2 days to train. The dataset sizes and test set performance for all networks are shown in Table 3.4 and Figure 3.2.

# training	# validation	F1-road	F1-lane	F1-void	FreeSpace TP
100	26	92.1	77.8	93.6	49.1
250	65	92.9	79.4	94.4	53.4
1 000	260	94.7	81.5	95.9	63.4
2 000	520	95.1	81.8	96.2	69.4
4 000	1040	95.4	82.3	96.5	69.6
8 000	2080	95.6	82.3	96.7	70.7
16 000	4160	96.2	83.9	97.1	73.2
37 515	9785	96.2	83.8	97.2	73.0

Table 3.4: *Performance of the network depending on dataset size. F1 score and FreeSpace TP are given in percent. Data visualized in Figure 3.2.*



(a) *F1 score* ***(b)*** *Free Space*

Figure 3.2: *Performance of the network. Number of training samples on the x-axis, performance in percent on the y-axis. Logarithmic x-axis.*

Beforehand the performance was expected to be more dependent on the size of the training set. For the F1 score, the performance does not increase that much given more training data. Training with 100 training samples (0.3% of the complete training set), the F1 score for the classes *road*, *lane*, and *void* are 92.1%, 77.8%, and 93.6% respectively. Training with 37 515 data points, the performances are 96.2%, 83.8%, and 97.2% respectively, yielding an average increase of 4.6%. For FreeSpace TP, an upwards trend is more noticeable up to 2 000 training samples. After that the performance flat out. Though, it is worth noting that even marginally increases in performance is regarded as important to meet typical requirements in forward-looking cameras. The performance for F1-road and F1-void are over 90%, meaning the system performs well even with such a small dataset.

This is likely due to the fact that most road scenes in the training set are in many aspects similar. The road is almost always in front of the car and all non-road regions are in the upper parts of the image. To investigate this, all training samples are averaged and showed in Figure 3.3. First, in **a**, is the average of all pixels shown and secondly, in **b**, is the most common class set as the true class.

Should the network always predict the image in Figure 3.3 **b**, the F1 score for *road*, *lane*, and *void* are 80.2%, 0.0%, and 83.1%. *Lane* is not present in **b** and receives the expected score of 0. But, both *road* and *void* receive over 80% in F1 score which is remarkably high. The FreeSpace TP value is 12.7%. Such a low value is expected as this measure only allow a narrow margin of error.

This information needs to be kept in mind when interpreting the performances in this thesis.

(a) *Pixel distribution for all classes*

(b) *Most common class for each pixel*

Figure 3.3: *Class distribution of the training set.* ***a:*** *raw distribution for the classes. Note the faint yellow lines from the lane class.* ***b:*** *the most common class for each pixel. Road dominates the lower regions of the image and void the upper regions.*

3.4.1 Dataset Reduction

The inference to find informative samples takes a lot of time and requires large amounts of memory. Having 100 dropout networks, inference over 47 300 samples (training and validation the first active learning iteration) takes around 25 hours. Each prediction image takes up around 2.5 MB, totaling around 11.8 TB of temporary storage.

For these reasons, with the main objective to see a larger trend in the performance, a decision was made to reduce the training set to include 10 000 images. The validation set is also reduced with the same ratio to 2 600 images.

3.5 Active Learning Framework

In this project, a large portion of the work is dedicated to building an easy-to-use, reliable framework. A project file is given as input to the framework. The project file defines how the network should be trained; which initial samples to use, how to select new samples and all other necessary model parameters.

3.5.1 Active Learning Iterations

The network is trained for a number of active learning iterations. For each active learning iteration, the model is initialized, compiled, and trained on the training set. In theory, samples should be added individually. However, due to the model's complexity, and long training time, samples are added in batches. A pre-defined number of samples are added each active learning iteration ending with all samples used in the last active learning iteration. Apart from the dataset network in Table 3.4, all networks in the thesis are trained for 13 active learning iterations. The number of samples in the active learning iterations are shown in Table 3.5.

3.5.2 Active Learning Flowchart

The active learning algorithm is visualized in Figure 3.4. The pool of available unannotated samples is denoted D^U, and the annotated training samples D^L. In iteration i, k_i samples are selected. The *Oracle* provides the samples' correct annotation. In practice, all samples are already annotated and the *Oracle* is a metaphor for giving the algorithm the samples' ground truth. Variations on the Selection Criteria (*SC*) are defined in Section 3.8.2.

ALI	# Training	# Validation
1	250	65
2	500	130
3	750	195
4	1 000	260
5	1 500	390
6	2 000	520
7	2 500	650
8	3 500	910
9	4 500	1170
10	5 500	1430
11	7 000	1820
12	8 500	2210
13	10 000	2600

Table 3.5: *Sizes of the dataset during the training of the CNN. The validation sizes are approximately 26% of its respective training sizes. Note that the number of added samples are increased every third active learning iteration with the sizes: 250, 500, 1000 and 1500.*

3.5.3 Network Evaluation

During training, the training and validation samples are saved for every active learning iteration as well as the network's weights. All samples are always used in the test set.

The evaluation is performed for every active learning iteration, the procedure works as follows:

1. Predictions for the validation and test sets are produced.
2. For the validation predictions are the F1 score and FreeSpace TP computed. Multiple thresholds are tested and the optimal thresholds are stored.
3. F1 score and FreeSpace TP are computed for the test set using the thresholds from step 2.
4. The results, from the test set, of every active learning iteration are extracted and saved in a network results file.

The complete evaluation procedure takes around 9 hours to run.

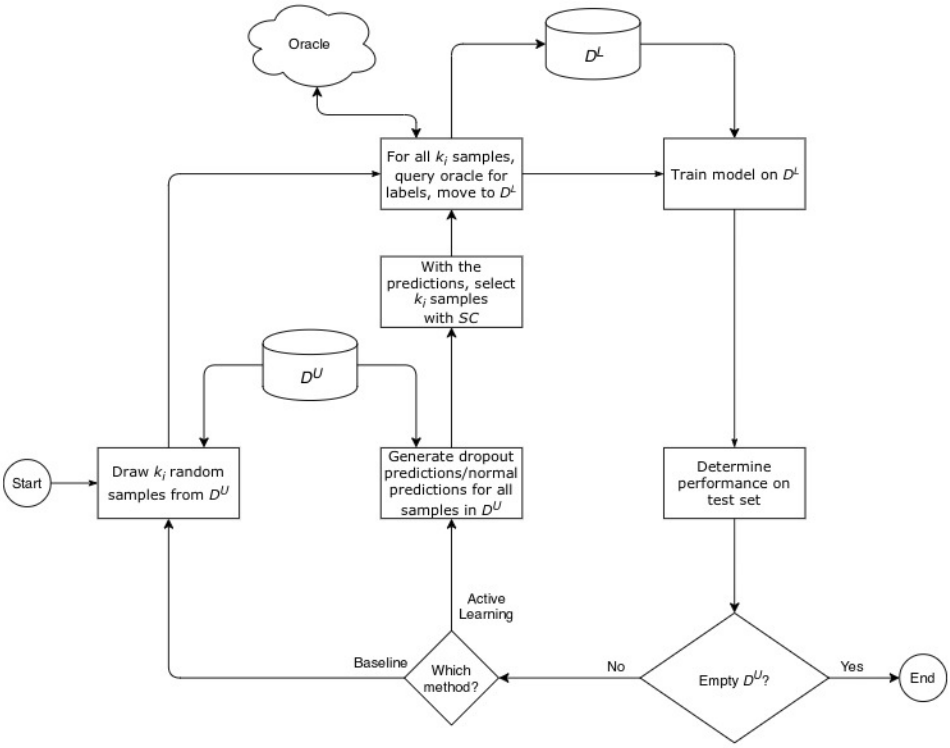


Figure 3.4: *Flowchart for the active learning algorithm. The algorithm runs for several active learning iterations, either until satisfactory performance is achieved or all samples are used in the model.*

3.6 Model

The network architecture used in the thesis is ENet [38]. ENet is built up of several bottleneck layers as described in Section 2.2. In total 29 layers, where 23 layers are encoder layers and 6 decoder layers. ENet is trained with spatial dropout. A thorough explanation of ENet is found in Appendix A.

An implementation of ENet is available at Veoneer. Written in Keras [11] with TensorFlow [5] as the backend. The networks are trained with multiple GPUs, all benchmarking are performed on a GTX 1080 Ti graphics card.

3.6.1 Pseudo-epochs

In the context of CNNs, one epoch means that the model has trained on every training sample once. The number of *samples per epoch*, *SPE*, is defined by the number of batches multiplied with the batch size.

In active learning, the training set is iteratively increased in size (number of batches grows). Having the effect that subsequent active learning iterations are

allowed more training iterations. To be fair, each active learning iteration should use the same amount of training iterations. Therefore, the concept of *pseudo-epochs* is introduced. In a pseudo-epoch, *SPE* is independent of the training size and fixed throughout the active learning iterations. When training, the samples in the current training set are drawn with replacement until *SPE* samples are reached.

Before reducing the training set, it contained 37 515 samples. Network parameters such as the number of epochs are tuned against the number of training samples. Therefore is *SPE* set to 37 500. In the remainder of this thesis are all epochs in fact pseudo-epochs.

3.6.2 Learning Rate Decay

In the thesis, learning rate decay is used. The settable parameters are the *patience* and the *decay rate*. If the validation accuracy does not improve for 6 consecutive epochs, the learning rate is multiplied by the factor 0.5. The learning rate is reset at the start of each active learning iteration.

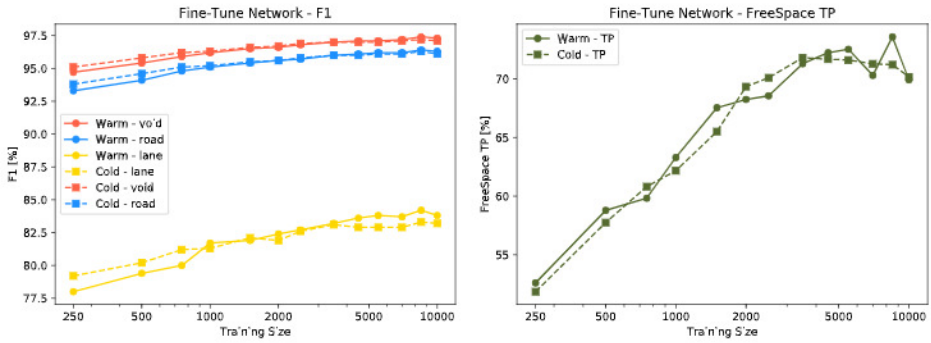
3.6.3 Fine-tune Network

When training CNNs, much time is spent on learning to detect basic features like edges and shapes. A network's weights are usually initialized randomly. Another way to lower the long training time is to initialize the current active learning iteration with the weights from the previous one.

However, when evaluating the performance, a potential risk needs to be considered. With inherited weights, the (assumed) increase in performance can either come from using more samples, or from the fact that the network has had more training iterations to fine tune the weights. This could lead to a performance increase without actually *learning* more from the additional data but just from more training iterations.

To investigate this potential error source, two different networks are trained. One where all iterations are trained with randomly initialized weights. One where the weights are inherited from the previous active learning iteration. In the latter, early stopping is used with patience 8 and $\delta_{min} = 0.0002$, explained in Section 3.6.4. Both networks are trained on identical samples. The results of these trainings are shown in Figure 3.5.

The results show that the F1 score is very similar between the networks. The performance of FreeSpace TP is fluctuating a bit more. In the first active learning iteration, the networks are trained under the same conditions. The performance differences come from a stochastic training procedure, gradient computations are non-deterministic. In the thesis, the weights from the previous active learning iteration are used.



(a) F1 score *(b) Free Space*

Figure 3.5: *Initialized every active learning iteration with new weights versus inheriting weights from the previous active learning iteration. Denoted Cold or Warm, respectively. Number of training samples on the x-axis, performance in percent on the y-axis. Logarithmic x-axis.*

3.6.4 Early Stopping

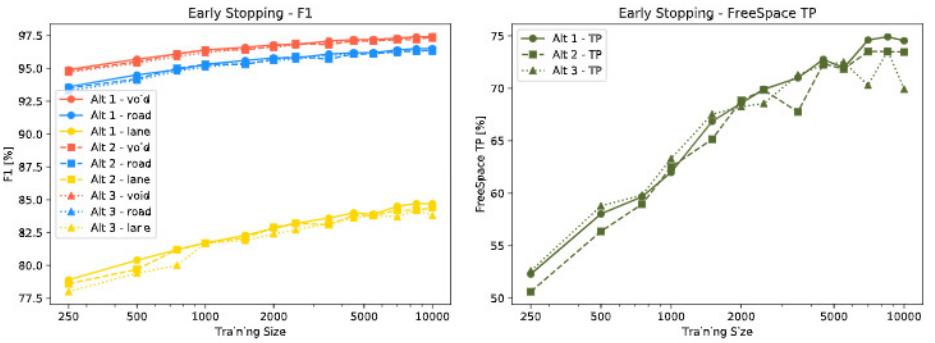
One active learning iteration runs for 40 epochs, taking around 2 days to complete. A model with 13 active learning iterations, training one complete network takes around 26 days (not accounting for uncertainty estimation). This constitutes a big problem, with too long training time, the number of networks in the thesis will be heavily limited.

To lower the training time, early stopping is investigated. If the validation accuracy does not improve for a number of epochs, the training stops and the best epoch is returned. This parameter is called *patience*. The metric increase needs to be larger than δ_{min} to be counted as an improvement.

Three variations are tested and presented in Table 3.6. One without, and two with early stopping. All networks are trained on identical samples and with inherited weights. Their F1 score and FreeSpace TP performance are shown in Figure 3.6.

Name	Patience	δ_{min}	Time
Alt 1	-	-	30 days, 1 hours
Alt 2	8	0	15 days, 16 hours
Alt 3	8	0.0002	10 days, 8 hours.

Table 3.6: *Variations of early stopping. Early stopping greatly reduces the training time. Performance is shown in Figure 3.6.*



(a) F1 score *(b) FreeSpace TP*

Figure 3.6: *Evaluation of early stopping. Number of training samples on the x-axis, performance in percent on the y-axis. Logarithmic x-axis.*

Their performances are similar, especially for the F1 score. The training time for Alt 1 is a month. A training time that long is not feasible for a 20-week thesis. Alt 3 is chosen, with the *patience* set to 8 and δ_{min} to 0.0002. Early stopping parameters are reset at the beginning of every new active learning iteration.

3.7 Uncertainty

In active learning, new samples are identified based on their informativity. For this, the main idea in the thesis is to use Monte Carlo dropout. Monte Carlo dropout is computationally demanding and time-consuming. Aside from Monte Carlo dropout, other methods that analyze the model's default predictions are investigated. These methods are *least confident*, *margin sampling* and *entropy*, defined in Section 2.3.1. They are significantly simpler, both in implementation but also in the aspect of computing resources.

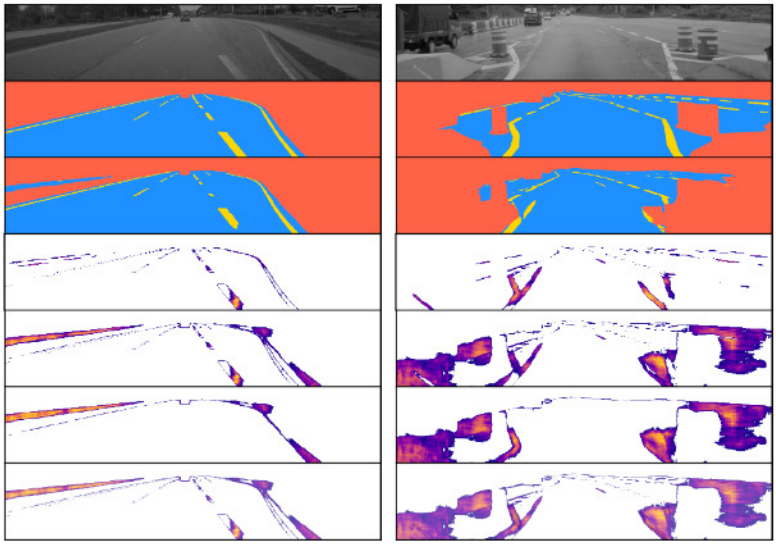
After an active learning iteration, inference is made over the remaining samples in the training and validation sets. The presented methods are developed and used for classification, not semantic segmentation, and therefore carried out pixel by pixel. To aggregate pixel uncertainties to sample uncertainty, the mean over all pixels is calculated. The result of the inference is a ranked list of all samples' uncertainties. In the next active learning iteration, the new samples are chosen from that list.

3.7.1 Monte Carlo Dropout Uncertainty

Monte Carlo dropout uncertainty tries to represent epistemic uncertainty, the model's uncertainty. All networks in the thesis are trained with spatial dropout. If the model is *certain* on a sample, that knowledge is distributed in the model and is, for that reason, robust to changes [43]. Dropping units in a *certain model*,

should not change the model's prediction to a large degree. In contrast, for a sample that differs from the training data, few units can greatly influence the model's current prediction. By dropping these units the predictions will more likely differ, indicating an *uncertain* model. The samples, that show a high variance in the predictions are deemed informative. By adding them to the training set, the model is expected to learn the *difficult* situations quicker than by selecting samples randomly.

A new model is created with permanent dropout and the weights are loaded from the regular model. Every sample is passed through the network n times and the predictions are stored. The pixel uncertainties are derived by calculating the variance from the n predictions, for every pixel and class, as in Section 2.3.2. Two images with Monte Carlo dropout are shown in Figure 3.7 with the uncertainty for each class present. Succeeding images in the thesis will only display uncertainty for all classes combined.



(a) *Image 1* ***(b)*** *Image 2*

Figure 3.7: *Two images from the test set with Monte Carlo dropout uncertainty. From the top: input image, ground truth, prediction, dropout uncertainty; lane, road, void, all classes. Note that the model's uncertainty to some extent correlates with the model's incorrect predictions, in particular, the strip of road to the left in image* ***a****. This network is trained on 7000 samples,* $n = 100$.

Number of Dropout Networks

The number of dropout networks, n, that each sample is passed through, is an important parameter. Time and storage requirements are proportional to n. The images need n x 2.5 MB of temporary storage and around n x 0.02 seconds of processing time. For 10 000 samples and $n = 100$, this sums up to around 6 hours of processing time and 2.5 TB of temporary storage. By lowering the parameter n, the inference is quicker and requires less computing resources.

To correctly investigate how the model performs depending on n, active learning networks should be run with different values of n. Though, this is not feasible as a full training takes around 10 days to complete (excluding inference time). Instead, an analysis is performed of samples that *would* have been selected.

Multiple values of n are tested and the samples are ranked by their informativity and the number of overlapping samples is recorded. The evaluation is performed after the first active learning iteration (trained with 250 samples). This network is intuitively the most uncertain network as so few samples are in the training set and many road scenes are unknown for the network. All samples in the remaining training set are analyzed, in total 9 750 images. For a range of n, the top 250 and top 1 000 samples are stored and ranked by their uncertainty. As a reference, in the following active learning iteration, 250 new samples will be added to the training set. The number of overlapping samples, compared to $n = 200$ are presented in Figure 3.8. At $n = 20$, 87% for 250 and 90% for 1000 of the samples overlap, but only takes up a tenth of the resources that $n = 200$ require.

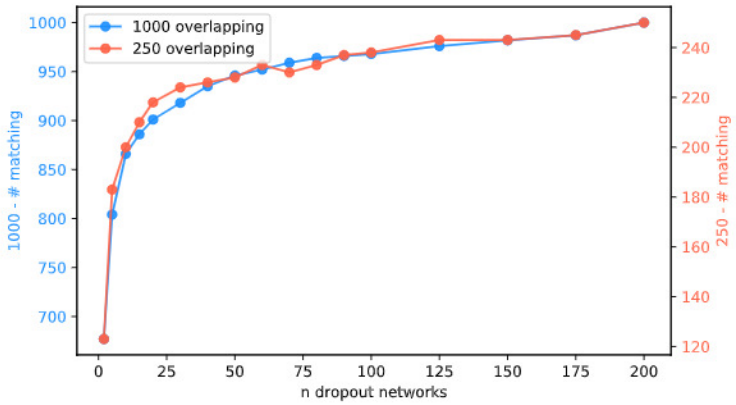
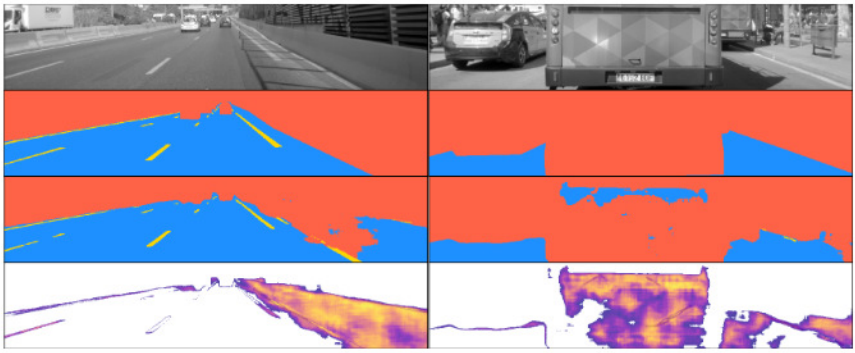


Figure 3.8: *Overlapping samples, red: 250 samples, blue: 1 000 samples.*

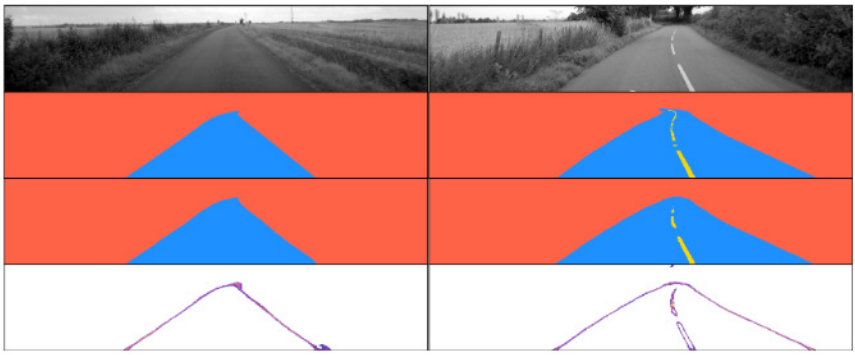
It is desirable to have a low value of n as the memory and time requirements are proportional to n. However, compared to the training time is the inference time not that long. Having a larger pool of samples may be a stronger incentive to lower n. Tuning this parameter is beyond the scope of this thesis and no significant impact is noticed for $n > 100$, n is therefore set to 100.

Sample Selection - Hard/Easy

All samples are ranked by their Monte Carlo dropout uncertainty and in the following active learning iteration, the *hardest* samples are chosen. Four images are shown in Figure 3.9. The model considers images **a** and **b** *hard* and **c** and **d** *easy*. Note that the uncertainty tends to correlate to the faulty predictions.



(a) *Hard 1* ***(b)*** *Hard 2*

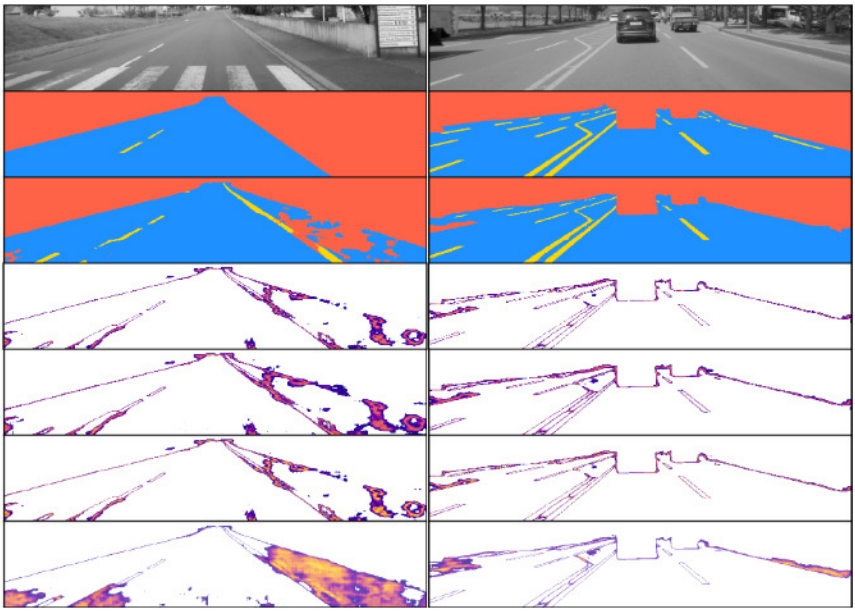


(c) *Easy 1* ***(d)*** *Easy 2*

Figure 3.9: *Four images in the training set with Monte Carlo dropout uncertainty. From the top: input image, ground truth, prediction, dropout uncertainty all classes. This network is trained on 250 samples, $n = 100$.*

3.7.2 Prediction Uncertainty

This type of uncertainty is derived by only analyzing the model's predictions. Three methods are evaluated; *least confident*, *margin sampling* and *entropy*, described in Section 2.3.1. These three methods are proven to work in a similar fashion, two example images are shown in Figure 3.10.



(a) *Image 1* ***(b)*** *Image 2*

Figure 3.10: *Two images from the test set with prediction uncertainty. From the top: input image, ground truth, prediction, uncertainty; least confident, margin sampling, entropy, dropout. Note the similarities between the prediction uncertainty images. This network is trained on 250 samples, $n = 100$ for the dropout uncertainty.*

Extracting uncertainty based on the predictions is very fast and requires little memory compared to Monte Carlo dropout. For 10 000 samples, it takes around 15 minutes for each method and takes up around 25*GB* of storage. These three methods all select similar samples, and the overlap for the 250 and 1 000 top samples are shown in Table 3.7.

The prediction uncertainty methods have an overlap of at least 96%. Because of time requirements in the thesis, only entropy is selected for evaluation by training. However, the high overlap indicates a similar outcome. Between Monte Carlo dropout and the other methods, there is a relatively low overlap, meaning they recognize different types of uncertainty.

Method 1	Method 2	250 samples	1000 samples
Least Confident	Margin Sampling	98.8	99.4
Least Confident	Entropy	96.4	97.9
Margin Sampling	Entropy	96.8	97.8
MC Dropout	Least Confident	42.8	55.3
MC Dropout	Margin Sampling	42.8	55.1
MC Dropout	Entropy	44.0	56.3

Table 3.7: *Percent of overlapping samples between different methods for uncertainty estimation. The top three are based on the model's predictions and the bottom three compared against dropout with* $n = 100$*. The top 250 and top 1 000 samples are compared.*

3.8 Selection Criteria

This section motivates and summarizes the networks that will be trained and be part of the results of the thesis.

3.8.1 Alternative Dropout Methods

Results of dropout can be found in Section 4.4. Even if Monte Carlo dropout performed slightly better than baseline, the performance was not as good as anticipated. An analysis of *why* dropout did not perform all too well is found in the analysis chapter in Section 5.2. Therefore, three other selection methods are implemented and networks trained with them. Listed in Table 3.8.

Name	Explanation
Dropout distribution	Active learning with Monte Carlo dropout that follows the original dataset distribution (Section 3.2.1). When drawing samples from the ranked uncertainty list, they are drawn to follow the subset distribution. The number of samples in each subset category are calculated and when that quota is filled, remaining samples are ignored.
Dropout 50/50	Half active learning Monte Carlo dropout, half random. When drawing new samples, 50% is drawn as usual, from the ranked uncertainty list. The remaining 50% are drawn randomly. The idea behind this method is to still gain the benefits from using *hard* samples and at the same time cover up the width of all possible samples by drawing some of them randomly.
Dropout 50/50 + distribution	A combination of the previous two. As Dropout 50/50 but with the addition that the active learning samples are drawn following the subset distribution (Section 3.2.1).

Table 3.8: *Variations on how to draw samples with active learning dropout.*

3.8.2 Network Variations

At the end of the active learning iterations, new samples are added with one of the following selection criteria:

- **Random** - baseline.
- **Monte Carlo Dropout** - uncertainty from dropout, defined in Section 2.3.2.
- **Entropy** - entropy uncertainty from the model's predictions, defined in Section 2.3.1.
- **Dropout distribution** - dropout following the subset distribution, defined in Table 3.8.
- **Dropout 50/50** - half drawn with dropout, half randomly, defined in Table 3.8.
- **Dropout 50/50 + distribution** - half with dropout following the distribution, half randomly, defined in Table 3.8.

4

Experiments and Results

This chapter describes the networks to be trained and the results in the thesis, the chapter is structured as follows:

1. **Experiments** - describes the networks to be trained.
2. **Results Description** - explain how the results are presented.
3. **Baseline** - baseline results.
4. **Monte Carlo Dropout** - Monte Carlo dropout results compared against baseline.
5. **Entropy** - entropy results compared against baseline.
6. **Dropout 50/50** - dropout 50/50 results compared against baseline.
7. **Dropout Distribution** - dropout distribution results compared against baseline.
8. **Dropout 50/50 + Distribution** - dropout 50/50 + distribution results compared against baseline.
9. **All Results** - results for all networks compared against each other.
10. **Subset Results** - Monte Carlo dropout and baseline calculated on the 8 subsets.
11. **Network Improvements** - images that show the networks' improvement over time.

4.1 Experiments

Prior to these experiments, motivations for the networks, decisions on model parameters and the dataset are explained in Chapter 3. The model is trained for 13 *active learning iterations*, (ALI). Sizes of the training and validation sets are shown in Table 4.1. The test set is always fixed at 7 361 samples.

ALI	# Training	# Validation
1	250	65
2	500	130
3	750	195
4	1 000	260
5	1 500	390
6	2 000	520
7	2 500	650
8	3 500	910
9	4 500	1170
10	5 500	1430
11	7 000	1820
12	8 500	2210
13	10 000	2600

Table 4.1: *Sizes of the dataset during the training of the CNN. The validation sizes are approximately 26% of their respective training size. The test set is fixed for all active learning iterations at 7 361 samples.*

In every new active learning iteration, the new samples are chosen from the remaining pool and added to the current training and validation sets. Except for baseline, at the end of every active learning iteration uncertainties are estimated for the remaining samples in the training and validation sets. For active learning iteration 2 and forward, the weights are loaded from the previous iteration.

ENet [38] is chosen as the CNN model. At the beginning of every active learning iteration, a new model is created, with the parameters stated in Table 4.2. Settings and network parameters not mentioned remains unchanged from the original paper.

The networks that will be trained are shown in Table 4.3. For each sample selection method, three trainings are performed using different initial datasets. All sample selection methods start out with the same initial data set.

Name	Value	Description
Batch size	6	Batch size.
Epochs	40	Number of pseudo-epochs. Explained in Section 3.6.1.
SPE - Samples Per Epoch	37 500	Number of samples per epoch. The current training samples are drawn with replacement until *SPE* samples are used in the training. Explained in Section 3.6.1.
Learning rate	0.0005	Learning rate.
Learning rate decay; *patience*, *decay rate*	6, $\frac{1}{2}$	If no validation accuracy improvement on the validation set for *patience* epochs, the learning rate is multiplied by the *decay rate*. Explained in Section 3.6.2.
Early stopping; *patience*, δ_{min}	8, 0.0002	I no validation accuracy improvement larger than δ_{min} for *patience* epochs, the active learning iteration stops. Explained in Section 3.6.4.

Table 4.2: *Model parameters. All parameters are reset at the start of every active learning iteration.*

Number	Sample selection criteria	Initial dataset
1	Baseline - Random	1, 2, 3
2	Monte Carlo Dropout	1, 2, 3
3	Entropy	1, 2, 3
4	Dropout distribution	1, 2, 3
5	Dropout 50/50	1, 2, 3
6	Dropout 50/50 + distribution	1, 2, 3

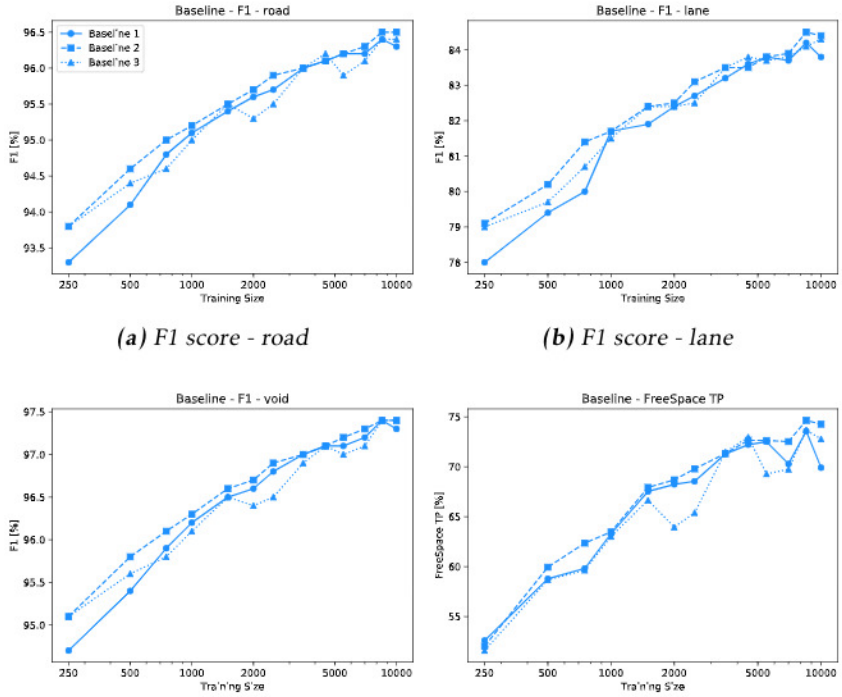
Table 4.3: *The trainings that will be performed in the thesis. The selection criteria are defined in Section 3.8.2.*

4.2 Results Description

For each network type figures are shown for FreeSpace TP and the F1 scores: road, lane, and void. Following the figures is a table presented for each method where the performance is averaged over the dataset sizes. For each column is the difference against baseline computed and shown in the following column. The last column shows the mean over all metrics.

4.3 Baseline Results

The performances for the baseline networks are shown in Figure 4.1 and Table 4.4.



(a) F1 score - road *(b) F1 score - lane*

(c) F1 score - void *(d) FreeSpace TP*

Figure 4.1: *Results of the baseline networks. Number of training samples on the x-axis, performance in percent on the y-axis. Logarithmic x-axis.*

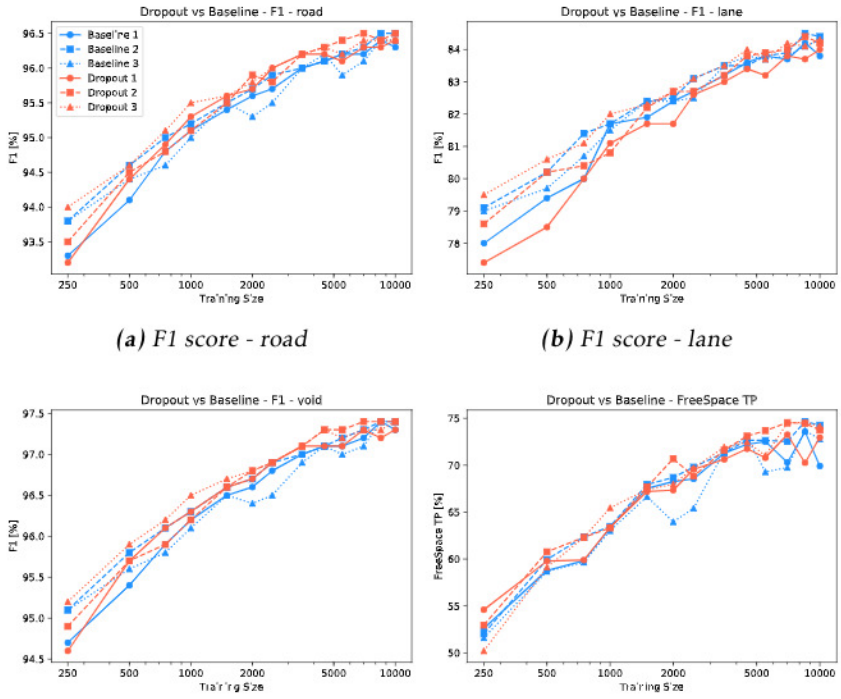
Name	road	ΔBL	lane	ΔBL	void	ΔBL	FS	ΔBL	Mean
Baseline	95.53		82.41		96.59		66.92		85.36

Table 4.4: *Averaged result over all data sizes for baseline. Results given in percent.*

One thing to note is that the third baseline training performed quite bad for the dataset sizes 2000 and 2500 (active learning iteration 6 and 7). This is most apparent for FreeSpace TP but also for F1 - road and F1 - void.

4.4 Monte Carlo Dropout Results

The Monte Carlo dropout results are shown in Figure 4.2 and Table 4.5 where they are compared against baseline.



(a) F1 score - road *(b) F1 score - lane*

(c) F1 score - void *(d) FreeSpace TP*

Figure 4.2: *Results of the Monte Carlo dropout networks compared to baseline. Number of training samples on the x-axis, performance in percent on the y-axis. Logarithmic x-axis.*

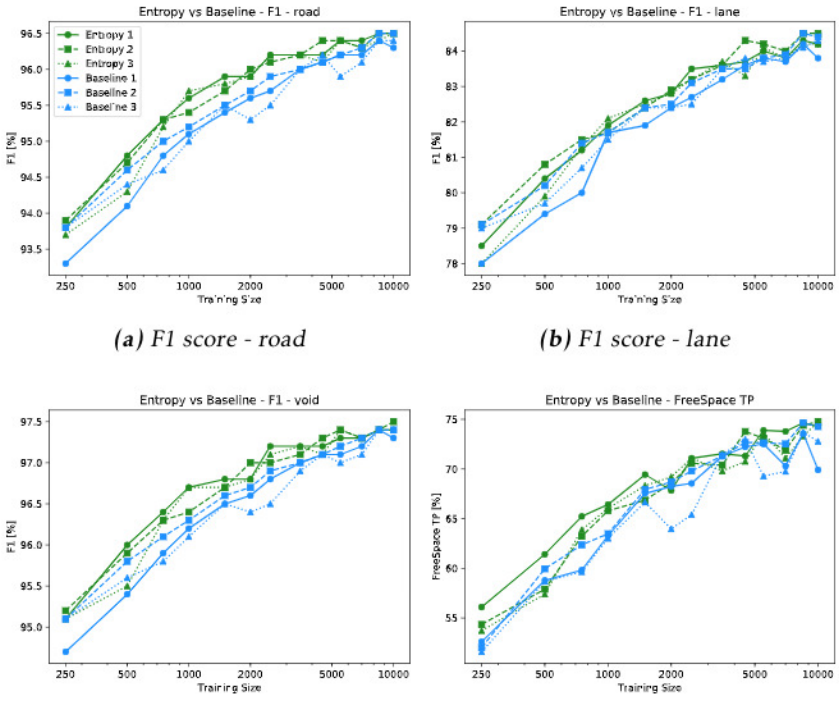
Name	road	ΔBL	lane	ΔBL	void	ΔBL	FS	ΔBL	Mean
Baseline	95.53		**82.41**		96.59		66.92		85.36
MC Dropout	**95.65**	0.12	82.32	-0.09	**96.68**	0.09	**67.69**	0.77	**85.58**

Table 4.5: *Averaged result over all data sizes for Monte Carlo dropout and baseline. Best performing for each metric is marked bold. Results given in percent.*

Monte Carlo dropout performs slightly better compared to baseline. Better performance for F1 - road (+0.12%), F1 - void (+0.09%) and FreeSpace TP (+0.77%) is noted. The F1 - lane performance is lowered (-0.09%).

4.5 Entropy Results

The entropy results are shown in Figure 4.3 and Table 4.6 where they are compared against baseline.



(a) F1 score - road *(b) F1 score - lane*

(c) F1 score - void *(d) FreeSpace TP*

Figure 4.3: *Results of the entropy networks compared to baseline. Number of training samples on the x-axis, performance in percent on the y-axis. Logarithmic x-axis.*

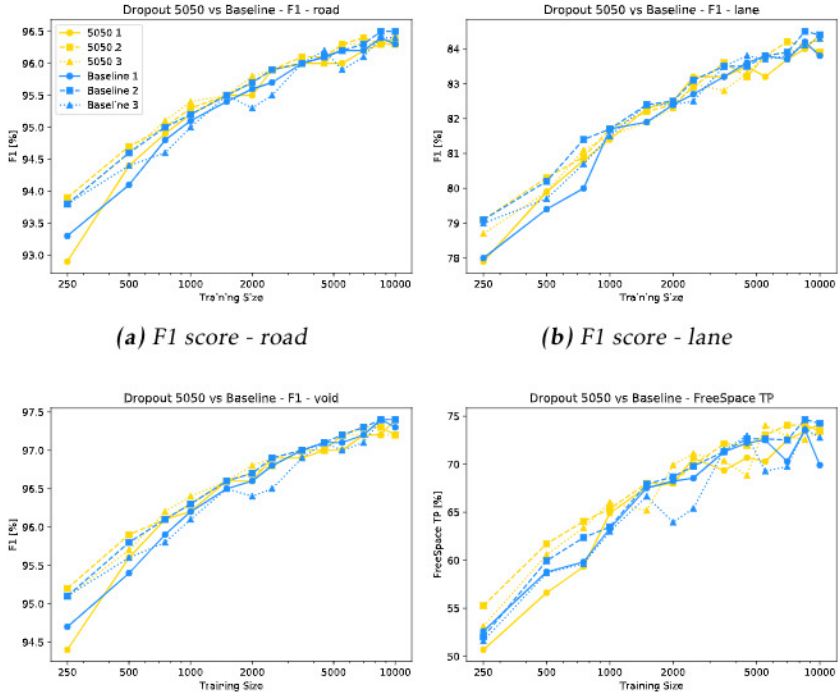
Name	road	ΔBL	lane	ΔBL	void	ΔBL	FS	ΔBL	Mean
Baseline	95.53		82.41		96.59		66.92		85.36
Entropy	**95.79**	0.26	**82.67**	0.26	**96.80**	0.21	**68.34**	1.42	**85.90**

Table 4.6: *Averaged result over all data sizes for entropy and baseline. Best performing for each metric is marked bold. Results given in percent.*

Entropy performs better than baseline for all metrics. Improvement for each metric: F1 - road (+0.26%), F1 - lane (+0.26%), F1 - void (+0.21%) and FreeSpace TP (+1.42%).

4.6 Dropout 50/50 Results

The dropout 50/50 results are shown in Figure 4.4 and Table 4.7 where they are compared against baseline.



(a) F1 score - road *(b) F1 score - lane*

(c) F1 score - void *(d) FreeSpace TP*

Figure 4.4: *Results of the dropout 50/50 networks compared to baseline. Number of training samples on the x-axis, performance in percent on the y-axis. Logarithmic x-axis.*

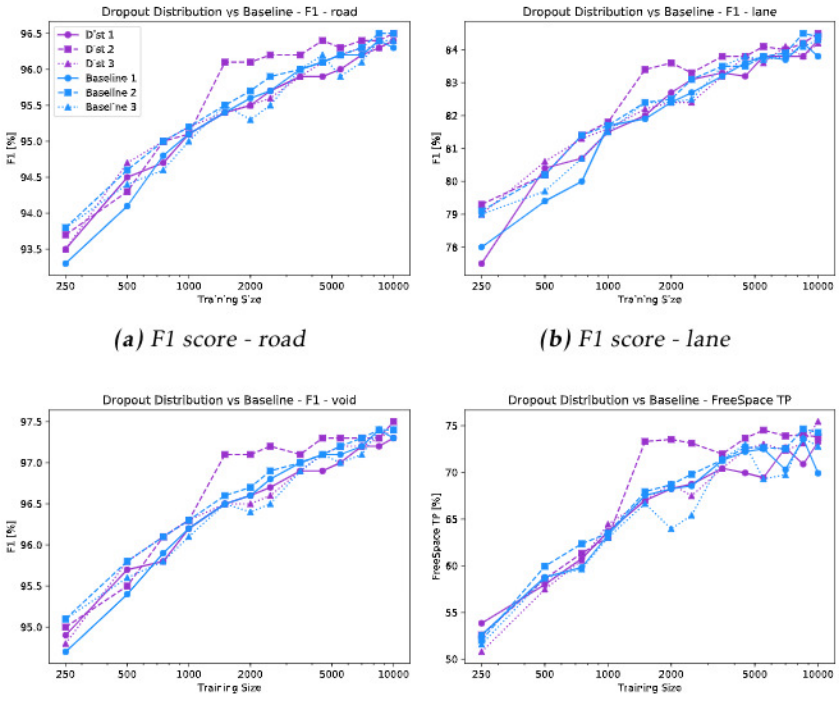
Name	road	ΔBL	lane	ΔBL	void	ΔBL	FS	ΔBL	Mean
Baseline	95.53		**82.41**		96.59		66.92		85.36
50/50	**95.59**	0.06	82.38	-0.03	**96.62**	0.03	**67.72**	0.80	**85.58**

Table 4.7: *Averaged result over all data sizes for dropout 50/50 and baseline. Best performing for each metric is marked bold. Results given in percent.*

Dropout 50/50 performs quite similar to baseline. An increase is noted for F1 - road (+0.06%), F1 - void (+0.03%) and FreeSpace TP (+0.80%) and a decrease for F1 - lane (-0.03%).

4.7 Dropout Distribution Results

The dropout distribution results are shown in Figure 4.5 and Table 4.8 where they are compared against baseline.



(a) F1 score - road *(b) F1 score - lane*

(c) F1 score - void *(d) FreeSpace TP*

Figure 4.5: *Results of the dropout distribution networks compared to baseline. Number of training samples on the x-axis, performance in percent on the y-axis. Logarithmic x-axis.*

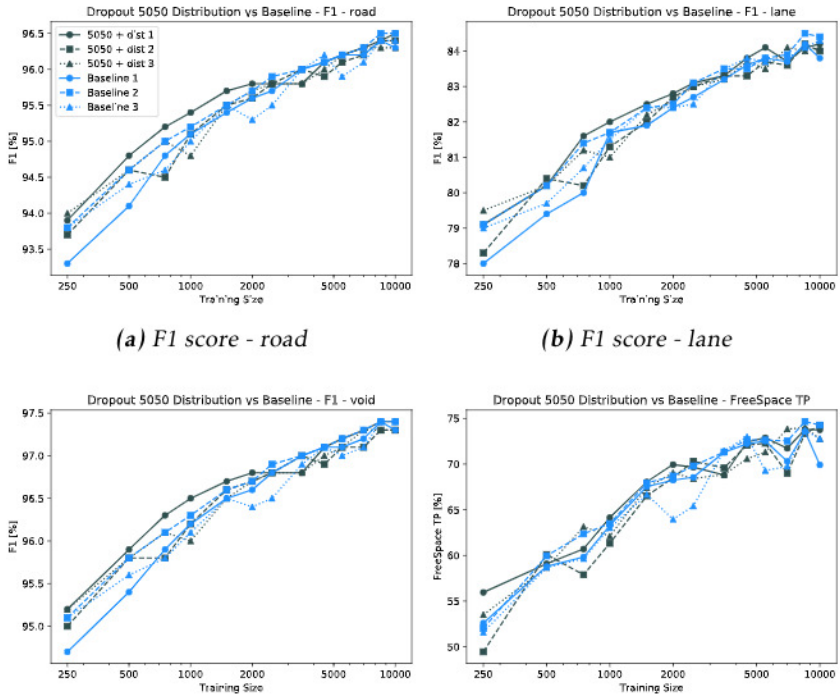
Name	road	ΔBL	lane	ΔBL	void	ΔBL	FS	ΔBL	Mean
Baseline	95.53		82.41		96.59		66.92		85.36
Distribution	**95.59**	0.06	**82.56**	0.15	**96.63**	0.04	**67.63**	0.71	**85.60**

Table 4.8: *Averaged result over all data sizes for dropout distribution and baseline. Best performing for each metric is marked bold. Results given in percent.*

The method performs better than baseline for all metrics: F1 - road (+0.06%), F1 - lane (+0.15%), F1 - void (+0.04%) and FreeSpace TP (+0.71%).

4.8 Dropout 50/50 + Distribution Results

The Monte Carlo dropout 50/50 and distribution results are shown in Figure 4.6 and Table 4.9 where they are compared against baseline.



(a) F1 score - road *(b) F1 score - lane*

(c) F1 score - void *(d) FreeSpace TP*

Figure 4.6: *Results of the dropout 50/50 + distribution networks compared to baseline. Number of training samples on the x-axis, performance in percent on the y-axis. Logarithmic x-axis.*

Name	road	ΔBL	lane	ΔBL	void	ΔBL	FS	ΔBL	Mean
Baseline	95.53		82.41		96.59		66.92		85.36
50/50 + dist	**95.59**	0.06	**82.48**	0.07	**96.64**	0.05	**67.16**	0.24	**85.47**

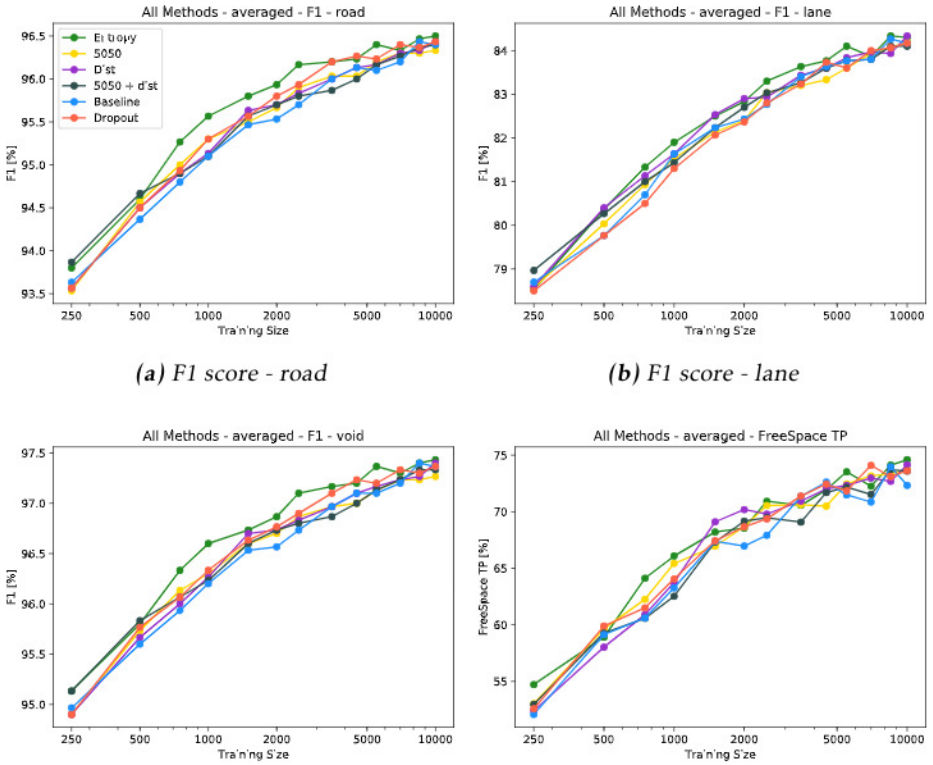
Table 4.9: *Averaged result over all data sizes for dropout 50/50 + distribution and baseline. Best performing for each metric is marked bold. Results given in percent.*

Dropout 50/50 + distribution achieves higher performance, compared to baseline, for all metrics: F1 - road (+0.06%), F1 - lane (+0.07%), F1 - void (+0.05%) and FreeSpace TP (+0.24%).

4.9 All Results

The results for all methods compared together are shown in Figure 4.7. Each method is trained three times, which are averaged in the figure.

The performance averaged over the data set sizes are shown in Table 4.10. In Table 4.11 is the performance for each active learning iteration given for baseline, Monte Carlo dropout, and entropy. Where marked bold, the method achieves higher performance than baseline the following active learning iteration.



(a) F1 score - road

(b) F1 score - lane

(c) F1 score - void

(d) FreeSpace TP

Figure 4.7: *Results of all methods compared in the same figure, averaged. Number of training samples on the x-axis, performance in percent on the y-axis. Logarithmic x-axis.*

Name	road	ΔBL	lane	ΔBL	void	ΔBL	FS	ΔBL	Mean
Baseline	95.53		82.41		96.59		66.92		85.36
MC Dropout	95.65	0.12	82.32	-0.09	96.68	0.09	67.69	0.77	85.58
Entropy	**95.79**	0.26	**82.67**	0.26	**96.80**	0.21	**68.34**	1.42	**85.90**
50/50	95.59	0.06	82.38	-0.03	96.62	0.03	67.72	0.80	85.58
Distribution	95.59	0.06	82.56	0.15	96.63	0.04	67.63	0.71	85.60
50/50 + dist	95.59	0.06	82.48	0.07	96.64	0.05	67.16	0.24	85.47

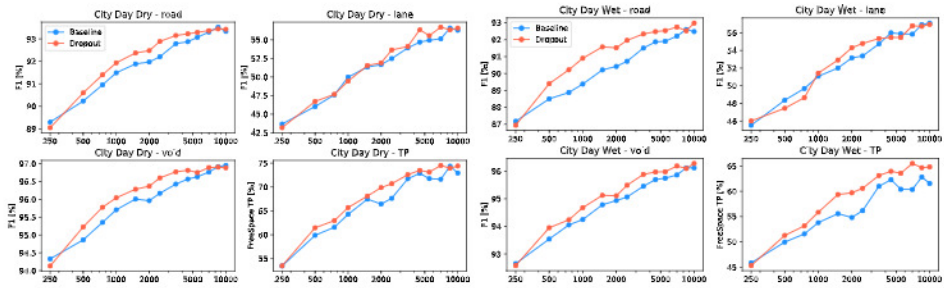
Table 4.10: *Averaged result over all data sizes for all methods. Best performing for each metric is marked bold. Results are given in percent along with the difference against baseline.*

Size	BL_road	MC_road	EN_road	BL_lane	MC_lane	EN_lane
250	93.63	93.57	93.80	78.70	78.50	78.53
500	94.37	94.50	94.60	79.77	79.77	80.37
750	94.80	94.93	**95.27**	80.70	80.50	81.33
1000	95.10	95.30	**95.57**	81.63	81.30	81.90
1500	95.47	**95.57**	**95.80**	82.23	82.07	**82.50**
2000	95.53	**95.80**	**95.93**	82.43	82.37	**82.83**
2500	95.70	95.93	**96.17**	82.77	82.80	83.30
3500	96.00	**96.20**	**96.20**	83.40	83.23	**83.63**
4500	96.13	**96.27**	**96.23**	83.63	83.73	**83.77**
5500	96.10	**96.23**	**96.40**	83.77	83.60	**84.10**
7000	96.20	96.40	96.33	83.80	84.00	83.87
8500	96.43	96.37	**96.47**	84.27	84.07	**84.33**
10000	96.40	96.43	96.50	84.17	84.17	84.30
Size	BL_void	MC_void	EN_void	BL_FS	MC_FS	EN_FS
250	94.97	94.90	95.13	52.07	52.58	54.72
500	95.60	95.77	95.80	59.14	59.89	58.90
750	95.93	96.07	**96.33**	60.61	61.50	**64.11**
1000	96.20	96.33	**96.60**	63.26	64.06	66.08
1500	96.53	**96.63**	**96.73**	67.38	**67.43**	**68.20**
2000	96.57	**96.77**	**96.87**	66.97	**68.67**	**68.54**
2500	96.73	96.90	**97.10**	67.92	69.36	70.92
3500	96.97	**97.10**	**97.17**	71.36	71.38	70.56
4500	97.10	**97.23**	**97.20**	72.62	**72.44**	**71.95**
5500	97.10	**97.20**	**97.37**	71.48	**71.84**	**73.52**
7000	97.20	97.33	97.30	70.86	**74.09**	72.26
8500	97.40	97.30	**97.40**	73.97	**73.10**	**74.11**
10000	97.37	97.37	97.43	72.32	73.61	74.59

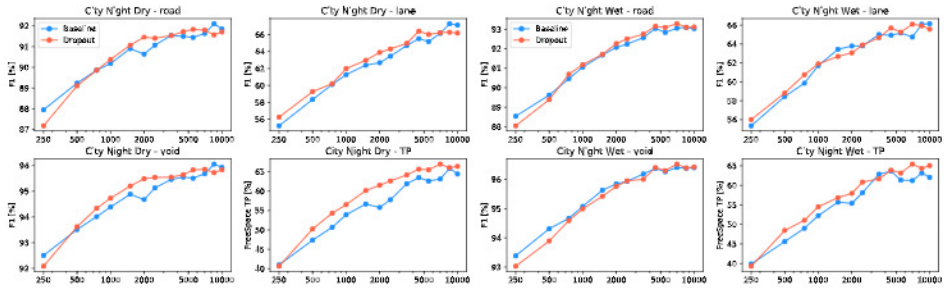
Table 4.11: *Result compared for baseline (BL), Monte Carlo dropout (MC) and Entropy (EN) for all dataset sizes. The data is averaged over the three independent trainings, given in percent. When marked bold, the performance is higher than the next active learning iteration for baseline.*

4.10 Subset Results

Results on the 8 subsets (Section 3.2.1) are presented for baseline and Monte Carlo dropout in Figure 4.8 and Figure 4.9. Each method is trained three times and the results presented are the averages of those three. Each subset is presented with the F1 score for the classes: road, lane, and, void and the FreeSpace TP measure.

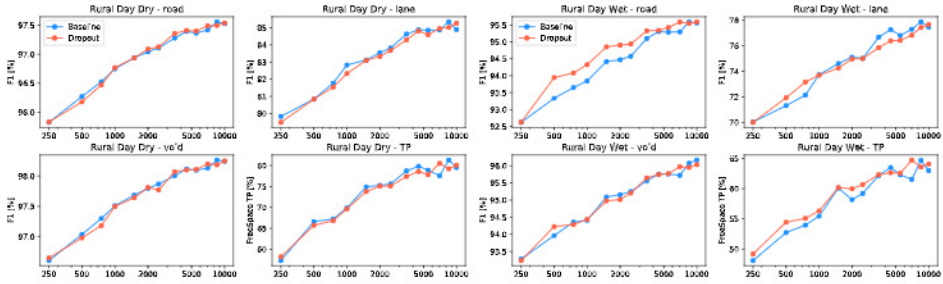


(a) City Day Dry - 14.9% *(b) City Day Wet - 9.6%*

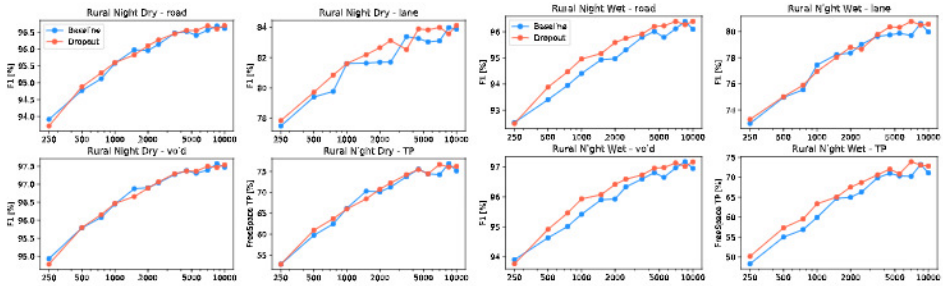


(c) City Night Dry - 4.4% *(d) City Night Wet - 2.9%*

Figure 4.8: *Averaged results for each method on the 8 subsets for Monte Carlo dropout and baseline. Size of each subset is given in percent, from Table 3.3.*



(a) Rural Day Dry - 36.8% *(b)* Rural Day Wet - 14.2%

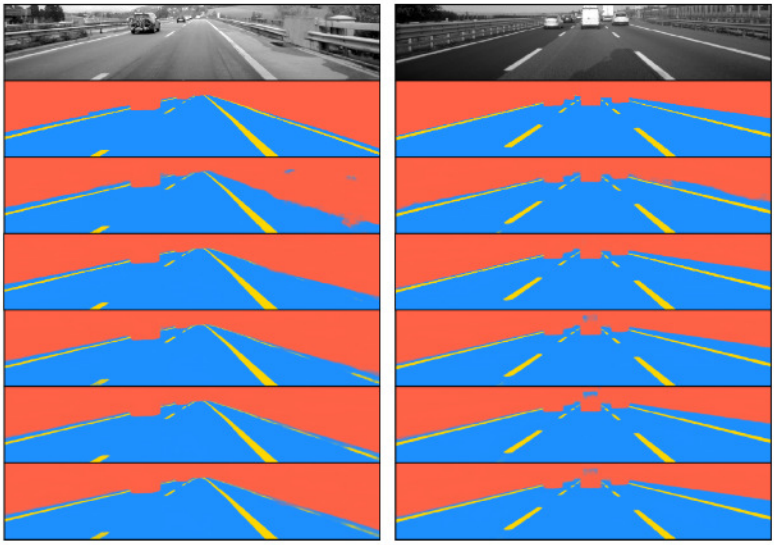


(c) Rural Night Dry - 11.8% *(d)* Rural Night Wet - 5.4%

Figure 4.9: *Averaged results for each method on the 8 subsets for Monte Carlo dropout and baseline. Size of each subset is given in percent, from Table 3.3.*

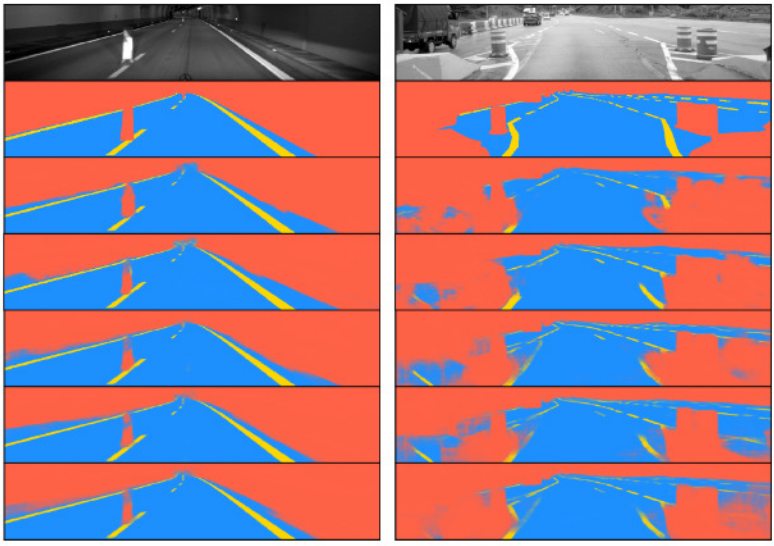
4.11 Network Improvements

Some scenarios are quite easy for the network to learn and the predictions are quickly reasonable good. Other test images are hard and it takes a long time for the network to perform well on them (if even that). In Figure 4.10, 4.11, 4.12, and 4.13 are a couple of images shown for how the network's predictions develops over time. All images are from the baseline network and from active learning iterations: 1, 4, 7, 10 and 13.



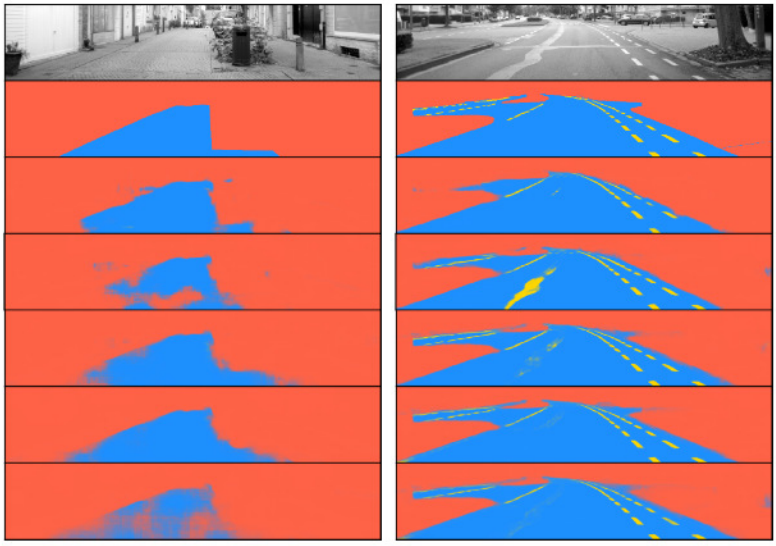
(a) Image 1 *(b)* Image 2

Figure 4.10: *Baseline predictions for a couple of active learning iterations. From the top: input image, ground truth, predictions for ALI; 1, 4, 7, 10, 13.*



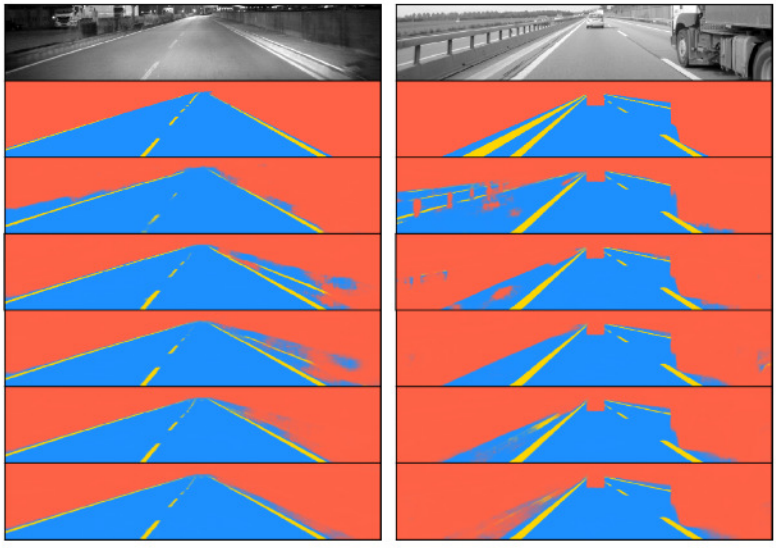
(a) Image 1 *(b)* Image 2

Figure 4.11: *Baseline predictions for a couple of active learning iterations. From the top: input image, ground truth, predictions for ALI; 1, 4, 7, 10, 13.*



(a) Image 3 *(b)* Image 4

Figure 4.12: *Baseline predictions for a couple of active learning iterations. From the top: input image, ground truth, predictions for ALI; 1, 4, 7, 10, 13.*



(a) Image 3 *(b)* Image 4

Figure 4.13: *Baseline predictions for a couple of active learning iterations. From the top: input image, ground truth, predictions for ALI; 1, 4, 7, 10, 13.*

5 Analysis

The analysis chapter consists of three parts. First is the networks' performance analyzed, followed by an analysis of the subset distribution. Finally is a t-SNE analysis performed on the samples selected by baseline and Monte Carlo dropout.

5.1 Network Performances

The compared results presented in Chapter 4 are in some cases relatively small. It is worth noting that the difference could come from better performance or just variance in the trainings.

- **Baseline**
  The results are presented in Section 4.3.
- **Monte Carlo dropout**
  Monte Carlo dropout, presented in Section 4.4, shows slightly better performance than baseline.
- **Entropy**
  The entropy method, Section 4.5, shows that for almost all trained networks, entropy is better than baseline.
- **Dropout 50/50**
  The results are presented in Section 4.6. Compared to Monte Carlo dropout is this method quite similar in performance.
- **Dropout distribution**
  The results are presented in Section 4.7. Dropout distribution is similar in performance as Monte Carlo dropout except for an improvement for the F1

- lane metric.

- **Dropout 50/50 + distribution**
  The results are presented in Section 4.8. Compared to Monte Carlo dropout is this method similar in performance. Dropout distribution performs better than dropout 50/50 + distribution in all metrics except a 0.01% decline for the F1 - void metric.

When comparing the mean over all metrics in Table 4.10, all methods perform better than baseline. Entropy is by far the best method with the highest score in every metric. Entropy, dropout distribution, and dropout 50/50 + distribution get a higher score for every metric. Monte Carlo dropout and dropout 50/50 get a higher score for every metric except F1 - lane.

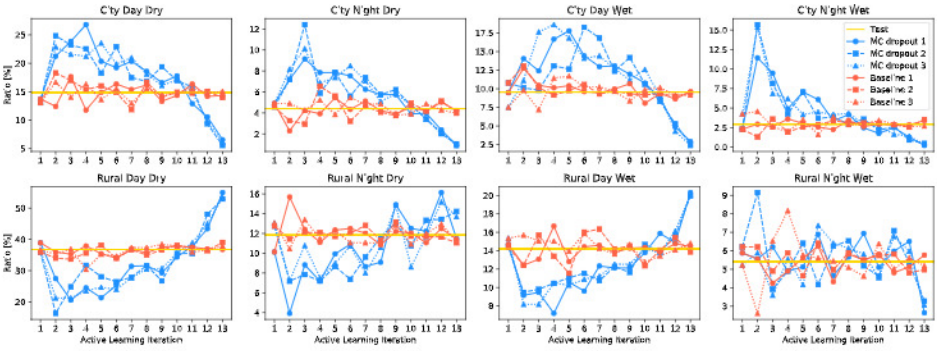
In Table 4.11, a comparison is made between baseline, Monte Carlo dropout, and entropy. The bold markings indicate that with that particular dataset size, the method gets at least the same performance as baseline the **following** active learning iteration. Monte Carlo dropout does not achieve a better score, compared to baseline, for the F1 - lane metric. For Monte Carlo dropout and the metrics F1 - road, F1 - void and FreeSpace, the score is at least higher than the next baseline iteration for the dataset sizes 1500, 2000, 4500, and 5500. For entropy that is the case for all metrics for the dataset sizes 1500, 2000, 4500, 5500, 8500.

This means that, for entropy, a smaller dataset is required to achieve at least the same performance as baseline; effectively lowering the dataset requirement with 500, 500, 1000, 1500, 1500 samples respectively. This indicates that entropy roughly reduces the data need by 15-25% compared to random selection. The same conclusion can not be drawn for Monte Carlo dropout as the F1 - lane metric to not perform better than baseline.

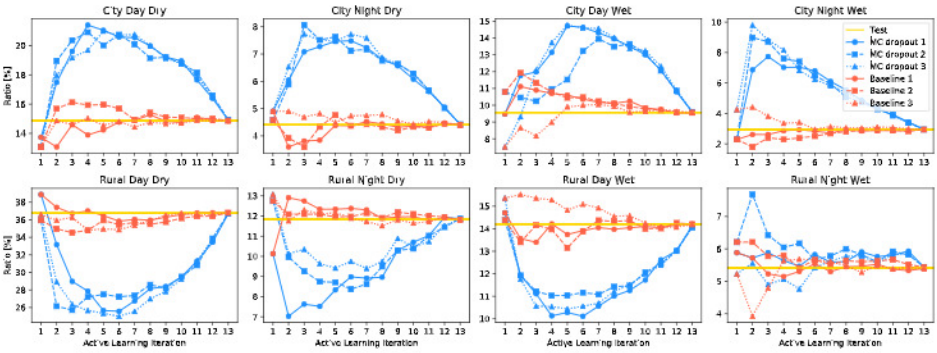
5.2 Dropout Subset Distribution

In some aspects, Monte Carlo dropout seems to select special types of samples that are not general enough to be part of the test set. This could have the effect that the algorithm benefits more by selecting samples randomly as the sample feature space in the dataset is covered better. One indication of this is noticed by looking at *how* the new samples are selected according to the dataset distribution, explained in Section 3.2.1.

The new samples chosen every active learning iteration are analyzed with respect to the dataset distribution, that is explained in Table 3.3. Both the training, validation and test sets follow this distribution. The ratio for the test set is static (since the same test set is used for evaluation in every active learning iteration). Baseline is compared to Monte Carlo dropout in Figure 5.1 where **a** show the distribution for the additional samples in the current active learning iteration and **b** the cumulative distribution. Number of samples in each active learning iteration is defined in Table 3.5.



(a) *Current active learning iteration*



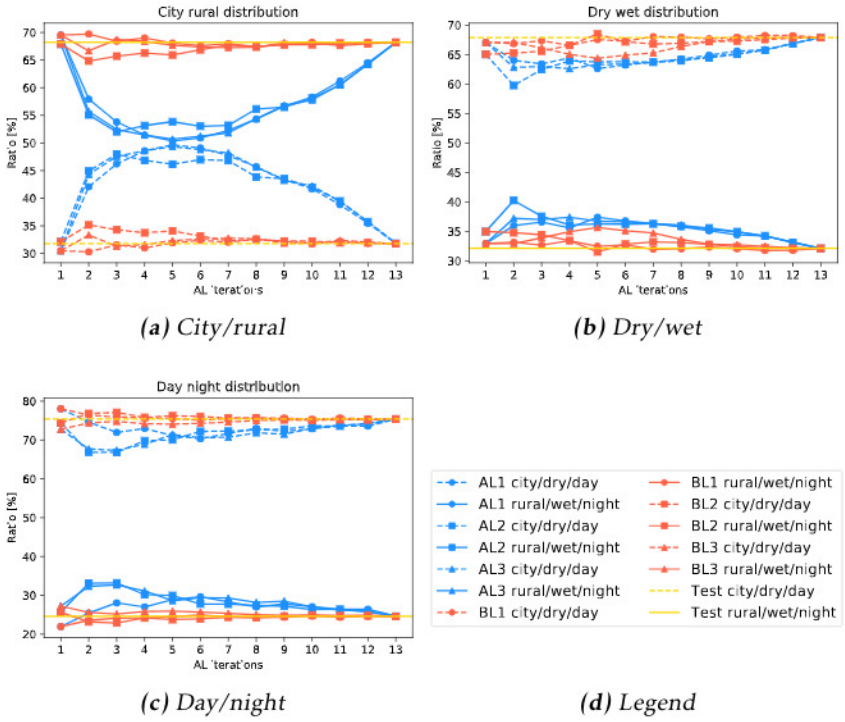
(b) *Cumulative samples selected*

Figure 5.1: *Distribution of the new samples every active learning iteration for active learning with Monte Carlo dropout and Baseline. Active learning iterations on the x-axis and percent on the y-axis. The top row is city samples and the bottom row rural samples.*

From Figure 5.1 **b**, it is clear that Monte Carlo dropout favors images captured in cities. Already from active learning iteration 2 and forward is a clear trend present. There exist more objects in those scenes. Intersections and varying road types are more likely to be encountered. In contrast, the rural scenes are often captured on highways and are more similar to each other. The vehicles tend to be seen from the rear and many roads are fairly straight.

In Section 4.10 the average performance of baseline and Monte Carlo dropout is calculated on the 8 subsets (explained in Section 3.2.1). In Figure 4.8 the city samples is present and Figure 4.9 the rural samples. In general, Monte Carlo dropout performs better for the city samples than baseline, especially noted in Figure 4.8 **a**, **b**. It is worth noting that the subset *Rural Day Dry*, Figure 4.9 **c**, is the most common subset with over a third (36.8%) of all samples in it. For that specific subset is no apparent performance difference noted.

Network performance is calculated on the test set. One possible explanation for the marginal improvement in Monte Carlo dropout could come from that too many samples in the training and validation sets are unlike the ones in the test set. Figure 5.2 shows a comparison of the selected samples in the training set for city/rural, dry/wet and day/night. For all three categories, the intuitively *harder* counterpart (city, wet, night) are more common in Monte Carlo dropout. This effect is most noticeable for the city samples. The majority of the samples are captured in rural environments (68.2% vs 31.8%). The test set performance could gain more by focusing more on the rural scenes.



(a) City/rural *(b) Dry/wet*

(c) Day/night *(d) Legend*

Figure 5.2: *Samples in baseline and Monte Carlo dropout divided into the 3 subset categories. Values are given in percent and taken from the cumulative images. Active learning iterations on the x-axis and percent on the y-axis. Common legend in* ***d****.*

Because of this, a couple of other networks were trained, defined in Table 3.8. They were designed to try to improve the performance by mitigating the possible error source from the skewness in the subset distribution. These results are presented in Section 4.6, Section 4.7, and Section 4.8. Their effect is analyzed in Section 5.1.

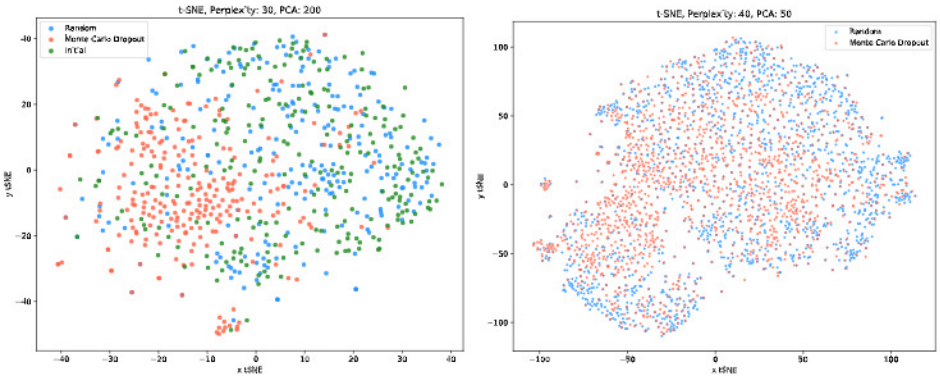
5.3 t-SNE Analysis

To analyze how samples are selected by Monte Carlo dropout a t-SNE [36] analysis is performed. t-SNE is a technique for visualizing high dimensional data by placing *similar* objects close to each other and *dissimilar* farther away. Sener and Savarese [40] perform a t-SNE analysis to motivate why their method is superior to uncertainty based methods.

The samples are passed through the network and the output at the last encoder layer is saved (layer 23 in Appendix A). The layer's output size is: $128x26x128 = 425\,984$ dimensions. After reducing the dimensionality with Principal Component Analysis (PCA), the data is fed to the t-SNE algorithm which runs for 5000 iterations.

Firstly, samples that are selected by baseline and Monte Carlo dropout the second active learning iteration are analyzed in Figure 5.3 **a**. The high dimensional representation of the samples are reduced to 200 PCA components, the t-SNE perplexity is set to 30. In the initial active learning iteration, both models share the 250 first samples. The models choose samples quite differently, the Monte Carlo dropout samples are clustered together more than the initialized and baseline samples.

Secondly, the baseline and Monte Carlo dropout samples are compared for active learning iteration 5 in Figure 5.3 **b**. The high dimensional representation of the samples are reduced to 50 PCA components, the t-SNE perplexity is set to 40. That iteration is chosen as that is where the highest difference is noted in Figure 5.2 **a**. Out of 1500 samples, they have an overlap of 401 samples (where 250 are initial and the remaining 151 are chosen by both). The difference is not as noticeable as in Figure 5.3 **a**, but the Monte Carlo dropout samples cluster together more in the center and left parts of the figure.



(a) ALI 2 *(b) ALI 5*

Figure 5.3: *t-SNE analysis of the samples chosen by baseline and Monte Carlo dropout. blue: random, red: Monte Carlo dropout, green: shared initial.*
a*: active learning iteration 2. Initial 250 shared between baseline and Monte Carlo dropout and an additional 250 by each method.*
b*: active learning iteration 5. An analysis made of the 1500 first samples. They have an overlap of 401 samples.*

6

Discussion

The discussion is divided into six parts:

1. **Results Discussion** - The results of the thesis are discussed.
2. **Informative Samples** - What an informative sample is, what is a hard sample?
3. **Uncertainty Aggregation** - How are the pixels' uncertainties transformed to sample uncertainty?
4. **Is Active Learning Worth it?** - Discussion of why active learning is a good technique even with a small performance gain.
5. **Uncertainty Applications** - Possible real-world applications.
6. **Thesis Reflection** - My own thoughts are given on the thesis.

6.1 Results Discussion

Monte Carlo dropout, as implemented in this thesis, does work slightly better for all metrics except F1 - lane, Section 4.4, Section 5.1. Even if the lane class is small (2.2%, Table 3.2), lane is maybe the most important class as the lane markings are critical for autonomous driving. For some metrics in Monte Carlo dropout does the method require less amount of data than baseline (Table 4.11), however, this is not the case for all metrics.

Entropy works well for uncertainty selection as the method gets, by a large degree, higher performance than all other evaluated methods. As stated in Section 5.1 entropy succeeds with lowering the dataset size required for 5 active learning iterations compared to baseline, averaging a lower dataset size of 19.9%. The average increase in performance is 0.54% and as seen in Section 4.5, the metrics show consistently higher performance. This demonstrates a significant improvement by using entropy for informative sample selection.

The three additional methods perform quite similar to Monte Carlo dropout. Even if dropout distribution has slightly better mean value than Monte Carlo dropout (85.60 compared to 85.58, Table 4.10), that increase does not make the extra effort worthwhile.

Even if the effect of uncertainty selection is better than random selection, the method in general and the active learning framework could improve more. Two possible improvements are:

1. **Smaller batches:** As stated in Section 2.4.4 and Section 2.4.5, small active learning batch sizes are desired in order for active learning to work. Limiting the batch size would most likely improve the model's performance. The model incorporates new information more often and can give a different uncertainty estimate the next active learning iteration.

2. **Extended informative selection:** As stated by Sener and Savarese [40] in Section 2.4.5, methods based on only uncertainty do not work well compared to random selection and the author's method. By using clustering techniques the effect with the difference in sample selection (found in the t-SNE analysis, Section 5.3) could likely be mitigated. Combining uncertainty and clustering would be interesting for further evaluation.

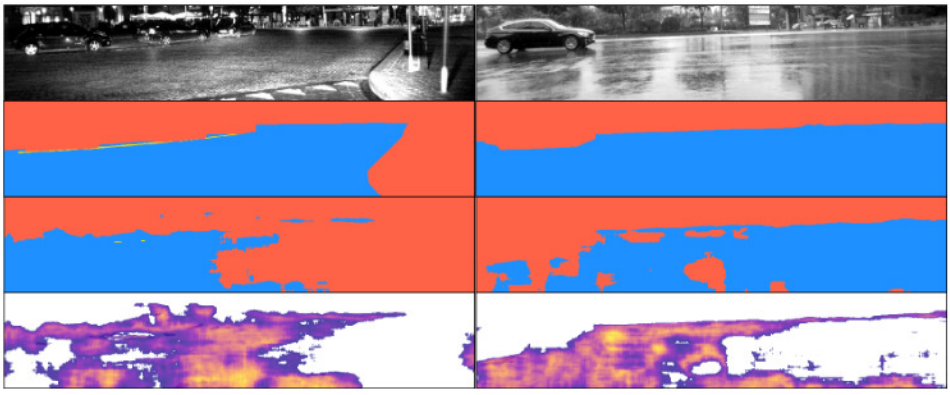
6.2 Informative Samples

To understand how the model selects samples, it is important to understand what uncertainty in this regard actually means. The model is trained with dropout to make it robust to changes. By limiting the dependency of certain units in the network's predictions, the model is supposed to have better generalization performance. When finding uncertain samples, n dropout networks predict what they think is correct. In general, Monte Carlo dropout finds a lot of borders around objects, which is expected as the exact contour of objects are hard to position.

This can lead to a problem; samples with a lot of objects can conduce to a high uncertainty by just having a lot of objects in them.

A high performing model should be able to handle all types of possible cases. Finding a pool of samples that is the foundation for a stable and broad model is desirable. For this, the model would likely need common samples, the more *uncommon* and *harder* samples should maybe be added later in the process. The model could possibly gain more by pre-training on a broad set of samples and adding, for example, city samples when the basic knowledge is distributed in the network.

Two of the top uncertain images that the model selects are shown in Figure 6.1. Both these samples are important situations but in the aspect of performing well on the test set; samples that show situations more common to the test set will likely benefit the performance more.



(a) *Hard image 1* ***(b)*** *Hard image 2*

Figure 6.1: *Two images in the training set with Monte Carlo dropout uncertainty. From the top: input image, ground truth, prediction, dropout uncertainty all classes. This network is trained on 250 samples, $n = 100$.*

The network's *knowledge* and potentially broader capabilities could improve with Monte Carlo dropout. But as the test set determines how good the network is, this *complex knowledge* do not eventuate in better results. By just focusing on easier (and more common) situations, the model could possibly benefit more in the end.

In Section 5.2, the selected samples are analyzed according to the subset distribution. In Figure 5.2 is it clear that Monte Carlo dropout favor images in city environments. Monte Carlo dropout also performs better on the city samples, Section 4.10.

In Section 5.3 is a t-SNE analysis performed which further show that the Monte Carlo dropout samples are different from the ones in baseline.

6.3 Uncertainty Aggregation

Research question **1a** in Section 1.2 state the problem with transforming pixel uncertainties to sample uncertainty. This thesis was mostly based on CEAL [46], Section 2.4.3, where active learning is used for classification. Sample uncertainty in semantic segmentation needs to aggregate the pixels' uncertainties to a comparable scalar.

Initially, the idea was to rank specific regions of the image as more *important* or in some way make a more clever aggregation of the pixels' uncertainties. Other parts of the thesis were given more focus and the average over all pixels was chosen as the way to convert the pixels' uncertainties to sample uncertainty. As stated in Section 3.4, and especially Figure 3.3, the regions in the lower middle and upper parts of the image are often static. By weighting specific parts in the image, maybe a more clever uncertainty aggregation could be found.

6.4 Is Active Learning Worth it?

The purpose of active learning is to limit the amount of annotated training samples. Is active learning worth the extra effort by training for multiple active learning iterations? Can not just a large dataset be annotated from the get-go? Assuming that this implementation was successful, how much time/resources would be saved? A GPU will be dedicated to the active learning framework. Numbers presented are rough estimates:

- An annotator can mark 10 samples/hour and are paid \$10/hour.
- The complete training set is 37 500 samples.
- Active learning only requires 80% of the samples' annotation.
- The price of a GTX 1080 Ti is \$700 [2].
- Electricity cost is assumed to be 0.

This sums up to: $37\,500 * 0.2/10 = 750$ hours of extra annotation time and a cost of \$7 500. In comparison, active learning takes up around 10 days (240 hours) of extra training time than a normal CNN training, Table 3.6.

The cost for personnel is high, a GPU is quite cheap in comparison. It is neither necessary with a private GPU, the network could train on a multi-GPU park to lower the training time even further. Relatively low performance of active learning is needed to limit the total cost of creating a *good* CNN.

A good workflow in a real-world application could be that active learning trains an additional active learning iteration overnight and select new samples for annotation. During the following day, the annotators mark the samples. The same could be done during the week + weekend. The annotation loop is repeated to build up a good, informative pool of samples and a reliable network.

6.5 Uncertainty Applications

Vast amounts of data are collected from various places around the world to cover all possible situations that the real world application could encounter. This gives a pool in the order of millions of unannotated samples to choose from. In contrast, this dataset (before the dataset reduction) contains around 54 000 annotated samples, a mere fraction of all available samples recorded.

Finding new samples demands careful consideration as the annotation time (and cost) is quite high, it takes minutes to annotate one sample. Human evaluation is not feasible with that large pool of samples. Uncertainty selection could most likely be used to identify new samples, finding an interesting subset of samples that the model is uncertain about. These samples could be the basis for human sample selection.

6.6 Thesis Reflection

As the combination of active learning, semantic segmentation, and Monte Carlo dropout are relatively unexplored, the outcome of the thesis was hard to foresee. I knew that writing a thesis requires a lot of time but I did not think that I would end up spending so much implementation time on other things than the actual active learning framework implementation. The great portion of the code written is not for the active learning framework but more to analyze samples, different dropout methods, explore ways to evaluate the network's performance and so on.

As with everything related to research, if I was supposed to write the thesis again, the path to the final goal would be much shorter. Writing this thesis has been both a great deal of fun and a really good learning experience.

7

Conclusion

This chapter contains two parts; short answers to the research questions and future work within the field.

7.1 Research Questions

Brief answers to the research questions:

1. **Can model uncertainty, estimated by dropout and Monte Carlo methods, be used for informative sample selection in Active Learning?**
   Yes, with Monte Carlo dropout is it possible to find informative samples which can be the foundation for sample selection.

   (a) **What is a good way to aggregate pixel uncertainty to sample uncertainty?**
   This question was not given enough focus in the thesis. Sample uncertainty was computed by averaging the pixel uncertainties.

2. **Is Active Learning applicable for Semantic Segmentation?**
   To some extent. It was possible to train networks for several active learning iterations for semantic segmentation. Long training times have restricted the number of active learning iterations and the number of networks that could be trained in total. Even if Monte Carlo dropout and Entropy perform better than baseline, more time would be needed to fully evaluate the question.

3. **Can Active Learning be used to limit the amount of annotated data in Semantic Segmentation?**
   Entropy successfully lowers the amount of training data for 5 active learning iterations compared to baseline, averaging 19.9% fewer samples required and achieving at least the same performance as baseline.

   Monte Carlo dropout, that was the main focus in the thesis, lower the number of samples required for 3 out of 4 metrics. However, no conclusion of limiting the total amount of annotated samples can be drawn as that is not the case for F1 - lane.

7.2 Future Work

Interesting aspects of the field that did not fit into this thesis:

- Investigate how uncertainty based on the default predictions can be used for sample selection.
- Repeat the thesis on another dataset. Maybe a dataset set that is *harder* in the sense that it is more dependent on the dataset size. This would likely be achieved with more classes, avoiding the problem explained in Section 3.4. Examples of datasets could be CityScapes [12], CamVid [9], or Apollo [23].
- Select new samples with another metric than uncertainty. One idea is to evaluate the core-set approach, by Sener and Savarese [40], explained in Section 2.4.5.
- Investigate a suitable number of dropout networks, the parameter n, explained in Section 3.7.1. In the thesis was $n = 100$, maybe a much lower value could achieve the same result.
- A big hassle in the thesis was the networks' long training time. The training time constrained the number of active learning iterations. Selecting 250-1500 samples each active learning iteration is not preferable. Applying active learning to a problem with shorter training time entails smaller selection sizes. This could possibly remove the problem with biased dataset selection.

Appendix

A

ENet Model

A detailed description of the ENet [38] model.

Layers are explained in detail in Table A.1. The network consists of 29 layers, divided into an encoder with 23 layers and a decoder with 6 layers. Spatial dropout is applied with a rate of 1% before layer 7 and with a rate of 10 % from layer 7 to 23.

ENet is constructed of bottleneck layers, Figure 2.4. Types of layers in ENet:

- **Regular Convolution:** Standard convolution with a $3x3$ filter. If not mentioned otherwise, the convolution type is regular convolution.
- **Dilated Convolution:** Provided number indicates the dilation rate. Dilated convolutions have a broader receptive field than regular convolutions.
- **Asymmetric Convolution:** Provided number n, indicated the size of the convolution. In this network, all asymmetric convolutions have $n = 5$. Asymmetric convolution replaces the regular convolution with a $5x1$ followed by a $1x5$ convolution. Asymmetric convolution requires fewer parameters than a $5x5$ convolution.
- **Downsample:** In downsample layers, max pooling is used. The indices of the max values are stored.
- **Upsample:** In upsample layers, max unpooling with the indices from their respective downsample layer.
- **Backward Convolution:** Upsamples the image with learnable parameters.

#	Name	Conv Type	Size (w x h x d)	Dropout rate
Input	Input		1024 x 208 x 3	
1	Initial		512 x 104 x 16	1%
2	Bottleneck 1.0	Downsample	256 x 52 x 64	1%
3	Bottleneck 1.1		256 x 52 x 64	1%
4	Bottleneck 1.2		256 x 52 x 64	1%
5	Bottleneck 1.3		256 x 52 x 64	1%
6	Bottleneck 1.4		256 x 52 x 64	1%
7	Bottleneck 2.0	Downsample	128 x 26 x 128	10%
8	Bottleneck 2.1		128 x 26 x 128	10%
9	Bottleneck 2.2	Dilated 2	128 x 26 x 128	10%
10	Bottleneck 2.3	Asymmetric 5	128 x 26 x 128	10%
11	Bottleneck 2.4	Dilated 4	128 x 26 x 128	10%
12	Bottleneck 2.5		128 x 26 x 128	10%
13	Bottleneck 2.6	Dilated 8	128 x 26 x 128	10%
14	Bottleneck 2.7	Asymmetric 5	128 x 26 x 128	10%
15	Bottleneck 2.8	Dilated 16	128 x 26 x 128	10%
16	Bottleneck 3.1		128 x 26 x 128	10%
17	Bottleneck 3.2	Dilated 2	128 x 26 x 128	10%
18	Bottleneck 3.3	Asymmetric 5	128 x 26 x 128	10%
19	Bottleneck 3.4	Dilated 4	128 x 26 x 128	10%
20	Bottleneck 3.5		128 x 26 x 128	10%
21	Bottleneck 3.6	Dilated 8	128 x 26 x 128	10%
22	Bottleneck 3.7	Asymmetric 5	128 x 26 x 128	10%
23	Bottleneck 3.8	Dilated 16	128 x 26 x 128	10%
24	Bottleneck 4.0	Upsampling	256 x 52 x 64	0%
25	Bottleneck 4.1		256 x 52 x 64	0%
26	Bottleneck 4.2		256 x 52 x 64	0%
27	Bottleneck 5.0	Upsampling	512 x 104 x 16	0%
28	Bottleneck 5.1		512 x 104 x 16	0%
29	Full Convolution		1024 x 208 x 3	0%

Table A.1: *ENet layer architecture consisting of bottleneck modules. Note that Bottleneck 3.X is a repetition of Bottleneck 2.X except the downsample layer. The encoder is the 23 first layers and the decoder the remaining 6. Input is the image size of 1024x208 pixels with 3 color channels. The output has the same spatial dimensions with one channel for each class (road, lane, void).*

Bibliography

[1] Bayesian Segnet - tutorial. http://mi.eng.cam.ac.uk/projects/segnet/tutorial.html. Accessed: 2018-10-22. Cited on page 13.

[2] Nvidia price for GTX Ti. https://www.nvidia.com/en-us/geforce/products/10series/geforce-gtx-1080-ti/. Accessed: 2018-10-22. Cited on page 60.

[3] ABSFreePic - image released under CC0 - Public Domain. http://absfreepic.com/free-photos/download/horse-racing-competition-3424x2245_72780.html. Accessed: 2018-10-22. Cited on page 8.

[4] Panoptic Leaderboard. http://cocodataset.org/#panoptic-leaderboard. Accessed: 2018-10-22. Cited on page 8.

[5] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org. Cited on page 25.

[6] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007. Cited on page 16.

[7] 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, 39(12):2481–2495, 2017. Cited on pages 6, 9, 10, and 12.

[8] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738. Page 229. Cited on page 9.

[9] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recogn. Lett.*, 30(2):88–97, January 2009. ISSN 0167-8655. doi: 10.1016/j.patrec.2008.04.005. Cited on pages 10, 13, and 64.

[10] Bor-Chun Chen, Chu-Song Chen, and Winston H. Hsu. Cross-age reference coding for age-invariant face recognition and retrieval. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. Cited on page 15.

[11] François Chollet et al. Keras, 2015. Cited on pages 15 and 25.

[12] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. Cited on pages 10 and 64.

[13] B. Demir and L. Bruzzone. A Novel Active Learning Method in Relevance Feedback for Content-Based Remote Sensing Image Retrieval. *IEEE Transactions on Geoscience and Remote Sensing*, 53(5):2323–2334, May 2015. ISSN 0196-2892. doi: 10.1109/TGRS.2014.2358804. Cited on page 14.

[14] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or Epistemic? Does it Matter? *Structural Safety*, 31(2):105–112, 2009. Cited on page 11.

[15] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`. Accessed: 2018-10-22. Cited on page 14.

[16] Yarin Gal and Zoubin Ghahramani. Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference. *arXiv preprint arXiv:1506.02158*, 2015. Cited on page 7.

[17] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR. Cited on page 7.

[18] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. *arXiv preprint arXiv:1703.02910*, 2017. Cited on page 15.

[19] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. Cited on page 10.

[20] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007. URL `http://authors.library.caltech.edu/7694`. Accessed: 2018-10-22. Cited on pages 15 and 16.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *CoRR*, abs/1502.01852, 2015. Cited on pages 1 and 5.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. Cited on pages 9 and 10.

[23] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. The apolloscape dataset for autonomous driving. *arXiv: 1803.06184*, 2018. Cited on pages 10 and 64.

[24] Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-Class Active Learning for Image Classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2372–2379. IEEE, 2009. Cited on page 14.

[25] A. Kendall, V. Badrinarayanan, and R. Cipolla. Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding. *ArXiv e-prints*, November 2015. Cited on pages 7, 9, 10, 12, and 13.

[26] Alex Kendall and Yarin Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5574–5584. Curran Associates, Inc., 2017. Cited on pages 7 and 11.

[27] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. CIFAR-10 (Canadian Institute for Advanced Research). . URL `http://www.cs.toronto.edu/~kriz/cifar.html`. Accessed: 2018-10-22. Cited on pages 7 and 16.

[28] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research). . URL `http://www.cs.toronto.edu/~kriz/cifar.html`. Accessed: 2018-10-22. Cited on page 16.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. Cited on page 5.

[30] R.; Perona L. Fei-Fei; Fergus. One-shot learning of object categories. *IEEE*

Transactions on Pattern Analysis Machine Intelligence, 28:594–611, April 2006. Cited on page 14.

[31] Roger Lanctot. Accelerating the Future: The Economic Impact of the Emerging Passenger Economy. june 2017. URL https://newsroom.intel.com/newsroom/wp-content/uploads/sites/11/2017/05/passenger-economy.pdf. Accessed: 2018-10-22. Cited on page 1.

[32] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, Winter 1989. Cited on page 5.

[33] Yann LeCun. The mnist database of handwritten digits. 1998. Accessed: 2018-10-22. Cited on page 15.

[34] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. *CoRR*, abs/1405.0312, 2014. Cited on pages 8 and 9.

[35] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. Cited on pages 8, 9, and 10.

[36] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. Cited on page 55.

[37] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. Cited on page 16.

[38] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation. *ArXiv e-prints*, June 2016. Cited on pages 3, 9, 10, 25, 36, and 67.

[39] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. ImageNet Large Scale Visual Recognition Challenge. *CoRR*, abs/1409.0575, 2014. Cited on pages 1, 5, and 9.

[40] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/pdf?id=H1aIuk-RW. Accessed: 2018-10-22. Cited on pages 15, 16, 55, 58, and 64.

[41] Burr Settles. Active Learning Literature Survey. Computer Sciences Tech-

nical Report 1648, University of Wisconsin–Madison, 2009. Cited on pages 13 and 14.

[42] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*, 2014. Cited on pages 6 and 9.

[43] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014. ISSN 1532-4435. Cited on pages 6, 7, 12, and 28.

[44] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient Object Localization Using Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 648–656, 2015. Cited on page 6.

[45] A. Vezhnevets, J. M. Buhmann, and V. Ferrari. Active Learning for Semantic Segmentation with Expected Change. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3162–3169, June 2012. doi: 10.1109/CVPR.2012.6248050. Cited on page 14.

[46] K. Wang, D. Zhang, Y. Li, R. Zhang, and L. Lin. Cost-Effective Active Learning for Deep Image Classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2591–2600, Dec 2017. ISSN 1051-8215. doi: 10.1109/TCSVT.2016.2589879. Cited on pages 11, 14, 15, and 60.