NTNU

Norwegian University of
Science and Technology

Using Deep Convolutional Networks to Detect Roads in Aerial Images

Olav Kåre Vatne

Master of Science in Informatics
Submission date: May 2016
Supervisor: Keith Downing, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

This thesis presents a brief introduction to aerial road detection and semantic segmentation of images. Datasets based on aerial imagery is often automatically generated from existing map data, which causes the dataset to be afflicted by label noise. Supervised training with datasets containing inconsistent labeling will penalize the classifier for making correct predictions, and can impact the resulting performance. The thesis investigates different approaches to decrease the impact of noisy labels in deep neural networks. This includes the bootstrapping method which modifies the loss function, and adjusting the training regime through the use of curriculum learning.

The bootstrapping method incorporates the predictions of the model in the cross-entropy loss function. This loss function modifies the label targets through a convex combination between the prediction and the label.

The thesis investigates curriculum learning and its impact on classifier accuracy. A curriculum strategy is first defined, which estimates the difficulty of every example. The classifier is then trained by presenting "easier" examples at the beginning, and then gradually introduce "harder" examples to the training set. This thesis proposes a curriculum strategy based on estimating inconsistency between a prediction made by a teacher model and the corresponding label.

The results from this thesis demonstrate that curriculum learning can improve generalization accuracy for the road detection task, and that a curriculum strategy based on estimating inconsistency is valid. Applying the bootstrapping loss function showed some robustness to the label noise present in aerial image datasets. However, this result was not statistically significant.

Sammendrag

Denne masteroppgaven undersøker temaer som deteksjon av veier fra flyfoto og semantisk segmentering av bilder. Datasett basert på flyfoto er ofte generert automatisk fra eksisterende kartdata. Dette kan føre til datasett som inneholder feil i målbildene. Veiledet læring med slike datasett vil potensielt føre til at klassifieringsalgoritmen blir straffet for korrekte prediksjoner, noe som kan påvirke ytelsen til algoritmen. Oppgaven undersøker derfor ulike måter å redusere de negative konsekvensene som kan oppstå ved bruk av motsigende målbilder, spesielt for dype nevrale nettverk. Den første metoden er kalt bootstrapping og endrer på tapsfunksjonen til nettverket. Den andre metoden, curriculum learning, strukturerer treningssettet som læringsalgoritmen trener på.

Bootstrapping metoden tar i bruk klassifieringsalgoritmens egne prediksjoner i tapsfunksjonen. Den modifiserte tapsfunksjonen produserer et nytt mål basert på prediksjonen og målbildet.

Videre tester oppgaven ut curriculum learning og hvordan dette påvirker nøyaktigheten til et dypt nevralt nettverk. Først må en strategi for hvordan man strukturer treningssettet bli utformet. Denne strategien må kunne sortere treningssettet fra lette til vanskelige eksempler. I denne oppgaven blir vanskelighetsgraden til et eksempel estimert ved å se på ulikheten mellom et målbilde og en prediksjonen gjort av en lærer-algoritme.

Ut ifra oppgavens resultater ser man at curriculum learning kan øke den generelle nøyaktigheten til en algoritme trent til å gjenkjenne veier i flyfoto. Sorteringsstrategien ser også ut til å fungere bra. Resultatene fra bootstrapping metoden var derimot mindre overbevisende. Selv om metoden viste seg å være en anelse mer robust mot motsigende mål, var det ikke statistisk signifikans i resultatene.

Preface

This project is the author's master thesis at the Department of Computer and Information Science, Norwegian University of Science and Technology.

I would like to thank my supervisor, Professor Keith Downing, at the Department of Computer and Information Science, Norwegian University of Science and Technology, for his helpful advice and guidance throughout this project.

I would also like to thank my girlfriend, friends and family for all their support. Special thanks to Ingvild Olaussen for proofreading, and Torgeir Haaland and Kåre Vatne for their insights and critiques.

Olav Kåre Vatne
Trondheim, May 31, 2016

List of Abbreviations

CNN convolutional neural network

CPU central processing unit

CRF conditional random fields

GIS geographic information system

GPU graphics processing unit

GSD ground sampling distance

SGD stochastic gradient descent

MAP maximum a posteriori

MSE mean squared error

ReLU rectified linear unit

SLR structured literature review

SPL self-paced learning

SPLD self-paced learning with diversity

SVM support vector machines

Contents

1 Introduction **1**
- 1.1 Background and Motivation . . . 1
- 1.2 Goals and Research Questions . . . 3
- 1.3 Research Method . . . 4
- 1.4 Contributions . . . 5
- 1.5 Thesis Structure . . . 5

2 Background Theory and Motivation **7**
- 2.1 Background Theory . . . 7
  - 2.1.1 Convolutional Neural Networks . . . 7
  - 2.1.2 Label Noise . . . 10
  - 2.1.3 Curriculum Learning . . . 12
  - 2.1.4 Road Extraction by Machine Learning . . . 12
  - 2.1.5 Evaluation Metrics . . . 14
- 2.2 Structured Literature Review . . . 15
  - 2.2.1 Identification of Research . . . 15
  - 2.2.2 Selection Process . . . 16
  - 2.2.3 Other Resources . . . 17
- 2.3 Related Work . . . 17
  - 2.3.1 Road Extraction Systems . . . 17
  - 2.3.2 Dealing with Noisy Labels . . . 25
  - 2.3.3 Learning by a Curriculum . . . 30
- 2.4 Background Discussion . . . 34

3 Methods **37**
- 3.1 System Overview . . . 37
- 3.2 Semantic Segmentation with CNN . . . 38
  - 3.2.1 Patch-based Approach . . . 39
  - 3.2.2 Network Architecture . . . 39
  - 3.2.3 Optimization . . . 41

3.2.4 Regularization . . . 44
3.2.5 Data Preprocessing . . . 45
3.3 Bootstrapping Loss . . . 46
3.4 Curriculum Learning . . . 48
3.5 Datasets . . . 51
3.5.1 Massachusetts Roads Dataset . . . 51
3.5.2 Norwegian Roads Dataset . . . 53
3.6 Implementation Details . . . 55

4 Experiments and Analysis **59**
4.1 Experimental Design . . . 59
4.2 Experimental Setup . . . 63
4.3 Experimental Results . . . 70
4.3.1 Bootstrapping for Datasets with Noisy Labels . . . 70
4.3.2 Curriculum Learning with Aerial Imagery . . . 72
4.3.3 Road Detection System . . . 78
4.4 Experimental Analysis . . . 79
4.4.1 The Effect of Bootstrapping . . . 80
4.4.2 Curriculum Learning by Using an Artificial Teacher . . . 82
4.4.3 Performance of the Road Detection System . . . 83

5 Conclusion **87**
5.1 Overview . . . 87
5.2 Evaluation . . . 88
5.3 Contributions . . . 91
5.4 Future Work . . . 92

Appendices **95**
A System Instructions . . . 95
B Experiment Tools Overview . . . 96
C Experiment Population Normality Assumption . . . 97
D Experiment E2 Results . . . 99
E Random Noise Experiment . . . 100
F Road Detection Results . . . 102

Bibliography **103**

List of Figures

1.1 Aerial image 2

2.1 Convolution example 8
2.2 Convolutional neural network 10
2.3 Example from the Norwegian Roads Dataset 13
2.4 Patch dataset examples 14
2.5 Shallow neural network 22
2.6 Noise matrix Q 27

3.1 Input patch and prediction 38
3.2 Components of the system 38
3.3 Visualization of feature maps 40
3.4 Activation functions 41
3.5 The Massachusetts Roads Dataset 52
3.6 Inconsistent labeling in the Norwegian Roads Dataset N50 54
3.7 Road centerline vector quality of Vbase 55
3.8 Example from Norwegian Roads Dataset N50 56

4.1 E1 - Robustness of bootstrapping in the Massachusetts Roads Dataset 70
4.2 E2 - Robustness of bootstrapping in the Norwegian Roads Dataset 71
4.3 E3 - Comparison of loss functions using the Norwegian Roads Dataset Vbase 72
4.4 E4 - Performance of curriculum learning with the Massachusetts Roads Dataset 74
4.5 E4 - Difficulty distribution 75
4.6 E5 - Performance of curriculum learning with the Norwegian Roads Dataset Vbase 75
4.7 E6 - Results from experiments with a less experienced teacher . . 77

4.8 E7 - Performance of the M1 road detection system trained with the Massachusetts Roads Dataset 79
4.9 E7 - Qualitiative results of the road extraction system 85
4.10 Examples of ill-suited line thickness 86

1 Normal Q-Q plot examples 98
2 E2 - Test loss comparisons for several levels of omission noise . . . 99
3 E2 - Precision and recall plots for several levels of omission noise . 99
4 Artifical noise examples 100
5 Label flipping results 101
6 Norwegian Roads Dataset extraction results 102
7 Massachusetts Roads Dataset extraction results 103

List of Tables

2.1 The terms and groups 16
2.2 Inclusion and quality criteria for the selection process 17

3.1 Hyperparameters for the CNN 42
3.2 Percentage of road pixels in the dataset 46
3.3 Hyperparameters for bootstrapping loss 48
3.4 Hyperparameters for curriculum learning 50
3.5 Raster line thicknesses for the Norwegian Roads Dataset 56

4.1 Experiments overview 60
4.2 Parameters of Experiment E1 64
4.3 Parameters of Experiment E2 65
4.4 Parameters of Experiment E3 66
4.5 Parameters of Experiment E4 66
4.6 Parameters of Experiment E5 67
4.7 Parameters of Experiment E6 69
4.8 Parameters of Experiment E7 69
4.9 Bootstrapping results 73
4.10 Curriculum learning results 78
4.11 Road detection system results 80

Chapter 1

Introduction

This chapter aims at giving an introduction to this thesis. This includes a brief presentation of the field of photogrammetry, road extraction from aerial imagery and an introduction to the thesis' research questions. Section 1.1 outlines the background and motivation, and in Section 1.2 goals and research questions are presented. Next, the research methods are described in Section 1.3. The contributions of this thesis are outlined in Section 1.4, and Section 1.5 presents an overview of the thesis structure.

1.1 Background and Motivation

Photogrammetry, or remote image sensing, is a field occupied with obtaining measurements about object or areas from overhead imagery, typically captured from an airplane or satellite. Common tasks in remote image sensing are land cover classification or road extraction. In land cover classification, each pixel of an aerial image is assigned a land cover label, such as grass, water, building or road. Road extraction[1] is a binary land cover classification task where each pixel is categorized as being a road or a non-road pixel.

Aerial and satellite images contain a variety of different features. Identification of these features is often done by a human expert, which can be expensive, both in terms of cost and time. Additionally, there is an increasing availability of high-resolution overhead imagery, which makes a machine learning approach for automatic land cover classification compelling.

[1] Also referred to as road segmentation or road detection in this thesis.

Figure 1.1: An aerial image captured above NTNU.

Feature extraction from aerial images is a non-trivial task because of the complexity presented by images. An array of pixel intensities might represent natural land covers such as terrain, vegetation, or artificial objects such as roads or buildings. Each type can look very different in terms of shape and texture. Additionally, aerial images are exposed to different variations of illumination such as objects casting shadows or changes in brightness. There is also the issue of occlusion. Roads can, for example, be partly occluded by cars and trees. The result is that extraction of information from aerial or satellite images can be challenging for an automatic extraction system.

A machine learning approach to land cover classification is typically formulated as a semantic segmentation task. Given an aerial image as seen in Figure 1.1, an algorithm should segment the image into disjoint regions, such as water, road, building, grass or tree. Alternatively, the algorithm could do a binary classification of the image, where each pixel is either a member or non-member of a class.

A convolutional neural network (CNN) is a special variant of a neural network, where connectivity between units have been constrained and parameter sharing is employed. This will reduce the number of parameters in the model. CNNs can, therefore, have many hidden layers, which enables the network to learn a

hierarchical representation of the input data. By having a large dataset and conducting normal backpropagation, the network can learn to extract informative features from raw pixel values.

This might be well suited for a road extraction system, where there is an abundance of aerial images available covering large areas. Labels can easily be generated for these areas from digital maps stored in a geographic information system (GIS) database. However, a large issue with utilizing aerial images for training a machine learning algorithm is the presence of noise in the labels. For most purposes, maps can be created without pixel level accuracy and still retain their usefulness. The result is that datasets created from digital maps have some degree of label noise, which can have negative consequences for the accuracy achieved by a machine learning algorithm. Mnih and Hinton [2012] have identified two types of label noise present in aerial images: Omission and registration noise. The former occurs when an object in an aerial image is missing in the label, and the latter happens when there is a misalignment between the object in the image and in the ground truth of the label.

One way of reducing the impact of noisy labels is by modifying the loss function. Cross-entropy loss assumes that the labels are correct, which results in noisy labels incorrectly penalizing an accurate classifier. The bootstrapping method proposed by [Reed et al., 2014], utilizes the classifier's implicit knowledge about the task by incorporating the classifier's predictions in a convex combination with the labels. The quality of these modified targets improves as the classifier's accuracy increases, which is why the method is called bootstrapping.

Another compelling way of improving generalization accuracy of a machine learning algorithm, is the use of curriculum learning [Bengio et al., 2009]. The method involves organizing the dataset according to a curriculum, where examples are sorted based on a criterion of "easiness". The examples considered "easy" are presented early on in the optimization process, whereas "harder" examples are presented later on.

1.2 Goals and Research Questions

In this section, the goal and research questions of this thesis are presented, as well as a brief motivation for each research question.

Goal statement:

The goal of this thesis is to create a convolutional neural network that can extract roads from aerial images.

The thesis will investigate how to further improve road extraction by considering the two research questions defined below. A system consisting of a convolutional neural network will be created, and experiments will reveal how well the system performs.

Research question 1:

Does the bootstrapping loss function give a significant improvement of precision and recall for datasets with noisy labels?

Because of the costs involved in creating accurate datasets, the thesis will look at techniques to reduce the effect of inconsistent labeling. For datasets related to aerial images, it is common to find omission and registration noise. Small and private roads are often unmarked on maps, and roads might be incorrectly placed on them.

Research question 2:

How can curriculum learning improve results in deep learning, and does this improve precision and recall for aerial images?

The thesis will also investigate the benefits of curriculum learning for a machine learning algorithm. By sorting examples from easy to hard, the learner can potentially be guided to a more advantageous area of parameter space and result in the learner finding a better local minimum, as well as reducing the time of convergence. This can be beneficial for deep learning, which often involves optimization of a lot of parameters. The challenge is to find ordering criteria that are applicable for aerial images.

1.3 Research Method

To address the research questions outlined in Section 1.2, a CNN has been developed. An aerial image dataset containing ground truth for roads have been acquired from [Mnih, 2013], and is publicly available under the name Massachusetts Roads Dataset. Additionally, a new dataset containing parts of the Norwegian road network has been created using aerial images and road centerline vectors provided by Kartverket.

These datasets will be used for training and testing the performance of the system, as well as evaluating the research questions. Furthermore, by utilizing a publicly available dataset, the system presented in the thesis can be compared to the performance of other similar systems [Mnih, 2013][Saito and Aoki, 2015].

1.4 Contributions

The thesis' main contribution to the field of machine learning is the examination of different approaches that can reduce the impact of inconsistent labeling. This involves experiments conducted on semantic road labeling task which has a high rate of naturally occurring inconsistent labeling. Experiments demonstrated that curriculum learning improved the generalization accuracy of a deep neural network, trained with real-world datasets. Furthermore, the thesis shows that a curriculum teacher based on estimating the inconsistency between a model prediction and a label can be an effective approach for curriculum learning. Whether the bootstrapping loss function can reduce the effect of inconsistent labeling is unclear. Experiments testing the bootstrapping loss function were, unfortunately, inconclusive.

1.5 Thesis Structure

The thesis is divided into five chapters. This chapter presents the motivation and research questions. In Chapter 2, sections describing the background theory, the structured literature review, and related work can be found. Chapter 3 outlines the methods and implementation, as well as presenting details about the datasets. Experiment results and analysis can be found in Chapter 4. The final chapter, Chapter 5, concludes the thesis by summarizing the results, outlining future work and presenting the thesis' contributions.

Chapter 2

Background Theory and Motivation

This chapter presents background theory and related works relevant to the research goal and questions. In Section 1.1 topics such as CNN, label noise and road detection are briefly introduced. The structured literature review is outlined in Section 2.2. The relevant research found by the structured literature review are then presented in Section 2.3. Finally, Section 2.4 concludes the chapter by a background discussion.

2.1 Background Theory

2.1.1 Convolutional Neural Networks

A CNN is a special kind of neural network, and it was one of the first deep learning models to perform well in commercial applications. A CNN is loosely based on principles drawn from neuroscience. According to [Goodfellow et al., 2016, Chapter 9], local connectivity and parameter sharing are properties characteristic for CNNs. These properties and the architecture of CNNs will be explored further below.

Convolution

In mathematics, convolution is a mathematical operation on two real-valued functions that express the amount of overlap of one function as it is shifted over another function. For machine learning applications, the data is usually discretized. The operation is therefore a discrete summation over the data, and is used to

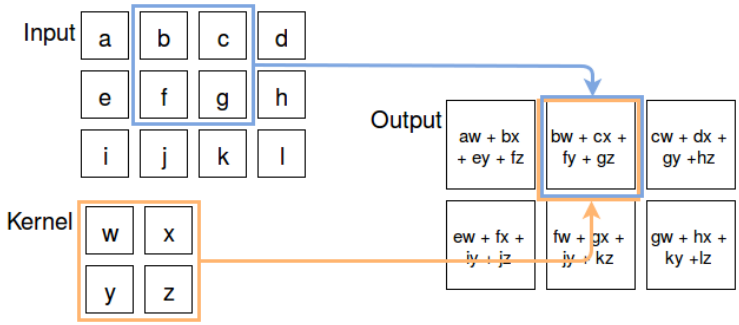


Figure 2.1: Example of 2D convolution without kernel-flipping.

calculate the weighted sum between the activations and the connection weights in a CNN. The discrete convolution operation without kernel-flipping:

$$(x * w)(t) = \sum_{a=-\infty}^{\infty} x[a]w[t+a].$$

For aerial images we extend the convolution operation to two dimensions, and limit the summation to a finite number of pixels. To convolve an image I, a two-dimensional kernel K containing the weights is shifted across the image:

$$(I * K)[i,j] = \sum_{m}\sum_{n} I[i+m, j+n]K[m,n].$$

This operation is visualized in figure 2.1 where a 2×2 kernel of weights is convolved with a 3×3 matrix of input values, and produces 2×3 outputs.

Local Connectivity

In a traditional neural network, each layer is typically fully connected. Each unit has connections to every unit in the previous layer. In a CNN, however, a unit interacts only with a small region of units in the previous layer. This region is often referred to as the unit's local receptive field. This kind of local connectivity can be very practical for high-dimensional data, such as images where meaningful features can be extracted using only a small area of the total image.

Local connectivity can be achieved by using a small kernel as seen in Figure 2.1. Instead of each unit being connected to all inputs, the unit only depends on a 2×2 input region.

If there are m inputs and n units, a matrix multiplication for a fully-connected network would require $m \times n$ parameters, as well as having a runtime of $O(m \times n)$. By using a kernel we limit the number of connections each unit may have to k. This requires only $k \times n$ parameters and a runtime of $O(k \times n)$. For image applications, the kernel size can be relatively small and still achieve good results, which can give big improvements in efficiency.

Parameter Sharing

The number of model parameters is further reduced by using parameter sharing. Each weight in the kernel is applied to every position of the input. In contrast, a neural network which is fully connected will have a separate weight for every connection. This can be redundant for high-dimensional data, where most of the features are localized. In images, for example, an important feature to extract are edges. A kernel with weights that are good at detecting edges at one location, will be equally good at detecting them in other locations.

The use of parameter sharing further reduces the storage requirement to k parameters. Usually, one kernel per layer is not enough, so several kernels with tied weights convolve the input. The layer will then produce output activations for different features. The outputs of several kernels are often referred to as feature maps.

Pooling

The pooling function is another operation typically associated with CNNs. A pooling function modifies the output of a layer in some way. It replaces a rectangular region of the output by a single value that has been determined by a summary operation. A common pooling function is the max pooling operation, which outputs the maximum within a rectangular neighborhood. The reason for utilizing pooling is that it helps the representation become invariant to small translations in the input. For example, a network created to classify whether an image depicts a cat or not will benefit from pooling, since the location of the cat in the picture is irrelevant. For tasks where the location of a feature is important, such as semantic segmentation, applying pooling should be done with restraint. Additionally, pooling reduces the number of input parameters for the next layer.

Layer Structure

A typical convolutional layer in a network consists of three stages. First, convolution sums the weighted inputs for every unit in the layer. Second, an activation function is applied to the resulting values. The rectified linear unit (ReLU) is a

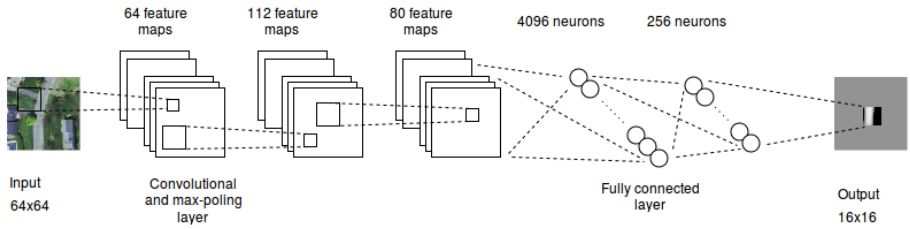


Figure 2.2: Convolutional neural network.

popular choice, and outputs either 0 or the weighted sum, depending on which is biggest: $f(x) = max(0, x)$. Finally, the pooling function modifies the output of the layer.

Network architecture

A CNN usually consists of both convolutional layers and fully-connected layers. The input layer and the initial hidden layers are convolutional layers, with fully connected layers attached at the end. Figure 2.2 shows a convolutional neural network configuration. The input layer and the two first hidden layers are convolutional layers. During training, the kernel weights for these layers are adjusted by backpropagation. Each feature map defines a set of kernel weights that are applied to all input pixels or activations. Usually, a CNN will reduce the necessity of feature engineering because it learns what suitable features to extract from input data. In images, a CNN is able to learn from raw pixel values without the use of feature extraction techniques found in computer vision.

2.1.2 Label Noise

There are several reasons for the presence of inconsistent labels in real-world datasets. For instance, the labeler was presented with insufficient information, or the dataset was automatically generated from a source with poor quality labels. Additionally, the samples could be ambiguous and therefore hard to label correctly by a human expert. Label noise, can in many cases, lead to negative consequences for a classifier. This can include reduced accuracy, increased model complexity, and more samples required for learning a target concept. Approaches for dealing with noisy labels can generally be divided into three groups: Data cleansing methods, noise-robust models, and noise-tolerant algorithms [Frénay and Verleysen, 2014]. These three groups are presented below.

Data Cleansing Methods

Data cleansing methods are filtering techniques applied to the training data in order to remove noisy samples before training. Noisy labels are first identified and then either relabeled or removed. An obstacle encountered by these methods is that harmful mislabeled samples can be difficult to distinguish from informative, but hard samples. Another problem is that filtering often relies on classifier predictions to automatically identify mislabeled samples. Such filtering techniques also run the risk of removing too many samples from the training set, which can also cause harm to the accuracy. Voting ensembles of several classifiers have been suggested to further improve classification filtering.

Another filtering technique is to simply remove the class label of samples deemed suspicious, and employ semi-supervised learning. This way, the distribution of samples are preserved while simultaneously reducing the consequences of inconsistent labels.

Noise-robust Models

Noise-robust models are algorithms that are naturally robust against label noise. Many algorithms have been shown to be less sensitive to label noise than others, especially to small amounts of label noise. This approach requires no noise modeling nor cleansing of the training set beforehand, because the algorithm is assumed to offer some robustness to mislabeled samples.

Algorithms that utilize regularization techniques to avoid overfitting, can be considered more robust to label noise. This can include convolutional networks that utilize regularization schemes such as dropout or weight decay. For ensemble methods, bagging often gives better results than boosting when faced with noisy labels [Dietterich, 2000]. The boosting algorithm AdaBoost, for example, combines many weak classifiers by iteratively re-weighting the training set to target samples the previous classifier had trouble predicting. Because mislabeled samples can be harder to predict, AdaBoost tends to put larger emphasis on mislabeled samples in later stages of learning, which can lead to increased sensitivity to label noise. In bagging methods, however, different subsets of the training data are used to create a diverse set of classifiers that are employed in a voting scheme. In this case, mislabelled samples can impact the performance positively, due to the increased variability in the classifiers.

Noise-tolerant Algorithms

In noise-tolerant approaches, existing algorithms are modified to be more robust towards label noise. This is often done by explicitly modeling a noise model during training. This way, a classifier learns to classify samples according to their true uncorrupted label, instead of the observed noisy label. Typically, the noise distribution and the model parameters are estimated simultaneously when training the classifier.

Techniques that incorporate label noise tolerance, such as particle competition, noise model estimation, bootstrapping, and co-training, will be further discussed in Section 2.3.

2.1.3 Curriculum Learning

Curriculum learning is inspired by how humans learn, and that learning typically is highly organized. For instance, by the use of a curriculum in educational institutions. Easier concepts tend to be introduced first. In terms of machine learning, this means presenting the classifier with easier samples first while training. To do so, a curriculum strategy has to be defined, which sorts the training set from easy to hard. Samples that are not near the decision boundary could be considered easy, for instance. Utilizing curriculum learning might lead to a faster convergence time, and help the algorithm reach a better local minimum. Different works show that curriculum learning can achieve better generalization for many tasks [Bengio et al., 2009] [Kumar et al., 2010] [Jiang et al., 2014].

A challenge for curriculum learning is defining a sorting measure that enables a curriculum strategy of gradually introducing harder training samples to the learner. This issue, and works related to curriculum learning, is further explored in Section 2.3.

2.1.4 Road Extraction by Machine Learning

Road extraction is a part of the field of photogrammetry, and involves technology for map production and measurements of objects in images. As digital acquisition systems for capturing aerial images have become commonplace, the availability of high-resolution aerial images have increased. Coupled with the increasing need for detailed spatial information in GIS databases and production of digital maps, a lot of approaches for automatic object extraction from aerial imagery have been suggested. A reliable and accurate detection system could be beneficial in terms

(a) Aerial image. (b) Label image.

Figure 2.3: Image and label example from the training set of the Norwegian Roads Dataset.

of increased levels of details, while reducing the cost associated with map production.

There are three distinctive approaches for extracting objects from aerial imagery. Manual, semi-automatic and automatic methods. The semi-automatic approach integrates computer vision techniques and machine learning into the workflow of skilled human labelers. For instance, Google's Ground Truth project employs skilled operators that utilize advanced software tools to improve the accuracy of Google's map products [Google, 2013].

To do automatic road detection, supervised learning is often employed. This requires a training dataset of aerial images and labels. The labels for road detection are usually binary images that show the ground truth of roads. Creating the label images by manually labeling aerial imagery would be prohibitively expensive, which is why ground truth labels are often generated from existing map data. Figure 2.3 shows an example of an aerial image and a label typically found in a road segmentation dataset.

From these large training set images, smaller training set patches are extracted. The supervised learning algorithm is given a patch dataset d containing N training examples in the form $d = \{(s_1, m_1), ..., (s_N, m_N)\}$, where s_i is an aerial image patch and m_i is the corresponding ground truth label. Examples of aerial image patches and labels can be found in Figure 2.4. The learning algorithm's task is to learn a suitable mapping from the input space of aerial images to the output

Figure 2.4: Aerial image and label examples from a patch dataset. The patches originate from the Norwegian Roads Dataset. Each label depicts the road ground truth from the center location of the aerial patch.

space of road ground truth. In neural networks, this mapping is typically learned by minimizing the cross-entropy loss by gradient descent optimization.

The resulting classifier has hopefully extracted some useful patterns from the training data, which enables it to generalize to the task of road extraction. This is verified by computing the mean squared error (MSE) on a test set, containing examples not seen during training. The machine learning approach for road extraction should therefore be able to train algorithms which can predict the ground truth reasonably well for new unseen aerial image patches.

2.1.5 Evaluation Metrics

A common way to evaluate road extraction systems is by the quality measures, correctness and completeness [Wiedemann et al., 1998]. These are closely related to precision and recall. Precision measures the fraction of true roads that are correctly detected, while recall is the fraction of predicted roads that are true roads. Because the label maps are not perfectly aligned with the images, it is also common to use a relaxed measure of precision and recall. This is accomplished by treating predicted road pixels within p pixels of a true road pixel as being correctly detected. True roads within p pixels of a predicted road pixel are considered correctly recalled. The slack parameter p is often set to 3 pixels [Mnih and Hinton, 2010].

2.2 Structured Literature Review

The purpose of conducting a structured literature review (SLR) is to get an overview of the field of remote image sensing, as well as the research related to curriculum learning and dealing with noisy labels. The SLR method has been chosen to investigate these topics, and provides a formal way of identifying the information available.

2.2.1 Identification of Research

This section outlines the strategy that was utilized to search for primary studies. By utilizing a search strategy, literature relevant to the defined research questions can be identified and collected. Search terms have been defined, as well as literature resources.

Literature Resources

The following resources were searched in order to identify and collect relevant material:

- ACM digital library
- IEEExplore
- ScienceDirect
- CiteSeer
- Springer Link
- ECCV (conference)
- NIPS (conference)

Key Terms and Groupings

Table 2.1 describes the different word groups employed by the search strategy. Each group specifies a number of terms that are either synonyms or related to each other. Additional terms have been appended, as a result of a search strategy validation.

Table 2.1: The terms and groups.

	Group 1	**Group 2**	**Group 3**	**Group 4**	**Group 5**	**Group 6**
T1	Aerial images	Curriculum learning	Noisy labels	Neural network	Segmentation	Roads
T2	Satellite images	Guided learning	Missing labels	Convolutional neural network	Classification	
T3	Remote sensing	Example ordering	Semi-supervised	Machine learning		
T4	Images		Noisy data (appended)	Deep neural networks (appended)		

Search Strategy

Based on Table 2.1 several search expressions were devised. These expressions were run for each of the resources listed in the literature resources section.

- (aerial images OR satellite images) AND (segmentation OR classification)
- (remote sensing OR aerial images) AND (noisy labels OR missing labels OR semi-supervised OR noisy data)
- (neural network OR machine learning OR convolutional neural networks) AND (aerial images OR satellite images)
- (curriculum learning OR guided learning) AND (machine learning OR neural networks OR deep neural networks OR convolutional neural network)
- (segmentation OR classification) AND (roads)
- (Noisy labels OR missing labels OR semi-supervised OR noisy data) AND (machine learning OR neural network OR convolutional neural network)
- (aerial) AND (noisy labels OR missing labels OR semi-supervised OR noisy data) AND roads

2.2.2 Selection Process

The search expressions resulted in a large number of hits for each resource. The top 15 results for each expression were stored, and a selection process was created

to reduce the over 400 studies to a more manageable number. At first, the title of each study was evaluated. If the title seemed unrelated to the research goal or the research questions found in Section 1.2 the study was removed. Then the title and the abstract were evaluated using the inclusion criteria defined in Table 2.2. Finally, the remaining studies were read while considering both the inclusion and quality criteria.

Table 2.2: Inclusion and quality criteria for the selection process.

Id	Criteria	Screening step
IC 1	The study's main concern is curriculum learning, dealing with noisy labels or road extraction systems	1
IC 2	The study is presenting empirical results	1
IC 3	The study preferably involves machine learning	1
QC 1	The research has a clear aim	2
QC 2	Is there an adequate description of related works?	2
QC 3	How rigorously has the method or technique been tested?	2
QC 4	Is there a future work section?	2

2.2.3 Other Resources

By conducting a SLR, a number of relevant studies were identified. Additional literature was discovered by finding works citing the SLR papers on Google Scholar, and reading survey papers on road extraction systems [Mena, 2003] [Trinder, 2009] and label noise [Frénay and Verleysen, 2014]. The surveys provided a comprehensive overview of these topics and enabled further identification of relevant literature.

2.3 Related Work

2.3.1 Road Extraction Systems

There is a large amount of literature regarding proposed methods for automatic road extraction systems. This includes segmentation, edge detection, knowledge

based methods, fuzzy classification methods, and region growing methods. For a thorough review of different road extraction systems, see [Trinder, 2009] [Mena, 2003]. The main aim of this section is to present approaches based on machine learning.

Types of Aerial Imagery

Aerial and satellite imagery are captured using a whole range of sensors. A lot of approaches found in remote sensing are developed for certain types of sensor data. Below is a list of different types of aerial imagery:

- Monochromatic (single channel or grayscale) images
- Infrared band
- Color images (Red, green and blue channel)
- Hyper-spectral images
- Synthetic aperture radar images (SAR)
- Laser images (LIDAR)

The focus of this review is approaches for color images, which is the most common type of aerial imagery.

Machine Learning

In the previous two decades, the availability of high-resolution images covering large areas has increased. These images can have a resolution of around 1 square meter per pixel or higher. At this resolution, finer details such as cars, buildings and trees can be distinguished. Having a higher resolution for images also result in much more variability in terms of shape, texture and illumination. Machine learning algorithms that can learn highly non-linear decision boundaries have therefore become more common for aerial imagery applications. To successfully discriminate between object classes, more spatial context has been used to create a richer feature representation, as well as more data being used for training. Additionally, structured prediction methods such as conditional random fields (CRF), have become popular for smoothing in semantic segmentation applications.

Classifiers

High-resolution aerial imagery has created a need for more sophisticated classifiers. A classifier must encode knowledge of shape and context in order to discriminate between similar objects. In a road extraction system for high-resolution

imagery, the classifier should, for example, be able to distinguish between roads and gray rooftops. The classifier is therefore required to learn highly non-linear decision boundaries. This can include support vector machines (SVM), ensemble methods and deep neural networks.

In Mayer et al. [2006] six road extraction approaches were compared on both aerial images and satellite images. The approaches were based on image processing techniques, especially line detection. Many of the approaches rely on characteristics specific to roads, such as identifying parallel lines in images. In addition, some approaches utilized fuzzy classification or unsupervised clustering. Both scenes from urban and rural areas were used for the comparison. The authors defined a minimum of 0.6 and 0.75 in precision and recall, in order for a road extraction system to be of any practical use. In summary, most of the approaches performed well for images with limited complexity, such as rural areas. None of the methods performed above the defined threshold for images containing suburban or urban scenes. The low performance for urban areas reinforces the need for classifiers that can learn complex decision boundaries.

A hybrid approach for road extraction using SVM and image processing techniques was proposed by Song and Civco [2004]. First, a SVM classifies images into a road and a non-road set. Second, the images in the road set are segmented into homogeneous areas by utilizing the region growing technique.

The SVM did not perform sufficiently well for areas that appear similar to roads, especially for urban areas, with structures such as parking areas and roof tops. Therefore, they extracted shape descriptions from the segmented regions, and exploited road characteristics to remove areas not corresponding to roads. This is done by a threshold operation on shape descriptions such as smoothness and density.

The approach was tested experimentally on IKONOS satellite images. The approach performed well. However, road segments obscured by shadows or overhanging trees were a problem, as well as narrow roads and intersections.

Ensemble methods have also been applied to tasks involving aerial imagery. In [Kluckner et al., 2010], randomized forest was used for land cover classification on high-resolution imagery. Training a randomized forest involves training several binary decision trees on subsets of the training data. The result is an ensemble of weak classifiers that together can provide robust and accurate predictions. Dollar et al. [2006] proposed a supervised edge detection method, which was tested for road detection. This method trains a boosted tree classifier, which is similar to

a decision tree, except that boosted classifiers are used to split the data at each node in the tree.

Mnih and Hinton [2010] proposed an automatic road extraction approach for large real-world datasets. The system consists of a neural network with millions of weights that is trained on a large dataset of aerial images. A graphics processing unit (GPU) was utilized to train the network.

They formulated detection of road pixels from aerial images as a patch-based semantic segmentation problem. The goal of the model is to predict whether or not pixels belong to a road class, given an image patch. This is achieved by having the neural network model the distribution:

$$p(N(M(i,j), w_m) \mid N(S(i,j), w_s),$$

where S is an aerial image and M is a corresponding road label image. $M(i,j) = 1$ if $S(i,j)$ is a road pixel and 0 otherwise. $N(I(i,j), w)$ denotes a $w \times w$ large patch of pixels centered at location (i,j) of a large image I. Using smaller image patches instead of entire images for modeling the distribution, limits the image context which the model use to make predictions. This approach is less computationally expensive, and by retaining a relatively large $w_s \times w_s$ can still provide enough image context to create a competent road detector.

The network consisted of a single hidden layer with 12288 units, an input layer of 4096 units, and 256 output units. This enables the network to predict 16 x 16 road prediction patches given 64 x 64 aerial image patches. The network was trained by stochastic gradient descent (SGD) to minimize the cross-entropy loss between training labels and the predicted map patches. Furthermore, unsupervised pre-training was used to initialize the parameters of the network.

Experiments were conducted on two large aerial image datasets, and the network achieved good performance both in terms of precision and recall. A problem identified by Mnih and Hinton [2010], is that the model is penalized for correct predictions because of noisy labels. Smaller roads or paved areas have often not been marked in the dataset. Additionally, the road labels in the dataset have been generated from road centerline vectors with a fixed width, which results in some roads not being covered by the ground truth. This may lead to a decrease in model performance, since the model is penalized for correctly labeling the roads when minimizing the cross-entropy between predictions and inconsistent labels.

The problem with inconsistent labels in the context of aerial images was investigated in [Mnih and Hinton, 2012]. Two loss functions were proposed to deal

with label noise found in aerial images. This model resembles the patch-based approach used in Mnih and Hinton [2010]. However, a deep neural network consisting of three hidden layers was used. The first two hidden layers are locally connected layers, while the final hidden layer is fully connected. Unlike CNNs, there are no parameter sharing involved.

The proposed deep neural network performed significantly better in terms of precision and recall, compared to the shallow neural network in [Mnih and Hinton, 2010]. By utilizing the robust loss functions, performance was further improved.

In addition, CNNs have been used to do road detection. Both Mnih [2013] and Saito and Aoki [2015] have tested this type of network, on the publicly available datasets, Massachusetts Roads Dataset and Massachusetts Buildings Dataset. The networks consisted of 5 layers, three convolutional layers and two fully connected layers. The networks utilized stride and max-pooling, but only in the first layer. They did, however, choose different kernel sizes for the convolutional layers. For instance, Mnih [2013] used a first layer kernel size of 16×16, whereas Saito and Aoki [2015] used a kernel size of 9×9. Both networks were trained on patches of 64×64 pixels. However, Saito and Aoki [2015] extended the detection task to simultaneously prediction road and building footprint pixels, by training the network on a merged version of the previously mentioned datasets. Predictions were made by an output layer of 768 units, which represent a three channel 16×16 prediction patch. Each prediction patch pixel has three probability values which indicate whether an input pixel belongs to the non-road, road, and building class.

Datasets

Learning complex decision boundaries and the variations present in the high-resolution aerial imagery requires a lot of training data. In previous studies, much smaller datasets have been used [Mokhtarzade and Zoej, 2007] [Song and Civco, 2004]. Only eight test images ranging from 1600 to 4000 pixels in width and height were utilized to evaluate automatic road extraction approaches in [Mayer et al., 2006]. The trend in road extraction and land cover classification literature is to use increasingly larger datasets.

A high-resolution aerial dataset used to optimize the randomized forest algorithm in [Kluckner et al., 2010], contains 155 images and covers an area of about 85 square kilometers. Each of these images is 11500 x 7500 pixels in size and have a ground sampling distance (GSD) of 8 centimeter.

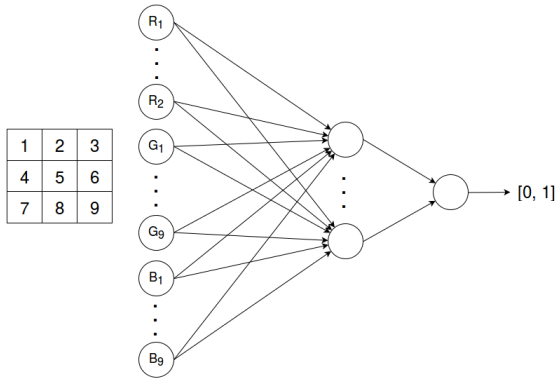


Figure 2.5: Pixel neighborhood and shallow neural network used for road detection by Mokhtarzade and Zoej [2007].

In both [Mnih and Hinton, 2010] and [Mnih and Hinton, 2012], two large datasets were used to optimize neural networks with many parameters. These datasets cover 500 square kilometers at GSD of around 1.20 meters per pixel.

The Massachusetts Roads Dataset introduced by Mnih [2013], consist of 1171 aerial images, each with a height and width of 1500 pixels. The dataset covers an area of over 2600 square kilometers, and contains a variety of regions, such as urban, suburban and rural areas. The GSD is 1 meter per pixel.

Feature Representation

Another trend in road detection is to extract features from larger contexts or pixel neighborhoods. In addition to increasing the classifiers ability to discriminate between objects sharing similar texture and shape, using larger contexts are necessary for images with a low GSD.

Mokhtarzade and Zoej [2007] proposed using a shallow neural network for road detection. For this approach, a normalized 3×3 neighborhood of pixel values was used as input features to a neural network. The neural network is illustrated in Figure 2.5.

In Song and Civco [2004], shape description is generated from segmented images, and certain characteristics descriptive of roads, such as being lengthy and

narrow, are used to remove objects that share spectral similarities with roads. Each shape's border length, area, pixel count and approximate radius are used to measure the shape index and density. Road shapes should have a large shape index value and a small density. Questionable shapes with measurements not characteristic of roads are removed by a threshold operation.

A much larger neighborhood of pixels was utilized as features in [Mnih and Hinton, 2012] and [Mnih and Hinton, 2010]. In this case, 64×64 pixels with minimal pre-processing formed the input to both networks. Furthermore, the neural network learns suitable features which enables it to distinguish road pixels from non-road pixels. The large neighborhood is helpful when resolving ambiguities often found in urban environments.

Combining images with height or elevation information can also increase classifier accuracy. For instance, height information makes gray rooftops easier to distinguish from street areas. The effectiveness of combining images and height maps was demonstrated by Kluckner et al. [2010] where both color and height cues were integrated as features. The height information significantly improved the performance of the classifier.

Conditional Random Fields

The smoothness assumption is a strong piece of prior knowledge we have about images. Neighboring pixels tend to influence each other, and are more likely to belong to the same object or class. CRF is a way to explicitly model dependencies between neighboring pixels, and is often utilized in semantic segmentation tasks to obtain a smoother segmentation result. The goal of semantic segmentation is to split an image into disjoint regions, where each region is associated with a certain class label.

In the study by Kluckner et al. [2010], an approach for land cover classification given high-resolution aerial images was presented. Aerial images were segmented according to five classes: Building, tree, waterbody, green area, and streetlayer. Attributes such as color and edge response were extracted from aerial images, and combined with height information to create an efficient feature representation based on covariance matrices. A randomized forest classifier was trained to learn a conditional probability distribution over the possible class labels given the feature representation. The CRF approach was tested in combination with this classifier, and was shown to significantly improve accuracy for semantic segmentation tasks.

Normally, a classifier predicts each label independently. However, for structured prediction tasks, such as segmentation, contextual information can be useful. The CRF approach combines graphical modeling and classification. It involves minimizing the cost of a label assignment for a pixel, as well as the cost of this assignment in relation to the neighboring pixel assignments. The cost of a label assignment, which is called the unary potential can be estimated from a prediction made by a classifier. The neighborhood cost is calculated through the pairwise class potentials between an output pixel and its neighbors.

CRF is often formulated as a graph with V nodes, where each node represents a pixel, and can be assigned a label l from a discrete set of classes, such as grass, tree, and roads. The edges are represented by the set E, and models the relationship between neighboring pixels or nodes. The energy of a label assignment is modeled by:

$$E(y) = \sum_{i \in V} \Psi_i(l_i, x_i) + \sum_{i,j \in E} \Psi_{ij}(l_i, l_j),$$

where $\Psi_i(l_i, x_i)$ is the unary potential modeling the likelihood of pixel x having a label assignment l. These estimates can be obtained from a classifier. $\Psi_{ij}(l_i, l_j)$ is the pairwise potential modeling the coherence of neighboring pixels. The final label assignment $\hat{y}$, which takes the label assignments of neighboring pixels into account, is obtained by minimizing the energy: $\hat{y} = argmin_y E(y)$. Whereas the first term of $E(y)$ prefers the label assignment with the lowest cost, the pairwise potentials prefer pixel neighborhoods to have similar label assignments. This results in the most probable label assignment $\hat{y}$ given the neighborhood.

CRF is also commonly used for semantic segmentation tasks involving general scene understanding. Alvarez et al. [2012] investigated road scene understanding by utilizing semantic segmentation for images found in environments encountered by vehicles. In this approach, a CNN is trained to extract features and predict image patches. These class predictions are in turn utilized by the CRF as unary potentials.

The proposed method was tested on Cambridge-driving Labeled Video Database, which contains high-resolution images of roads, signs, pedestrians and other objects found in a driving vehicle environment. The use of CRF did improve the overall accuracy, especially for large classes such as roads. For classes with less presence in the dataset the accuracy decreased.

Improvements in classification accuracy is further shown empirically by Schindler [2012]. Several random field techniques were tested on different aerial image

datasets with low ground sampling distance. Enforcing a smoothness prior significantly improved accuracy.

An approach similar to CRF is proposed by Mnih and Hinton [2010], where a post-processing step is introduced to improve the predictions produced by the neural network by incorporating knowledge about nearby predictions. This is achieved by training another neural network that predicts 16×16 map patches given 64×64 patches of predictions. The approach was applied to reduce the amount of gaps and disconnected road present in the predictions of the base network.

2.3.2 Dealing with Noisy Labels

Supervised learning works well for applications where there are a lot of labeled data available. For object recognition, the large-scale image database ImageNet has often been utilized. This database provides millions of manually annotated and quality controlled images, organized in a semantic hierarchy [Deng et al., 2009]. However, for some tasks, such as semantic segmentation and object detection, manually labeled data are expensive and time consuming to create, and high quality datasets can be hard to obtain. Automatically creating large datasets from internet resources, such as image search engines, GIS databases, and user annotated images can be very practical in terms of reducing the costs of creating very large datasets.

Unfortunately, such datasets will often contain noisy or weak labels. User annotation for images are usually incomplete, image search engines often return images unrelated to the search term, and GIS databases might be outdated and missing important object information. This can have negative consequences for a supervised algorithm, which in most cases assume that the labels are correct. These consequences are more evident in datasets containing substantial amounts of inconsistent labels.

There are three main approaches to dealing with noisy label, as outlined in Section 2.1. However, in this section only noise-tolerant algorithms are explored, and especially methods that explicitly introduce noise tolerance into deep neural networks.

Learning to Label Aerial Images from Noisy Data

The problem with inconsistent labels in aerial images was investigated in [Mnih and Hinton, 2012]. In this work, two types of label noise were identified, which datasets constructed from maps are especially susceptible to. Omission noise is

defined as objects that appear in the aerial image, but not in the map. Registration noise occurs when the location of the object in the map is inaccurate. Considering that the presence of label noise might negatively impact the classifier accuracy, Mnih and Hinton [2012] proposed two robust loss functions that can be incorporated in a deep learning framework.

The first loss function proposed explicitly models asymmetric noise, and is designed to deal with omission noise. It treats label $\tilde{y}$ as a noisy observation generated from true label y, according to a noise distribution $p(\tilde{y} \mid y)$. This distribution is determined by two parameters, set before training. The noise model modifies the derivatives produced by the loss function, which results in the neural network being penalized less for making confident, but incorrect predictions.

The second loss function is an extension of the first, and considers both omission and registration error. The noise distribution model is combined with a generative model, where different crops of an unobserved, perfectly registered map are generated. These crops are used by an expectation–maximization like algorithm to estimate the true label, which can reduce the effect of local registration errors.

The loss functions were evaluated on two large aerial road detection datasets. There was a significant improvement in precision and recall for both loss functions, compared to the baseline deep neural network and the network in [Mnih and Hinton, 2010]. The second loss function performed slightly better than the first for one of the datasets. This dataset had substantial amounts of registration errors.

Training Convolutional Networks with Noisy Labels

Sukhbaatar and Fergus [2014] demonstrate robustness towards label noise in a modified CNN. The method models the noise through an additional noise layer which is estimated alongside the network parameters during SGD training. The combined model is optimized to predict the noisy labels. The goal of the noise layer is to approximate the noise distribution of the data and thereby forcing the base model to predict the true labels. Experiments were conducted for several datasets, and showed that the approach does well for higher levels of label noise.

Like other noise-tolerant methods, the algorithm involves learning a noise distribution, where the examples observed by the algorithm have been altered by noise. The noise distribution is parameterized by a matrix Q, where each value specifies the probability of observing noisy label $\tilde{y}$ given the true label y: $q_{ji} := p(\tilde{y} = j \mid y = i)$.

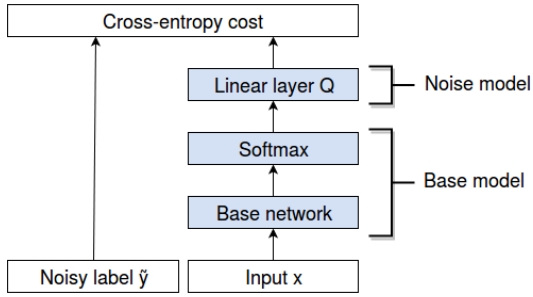


Figure 2.6: Noise matrix Q is inserted between loss function and output layer of the model.

The matrix Q is implemented by a constrained linear noise layer, which is added to the base model. This is illustrated in Figure 2.6. The weights between the output layer of the base model and the noise layer correspond to probabilities found in Q. These conditional probabilities q_{ji} are usually unknown, but an approximate noise distribution can be estimated by conventional backpropagation.

The training procedure starts with Q fixed to the identity matrix, while the base model is trained. After a number of epochs, the weights in the linear noise layer will also be adapted by backpropagation. To ensure that Q captures the noise properties of the data, a regularization term is used to make it converge to the true noise distribution. Effectively, the prediction of the combined model will be given by:

$$p(\tilde{y} = j \mid x) = \sum_i q_{ji} p(y = i \mid x),$$

where the noisy predictions are made by the conditional probabilities encoded in q and the prediction of the base model. Hopefully, the base model will learn to predict the true labels y instead of the noisy labels $\tilde{y}$.

The approach was tested using the Google street-view house number, CIFAR10 and ImageNet dataset. Noisy labels were synthesized by switching labels of examples with a fixed probability defined by a probability matrix.

The noise layer extension consistently achieved better accuracy compared to the baseline network, and also displayed more robustness to inconsistent labeling. Furthermore, the performance of the modified network decreased slower with increasing noise levels. This approach was also efficient in learning the noise dis-

tribution.

For the ImageNet dataset, the labels were switched on half of the examples in the dataset. The approach did better than the baseline model. Additionally, the approach also outperformed the baseline model trained on the clean unaltered subset of the dataset, showing that the noisy examples carry useful information.

Training Deep Neural networks on Noisy Labels with Bootstrapping

In Reed et al. [2014], a generic approach to handling noisy and incomplete labels in supervised deep learning was presented. The approach incorporates a notion of perceptual consistency in the loss function. A prediction is consistent if the same prediction is made given similar percepts. The learner is allowed to disagree with inconsistent labels by using its own implicit knowledge stored in the network parameters.

They present two ways of incorporating perceptual consistency in a network. The first involves a reconstruction loss and a noise distribution model. The other method is introduced as bootstrapping, and avoids directly modeling the noise distribution, by using a combination of training labels and the current model's prediction to generate targets.

The bootstrapping approach tweaks the loss function to be a convex combination of the model prediction q and the label y. The β parameter decides the prediction's contribution to the convex combination, and is usually set to a relatively low value. The bootstrapping loss function is denoted:

$$\mathcal{L}(q, y) = -\sum_{k=1}^{L}[\beta y_k + (1-\beta)z_k]log(q_k),$$

where z_k is assigned a value of 1 only for the most probable class $l \in L$, according to the class prediction probabilities q_k. This is denoted $z_k := \mathbb{1}[k = \text{argmax } q_i, i = 1 \dots L]$, and is the maximum a posteriori (MAP) estimate of q given the data x.

This approach was tested on several image tasks, and yielded substantial improvements for several datasets. For all tasks, deep neural networks were trained and used as the baseline. The networks that were modified to include perceptual consistency were trained by fine-tuning the baseline networks.

The developed method performed better than the baseline. For MNIST handwritten digits dataset, artificial label noise was added. The bootstrapping performed

better than the baseline for noise fractions above 35 percent.

In the task of emotion recognition using Toronto Faces Database, bootstrapping performed better than the baseline and other approaches. This dataset contains over 4000 face images with emotion labels. This kind of labeling can be subjective and the dataset might therefore contain mislabeled samples.

Overall, bootstrapping improves the robustness of a model and is fairly simple to implement. The bootstrapping approach achieved comparable performance to the loss function that requires a noise distribution model.

Semi-supervised Learning

Literature for semi-supervised learning has also covered noisy labels. In semi-supervised learning, a fraction of the dataset is assumed to be correctly labeled or clean, while the remaining data either have no label or is weakly labeled. For tasks where labels are expensive to produce, semi-supervised learning can be advantageous. Furthermore, label noise can have a big impact on semi-supervised learning, since only a small subset of the dataset is labeled. The approaches described below takes an active role in the learning process by iteratively improving the quality of the dataset, which is similar to the bootstrapping technique.

Self-training has been suggested as an approach to training a classifier when there is a considerable amount of missing or weak labels present in the dataset [Rosenberg et al., 2005]. A classifier is trained using an initial set of fully labeled examples, which is then used to predict weakly labeled examples. The set of fully labeled examples is expanded by adding a selection of the predicted examples. The selection can be based on the prediction confidence of the model. The process is repeated, which will incrementally increase the number of fully labeled examples until the entire dataset has been assigned a label.

In co-training, proposed by Blum and Mitchell [1998], two classifiers are trained on separate views of the data. This requires two feature sets that are conditionally independent given the class, and that each feature set is sufficient for label predictions. The training set is iteratively expanded by adding unlabeled examples both classifiers can predict with a high confidence.

In [Breve et al., 2015], particle competition and cooperation is used to address noisy labels in a semi-supervised setting. A graph is constructed from the dataset where each example has a node, and edges connect similar examples. Each labeled example has an associated particle, which will traverse the graph, cooperate with other particles of the same class, and compete with other particle teams.

Each particle visits different nodes according to simple rules and will, for each visit, increase the probability of the node belonging to the particle's own class. When the algorithm converges, each node or example is labeled according to the particle team or class that has the largest node probability. The structure of the graph and the particle dynamics will in effect classify unlabeled examples and discover inconsistent labeling.

Most of these approaches require an initial dataset that contains clean labels, which, in many cases, require precise manual labeling. For road extraction, a fully labeled, but noisy dataset, can be generated from existing map data. The problem is that we cannot assume that a subset of this dataset contains clean labels, without a thorough inspection. Therefore, techniques that treat the entire dataset as noisy are better suited for this task.

2.3.3 Learning by a Curriculum

Inspired by how humans learn in an organized fashion, Bengio et al. [2009] presented curriculum learning and investigated how machine learning can benefit from modifying the training regime. In curriculum learning, a learner is gradually presented with harder training samples. In order to do this, an ordering criterion that can identify easy samples must be devised. Experiments showed that a simple multi-stage curriculum reduced convergence time and increased accuracy.

A study conducted by Erhan et al. [2010] investigated why supervised learning tasks benefit from unsupervised pre-training. It showed that examples presented early on in training have a disproportionate influence on the outcome of the training procedure. Earlier training can trap the SGD in a basin of attraction, which can be hard to escape from. By using a curriculum strategy the learner can potentially be guided to better areas in parameter space and lead to better local minima.

Curriculum learning shares similarities with boosting algorithms such as AdaBoost. This algorithm trains several weak classifiers by iteratively re-weighting the training set, which gradually puts more emphasis on difficult samples. Unlike boosting, curriculum learning starts off training by considering the easiest examples found in the dataset.

Active learning [Cohn et al., 1996] is also considered related to curriculum learning. In active learning, the learner participates in selecting samples for training. Contrary to a curriculum strategy, an active learner prefers a strategy of picking

examples close to the decision boundary, in order to reduce the number of examples necessary for learning a target concept.

Formally, training by a curriculum can be seen as gradually increasing the influence of difficult examples, and shares similarities to continuation methods [Bengio et al., 2009]. Let $P(z)$ be the target training distribution, and $W_i(z)$ be a weight applied to example z at step $1 \leq i \leq N$. The weight $W_i(z)$ is reweighted at each step, until $W_N(z) = 1$ for all examples. The training distribution $Q_i(z)$ at step i:

$$Q_i(z) \propto W_i(z)P(z)\forall z.$$

At each step, examples are reweighed, which changes the training distribution $Q_i(z)$. The weights are first increased on examples considered easy. At $i = N$ all weights $W_1(z)$ are set to one, and the target training distribution $P(z)$ is recovered. This process should iteratively increase the entropy of the distribution $Q_i(z)$. For instance, curriculum learning could be achieved by two steps $N = 2$. At step 1 the influence of harder examples h is eliminated by setting $W_1(h) = 0$. Whereas, in step 2 the target training distribution $P(z)$ is recovered by setting all weights $W_2(z) = 1$.

An experiment was conducted on the task of shape recognition, where images of geometrical shapes were classified. Two artificially generated datasets consisting of 32x32 grayscale images were constructed. The simpler dataset contained only squares, circles, and equilateral triangles, while the complex dataset was composed of all types of rectangles, circles, and triangles. Two neural networks were trained for 256 epochs by SGD to classify these shapes. The network trained using a curriculum strategy would start by training only on easier examples found in the simple dataset, and switch to the complex dataset after a certain number of epochs. The baseline network would only train using the complex dataset. The best generalization was obtained by the network using a curriculum, where half of the total epochs was spent on easier examples.

Another experiment was conducted on a language modeling task, where a learner predicts the most fitting word that can follow a given sentence. The curriculum strategy in this case, was to iteratively grow the allowed vocabulary. Only sentences in the dataset where all words were present in the vocabulary would be included in the training set. The network that utilized curriculum learning performed better than the baseline for this task as well.

A considerable challenge with curriculum learning is to define an appropriate curriculum strategy that can work well for a task. In this study, the strategies

were task specific, and not generally applicable. Furthermore, the tasks presented were fairly simple, and curriculum strategies were easy to identify. This may not be the case for datasets containing images where a measure of easiness can be harder to define.

Curriculum learning's main challenge is to find a sequence of samples that are meaningful, and can facilitate learning. This requires ordering criteria that can sort samples based on some measure of "easiness". The criteria presented by Bengio et al. [2009] were problem-specific. To address this issue, Kumar et al. [2010] proposed self-paced learning (SPL). Instead of a teacher providing the algorithm with a fixed curriculum, the algorithm iteratively selects samples based on its own abilities.

SPL is an iterative approach, where the learner both selects easy samples and updates its parameters at each iteration. The number of samples is determined by a weight that is gradually annealed. At the start of training, only easy samples are considered. In self-paced learning, easy samples are defined as samples that have labels that are easy to predict. The algorithm finishes when all samples have been considered by the learner, or at convergence.

Specifically, SPL integrates curriculum learning by modifying the loss function of the model. The model parameters w and binary variables v_i are simultaneously estimated by minimizing the modified loss function. The binary variables v_i, indicate whether the model considers sample i easy or not. A parameter K is gradually annealed to modify the learning pace by increasing the effect of a regularization term. As K is cooled, harder samples are included in order to minimize the loss.

The SPL approach was tested on several tasks, including hand-written recognition and object detection. The self-paced learning approach was implemented for a latent structural support vector machine. Predicting digits found in the MNIST dataset, self-paced learning performed significantly better on most runs. In the task of object detection, self-paced learning also produced better results.

According to Jiang et al. [2014], the self-paced learning approach is limited because it does not consider diversity in sample selection. They therefore, proposed an extension to SPL called self-paced learning with diversity (SPLD). In this approach, a new regularization term is introduced which encourages selecting diverse samples. SPLD was evaluated on different tasks, such as multimedia event detection and video action recognition, and compared against SPL and three other baseline methods. In the task of event detection, the SPLD method out-

performed both SPL, RandomForest, and AdaBoost. This was also the case for action recognition.

A human behavioral study was conducted by [Khan et al., 2011], in which participants were tasked with teaching a robot a target concept. The goal of the study was to explore what teaching strategies humans employ. Empirical results suggest that human teachers follow the principles of curriculum learning.

There are two prominent teaching models in computational teaching. The teaching dimension and the curriculum learning principle. The former is based on showing samples closest to the decision boundary, in order to minimize the number of samples needed to reveal a target concept. The latter suggests an easy-to-hard teaching strategy.

The experiment involved 31 participants, each tasked with teaching a robot the concept of graspability. The robot would not learn anything but followed motions with its gaze. This provided a consistent environment for the trials. Participants were first asked to sort images of common objects based on how easy they are to hold with one hand. The images were placed along a ruler. The participants then assigned a binary value indicating whether an object is graspable or not to each object. Finally, the participants would act as teachers by showing the robot images and giving a description about the depicted object's graspability. The participants could choose any order, and use as few images that they felt were needed to teach the robot the target concept. The task represents a simple one-dimensional machine learning problem of binary classification, where participants assigned an x and y value to each example.

Based on the ordering each participant chose, three major human teaching strategies were observed. A large percentage of the participants employed the curriculum learning approach, by gradually presenting samples approaching the decision boundary. 20 of the participants started by showing the most graspable object, and 6 started with the least graspable. None of the subjects started by showing samples close to the decision boundary, which the teaching dimension model suggests.

The experiment showed that there is evidence of curriculum learning being employed by humans in a teacher role. This might indicate that this is an intuitive and efficient way to learn, and might be beneficial in machine learning as well.

2.4 Background Discussion

This chapter has given a brief introduction to several topics. This includes road extraction systems based on machine learning, convolutional neural networks, curriculum learning and label noise. In this section, these topics will be summarized.

Manual labeling of aerial imagery for map purposes is a laborious task. Furthermore, the surface depicted in aerial imagery is always changing. These changes should preferable be reflected in the map data in a timely manner. Automatic object extraction has therefore become a compelling area of research.

The increasing spatial detail of aerial imagery is reflected in the choice of learning algorithm. The latest iteration consists of deep neural networks, which are capable of learning complex decision boundaries, and extract suitable image features by learning. This is necessary, since the decreasing GSD has lead to more features being distinguishable from aerial imagery and increased the complexity of the data. The use of learning algorithms with high model capacity have also resulted in the usage of larger datasets. Another trend is the increasing context window. For instance, Mnih and Hinton [2010] used a pixel neighborhood of 64×64 to distinguish road from non-road pixels in a 16×16 center area. There are also many examples of systems that have combined various input features to increase accuracy.

Furthermore, CRF and post-processing neural networks have been used to increase the performance of various systems. In structured output learning, such as semantic segmentation, smoothness between neighboring predictions is an important consideration when performing a label assignment. For road extraction, these methods can reduce prediction artifacts such as disconnected roads, and can result in a more even segmentation.

Training deep neural networks to extract roads from images require large datasets. The labels in these datasets are often automatically generated from existing map data. Unfortunately, these labels are often affected by label noise, which can affect the performance of supervised learning. The types of label noise in aerial imagery have been identified by Mnih and Hinton [2012] as registration and omission noise.

The first research question involves reducing the effect of inconsistent labels when training a classifier. The bootstrapping approach presented by Reed et al. [2014] was, therefore, evaluated for road detection in Chapter 4. Whereas the loss func-

tions presented by Mnih and Hinton [2012] model the noise distribution, bootstrapping utilizes a convex combination of the classifier's prediction and the label. Furthermore, the method was tested on several datasets, and achieved good results. To test this method for road extraction, the robustness of this method can be evaluated on aerial image datasets with increasing levels of artificial omission and registration noise. Similar experiments of increasing noise levels have been conducted in [Sukhbaatar and Fergus, 2014] and [Reed et al., 2014].

The studies involving curriculum learning demonstrated that a curriculum strategy which gradually introduce "harder" examples while training, resulted in an improved generalization accuracy and faster convergence. However, the curriculum strategies that Bengio et al. [2009] used to estimate the difficulty of each example were domain specific. This was rectified by SPL [Kumar et al., 2010], where curriculum learning is internalized in the classifier. The model simultaneously estimates the difficulty of the examples and the loss. Jiang et al. [2014] extended this method by also considering the diversity of the examples. The curriculum learning approach is also compliant with how humans prefer to teach, as demonstrated by Khan et al. [2011].

The second research question, therefore, investigates what effect a curriculum strategy can have on the road extraction system's precision and recall. Aerial image datasets often contain label noise, which could be considered harder in the context of curriculum learning. The learner would most likely create unnecessarily complex decision boundaries to fit the inconsistent examples. Postponing the introduction of these examples could be beneficial for a road extraction system. An interesting curriculum strategy, mentioned by Bengio et al. [2009], is creating an easy-to-hard ordering of samples according to how noisy the examples are. A potentially valid curriculum strategy is to estimate the degree of noise in examples, by measuring disagreement between labels and predictions.

Chapter 3

Methods

This chapter will describe the architecture and the methods that were needed in order to investigate the research questions presented in Section 1.2. An overview of the system and its components can be found in Section 3.1. Section 3.2 presents the architecture, regularization, optimization, and data preprocessing utilized by the convolutional neural network. Furthermore, the methods specifically related to the research questions can be found in Section 3.3 and Section 3.4. These sections outline curriculum learning and bootstrapping loss in detail. The datasets used are described in Section 3.5, and implementation details can be found in Section 3.6.

3.1 System Overview

The objective of the system is to detect and segment roads found in aerial color images. A supervised learning algorithm that works well for images is a CNN, and it has therefore been chosen for this task. CNNs have been applied to many computer vision tasks lately, with results outperforming other approaches [Krizhevsky et al., 2012]. As discussed in Section 2.1, these networks reduce the need for feature engineering by having the network learn suitable feature detectors, which are a necessity for tasks involving images. Additionally, an important part of creating a competent road detection system from supervised learning is by having a large dataset containing aerial images and label images showing exactly which pixels represent roads.

The system shares many similarities with the patch-based deep neural network presented by Mnih and Hinton [2012]. Their system produces a patch of predictions given a patch of pixel values, where each value indicate the probability

Figure 3.1: Output prediction patch superimposed on input aerial image patch.

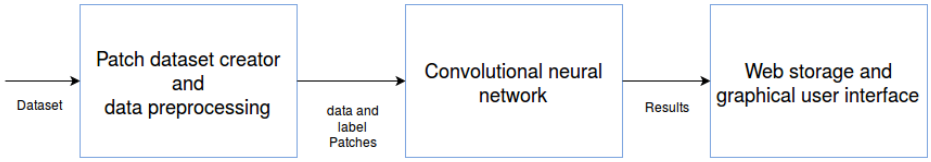


Figure 3.2: Components of the system.

of the input pixel representing a road. Given a 64×64 aerial image patch, the system computes a 16×16 patch of probabilities. These probabilities indicate the presence of road for the pixels found in the center of the aerial image patch. An example of an input image patch and an output prediction patch can be seen in Figure 3.1.

The actual implementation consists of two primary components, and an optional storage and graphical user interface web server component. The components, and how they are related, can be seen in Figure 3.2. The storage and user interface component is not integral to the task of road detection, but has been a helpful aid when conducting experiments. Additionally, the patch dataset creator component is interchangeable, and can easily accommodate other varieties of segmentation datasets. In relation to the research questions, the patch dataset creator component is altered when investigating curriculum learning, and the loss function of the CNN component is replaced when bootstrapping is tested.

3.2 Semantic Segmentation with CNN

This section presents the architecture of the convolutional neural network, as well as the details surrounding the optimization and regularization of the network.

3.2.1 Patch-based Approach

This thesis formulates the problem of road segmentation in aerial images in the same manner as Mnih and Hinton [2010] did. This patch-based approach is presented in detail in Section 2.3. In short, the convolutional neural network should learn the label distribution:

$$p(m|s) = \prod_{i=0}^{w_m^2} p(m_i|s),$$

where $p(m|s)$ denote a conditional probability of a label map patch given an aerial image patch. The patches m and s have been extracted from label image M and aerial image S. Therefore, the patches can be defined as $m = N(M_{i,j}, w_m)$, and $s = N(S_{i,j}, w_s)$, where an area of $w_m \times w_m$ and $w_s \times w_s$ centered at location (i, j) have been extracted from M and S. By having a label patch m, the model can simultaneously make several predictions from the same context s, which is more effective than making a separate prediction per aerial image patch. Furthermore, limiting the size of each example by creating patch examples (s, m), reduces the number of operations needed to convolve the input and subsequent layers. This is achieved without losing too much of the image context.

3.2.2 Network Architecture

The network is based on the deep neural network outlined by Mnih [2013]. See Section 2.1.1 for background theory about CNN. The network has three convolutional layers and two fully connected layers. It predicts whether or not roads are present in a 16×16 pixel area from the center of a 64×64 aerial image patch. The input patch is considerably larger than the output patch, so that the network can utilize the surrounding image context when making predictions. The default network architecture used for experiments is depicted in Figure 2.2.

The first layer convolves the input using 13×13 kernels. Only the first layer utilizes stride and max pooling. The first controls how the kernel should move spatially across the input. A stride of 2×2 makes the kernel perform convolution for every second pixel, making the receptive fields overlap less. As a result, this decreases the number of connections between the input and the first layer. The second reduces the number of inputs to the next layer, as well as introducing some translational invariance in the model. In the default configuration, a stride of 4×4 and max pooling of 2×2 are used. In addition, the pooling step uses a stride of 2×2, which results in non-overlapping pooling regions. The default kernel size in the second and third layer is 4×4 and 3×3, respectably. The

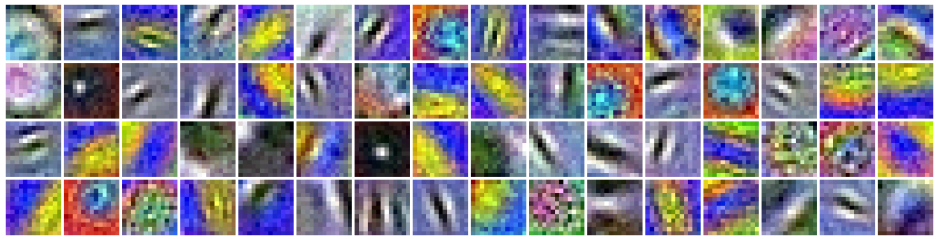


Figure 3.3: Visualization of feature maps from the first convolutional layer of a trained network.

output of the third convolutional layer is used as input to a fully connected hidden layer with 4096 units. Finally, the fully connected output layer contains 256 units, where each output is the probability of a pixel representing a road.

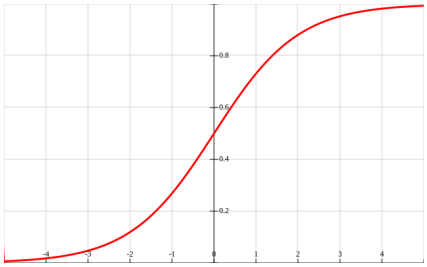
The convolutional layers each have 64, 112 and 80 different feature maps, respectably. During training, these feature maps or kernels typically learn to respond to common local patterns in the input. The kernels in the first convolutional layer, for example, will learn to detect low-level features in the aerial image, such as edges, colors, and textures. This happens because the CNN employ parameter sharing by convolving the same kernel across the input data, forcing the model to find local image features that are present throughout the image. A visualization of the 64 feature maps from the first layer of a trained model can be viewed in Figure 3.3. Many of the kernels seem to have developed into something similar to Gabor filters, that are effective at detecting edges at certain orientations. Coincidentally, oriented two-dimensional Gabor functions are often used to model the response from simple-cell receptive fields in the primary visual cortex [Ringach, 2002].

Common for both convolutional layers and fully connected layers are the use of non-linear activation functions, which allow the network to learn non-linear decision boundaries. To compute the outgoing activation of any unit in the network, the incoming weights are first summed with the activations coming from either the input or the previous layer. Then, a bias is added, before feeding the result through the activation function. This is typically denoted:

$$y = \sigma(aw + b),$$

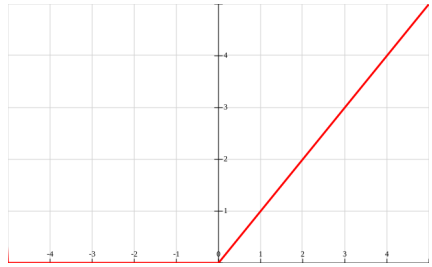
where a and w are the unit's incoming activations and weights, b is the bias and σ is the non-linear activation function.

In the default hyperparameter configuration of this system, the output layer applies the logistic activation function, while the ReLU activation function is used by the input and hidden layers. The logistic activation function is appropriate for the output layer, because it squashes any input to be between the value of 0 and 1. This is useful for the road detection system, where the activation of each output unit should predict a probability of whether the corresponding aerial image patch pixel belongs to the road class or not. The ReLU, however, has been used for all other layers. This activation function has been shown to train deep neural networks faster. This non-linear activation function does not suffer from the gradient vanishing problem because it is non-saturating [Krizhevsky et al., 2012]. The activation functions are displayed in Figure 3.4.



$$y = \frac{1}{1 + e^{-x}}$$

(a) Logistic.



$$y = \max(0, x)$$

(b) Rectified linear unit.

Figure 3.4: Activation functions.

The specific network configuration values described in this section is the default configuration used in experiments found in Chapter 4. However, the network architecture can easily be changed through a configuration file. All configurable parameters related to the CNN and their default values are listed in Table 3.1.

3.2.3 Optimization

The model parameters are optimized with a special form of SGD, called Nesterov's accelerated gradient descent, and has an improved stability and faster convergence compared to standard SGD [Bengio et al., 2013]. The standard momentum approach extends the gradient update step by introducing velocity. The

Table 3.1: Hyperparameters for the CNN.

Parameter	**Description**	**Value**
$K^{(l)}$	Number of kernels for convolutional layer l	64, 112, 80
$CLF^{(l)}$	Filter size of convolutional layer l	(13,13),(4,4),(3,3)
$CLS^{(l)}$	Stride size of convolutional layer l	(4,4),(1,1),(1,1)
$CLP^{(l)}$	Pool size of convolutional layer l	(2,2),(1,1),(1,1)
$\mathcal{L}$	Loss function	Cross-entropy
Epochs	How many epochs of training	100
$\sigma^{(l)}$	Activation function for each layer l	ReLU x4, sigmoid x1
h	Number of neurons in hidden layer	4096
$p^{(l)}$	Dropout rate for layer l	1.0, 0.9, 0.8, 0.5, 1.0
ES_{init}	Initial early stopping patience in iterations	100 000
ES_{inc}	Increase factor for patience	2
ES_{thres}	Necessary improvement of validation loss before increasing patience	0.997
a	Learning rate	0.0014
$a_{decrease}$	learning rate decrease factor	0.95
λ	L2 weight decay strength	0.0001
m	Momentum	0.9
b	Batch size	64

loss can be interpreted as a hilly error surface, where loss optimization can be viewed as a ball gaining and losing velocity while interacting with the landscape. When the loss optimization has gained velocity, it will stop doing purely steepest decent, which results in a smoother decent with fewer oscillations. Basically, instead of directly using the gradient to move in the landscape, the gradient will influence the velocity. The only hyperparameter associated with momentum is the decay coefficient m, which damps the velocity.

Nesterov momentum is a variation of momentum. Unlike standard momentum, the Nesterov method first makes a step in the direction of the accumulated velocity, and then makes a correction of the velocity based on the new location. In standard momentum, a correction to the velocity is made before taking a step. The Nesterov gradient is defined by the following update rule:

$$v_t = mv_{t-1} - a\nabla f(\theta_{t-1} + mv_{t-1})$$
$$\theta_t = \theta_{t-1} + v_t,$$

where θ_t is the model parameters, v_t the velocity, m the momentum decay, a the current learning rate, and $\nabla f(z)$ is the gradient.

An important hyperparameter for gradient descent optimization methods is the learning rate a. In this implementation, the learning rate is annealed over time by step decay. The learning rate is decreased in a regular interval of epochs by some factor $a_{decrease}$ during optimization.

The default loss function of the system is cross-entropy loss, which in the context of the patch-based approach is denoted:

$$\mathcal{L}_{\text{cross-entropy}}(q, y) = -\frac{1}{w_m^2}\sum_{i=1}^{w_m^2}[y_i \log(q_i) + (1 - y_i)\log(1 - q_i)],$$

where q and y are the prediction and label patch. Each of the patches has w_m^2 probabilities and label values, that are used for computing the element-wise cross-entropy. Cross-entropy loss is non-negative and moves towards 0 as the network improves at the task. Given a training label and a predicted output, the loss function measures how well a neural network fits the training data. In addition, the backpropagation computes gradients based on the output of this loss function. At each iteration of training, the loss is minimized by slightly adjusting the weights of the network in the direction of these gradients.

3.2.4 Regularization

To avoid overfitting the training data, and hopefully achieve better generalization, different regularization schemes are applied during optimization, such as L2 weight decay, early stopping, and dropout. The classifier's task is to learn the regularities found in the mappings between the data and labels in the training set. However, the training set also contains sampling errors and accidental patterns not relevant to the task at hand. Without regularization, a classifier will learn the useful, but also the erroneous regularities a bit too well, and can, therefore, start to overfit the data.

L2 regularization is a weight decay method which applies a loss function penalty to prevent weights in the network from growing large. This is achieved by adding an extra term, $\lambda \sum_{i=0}^{|w|} w_i^2$ to the loss function, which penalizes large valued weights, and encourages a smoother parameter configuration [Hinton, 2014]. The strength of the weight decay is controlled by the λ hyperparameter, which is often set to a low value.

Early stopping interrupts the optimization process when performance on the validation set starts to consistently decrease. When the validation loss stagnates or starts to increase in value, the method waits for a certain number of iterations before stopping the training process. There are three parameters that control the early stopping behavior, ES_{init}, ES_{thres}, and ES_{inc}. The first parameter controls how many iterations the training should run for, regardless of validation loss performance. The second parameter dictates the required amount of improvement of the current validation loss compared to the best previously seen validation loss, before incrementing the patience variable. The latter parameter decides the increment factor of the patience. The patience is first initialized to wait for ES_{init} iterations. If there is an improvement of validation loss above the ES_{thres}, the patience is set by $patience = max(patience, iteration \times ES_{inc})$.

The dropout method forces the units to rely less on each other by randomly disabling units in the network during training. This encourages units to encode independently useful information, since dropout penalizes co-adaptation between units [Srivastava et al., 2014]. Dropout can also be viewed as an ensemble method approximation. By randomly removing subsets of units from the network, an exponential number of parameter sharing subnetworks are trained simultaneously. This is less computationally expensive than training an ensemble of many individual deep neural networks. At test time, the predictions of these subnetworks are combined by an approximate averaging method. This is done by having the neural network produce predictions without dropping any units. Because all units make a contribution at test time, the weights of the network have to be scaled

down. If a unit is retained with a probability of p during training, the outgoing weights of that unit should be multiplied by p at test time. Each layer l has a unique dropout rate parameter $p^{(l)}$, which controls the probability of dropping a unit at that layer. In the CNN implementation, dropout has been implemented with some minor tweaks:

$$y = \frac{1}{p^{(l)}} r \circ \sigma(W^{(l)} x + b^{(l)}),$$

where W and b are the weights and biases of layer l, x denotes the input potentials, and $\sigma(z)$ is the activation function. r is a vector of independent Bernoulli random variables, each having a probability of $p^{(l)}$ being 1 (and otherwise 0). The element-wise multiplication of this vector and the output vector of $\sigma(z)$, causes the activation of randomly picked units to be nullified. These units are essentially dropped out. Instead of scaling the outgoing weights at test time, the output activations are divided by $p^{(l)}$ during training, which should essentially give the same scaling effect.

3.2.5 Data Preprocessing

The datasets used in this thesis contains aerial images and road label maps that are very large. Each image is typically around 1500×1500 pixels in size, which is arguably too large for the model to use directly as input. Smaller patches of examples are therefore extracted from these larger images. From each image, a certain number of data patches and label patches of 64×64 and 16×16 are extracted randomly and added to a patch dataset.

However, before extracting patches, each aerial image and road label map pair is rotated by a random number of degrees between 0 and 360 degrees. This is necessary because of orientation bias in the datasets. In many cities, the road network is often constructed in a grid pattern, which leads to roads having certain orientations more often than not. Without random rotations of the training data, the model might become better at detecting roads only at certain orientations [Mnih and Hinton, 2010]. Each image is therefore rotated by a random angle before extracting patches, which results in a patch training set that encourages the model to learn invariance to road orientation.

Another data augmentation method utilized when constructing the patch training set is flipping. This is a commonly used method to artificially increase the size of the dataset [Krizhevsky et al., 2012]. Since the dataset contains aerial

images that depict landscape from a top-down perspective, the patches can both be flipped vertically and horizontally.

Contrast normalization is also applied to every patch. Each patch is normalized by subtracting the mean pixel value from the pixels of the patch. The result is then divided by the standard deviation found for all pixels in the dataset.

Additionally, the datasets exhibit some class imbalance [Japkowicz, 2000], which can be problematic since the optimization might prioritize loss minimization of classes that frequently occur in the training set and ignore rare classes. Random sampling of the patch datasets leads to a large imbalance between road examples and non-road examples. According to the label maps, the proportion of patches containing any road pixels, ranges between around 8% and 25%, depending on the dataset. This is solved by artificially increasing the proportion of road examples when sampling the patch dataset. Normally, a distribution of around 50% road patches and 50% non-road patches are used. However, this method is not applied to the test and validation patch set, which means that these sets will retain their natural class distribution. In Table 3.2 the percentage of road pixels for each dataset is listed.

Table 3.2: Percentage of road pixels in the datasets used by this thesis.

Set	**Massachusetts**	**Norwegian Vbase**	**Norwegian N50**
Test	4.70	4.60	2.76
Validation	6.90	4.06	2.45
Training	4.77	4.60	2.78

3.3 Bootstrapping Loss

Normally, the loss function calculates the loss assuming the labels are correct. Yet, many datasets contain label noise, which will result in the learner being penalized for making a correct prediction if the label happens to be inconsistent. According to Reed et al. [2014], tweaking the loss function by incorporating the network's own predictions can yield improved robustness to inconsistent labeling. Whereas other approaches [Mnih and Hinton, 2012][Sukhbaatar and Fergus, 2014]

explicitly model the noise distribution, the proposed loss function utilizes the implicit knowledge acquired by the network during optimization. The bootstrapping method computes the loss by a convex combination of the model prediction q and the label y. Additionally, this thesis proposes a variation of the bootstrapping loss function, which only incorporates confident predictions.

The bootstrapping loss function uses the current model prediction q and the noisy training label y in a convex combination, which results in a modified target. Apart from that, the loss function is similar to the cross-entropy loss function. The prediction's contribution to the convex combination is decided by the β parameter. The bootstrapping loss for the aerial road detection system is defined below:

$$\begin{aligned}\mathcal{L}_{\text{hard}}(q, y) = &- \sum_{i=1}^{w_m^2} [\beta y_i + (1-\beta)\mathbb{1}_{q_i>0.5}] \log(q_i) \\ &- \sum_{i=1}^{w_m^2} [\beta(1-y_i) + (1-\beta)(1-\mathbb{1}_{q_i>0.5})] \log(1-q_i),\end{aligned}$$

where $\mathbb{1}_{q_i>0.5} = 1$ if the road prediction probability q_i for pixel i is above 0.5. For the task of road detection, it is simply a threshold operation. In multi-class classification, $\mathbb{1}_{q_i>0.5}$ is replaced by the MAP estimate, as described in Section 2.3. This is also the reason why this loss function is denoted hard. A soft version of bootstrapping, where the predictions q_i are used directly, has also been tested by Reed et al. [2014], but generally performed worse than the bootstrapping hard variant.

Bootstrapping loss combined with gradient descent results in an EM-like algorithm. In the E-step, the modified targets are generated, whereas in the M-step, the network weights are adjusted to better predict the modified targets. The goal is for the learner to rely less on the inconsistent labels, and develop more consistent implicit knowledge, which in turn further improves the quality of the modified targets.

Since the task involves the binary task of discriminating road pixels from non-road pixels, a slightly modified version of bootstrapping is also tested in this thesis. This approach is named confident bootstrapping, since all model predictions between 0.2 and 0.8 are ignored. Only predictions that the learner has a high confidence in are allowed to contribute in the convex combination. An added benefit of this loss function is the possibility of increasing the factor β. The confident bootstrapping loss function is denoted:

$$\mathcal{L}_{\text{confident}}(q, y) = -\sum_{i=1}^{w_m^2} [\beta y_i + (1-\beta)\mathbb{1}_{q_i>0.8}] \log(q_i) \\ -\sum_{i=1}^{w_m^2} [\beta(1-y_i) + (1-\beta)(\mathbb{1}_{q_i<0.2})] \log(1-q_i),$$

where $\mathbb{1}_{q_i>0.8}$ and $\mathbb{1}_{q_i<0.2}$ are threshold operations which only keep fairly confident predictions. In the context of road detection, this translates to saying pixel predictions above 0.8 are most likely road pixels, and predictions below 0.2 are likely to be non-road pixels.

In the actual implementation, the β parameter is annealed from the max value β_{max}, down to the minimum value of β_{min}, starting from epoch M. This is because the learner should have some implicit knowledge before using its predictions to modify the target. The configurable parameters for bootstrapping can be found in Table 3.3.

Table 3.3: Hyperparameters for bootstrapping loss.

Parameter	**Description**	**Value**
Cost function	Modified cost function	bootstrapping or confident boostrapping
β_{max}	Mix factor	1.0
β_{min}	Minimum mix factor	0.90
$\beta_{decrease}$	Factor decrease rate	0.90
M	When to start mixing	60

3.4 Curriculum Learning

The curriculum learning method presented by Bengio et al. [2009] showed that presenting the dataset in a certain order yielded better generalization as well as faster convergence. Similar to an educational curriculum, easier concepts are

taught before harder concepts, which most likely is the preferred way of learning for humans [Khan et al., 2011]. This translates to having the learning algorithm do optimization on a dataset consisting of "easier" examples, before introducing "harder" examples. Continuing this analogy, there is a teacher who decides the curriculum by judging how easy concepts are to grasp. For some datasets, such as language modeling datasets, the role of the teacher can easily be performed by a human. In language modeling tasks, a viable curriculum strategy is to start training using a simple dataset containing sentences with only a limited vocabulary. During training, the vocabulary is gradually increased, which will allow more complex sentences to enter the training set. However, similar curriculum strategies are harder to identify in datasets involving images. The solution is therefore to outsource the job of curriculum teacher to a supervised learning algorithm.

This thesis utilizes aerial image datasets. Based purely on color image patches, a supervised learning algorithm should be able to detect and segment roads found in these patches. A curriculum strategy involving this type of data is harder to identify. For instance, are there certain types of road segments a machine learning algorithm would find less challenging? Are roads in rural scenes easier to learn than roads in an urban setting? What configurations of pixel intensities are easiest? It is hard for a human to identify features in images that a machine learning algorithm would consider easier to learn.

Hence, a classifier is first trained on the available examples and given the role of the teacher. The classifier, regardless of competence, will generate predictions that are used in a difficulty estimation of each example. The difficulty estimation is simply based on the amount of disagreement between the teacher's prediction and the example's label. The difficulty estimator is defined below:

$$d(y, q) = \frac{1}{w_m^2} \sum_{i=0}^{w_m^2} |y_i - \mathbb{1}_{q_i > r}|,$$

where q is the curriculum teacher's prediction, and y is the label. The $\mathbb{1}_{q_i > r}$ is a threshold operator, where r is the threshold value which gives the best precision and recall breakeven for the curriculum teacher classifier. The operator essentially creates a binary image patch from the prediction probabilities. The estimator computes the average difference value from the $w_m \times w_m$ patch of label pixels and prediction probabilities. Even though the estimator has been customized

to the patch-based approach of aerial road detection, the method can easily be adapted to other types of datasets. This curriculum strategy can probably be considered generally applicable. The disagreement between a prediction created by a trained classifier and a label can probably be computed for most datasets.

An interesting thing to note is that a very competent teacher classifier, computing a high difficulty score for an example, might indicate that the example's label is inconsistent.

The implementation of curriculum learning involves N training set stages. Each stage θ only contains examples where $d(y, q) < D_\theta$, in which the difficulty threshold $D_\theta \geq D_{\theta-1}$ for all stages $\theta > \theta-1$. Each subsequent stage, therefore, contains examples with a wider range of difficulty estimates. In the final stage N, the threshold D_N is typically set to 1, which results in every example being added to the patch dataset. This stage has a training distribution equal to what a patch dataset created from random sampling would have had.

The supervised learning algorithm is gradually exposed to harder examples by stage switching. The classifier is first optimized with the examples in the first stage, which has the lowest difficulty threshold. At epoch t_{start}, the next stage replaces the existing training set by assigning each new example to a random index in the training set. The indices are sampled without replacement. Following stages are mixed into the training set every t_{stage} epoch, until the model has trained with examples from all stages. The hyperparameters and default values used by curriculum learning are listed in Table 3.4.

Table 3.4: Hyperparameters for curriculum learning.

Parameter	**Description**	**Value**
N	Number of stages	2
D_θ	Threshold value for difficulty estimate of stage θ	0.25, 1.0
t_{start}	What epoch to load stage 1	50
t_{stage}	Load subsequent stage after every t_{stage} epoch	50

3.5 Datasets

When testing the bootstrapping loss function and curriculum learning, as well as measuring the performance of the road detection system, two datasets have been utilized. Both datasets involve the task of semantic segmentation of images, more precisely, the task of extracting roads from aerial images. The first dataset, the Massachusetts Roads Dataset, provided by Mnih [2013], has been used in several works. The second dataset, the Norwegian Roads Dataset, was specifically made for this thesis, and was generated from publicly available data. Differences, similarities, and challenges of employing these datasets will be further discussed below.

3.5.1 Massachusetts Roads Dataset

The Massachusetts Roads Dataset contains aerial images depicting urban, suburban, and rural areas in the state of Massachusetts, USA. In all, the dataset consists of 1171 aerial images, where each image is 1500×1500 pixels in size. 1108 of these images have been randomly assigned to the training set. The remaining 49 and 14 images can be found in the test and validation set. The dataset covers an area of approximately 2600 square kilometers in total, which gives a GSD of 1.0 meter per pixel.

Each aerial image has an accompanying identically sized binary label image, which indicates whether a pixel in the aerial image belongs to either the road or non-road class. Road centerline vectors retrieved from the OpenStreetMap project were used to generate the label images. The vectors were rasterized as white lines with a line thickness of 7 pixels [Mnih, 2013], which, based on the GSD, is equivalent to 7 meters on the ground. An aerial image and label image pair from this dataset can be seen in Figure 3.5.

There are many reasons for using this dataset to conduct experiments related to the research questions. The task involves semantic segmentation of roads based exclusively on aerial color images. This is arguably a hard task, and might justify the use of a large model for training, such as a CNN. The dataset has also been used in other works, which enables comparison of the results obtained in this thesis to the results obtained in other works. Additionally, the labels have been generated from existing map data, and therefore contain many instances of naturally occurring inconsistent labeling. This is compelling in relation to the research questions of this thesis.

(a) Aerial image. (b) Label image.

Figure 3.5: Image and label example taken from test set in the Massachusetts Roads Dataset.

Aerial image datasets typically suffer from two distinct types of label noise, which Mnih and Hinton [2012] have named omission and registration noise. The dataset is generated from existing map data, which can have some deficiencies when coupled with supervised learning. There are many instances of omission noise in this dataset, where smaller roads and parking areas have not been marked as road class pixels in the label images. These omissions are most likely perfectly acceptable for map purposes, but might negatively impact the performance of a classifier. For instance, unlabeled paved areas that share a high spectral similarity to roads could be considered inconsistently labeled. These label errors might compel the classifier to minimize the loss by learning complicated distinctions between surfaces that are essentially the same thing.

There are also instances of registration noise in this dataset. This happens when there are misalignments between the roads found in the aerial images and the road centerline vectors from the map data. Additionally, the road centerline vectors have, in many cases, been rasterized with the incorrect line thickness, which might cover too much or not enough depending on the road's lane width. The result is a lot of aerial image pixels that have been assigned the wrong class, which might impact the learning procedure negatively. Slightly misaligned road centerline vectors probably do not affect map products much, whereas they create a lot of inconsistent examples in a machine learning dataset.

3.5.2 Norwegian Roads Dataset

In addition to the Massachusetts Roads Dataset, the proposed methods have been tested with the Norwegian Roads Dataset. This dataset was constructed from aerial images retrieved from Kartverket, which depicts both rural, suburban, and urban areas from different locations in Norway. The entire dataset consists of 1225 aerial images, each being 1536×1536 pixels in size. 1100 of them have been randomly assigned to the training set, 75 to the test set, and the remaining images were put in the validation set. Even though there are more aerial images in this dataset compared to the Massachusetts Roads Dataset, it only covers an area of around 1910 square kilometers. This is due to a much lower GSD of about 0.66 meters per pixel.

The label images in this dataset have been generated from road centerline vectors found in the publicly available topographic vector database, N50, provided by Kartverket [2001]. Unlike the Massachusetts Roads Dataset, the centerline vectors have been rasterized with a variable line thickness. This is possible because all road segments in N50 have a set of properties, which can be utilized to determine a custom line thickness. The actual thickness of each road type has been based on numbers found in a road specification manual published by the Norwegian Public Roads Administration [Vegdirektoratet, 2014]. The road segment properties and line thickness for each road type are listed in Table 3.5. Roads that are underground and cannot be seen from aerial imagery have also been removed from the rasterized label images.

The aerial images have been taken from over 30 different locations in Norway, and offers a large variety of topographical features. There are images depicting coastlines, rivers, mountain terrains, snow, cultivated land, and forests. Compared to Massachusetts Roads Dataset, the aerial images of this dataset have been sampled from a much larger area. Furthermore, the aerial images might present a challenge in terms of image quality. Some of the images vary in terms of color balance and image contrast. There are also aerial images that have been stitched together from images captured by aerial surveys conducted at different occasions.

The quality of the generated label maps might also present a challenge to a machine learning algorithm. There are both omission and registration errors present in many of the label images. Compared to the Massachusetts Roads Dataset there is a much higher degree of registration noise. The road centerlines in N50 generally appear more coarse, and result in less overlap between roads in the aerial image and the raster lines in the label image. This is especially evident in road centerline vectors for divided highways. Instead of having centerline vectors for both roadways, there is only one placed between the roadways on the median.

Visual examples of missing and misplaced labels can be seen in Figure 3.6.

(a) Divided highways. (b) Roundabout. (c) Private road.

Figure 3.6: Examples of inconsistent labeling found in the Norwegian Roads Dataset N50.

In addition to the set of label images generated from N50, the dataset also includes an alternate set of label images generated from the road centerline vector database, Vbase [Kartverket, 2006]. This database has more accurate road centerline vectors. There are still omission and registration errors present in this label image set, but to a less extent. Surfaces that share spectral similarities to asphalt, such as private roads and parking areas, have not been marked, similar to the Massachusetts Roads Dataset. The difference in accuracy between N50 and Vbase can be seen by comparing Figure 3.6 and Figure 3.7. A minor downside of using this alternate vector database is that Vbase provides fewer road segment properties, which have resulted in a smaller set of line thicknesses applied to the label images.

The dataset was constructed by using QGIS, an open source geographic information system application. The application enables viewing and editing of map data, but also provides a Python interface. A script to create label images was developed, taking the map coordinates associated with each corner of an aerial image, and generating a raster image of road centerline vectors found inside that area. The resulting raster images can be used as target maps in supervised learning.

An aerial image and its corresponding label image from the Norwegian Roads Dataset N50 can be seen in Figure 3.8. Figure 3.8c shows the same label image superimposed on the aerial image. Observe that some roads are missing from the label image, as well as the ground truth not covering the roads properly.

(a) Divided highways. (b) Roundabout. (c) Private road.

Figure 3.7: Examples of road centerline vector quality in Vbase.

3.6 Implementation Details

The road detection system was written in the Python programming language, and uses the open source library Theano [Bergstra et al., 2010]. Theano enables the developer to define and evaluate mathematical expressions involving tensors. The library implements several useful features to develop CNNs, such as back-propagation, convolution, and max pooling. Training deep neural networks on a GPU can be considerably faster than on a central processing unit (CPU), and Theano can utilize both the CPU and GPU without making any modifications to the code.

However, the use of a GPU was key in making the experiments feasible to run. The system has a lot of adjustable parameters and requires a big dataset in order to generalize well to the task of road detection. The experiments were conducted on a machine with an Nvidia GTX 980 GPU with 4 gigabytes of video memory.

Because the size of the training set in many cases exceeded the capacity of the video memory, the video memory only contains the model, validation set, test set, and a subset of the training set at any given time during training. A switching mechanism was implemented, where chunks of the training set residing in main memory were loaded onto the GPU sequentially during an epoch of training.

The system also has a large number of hyperparameters which can be changed by the user. This includes learning rate, network architecture, backpropagation method, dropout rates, curriculum and bootstrapping specific parameters, and

Table 3.5: Raster line thickness and road segment filtering rule for each type of road. A margin of 10% is removed from the line thicknesses, which are based on numbers found in the road specification manual.

Road type	**Road segment property**	**Line thickness**
Dirt road	OBJTYPE=Traktorveg	2.50 m
Trail	OBJTYPE=Sti	1.50 m
pedestrian road	OBJTYPE=GangSykkelveg	2.25 m
Highway	MOTORVEGTYPE=motorveg	10.80 m
International E-road network	VEGKATEGORI=E	6.30 m
Norwegian national road	VEGKATEGORI=R	5.85 m
municipal road	VEGKATEGORI=K	4.95 m
Private road	VEGKATEGORI=P	3.50 m

(a) Aerial image (b) Label image (c) Overlay image

Figure 3.8: Example from test set in Norwegian Roads Dataset N50.

more. These values can be accessed and modified in the system's config.py file.

The code for the road detection system, as well as the machine learning monitoring interface, are publicly available at `https://github.com/olavvatne/CNN` and `https://github.com/olavvatne/ml-monitor`. The tools used to create the Norwegian Roads Dataset can be found at `https://github.com/olavvatne/MapDataset`. More details about these projects can be found in Appendix A. Additionally, all experiments conducted for this thesis can be examined at `http://www.interface.ml/experiments`. All the projects above are licensed under the *MIT* license.

Chapter 4

Experiments and Analysis

In the following sections, all experiments conducted to resolve the research questions will be presented and analyzed. Section 4.1 outlines the experimental design, and Section 4.2 presents the parameter configuration of each experiment. In Section 4.3 the results are presented. The results are then further discussed in Section 4.4.

4.1 Experimental Design

To resolve the research questions and test the performance of the road detection system, a number of experiments were planned. Experiments related to the first research question included testing the robustness of the bootstrapping loss function compared to a baseline loss function. For the other research question regarding the performance of curriculum learning, the results from a network trained according to a curriculum strategy were compared with a baseline network. The baseline network was trained using an ordinary dataset with no particular example ordering. In addition, tests measuring the performance of the road detection system were conducted. The results of these tests were then compared to other works.

Throughout the rest of this chapter, the experiments will be referred to by their assigned ID. An overview of all experiments and their assigned IDs, can be found in Table 4.1.

Each experiment found in Table 4.1 has been replicated and measured 10 times. The experiments have many sources of variability, and averaging the measurements improves the reliability of the results. The network, for instance, is ini-

Table 4.1: Experiments overview.

ID	RQ	Dataset	Description
E1	RQ1	Massachusetts Roads Dataset	Bootstrapping versus baseline at different levels of label noise.
E2	RQ1	Norwegian Roads Dataset Vbase	Bootstrapping versus baseline at different levels of label noise.
E3	RQ1	Norwegian Roads Dataset N50/Vbase	Performance of bootstrapping with a label set with large amounts of label noise.
E4	RQ2	Massachusetts Roads Dataset	Curriculum, baseline and anti-curriculum comparison.
E5	RQ2	Norwegian Roads Dataset Vbase	Performance of curriculum learning at different thresholds D_θ.
E6	RQ2	Massachusetts Roads Dataset	Curriculum learning with an inexperienced teacher.
E7	-	Massachusetts Roads Dataset	Best performing road detection network. Larger patch datasets and increased model capacity.

tialized with random weights, and the patch dataset is constructed from random sampling. These factors influence the optimization process and create measurements that can fluctuate.

Most of the experiments listed in Table 4.1 compare results from a certain method with a baseline. Running each experiment several times creates two independent samples from these populations. By statistical hypothesis testing and Welch's t-test, it is possible to assert whether it is plausible that the samples were produced from two distinct populations. Population i can be described by mean μ_i and variance σ_i^2. The null hypothesis $H_0 \colon \mu_1 = \mu_2$ is rejected in favor of the alternate hypothesis $H_1 \colon \mu_1 \neq \mu_2$, based on the p-value found by the t-test. If the p-value is below the significance level α of 0.05, H_0 is rejected. This indicates that it is unlikely that the samples came from the same underlying population.

Welch's t-test is derived from the Student's t-test, and is suitable in situations where the variance and mean of the underlying populations are unknown [Walpole et al., 2011, Chapter 10]. The mean and variance are estimated from the samples. Unlike the Student's t-test, Welch's t-test does not assume that the variances of the two populations are equal. However, both tests assume that the populations are normally distributed. Visual interpretation of normal Q-Q plots created from the measurements can give an indication of whether the populations are normally distributed. Unfortunately, whether or not the underlying populations were normal could not be confidently asserted because of the small sample size. Any inferences made in this chapter based on Welch's t-test are therefore conditional on the assumption of normality. In Appendix C, examples of normal Q-Q plots can be found.

The proposed methods were tested on two different datasets. Despite the datasets involving the same task of road segmentation, they do differ in many ways. The datasets depict separate aerial regions, have different GSD, contain different topography, and differ in label quality. Conducting experiments on both datasets might give an indication of whether the methods can be generally applicable or not.

For all experiments, most hyperparameters were kept constant. Only the parameters related to the research questions were varied. These hyperparameters are presented in Section 4.2. Furthermore, all regularization methods were enabled during testing. The proposed methods should provide some additional benefit when used in combination with a typical configuration of a CNN.

To show that the bootstrapping loss function is effective at handling inconsistent

labeling, it was compared with the cross-entropy loss function at different levels of label noise. To do so, label errors in the form of omission noise were artificially introduced to the label images. The label degradation involved removing between 0% and 40% of the road class pixels from the label images, before sampling the patch datasets. The road removal involved iteratively setting randomly sized regions of label pixels to 0. The experiment should show how well the various loss functions cope with omission noise, and whether the bootstrapping loss function is more robust than the cross-entropy loss function. Artificially introducing omission noise was tested with both road datasets in Experiment E1 and E2.

The bootstrapping loss function was further tested by Experiment E3, where the training set consisted of labels from the label set N50. The more accurate label set, Vbase, was used in the validation and test set. The N50 label set contains a lot of naturally occurring registration error, which means that the robustness of the different loss functions can be tested without artificial label degradation.

The effectiveness of curriculum learning was tested by training the network on two different patch datasets. The first was created according to a curriculum strategy, where each stage θ only contained examples with a difficulty estimate below D_θ. Subsequent stages have increasing difficulty thresholds, which results in the network being gradually introduced to harder examples. The second patch dataset was created with no regard to the difficulty, and sampled patches randomly. Specifically, it was created with the difficulty threshold $D_\theta = 1.0$ for all stages θ. The performance from training the network on this dataset formed the baseline that was compared to the performance of a network trained on the curriculum dataset. Both datasets contained the same number of stages, and therefore the same number of examples. The network configuration was also identical. Essentially, the only difference was the ordering of the examples in the patch datasets. Curriculum learning was tested with both aerial image datasets in Experiment E4 and E5.

The approach of curriculum learning also raises some other interesting questions. How well does a network perform if presented with the "harder" examples first? In Experiment E4, anti-curriculum learning was therefore tested. This resembles the teaching dimension method, as discussed in Section 2.3, in which harder examples closer to the decision boundary are presented first in order to resolve ambiguities quickly. Furthermore, Experiment E5 explored what happens when setting the difficulty threshold D_0 at different values.

In Experiment E6, the curriculum strategy was tested with a less experienced curriculum teacher than in Experiment E4 and E5. This should test whether

the proposed curriculum strategy is viable with a teacher that has trained with a limited number of examples. The experiment also tested the effect of training only on the first stage examples. In addition, a more gradual approach to changing the training set distribution was tested. This involved having several smaller stages mixed into the training set by random replacement.

In addition to presenting the experimental results as MSE test loss per epoch plots, the network performance is also presented as precision and recall curves. This is a common metric used to evaluate road detection systems, as discussed in Section 2.1.

A precision and recall curve is created by thresholding the network's prediction probabilities by values between 0 and 1, and then computing the precision and recall using the binarized predictions and labels. The result is a curve that illustrates the trade-off between precision and recall. As the recall increases, the precision usually decreases. Similar to [Mnih and Hinton, 2012], this thesis utilizes a relaxed measure of precision and recall, with a slack variable p set to 3. The relaxed precision is denoted as the fraction of detected road pixels that are within p pixels of a label road pixel. Whereas, the relaxed recall is defined as the fraction of true pixels that are within p pixels of a detected pixel. All experiments listed in Table 4.1 utilized the relaxed version of precision. However, only Experiment E6 and E7 recorded results using the relaxed recall measure, which means that most experiments have precision and recall values that are a bit more sensitive to minor deviations between predictions and labels.

The reason for using the relaxed version of precision and recall is that the majority of the label maps exhibit a small amount of registration noise. Therefore, it is unreasonable to count predictions that are slightly off target as errors. Consequently, misalignments of 3 pixels or less between the prediction and label will not affect the values of the relaxed precision and recall curve.

The results are also presented in tables that include the precision and recall breakeven point, which is derived from the precision and recall curve. This is the point on the curve where the precision and recall have an equal value. These points are also marked in the figures by a black dot.

4.2 Experimental Setup

The network configuration used by most experiments is listed in Table 3.1. Any deviations from this configuration will be detailed in this section. In addi-

tion, the specific hyperparameters used for each experiment can be found at `http://interface.ml/experiments`. Descriptions of tools used for conducting and presenting the experiments can be found in Appendix B. The relevant parameters for each experiment are detailed below.

E1 - Bootstrapping with the Massachusetts Roads Dataset

In this experiment, the bootstrapping loss function was compared to the cross-entropy loss function. The loss functions were compared by their performance on patch datasets with several levels of label degradation. The network was trained for 140 epochs and with 110800 examples. The learning rate was slightly decreased compared to the default configuration. The bootstrapping loss function's β parameter was set to 1.0, and was gradually decreased after epoch $M = 90$, to $\beta_{min} = 0.9$, by multiplying β with $\beta_{decrease} = 0.9$ each epoch. The parameters relevant for this experiment are listed in Table 4.2.

Label noise has been artificially added to the label images before constructing the patch datasets. Specifically, areas of road pixels have been removed incrementally by setting the label pixel values to zero. This process is continued until a certain percentage of road class pixels have been removed. In the context of aerial imagery, the artificially added label noise simulates omission noise. This experiment utilized patch datasets where 0, 10, 20, 30, and 40 percent of roads were removed from each label image. The top row of Figure 4 in Appendix E displays examples of artificial omission noise being added to a label.

Table 4.2: Key parameters of Experiment E1.

Method	**Parameters**
Baseline	100 epochs, s=110800, $a = 0.0011$, $\mathcal{L}$ = cross-entropy, omission noise levels=0%, 10%, 20%, 30%, 40%
Bootstrapping	100 epochs, s=110800, $a = 0.0011$, $\mathcal{L}$ = bootstrapping, β_{max}=1.0, β_{min}=0.9, M=60, omission noise levels=0%, 10%, 20%, 30%, 40%

E2 - Bootstrapping with the Norwegian Roads Dataset VBase

Experiment E2 tested the robustness towards label noise for the Norwegian Roads Dataset Vbase. This was done by comparing the performance of networks with different loss functions at several levels of omission noise. The experiment shared a very similar setup to Experiment E1. However, the parameter β was decreased slightly more than in E1. The experiment configuration is displayed in Table 4.3.

Table 4.3: Key parameters of Experiment E2.

Method	**Parameters**
Baseline	100 epochs, s=110000, $a = 0.0011$, $\mathcal{L}$ = cross-entropy, omission noise levels=0%, 10%, 20%, 30%, 40%
Bootstrapping	100 epochs, s=110000, $a = 0.0011$, $\mathcal{L}$ = bootstrapping, $\beta_{max} = 1.0$, $\beta_{min} = 0.8$, $M = 60$, emission noise levels=0%, 10%, 20%, 30%, 40%
Confident bootstrapping	100 epochs, s=110000, $a = 0.0011$, $\mathcal{L}$ = confident-bootstrapping, $\beta_{max} = 1.0$, $\beta_{min} = 0.8$, $M = 60$, omission noise levels=0%, 10%, 20%, 30%, 40%

E3 - Bootstrapping with the Norwegian Roads Dataset N50/VBase

In contrast to Experiment E2, this experiment tested the effect bootstrapping had on labels with a lot of registration noise. The training set consisted of examples from the Norwegian Roads Dataset N50. The label set N50 has coarser road centerline vectors, which result in a lot of registration noise compared to the other aerial image datasets. The experiment optimized models with $s = 165000$ patch examples, for 140 epochs. The learning rate a was slightly lower than the default. The bootstrapping parameter β was gradually decreased from $\beta_{max} = 1.0$ at epoch $M = 90$, to $\beta_{min} = 0.8$. Essentially, the bootstrapping loss functions incorporated its own predictions starting from epoch 90. Any improvements should, therefore, be seen in the test loss after this epoch. In addition, the confident bootstrapping loss function was also tested in this experiment. A summary of the experiment and key parameters can be found in Table 4.4.

Table 4.4: Key parameters of Experiment E3.

Method	**Parameters**
Baseline	140 epochs, s=165000, $a = 0.0011$, $\mathcal{L}$ = cross-entropy
Bootstrapping	140 epochs, s=165000, $a = 0.0011$, $\mathcal{L}$ = bootstrapping, β_{max}=1.0, β_{min}=0.8, M=90
Confident bootstrapping	140 epochs, s=165000, $a = 0.0011$, $\mathcal{L}$ = confident-bootstrapping, β_{max}=1.0, β_{min}=0.8, M=90

E4 - Curriculum Learning with the Massachusetts Roads Dataset

This experiment involved measuring the performance between a baseline and a curriculum patch dataset generated from the Massachusetts Roads Dataset. Both datasets had $N = 2$ stages, where each stage included 110800 training examples. Each stage θ contained examples where the difficulty estimate $d(y, q)$ was less than a threshold D_θ. The baseline patch dataset had a threshold D_θ of 1.0 for both stages, which is equivalent to random sampling. When switching between the first and second stage, the entire training set was replaced by examples from the second stage. This occurred at epoch $t_{start} = 50$. An additional patch dataset was also included, which tested the performance of anti-curriculum learning. In this dataset, the first stage excluded patches containing road pixels with a difficulty estimate $d(y, q)$ less than 0.25. Table 4.5 displays key parameters of this experiment.

Table 4.5: Key parameters of Experiment E4.

Method	**Parameters**
Baseline	120 epochs, s=220000, $d(y, q) < D_\theta$, $D_0 = 1.00$, $D_1 = 1.0$, $t_{start} = 50$
Curriculum	120 epochs, s=220000, $d(y, q) < D_\theta$, $D_0 = 0.25$, $D_1 = 1.0$, $t_{start} = 50$
Anti-curriculum	120 epochs, s=220000, $d(y, q) > D_\theta$, $D_0 = 0.25$, $D_1 = 0.0$, $t_{start} = 50$

The experiment's curriculum teacher was trained with a patch dataset of 442800 examples, sampled from the Massachusetts Roads Dataset. The teacher classifier was trained for 272 epochs. It achieved an MSE test loss of 0.0225, and a relaxed precision and recall breakeven point of 0.79.

E5 - Curriculum Learning with the Norwegian Roads Dataset

Experiment E5 had a similar setup to Experiment E4. There were four different patch datasets, each with two stages. The baseline patch dataset was created by random sampling, whereas the others were created by a curriculum strategy. Three different curriculum patch datasets were tested, where each stage 0 had a different D_0 threshold value.

The curriculum teacher that generated predictions for the difficulty estimation was a previously trained model. The teacher classifier was trained with a dataset consisting of 440000 examples for 174 epochs. The learning rate was multiplied by 0.85 every 50th epoch. Apart from that, the network parameters closely resembled the default parameters listed in Table 3.1. The classifier's final MSE test loss was 0.0222, and the relaxed precision and recall breakeven point was around 0.71.

The networks trained on the patch datasets used the default network configuration. Essentially, the only real difference between them was the first stage of the datasets. The networks were trained for 120 epochs with a stage switch at epoch 50. Important parameters of this experiment are listed in Table 4.6.

Table 4.6: Key parameters of Experiment E5.

Method	**Parameters**
Baseline	100 epochs, s=221600, $D_0 = 1.0$, $D_1 = 1.0$, $t_{start} = 50$
Curriculum 0.15	100 epochs, s=221600, $D_0 = 0.15$, $D_1 = 1.0$, $t_{start} = 50$
Curriculum 0.25	100 epochs, s=221600, $D_0 = 0.25$, $D_1 = 1.0$, $t_{start} = 50$
Curriculum 0.35	100 epochs, s=221600, $D_0 = 0.35$, $D_1 = 1.0$, $t_{start} = 50$

E6 - Curriculum Learning with an Inexperienced Teacher

While Experiment E4 and E5 had teachers that were trained with over 400000 examples, this experiment assumed that there is a limited amount of patches available. The teacher of this experiment was therefore only trained with 221600 examples, which is the same number of examples in each patch dataset. The teacher classifier was trained for 156 epochs, and achieved an MSE test loss of 0.0253, and a relaxed precision and recall breakeven point of 0.79.

In the experiment involving gradually increasing the difficulty of the training set, there were 326800 examples in total, split between $N = 5$ stages. The first stage had 110800 examples, whereas the remaining stages had 54000 examples each. In the baseline, the difficulty threshold D_θ was set to 1 for every stage. The first stage of the curriculum patch dataset only allowed examples with a difficulty below 0.25. The key parameters of Experiment E6, are displayed in Table 4.7.

E7 - Performance of the Road Detection System

The performance of the road detection system was tested in Experiment E7. The M1 network utilized a patch dataset with 3985200 examples that were randomly sampled from the Massachusetts Roads Dataset. The network was trained for 300 epochs, unless early stopping terminated the optimization at an earlier point. In order to train with such a large dataset, the network trained with only a subset of the examples at any given epoch. The training data was switched with another subset at every $epoch \mod 30 = 0$, starting from epoch 50. At any given epoch, the network was optimized by a total of 442800 examples. Any discrepancies from the default network configuration are listed in Table 4.8.

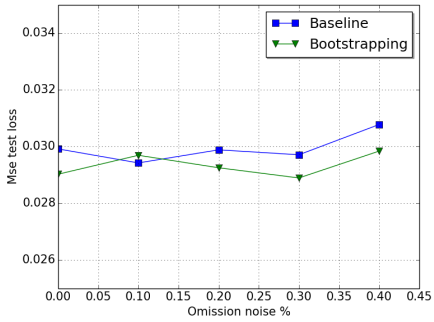
In addition to the M1 network, Network M2 and N1 further tested the road detection performance on both road datasets. The default network architecture was altered in both networks, as seen in Table 4.8. This resulted in a significant increase in adjustable parameters compared to M1. Both networks utilized the *confident bootstrapping* loss function, and were trained by curriculum patch datasets. The curriculum teachers of Experiment E4 and E5 were used to create the patch datasets.

Table 4.7: Key parameters of Experiment E6.

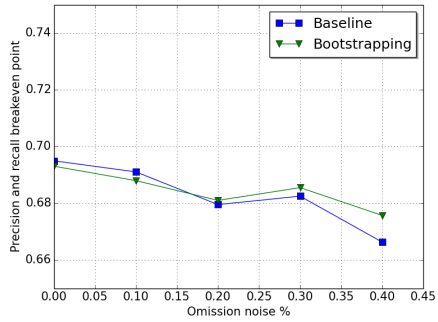
Method	**Parameters**
Baseline	120 epochs, s=221600, $D_0 = 1.0$, $D_1 = 1.0$, $t_{start} = 60$
Curriculum	120 epochs, s=221600, $D_0 = 0.25$, $D_1 = 1.0$, $t_{start} = 60$
Baseline, First stage only	120 epochs, s=221600, $D_0 = 1.0$
Curriculum, First stage only	120 epochs, s=221600, $D_0 = 0.25$
Baseline, Gradual	120 epochs, s=326800, $D_\theta = 1.0, \theta \in \{0, 1, 2, 3, 4\}$, $t_{start} = 60$, $t_{stage} = 15$
Curriculum, Gradual	120 epochs, s=326800, $D_0 = 0.25$, $D_\theta = 1.0, \theta \in \{1, 2, 3, 4\}$, $t_{start} = 60$, $t_{stage} = 15$

Table 4.8: Key parameters of Experiment E7.

Method	**Parameters**
M1- Massachusetts, cross-entropy, no-curriculum	300 epochs, $s = 3985200$, $a = 0.0015$, $b = 128$, $\mathcal{L}$ = cross-entropy, $ES_{init} = 400000$
M2- Massachusetts, confident, curriculum	85 epochs, $s = 1772800$, $a = 0.0025$,$a_{decrease} = 0.9$ $b = 128$, $m = 0.93$, $\mathcal{L}$ = confident bootstrapping, $ES_{init} = 500000$, $K^{(0)} = 120$, $CLF^{(0)} = (16, 16)$, $CLF^{(1)} = (8, 8)$, $CLS^{(0)} = (2, 2)$, $p^{0,1,2} = 0.85$, $D_0 = 0.25$, $D_\theta = 1.0, \theta \in \{1, 2, 3, 4, 5, 6\}$, $t_{start} = 30$, $t_{stage} = 15$
N3 - Norwegian, confident, curriculum	85 epochs, $s = 1760000$, $a = 0.0025$, $a_{decrease} = 0.9$, $b = 128$, $m = 0.93$ $\mathcal{L}$ = confident bootstrapping, $ES_{init} = 500000$, $K^{(0)} = 120$, $CLF^{(0)} = (16, 16)$, $CLF^{(1)} = (8, 8)$, $CLS^{(0)} = (2, 2)$, $p^{0,1,2} = 0.85$, $D_0 = 0.25$, $D_\theta = 1.0, \theta \in \{1, 2, 3, 4, 5, 6\}$, $t_{start} = 30$, $t_{stage} = 15$



(a) MSE test loss.



(b) Precision and recall breakeven.

Figure 4.1: E1 - Robustness of bootstrapping for increasing amounts of label noise in the Massachusetts Roads Dataset.

4.3 Experimental Results

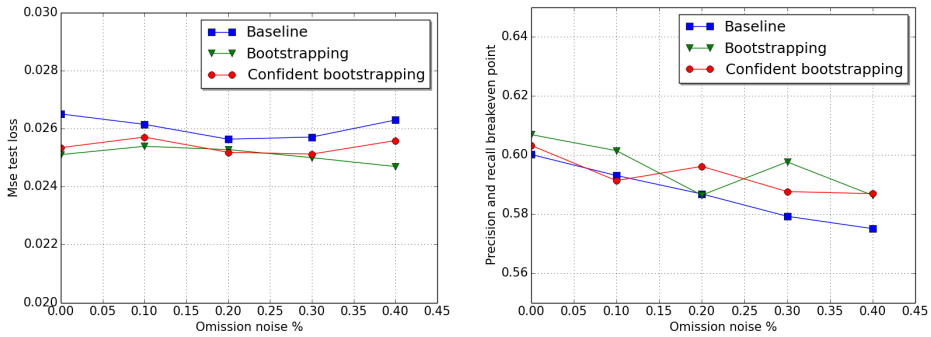
4.3.1 Bootstrapping for Datasets with Noisy Labels

The results from Experiment E1 and E2, comparing the bootstrapping methods and the baseline method at several levels of label noise, are displayed in Figure 4.1 and Figure 4.2. The plots in the first figure show the performance of models trained on patch examples from the Massachusetts Roads Dataset, whereas the second figure displays results from training on patch examples from the Norwegian Roads Dataset. Experiment E2 also includes the performance of confident bootstrapping. The results from Experiment E3 are displayed in Figure 4.3. This experiment did not add any artificial omission noise, but utilized the label set N50.

Figure 4.1a displays the test loss at different levels of label noise. The baseline test loss seems to increase slightly as the noise level is increased. The bootstrapping loss function has a lower test loss compared to the baseline for noise levels above 10 percent.

Figure 4.1b displays the precision and recall breakeven points for increasing levels of label noise, and shows a trend of decreasing values. At around 20 percent, the breakeven point of bootstrapping surpasses the baseline. However, the difference in performance seems to be small.

In Figure 4.2a, the test loss of bootstrapping and confident bootstrapping is con-



(a) MSE test loss.

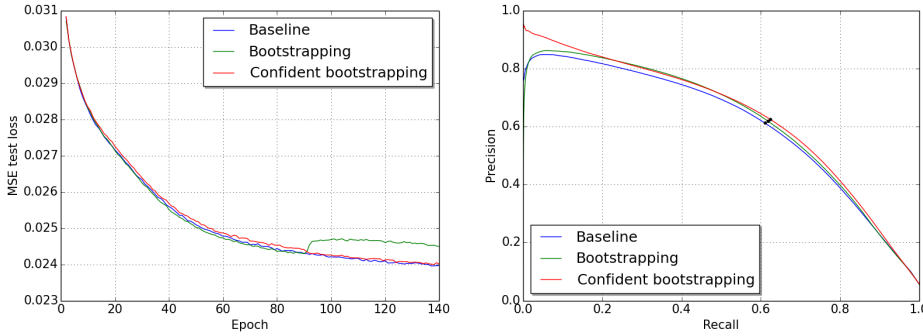
(b) Precision and recall breakeven.

Figure 4.2: E2 - Robustness of bootstrapping for increasing amounts of label noise in the Norwegian Roads Dataset.

sistently lower than the baseline. The test loss plots do not seem to increase as much as expected. However, the precision and recall breakeven points of the baseline decrease with increasing levels of omission noise. This can be seen in Figure 4.2b. Although the bootstrapping methods behave a bit more erratic, the plots seem to be decreasing at a slightly slower rate. The plots also show that the overall difference in performance is small. Detailed figures of the precision and recall curves, and test loss per epoch plots of each level of label noise can be found in Appendix D.

The difference between bootstrapping and confident bootstrapping is clear in Figure 4.3a, which shows the test loss per epoch for Experiment E3. After epoch 90, the factor β is decreased in favor of the model predictions. The test loss of confident bootstrapping sees an immediate decrease compared to the baseline, whereas the bootstrapping test loss increases in value. In terms of test loss, confident bootstrapping performed comparably to the baseline, while the regular bootstrapping performed worse. However, this result is not evident from the precision and recall curve, displayed in Figure 4.3b. Both bootstrapping methods have better relaxed precision values for every level of recall.

The results from Experiment E1, E2, and E3 are displayed in Table 4.9. The table contains a row for every test, which includes the percentage of artificial omission noise, precision and recall breakeven point, test loss, and a p-value. The p-values were computed by the Welch's t-test from test loss samples of two different methods. For instance, the p-value of *E1 - bootstrapping 0%* are com-



(a) MSE test loss

(b) Precision and recall comparison.

Figure 4.3: E3 - Comparison of loss functions using the Norwegian Roads Dataset Vbase. The training set consists of labels affected by omission and registration noise.

puted by the test loss samples of *E1 - Bootstrapping 0%* and *E1 - Baseline 0%*. If a p-value is below the significance level of 0.05, it is plausible that the samples originate from a population different from the population of the baseline samples. The bootstrapping test loss results were not statistically significant.

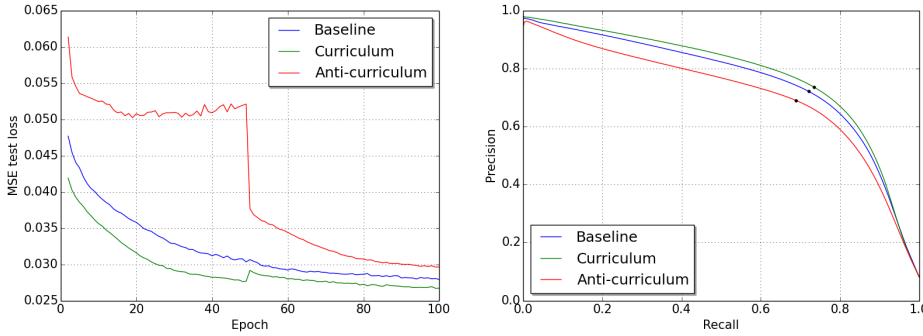
4.3.2 Curriculum Learning with Aerial Imagery

Curriculum learning was tested in Experiment E4, E5, and E6. Results from Experiment E4 are displayed in Figure 4.4, whereas results from Experiment E5 are displayed in Figure 4.6. In addition, Experiment E4 explored anti-curriculum learning, while Experiment E5 tested patch datasets constructed with various difficulty thresholds D_0. Finally, Experiment E6 investigated gradual curriculum learning and the performance of only training with the first stage of a curriculum patch dataset. The results from this experiment can be found in Figure 4.7.

Experiment E4 was conducted using the Massachusetts Roads Dataset. In Figure 4.4a, the switch from stage 0 to stage 1 at epoch 50 is clearly visible for the *curriculum* and *anti-curriculum* test loss plots. The network trained with the curriculum patch dataset performed better than the baseline, as seen in Figure 4.4a and Figure 4.4b. The test loss gap between the *baseline* and *curriculum* is largest before epoch 50, but is still maintained to a lesser extent after this epoch. The precision and recall breakeven point of *curriculum* is above that of the *base-*

Table 4.9: Bootstrapping results.

Experiment	**%**	**Dataset**	**Breakeven**	**Test loss**	**P-value**
E1 - Baseline	0	Massachusetts	0.6949	0.0299	-
E1 - Baseline	10	Massachusetts	0.6910	0.0294	-
E1 - Baseline	20	Massachusetts	0.6795	0.0299	-
E1 - Baseline	30	Massachusetts	0.6825	0.0297	-
E1 - Baseline	40	Massachusetts	0.6663	0.0308	-
E1 - Bootstrapping	0	Massachusetts	0.6930	0.0290	0.15164
E1 - Bootstrapping	10	Massachusetts	0.6879	0.0297	0.64355
E1 - Bootstrapping	20	Massachusetts	0.6810	0.0292	0.25650
E1 - Bootstrapping	30	Massachusetts	0.6855	0.0289	0.22492
E1 - Bootstrapping	40	Massachusetts	0.6757	0.0299	0.15764
E2 - Baseline	0	Norwegian	0.6001	0.0265	-
E2 - Baseline	10	Norwegian	0.5930	0.0261	-
E2 - Baseline	20	Norwegian	0.5868	0.0256	-
E2 - Baseline	30	Norwegian	0.5792	0.0257	-
E2 - Baseline	40	Norwegian	0.5750	0.0263	-
E2 - Bootstrapping	0	Norwegian	0.6069	0.0251	0.09136
E2 - Bootstrapping	10	Norwegian	0.6014	0.0254	0.29272
E2 - Bootstrapping	20	Norwegian	0.5864	0.0252	0.48285
E2 - Bootstrapping	30	Norwegian	0.5976	0.0250	0.11260
E2 - Bootstrapping	40	Norwegian	0.5863	0.0247	0.00480
E2 - Confident	0	Norwegian	0.6032	0.0253	0.09433
E2 - Confident	10	Norwegian	0.5913	0.0257	0.43790
E2 - Confident	20	Norwegian	0.5961	0.0252	0.36447
E2 - Confident	30	Norwegian	0.5876	0.0251	0.09397
E2 - Confident	40	Norwegian	0.5869	0.0256	0.17952
E3 - Baseline	0	Norwegian	0.6114	0.0240	-
E3 - Bootstrapping	0	Norwegian	0.6186	0.0245	0.18114
E3 - Confident	0	Norwegian	0.6244	0.0240	0.90851



(a) Comparison of test loss.

(b) Precision and recall comparisons.

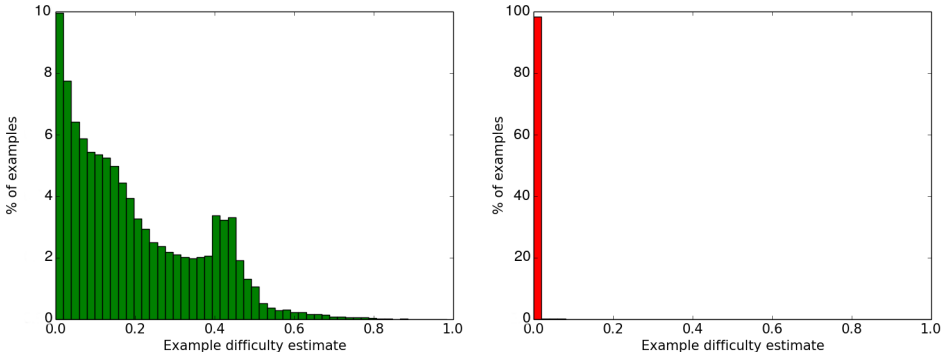
Figure 4.4: E4 - Performance of curriculum learning and anti-curriculum learning using the Massachusetts Roads Dataset.

line as well.

Training with an anti-curriculum strategy, where the first stage consists of harder examples, seemed to do worse than both the *baseline* and *curriculum*. This is especially evident in the test loss plot, where the test loss stagnates at around epoch 25, and starts to overfit towards epoch 50. From epoch 50 when stage 1 examples entered the training set, there is a dramatic decrease in test loss. The test loss decrease continues at a steady pace until epoch 85. Yet, the final performance of anti-curriculum learning is substantially lower than both the baseline and curriculum learning, as illustrated by the breakeven points in Figure 4.4b.

The training set difficulty distribution, according to the difficulty estimator $d(y, q)$ from Experiment E4, is illustrated in a histogram in Figure 4.5. The training set of the Massachusetts Roads Dataset was sampled, and the difficulty estimate was computed for every patch. Patches that contained no road label pixels were counted separately from patches that contained road label pixels. The resulting histogram for non-road patches in Figure 4.5b shows that the trained curriculum teacher of Experiment E4 is excellent at predicting non-road patches, because the predictions and labels match very well. This results in very low difficulty estimates. This is not the case for road patches, illustrated by Figure 4.5a, where the distribution is clearly skewed right and has a greater variance.

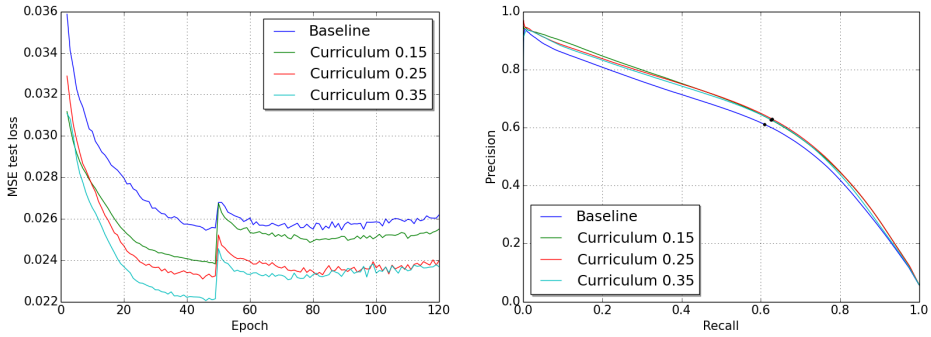
In Experiment E5, the road detection system was trained on four different patch datasets created from the Norwegian Roads Dataset. A comparison of the test



(a) Distribution of road patches. (b) Distribution of non-road patches.

Figure 4.5: E4 - Example difficulty distribution according to estimator $d(y, q)$.

loss per epoch plots is displayed in Figure 4.6a, whereas the final performance of each patch dataset is shown by a precision and recall curve in Figure 4.6b. The network configuration used for the tests was identical, as well as the number of training examples seen during training. Essentially, the gap between the *baseline* and *curriculum* curve is the result of the different difficulty thresholds D_0.



(a) Comparison of test loss. (b) Precision and recall comparisons.

Figure 4.6: E5 - Performance of curriculum learning at different thresholds, D_0 using the Norwegian Roads Dataset Vbase.

Observing the plots in Figure 4.6a, the switch from stage 0 to stage 1 is clearly visible at epoch 50. The increase in test loss is most severe in the datasets formed

by a curriculum strategy. Leading up to epoch 50, the curriculum plots show an increasing gap in test loss if compared to the baseline plot. After the switch, the curriculum datasets still outperform the baseline dataset, even though the example distribution of stage 1 are the same for all patch datasets.

In Figure 4.6b, the networks trained with a curriculum strategy show an improved relaxed precision for all levels of recall compared to the baseline.

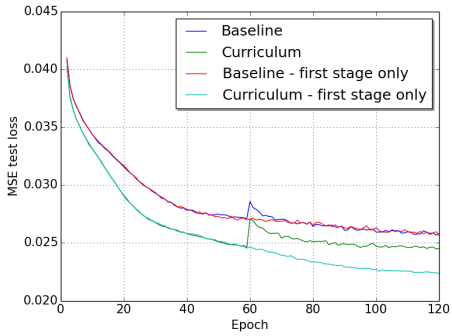
An interesting trend between the thresholds D_0 and the test loss is that decreasing the difficulty threshold does not necessarily produce better results. The patch dataset *Curriculum 0.15* has the easiest first stage, yet performed worse than *Curriculum 0.25* and *Curriculum 0.35*.

Figure 4.7 displays the results from Experiment E6. The patch datasets in this experiment were created with the use of a teacher classifier that was trained with the same number of training examples as the patch datasets. Curriculum learning still achieved a higher test loss and gave better relaxed precision for all levels of relaxed recall, compared to the baseline.

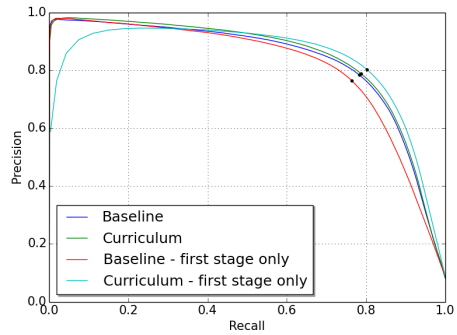
Figure 4.7a and 4.7b illustrate the performance of training with only the first stage of the patch datasets. As expected, the first stage baseline has a lower breakeven point than the two stage baseline. More surprisingly, the first stage curriculum outperformed the baseline and the curriculum, both in breakeven point and test loss. However, at recall levels less than 0.4, the precision of the first stage curriculum decreases rapidly. The test loss of the curriculum and the baseline plot increased substantially at epoch 60 when the stage switch occurred.

This was further explored by creating two additional patch datasets with smaller stages that were gradually mixed into the training set. Subsequent stages replaced parts of the training set by assigning each example to a random position in the training set. Because the positions are picked by random sampling with replacement, only around 40% of the existing examples in the training set were replaced at each switch. This results in a more gradual transition of the difficulty distribution. As seen from Figure 4.7c, the test loss plots do not exhibit the same test loss increase after a stage switch. The baseline plot even decreases substantially after epoch 60. The gradual approach also seems promising in terms of precision and recall, as seen in Figure 4.7. The network trained with a curriculum patch dataset achieved a breakeven point of 0.8088, compared to a breakeven of 0.7952 for the network trained with the baseline patch dataset.

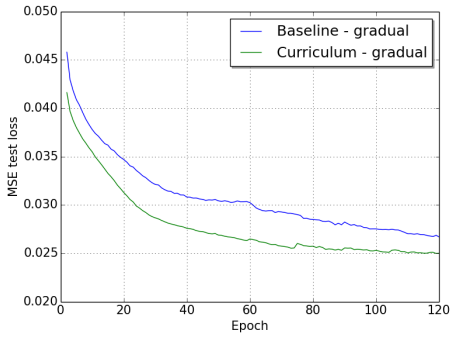
The results from Experiment E4, E5, and E6, are listed in Table 4.10. Each row



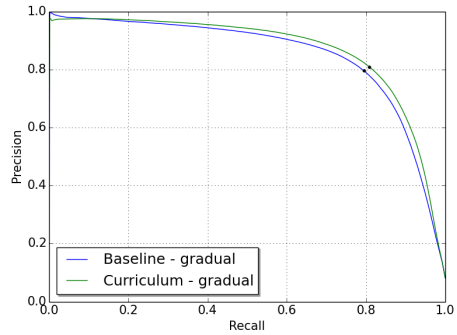
(a) Test loss for two stage dataset.



(b) Precision and recall comparisons with two stage dataset.



(c) Test loss with dataset with gradual stage switching.



(d) Precision and recall for dataset with gradual stage switching.

Figure 4.7: E6 - Results from experiments with an less experienced teacher model.

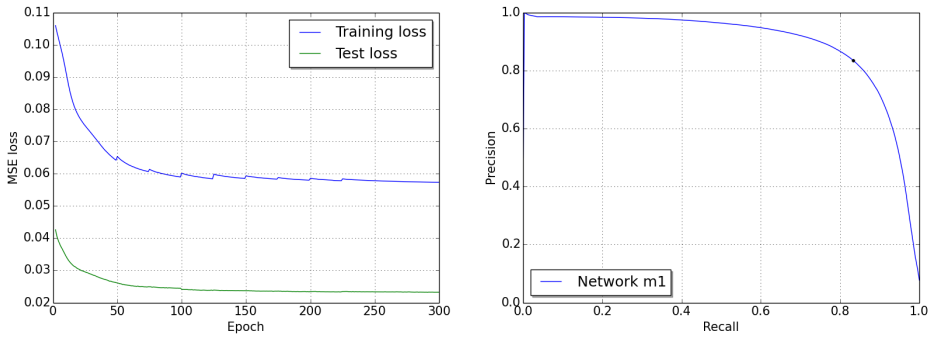
Table 4.10: Curriculum learning results.

Experiment	D_0	Dataset	Breakeven	Test loss	P-value
E4 - Baseline	1.0	Massachusetts	0.7211	0.0279	-
E4 - Curriculum	0.25	Massachusetts	**0.7353**	0.0267	0.01023
E4 - Anti-curriculum	0.25	Massachusetts	0.6904	0.0296	0.00057
E5 - Baseline	1.00	Norwegian	0.6105	0.0262	-
E5 - Curriculum	0.15	Norwegian	0.6264	0.0255	0.18160
E5 - Curriculum	0.25	Norwegian	**0.6292**	0.0240	0.00817
E5 - Curriculum	0.35	Norwegian	0.6269	0.0237	0.00017
E6 - Baseline	1.00	Massachusetts	0.7836	0.0258	-
E6 - Curriculum	0.25	Massachusetts	0.7880	0.0245	0.01519
E6 - Baseline, gradual	1.00	Massachusetts	0.7952	0.0267	-
E6 - Curriculum, gradual	0.25	Massachusetts	**0.8088**	0.0250	0.00007

lists a test, and shows the difficulty threshold for that particular patch dataset, precision and recall breakeven point, test loss, and a p-value. The p-values were computed by the Welch's t-test, using the test loss samples from a curriculum test and the baseline. If a p-value is below the significance level of 0.05, it is plausible that the curriculum and baseline samples originate from distinct populations. The curriculum tests that used a difficulty threshold of 0.25 and 0.35 have small p-values well below 0.05. The results were probably not sampled from the same underlying population. Therefore, it is plausible that curriculum learning can improve the generalization accuracy of a CNN.

4.3.3 Road Detection System

In Experiment E7, the road detection network M1 was trained on a very large patch dataset. In order for the patch dataset to fit in main memory, the examples were split into nine separate stages, each containing 442800 training examples. The content of the training set was switched during training. These switches are noticeable in the training loss plot in Figure 4.8a, where the training loss



(a) MSE loss.

(b) Precision and recall.

Figure 4.8: E7 - Performance of the M1 road detection system trained with the Massachusetts Roads Dataset.

increases slightly after a training set switch. The network converged to a test loss of 0.0232. In Figure 4.8b the precision and recall curve of Network M1 is displayed.

An additional road detection system was trained with a modified network architecture. This increased the number of adjustable model parameters considerably. The trained network, M2, achieved a higher precision and recall breakeven point than Network M1. The same architecture was also tested with a large patch dataset constructed from the Norwegian Roads Dataset.

The precision and recall breakeven point of these networks and comparable experiments conducted in other works are listed in Table 4.11. These systems have been described in Section 2.3.

4.4 Experimental Analysis

In the following analysis, the results presented in the previous section will be explored more in-depth. Furthermore, qualitative results from the road detection system are evaluated, and might illustrate why methods for dealing with noisy labels are important for this type of dataset.

[1]The number of experiment runs is unclear.
[2]See footnote 1.

Table 4.11: Road detection system results. The values in this table represent the precision and recall breakeven points achieved by the systems.

System	Dataset	Breakeven	Best network
Network M1	Massachusetts	0.8341	0.8413
Network M2	Massachusetts	0.8494	0.8627
Mnih [2013][1]	Massachusetts	0.8873	
Saito and Aoki [2015][2]	Massachusetts	0.8866	
Network N1	Norwegian	0.7508	0.7620

The CNN and the proposed methods are in some sense comparable to the three groups of approaches for dealing with label noise, described in Section 2.1.2. The bootstrapping loss function is clearly a noise-tolerant approach, where the loss function is modified in an attempt to make the network more robust towards label noise. The CNN and the regularization methods can be categorized as a noise-robust model. Whereas, curriculum learning in some sense can be described as a data cleansing method. The "simple" stages are created based on filtering techniques, where inconsistency between label and prediction determines whether an example is excluded or not. However, examples are not excluded from every stage, but only from the initial stages of training.

4.4.1 The Effect of Bootstrapping

Unfortunately, the effect of employing the bootstrapping loss function is quite small. However, the bootstrapping methods did perform nearly equal or slightly better when observing the breakeven values in Experiment E1, E2, and E3, and for increasing levels of omission noise, as seen in Figure 4.1 and Figure 4.2. It seems that bootstrapping do exhibit some robustness towards label noise. Compared to the performance of the baseline, the difference in both test loss and breakeven seems to be increasing.

Even though up to 40% of the roads present in the label images were removed, it did not particularly affect the baseline much. The baseline network seems to be surprisingly robust towards label noise. However, the default patch dataset sampling policy might be somewhat responsible for this.

In Experiment E1, and E2, only omission noise was artificially added to the label images. This simply removed road pixels from the label images, until a certain percentage of road pixels had been removed. However, when the patch creator sampled the aerial dataset, the preference for an even balance between patches containing road pixels and patches not containing any road pixels were enabled. Even for increasing levels of omission noise, the patch creator still sampled around 50% road patches with almost no noise added. The portion of non-road patches in the patch dataset was of course affected by the increase in label noise. Adding artificial, but realistic registration noise to the label images were not done in the experiments. This would have affected the entire patch dataset. In Appendix E, the results from increasing levels of label noise by flipping label pixels are presented. In this scenario, bootstrapping is much more effective. Unfortunately, this type of label noise is highly unrealistic for this type of dataset.

There is also the issue of seemingly contradictory results in Experiment E3. Even though bootstrapping in Figure 4.3 shows an increase in the loss towards the end, it still achieves a better precision and recall curve than the cross-entropy loss. This is probably an indication of bootstrapping actually working. It seems that the bootstrapping loss function slightly adjusts the predictions to be more consistent between perceptually similar examples at the cost of an increasing test loss. The Vbase test set labels are not perfect, and exhibit some registration and omission noise, which probably explains the increase in test loss. The bootstrapping function might be reducing the impact of local registration noise by adjusting the predictions to fit the road pixels better. These prediction adjustments will be visible in the test loss plot, but will not affect the precision and recall curve because of the relaxed measure of precision.

The confident bootstrapping loss function behaves slightly different compared to bootstrapping, as demonstrated by the test loss figures 4.2a, and 4.3a. In these figures, the confident bootstrapping seems to follow the general outline of the baseline more closely than bootstrapping. The only difference between the two loss functions is that confident bootstrapping modifies targets using only confident pixel predictions. In addition, for the N50 label set in Experiment E3, this loss function actually performed slightly better than bootstrapping and the baseline in terms of precision and recall.

In summary, the experiments show that bootstrapping has a small positive effect on the results. However, the performance gains were not statistically significant for tests that involved omission and registration noise.

4.4.2 Curriculum Learning by Using an Artificial Teacher

All experiments comparing a randomly sampled patch dataset and a patch dataset constructed from a curriculum strategy, showed that presenting easier examples first have a positive impact on the classifier's ability to generalize. Furthermore, the proposed curriculum strategy that is based on measuring disagreement between the labels and the predictions that were produced by a teacher classifier, seems to be a viable approach for conducting curriculum learning.

A benefit of using this particular curriculum strategy is that it appears to be generally applicable. As long as a teacher classifier is trained to a certain level for an arbitrary dataset, it should be possible to create a curriculum dataset using the difficulty estimator $d(y, q)$. However, it does require training an additional classifier, and the resulting quality of the curriculum dataset probably corresponds to the achieved accuracy of the teacher classifier. For images this strategy is compelling, since judging the perceived difficulty of examples based on the actual image content is hard.

Experiment E4 and E5 demonstrated that the examples presented first do impact the final performance of the network. This is evident from both the precision and recall curve, as well as the test loss. It is conceivable that a less challenging training set distribution puts the network in an advantageous area of parameter space. Even though the latter half of training for the curriculum tests were conducted on a training set with the same example distribution as the baseline tests, the starting advantage of curriculum learning was still preserved in the final performance.

From Experiment E4 it is also clear that anti-curriculum learning does not provide the same advantage as curriculum learning. The performance in Figure 4.4a converges already at around epoch 25, with a test loss considerably higher than the baseline. It is only able to approach the test loss of the baseline after switching to the natural example distribution at epoch 50. The final test loss converged to a level well above the baseline. This also illustrates that the examples presented first have a large influence on the outcome. It is possible that early optimization on harder examples can guide the network to an unfavorable local minimum, which is hard to escape from later on.

The results further show that the outcome of curriculum learning is sensitive to the threshold parameter D_0. Figure 4.6a reveals that decreasing threshold value D_0 diminishes the effect of curriculum learning. Decreasing the threshold value results in a smaller pool of eligible examples that can be included in the first stage of a training set, which, in turn, can reduce the training set variability.

A bit surprising are the results from Experiment E6, which showed that training with the first stage only, did better than curriculum learning. This indicates that the inexperienced teacher model actually did a good job separating the very hard and possibly inconsistent examples out from the first stage. It also shows that the second stage probably contained a good amount of inconsistent examples, which can penalize the network incorrectly and affect the outcome. Alternatively, this might indicate that the threshold parameter D_0 was set to a value which resulted in sufficient example variation in the first stage training set. The network is therefore able to generalize well to the task of road detection.

However, assuming that harder examples are inconsistent and exclude them entirely from the training set should be done with caution. This could lead to hard, but correctly labeled examples being excluded, as well as unfamiliar examples that the curriculum teacher has not seen before. If the teacher classifier could accurately detect inconsistent labeling, there would be no reason for doing curriculum learning in the first place.

A potential reason for the large spike in test loss after a stage switch is that the entire training set is suddenly replaced. As seen in the test loss of Figure 4.7c, gradually mixing in examples from smaller subsequent stages seems to alleviate this. The breakeven point for this test is also substantially higher than that of the baseline, the first-stage-only curriculum, and the inexperienced teacher classifier.

In summary, the composition of the first stage has a considerable effect on the outcome of training. Furthermore, the results demonstrate that the proposed curriculum strategy works well in practice.

4.4.3 Performance of the Road Detection System

The images in Figure 4.9 illustrate qualitatively the performance of the network with the best performance on the Massachusetts Roads Dataset. The prediction image in Figure 4.9c has been stitched together from 16×16 prediction patches. For this particular test image, the model was able to identify the majority of the roads present, except for an almost imperceptible dirt road on the right side of the image. There are also some prediction errors, such as roads being disconnected, and prediction artifacts in the forest areas. However, the majority of the forest artifacts have low prediction probabilities, and can be removed by a threshold operation applied to the probabilities. For instance, the threshold value which results in the best precision and recall trade-off of the model was used to binarize the predictions in Figure 4.9d.

An interesting observation is that the model also correctly predicts small private roads leading up to houses, as seen in Figure 4.9. Furthermore, the model detects construction roads in the upper left corner. Since these roads are not present in the label image, the model is penalized for making these predictions by the cross-entropy loss function.

The prediction errors are displayed in Figure 4.9d, where the road label pixels and road prediction pixels are superimposed on the aerial image. The road pixels that are colored green have been correctly predicted, whereas the red and blue colored pixels show the prediction errors. The red pixels indicate areas where the system failed to predict road, and the blue pixels show areas where the system incorrectly predicted road. Yet, the majority of the prediction errors are understandable, and arguably not actually errors at all. Most of the blue areas are covering pixels that depict asphalt surfaces, and some of the red areas have trees covering the road. However, a certain challenge is the amount of disconnected roads, that especially occur at road junctions and highway ramps. Possible reasons for these prediction errors can be the low frequency of junctions and ramps in the dataset, or that the model capacity is inadequate. More results similar to Figure 4.9 can be found in Appendix F.

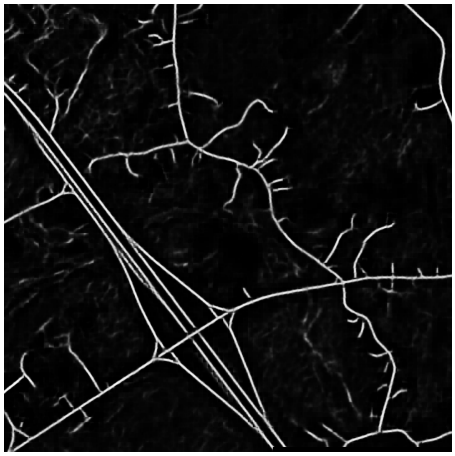
One of the most compelling ways of reducing the number of disconnected roads is by utilizing structured output prediction methods, as discussed in Section 2.3. Several studies [Kluckner et al., 2010] [Alvarez et al., 2012] [Mnih and Hinton, 2010] have shown that employing a smoothness prior by taking neighboring predictions into account can significantly improve generalization in semantic segmentation tasks. This is unfortunately outside the scope of this thesis.

The precision and recall breakeven point of M1 is considerably lower compared to other works, as seen in Table 4.11. The most likely explanation is the default configuration of this network. The configuration of the first layer in the network trained by Mnih [2013] was different. Mnih [2013] used overlapping max pooling in the first layer, whereas the max pooling of M1 was non-overlapping. This resulted in fewer learnable parameters in M1, which could have affected the network's model capacity or ability to fit the data. The spatial reduction of the input in the first convolutional layer especially affects the number of incoming connections to the first fully connected layer of the network. The network by Mnih [2013] has approximately 17.3 million adjustable weights.

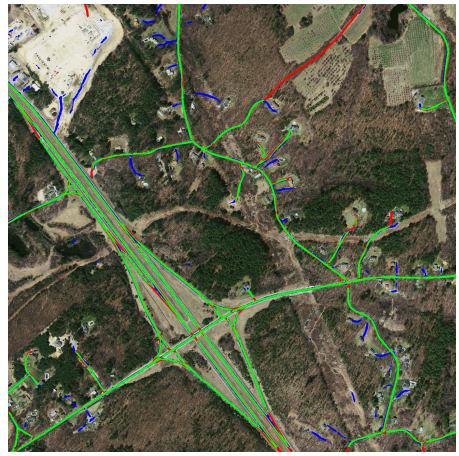
This was tested in Network M2, which used a different stride and kernel sizing. The number of adjustable weights in this network was around 5 million compared

(a) Label image.

(b) Aerial image.



(c) Model predictions.



(d) Prediction hit and miss image.

Figure 4.9: E7 - Example of M2's road detection performance. The aerial image is part of the test set in the Massachusetts Roads Dataset.

(a) Non-road pixels superimposed.

(b) Road pixels superimposed.

Figure 4.10: Examples of ill-suited line thickness.

to around 1.6 million in Network M1. A large portion of the difference can be traced back to the massive increase in incoming connections to the fourth layer. The number of incoming connections in M1 was 80, compared to 720 in M2. This network achieved a precision and recall breakeven point of 0.8494, even though it was trained for fewer epochs, and with a smaller training set. In addition, this network was trained using a gradual curriculum strategy, and used the confident bootstrapping loss function.

The same network configuration, loss function, and training regime were tested with the Norwegian Roads Dataset Vbase in Network N1. The precision and recall breakeven point of this network was substantially lower. There are probably several reasons for this. First, the lower GSD of this dataset implies that image patches of 64×64 pixels convey less context compared to similarly sized patches from the Massachusetts Roads Dataset. This might reduce the network's ability to discriminate between road and non-road pixels in situations where surrounding context is key. Second, the results from the two roads datasets cannot be directly compared because they are different. The Norwegian Roads Dataset has images of varying image quality, and depict a wide range of topographical features. Furthermore, The Vbase label set has been rasterized with ill-suited line thicknesses for roads that are particularly wide or narrow. The examples in Figure 4.10 illustrate this.

Chapter 5

Conclusion

The goal of this thesis was to create a road detection system able to detect roads from aerial images. This was achieved by utilizing a convolutional neural network and training it on datasets created from existing map data. This thesis has given a brief introduction to road extraction systems, and the related problem of label noise often found in aerial image datasets. This chapter concludes this work, first by giving a brief overview of the thesis in Section 5.1. Then, in Section 5.2, the research goal and questions are resolved based on the results from Chapter 4. Contributions made by this thesis are discussed in Section 5.3, and suggestions for future work are presented in Section 5.4.

5.1 Overview

The research questions and motivations were introduced in Chapter 1. The task of segmenting objects from aerial imagery was presented, as well as the related problem of label noise.

In Chapter 2, the thesis presented background theory and related works. The components of a CNN was briefly introduced, as well as topics such as curriculum learning and approaches for dealing with noisy labels. The related works presented studies involving road detection systems, semantic segmentation, methods for dealing with noisy labels, and curriculum learning. The issue of label noise was influential in the choice of a curriculum strategy, and the studies inspired the testing methodology used in Chapter 4.

Chapter 3 presented a detailed description of the selected methods and the aerial

image datasets. This included a presentation of the default hyperparameters, the architecture, the optimization and regularization methods used by the CNN, the bootstrapping loss function, and the curriculum strategy. The chapter also described how the Norwegian Roads Dataset was constructed from aerial imagery and existing map data, and why it is compelling to test the proposed methods with this type of dataset.

Experimental results from the proposed methods and the road detection system were presented in Chapter 4. First, the design of each experiment was described. Then, the specific details and parameters for each experiment were listed. Finally, the results were presented and then analyzed.

5.2 Evaluation

The goal of this thesis was to segment road pixels from aerial images using a CNN. This network shares many similarities to the networks employed by Mnih and Hinton [2012] and Mnih [2013]. Experiments demonstrated that the best architecture trained on the Massachusetts Roads Dataset achieved an averaged precision and recall breakeven point of 0.8494. This is a bit below comparable results from other works. The most likely explanation is that the network architecture constrained the capacity of the model. From the qualitative analysis, the classifier seemed to have generalized fairly well to the task of road segmentation.

The research questions of this thesis are related to this goal. Automatically generated aerial image datasets often suffer from label noise, and the research questions involve methods that can possibly alleviate the negative effects of inconsistent labeling. This section will attempt to resolve the research questions defined in Section 1.2 by a brief discussion based on the results from Chapter 4.

Research question 1:

Does the bootstrapping loss function give a significant improvement of precision and recall for datasets with noisy labels?

Mnih and Hinton [2012] showed that performance could be improved for aerial imagery by having the loss function model the noise distribution. They also found two particular breeds of label noise in aerial image datasets, which they called omission and registration noise. The bootstrapping loss function proposed by Reed et al. [2014] has also shown promising results for several noisy datasets. In this thesis, this particular loss function was therefore tested on aerial image datasets, to see if it provided robustness towards omission and registration noise.

Furthermore, a proposed variation of bootstrapping was also tested.

The experiments demonstrated that the bootstrapping method did provide some robustness towards label noise. For increasing levels of omission noise, the gap in performance between the cross-entropy and the bootstrapping loss function seemed to be somewhat increasing. However, the results were not statistically significant. The loss function was not tested on artificially increasing levels of registration noise, which could make the results more convincing. This would require some sort of image morphing, and local skewing of the roads present in the label images.

Tests performed on the Norwegian Roads Dataset N50 showed that both bootstrapping methods had precision and recall values slightly better than the baseline. In summary, the bootstrapping loss function seemed to have a slightly positive effect on noisy labels. Unfortunately, the effect was not statistically significant for increasing levels of omission noise.

Research question 2:

How can curriculum learning improve results in deep learning, and does this improve precision and recall for aerial images?

Curriculum learning proposed by Bengio et al. [2009] demonstrated compelling results for several tasks. The method increased generalization accuracy and resulted in faster convergence, by organizing each dataset according to the estimated difficulty of each example. The classifier was trained by gradually introducing harder examples to the training set. However, the curriculum strategies used for sorting the datasets were specific for each task, and were not particularly adaptable to road detection. This challenge was addressed by SPL [Kumar et al., 2010], where the curriculum strategy is internalized in the model. The model simultaneously optimizes the objective function and determines what examples to consider at each iteration. Examples that are easily predicted are considered "easy" by this approach.

The proposed curriculum strategy in this thesis does not alter the classifier algorithm. Instead, the example difficulties are estimated based on inconsistencies between labels and predictions. A curriculum dataset is built based on these estimates. The examples of the curriculum dataset are then mixed into the training set during training. A limitation of this method is that its effectiveness is based on the predictions generated from a curriculum teacher classifier, which has to be trained beforehand. However, the method is not task specific and can probably be applicable to any domains. This curriculum strategy can also be applied to any supervised algorithm, since the strategy primarily affects the dataset and not

the algorithm.

The experiments demonstrated consistently better generalization accuracy when utilizing curriculum learning. The resulting precision and recall curves from experiments conducted with both the Norwegian Roads Dataset and the Massachusetts Roads Dataset, showed that networks trained on the curriculum datasets performed better than those trained on the baseline datasets. The experiments used a two stage training set, with the first stage containing examples with low difficulty estimates. The entire training set was replaced mid-training.

The baseline and curriculum tests had the same network configuration and the same second stage difficulty distribution. The only variable between the tests was the content of the first stage. The experiments revealed that the first stage did impact the final accuracy of the classifier. The performance advantage of a simple first stage did not vanish in later stages of training. This might indicate that the examples presented early exert a larger influence on the outcome than the examples presented later in the optimization process. This is also evident from results of anti-curriculum learning, which converged to a generalization accuracy worse than that of the baseline.

However, it is unclear whether the approach of simply excluding inconsistent labels is better than delaying the presentation of inconsistent labels. Compared to data cleansing methods, curriculum learning is a safer approach since it does not accidentally remove correct examples that are difficult to predict. The switch between stages also resulted in an immediate and significant increase in test loss. This was alleviated by gradually mixing in harder examples.

There is also uncertainty about whether the advantage of curriculum learning is present for very large training sets. The size of the training sets was limited to around 200000 examples because of runtime concerns. Experiments did show that curriculum learning can be advantageous for domains where a limited amount of data is available. Increased accuracy with curriculum learning was observed when the teacher classifier had been trained with a training set of limited size. The gradual curriculum learning approach even outperformed the teacher classifier in terms of precision and recall.

In conclusion, improved precision and recall were observed for both aerial image datasets when using curriculum learning. The curriculum strategy of measuring inconsistency between a label and a prediction works well, and can easily be combined with deep learning.

5.3 Contributions

The thesis sought to test approaches for dealing with noisy labels in real-world datasets. This is a compelling inquiry in the field of machine learning, where the trend of using deep neural networks with a huge number of adjustable parameters requires large training sets to generalize well. There is an abundance of existing data available online that can be used for learning. Unfortunately, this data can, in many cases, lack accurate labels for supervised training. To manually label the data can be expensive and very time-consuming in many domains, such as transcribing speech for speech recognition, and tracing ground truth for semantic segmentation. Automatically generating datasets from existing data sources are a quick and economical solution, but can result in datasets with a lot of label noise.

The problem with label noise was therefore tackled in this thesis by testing two different methods: bootstrapping and curriculum learning. Bootstrapping modifies the loss function in order to reduce the impact of inconsistent labels. Curriculum learning, on the other hand, modifies the training regime by sorting the training set into stages from "easy" to "hard" examples. The sorting mechanism, or curriculum strategy, is based on measuring inconsistencies between labels and teacher predictions. Coincidentally, "hard" examples often have inconsistent labeling, and are therefore more likely to be presented at a later stage of optimization if the dataset is organized according to this curriculum strategy. In effect, inconsistent examples are a less frequent occurrence in the first stage of the curriculum dataset.

The curriculum strategy can most likely be applied to tasks in other domains, since the example difficulty estimator $d(y, q)$ does not rely on any intrinsic features of image data. However, the effectiveness of this curriculum strategy has not been verified in other domains than road detection in aerial images.

The thesis found that bootstrapping did not show much robustness towards omission noise compared to the baseline. The effect was not of any statistical significance. Furthermore, the base network also performed surprising well for very high rates of omission noise.

Curriculum learning demonstrated consistently improved generalization accuracy in the experiments. The improved accuracy was observed both for the Massachusetts Roads Dataset, as well as for the Norwegian Roads Dataset. The experiments also showed that adjusting the example distribution in the first stage of the training set affected the final outcome of the training procedure.

5.4 Future Work

This section presents future work, such as how to further verify the proposed curriculum strategy and bootstrapping loss function. In addition, this section will suggest further improvements of the road detection system.

The most compelling improvement of the road detection system is by incorporating CRF or a post processing neural network, as described in Section 2.3. This was tested by Kluckner et al. [2010] and Mnih and Hinton [2012], and yielded accuracy improvements. These methods can also result in a smoother image segmentation.

To use the road predictions further in GIS applications the binary prediction images should be converted into road centerline vectors. To do this, the prediction images have to be combined, and the resulting segmentation image must be cleaned. Methods from computer vision might be applicable for this work. For instance in [Song and Civco, 2004], shape descriptions were extracted from the segmentation images, which can be used to measure density and shape index. Based on these values, shapes, which have measurements not characteristic of roads, can be removed by a threshold operation. In addition, morphological operations, such as thinning or skeletonization, can be applied to reduce the segmented road regions down to one pixel thick lines.

Another potential improvement of the results is by finding better hyperparameters. For instance, the experiments testing the performance of the road detection system showed that the model capacity was initially constrained. Increasing the number of adjustable weights in the network improved the precision and recall breakeven point substantially. Combining dropout with max-norm regularization instead of L2 weight decay could also be interesting, since this configuration achieved better test classification error in [Srivastava et al., 2014].

The bootstrapping methods did not improve performance significantly. This might be related to ill-suited parameters. Further testing of parameter configurations could be considered. In addition, the bootstrapping method could be tested with increasing levels of registration noise. The confident bootstrapping loss function could also be explored further. An interesting comparison of the bootstrapping methods is mapping the relationship between a decreasing parameter β and their resulting performances. Based on the experiments conducted, it seems that confident bootstrapping might be less sensitive to decreasing the β_{min} parameter.

There are also some unresolved questions regarding curriculum learning, and the proposed curriculum strategy. For instance, is it worth including examples with a very high difficulty estimate? How do the data cleansing approaches described in Section 1.1 compare to curriculum learning? The thesis also does not properly determine how experienced a teacher classifier has to be in order to create an effective curriculum dataset. Further tests could illuminate the relationship between the teacher classifier's competency and the effectiveness of the curriculum strategy.

The curriculum learning approach should also be tested on a network trained on a very large dataset. This could alternatively be tested by mapping the impact of curriculum learning for increasing training set sizes. The proposed strategy should also be compared to SPL [Kumar et al., 2010], which internalizes the curriculum learning mechanism in its loss function.

Furthermore, Jiang et al. [2014] illustrated the need for balancing diversity and easiness in curriculum learning. The SPL approach is extended by a preference for both easy and diverse examples. This could potentially be done for the proposed curriculum strategy as well. For instance, unsupervised learning techniques, such as clustering, could organize the image patches into groups based on their similarity. The curriculum dataset can then be constructed with an equal representation of every cluster group. This might allow a reduction of the difficulty threshold D_0 without negatively impacting the performance.

The current version of the curriculum strategy was effective for two different datasets containing aerial imagery. However, the method should be tested on tasks in other domains as well. This might properly determine whether the proposed curriculum strategy can be generally applicable.

Appendices

A System Instructions

In order to run the core system the following dependencies are required:

- Python 2.7
- Theano
- Numpy
- Unirest
- Python Imaging Library (PIL)

The system has only been tested with Ubuntu 14.04, but it should be possible to run on both Windows and Linux as long as the listed dependencies have been installed. Ubuntu is highly recommended because of a more convenient installation process.

A Nvidia GPU is also highly recommended for running the system. In most instances a GPU can give considerable speed improvements when training compared to a CPU. This is critical when having to deal with large datasets and models with millions of parameters. In order for Theano to efficiently use your GPU while training, CUDA Toolkit has to be installed.

A graphical user interface can also be utilized for monitoring the training. Running experiments can be stopped from this user interface, as well as a debugging option which displays examples and model predictions. In addition all experiment data and results are stored as JSON, and can be viewed in the user interface. This includes, a loss per epoch plot, precision and recall curve and hyperparameter configuration.

The monitoring system can either be run locally, or installed on a server. All communication between the core system and the monitoring system is done by HTTP messaging. To enable monitoring, *enable_gui* must be set to true in the core system's config file. Furthermore, the url to the monitoring system's api has to be set for the *endpoint* parameter. Utilizing this monitoring system requires these dependencies:

- Node.js
- MongoDB

For installation instructions, please read the included README files of the repositories. The URL of these repositories are listed below:

- https://github.com/olavvatne/CNN
- https://github.com/olavvatne/ml-monitor

B Experiment Tools Overview

Important tools that were created for this thesis are listed below. The source code can be found inside the tools module of the road detection system's repository.

- measurement.precisionrecall.py
  The tool creates the precision and recall curve. Command line options can be supplied.
- visualize.aerial.py
  The tool stitches together model predictions and creates the prediction error images seen in the qualitative analysis.
- layer.visualize.py
  Opens a params.pkl file containing the weights and hyperparameter configuration of a trained network. It creates a visualization of the kernels in the network's input layer.
- distribution.curriculum_diff.py
  Samples patch examples and creates a histogram showing the patch dataset difficulty estimate distribution. Useful for setting the threshold D_θ when conducting experiments. Also useful for verifying the content in a curriculum patch dataset stage.
- distribution.dataset_std.py
  Tool for finding an estimate of a dataset's standard deviation. This value is used by the contrast normalization in the pre-processing step.

- distribution.label_dist.py
  Counts the percentage of true label pixels in a dataset.

- curriculum.dataset_create.py
  Tool for pre-generating a curriculum patch dataset. Command line options can be supplied to select a teacher, the thresholds D_θ, the teacher's optimal threshold value, and dataset. The tool comes with a baseline option that generates a staged patch dataset without curating the content of each stage.

- distribution.statistics.py
  The tool performs the Welch's t-test between two populations and Shapiro-Wilk normality test for each population. The tool also displays normal Q-Q plots generated from the samples of the experiment runs.

- figure.average_compare.py
  Averages experiment runs, and plots the averaged MSE loss and precision and recall curve from the test dataset. The tool also marks the precision and recall breakeven point of each plot. The resulting figures are used for comparison purposes in this thesis.

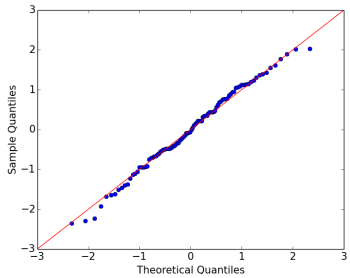
- figure.average_loss.py
  Averages the test, validation and training loss of experiment replicates and plots the result in a loss per epoch plot.

- figure.average_noise_levels.py
  Averages the test losses of experiments' final epoch. It also averages the precision and recall breakeven point of each noise level. The figures created by this tool shows the MSE test loss and breakeven point over increasing levels of label noise.

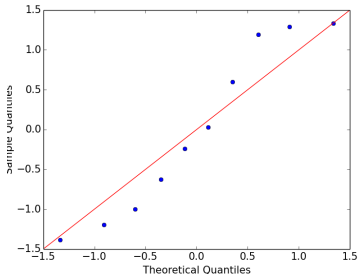
C Experiment Population Normality Assumption

The Welch's t-test assumes that the samples are independent, and drawn from an approximately normal distributed population. In this appendix, the assumption of normality is explored further.

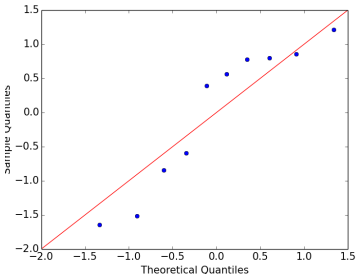
One way of verifying that a population does not violate the normality assumption is to create a normal Q-Q plot. These plots plot each sample based on its quantile and the corresponding theoretical quantile expected from a normal distribution. The population samples are normally distributed if the plotted points approximately fit a 45 degree line. The normal Q-Q plots comparing the test loss populations of *Experiment E1 - 0% omission noise* to normal distributions can be found in Figure 1c and Figure 1d. The plots are clearly affected by sample



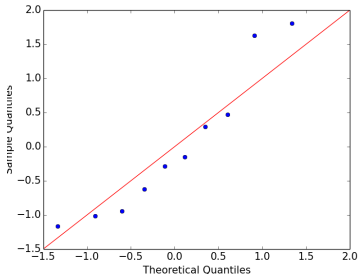
(a) 100 samples from a normal distribution.



(b) 10 samples from a normal distribution.



(c) Baseline samples from Experiment E1.



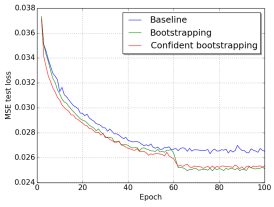
(d) Bootstrapping samples from Experiment E1.

Figure 1: Normal Q-Q plots generated by samples from a normal distribution and from the baseline and curriculum population of Experiment E1 - 0% omission noise.

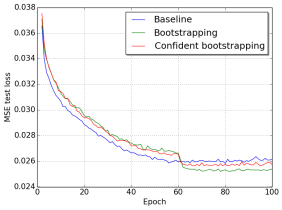
variability caused by the small sample sizes. This can be seen in Figure 1b, where samples randomly picked from a normal distribution are plotted. This plot is created from 10 randomly drawn samples from a normal distribution with mean and variance equal to that of the baseline population. Figure 1a displays the normal Q-Q plot of 100 random samples drawn from a similar normal distribution. This plot fits the line more closely than 1b. Because of the small sample sizes, it can be hard to confidently assess whether the experiment populations are normally distributed or not. At least there does not seem to be any compelling evidence in the available sample data to suggest that the populations are clearly deviating from a normal distribution. This was also confirmed by the Shapiro-Wilk normality test.

D Experiment E2 Results

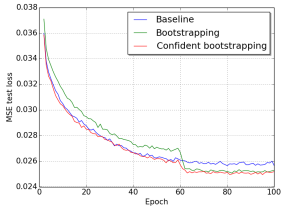
The results from Experiment E2 were summarised by plotting the final test loss and precision and breakeven point for increasing levels of omission noise. In this appendix, the test loss figures for every noise rate is shown in Figure 2, while the precision and recall curve figures are depicted in Figure 3.



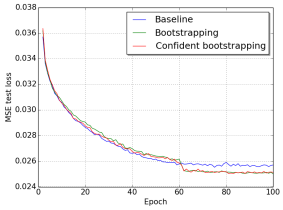
(a) 0%



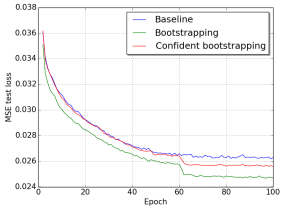
(b) 10%



(c) 20%

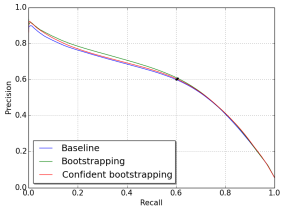


(d) 30%

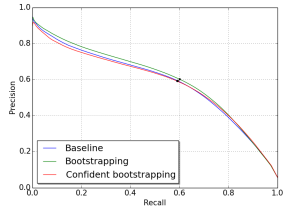


(e) 40%

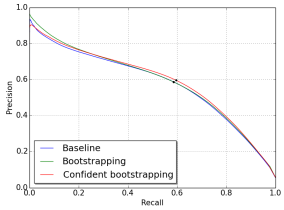
Figure 2: E2 - Test loss comparisons for several levels of omission noise.



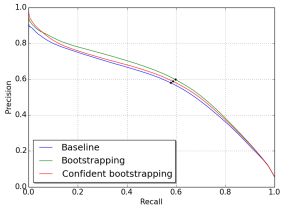
(a) 0%



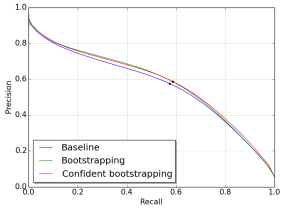
(b) 10%



(c) 20%



(d) 30%



(e) 40%

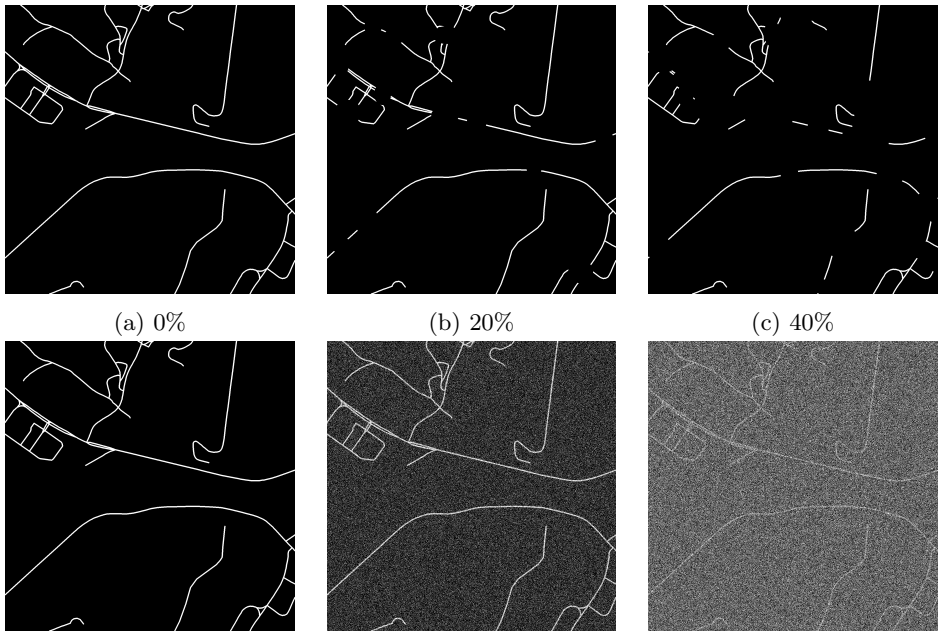
Figure 3: E2 - Precision and recall plots for several levels of omission noise.

E Random Noise Experiment

This appendix presents an additional experiment that tested the robustness of the different loss functions on labels with increasing levels of random noise. The noise was artificially added by flipping random label pixels until a certain percentage of label pixels had been flipped. The bottom row of Figure 4 displays a label image with different levels of random noise. Furthermore, comparable omission noise examples can be found in the top row of this figure.

Networks were trained on 221600 examples for 100 epochs, with a learning rate of 0.0025, no curriculum learning, number of kernels in first layer $K_0 = 64$, and a batch size of 64. Apart from that, the network architecture is similar to that of Network M2 in Experiment E7.

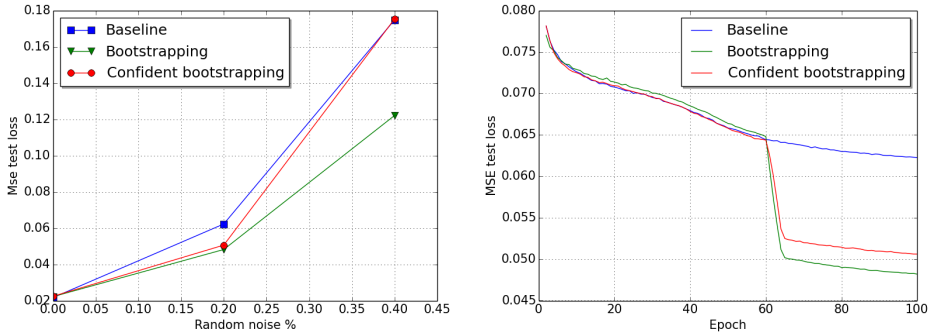
The results in Figure 5a, show a steeper increase in test loss for increasing levels



(a) 0% (b) 20% (c) 40%

(d) 0% (e) 20% (f) 40%

Figure 4: Examples of different levels of artifical noise added to a label image. The top row shows increasing levels of omission noise, while the bottom row illustrates increasing levels of random label flipping noise.



(a) Test loss for increasing label noise (b) Test loss per epoch at 20% label noise

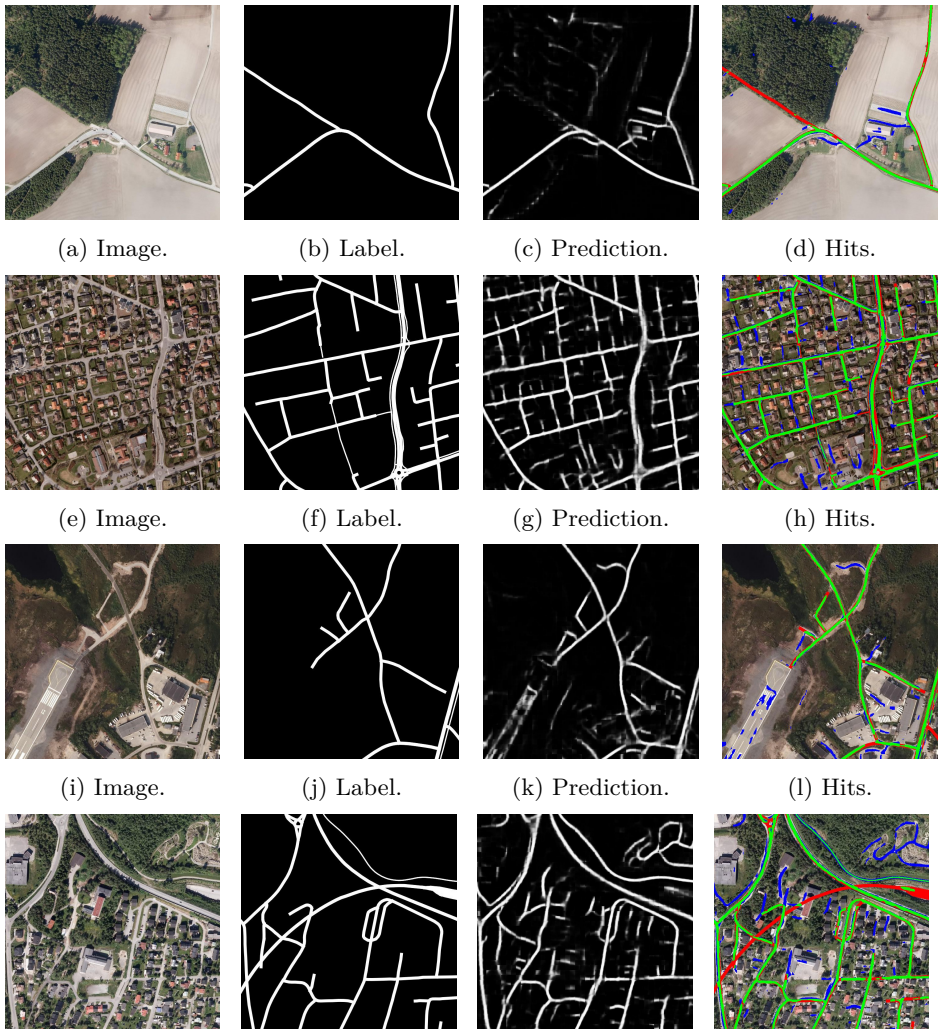
Figure 5: The test loss results of artificially increasing the level of random noise in labels.

of noise, compared to Experiment E1 and E2. With no random label noise, the baseline performed slightly better than the bootstrapping methods. However, the test losses at 20% and 40% show that bootstrapping has a considerably lower test loss compared to the baseline. The gap in test loss also seems to be increasing. The confident bootstrapping has a test loss comparable to bootstrapping at 20% label noise. However, this test loss improvement is not maintained towards 40% label noise.

In Figure 5b, test loss per epoch plots are displayed. These show the averaged training run achieved at 20% label noise. The bootstrapping plots and the baseline have similar results until epoch 60. At epoch 60, the test loss of both bootstrapping methods decreases rapidly, to a level well below the baseline. This epoch is also when the β parameter was gradually decreased from 1.0 to 0.8. The results shows that bootstrapping seems to provide robustness against random label flipping. This is evident from the decrease in test loss after epoch 60 when the loss function starts modifying the labels using the model predictions. Figure 5a also shows an increasing gap between the test losses of bootstrapping and the baseline. Whereas the results of Experiment E1 and E2 were inconclusive, these results are significant. However, this type of label noise is unrealistic in the aerial image datasets used in this thesis. The label maps are often generated from existing map data. This process would not produce this type of label noise. however, this experiment suggests that bootstrapping can be applicable in some domains.

F Road Detection Results

In this appendix results from the best performing convolutional neural networks are displayed. The precision and recall breakeven point of the model trained on the Massachusetts Roads Dataset is 0.8627. Whereas, the breakeven point of the best model trained on the Norwegian Roads Dataset is 0.7620.



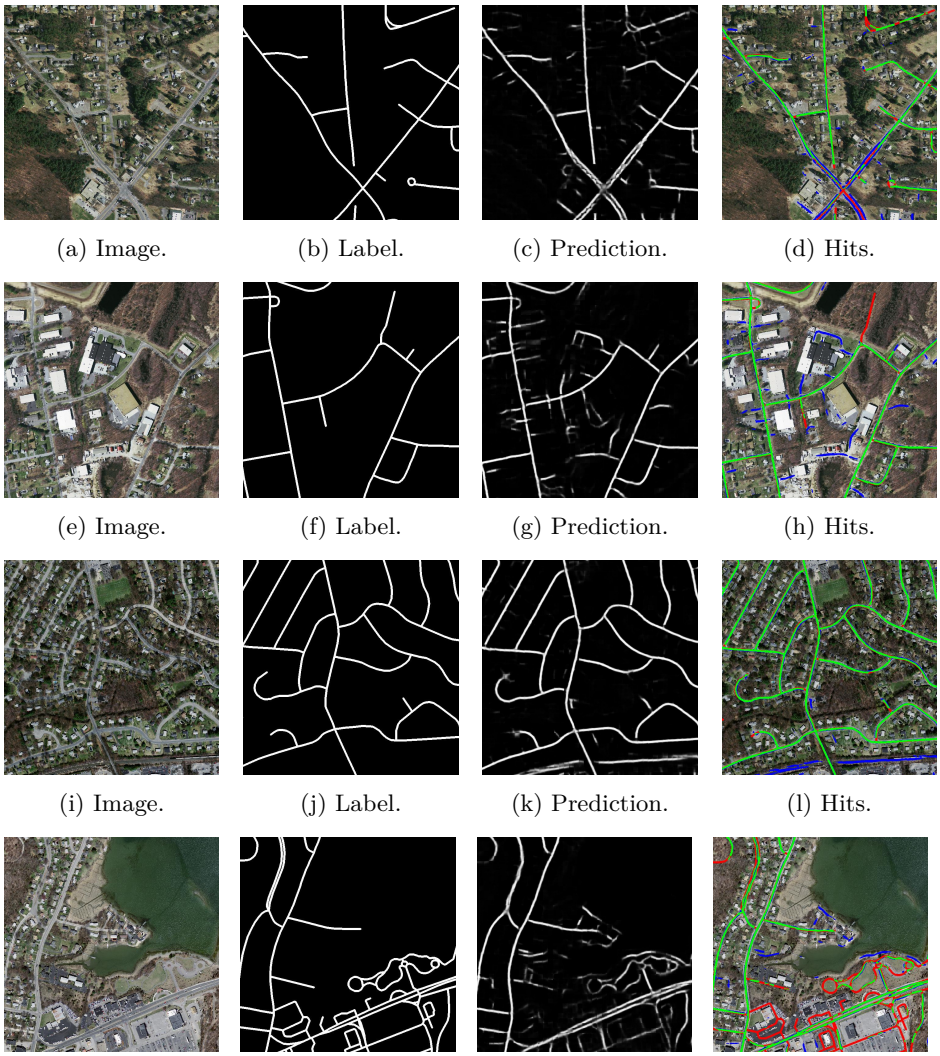
(a) Image. (b) Label. (c) Prediction. (d) Hits.

(e) Image. (f) Label. (g) Prediction. (h) Hits.

(i) Image. (j) Label. (k) Prediction. (l) Hits.

(m) Image. (n) Label. (o) Prediction. (p) Hits.

Figure 6: Road extraction results from the Norwegian Roads Dataset.



(a) Image. (b) Label. (c) Prediction. (d) Hits.

(e) Image. (f) Label. (g) Prediction. (h) Hits.

(i) Image. (j) Label. (k) Prediction. (l) Hits.

(m) Image. (n) Label. (o) Prediction. (p) Hits.

Figure 7: Road extraction results from the Massachusetts Roads Dataset.

Bibliography

Alvarez, J., LeCun, Y., Gevers, T., and Lopez, A. (2012). Semantic road segmentation via multi-scale ensembles of learned features. In Fusiello, A., Murino, V., and Cucchiara, R., editors, *Computer Vision – ECCV 2012. Workshops and Demonstrations*, volume 7584 of *Lecture Notes in Computer Science*, pages 586–595. Springer Berlin Heidelberg.

Bengio, Y., Boulanger-Lewandowski, N., and Pascanu, R. (2013). Advances in optimizing recurrent networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8624–8628. IEEE.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*. Oral Presentation.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

Breve, F. A., Zhao, L., and Quiles, M. G. (2015). Particle competition and cooperation for semi-supervised learning with label noise. *Neurocomputing*, 160:63–72.

Cohn, D. A., Ghahramani, Z., and Jordan, M. I. (1996). Active learning with statistical models. *Journal of artificial intelligence research*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157.

Dollar, P., Tu, Z., and Belongie, S. (2006). Supervised learning of edges and object boundaries. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1964–1971. IEEE.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660.

Frénay, B. and Verleysen, M. (2014). Classification in the presence of label noise: a survey. *Neural Networks and Learning Systems, IEEE Transactions on*, 25(5):845–869.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep learning. Book in preparation for MIT Press.

Google (2013). Project ground truth: Accurate maps via algorithms and elbow grease. `https://developers.google.com/events/io/2013/sessions`. Accessed: 2016-07-05.

Hinton, G. (2014). Neural networks for machine learning, lecture note 9b - limiting the size of the weights. `http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec9.pdf`. Accessed: 2016-04-14.

Japkowicz, N. (2000). The class imbalance problem: Significance and strategies. In *Proc. of the Int'l Conf. on Artificial Intelligence*. Citeseer.

Jiang, L., Meng, D., Yu, S.-I., Lan, Z., Shan, S., and Hauptmann, A. (2014). Self-paced learning with diversity. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 27*, pages 2078–2086. Curran Associates, Inc.

Kartverket (2001). N50 topographic vector database. `https://kartkatalog.geonorge.no/metadata/kartverket/n50-kartdata/ea192681-d039-42ec-b1bc-f3ce04c189ac`. Accessed: 2016-04-05.

Kartverket (2006). Vbase road centerline vector database. `https://kartkatalog.geonorge.no/metadata/kartverket/vbase/96104f20-15f6-460e-a907-501a65e2f9ce`. Accessed: 2016-14-04.

Khan, F., Mutlu, B., and Zhu, X. (2011). How do humans teach: On curriculum learning and teaching dimension. In Shawe-Taylor, J., Zemel, R., Bartlett,

P., Pereira, F., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 24*, pages 1449–1457. Curran Associates, Inc.

Kluckner, S., Mauthner, T., Roth, P. M., and Bischof, H. (2010). Semantic classification in aerial imagery by integrating appearance and height information. In *Computer Vision–ACCV 2009*, pages 477–488. Springer.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kumar, M. P., Packer, B., and Koller, D. (2010). Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197.

Mayer, H., Hinz, S., Bacher, U., and Baltsavias, E. (2006). A test of automatic road extraction approaches. *International Archives of Photogrammetry, Remote Sensing, and Spatial Information Sciences*, 36(3):209–214.

Mena, J. B. (2003). State of the art on automatic road extraction for gis update: A novel classification. *Pattern Recogn. Lett.*, 24(16):3037–3058.

Mnih, V. (2013). *Machine Learning for Aerial Image Labeling*. PhD thesis, University of Toronto.

Mnih, V. and Hinton, G. E. (2010). Learning to detect roads in high-resolution aerial images. In *Proceedings of the 11th European Conference on Computer Vision: Part VI*, ECCV'10, pages 210–223, Berlin, Heidelberg. Springer-Verlag.

Mnih, V. and Hinton, G. E. (2012). Learning to label aerial images from noisy data. In Langford, J. and Pineau, J., editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 567–574, New York, NY, USA. ACM.

Mokhtarzade, M. and Zoej, M. V. (2007). Road detection from high-resolution satellite images using artificial neural networks. *International journal of applied earth observation and geoinformation*, 9(1):32–40.

Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., and Rabinovich, A. (2014). Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*. Accepted as a workshop contribution at ICLR 2015.

Ringach, D. L. (2002). Spatial structure and symmetry of simple-cell receptive fields in macaque primary visual cortex. *Journal of neurophysiology*, 88(1):455–463.

Rosenberg, C., Hebert, M., and Schneiderman, H. (2005). Semi-supervised self-training of object detection models. In *Proceedings of the Seventh IEEE Workshops on Application of Computer Vision (WACV/MOTION'05) - Volume 1 - Volume 01*, WACV-MOTION '05, pages 29–36, Washington, DC, USA. IEEE Computer Society.

Saito, S. and Aoki, Y. (2015). Building and road detection from large aerial imagery. In *IS&T/SPIE Electronic Imaging*, pages 94050K–94050K. International Society for Optics and Photonics.

Schindler, K. (2012). An overview and comparison of smooth labeling methods for land-cover classification. *Geoscience and Remote Sensing, IEEE Transactions on*, 50(11):4534–4545.

Song, M. and Civco, D. (2004). Road extraction using svm and image segmentation. *Photogrammetric Engineering & Remote Sensing*, 70(12):1365–1371.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Sukhbaatar, S. and Fergus, R. (2014). Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*. Accepted as a workshop contribution at ICLR 2015.

Trinder, J. (2009). Towards automation of information extraction from aerial and satellite images. In Li, D., Shan, J., and Gong, J., editors, *Geospatial Technology for Earth Observation*, pages 289–327. Springer US.

Vegdirektoratet (2014). *Håndbok N100: veg- og gateutforming*. Vegdirektoratet.

Walpole, R. E., Myers, R. H., Myers, S. L., and Ye, K. (2011). *Probability and statistics for engineers and scientists*. Pearson Education, 9th edition.

Wiedemann, C., Heipke, C., Mayer, H., and Jamet, O. (1998). Empirical evaluation of automatically extracted road axes. In *Empirical Evaluation Techniques in Computer Vision*, pages 172–187.