Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Olav Reppe Husby

Traffic sign anomaly detection with unsupervised learning

Master's thesis in Informatics
Supervisor: Helge Langseth
June 2020

NTNU
Norwegian University of Science and Technology

Olav Reppe Husby

Traffic sign anomaly detection with unsupervised learning

Master's thesis in Informatics
Supervisor: Helge Langseth
June 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Abstract

Currently, Statens Vegvesen (The Norwegian Public Roads Administration) (SVV) and other road agencies employ a vast number of educated individuals to perform manual inspection work to detect anomalies in road infrastructure. This thesis aims to investigate the use of unsupervised anomaly detection to alleviate the workload associated with this task.

Real-world data is often noisy, has wrong labels or no labels at all, and presents many challenges. This thesis tackles this problem by developing an unsupervised pipeline that detects traffic signs, uses them for training, and finds anomalies in real-world data. This hopes to aid the development of a machine learning solution to be utilized by road authorities in Norway to detect anomalies in various traffic-related objects. The model presented in this thesis achieves a ROC-AUC score of 0.92.

The results show that developing an anomaly detection system for use by road authorities to ease the manual labor involved is possible with a high degree of accuracy. It also shows that this is possible utilizing only unlabeled, real-world data, with little human interference.

Sammendrag

Nåværende har Statens Vegvesen (SVV) og andre veg-etater ansatt et stort antall utlærte individer for å gjøre manuell befaring av veisystemet. Denne oppgaven utforsker bruken av uovervåket avviks-deteksjon for å redusere den manuelle arbeidsmengden dette fører til.

Data i den virkelige verden inneholder ofte støy, har feil merking, eller ingen merking i det hele tatt, som fører til mange utfordringer. Denne oppgaven takler dette problemet ved å utvikle et uovervåket pipeline som oppdager trafikkskilt, bruker de til trening, og finner avvik i disse skiltene. Dette i håp om å hjelpe utviklingen av et maskinlæring system som i framtiden kan brukes av veg-etater i Norge for å finne avvik i forskjellige veg-relaterte objekter. Modellen presentert i denne oppgaven oppnår en ROC-AUC verdi på 0.92.

Resultatet viser at å utvikle et avviks deteksjonssystem til bruk av veg-etater for å redusere manuell arbeidsbruk er mulig med høy nøyaktighet. Resultatet viser også at dette er mulig med kun umerket, ekte data, med lite menneskelig innblanding.

Preface

This thesis was written as the author's master thesis at the Faculty of Information Technology and Electrical Engineering (IE), Department of Computer Science (IDI), Norwegian University of Science and Technology (NTNU).

The assignment is in collaboration with Kantega and Statens Vegvesen (The Norwegian Public Roads Administration) (SVV).

I want to thank my supervisor Professor Helge Langseth for his guidance, my contact at SVV Johan Wåhlin for being a great help and source of information, without the involvement of him and SVV this thesis would not have been possible. I would also like to thank my contact at Kantega Edvard Neset Karlsen for his dedication to the project and continual insights and expertise. Finally, I would like to thank my friends at NTNU and colleagues at Kantega for their support.

A special thanks to NTNU HPC for allowing me to utilize their vast CPU and GPU resources to complete the computationally heavy parts of this thesis.

Olav Reppe Husby
Trondheim, June 10, 2020

Contents

Acronyms

ANN Artificial Neural Network. 5, 13

AUC Area under curve. i, 11, 31, 37, 54, 55

CNN Convolutional Neural Network. 8, 9, 13, 17, 18

FPR False positive rate. 11, 12, 37, 45–47

FPS Frames Per Second. 24, 25, 31, 57

LSTM Long short-term memory. 57

MSE Mean Square Error. 2, 14, 31, 34, 37, 40, 44, 48

NVDB Nasjonal vegdatabank (National Road Database). 35, 56

ReLU Rectified Linear Unit. 34

ROC Receiver operating characteristic. i, 11, 12, 31, 37, 46, 49, 51, 54, 55

SVM Support Vector Machines. 9

SVV Statens Vegvesen (The Norwegian Public Roads Administration). i, v, 1, 23, 26, 30, 32, 35–40, 45, 53, 55, 56

TPR True positive rate. 11, 12, 37, 45–47

VAE Variational Autoencoder. 28

YOLO You Only Look Once. 23, 24

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

Most developed countries have a vast network of roads that needs to be maintained. To this end, they have employed many people to oversee the quality and continued usability of the current road infrastructure. This creates a high cost in repetitive work, which is a prime candidate for automation.

SVV is the agency that has the responsibility of creating and maintaining the road network in Norway. They accomplish this task by leasing contracts to subcontractor companies in Norway. After a contract has been assigned, SVV needs to inspect the contractors' work to make sure they fulfill their obligations. SVV employs a large number of educated individuals to inspect contractor work, and in this thesis, inspecting traffic signs will be the main focus.

The work of these inspectors consists of being assigned a contract area and driving these roads with a list of assignments to perform, such as inspecting traffic signs, road conditions, guard rails, cleaning, etc. The task is to evaluate whether or not the contractors are fulfilling their end of these maintenance contracts. Many of the entries in this list are visual inspection tasks, that could potentially be automated or otherwise aided by modern computer vision methods.

In recent years there has been a substantial increase in the availability of Machine Learning solutions for video analysis. Solutions such as self-driving cars, automatic product inspection on assembly lines, and object detection networks have become more prevalent in recent years.

SVV has preemptively collected large quantities of data from its inspectors using dashcams. However, they have not yet utilized this data in any production capacity. This thesis aims to use this data to find anomalies in traffic signs present in these videos.

1.2 Goals and Research Questions

Goal *Assess the viability of computer vision for anomaly detection in traffic signs.*

The thesis will investigate how to utilize computer vision to detect anomalies in traffic signs by considering the two research questions defined below. A system will be created that will attempt to solve this problem, and experiments will detail the performance of this system. To be able to address this goal, the following research questions are required.

RQ 1 *Which modern methods are suited for detecting anomalies in images of traffic signs?*

To create such a system, a suitable method has to be researched and chosen. The method has to be suitable for images, specifically images of traffic signs.

RQ 2 *To which degree is it feasible to detect anomalies in images of traffic signs extracted from dashcam video?*

The thesis will also investigate the viability of a system created from methods discovered in **RQ 1**. The system will be created to detect anomalies, and its performance will be measured.

1.3 Contributions

This thesis will establish a model's ability to detect anomalies in traffic signs utilizing real-world data. The thesis will also establish a pipeline of object detection and anomaly detection, which might be an important component in creating a production-ready system to tackle real-world problems using approaches discussed in this thesis.

This thesis will showcase the following contributions:

1. A working pipeline that can go from a video to a list of potentially anomalous traffic signs.
2. A model that can detect anomalies in real Norwegian traffic signs.
3. A comparison between the model presented, a MSE model, and a template-based model when applied to the same data.

1.4 Thesis Structure

Chapter 1 Introduction:

This chapter provides an overview of this thesis's contents: the goals, contributions, and motivation.

Chapter 2 Background Theory:

This chapter provides necessary background theory which relates to the problem space and is necessary to understand the rest of the thesis.

Chapter 3 State Of The Art:

This chapter provides information on state-of-the-art techniques used in similar papers and problem spaces, which provides a background to the choices made in the practical part of this thesis.

Chapter 4 Architecture/Model:

This chapter explains the methods and choices made in this thesis to construct a prototype solution to address the main topic.

Chapter 5 Experiments and Results:

This chapter highlights the experiments performed and the results gathered from the system explained in the previous chapter.

Chapter 6 Evaluation and Conclusion:

This chapter evaluates the results and draws conclusions based on data gathered. It also highlights the contributions and explains how it could improve via further work.

Chapter 2

Background Theory

This chapter details necessary background theory for the thesis, necessary to understand the remainder of the thesis, the terminology and methods used.
In section 2.1, topics such as machine learning paradigms, object detection, and anomaly detection are introduced with insights into commonly used methods and approaches. In section 2.2, Artificial Neural Networks are presented in-depth with associated architectures, challenges, and components.

2.1 Machine Learning

Machine learning is a branch of artificial intelligence that has gained traction in the recent decade. The overarching goal of machine learning is to learn how to perform complex actions, perform intelligently based on available data, and uncover previously unknown relations in unstructured data.

Machine learning is being employed in a larger degree by businesses for a wide array of applications and has seen success as a practical application for artificial intelligence. Solutions such as voice assistants, autonomous driving systems, recommendation systems, smart-home solutions, image, and video analysis have become widespread in recent years, partly due to advancements in machine learning.

Machine learning aims to learn a model to predict unseen results by utilizing previous data; as opposed to traditional algorithmic methods or so-called "expert systems", machine learning systems do not contain an explicit modeling component that has to be handcrafted for each application. In this sense, machine learning allows for systems where constructing rules or models which produce the same results would be too time-consuming, or otherwise infeasible. Machine learning allows us to produce models for systems where no algorithmic solutions

are available, such as image recognition.

Machine learning is usually defined as the process of improving at task T, with regard to performance measure P, utilizing experience E.

In essence, this boils down to creating a function $\hat{f}$ that should produce the same output y as the underlying function f. This can be done by using a performance measure to rate the performance of the function, which will be created by supplying experience.

An example of a machine learning task is to predict the temperature tomorrow. In this example, the task T is to predict the temperature. A typical performance measure P for this task would be the difference from the observed temperature and the predicted temperature. Finally, the experience E to learn from would be historical temperature data, which will be used to learn the model. Without machine learning, to construct such a system, one would have to craft custom models to dictate the predictions of the systems. This could require extensive manual statistical analysis and expert assistance.

Machine learning was developed as a field of research for several decades with the goal of allowing computers to learn automatically without human assistance. However, it is only in recent years that machine learning has managed to see better than human results and production usage by large corporations. Large corporations such as Google and Facebook have become some of the most significant contributors to machine learning research, such as the impactful paper on AlphaZero by Google (Silver et al. [2017]).

2.1.1 Paradigms

Supervised Learning

Supervised learning is the process of learning a mapping in which the input $x \in \chi$ is mapped to the output space $y \in \gamma$. This mapping is done via a function $f(x) = y$ in which $f(\cdot)$ is unknown. The goal of supervised learning is to learn this function by receiving input values $\boldsymbol{x_1, ..., x_n}$ and matching target values $\boldsymbol{y_1, ..., y_n}$ that can be utilized to estimate the function numerically. Once the model is learned, it can be utilized to estimate examples that have not been seen before, if the model is trained well.

Supervised learning functions by initially guessing a function $\hat{f}$, which is applied to the input x_n and its output compared to the target value y_n. Based on the difference between the output value and target value, the function is corrected to make the output value closer to the target value. This process continues for each $x \in \chi$ until the function produces the desired results. Doing this, supervised learning hopes that the model $\hat{f}$ will approximate the unknown function f, allowing the model to behave as this hypothetical underlying function would.

Unsupervised Learning

Unsupervised learning is the case where the target value of each example is not defined, meaning the dataset contains only input values $\boldsymbol{x_1, ..., x_n}$, and no target values. The goal of unsupervised learning is to uncover hidden relations or clusters within the data. The most common form of unsupervised learning is clustering, in which one attempts to uncover *clusters*, which groups similar entries in the dataset. There are many types of clustering algorithms, such as k-means clustering and hierarchical clustering, however, the basic objective remains the same.

Clustering methods attempt to separate data points in an n-dimensional space, n being the number of features contained in each data point. If a data point has two features, x and y for example, the data could be plotted in a two dimensional plane, which allows the algorithm to separate clusters of data contained in the same general areas in the plane. These methods generally rely on accurate distance metrics used to calculate the distance or similarity between entries in the data. Common metrics in this field include the Manhattan distance; in which the distance between data is judged via the formula $d_{(x,y)} = \sum_{i=1}^{n}(|x_i - y_i|)$ and the Euclidean distance, where the formula is $d_{(x,y)} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$. However methods which do not utilize these metrics directly also exist, such as neural

models like Self-organizing maps.

Reinforcement Learning

Reinforcement learning can be utilized in situations where decisions need to be made sequentially. The agent is not given a target value for each action such as in supervised learning; the agent is however given reward if its decisions end up in a positive outcome.

Reinforcement learning is often utilized in scenarios where the desired outcome is known, but the values of the individual actions leading to said outcome are unknown. The agent is then given infrequent rewards when training, to incentivize it to perform actions which lead to a positive outcome. A typical example of use-cases for reinforcement learning is the game chess. We can identify positive outcomes of decisions (winning the game), however, we are unable to provide target values for every possible move, either because of computational infeasibility or other factors. An example of successfully implemented reinforcement learning is the Google agent AlphaZero (Silver et al. [2017]), which utilizes reinforcement learning by playing against itself, or the Dota 2 system by OpenAI (OpenAI et al. [2019]) which was able to beat top teams in a game with imperfect information by playing against itself repeatedly.

2.1.2 Object Detection

Object detection is a problem area within computer vision in which the task is to detect some sort of object in an image. The type of object to detect depends on the needs of the system. Examples include detecting cars, signs, faces and letters.

Historically, object detection was done via matching parts of an image to existing examples of the object in question, however, such methods were error prone, computationally expensive, and usually manually created for each object detection task. Such methods were made mostly obsolete with the advent of Convolutional Neural Networks, which has seen great results such as the YOLO (Redmon et al. [2016]) architecture, and its improvement YOLO v3 (Redmon and Farhadi [2018]) architecture.

Sliding window

Early attempts at object detection utilized a form of sliding window comparisons with a template that slid across the image and compared each section to the template image (Viola and Jones [2004]). This approach is computationally expensive and requires several passes with different variants of the object as the template image in order to get accurate predictions. If this method is to detect

an object from any angle it would need template images from every angle as well, leading to a large number of passes. Some systems attempted to alleviate these problems by breaking down more complex object shapes into smaller "feature" templates (Papageorgiou et al. [1998]), which somewhat reduced these problems.

This method also does not do well with objects of different sizes, perspectives, or angles, making it quite inefficient (Viola and Jones [2004]). However, as it was one of the first methods developed, it at least was able to detect certain objects and perform well in scenarios where the problems mentioned above were not present. Such methodology remains useful in areas such as OCR (Optical Character Recognizing) and other domains.

Later attempts utilized a similar approach, however, this time models were created for each type of object to detect; instead of using templates, these methods used learning models such as SVMs. These methods still relied on sliding window and multiple models to combine into a complete system which could detect several types of objects (Felzenszwalb et al. [2009]).

Convolutional Neural Network (CNN)

Convolutional Neural Networks brought about a revolution in object detection; while detecting some objects was possible using the previously discussed method, this did not generalize well to detectors able to detect multiple object types at different scales. CNNs were able to mitigate these problems to some degree (Cai et al. [2016])

CNNs are effective at this task because they effectively implement the sliding window approach in its architecture while retaining the core benefits of neural networks allowing it to learn the best weights to achieve the optimal results. This approach also means there is no need for sliding windows of different sizes in this method, as the network can learn to combine multiple filters in the convolutional layers to achieve the same result automatically, resulting in a more efficient, self-learning approach which can produce best in class results with certain specialized architectures. CNNs are further discussed in detail in Section 2.2.2.

2.1.3 Anomaly Detection

Anomaly detection is the process of identifying data points in a data set which does not conform to the common structure of the data. So-called outliers are data points with sufficient difference from the norm as to warrant additional investigation. The reason for wanting to identify such anomalies can vary depending on the needs of the system utilizing such methods. Anomaly detection is commonly used for intrusion detection, fraud detection, outlier detection, and statistical analysis.

On traditional low dimensionality datasets, there exist algorithms such as KNN clustering which has been used. However, with higher dimensionality datasets such as in the realm of computer vision, such methods are computationally infeasible or simply unable to produce high-quality results (Beyer et al. [1999]).

Clustering

Clustering is an unsupervised method that aims to uncover hidden clusters or similar data groups by analysing its underlying features. This method seeks to group data based on their similarity, which also allows it do detect data that is dissimilar from other data.

Clustering methods such as DBSCAN (Ester et al. [1996]), work in essence by analyzing the density of data in a plane, then creating boundaries where the density is lower, then iteratively refining these borders to fit the data in the best possible way.

This method can also be utilized for anomaly detection, as the same similarity metrics which determine the borders of the clusters can be used to find data that has an abnormally large distance from other data-points. This method also allows for easy visualization in some instances where the data can be represented in a few dimensions, making it simple to visualize anomalies.

KNN

KNN or K-nearest neighbor is a type of supervised classification algorithm. Unlike other methods discussed in this thesis, it does not utilize neural nets or training of any kind, however, it can be seen as a benchmark to judge other methods.

The algorithm functions by selecting a number k, an integer which specifies how many neighbors to compare against, usually an odd number to prevent ties. The algorithm then compares a new data point to existing labeled data and chooses the predicted class based on the nearby points. For example, if k is 3, and the nearest points are classified as $true, true, false$, the prediction would be $true$. This method can also be used on an arbitrary amount of classes, however, this might require k to be altered to prevent ties, for example if there are 3 classes, a k value of 3 could still result in a tie, which means a different number will have to be chosen.

The method suffers from a sensitivity to outliers, and struggles to classify high dimensional data (Beyer et al. [1999]). Methods exist which alleviate this, such as dimension reduction via feature extraction. However the efficacy of the method relies heavily on the quality of the distance metrics used.

This method can be used to detect outliers in multiple ways; one can use

the distance from its neighbors as the anomaly score, with a very high distance signifying outliers.

If the true class of the data point is known, one could also compare this with its neighbors to determine if the prediction is reasonable. If the true class is partially or wildly different from the neighboring class, the point may be classified as an anomaly.

Reconstruction

Usually thought of as an unsupervised method, reconstruction works by training a model to *reconstruct* or recreate high dimensional data. The point usually is to utilize a lossy model or algorithm, one in which somewhere in the process, the model has to represent the data using less information than the original. This hopes to capture some underlying structure from the data in question, which the model will learn to reconstruct it.

The final score is usually determined by reconstruction error, a metric that measures how well the data was reconstructed. These methods in essence compare the input and output data's values, and create a score based on these. If we use image reconstruction as an example, a naive way to do this would simply be to take the difference between equally placed pixels from either image, summing these, and dividing by the number of pixels. This would provide an average pixel difference, which could be used as a metric.

As the model needs to retain less information at some stage of reconstruction, a natural choice is an autoencoding neural network, as these are defined as having exactly those properties.

2.1.4 Receiver operating characteristic

Receiver operating characteristic (ROC) is a metric used to judge a classifier's ability to separate positive and negative examples. It does this by comparing the True positive rate (TPR) and the False positive rate (FPR) at various cutoff thresholds. This is useful for determining which values to use as cutoffs for a model to achieve the desired ratio of TPR to FPR. Combining this metric with Area under curve (AUC), allows a total scoring of the models' ability to separate a random positive and a random negative example. ROC is useful as explained for determining cutoffs for classifications, which can be a powerful tool in cases of unequal class distributions (Fawcett [2006]), or in cases where the associated cost of false negative is higher than the positive gain of true positives, or the other way around.

In a confusion matrix, True positive rate (TPR), also known as recall or hit-rate, is defined as the number of true positives TP divided by the total number

of positive samples in a dataset P. This results in the formula $TPR = \frac{TP}{P}$.

False positive rate (FPR) is in a similar manner defined as the number of false positives FP divided by the number of negative samples N. Resulting in the formula $FPR = \frac{FP}{N}$.

The ROC curve can be created by calculating these values as various cutoff thresholds in the model in question. Further examples of this can be seen in the Experiments and Results chapter.

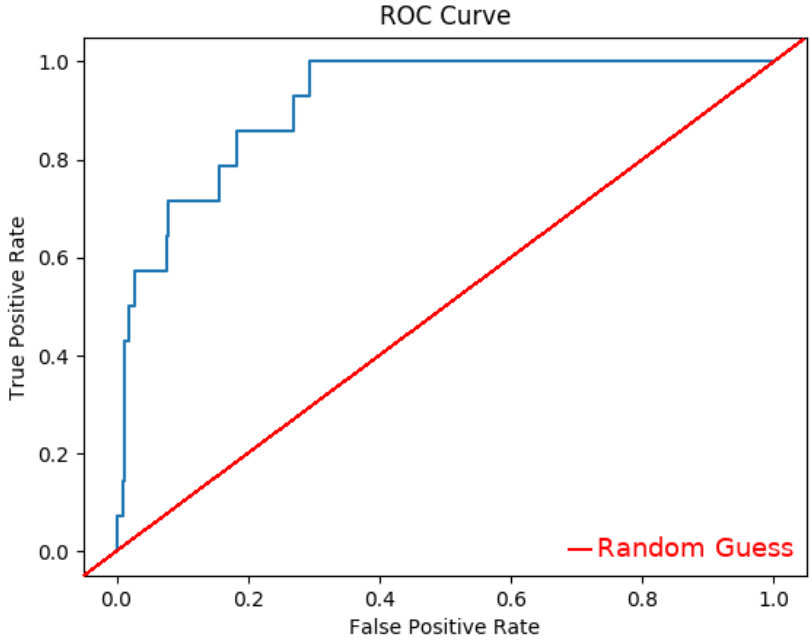


FIGURE 2.1 – ROC Curve

As figure 2.1 illustrates, the red line indicates a random guess by the model. The blue line illustrates the ratio of TPR to FPR at various thresholds. To achieve a better TPR, a higher FPR is usually the result. This is a trade-off, where the model can be configured to use a certain threshold depending on the needs of the model at hand.

2.2 Deep Learning

Deep learning is a subset of machine learning that has become more prevalent in recent years. While the difference between machine learning and deep learning is not well defined, it is commonly understood that deep learning yearns for deeper models and more abstracted features.

A common feature of deep learning is that it is able to work out which features to prioritize itself, which is something normal machine learning struggles to do. It does this by utilizing deeper models than common machine learning methods, which allows it to abstract away the input features and create its own abstracted features that might work better than using the input features directly. The thinking here is that these abstracted features allow the model to incorporate automatic feature extraction and feature engineering to gain a higher level of understanding, which is not possible with shallower networks.

Typically deep learning is implemented as a Convolutional Neural Network (CNN), or an Artificial Neural Network (ANN). A common theme with these implementations compared to traditional machine learning is that the models have more layers to accomplish the goals outlined in this section.

2.2.1 Artificial neural network

Perceptron

The perceptron (Rosenblatt [1957]) is a type of linear classifier; a model which separates data-points by categorizing them a certain class based on their position in relation to a separation line.

A perceptron takes n number of inputs $\boldsymbol{x_1, ..., x_n}$, multiplies these inputs by corresponding weights $\boldsymbol{w_1, ..., w_n}$, sums these values, and passes the result through an activation function ϕ to produce the output y.

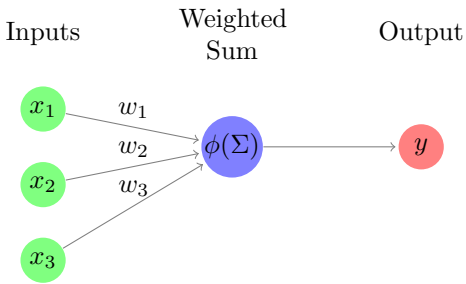


FIGURE 2.2 – Perceptron

While only capable of solving basic tasks on its own, the perceptron is a base unit in more advanced neural networks, which connects several of these units to create more complex behavior.

Loss functions

In neural networks, the loss function (also known as the cost function) is used to compare the output of a neural network to the desired output. The loss function aims to quantify how well a model produces the correct output, and is essential for training and evaluating a model. Output that closely resembles the correct output should be given a low loss value, and output that is far from the desired output should be given a high loss value.

The specific loss function used in a neural network depends on which task the network aims to solve. In regression tasks, the most common loss function is Mean Square Error (MSE), which takes the mean of the square distance from the output and target values. Such a function would be useful in regression tasks, where the goal is to approximate some numerical value.

For classification tasks, the most common loss function is cross-entropy. This function is used when the output is a probability of the input being a certain class, and functions by penalizing wrong probabilities. This function is also known as log loss, as the penalty is calculated by taking the negative log of the probability. So if the correct label is 1, and the output of the model is 0.3, the log loss would be $-log(0.3) = 0.523$. However, if the label is 1 and the prediction is 0.9, the loss would be $-log(0.9) = 0.0457$. If the label is 0, the loss is calculated as $-log(1 - p)$ instead. This way of calculating the loss heavily penalizes wildly incorrect predictions and mildly penalizes slightly incorrect predictions.

Training neural networks

As previously explained, neural networks function by multiplying the inputs with weights and passing it through an activation function. It is primarily these weights that require to be learned.

This can be accomplished by comparing how the output of the neural network differs when changing the weights a small amount ϵ, and observing how this affects the loss of the network. The effect of changing the weights on the loss of the network is known as the gradient of the loss function, and is an essential element in finding the optimal weights for the neural network.

By changing the weights in the negative direction of the loss gradient, the loss of that specific input example will decrease, which hopefully will improve the network for unseen examples as well. This leap in logic is called the inductive leap. By doing this process over and over again, the loss gradient will eventually converge to a minimum, which optimizes the performance of the neural network.

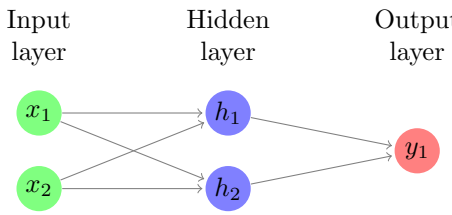


FIGURE 2.3 – Feed Forward Neural Network

A problem with a naive implementation of this algorithm is that a loss gradient can have several local minima, which while being a minima, is not the true global minima that minimizes the loss of the network. These can cause the algorithm to get "stuck", unable to reduce the loss of the neural network even if a better solution exists. This problem has been mitigated by the advent of advanced optimization algorithms such as Adam, RMSProp, and Adagrad, which employ techniques such as momentum, decay, and randomization of the gradient vector.

Dense Neural Network

By making the output of a perceptron the input of another perceptron, we can construct more complex models that can be used in more complicated tasks. An architecture where each perceptron in a layer is connected to every node in the next layer, we get what is known as a fully connected, or dense neural network. By combining this with non-linear activation functions, we are able to make predictions in a non-linear fashion, which greatly expands the number of use cases over the perceptron. For example, this makes these networks able to solve the XOR problem, which famously was seen as a big problem for neural networks, as shown by Minsky and Papert [1969].

Neural networks can be seen as an encapsulation of a function approximation, in which the neural network is a numerical method used to approximate the unknown function, this is useful in cases where the unknown function is too hard to construct manually, or the underlying function is unknown.

They function by connecting nodes (neurons) via weights, in which the input of the neurons and weights are multiplied together, summed, and passed through a non-linear activation function in combination with a bias input, to produce some result. By changing these weights, the network can be taught to produce the correct output given some input. The algorithm most commonly used to train a neural network is known as backpropagation.

Figure 2.3 shows a simple fully connected feed-forward neural network, in which two input variables are used to calculate a single output variable. The

two input variables are passed through the network, activating the hidden layer neurons, which in turn activate the output neuron to produce the result.

Neural networks work by taking several inputs $\boldsymbol{x_1, ..., x_n}$, multiplying them by trainable weights $\boldsymbol{w_1, ..., w_m}$, and passing the output of this computation through an activation function $\phi(a)$ where a is the result of computing $\boldsymbol{w^T x}$, which is the aforementioned multiplication of the inputs and the weights.

The output of a single neuron can thus be calculated in the following manner.

$$y_i = \phi(a) = \phi(\sum_{j=1}^{N} w_{ij} \cdot x_i) \tag{2.1}$$

In which the ith output y_i is defined as the activation of the neuron a_i passed through the activation function ϕ. N is the total number of weights to this neuron.

$$\begin{aligned} \alpha &:= w^T x \\ y &:= \phi(\alpha) \end{aligned} \tag{2.2}$$

Formula 2.2 is another way to represent equation 2.1, in which the individual calculations can be replaced with a matrix multiplication if the weight matrix and the input is represented as matrices.

In most modern neural networks, the RELU (REctified Linear Unit) activation function is used to introduce non-linearity to the neural network, allowing it to distinguish non-linear output spaces.

$$y(a) = max(a, 0) \tag{2.3}$$

The softmax function is also used, often in categorical use cases, as the output over all nodes in a softmax layer sums to one, making it suitable for layers where the desired output is a probability distribution.

$$softmax(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{2.4}$$

Bias

Every layer in a neural network employs a *bias* neuron, which is a special neuron that does not depend on the previous layer output. This node can be seen as the adjustment factor which adjusts the output space of the network, similar to how the b coefficient in linear equations of the form $y = ax + b$ is used to adjust the position of the linear line, the bias is used to adjust the position of the output space of the neural network layer.

When utilized, this changes the formula displayed in Equation 2.1 to account for the bias as such.

$$y_i = \phi(a_i - \beta_i) = \phi(\sum_{j=1}^{N} w_{ij} \cdot x_i - \beta_i) \tag{2.5}$$

$$\begin{aligned} \alpha &:= w^T x - \beta \\ y &:= \phi(\alpha) \end{aligned} \tag{2.6}$$

This also applies to equation 2.6, which simplifies the calculation.

Training

Neural networks are traditionally trained using the backpropagation algorithm, a deterministic method which iteratively improves the weights in the neural network to achieve the lowest possible loss.

Backpropagation (Rumelhart et al. [1988]) functions by calculating the gradient of the weights with respect to the loss function. This gradient is then utilized in every layer to adjust the weight in the negative direction of the gradient, which reduces the loss of the network. This method iteratively adjusts the weights after every input-output pair or after every epoch, as desired, to reduce the total loss of the network over time, until a point of convergence has been reached.

Overfitting

A common problem in machine learning is the act of overfitting (Hawkins [2004] for a more in-depth explanation), in which the model trains for too long, and thus begins to incorporate very specific features of the training set in order to improve performance. In essence, this concludes in a model that is too complex to be able to generalize. This leads to the model being unable to generalize to unseen examples, or its ability to do so is reduced.

Methods to relieve overfitting have been developed over the years, such as cross-validation, removing features, stopping training early, and ensembles.

2.2.2 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a variation of neural networks in which neurons in a layer connect to neurons in the next layer only in a local area. This means instead of every neuron in a layer connecting to every neuron in the next layer, only the neurons in a local area (known as the *kernel*) are connected to neurons in the next layer. This technique significantly reduces the number of parameters in the network, which makes it faster to train, and introduces

elements of translation invariance, because of the use of shared weights. This element of translation invariance is useful in cases where the position of the data of interest is irrelevant to the result. For example, this means a convolutional network should, in theory, be able to correctly classify an object regardless of its position in an image, while a fully connected neural network would not be able to do this out of the box.

CNNs are often used with images, videos, or other domains where the amount of input-features are very large, as to reduce the amount of computation needed. For example, an image of size 1000x1000 has 1 million input features; in a fully connected neural network the second layer (if it has equal dimensions) would therefore have 1 million weights for each input neuron to calculate, which equates to 1 billion weights ($1.000.000^2$). Even if the second layer only had a dimension of 10, the number of weights would still be 10.000.000.

Using a CNN with 64 3x3 kernels however, this equates to 576 (64·3·3) weights, which is considerably less computationally exhaustive, as well as providing the benefits of CNNs discussed prior.

Convolutional Neural Network (CNN)s were first utilized in combination with backpropagation in LeCun et al. [1989], this formed the fundamental algorithm which is still utilized in modern CNNs to this day. CNNs had been utilized previously in a similar domain in Denker et al. [1989], however, this paper used manually crafted filters for the CNN, which was much improved by the backpropagation usage in LeCun et al. [1989].

Convolutions

Convolutional neural networks are based on convolutions, which can be seen as a sliding-window style mathematical operation.

The convolution starts with the *kernel*, which is a small matrix of weights. This kernel is placed over the input matrix in a sliding manner, and the result of the element-wise multiplication and summation is placed in the output matrix.

$\mathbf{1_0}$	$\mathbf{2_1}$	2
$\mathbf{3_2}$	$\mathbf{1_3}$	3
1	2	3

$\rightarrow$

11	-
-	-

In this example, the kernel is placed in the top left of the input, being a 2x2 matrix with the values $\mathbf{[0, 1, 2, 3]}$ indicated as subscript numbers. The matrix is then multiplied element-wise with each value in the input matrix, and the result is summed, this produces the calculation $\mathbf{1 \cdot 0 + 2 \cdot 1 + 3 \cdot 2 + 1 \cdot 3}$, and the output 11.

1	**2_0**	**2_1**
3	**1_2**	**3_3**
1	2	3

$\rightarrow$

11	**13**
-	-

In this illustration, the kernel has been moved to the next position, and is now used to calculate the next output value, which in this case results in the number 13 being placed in the output matrix.

In this example, the output space has the same dimensions as the kernel, which results in a simple calculation with no need for padding in the input network. However, if the size of the output matrix is different from the size of the kernel, the input array can be padded with zeros to achieve different output dimensions.

If the input matrix is of size $n \times n$, multiplied by the filter matrix of size $f \times f$, the output matrix will be of size $n - f + 1 \times n - f + 1$.

By padding the input matrix to increase the dimension of the output matrix, a border around the input matrix of p size is added, so a p value of 1 would convert a $n \times n$ matrix to a matrix of size $n + 2 \times n + 2$, because the padding is added around the matrix, increasing the width and height by one on both sides.

Taking this into account when calculating the size of the output matrix, the calculation becomes $n + 2p - f + 1 \times n + 2p - f + 1$.

Usually, having the same dimensions in the output space as the input space is desired (known as "Same" padding), in which case the padding needed would be $p = \frac{f-1}{2}$, after solving the previous equation for p. This is the main reason the filter size is almost always an odd number, because this allows the input to be padded such that the output and input space have the same dimensions. It is also possible to pad to get the same output dimensions as the input dimension with even number filter, this requires padding to be on only one side of the input matrix, which can increase the complexity of the calculations.

In some scenarios, "Valid" padding might be desired, which means no padding at all. However, this strategy might lead to some columns of the input matrix not being used for the calculation to achieve the desired output dimension, which might not be desirable.

Pooling

Pooling is a commonly used technique in convolutional neural networks mainly used to downsample the number of inputs for the next layer. Pooling works by reducing the dimensions of the input by using some method to combine several input nodes to one. The most commonly used techniques for pooling is average pooling and max pooling.

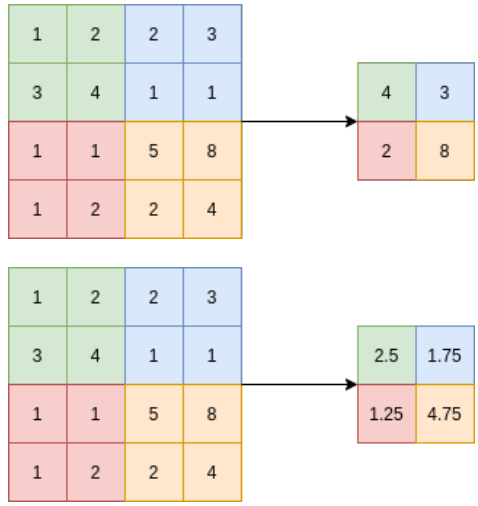


FIGURE 2.4 – Max and Average Pooling

The top matrix in Figure 2.4 shows how max pooling reduces the dimensions by a factor of two. In this example, max pooling is applied with a size of 2x2, and a stride of 2x2, however other sizes are also possible. Max pooling works by taking the highest value from the input as the output. The second matrix shows average pooling, which takes the average of every value in the input range and uses that as the output.

There are merits to each type of pooling; one can argue that max pooling extracts the most important data from each block, and that average pooling preserves the most detail, however, the method of choice depends on the application at hand.

2.2.3 Autoencoder Architecture

Autoencoders are a type of neural network architecture that learn a compressed representation of the inputs, so they can reconstruct it. This is useful for detecting outliers or anomalous readings in the data, as the autoencoder cannot reconstruct these, which means a subtraction between the input and output data will reveal the anomalous data. Autoencoders are generally seen as an unsupervised learning method, though during training the model does have target values to compare the input to, however, these values are the same values as the input, which means no additional information other than the input values are required to learn such a model. In this sense, autoencoders can be seen as an unsupervised learning method which utilizes aspects from supervised learning to accomplish its results.

Autoencoders are composed of the encoder and the decoder; the encoder takes the input data and encodes it to a lower dimensionality output, and the decoder takes this output as input and attempts to decode it into the original data.

An autoencoder is a special architecture of neural networks, that aims to replicate a generalized version of the identity function, which is the function in which $f(x) = x$. However, an autoencoder does this using a bottleneck layer, which is a layer within the neural network which is not able to store the entirety of the information contained within the input data. By training this network, this bottleneck layer can be trained to learn a compressed version of the input. This is useful for feature extraction, the process of extracting features from a rich dataset. This is possible because if the autoencoder can reconstruct the initial output with any degree of accuracy, we know that some subset of features in the dataset represent the data in a more compact manner.

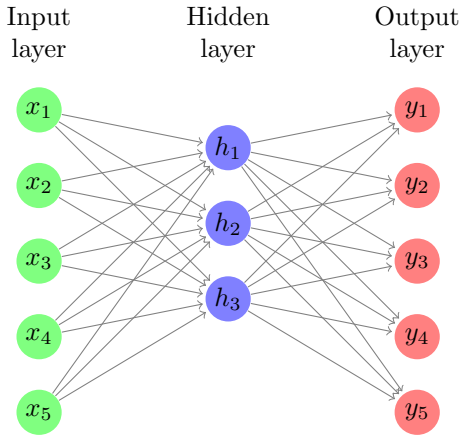


FIGURE 2.5 – Autoencoder

Figure 2.5 shows a simple autoencoder architecture, which attempts to reconstruct data with 5 input features through a bottleneck layer with 3 nodes.

Use cases

The most common use of autoencoders is in anomaly detection, where the autoencoder learns the representation of the item of interest and attempts to reconstruct unseen data. The idea is that data that is able to be reproduced well is similar enough to normal data as not to be considered anomalous, while data which is unable to be reproduced well is significantly different from normal data and is therefore an outlier or anomaly.

Another use of autoencoders is denoising, where an autoencoder is able to remove noise or random data from the input, producing the input without noise as its output. This can be applied in a wide array of applications, such as removing noise from images, or other fields where noisy data is a problem. Cho [2013] demonstrated the use of autoencoders for denoising in his paper.

Denoising autoencoders work on the principle that the model will not learn the noise in the data, as this noise provides no information about the final output, and can thereby be ignored by the model to preserve more information about the parts of the input which affects the output.

Chapter 3

State Of The Art

This chapter will detail related work and papers in relevant domains regarding the problems presented in this thesis.

3.1 Object Detection

Object detection of some kind is required to extract usable data from a dataset consisting of video files. Ideally, the extracted data would be perfectly framed individual signs; however, extracting such data is no simple task.

Object detection would also be integral in a completed system that might use techniques discussed in this thesis, such as a system created by SVV to maintain road infrastructure in Norway.

3.1.1 YOLO

Redmon et al. [2015] presented You Only Look Once (YOLO), a single unified neural network that takes a resized image and outputs bounding boxes with associated class probabilities.

YOLO broke with conventional object detection methods that relied on single object detector networks combined with a sliding window to check every piece of an image for each object. In contrast, YOLO uses a single model which uses the whole image as an input. This method requires significantly less computation and provides best in class results.
Limitations of this solution are its vulnerability to small objects, and poor performance on generalizing detection of objects in different configurations or aspect ratios.

These factors might affect the ability of YOLO to be utilized in the sign domain, as the images are often small and at an angle.

3.1.2 Traffic Sign Detection

Lee and Kim [2018] tackles the problem of sign recognition using a single CNN and overlaying sign templates. The CNN is used to extract a rough pose of the sign position from the original image, and sign overlays are used to determine the exact type and shape of a sign.
The paper describes how previously, handcrafted features were used to create such models. However, their model requires no such handcrafted features, and still achieves high scores for detection.

The main contribution of this method is its ability to extract the exact pose of the traffic sign, which includes the specific geometry of the sign in question. This is done by keeping a "database" of sign shapes, which includes round, square, octagon, and other shapes, fitted to the detected signs to extract the exact position and angle of the sign.
However, this method requires serious computation and results in low Frames Per Second (FPS) which might not be desirable for real-time use, and might also not be desirable for post-collection analysis, as the time to extract the image poses might simply be too large. For example, a 30 FPS 5 minute video would take 30 minutes to analyze with this method.
While the method used in said paper might not be suitable for this thesis, the techniques and results gathered show great potential, especially the ability to extract sign poses with very high accuracy. This could be used to detect signs that are not mounted correctly with regards to their angle.

3.1.3 Faster R-CNN

Ren et al. [2015] augment the existing Fast R-CNN object detection network by implementing a new type of region proposal for the network. Region proposals are a strategy used within object detection to identify potential areas where objects are likely to be present. This paper details the new region proposal method which improves upon existing solutions by only requiring a single CNN to do the proposals, instead of having to use several networks for different scales and transformations. They integrate their region proposal network and object detection network into a single network, which results in a significant improvement in speed over existing networks.

The model achieved best in class results on popular datasets, combined with the fact that training and utilizing this model is relatively simple, makes it an ideal choice for data extraction needs.

3.1.4 Object detection in videos

While object detection in images is a mature topic, utilizing multiple frames from videos to improve upon object detection is a recent endeavor. Indeed, solutions that simply treat videos as standalone images exist, and are trivial to implement from a single image object detector, solutions that take advantage of multiple frames while retaining continuity are rarer to find.

Since videos in this domain often have multiple consecutive images of the same object to be detection in a row, this could be taken advantage of to gain more data about the object do detect, possibly from multiple angles or with different parts of the object visible.

Timofte et al. [2014] proposes a solution which similarly tackles this problem. However their goals are achieved by mapping multiple 2D detections to a 3D hypothesis which can be utilized to calculate the best detections for the 2D detections.

While this method achieves great results, it requires the use of specialized camera hardware, which is infeasible for large scale deployment without a large upfront investment. The FPS measure of the model is also quite low, which might limit usage in big data environments.

The method developed in said paper would be ideal for use in a potential complete sign detection system, however the use of specialized camera hardware precludes its usage in this thesis, however the method is one that should be noted for future development.

Downs [2017] is another such system, which compares the performance of a single-frame model and a model which is given multiple frames. They found that the network which was given multiple frames was able to more accurately detect the objects of interest in the frames compared to the single-frame model.

To achieve this, they created a convolutional neural network that used a $300 \times 300 \times f$ input layer, f being the number of frames to analyze at a time. This architecture acted as both the single-frame and multi-frame version of the model, with the single-frame model having an f value of 1, and the multi-frame version having an f value of 10.

This paper shows excellent results, and might be able to be adapted to a future version of the architecture used in this thesis. The static number of frames used in the model might not be suited for this domain. However, this requires further research.

3.2 Anomaly Detection

Anomaly detection is the main focus of this thesis, and is a branch of artificial intelligence that has seen a wide array of methods and approaches utilized in different domains. Choosing an appropriate method is paramount to developing an adaptable solution which could be utilized by SVV and other actors in the future to automate and improve current infrastructure inspection procedures.

Chalapathy and Chawla [2019] describes a fitting taxonomy on types of anomaly detection models, semi-supervised, unsupervised, hybrid, and one-class. The choice of model depends on the input data available, sequential or non-sequential, low- or high-dimensionality. The data utilized in this thesis would be considered non-sequential, high dimensionality; seeing as it consists of separate images of traffic signs, with a large dimensionality because of the raw number of pixels present being very large, usually 200-500px x 200-500 px, totaling between 40.000 and 250.000 input features.
The problem would also be classed as unsupervised, because of the lack of labels present in the dataset.

The paper also describes a taxonomy of anomaly types, point, contextual, and collective anomalies, respectively. Point anomalies are anomalies that separate themselves from existing data. Contextual anomalies are anomalies that might seem normal on their own, but seen in a larger context would be classified as an anomaly. Finally, collective anomalies are groups of data that while seeming normal as point-data, observed as a group presents as anomalies.

The anomalies explored in this thesis would be classified as point anomalies, as they are not seen in the context of larger groupings of traffic signs, and are not viewed contextually, as sequential traffic signs would have no impact on the individual traffic signs' status as anomalous. If the data set consisted of multiple grouped images of the same sign, the problem could be construed as a collective-anomaly problem, which might help eliminate the problem of angles or obstructions increasing the anomaly score of certain traffic signs.

3.2.1 Deep One-Class Classification

In Chalapathy et al. [2018] the researchers augment similar deep methods with the one-class output component, which they argue allows the model to be designed for anomaly detection from the ground up, which aids data representation in the hidden layers to accomplish anomaly detection more efficiently.

The method does not show improved results compared to other state-of-the-art methods, however, it does out-compete more conventional methods.

Ruff et al. [2018] utilizes a form of one-class classification to accomplish

anomaly detection. The difference here is that they map the output from the input space to a hypersphere output space, the objective of this network is to map the most normal examples from the data set to the middle of the sphere, and thereby to be able to classify anomalous examples by looking at the examples which are classified as being outside the sphere.
They argue that common machine learning anomaly detection models are not designed with anomaly detection in mind, but are rather primarily generative models with anomaly detection attached. Their model is designed with anomaly detection as a primary objective.
This is achieved as mentioned by leaning a sphere representation enveloping the output space by outputting center c and radius r, and categorizing results inside the sphere as normal, and results outside the sphere as anomalous.
The main problem with the method seem to be avoiding the model simply learning a solution with a large radius that encompasses too large of a volume to be statistically relevant for anomaly detection.
Results show that on the MNIST dataset, their model outperforms competing models on most digits. On the CIFAR-10 dataset it shows overall good performance but does not out-compete existing models.

3.2.2 Autoencoder anomaly detection

In Cozzolino and Verdoliva [2016], the researchers attempt to detect forged images by utilizing an autoencoder to reconstruct the noise residual in the images. Their model outperforms previous state-of-the-art methods, which generally relied on information such as JPEG artifacts, and camera fingerprints. This model does not make any such assumptions. Instead, the method relies on the reconstruction error in the noise space to detect areas that seem anomalous compared to the rest of the image. A subtraction between the original noise and the reconstructed noise will then reveal the edited parts of the image.

Interestingly, the researchers found greater success utilizing overcomplete autoencoders rather than traditional bottleneck autoencoders. The researchers postulate the reason for this being that redundancy in the hidden layers aid the model in recognizing the correct features to utilize for its representation, however they concede that more research is needed on this topic to fully explain this phenomenon.

While this paper does not reconstruct images directly, and instead aims to reconstruct the noise residual in images, their approach is similar enough that it could conceivably be utilized in the sign domain as well.

3.2.3 VAE

Baur et al. [2019] describes the use of variational autoencoders for use in brain MR imaging. This technique uses unsupervised learning to reconstruct images using an autoencoder and comparing the original and reconstructed image, this reveals details in the original image that the neural network was not able to reconstruct, which indicates abnormal structures in the original image, this is a form of unsupervised anomaly detection, as it does not utilize labels.

The paper describes how the manual identification of anomalies in brain MR imaging is a tedious and labor intensive task often performed by highly skilled professionals, which is a similar situation to this domain. While their domain is entirely different from this thesis, the themes and data is almost entirely transferable.

The paper initially postulates that an autoencoder should provide the ability to reveal post-processed residual in the output differential which when of sufficient size, would reveal the location of anomalous growths in the brain. The paper showcases related work in similar domains which mainly consists of traditional data analysis such as clustering, other related papers use unsupervised learning to some extent, but mainly as an aid to further supervised learning as the main detection mechanism.

This paper lays out the necessity of high quality sharp reconstructed images for their domain, and concludes that typical generative models are not able to produce reconstructions of sufficient sharpness to allow the results to be utilized for brain MR imaging in an effective way. However, they display that a VAE is able to produce the sharpness required.

The paper also compares its VAE with other autoencoder methods, which shows this model performing better in this domain. Other methods mentioned such as dense autoencoders and spatial autoencoders also seem to produce workable results. It is clear from this paper that in the case where reconstruction sharpness is a sizeable challenge, VAEs outperform competing methods.

Chapter 4

Architecture/Model

This chapter will present the architecture designed to provide a basic solution to the problems presented in this thesis. The architecture should be able to detect anomalies in signs using the same data foundation which will be found in a potential real-world system that might utilize similar mechanisms.

4.1 Models

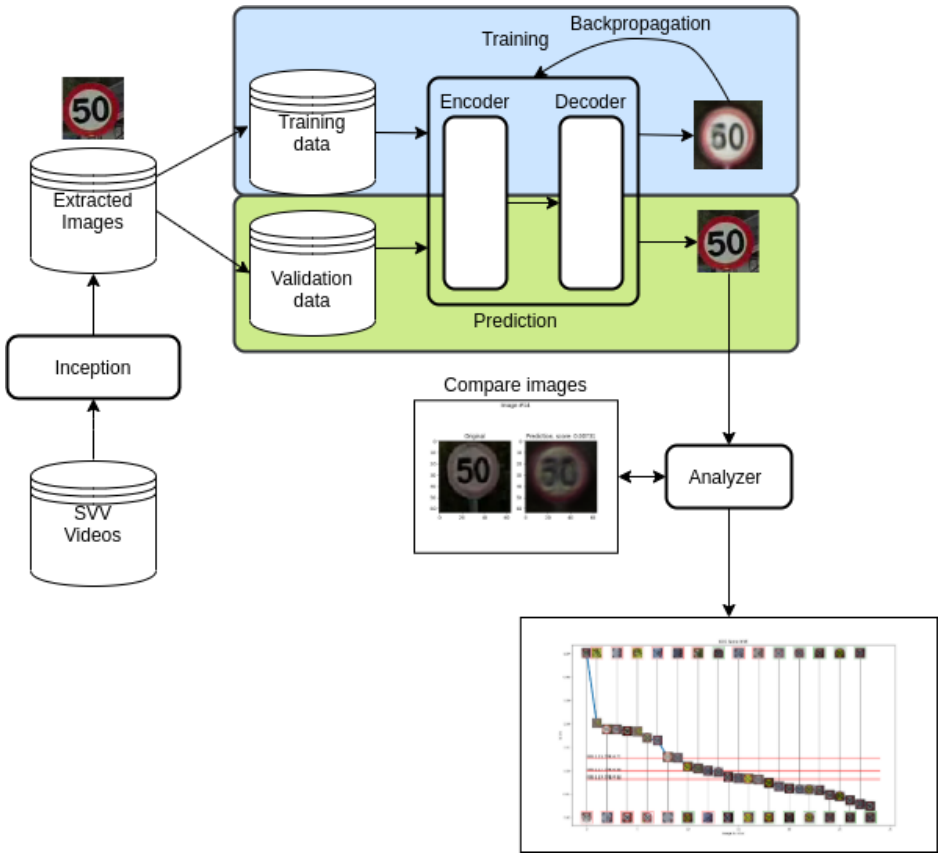


FIGURE 4.1 – Overall System Pipeline

The overall pipeline of the system is shown in Figure 4.1. This figure shows the pipeline of events, which starts with the database of videos and stops after analyzing the anomaly detection results. The overall goal of this architecture is to take videos as input, and produce predictions about which signs in these videos might be considered anomalies.
The pipeline begins at the database of SVV videos; this database is passed through the object detection model to extract images to be used in the anomaly detection model.
The training images are used to train the anomaly detection model on a relatively

clean dataset.
After the model is done training, it is utilized to reconstruct all the images in a separate dataset, which produces pairs of images, the original and the reconstructed version, this also stores the MSE loss of the reconstruction, which will be useful for calculating the ROC score.
These pairs of images are presented to the analyzer, which simply arranges this data to form the ROC metrics and plots in this thesis.

Practically, this pipeline was split into several smaller tasks which could be executed independently. This means the data extraction part was only conducted a few times, as this was the most time-consuming part of the pipeline. The training section was also quite time-consuming. However, the analysis part took only a few minutes.

4.1.1 Object Detection

Object detection is done using the Faster R-CNN Inception v2 implementation found in the Tensorflow Object Detection API (Huang et al. [2016]). This model was chosen because it exhibits good performance with high FPS, as well as ease of use as it is readily available in Tensorflow. Though there exists more advanced methods, these are not as easily available, and since object detection is not the primary focus of this thesis, this model was chosen.

4.1.2 Anomaly Detection

The anomaly detection model is based around a fully-connected autoencoder, which will use the extracted signs as input. The reconstruction error of the images is then used as the basis for detecting anomalies. This method was selected from a pool of potential methods which was created through similar solutions found in related papers, and the fully connected autoencoder proved the most effective out of these methods.

4.1.3 Analysis

Analysis on model performance is based on the ROC and ROC-AUC metrics. These were chosen for their ability to distinguish performance with a highly unbalanced dataset, such as the one used in this thesis, where the validation set contains only a handful of anomalous images, and a large number of normal images.

The standard accuracy metric which is common in machine learning systems is not useful in this case, as the large number of normal instances means the model could simply declare every image normal and achieve a 95% or higher accuracy.

4.2 Implementation

This section details the implementation of the anomaly detection pipeline using the data provided by Statens Vegvesen (The Norwegian Public Roads Administration) (SVV). The programming was done in Python 3 utilizing Keras. The source code can be found at `https://github.com/OlavH96/Master`, and as an addendum to this thesis.

4.2.1 Data Extraction

To be able to train a model to recognize anomalies in traffic signs, a dataset of traffic signs is required. However such a dataset for Norwegian traffic signs is virtually non-existent. Therefore the aforementioned Ren et al. [2015] model implemented in Tensorflow was trained by generating 20.000 images with a number of typical background images and standard images of the relevant Norwegian traffic signs; the signs were rotated, scaled, blurred, stretched, and otherwise manipulated in manners which might be expected to occur in normal road conditions. Some parts of existing code from Kantega was utilized to perform this generation, in accordance with our agreement with SVV.

This model for object detection was chosen because it shows good performance in theory, and it is also easily accessible since it is already implemented in Tensorflow, which makes it ideal for this Python based project. Object detection is also not the main focus of this thesis, so some errors in the data is allowable, however a complete system would need to have solid object detection to be functional.

Training the sign detection model with these images resulted in acceptable performance when tested on the videos provided by SVV. Specifically 84% of the extracted images were deemed usable for training the anomaly model, based on a sample of 5175 signs which were manually reviewed in which 4351 did not possess any qualities which made it obviously unusable for further use, such as not being images of signs at all, being too blurry, or too bright.

Utilizing this model, 100.000 images were initially extracted, however as it was later decided to only utilize the 50-signs, this resulted in this number being trimmed to 4351, which had been manually combed through to remove any obviously unusable signs, as explained in the last paragraph. This dataset was to be the training data for the anomaly detection model. A separate dataset was created by extracting signs from an additional 100 videos, this produced 778 images of fifty signs, which was combined with the 14 real anomaly images provided by SVV to create the anomaly detection validation data. The reason for extracting new images to create this dataset was to make sure the images were entirely different from the training data, which might contain some duplicates because of

the object detection model sometimes extracting the same sign multiple times; this strategy nullified this problem.

4.2.2 Data Preparation

Having trained the extraction model and extracted 100.000 signs to be used as training and validation data; initial testing revealed the need to prepare the data in order to improve results.

The first and simplest method to improve the data was to remove images with very low average brightness. A threshold of 0.1 mean value was chosen, taking the average of all pixels in the image proved to produce the best results, discarding the most erroneous images without discarding too many actual images. This proved necessary because the provided videos are censored, with cars, people, and other personal information being blacked out. The extraction model detected censored images as signs, even though they were almost completely black. This technique also discarded some other detections which were too dark to utilize.

The second method was to replace the image format used for the anomaly detection model from RGB (Red Green Blue) to HSV (Hue Saturation Value) color space. This change hopes to remove a bias observed in testing, in which darker signs were more easily reconstructed and given a lower error score.

Testing revealed that this technique provided no tangible improvement in model performance, and in some cases decreased the performance of the model. Therefore, using the standard RGB encoding was chosen.

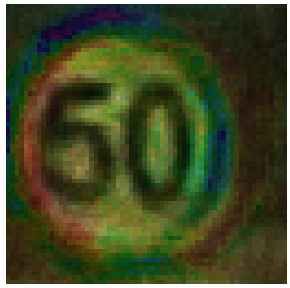


FIGURE 4.2 – Prediction using HSV model

As figure 4.2 shows, in some cases the HSV model produces discolored predictions, which makes it unable to determine whether or not the image is anomalous. While it is possible that using this scheme might improve the ability of the model to distinguish between images of high and low brightness, it adds additional difficulty while training the model. Therefore the hypothesis that using

HSV improves performance in low light conditions has not been further studied in this thesis, however, it could be a good point for further research.

4.2.3 Anomaly Detection

Fully Connected Autoencoder

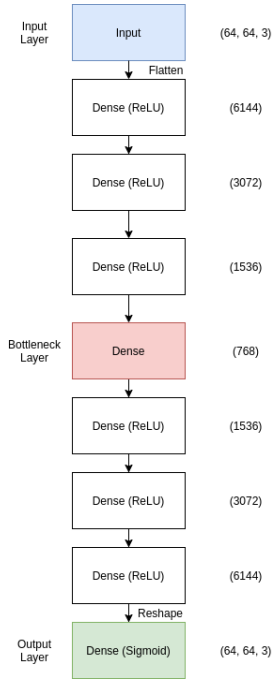


FIGURE 4.3 – Fully Connected Autoencoder

Through some experimentation, the parameters highlighted in Figure 4.3 were chosen as the best suited for this task. Three hidden layers before and after the middle layer produced the optimal results.

For the activation function ReLU was primarily utilized. Some experimentation with the Sigmoid activation function was also performed, however, this proved to make little difference in the overall performance of the network.

For the loss function, MSE was utilized, which is the industry standard for such neural networks.

The network was trained for 50 epochs with 8000 steps per epoch, using the Adam optimizer with a learning rate of 0.01.

Chapter 5

Experiments and Results

5.1 Experimental Plan

The initial plan was to apply an existing Kantega sign detection network to gather images of signs from the dataset provided by SVV, this data would then be utilized to train an anomaly detection network to detect anomalous signs. This network would then be integrated into the pipeline to run in real time. The anomaly network and detection network could alternatively be conjoined into a single network.

The initial plan proved challenging to implement for several reasons; there was no way to utilize the existing sign detection model directly because of technical limitations, so to achieve the same goal a new sign detection model had to be trained. The plan to utilize this model in real-time also was impossible as no real-time data was accessible, therefore using the model on older footage seemed the best approach, however, it could be feasible to use this model in real time as well, but this would require substantial hardware.

The plan also initially included plans to utilize Nasjonal vegdatabank (National Road Database) (NVDB) to aid both the object detection and the anomaly detection section of this thesis. However, this proved too labor intensive, and is left as future work.

Initially, it was planned for this anomaly detection system to encompass every type of traffic sign detected by the object detection model, but it was quickly discovered that such a system would require a substantial amount of fine-tuning and training which was not possible to achieve given the time constraints. Therefore it was selected to focus the anomaly detection on one type of traffic sign.

The revised plan consists of using an object detection model to extract usable

images of signs from the provided videos from SVV. This data will be used to train a model to detect anomalies in unseen images of signs. This can then be applied to a validation set to measure the effectiveness of the model in discovering the sparse anomalies.

The experiments performed in this thesis aims to uncover whether or not it is viable to utilize unlabeled data to create a sufficiently effective anomaly detection system without the need for human interaction.

The ideal model for this domain would be eager in its detection, opting instead to detect more false positives than false negatives. If the model can scale the amount of data for human supervision down to a manageable amount without discarding a disproportionate amount of anomalous data, the model would be a success.

5.1.1 Challenges

Extraction

The first challenge is extracting usable imagery from the videos provided. These videos are dashcam footage from SVV associated vehicles, recorded in 720p, and includes some relevant metadata such as coordinates, however this metadata was not utilized in this thesis.

To solve this challenge, the existing Tensorflow Object Detection API (Huang et al. [2016]) was utilized. This system makes it easy to train a model capable of detecting objects of choice by feeding images with labels of the object and corresponding bounding boxes. This proved to be quite a simple solution, and produced decent results when applied to the real data, allowing the extraction of images for further use.

Some images extracted were not of sufficient quality, so some manual removal was necessary. Some of the signs also had the wrong labels, however this did not really matter for this thesis. These problems could be alleviated by training the object detection model for longer and with more training data.

Labels

The data is unlabeled, as it simply is automatically extracted images of signs. There is no feasible way to label such a dataset in a short amount of time. The object detection model predicts what type of sign it believes the image is, however, this is not useful for anomaly detection purposes. The initial idea was to utilize the object detection loss score to detect anomalies; however, there proved to be not enough correlation in the score to create an effective anomaly detection

system based on this data.

Since the vast quantity of signs are not anomalous, it can be presumed that the model will not learn anomalous behavior. By looking at the data manually, it is estimated that less than 1% of the extracted images contain anomalies.

For the aforementioned reasons, it was selected to label every extracted image as "normal" for the purpose of anomaly testing, and the images of real anomalies provided by SVV will be used as anomalous images. These labels do not aid in the training of the anomaly detection model, however it might be possible to use such labeling strategies to create a one-class anomaly detection model.

Anomalies

To test the proficiency of the model, a dataset containing both normal and anomalous data is required. Initially, the plan was to generate such images by using templates of anomalies, such as snow cover, cracks, and scratches. However, SVV provided a dataset, which made it possible to create a dataset which contained real anomalies. These anomalies were similar to what was expected, the primary anomaly being snow cover, with the second most common anomaly being scratches or other deformations on the signs.

Some other anomalies present were warped signs and signs which have been moved or rotated accidentally in some way, however detecting such anomalies using a single image proved difficult, and it is the opinion of the author that such a system would be required to use sequential images of the same sign from multiple angles to correctly classify such anomalies.

Evaluation

Another challenge is evaluating the performance of a model. For this purpose the ROC-AUC approach was chosen, utilizing this metric to judge the performance of trained models on a dataset containing around 500 normal signs and 14 real anomaly images. This evaluation method is well suited when faced with sparse anomalies, as it measures the FPR and TPR in relation to each other. If the dataset was more evenly distributed, methods such as accuracy could be used instead, but this is not the case here.

Using this method a score of 0.5 would equate to random guessing, and any higher score would indicate the model was able to learn some underlying structure of the signs. Higher score indicates a better model, with a score of for example 0.95 indicating the model is able to score a random anomaly higher than a random non-anomaly 95% of the time. It is worth noting that this method is only used to evaluate models as a whole, and the output of a model is not given in this format, rather given in the loss function of the model at hand, usually Mean Square Error.

5.2 Experimental Setup

5.2.1 Dataset

This thesis utilizes a dataset provided by SVV, which consists of 5 minute mp4 videos in 720p available through an API. At the time of writing, 12000 such videos are available, which amounts to around 10000 hours of video footage, or 20TB worth of data. Utilizing all this data would require more computing power than available, and would probably be overkill. Therefore, a smaller randomly selected number of videos was used in this thesis to train the model and conduct the anomaly detection testing.

In this thesis, the mp4 videos are treated simply as a sequence of images, in which each image is passed through the object detection model to search for potential traffic signs. Other methods of treating the data were considered, but ultimately discarded for being too labor intensive for this thesis, and are further discussed in Section 6.2.4.

Real anomalies

In addition to the videos which are used as the basis for the anomaly detection model, SVV provided some additional images of real anomalies that will be utilized for testing the performance of the model.

These are images from the internal system used by SVV, which is used by SVV employees to report anomalies found in the field due to contractors not upholding their part of the contract to maintain their assigned road infrastructure.

FIGURE 5.1 – Real anomaly images

In contrast to the pictures extracted by the object detection model, these images are higher resolution and usually have a perfect angle on the sign plate. Signs detected by the object detection model might get more of a side view, however, this should make little difference. These images also do not exhibit any motion blur, as the images were taken from a stationary position, which is not the case for the images extracted from the videos.

As one might observe, the first image in the series is partially melted on the left side, which might become too small to notice when the image is resized, potentially uncovering the need for a model with a larger input dimension. The other images are covered by snow or blocked by bushes. The second to last sign is tilted to the right, which the model might struggle to recognise.

Looking at the data, it was initially expected that the two tree-covered images would be particularly easy to detect because the color is so radically different from normal traffic signs. It was also expected that the snow-covered signs would be reasonably easy to detect, and the first and second to last sign would be the hardest to detect.

Object Detection

The dataset for creating the object detection model was generated using a sample image of each sign type to detect, which was then placed on a background and randomly adjusted with different angles, noise, rotation, scale, stretch, and fade. 20 000 such images were generated, and used to train the object detection model. Which was then utilized to extract the signs from the real data.

This model was initially trained to detect 32 different types of Norwegian traffic signs. However, later it was decided to focus the anomaly detection portion of this thesis on a single type of traffic sign, so much of the data is unused.

5.3 Experimental Results

The loss presented in certain images below refers to the Mean Square Error (MSE), which is the calculated mean square error between all pixels in the original and the reconstructed image.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2 \tag{5.1}$$

Where y_i is a pixel from the reconstructed image and x_i is the corresponding pixel from the original image. n is the total number of pixels in the image.

(A) Original (B) Reconstructed

FIGURE 5.2 – Reconstruction Comparison 1

Figure 5.2 shows a reconstruction of an image where the color has faded, which would be classified as an anomaly. This image was found in the dataset of extracted images from the videos provided by SVV. The reconstructed version has added back the color, displaying the models' ability to recognize that normal signs of this type are supposed to have a red border.

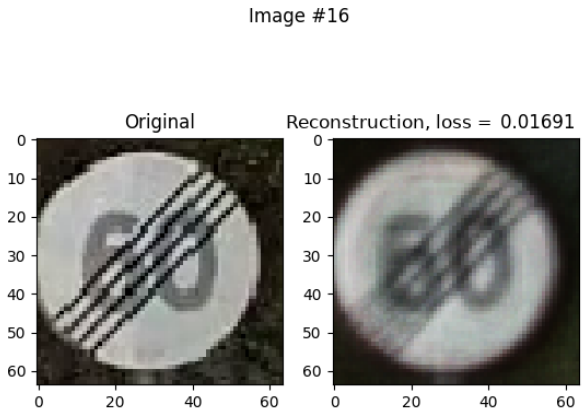


FIGURE 5.3 – Reconstruction Comparison 2

Figure 5.3 shows another example where another type of sign has been reconstructed well. The model has recognized that this particular type of sign is not supposed to have a red border; and has reconstructed it correctly. This type of sign was not included in the final model and associated results; however, this image was included to display the ability of the model to reconstruct these signs as well.

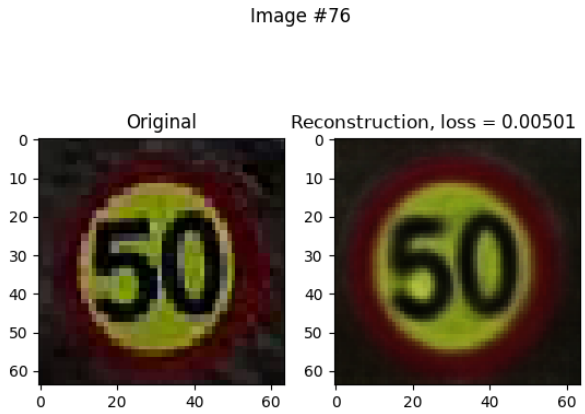


FIGURE 5.4 – Reconstruction Comparison 3

Figure 5.4 shows that the model can distinguish and reconstruct the version of signs with a yellow background. The yellow background usually indicates a temporary sign, but these signs are quite common in the dataset.

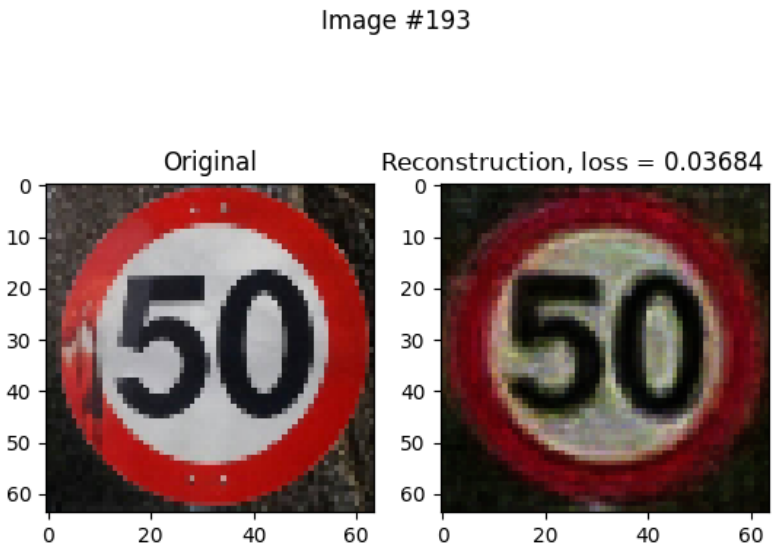


FIGURE 5.5 – Reconstruction Comparison Real Anomaly

Figure 5.5 highlights the result of the aforementioned real anomaly from Section 5.2.1, where the left side of the sign is partially melted. This sign was expected to score poorly using this model. This image was the 15th highest scored image, which could be considered a good result for such a hard to distinguish anomalous image in such a large dataset.

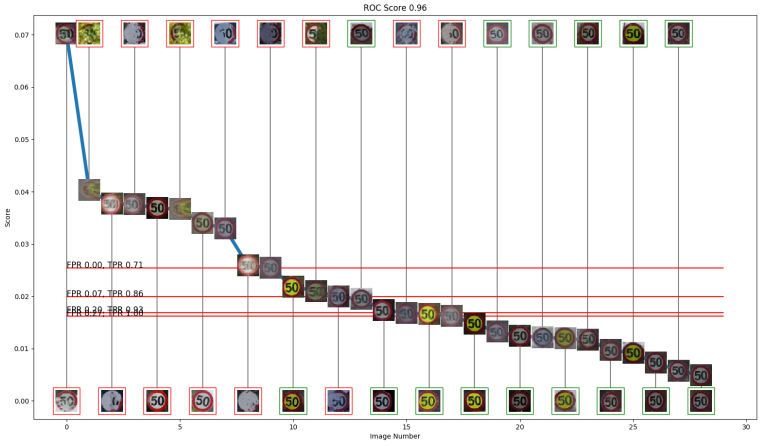


FIGURE 5.6 – Score plot for real anomalies 1

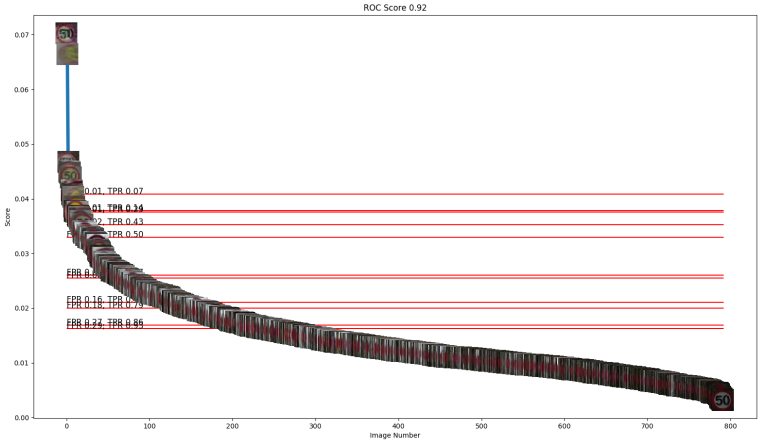


FIGURE 5.7 – Score plot for real anomalies 2

Figure 5.6 and Figure 5.7 shows the reconstructions of images plotted against their MSE scores on a dataset consisting of 778 images from a dataset separated

from the training data, and 14 anomaly images provided by SVV. The model achieved a ROC-AUC score of 0.92.

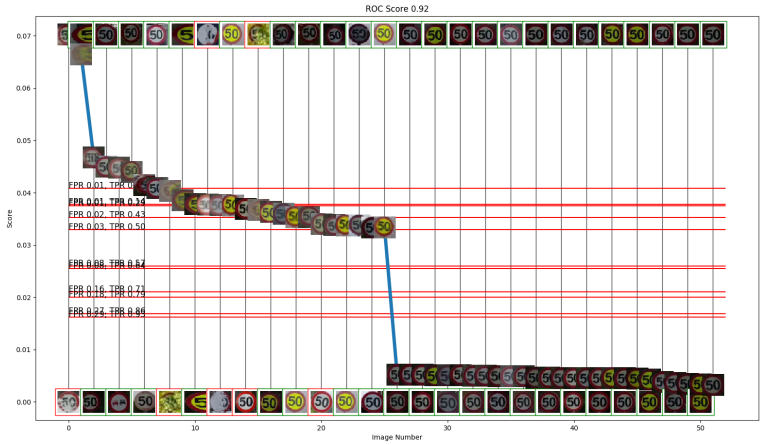


FIGURE 5.8 – Plot for real anomalies

Figure 5.8 shows the same model and dataset as Figure 5.7. This graph shows only the top 25 and bottom 25 scored images. The red outline on the top and bottom images indicates anomalous signs, while the green outline indicates normal signs.

One might observe that one of the highest rated images is a faded 50-sign, which would be classified as an anomaly. This sign is displayed as normal as it was part of the extracted signs dataset, however in reality it would be a good anomaly candidate.

One can also observe a sign which is not a 50-sign on the bottom 3rd from the left, which got a high error because the model was trained to reconstruct 50-signs and not this type of sign. This image was accidentally left in the test data instead of being removed.

The red horizontal lines in these graphs indicates the cutoff points for this model which can be used to predict anomalies with their respective FPR and TPR, choosing the lowest line for example, would result in the highest number of true positives, but would also result in the highest number of false positives.

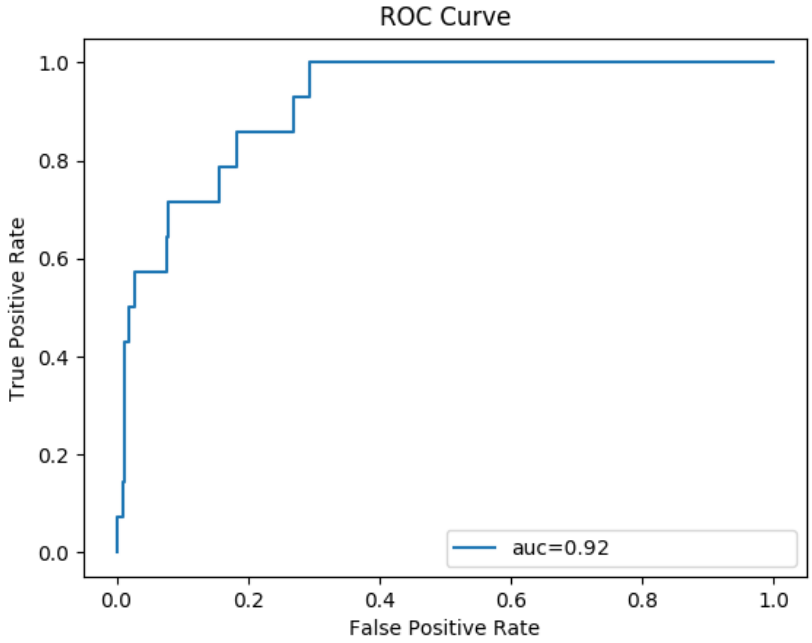


FIGURE 5.9 – ROC Curve

Figure 5.9 shows the ROC curve for the model, displaying the model's ability to achieve high TPR, while receiving low FPR. This model has learned some underlying structure to the data which allows it to separate the examples with a high degree of confidence. Depending on the threshold chosen the model can separate 60% of the anomalous data, while misclassifying 5% of the non-anomalous data.

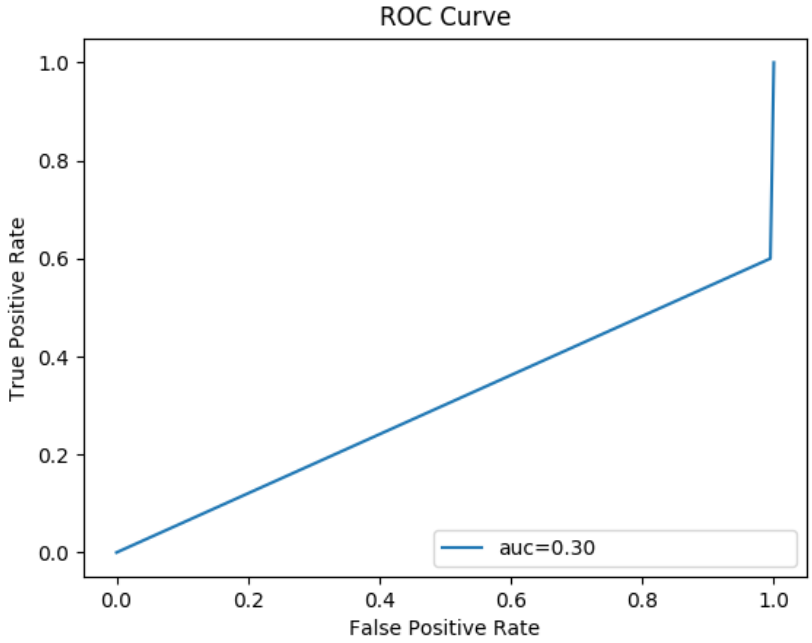


FIGURE 5.10 – ROC Curve for KNN model

Figure 5.10 shows the ROC curve for a KNN model applied to the same dataset as a comparison. This model was created as a baseline to compare other models against. The model achieves a high accuracy of 97%. However this is deceiving as the dataset contains mainly non-anomalous data. As for the FPR and TPR, the model displays no discernible ability to separate the anomalous and non anomalous data.

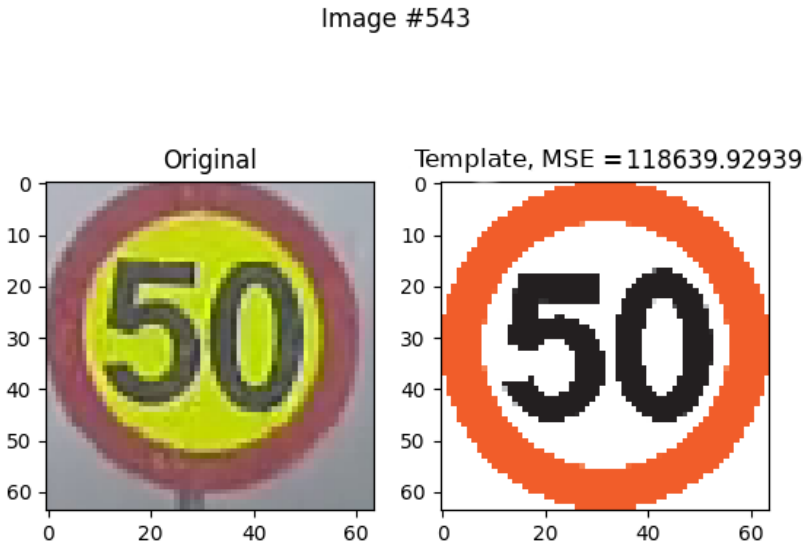


FIGURE 5.11 – Reconstruction Comparison with template

As another baseline a model which simply takes the MSE between the original image and a template image was constructed. This model is obviously completely unable to take into account that these signs may also have a yellow background, which is something that the autoencoder model is able to do. This model would also require each original image to be perfectly framed the same way as the template image to achieve the best results, and would also require the signs plate to be orthogonal to the camera, which is often not the case with real-world data. This resulted in a model with very poor ability to distinguish the normal signs and the anomalous signs.

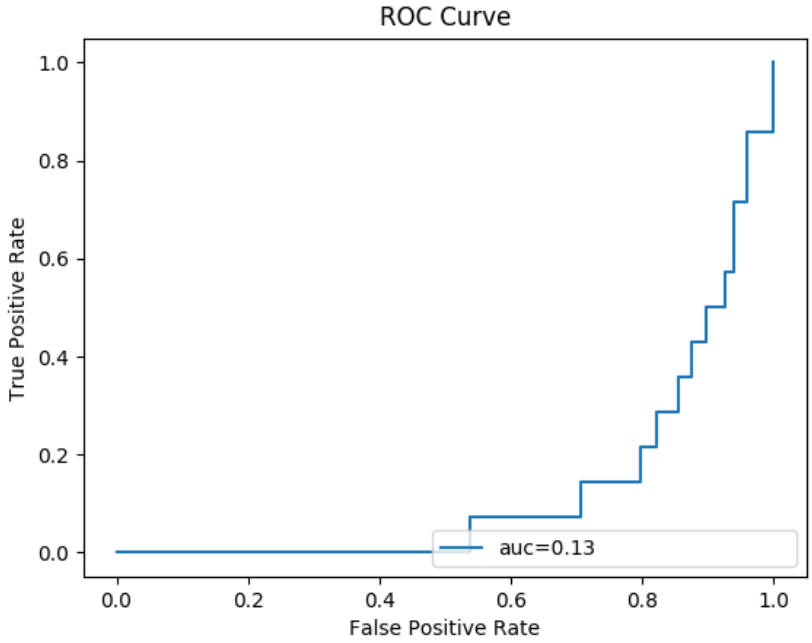


FIGURE 5.12 – ROC Curve for template model

To add to the previous figure, Figure 5.12 shows the ROC curve of this model, which displays worse than random ability to distinguish the signs.

Chapter 6

Evaluation and Conclusion

This chapter will evaluate the results presented in Chapter 5. It will also conclude the thesis and discuss the impact of the result and its potential weaknesses and further possibilities for development.

6.1 Evaluation

While the number of true anomalies utilized in this thesis to evaluate the model is low, the results gathered from these experiments show that it is possible to use such an architecture to accurately distinguish normal and anomalous traffic signs. A complete system would need to employ a robust and accurate sign-detection system to achieve such results in the real world. This task seems to be doable as well, considering a similar approach was employed here. With some fine-tuning and better data, it can be expected that road authoroties could widely employ such a system to great effect.

The model achieved high ROC scores on real data, which indicates its performance would generalize to real-world scenarios. The videos, and therefore the images used in this thesis were also extracted from real data, which further backs this hypothesis.

One must keep in mind that while the models presented in this thesis achieved a high score when presented with accurate data of the correct type, this does not mean the model can cope with signs or objects of the wrong type being input. This is visible in Figure 5.8, where some signs of the wrong type were included, which could have worsened performance.

The architecture presented in this thesis is currently only able to detect anomalies in a single class of traffic signs at a time. While it may be possible to create a model which can detect such anomalies in multiple types of signs

at a time, or even other types of objects as well; this is not supported by the results presented in this thesis.

The type of sign which is utilized in this thesis (50-signs) was chosen because it was the most abundant, which might cause issues when other types of traffic signs which are rarer are introduced. However, it can be expected that if a robust sign-detection method is used in combination with multiple anomaly detection models, each trained on the respective sign, could alleviate this issue. The difficulty in this approach is finding sufficient data to train every model to the same level of competence, however, this should be possible given the amount of data available.

6.2 Discussion

6.2.1 Limitations

Unclean Data

The data utilized in this thesis suffers in general from being suboptimal. The videos used as the base for the dataset are not of the highest quality, but this might be realistic for a full production system utilizing similar techniques.

The object detection system used to extract signs for anomaly purposes also suffered from the same problems, the model was unable to extract signs at a high enough level of precision to be utilized in a more extensive system, though this might simply be a case of having to train the model for longer and on better data. This lack of data quality made finding quality images to use in the anomaly detection segment difficult; this was done by taking every sign which was detected as the correct class and going through this dataset to remove any unsuited images. While this did not take much time, it did add an additional layer of manual labor to the system, which is undesirable in a proper production system.

It is believed that this did not affect the final results of the anomaly detection model, as these images were mostly removed, however, a proper production system would need to automate this task.

Wrong Class Detections

While the current model functions well for correct data, the behavior is not ideal when dealing with incorrect data. This is mostly due to the object detection segment of the pipeline not being created for this task. The anomaly detection model could conceivably be resistant to incorrect input. Currently, the model is quite likely to report these wrong class detections as anomalies, which would produce a lot of incorrect detections if used in a completed system. Again, this problem might be solvable by improving the object detection model.

Combined this leads to the model being unable to separate true anomalies, and anomalies in the form of the sign being the wrong class.

Environmental Conditions

The weather and light conditions present in the data is primarily daytime imagery on well lit days. This is because the data existing the the SVV database is in these conditions.

The performance of the models presented in this thesis has not been tested on nighttime imagery, or imagery in conditions like rain. Preliminary exploration of the SVV data has shown that most if not all data is collected during the day, as that is the time where the engineers at SVV are working. If cameras were employed on vehicles that perform nighttime work, this might lead to the introduction of nighttime imagery, which is a topic not covered in this thesis.

To draw some parallels, images from dark areas in the videos (e.g. tunnels) have shown the sign becoming completely overexposed, leaving no detail to perform anomaly detection, from which one may postulate that the same effect will happen in nighttime videos. This is further discussed in Section 6.2.4.

Quantity of data

The model presented in this thesis shows good results; however, the quantity of real anomalies used in testing is low. With only 14 real anomalous signs used for verification, the results may not generalize well to a larger quantity of real anomalous signs. There is no way for this to be tested in this thesis, so this remains speculation.

The real anomaly signs used in this thesis also might not be representative enough to ensure that the same performance can be reached in real world scenarios, however this again is speculation. It is equally likely that the images do generalize well, though this requires further testing to be certain.

6.2.2 Contributions

The overarching goal of this thesis was to detect anomalies in traffic signs utilizing only real-world data. This goal was achieved by designing a robust and reusable pipeline, which relies on the combination of object detection and anomaly detection to achieve its results.
This pipeline as illustrated in Figure 4.1, fulfills the needs of this particular domain, and may prove useful in other domains as well.
This thesis highlights a model that can detect anomalies with a ROC-AUC score of 0.92, which indicates the model is able to distinguish normal and anomalous traffic signs accurately on this particular type of sign. This model can potentially be utilized for other types of traffic signs, though this was not extensively explored in this thesis. An example of this can be seen in Figure 5.3.

This approach of using only real-world data to complete a task is a situation that is very common in the real world. This makes the approach used in this thesis of utilizing an object detection model with generated training data which is then applied to real data to extract the data needed for anomaly detection a template which can potentially be applied to many other domains.

Through this approach, it was demonstrated that it is possible to detect anomalous Norwegian traffic signs. This accuracy was demonstrated on a separate dataset than the one which was utilized for training the model, and was able to detect anomalies of different kinds, which signifies the ability of the model to adapt to different kinds of anomalies, as seen in Section 5.3.

Some comparisons with other methods are also showcased. This comparison shows that traditional methods are not suited for this domain, and are outperformed by this model.

6.2.3 Research Questions

The research questions are connected to the goal of the thesis; to assess the viability of computer vision, appropriate methods had to be chosen to tackle this complex domain. This section discusses the research questions, as well as the overall goal.

RQ 1 *Which modern methods are suited for detecting anomalies in images of traffic signs?*

RQ 1 asks which anomaly detection methods are best suited for detecting anomalies in traffic signs. This is answered in Section 3.2. Where several potentially suitable methods were discussed, such as multiple variants of autoencoders.

RQ 2 *To which degree is it feasible to detect anomalies in images of traffic signs extracted from dashcam video?*

RQ 2 is answered through Section 5.3, where it is discovered that in this limited experiment, it was possible to detect anomalous signs with an ROC-AUC score of 0.92. This research questions also encompasses the object detection segment of the experiments, which concluded that the object detection method utilized here achieved an extraction accuracy of 84%, which can be seen in Section 4.2.1.

Goal *Assess the viability of computer vision for anomaly detection in traffic signs.*

To give a short answer to the goal of this thesis, which is evaluating the viability of computer vision to complete this task, the answer is that computer vision is indeed a viable option to solve this problem, and could be employed at scale to assist SVV and other agencies to complete their inspection tasks in more efficiently.

6.2.4 Future Work

NVDB

Seeing as this model simply looks at individual images of signs, it is unable to detect meta-errors such as the signs being placed in the wrong location, the wrong type of sign being placed, or missing signs. This problem could be alleviated utilizing Nasjonal vegdatabank (National Road Database) (NVDB), which contains the correct data for where the signs should be positioned. This database could also be used to aid the object detection model with the expected sign to observe, as this data is also available. A complete production system would need to include such a component in order to be useful for SVV production purposes.

This can be achieved by utilizing the fact that the videos in this thesis contain the geolocation data for each frame of the video, which can be coupled with NVDB, which contains the geolocation for every sign in Norway.

Other road objects

While this method is suitable for signs, it would not be a working solution to uncover anomalies in other road objects such as fences, road markings, and tunnels. These problems probably require individual systems to be created for each, which itself is a daunting task. Creating a model which combines every one of these aspects is the holy grail in infrastructure anomaly detection, and would require a radically different approach than employed in this thesis.

An idea would be to use a different object detection system, one that can extract objects more finely which is required for objects of more advanced topology such as fences. A simple object detection model such as the one used in this thesis is only capable of extracting objects of rectangular shape, which might leave too much noise or unwanted data if applied to advanced geometrical objects such as fences.

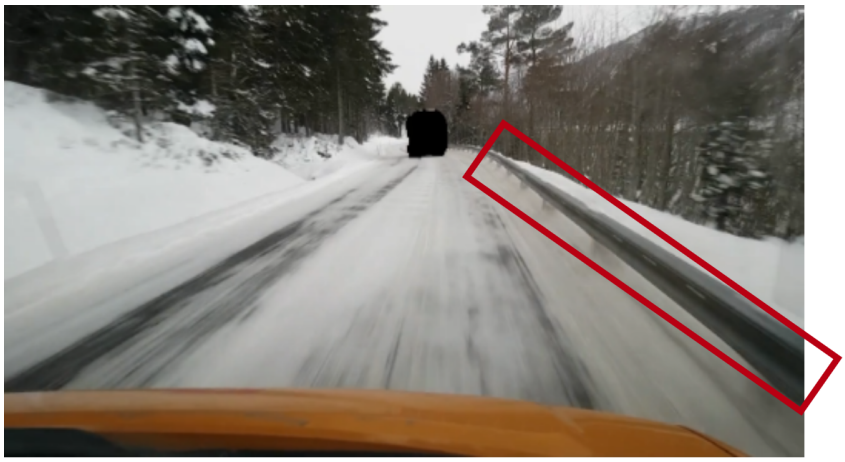


FIGURE 6.1 – Fence Problem

Figure 6.1 illustrates the issue, as a simple rectangle shape is unable to capture the fence without including too much unnecessary data. While this could be alleviated with trapezoid detection boxes, the issue still remains with adapting this data for the anomaly detection model without losing details in the distance due to perspective. One could also attempt to capture smaller parts of the fence at a time, however due to the FPS of the camera, this area could not be too small as to miss pieces of the fence. The issue, in this case, is selecting the appropriate amount of fence to capture at a time, which might have to vary with vehicle speed. It might also be the case that the image quality due to motion blur and camera focus on the closest sections of the fence is too poor to be utilized in an efficient manner, in which case other solutions have to be developed.

Sequential image anomalies

Certain anomalies are hard to detect using a single image, but could conceivably be detected using several sequential images of the same object. This includes signs that have warped plates, crooked signs, or signs which are placed too high or low.

A model could be explored to tackle this problem using alternative architectures such as an Long short-term memory (LSTM) network. Such a solution would need to take the data from several images into account to gain a full understanding of the properties of the sign, potentially allowing it to discover anomalies which are not visible from either image. This domain would become a collective anomaly task, instead of a point anomaly task as explored in this thesis.

Bibliography

Baur, C., Wiestler, B., Albarqouni, S., and Navab, N. (2019). Deep autoencoding models for unsupervised anomaly segmentation in brain mr images. In Crimi, A., Bakas, S., Kuijf, H., Keyvan, F., Reyes, M., and van Walsum, T., editors, *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, pages 161–169, Cham. Springer International Publishing.

Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U. (1999). When is "nearest neighbor" meaningful? In *International conference on database theory*, pages 217–235. Springer.

Cai, Z., Fan, Q., Feris, R. S., and Vasconcelos, N. (2016). A unified multi-scale deep convolutional neural network for fast object detection. In *European conference on computer vision*, pages 354–370. Springer.

Chalapathy, R. and Chawla, S. (2019). Deep learning for anomaly detection: A survey. *CoRR*, abs/1901.03407.

Chalapathy, R., Menon, A. K., and Chawla, S. (2018). Anomaly detection using one-class neural networks. *CoRR*, abs/1802.06360.

Cho, K. (2013). Boltzmann machines and denoising autoencoders for image denoising.

Cozzolino, D. and Verdoliva, L. (2016). Single-image splicing localization through autoencoder-based anomaly detection. In *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6.

Denker, J. S., Gardner, W., Graf, H. P., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. D., Baird, H. S., and Guyon, I. (1989). Neural network recognizer for hand-written zip code digits. In *Advances in neural information processing systems*, pages 323–331.

Downs, J. (2017). Multi-frame convolutional neural networks for object detection in temporal data.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2009). Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645.

Hawkins, D. M. (2004). The problem of overfitting. *Journal of Chemical Information and Computer Sciences*, 44(1):1–12. PMID: 14741005.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., and Murphy, K. (2016). Speed/accuracy trade-offs for modern convolutional object detectors. *CoRR*, abs/1611.10012.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Lee, H. S. and Kim, K. (2018). Simultaneous traffic sign detection and boundary estimation using convolutional neural network. *IEEE Transactions on Intelligent Transportation Systems*, 19(5):1652–1663.

Minsky, M. and Papert, S. A. (1969). *Perceptrons: An Introduction to Computational Geometry*. The MIT Press.

OpenAI, :, Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., de Oliveira Pinto, H. P., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., and Zhang, S. (2019). Dota 2 with large scale deep reinforcement learning.

Papageorgiou, C. P., Oren, M., and Poggio, T. (1998). A general framework for object detection. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 555–562.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2015). You only look once: Unified, real-time object detection.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

Redmon, J. and Farhadi, A. (2018). Yolo v3: An incremental improvement. *CoRR*, abs/1804.02767.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks.

Rosenblatt, F. (1957). The perceptron — a perceiving and recognizing automaton. *Cornell Aeronautical Laboratory*.

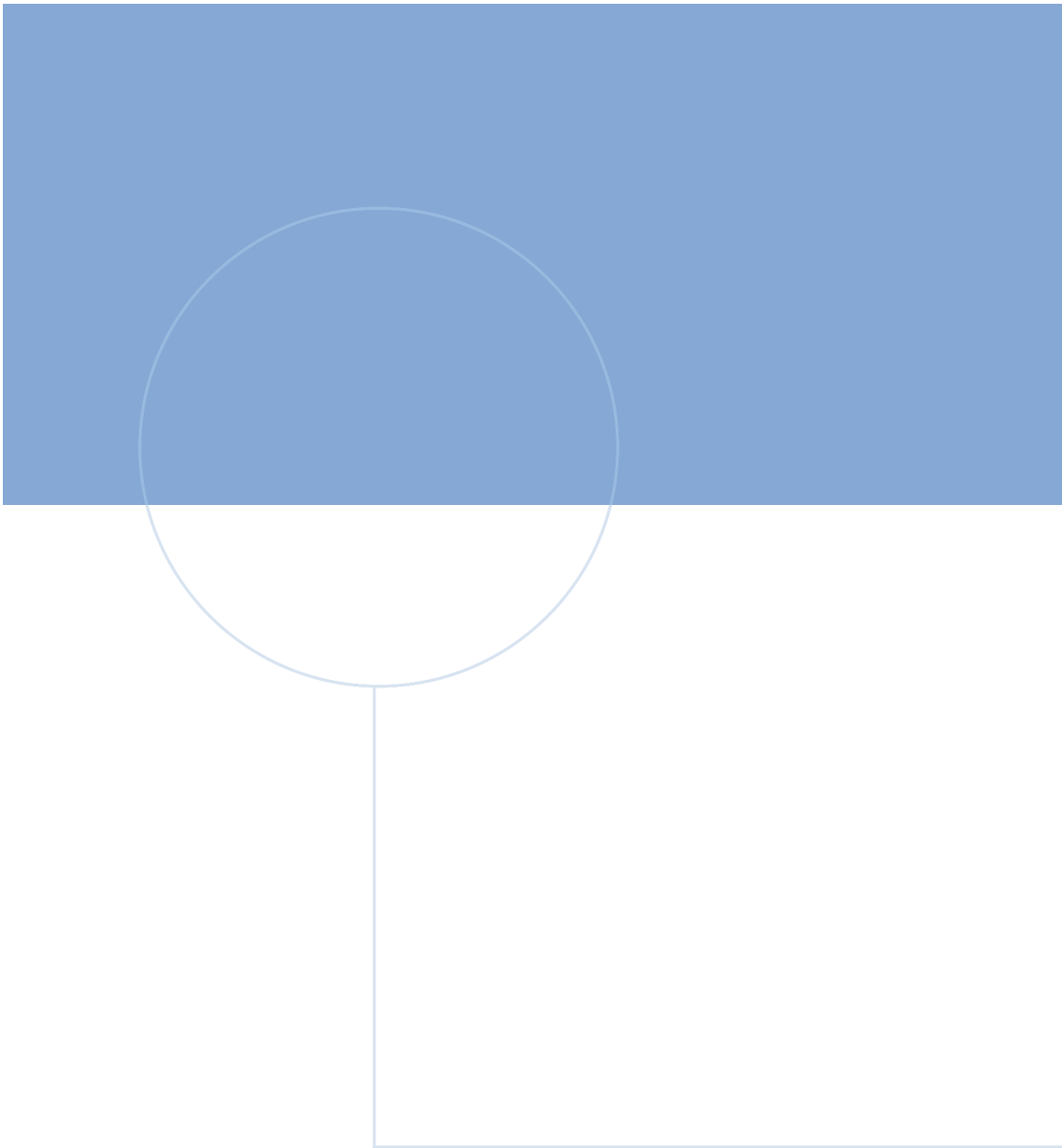
Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E., and Kloft, M. (2018). Deep one-class classification. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4393–4402, Stockholmsmässan, Stockholm Sweden. PMLR.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). *Learning Representations by Back-Propagating Errors*, page 696–699. MIT Press, Cambridge, MA, USA.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T. P., Simonyan, K., and Hassabis, D. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815.

Timofte, R., Zimmermann, K., and Van Gool, L. (2014). Multi-view traffic sign detection, recognition, and 3d localisation. *Machine vision and applications*, 25(3):633–647.

Viola, P. and Jones, M. (2004). Robust real-time face detection. *International Journal of Computer Vision*, 57:2.



NTNU

Norwegian University of
Science and Technology