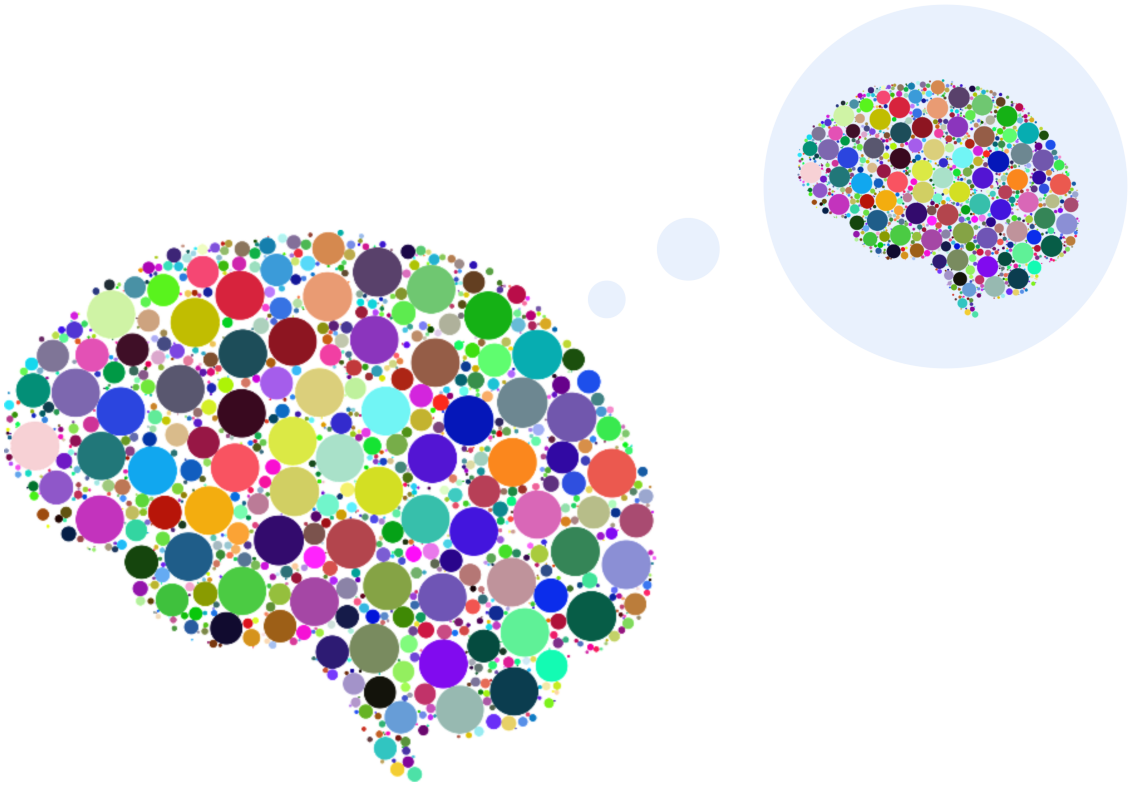
CHALMERS
UNIVERSITY OF TECHNOLOGY



# Input Verification for Deep Neural Networks

## Detection of data unfamiliar to deep neural networks

Master's thesis in Electrical Engineering

Mattias Landgren & Ludwig Tranheden

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Master's thesis 2018:EX040

# Input Verification for Deep Neural Networks

Detection of data unfamiliar to deep neural networks

MATTIAS LANDGREN & LUDWIG TRANHEDEN


CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2018

Input Verification for Deep Neural Networks
Detection of data unfamiliar to deep neural networks
MATTIAS LANDGREN & LUDWIG TRANHEDEN



Supervisor: Roman Sokolovskii, Department of Electrical Engineering
Supervisor: Jens Henriksson, Semcon Sweden AB
Examiner: Giuseppe Durisi, Department of Electrical Engineering

Master's Thesis 2018:EX040
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Visualisation of metacognition; thinking about thinking.

Typeset in LaTeX
Gothenburg, Sweden 2018

Input Verification for Deep Neural Networks
Detection of data unfamiliar to deep neural networks
MATTIAS LANDGREN
LUDWIG TRANHEDEN
Department of Electrical Engineering
Chalmers University of Technology

# Abstract

As deep learning systems are more frequently being applied to safety-critical domains, the protection against irrelevant and malicious data is of greater significance than ever before. By applying an algorithm, or supervisor, which removes this irrelevant data from the deep learning system, the credibility of the system is increased. The purpose of this project was to find and evaluate existing algorithms, and to develop a new method which can protect a neural network from irrelevant inputs. The thesis is centered on a hypothesis of capturing the learned domain of a CNN by using adversarial examples.

A thorough literature review resulted in 18 analyzed methods of which five methods were implemented. The methods were evaluated on three scenarios: MNIST versus Omniglot; CIFAR-10 versus CIFAR-100; Retinal OCT-images with a held out class. Results of the study point out the difficulties in using adversarial examples to represent the infinite set of novelties. Rather than using adversarial examples to train algorithms, only utilizing the existing training data to define what is normal is better suited for the protection against undesired inputs.

Methods that use the final layer activations of the neural network to detect abnormalities achieve the best results. A new supervisor, not only using the final layer but all layer activations to detect abnormalities, was created. The analysis found that valuable information can be found in the earlier layers of the network and that this information can be used for more than just novelty detection. The developed supervisor shows performance comparable to the better supervisors in the first two scenarios but show difficulties with the Retinal OCT scenario.

Keywords: verification, neural networks, machine learning, adversarial examples, novelty detection.

# Acknowledgements

Firstly, we would like to thank our supervisors Roman Sokolovski of the department of Electrical Engineering at Chalmers and Jens Henriksson at Semcon. We also wish to send our thanks to our examiner Giuseppe Durisi who helped guide and calm us in the beginning of the project when the project felt overwhelming.

A big thank you is sent to Laura Masaracchia and Jens Henriksson who aided us when our simulation laptop's fans broke. Without the use of the stationary computer we would be in deep water.

Finally, we would like to thank our friends at Chalmers who have gilded the previous five years of late nights and early mornings, times of joy and despair.

Mattias Landgren & Ludwig Tranheden, Gothenburg, June 2018

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

**AE** Autoencoder.
**AUC** Area Under Curve.

**CDF** Cumulative Distribution Function.
**CIFAR** Canadian Institute For Advanced Research.
**CNN** Convolutional Neural Network.
**CNV** Choroidal Neovascularization.

**DBF** Deep Belief Network.
**DCGAN** Deep Convolutional Generative Adversarial Network.
**DME** Diabetic Macular Edema.
**DNN** Deep Neural Network.

**EA** Evolutionary Algorithm.

**GAN** Generative Adversarial Network.

**IFCS** Intelligent Flight Control System.
**ILSVRC** ImageNet Large Scale Visual Recognition Competition.

**L-BFGS** Limited-Memory Broyden–Fletcher–Goldfarb–Shanno.

**ML** Machine Learning.

**NIST** National Institute of Standards and Technology.
**NN** Neural Network.

**OCT** Optical Coherence Tomography.

**PCA** Principal Component Analysis.

**RBF-SVM** Radial Basis Function-Support Vector Machine.
**ROC** Receiver Operating Characteristics.

**SMT** Satisfiability Modulo Theory.
**SVC** Support Vector Classifiers.
**SVDD** Support Vector Data Description.
**SVHN** Street View House Number.
**SVM** Support Vector Machine.

**VAE** Variational Autoencoder.
**VGG** Visual Geometry Group.

# 1
# Introduction

## 1.1 Background

Deep Learning systems are increasingly being deployed in a large variety of safety-critical domains such as Autonomous Driving (Bojarski et al. 2016) and Medical Diagnosis (Ting et al. 2017). The systems cannot by themselves detect new data and would infer it based on what they are trained on. An algorithm able to detect irrelevant input data or data out of the ordinary, will enable autonomous systems to prevent accidents, malware or other critical events from happening (Xu et al. 2017). The system would be able to alert its driver, user, operator or take other fail-safe actions when critical events occur. These events could consist of novelties, which differs from the relevant data. Scheirer et al. (2013) argues that while it is possible to train networks with an "other" class, the number of unknown objects in this class is infinite. It is therefore better to achieve a good representation of the finite learned space rather than the infinite unknown.

Two fields where the importance of an input-verified inference system is critical are autonomous driving and medicine. In both cases it is very important that systems intended for a specific input domain are not given inputs outside this domain. The result could be high-confident replies on questions the system knows nothing about. An example is a Machine Learning (ML) classifier trained to identify diseases in the skin being fed an image of an unseen skin disease. The new disease does not belong to the trained domain of the classifier and the classifier might infer with high confidence that no diseases are present in the image. The ambition is that a novelty detection algorithm might recognize the novel input (of the new disease) as something unseen and can alert an operating dermatologist for inspection.

Several advances towards the detection of novelties as well as the detection of adversarial examples have been made, see the literature review in Chapter 3. Meanwhile, work is done towards verifying the performance of neural networks and assuring that the outputs from the networks can be trusted (Huang et al. 2016). If the advances in these fields were combined, networks could be more trustworthy by removing irrelevant inputs from the equation.

Even when neural networks achieve impressive performances, they can be vulnerable to small variations and rare corner cases (Amodei et al. 2016). Changing one pixel or applying transformations which does not change the input image in a drastic way

could make the system react in an unpredictable way which could lead to hazardous and even fatal mistakes. Examples of these perturbations are fog, rain drops or snow on the vehicle cameras. An example of a fatal event is when a Tesla autonomous car was involved in a fatal crash since the system did not detect a white truck against a cloudy bright sky (Boudette & Vlasic 2017).

This thesis focuses on utilizing methods found in the fields of anomaly detection and the defense against adversarial examples to enable rejection of data not belonging to the training data distribution of the neural network. To train an algorithm to reject novel inputs would require training data from the infinite set of all possible inputs, which is impossible. A relatively new approach towards the defense against adversarial examples is the use of adversarial examples created from the existing training data distribution to find the model's decision boundaries (Feinman et al. 2017). The ambition is to use these decision boundaries to reject novelties.

Existing algorithms are re-implemented and evaluated on three scenarios: MNIST vs Omniglot; CIFAR-10 vs CIFAR-100; Retinal OCT-images. The key contributions of this study are

- a literature review of 20 articles containing insights and methods useful for input verification,
- a thorough analysis of six methods using different approaches that can be used for input verification,
- a new algorithm for the detection of novel inputs built upon the combination of insights from, and characteristics in, evaluated algorithms.

### 1.1.1 Primer on Machine Learning

ML can be divided into three different areas: supervised, unsupervised and semi-supervised. This thesis will only focus on supervised ML used in classification tasks. Supervised ML can be formulated as follows. Given $n$ inputs $x^1, ..., x^n$ and $n$ corresponding labels/classes $y^1, ..., y^n$, the objective is to find a function $\hat{f}$ that approximates the target function $f(x^i) = y^i$ for $i = 1, ..., n$. A simple classification example with two-dimensional inputs and two classes, x and o, is illustrated in Figure 1.1.

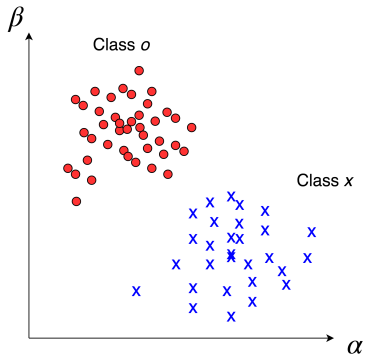



**Figure 1.1:** Illustration of a two-dimensional classification problem with two classes.

By defining the labels as being equal to 1 if the input belongs to the o class and 0 if the input belongs to the x class, the problem can be formulated as finding an approximation, $\hat{f}$, to the target function $f$. The target function is characterized by

$$f(x) = \begin{cases} 1 & \text{if } x \in \text{Class } o, \\ 0 & \text{if } x \in \text{Class } x. \end{cases} \tag{1.1}$$

For the $\hat{f}$ function to be formulated, the data points in Figure 1.1 need to be separated. Usually, a ML algorithm would be used to find that separation, but in this simple example the classes are visually separable by a line. The line with equation $w_1\alpha + w_2\beta - b = 0$ is illustrated in Figure 1.2.

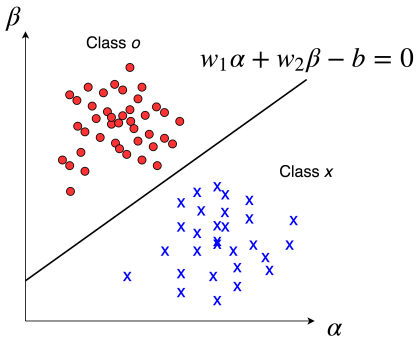



**Figure 1.2:** Illustration of the separation between the two classes in the two-dimensional classification problem.

Now the function, $\hat{f}$, can be formulated in terms of a score and a threshold. The score is some number, optimally, being different for different classes. In this case the score will be a measure of the likelihood that an input belongs to the class o. The threshold is a number that decides the classification by comparing it to the previously mentioned score. In this simple problem the score for an input $x = (x_1, x_2)^T$ will be

$w_1x_1 + w_2x_2$ and the threshold $b$. The $\hat{f}$ function is described below in Equation 1.2 .

$$\hat{f}(x) = \begin{cases} 1 & \text{if } w_1x_1 + w_2x_2 \geq b, \\ 0 & \text{if } w_1x_1 + w_2x_2 < b. \end{cases} \tag{1.2}$$

In this simple problem in two dimensions it was possible to simply draw a line separating the two classes, but usually the dimension of the data is much higher. For example, in a $128 \times 128 \times 3$ image the number of input features is 49152. It is no longer feasible to visually determine a line separating two classes in a space of that dimension. Some other way is needed to mathematically formulate the score, threshold and hence the classification. This is where ML takes over. A ML algorithm is self-adjusting its parameters to minimize a loss function using the existing data. In the example above it would translate to a loss function, for example assigning +1 for classifying an input to the wrong class and -1 for a correct classification. The parameters to be adjusted would be $w_1$, $w_2$ and $b$. A very powerful and expressive ML algorithm is deep learning. The algorithm consists of several layers with sometimes millions of parameters being optimized and learned to perform impressive tasks with high-dimensional data.

### 1.1.2 SMILE II-Project

Semcon is currently part of a project called SMILE II, which seeks to verify that incoming data is representative within the training data. The model should only act on data that is part of the same distribution as the data used during training. A method that indicates whether an output is unreliable, given the input and Deep Neural Network (DNN), will henceforth be referred to as a supervisor or supervising method. A schematic over the input verification process, including the supervisor, is shown in Figure 1.3.

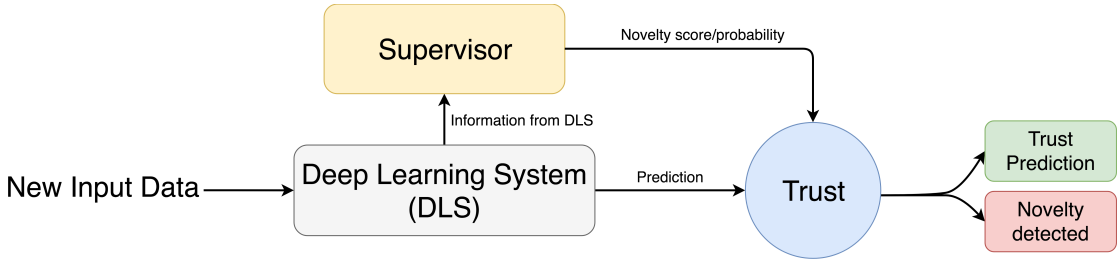



**Figure 1.3:** A schematic of the input verification process using a supervisor.

## 1.2 Objective

The objective of the thesis is to analyze related work and create a summary of the current state of development. Using the analysis, a new method for input verification is proposed.

## 1.3 Aim

The aim of the thesis is to create a method of supervising inputs to DNNs. This supervisor should, given an input, be able to verify that it belongs to the training data distribution of the DNN or reject it.

## 1.4 Delimitations

Since the SMILE II project mainly focuses on self-driving cars and their abilities to analyze their surroundings, the project will limit itself to only focus on DNNs solving classification tasks. The thesis will not focus on the development of well-performing neural networks. The project limits itself to use the programming language Python.

## 1.5 Specification of Issue Under Investigation

To achieve the goals of this thesis project, certain problems need to be solved:

- Compile a summary of existing possible supervising methods
  - i What possible methods of verifying and detect novelties for DNNs exist today?
  - ii What are their characteristics?
- Evaluate supervisors on a variety of datasets
  - i Which types of datasets are suitable and commonly used?
  - ii Which insights may input verification of such datasets provide?
- Develop a new supervising method
  - i What characteristics can be extracted from the tested supervisors?
  - ii How can a DNN's behavior to novelties be understood?
  - iii How will the new method be implemented?

## 1.6 Some Clarifications

Anomalies go under several names e.g. outliers, discordant observations, exceptions, aberrations, surprises, contaminants and could be compared to novelty detection (Chandola et al. 2009). As anomaly detection is used to detect data that might cohere to the normal data domain but do not lie in the regions of normal data, novelty detection is used to detect previously unseen data. However, because the algorithms used in novelty- and anomaly detection are often similar (Pimentel et al. 2014) in the subject of this thesis, the differences are small enough to be dismissed and the terms will be used interchangeably.

## 1.7 Thesis Outline

In the second chapter, the theory of the thesis is presented. The chapter consists of descriptions of necessary preliminaries for understanding the report. Following

is the literature review chapter, shortly summarizing relevant papers. In chapter four, the thesis method including the comparisons of existing algorithms for novelty detection, evaluation of characteristics and the experimental setups are described. The fifth chapter presents the results of the selection process, experimental process and the creation of a new supervisor. The last chapters will present discussions about the work and future work along with the conclusion of the thesis.

# 2
# Theory

In this chapter, theory necessary to follow the thesis is presented. The theory presented here will later be used in the implementations of supervisors.

## 2.1 Deep Neural Networks

Consider the task of learning how to map inputs $x^k \in \mathbb{R}^n$, $k = 1, ..., N$ to predetermined outputs $y^k \in \mathbb{R}^m$, $k = 1, ...N$. That is, to find a function $f : \mathbb{R}^n \to \mathbb{R}^m$ such that $f(x^k) = y^k$, $\forall k$ for every example. DNNs attempts to represent this mapping by propagating the inputs through multiple layers of linear transformations and nonlinear activation functions. The many parameters (weights $w$ and biases $b$) of the network are optimized based on some error metric and optimization algorithm. A well-used metaphor for the method is presented in Figure 2.1. It shows a simplified image of a neuron with the cell body acting as a neuron in deep neural networks and the dendrites and axons represents the connections between the neurons. In this section two general types of deep neural networks are presented as well as certain architectures of relevance.

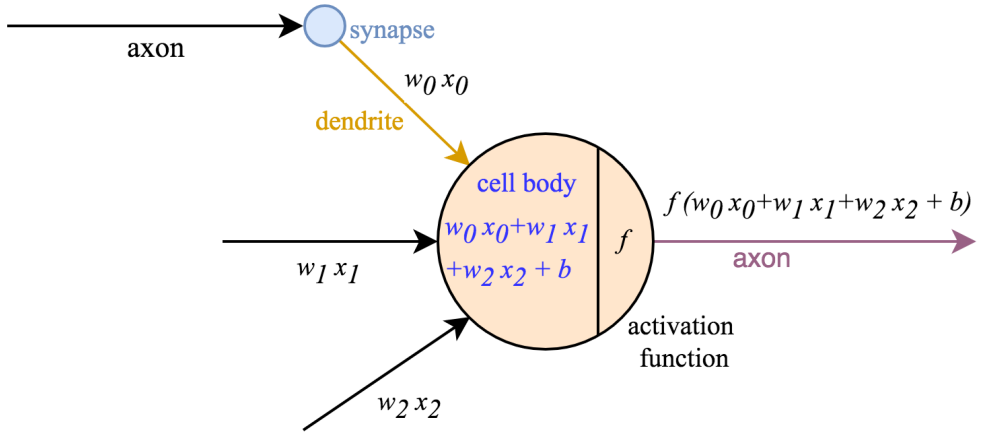



**Figure 2.1:** A mathematical model of a neuron.

### Softmax

The last layer of many classifying neural networks is generally the Softmax layer. The Softmax output can be interpreted as a normalized probability, $P$, given by the

neural network that a certain input, $x$, belongs to a class, $i$. The probabilities are calculated in

$$P(\hat{y} = i|\mathbf{x}, \mathbf{W}) = \frac{e^{\mathbf{x}^T \mathbf{W}_i}}{\sum_j e^{\mathbf{x}^T \mathbf{W}_j}} \tag{2.1}$$

where $\hat{y} = i$ is the network prediction that input $x$ belongs to class $i$. The operation $\mathbf{x}^T \mathbf{W}_i$ is the inner product of the network leading to the output node $i$. The equation results in probabilities which sums to one.

### 2.1.1 The Optimization Problem

Training a DNN corresponds to solving an unconstrained optimization problem. A loss is defined, which corresponds to the objective, and the problem is to minimize it. Consider the general case:

- there exists $n$ inputs of dimension $m$: $x^i \in \mathbb{R}^m, i = 1, 2...n$,
- there exists $n$ labels of dimension $k$: $y^i \in \mathbb{R}^k, i = 1, 2...n$,
- the network contains a set of parameters $W$ of some dimension defining its function $f(x|W)$: $\mathbb{R}^m \to \mathbb{R}^k$,
- the loss function is denoted by $L(\hat{y}) := E(\hat{y}, y)$: $\mathbb{R}^k \to \mathbb{R}$ where $y^j$ is one of the labels and $\hat{y}^j$ is a prediction of $x^j$.

The optimization problem can then be formalized as

$$\min_W \sum_{i=1}^{n} L\Big(f(x_i|W)\Big). \tag{2.2}$$

The optimization schemes used in neural networks are almost exclusively gradient-based and uses backpropagation. Usually the optimization is carried out over batches, which consists of a sample from the $n$ inputs and labels. One epoch corresponds to have optimized the parameters over enough batches such that all of the $n$ inputs have been included.

### 2.1.2 Fully Connected Neural Networks

Consider the following scenario:

- inputs $x^k \in \mathbb{R}^n$, $k = 1, ..., N$
- labels $y^k \in \mathbb{R}^m$, $k = 1, ..., N$
- the objective is to find a function $f : \mathbb{R}^n \to \mathbb{R}^m$ such that $f(x^k) = y^k$, $k = 1, ..., N$.

The approximation of the function $f$ will in this context be a fully connected network consisting of an input layer, an output layer and in between, one or more hidden layers. A fully connected network with one hidden layer is illustrated in Figure 2.2. The biases are omitted from this explanation for convenience and because they can be handled in the same way as the weights. By simply padding every layer (excluding the output layer) with a 1, the weight assigned to this node will be the bias. Note that the superscripts are not used in the figure, nor will be used for the remainder of the explanation for readability.

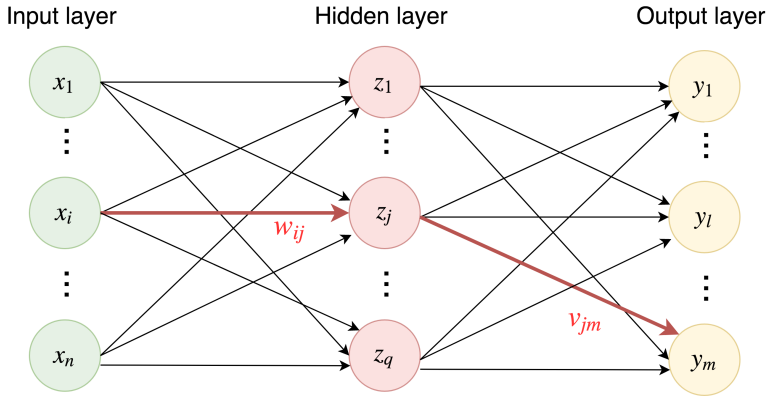



**Figure 2.2:** A fully connected neural network with one hidden layer.

To further define the mathematical context and explain the notation in Figure 2.2, the following notation is used (note that the index of vectors and matrices is denoted by subscript):

- the inputs $x_i \in \mathbb{R} \quad i = 1, ..., n$,
- the hidden layer activation $z_j \in \mathbb{R} \quad j = 1, ..., q$,
- the network output $\hat{y}_l \in \mathbb{R} \quad l = 1, ..., m$,
- the weight connecting $x_i$ and $z_j$ $w_{ij} \in \mathbb{R} \quad i = 1, ..., n, j = 1, ..., q$,
- the weight connecting $z_j$ and $\hat{y}_k$ $v_{jk} \in \mathbb{R} \quad j = 1, ..., q, k = 1, ..., m$.

The forward propagation from input to hidden layer can then be described by

$$z_j = g\left(\sum_{i=1}^{n} w_{ij} x_i\right), \quad j = 1, ..., q \tag{2.3}$$

where $g$ is an activation function. Most commonly used in this thesis are the ReLU activation function defined as $g(\alpha) = max(0, \alpha)$ and the leaky ReLU defined with a parameter $\beta$ as $g(\alpha) = max(\beta\alpha, \alpha)$. The propagation from the hidden layer to the output layer can be described by

$$\hat{y}_l = f\left(\sum_{i=1}^{q} v_{il} z_i\right), \quad l = 1, ..., m \tag{2.4}$$

where $f$ is an activation function. Since the problems presented in the thesis are predominantly classification problems, the Softmax activation function in (2.1) is most commonly used. It is however defined as taking the entire vector as input. In (2.4) this would mean that $f$ is removed and then applied to the entire vector $\hat{y}$ to get the network output. How to compute the gradients for the example with one hidden layer is described below.

### Gradients

With the labels defined as $y_i$ for $i = 1, ..., m$ (without superscript). A loss function comparing the output of the network and the label is denoted by

$$L(\hat{y}), \; L : \mathbb{R}^m \to \mathbb{R}. \tag{2.5}$$

The network learns how to approximate the target function $f$ by minimizing the predefined loss function with respect to the weights. The optimization methods most commonly use the gradients. The gradients can in turn be effectively computed by the backpropagation algorithm. First, consider the output weights $v_{ij}$. Through the chain rule the gradients can be calculated as

$$\frac{\partial L}{\partial v_{jk}} = \frac{\partial L}{\partial \hat{y}_k}\frac{\partial \hat{y}_k}{\partial v_{jk}} \quad j = 1, ..., q, k = 1, ..., m \tag{2.6}$$

In the same way the gradient for input weights $w_{ij}$ can be calculated as

$$\frac{\partial L}{\partial w_{ij}} = \sum_{l=1}^{m} \frac{\partial L}{\partial \hat{y}_l}\frac{\partial \hat{y}_l}{\partial w_{ij}} \quad i = 1, ..., I, j = 1, ..., J \text{ where} \tag{2.7}$$

$$\frac{\partial \hat{y}_l}{\partial w_{ij}} = \frac{\partial \hat{y}_l}{\partial z_j}\frac{\partial z_j}{\partial w_{ij}}. \tag{2.8}$$

The calculation of gradients and optimization of a fully connected network with arbitrary number of hidden layers is described below.

### 2.1.3 The Backpropagation Algorithm

Consider a fully connected network and the notation used above. The fully connected network has arbitrary number of hidden layers (of arbitrary dimensions) denoted by $l = 1, .., L-1$, where $l = 0$ is the input layer and $l = L$ is the output layer. The weighted input of node $j$ in layer $l$ is denoted by $\alpha_j^l$ and is defined by

$$\alpha_j^l = \sum_i w_{ij}^l z_i^{l-1} \tag{2.9}$$

where $z_i^{l-1} = g(\alpha_i^{l-1})$ is the activation of node $i$ in layer $l-1$ and $w_{ij}$ the weight connecting them. Now the error for neuron $j$ in layer l (the affect the neuron has on the loss function in (2.7)) can be defined as

$$\delta_j^l := \frac{\partial L}{\partial \alpha_j^l}. \tag{2.10}$$

Using the error notation, the output layer error can be expressed as

$$\delta_j^L = \frac{\partial L}{\partial \alpha_j^L} = \frac{\partial L}{\partial z_j^L}\frac{\partial z_j^L}{\partial \alpha_j^L} = \frac{\partial L}{\partial \hat{y}_j} g'(\alpha_j^L). \tag{2.11}$$

For an arbitrary hidden layer, $l$, the error can be expressed as

$$\delta_j^l = \frac{\partial L}{\partial \alpha_j^l} = \sum_i \frac{\partial L}{\partial \alpha_i^{l+1}}\frac{\partial \alpha_i^{l+1}}{\partial \alpha_j^l} = \sum_i \frac{\partial \alpha_i^{l+1}}{\partial \alpha_j^l}\delta_j^{l+1}. \tag{2.12}$$

The second to last term on the right of (2.12) (without differentiation) can be expanded as

$$\alpha_i^{l+1} = \sum_j w_{ji}^{l+1} z_j^l = \sum_j w_{ji}^{l+1} g(\alpha_j^l). \tag{2.13}$$

Differentiation yields

$$\frac{\partial \alpha_i^{l+1}}{\partial \alpha_j^l} = w_{ji}^{l+1} g(\alpha_j^l). \tag{2.14}$$

Now, substituting (2.14) into the expression for the layer error in (2.12) the error can be expressed as

$$\delta_j^l = \sum_i w_{ji}^{l+1} g(\alpha_j^l) \delta_j^{l+1}. \tag{2.15}$$

Note that a recursive expression appears; the layer error for earlier layers can be computed by using the error for later layers. The gradient for an arbitrary weight in an arbitrary layer can be computed as

$$\frac{\partial L}{\partial w_{ij}^l} = \frac{\partial L}{\partial \alpha_j^l} \frac{\partial \alpha_j^l}{\partial w_{ij}^l} = \delta_j^l z_i^{l-1}. \tag{2.16}$$

Now the backpropagation algorithm can be formulated as:

1. Propagate the input forward through the network and compute and store the $z_j^l$
2. Compute and store output layer errors $\delta_j^L$.
3. Backpropagate the output layer errors to compute the errors for all layers ($\delta_j^l$) using (2.15).
4. Update the weights using the chosen optimization method, for example by gradient descent: $w_{ij}^l \leftarrow w_{ij}^l - \eta \delta_j^l z_i^{l-1}$
5. Go to 1 and use the next input.

For more details about fully connected networks and backpropagation see Goodfellow et al. (2016, pp.164-223).

### 2.1.4 Convolutional Neural Networks

Opposed to fully connected neural networks using matrix multiplications between layers, Convolutional Neural Networks (CNN) consist of filters with height and width. One filter can be seen as a small fully connected layer where the weights being optimized are the filter values, see the red box in Figure 2.3 for a filter example. The filters are convoluted over the input pixels and the dot product between the filter values and the pixels is calculated as in Figure 2.3. The input map can be seen as an image with the entries representing pixel values. The outputs create an activation map, usually called feature map, that gives larger outputs on inputs resembling the filter feature as in Figure 2.3. That means a filter trained to find corners will return higher activations when fed an image patch containing a corner. In deep learning, several filters are used on the same input map and each produce an individual feature map. That way different filters learn to produce different features from the same input. CNNs are like fully connected networks optimized using the backpropagation algorithm (though not identical to the one used for fully connected networks).

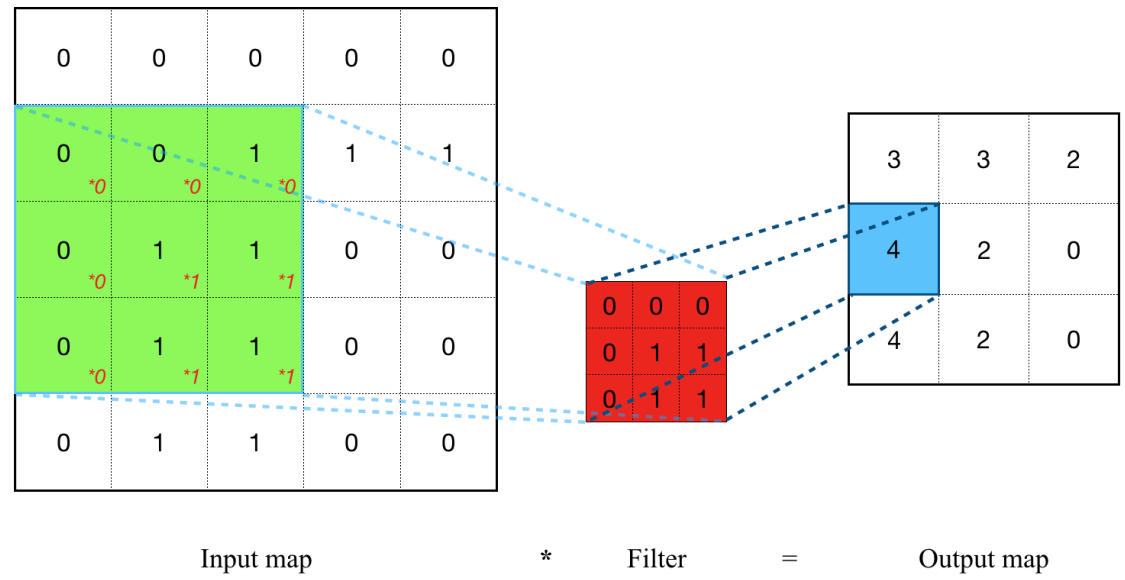



**Figure 2.3:** The computation of one value in the feature map produced by a filter through dot product.

In Figure 2.4 an example of a CNN is shown. The first three layers of the network are convolutional layers that each produce a number of feature maps. The number of feature maps produced by each filter is reflected in the blocks (feature maps) thicknesses. The final three layers are fully connected layers with the final one consisting of as many nodes as there are possible classes. The final layer outputs are called logits, and consist of values later received by the Softmax function.

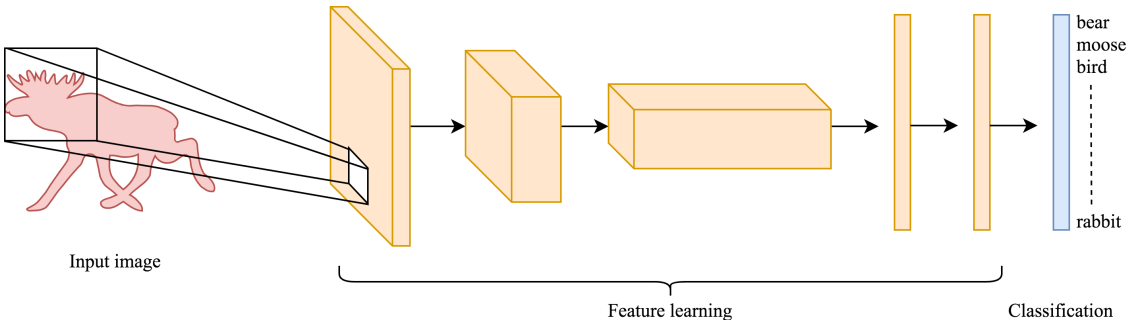



**Figure 2.4:** An example of a CNN used for classification with three convolutional layers and three fully connected layers.

For more details about CNNs see Goodfellow et al. (2016, pp. 326-366).

### 2.1.5 Generative Adversarial Networks

A Generative Adversarial Network (GAN) consists of two components, one discriminator and one generator, both being neural networks. The discriminator functions as a discriminative algorithm; given an input it classifies it. The generator functions as a generative algorithm. It does the opposite to a discriminative algorithm; given

a label/class it generates the input.

Assume there exists a training data distribution. In a GAN, the generator's task is to generate new instances belonging to the data distribution from a random vector sampled from a latent space (some predefined space). At the same time, the discriminator's task is to separate instances belonging to the data distribution (real) from the artificially generated ones from the generator (fake). The GAN can be thought of as of the combination of a counterfeiter (generator) and a cop (discriminator) where the counterfeiter tries to create items that fools the cop, and the cop tries not to be fooled by the items. The concept can be visualized as in Figure 2.5 with the notation listed below.

- $p_{data}$ - the data distribution,
- $D$ - the discriminator parameters (weights),
- $D(\mathbf{y})$ - the by discriminator estimated probability that $\mathbf{y}$ belongs to the data distribution $p_{data}$,
- $p_z$ - the latent space distribution,
- $G$ - the generator parameters (weights),
- $G(\mathbf{z})$ - the generated example from $\mathbf{z} \sim p_z(\mathbf{z})$.

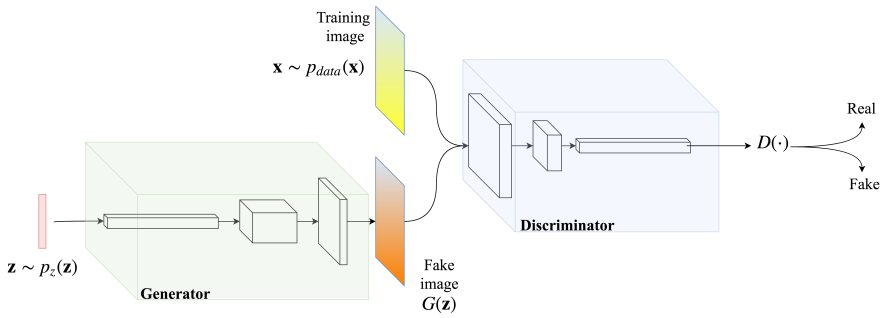



**Figure 2.5:** Illustration of an generative adversarial network architecture.

The training of a GAN can be described by (2.17).

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(\mathbf{x})}[\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \quad (2.17)$$

The maximization over the first term with respect to the discriminator parameters states that high probabilities should be assigned to inputs sampled from the data distribution. The maximization over the second term with respect to the discriminator parameters states that low probabilities should be assigned to input generated by the generator. The minimization of the generator parameters only affect the second term and translates into trying to make the discriminator assign high probabilities to inputs generated by the generator. The discriminator and generator are trained alternatively by maximizing and minimizing (2.17), respectively, using the backpropagation algorithm. For more information about GANs see Goodfellow et al. (2014).

### 2.1.6 Cross-Entropy Loss

For classification tasks, the loss used to train the neural network or other ML algorithm is the cross-entropy loss. Assume that the task is a $n$-class classification task. The label for class $m \in 1, ..., n$ can be described by (2.18).

$$y_{m,i} = \begin{cases} 1 & \text{if } i = m, \\ 0 & \text{otherwise.} \end{cases} \tag{2.18}$$

Let the probabilities of every class predicted by the network for an example with class $m$ be $p_1, ..., p_n$. The cross-entropy loss can then be formulated as

$$-\sum_{i=1}^{n} y_{m,i} \log(p_i). \tag{2.19}$$

### 2.1.7 Dropout

Dropout (Hinton et al. 2012$a$) is a regularization technique that is used to keep neural network from overfitting. It is applied between layers in the neural network and amounts to removing a fraction of the neurons from the layer before the dropout, meaning they never reach the layer after the dropout. The dropout rate is the probability of randomly dropping one neuron. When applying dropout, training is performed on various subsets of the full network and makes the network learn more robust features. It can also be used to approximate uncertainty (Gal & Ghahramani 2015).

### 2.1.8 Batch Normalization

Batch normalization is a technique applied to specific layers, where the layer of outputs are normalized before the activation function in order to give them a mean of zero and variance of one, unless the algorithm learns that other values are better suited. The algorithm was invented by Ioffe & Szegedy (2015) to solve the problem of internal covariate shift; the phenomenon of changes in each layer's input distributions such that learning rates need to be low and the initialization of weights need to be delicate. A good metaphor is the whisper game, where the first sentence rarely is the same after a couple of persons whispered what they heard. Batch normalization solves this problem and the consequences are lower learning rates and a decrease in training time. The algorithm also works as a regularizer which prevents the network from overfitted.

## 2.2 Principal Component Analysis

Given a set of data points with $n$ number of dimensions, Principal Component Analysis (PCA) finds a direction on which the data points has the maximum variance. It then continues to find $n - 1$ more directions, orthogonal to the first direction which in ranking order have the next largest variance. These directions are called principal components. The analysis is regarded as an eigenproblem, which enables

the analysis of corresponding eigenvalues to each principal component. If a principal component has a low eigenvalue, it is interpreted to have low impact on the dataset and can henceforth be ignored. Therefore, PCA is today used in machine learning algorithms as a dimensionality reduction method. For more information about PCA see Abdi & Williams (2010).

## 2.3 Logistic Regression

Logistic regression is a regression analysis suitable for problems where one variable is binary and another is not. The algorithm is used in machine learning problems as a binary classifier, where the model makes predictions. An example problem is the prediction whether an animal is a cat or a dog given the length of its ears. The model fits a logistic function to a large set of samples of ear lengths and the corresponding binary class of cat or dog. Given a new input length of an ear, the model predicts whether it belongs to a cat or a dog. For more information about logistic regression, see (Murphy 2014, pp. 245-281).

## 2.4 Support Vector Machine/Classifier

An Support Vector Machine (SVM) is a supervised machine learning algorithm that given labeled examples creates a hyperplane separating the examples by class. A Support Vector Classifiers (SVC) uses this hyperplane to classify new examples. For more information regarding SVMs see Steinwart et al. (2008).

## 2.5 Adversarial Examples

Adversarial examples were first discovered by Szegedy et al. (2013) as small perturbations leading to misclassifications by neural networks. These perturbations could be dead pixels, blurring or other distortions in images, small enough such that humans are able to correctly interpret the image but large enough for the network to incorrectly classify it. In Figure 2.6 an example of a normal image and a corresponding adversarial image can be seen. While Szegedy et al. (2013) were uncertain whether these cases occurred naturally in real-life situations, Zheng et al. (2016) proved the occurrence in video frames.


**(a)** **(b)**

**Figure 2.6:** Normal image and corresponding adversarial example.

# 3
# Literature Review

The literature review revolves around existing methods of finding inputs the network cannot handle confidently. Presented first are anomaly and novelty detection methods. While anomaly detection is used to detect irregular data such as outliers and bad readings, novelty detection is used to detect unfamiliar data. Finally, articles in the relatively new topic of adversarial detection are studied.

## 3.1 Anomaly and Novelty Detection

Algorithms, or existing methods, developed for novelty and anomaly detection are presented in this section. The methods are divided into three topics depending on the algorithm approach: Reconstruction-, Domain- and Probabilistic-based detection.

### 3.1.1 Reconstruction-Based Detection

Reconstruction-based methods autonomously model the underlying data without making any explicit assumptions on the data. Autoencoders (AE), Variational Autoencoders (VAEs) and GANs belong to this class of methods since the purpose is to construct or reconstruct samples belonging to the distribution of the training data.

#### Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction

Sakurada & Yairi (2014) developed a dimensionality reduction algorithm for anomaly detection, and tested it on high-dimensional data partly generated from the Lorenz system and partly from real spacecraft telemetry data. The algorithm uses an AE to reduce the number of dimensions in the input data. To use the algorithm, one has to assume that the data has features correlated to each other and when mapped to a lower dimension, normal and anomalous inputs will be notably different. The AEs compress the input data into a latent subspace of lower dimension, and tries to reconstruct it in the output and optimize by minimizing a reconstruction error. This error is used as the anomaly score and shows large values for anomalous inputs. Increased performance is also achieved by developing denoised AEs; autoencoders with more hidden units and noise added to the input. Apart from developing an anomaly detector, Sakurada & Yairi (2014) visualizes the activations of several neurons in a hidden layer. The figures show that anomalous inputs activate differently from normal ones. This visualization of learned features in the hidden layer has not been done before (Sakurada & Yairi 2014).

### Anomaly Detection Using Replicator Neural Networks Trained on Examples of One Class

Dau et al. (2014) develops an anomaly detection model by training a three-layered AE network (with a single hidden layer). The number of nodes in the hidden layer is dependent on the data dimensionality but the authors mean that the optimum is to use the same number of hidden neurons as in the input and output layer.

### Safe Visual Navigation via Deep Learning and Novelty Detection

Richter & Roy (2017) proposes a method to detect novel examples by training an AE on non-novel examples in the training set using a reconstruction error as loss function. When presented with a novel input, the authors hypothesize that the reconstruction error will be large. By computing the empirical Cumulative Distribution Function (CDF) of the distribution of errors in the training dataset the $99^{\text{th}}$ percentile is chosen as threshold. The novelty detection method works well for structured data but, as the authors explain, it might not work for less structured datasets. Instead they propose, in future work, to use density estimation in the feature space learned by a large CNN.

### Safer Classification by Synthesis

Wang et al. (2017) lets VAEs or GANs build generative models for images from each class. These models receive a random input vector and transform these into images. When the model receives a test input, the algorithm searches across the generated class images for an image that is sufficiently similar. The best similarity score represents the classification label. While this method performs worse than a standard CNN at classification, it is better at detecting out-of-distribution samples. It can therefore be used as a novelty detection supervisor and improve the CNN coverage.

### Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery

Schlegl et al. (2017) proposes a Deep Convolutional Generative Adversarial Network (DCGAN) (Radford et al. 2015), AnoGAN, that learns a manifold of normal anatomical variability. Given a novel input, the latent space is searched to find a latent vector that makes the reconstruction as similar to the input as possible. Two losses are proposed for the search:

- a *Residual loss* that enforces visual similarity between the image generated by the latent vector and the input image,
- a *Discrimination loss* that enforces that the generated image from the latent input lies in the vicinity of the learned manifold of the model. This is done using feature matching.

The total loss is an affine combination of the two losses and the loss in the final iteration defines the anomaly score. The residual image can then be used to identify anomalous regions in the input. Schlegl et al. (2017) were able to detect different

anomalies in medical images and also show where they appear on examples not seen during training.

#### Anomaly Detection With Generative Adversarial Networks

Lucas Deecke (2018) proposes, similar to Schlegl et al. (2017), a DCGAN to perform anomaly detection. The main difference is that only the reconstruction part of the loss is considered and that the generator is allowed to change during the anomaly detection. After the algorithm is done, the generator weights are reset to their previous value. To account for the non-convexity of the latent space, eight different seeds are used for initial values for the latent vectors. Each of them is then optimized separately and the average reconstruction loss makes the anomaly score.

### 3.1.2 Domain-Based Detection

Domain-based methods are focused on creating a boundary around the training set. The methods hence describe the boundary, or the domain, and can infer on whether a new example is anomalous based on its position relative to the boundary.

#### Validating Neural Network-based Online Adaptive Systems: A Case Study

Liu et al. (2007) uses a Support Vector Data Description (SVDD) technique for real-time detection of novelty data fed to an adaptive Intelligent Flight Control System (IFCS) simulator. The novelty detector is fed the input data before it reaches the adaptive system and can therefore detect novelties before they reach the network. The SVDD-method tries to find a sphere with a minimal volume which contains all data items. Since IFCS contains high-dimensional data, the SVDD uses a kernel which maps the data to a Hilbert space, making it separable and less complex. The authors use the SVDD to create a posterior probability novelty measure which gives a good visualization of the degree of novelty in an input. This measure enables the tuning of sensitivity and specificity in an observable way. With good results, the algorithm manages to detect novelties in the IFCS failure signals.

#### High-Dimensional and Large-Scale Anomaly Detection using a Linear One-Class SVM with Deep Learning

Building a robust anomaly detector for use in high-dimensional datasets requires the help of a feature extractor, according to Erfani et al. (2016). The difficulties in high-dimensional data mentioned in the paper are:

- *exponential search space* as the input dimensions increase in size, the amount of potential feature spaces grows exponentially
- *data-snooping bias* with a high amount of possible feature sub-spaces, each input point has one or more sub-spaces where it appears as an anomaly
- *irrelevant features* not all feature sub-spaces are relevant for the model task and can hence be seen as noise in the data. This noise can hide the true anomalies from the relevant data.

By robust, the authors means ”an accurate model for data drawn from a wide range of probability distributions, and is not unduly affected by small departures from the trained model” (Erfani et al. 2016).

As feature extractor, Erfani et al. (2016) uses a Deep Belief Network (DBF) which is an unsupervised network trained to find generic underlying features in the data set. The DBF can be seen as a dimensionality reduction algorithm which generates a non-linear manifold of relevant features. The features are used to train a one-class, unsupervised SVM. Instead of using a complex kernel, the hybrid DBF-SVM model makes it possible for the SVM to use a more basic, linear kernel to detect anomalies.

### 3.1.3 Probabilistic-based Detection

Probabilistic-based methods are usually based on the approximation of the probability density function of the data. This is not easy in high-dimensional spaces (Pimentel et al. 2014). The method presented under this topic takes an alternative approach in an attempt to model the possibility that an input does not belong to the predefined classes in the training data.

#### Towards Open Set Deep Networks

Recognition DNNs today are trained on a closed set of examples which makes the network unable to make qualified predictions on samples outside of it. Bendale & Boult (2016) suggest recognition towards open sets; a recognition algorithm used in the real world, available to reject unknown and unseen samples. Open is here for the set covering all possible inputs outside of the training domain, defined by (Scheirer et al. 2013), as opposed to the closed set within the training domain. In contrast to constructing a supervisor in front of a neural network which rejects novelties, Bendale & Boult (2016) suggest extending the SoftMax layer with an OpenMax layer in the end of the neural network. The OpenMax layer includes the open set by adding an open class to the SoftMax layer and helps the network to estimate whether an input belongs to an unknown class or not. By suggesting that activations in the penultimate layer are not per-class score estimates but rather distributions of which classes are related, Bendale & Boult (2016) use a multi-class meta-recognition algorithm to say whether the input image is unknown or not. For example, cats and tigers are related visually and hence produce similar activations. The algorithm succeeds in rejecting many unknown open set and fooling images as well as some adversarial images. Aside from developing an algorithm, Bendale & Boult (2016) provide several theories and ideas worth investigating.

## 3.2 Adversarial Example Detection

Algorithms, or existing methods, encountered in adversarial detection are presented in this section. The methods are divided into four different classes: one focusing on theoretical work and insights; three depending on the approach taken: binary adversarial classifiers; layer activation statistics; confidence estimation.

### 3.2.1 Insights

The hypothesis that adversarial examples are detectable is a hard one since the definition of adversarial examples is minimal changes to an example belonging to the training set. Hence the resulting adversarial example should, in principle, also belong to the training set. Below, theoretical and empirical insights on the topic are presented.

#### On the (Statistical) Detection of Adversarial Examples

Grosse et al. (2017) takes an unusual approach to detect adversarial examples; they hypothesize that adversarial examples are not drawn from the same distribution as the original data. Hence, it should be sufficient to detect them with a statistical test. Using maximum mean discrepancy and energy distance, the authors show that they can distinguish adversarial examples from normal data. By applying the statistical test, the authors are able to detect samples in 50 adversarial examples. Based upon the statistical difference, the authors also try to augment the model with an additional class for adversarials. The model now detects adversarial examples as outliers or significantly increase the cost of crafting an adversarial example.

#### Classification Regions of Deep Neural Networks

Fawzi et al. (2017) seek to analyze the geometric properties of deep neural network classifiers in the input space, especially the topology of classification regions, as well as the decision boundary. The authors find that the decision boundary is flat in most directions in natural images, but some are curved. These curved directions are shared between different data points and show that networks are sensitive to perturbations in these directions. Furthermore it is empirically shown that the classification regions are connected.

### 3.2.2 Binary Adversarial Classifiers

In this section, work using binary classifiers, trained using both natural and adversarial inputs, is presented. The methods have a great advantage having already seen adversarial data.

#### ReabsNet: Detecting and Revising Adversarial Examples

To address the issue of misclassified adversarial examples even when they are very similar to natural samples, Chen et al. (2017) propose the resorption network (ReabsNet). It consists of two components:

- *The guardian network* - The guardian network is trained to detect adversarial examples. It is trained on both natural and adversarial examples with a binary output
- *Modifier* - The modifier leverages the high similarity between the adversarial and a potentially natural image. By revising the adversarial image iteratively, using the output from the guardian, the modifier is able to remove the small adversarial perturbations.

ReabsNet is able to reject adversarial examples even when allowing larger perturbations than initially trained with. As the authors mention, the model was never tested for adversarial examples crafted to fool both the classification-network and ReabsNet.

**Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks**

Xu et al. (2017) attempt to find adversarial examples by making predictions in parallel to the classifier model. These parallel predictions are first transformed (squeezed) in an algorithm which tries to capture certain features in the input before they are sent to the network model. The method is based on the results of Hinton et al. (2012*b*) which showed that sub-models in the network commonly disagree on predictions when the inputs are adversarial examples. The idea is therefore to squeeze a specific feature, for example by reducing bit depth or smoothing out of the image, and then let the network predict the altered image. If the distance between the unaltered input prediction and any of the squeezed ones is larger than a certain threshold value, the input will be considered to be an adversarial.

### 3.2.3 Layer Activation Statistics

Below are three methods using the inner activations of layers in a network, as the input image is forward-propagated through the network, to detect adversarial examples. These approaches utilize statistical methods to model the layer activations and hence gather a representation of the learned space.

**Adversarial Examples Detection in Deep Networks with Convolutional Filter Statistics**

Li & Li (2016) start by investigating whether adversarial examples come from the same distribution as the normal examples. By using spectral analysis through PCA on the $14^{\text{th}}$ layer of a Visual Geometry Group (VGG) network, developed by Simonyan & Zisserman (2014), the authors notice that there are no significant differences in the PCA projection on the first eigenvectors. However, the adversarial examples seem to reside in the center of the projection while the normal examples occupy a larger space. Moving to the tail of the PCA projection, many adversarial examples have extremely large values relative to the normal examples. The authors observe that adversarial examples actually have lower predictions but appear more confident after the Softmax function. They conclude that extreme values and standard deviations are evident features but require many samples. They propose turning a single image into a distribution and extract statistics. The k-channel image (can be an output from a convolutional layer) pixels are considered to be a random k-dimensional vector drawn from some distribution. From each k-dimensional feature the following statistics are collected:

- normalized PCA coefficients
- minimal and maximal values
- $25^{\text{th}}$, $50^{\text{th}}$ and $75^{\text{th}}$ percentile values.

Using a cascade classifier (several classifiers in line) such that the $i^{\text{th}}$ classifier takes the statistics from the $i^{\text{th}}$ layer as input, normal examples are eliminated using a threshold, while the examples which still might be perturbed or anomalous are passed on to the $(i+1)^{\text{th}}$ classifier. The classifier is trained using a dataset and 2000 Limited-Memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS), developed by Liu & Nocedal (1989), adversarial examples. To test the out-of-sample generalization capability of the network, Evolutionary Algorithm (EA)-adversarial was included and detected with high accuracy using only the first layers. The hypothesis is that because EA-adversarial looks quite different from natural examples they can be detected at an early stage. Likewise, the 2000 L-BFGS adversarial examples were detected with high accuracy.

#### On Detecting Adversarial Perturbations

Metzen et al. (2017) propose to train a binary detector network which uses intermediate activations in the main classifier network to output the probability of the current input being adversarial. The attachment of the detector is tried on different positions of intermediate activations using ResNet, developed by He et al. (2015), and VGG-16, developed by Simonyan & Zisserman (2014). For each image in the training set, the detector is trained on the same set as well as one adversarial example per training instance. The detector generalizes well to larger perturbations when trained on small perturbations, but not the other way around. It is shown that training on 'static adversarials' (adversarial examples created in advance of training the network), the detector can be fooled by attacking the entire system again with new adversarial examples. In response to the observation, the authors introduce dynamic adversarial training. The idea is to compute adversarial examples during training rather than before. Then it achieves a minmax-game, where the adversarial attacker repeatedly tries to fool the detector as the detector tries not to be fooled.

#### SafetyNet: Detecting and Rejecting Adversarial Examples Robustly

Lu et al. (2017) pose a hypothesis that adversarial examples work by creating different patterns of activations in the late stage of a network, relative to the patterns of naturally occurring examples. The authors propose Safetynet: a detector that looks at the internal states of the last layers in a neural network to detect anomalies that represent adversarial examples. To make the detector robust, they use thresholds for the activations which produce binary code. The detector, a Radial Basis Function-Support Vector Machine (RBF-SVM) compares the code produced at test time with a collection of examples. The detector is able to reject adversarial attacks not previously trained on and, more specifically, outperforms the detector proposed by Metzen et al. (2017) with respect to generalization.

### 3.2.4 Confidence Estimation

Confidence estimation concerns the fact that neural networks, very counter-intuitively, confidently classify adversarial examples as a class different from what a human would.

#### Confidence Estimation in Deep Neural Networks via Density Modelling

Subramanya et al. (2017) take an alternative approach to the problems of uncertainty in deep neural networks. They hypothesize that high-confidence misclassifcations do not expose a flaw with the networks but rather that we need a better way to estimate confidence. The authors show that uncertainty estimates from SoftMax predictions contain pathologies, meaning it is sensitive to the scale of the input. The idea is to take a density modelling approach to the problem. By computing the conditional probability of the activations in the final layer, they use Bayes' Rule to compute the probability of a given label. They do, however, assume that the final layer activation distribution is approximately equal to the distribution of inputs. The authors use a multivariate Gaussian with diagonal covariance for the density modelling and achieve confidences superior to SoftMax. It is mentioned that more sophisticated density models could be used and would likely perform better.

#### Detecting Adversarial Samples from Artifacts

Feinman et al. (2017) compute two features to detect adversarial examples: kernel density estimates and bayesian uncertainty estimates. Density estimates are meant to detect points in the input that lie far from the data manifold. They function by looking at activations in the feature space of the last hidden layer when fed by the training data set. Bayesian uncertainty estimates are meant to detect points lying in low-confidence regions of the input space. The estimates are supposed to function when the density estimates are uncertain but can only work for dropout neural networks (networks with dropout enabled). The two methods are then combined in a logistic regression classifier with the uncertainty and density estimates as inputs.

## 3.3 Datasets

During the search for plausible algorithms for input verification, a collection of used datasets have been found in the literature review and used in the thesis.

### 3.3.1 MNIST

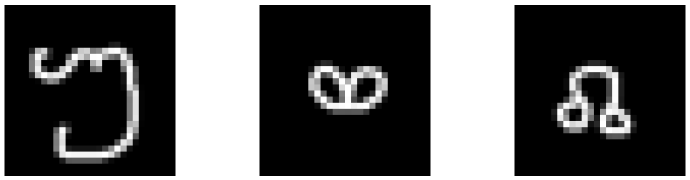
The first tutorial assigned to developers by the Tensorflow website is "MNIST For ML Beginners" (*Getting Started | TensorFlow* 2018). MNIST is for machine learning what 'Smoke on the water' is for guitar lessons. The dataset consists of 70000 (60000 for training and 10000 for testing) handwritten images of $28 \times 28$ pixels, see Figure 3.1. It was created by LeCun et al. (1998), who modified the National Institute of Standards and Technology (NIST) Special database 19, developed by Grother (1995), hence MNIST. It is commonly used in machine learning as a first dataset to implement techniques and pattern-recognition on and has been cited more than 11800 times according to Google Scholar search.

(a) (b) (c)

**Figure 3.1:** Sample images from the MNIST dataset.

### 3.3.2 Omniglot

Lake et al. (2015) created a dataset consisting of 1623 handwritten characters, see in Figure 3.2, from 50 different alphabets. The dataset was developed for one-shot learning which essentially is an attempt to train machine learning algorithms to find patterns and information about categories by only feeding them a few training images. This is in contrast to regular algorithms that often demands several thousand images for training.



(a) (b) (c)

**Figure 3.2:** Sample images from the Omniglot dataset.

### 3.3.3 CIFAR-10 & CIFAR-100

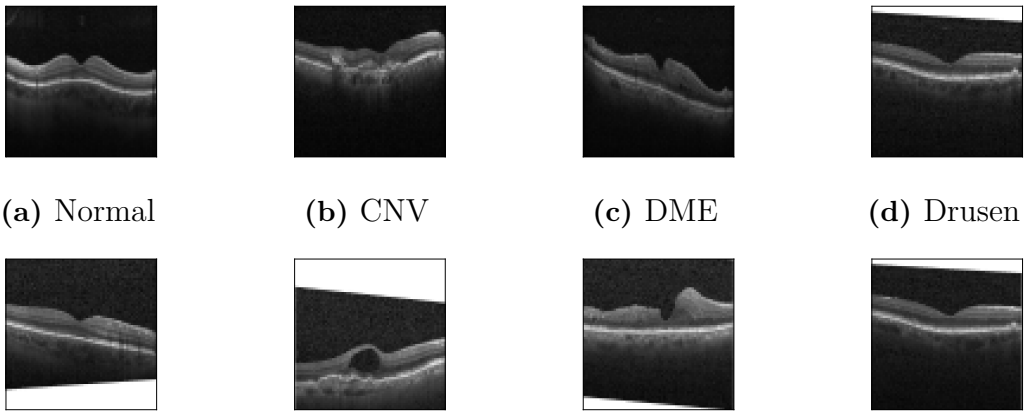
Krizhevsky & Hinton (2009) created the Canadian Institute For Advanced Research (CIFAR)-10 dataset with the help of groups from Massachusetts Institute of Technology and New York University. The images originate from a collection of 80 million colour images from the web. The dataset consists of ten classes with one thousand hand-labeled images each. Images in this dataset are downsized to $32 \times 32 \times 3$ pixels in size which facilitates the use for developing. The included classes are: airplane, car, bird, frog, cat, deer, horse, ship, truck and dog. In Figure 3.3, three example images of a cat, dog, and car respectively are shown. CIFAR-10 is a popular dataset for benchmarking machine learning techniques. Complementing CIFAR-10, the team also created the CIFAR-100 dataset which is similar but with 100 classes and 600 images for each class.

**(a)** **(b)** **(c)**

**Figure 3.3:** Sample images from the CIFAR-10 dataset.

### 3.3.4 Retinal Optical Coherence Tomography

Retinal Optical Coherence Tomography (OCT) is an imaging technique used to capture high-resolution cross-sections of the retinas. Around 30 million of these scans are done every year which means that the interpretation of them takes up a significant amount of time (Swanson & Fujimoto 2017). The dataset (Paultimothymooney 2018) contains 84484 images in total, divided into 4 classes: Normal, Choroidal Neovascularization (CNV), Diabetic Macular Edema (DME) and Drusen. Of the images are: 26565 normal; 37455 in the CNV class; 11598 in the DME class; 8866 in the Drusen class. Below in Figure 3.4, samples of images from each class are displayed.



**(a)** Normal **(b)** CNV **(c)** DME **(d)** Drusen

**(e)** Normal **(f)** CNV **(g)** DME **(h)** Drusen

**Figure 3.4:** Sample images from the Retinal OCT dataset.

The dataset contains high variance and images of the same class vary in appearance. It is not obvious for a non-professional how to classify by eye the different conditions and a normal retina. Hence each image went through a grading system with multiple steps of people of increasing expertise for verification and correction of the labels (Kermany et al. 2018).

# 4
# Methods

In this chapter, a description of how the literature review presented in Chapter 3 was carried out. Following are a description of how supervisors and datasets are chosen for experiments. The experiments and the chosen metrics are then motivated. Finally, the networks used for classification and the process for developing a new supervisor is presented.

## 4.1 Literature Review

The first task is to gather previous attempts of supervising DNNs, via the different notations of novelty detection and adversarial example defenses. The found supervising methods are then analyzed to enable a selection of interesting and promising methods which are going to be evaluated and compared in the next step of the thesis process.

### 4.1.1 Finding Papers

Starting from a batch of papers gathered in the SMILE II project, more articles are found by using an associative search method. The method consists of finding papers cited in the starting batch of papers, and then do another iteration in the second batch of papers. To broaden the search, papers related to found papers are found by using Google Scholar's 'related articles' function.

## 4.2 Criteria for Methods to Implement

The analysis of found supervising methods is done to enable the comparison between the different methods. Apart from being relevant to the task, several criteria matter for the selection. As the project has certain limitations in computational power and time, the accessibility of a specific supervising method will be important. Accessibility in this sense, is code implementations on GitHub or other sources and will greatly reduce the time necessary for a re-implementation. If inaccessible, the method needs to be described good enough to be implemented within a reasonable time frame.

## 4.3 Motivation for Scenarios to Evaluate Upon

For the supervisor characteristics to be properly evaluated, datasets representative of different scenarios are needed. Apart from image contents, the datasets have properties in form of pixel amounts and may be in colour or not.

#### MNIST vs Omniglot

The first datasets implemented upon are MNIST and Omniglot. Since both datasets include small, gray-scale images, they are suitable for primary testing. A strong reason to use these datasets against each other is that they are both fairly similar, handwritten characters but still distinguishable by humans. The CNN will be trained to classify MNIST images and the supervisor's task is to reject all Omniglot images sent to the network.

#### CIFAR-10 vs CIFAR-100

The second experiment is the use of a CNN trained to classify CIFAR-10 images while four, non-overlapping, classes from CIFAR-100 is presented as novelties. The CIFAR-10 and CIFAR-100 datasets are a step towards real-life scenarios for the supervisors, for example autonomous driving. The images are RGB with a reasonable amount of pixels which makes it suitable for a project lacking superior computational capacity.

#### Medical Retina Conditions vs Novelty Condition

The third dataset to implement upon is the medical OCT dataset containing images of retinas under normal conditions as well as under the condition of three pathologies: CNV, DME and Drusen. The images come in different sizes and are therefore resized to $64 \times 64 \times 3$ before being used. Since the images are quite low in features but still show a real-life problem, they are suitable for an implementation of supervisors. In the experiments on this dataset, three classes will be stated as known while the fourth is labeled as a novelty. In this way, a case is presented; the fourth pathology is used to act as a new, unknown pathology to the neural network. Here, normal, CNV and DME are labeled as known pathologies while drusen is labeled as a novelty.

## 4.4 Experimental Setup

In order to properly and fairly evaluate the implemented supervising methods, a clear process need to be stated. Below, in Figure 4.1, the evaluation process steps are shown in a flow chart. For each supervisor, the same training, validation, test and adversarial data are generated using, for reproducibility, a random seed number 42. When the supervisor has been run on the given data, related predictions, scores (or probabilities depending on the supervisor) and supervisor-specific hyper-parameters are saved. From the predictions and scores the performance metrics, which are motivated and explained below in Section 4.4.1, are computed.

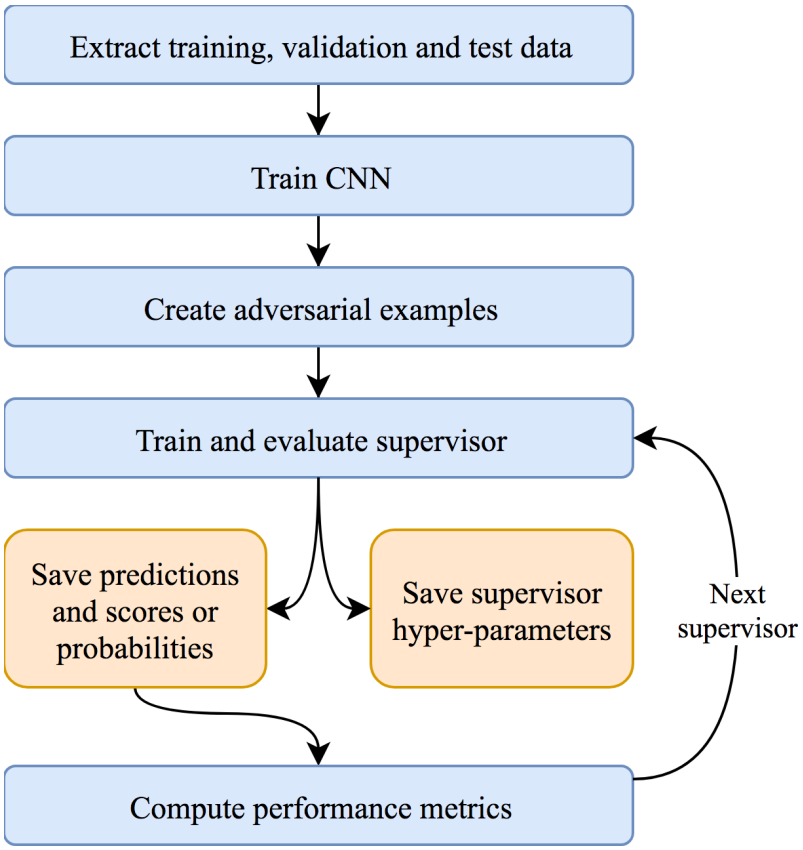



**Figure 4.1:** Process model for evaluation of supervisors.

### 4.4.1 Performance Metrics

To be able to compare the several supervisors' performances in the three scenarios, suitable metrics need to be used. The metrics used in the experiments are explained and motivated in this section.

#### AUC

The Area Under Curve (AUC), is as the name reveals the area under a curve. In the context of this thesis the curve in question is always the Receiver Operating Characteristics (ROC) curve. Consider a binary classification problem and classifier that generates a score through the score-function $f(\cdot)$ for each example. The classifier generates a classification for a new example, $x$ as

$$\text{Classification} = \begin{cases} 1, & \text{if } f(x) > \tau, \\ 0, & \text{if } f(x) \leq \tau. \end{cases} \tag{4.1}$$

Here $\tau$ is the classifier threshold. The ROC curve is then calculated by moving the threshold. As seen in Figure 4.2 the false positive rate (fraction of negative examples classified as positive) is on the abscissa and the true positive rate (fraction of positive examples correctly classified) is on the ordinate. In this case the threshold in the lower left at (0,0) is very high, meaning all examples are classified as negatives. As

the threshold is reduced new false positive rates and true positive rates (coordinates) are acquired which forms the ROC curve. At the upper right at (1,1) the threshold is very low, meaning that all examples are classified as positives. Note that all ROC curves start at (0,0), ends at (1,1) and are monotonically increasing. The ROC is hence an illustration of how well the scores are separated for the different classes. The line in Figure 4.2 is called the ROC baseline curve, which corresponds to a classifier randomly classifying examples with equal probabilities. To avoid confusion with the Baseline supervisor the ROC baseline curve will be denoted as the ROC 'straight line'. A curve consistently lying above the straight line would be a better classifier than the random classifier whereas one lying below would be worse. The ideal classifier would have a ROC curve that maximizes its distance from the abscissa at all coordinates.

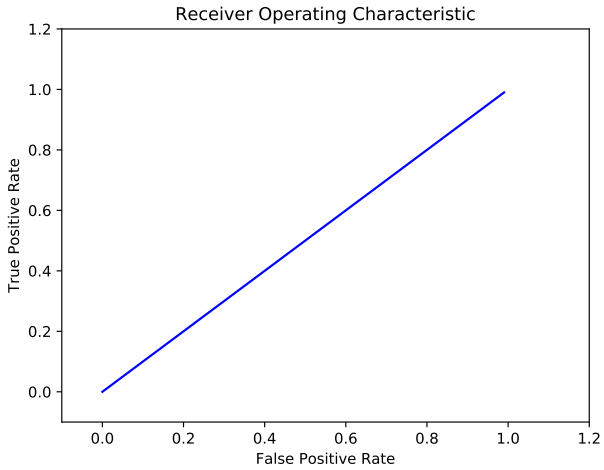



**Figure 4.2:** The ROC baseline representing using a random classifier.

The area under the ROC curve (AUC) has several possible interpretations but is simply a performance score of the classifier. The best AUC score a classifier can achieve is one and the worst is zero. The ROC straight line in Figure 4.2 has an AUC score of 0.5.

### Precision

The precision of a classifier is defined as the number of true positives divided by the number of true positives plus false positives. In words it is the classifiers ability to not label negative examples as positives. In this thesis it is a measure of how many of the images classified as novelties, really are novel.

### Recall

Recall is the same as the true positive rate; the number of true positives divided by the number of true positives plus the false negatives. In words it is the classifiers ability to label all positive examples as positives. In this thesis it is the ratio of correctly classified novelties over all possible novelties.

### F1-Score

Since it is possible to have varying results on precision and recall, the metrics are complementary to each other. For the classifier to perform well, both metrics need to be satisfactory. The F1-score is the harmonic mean of precision and recall, ranging from 0 (worst score) to 1 (best score). For a balanced classifier, there should exist a balance between precision and recall.

## 4.5 Neural Network Architectures

In this section, the applied CNN classifiers' architectures are presented.

### 4.5.1 MNIST Neural Network Architecture

In order to enable the evaluation of the supervising methods, a reasonably well-performing CNN is needed. Simultaneously, a smaller network is preferable to reduce run times. The resulting network being used is seen in Figure 4.3, consisting of four layers: two convolutional layers and two fully connected layer (dense) with dropout. The boxes represent the feature maps and the bold text beneath the boxes shows the resulting feature map sizes.

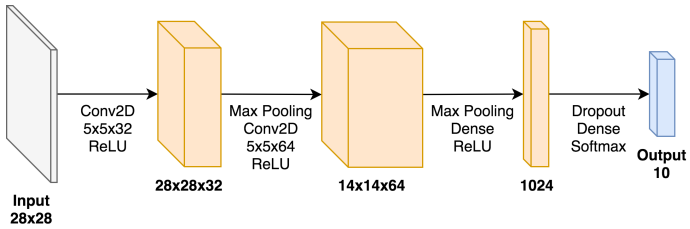



**Figure 4.3:** CNN architecture for MNIST classification.

### 4.5.2 CIFAR-10 Neural Network Architecture

For the experiments on the CIFAR datasets, a larger CNN is needed due to the larger varieties in the dataset. In Figure 4.4 the network can be seen. The boxes in the figure represent the resulting feature maps and their size is shown in bold text beneath. The CIFAR CNN is made of four convolutional layers and two fully connected (dense).

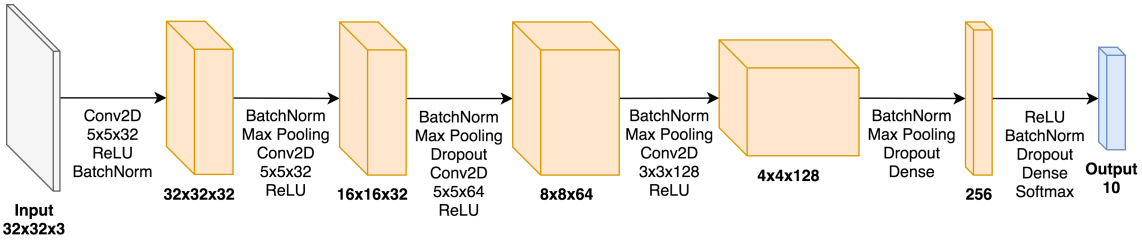



**Figure 4.4:** CNN architecture for CIFAR classification.

### 4.5.3 Retinal OCT Neural Network Architecture

The CNN used in the classification task for Retinal OCT dataset can be seen below in Figure 4.5. The architecture of the network is the same as for the MNIST experiments but with different input and output shapes. The network consists of two convolutional layers and two fully connected (dense) ones.

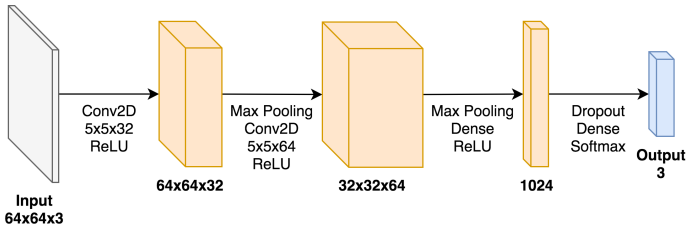



**Figure 4.5:** CNN architecture for Retinal OCT classification.

## 4.6 Creating a New Supervisor

Using the insights from the experiments, a new supervisor is implemented. The new supervising method is tested on the scenarios to evaluate its performance and compare it to the other supervisors. The supervisor developed will be referred to as the 'Thesis supervisor'. The results from the Thesis supervisor and the previous insights will be used to create recommendations for future work.

# 5
# Results

In this chapter, all results related to the thesis are presented. The selection process of algorithms and their implementation details are the first to be presented. Then, the results and insights from the experiments is presented, followed by the development of a new supervisor.

## 5.1 Selecting Existing Supervisors

In order to enable implementations of as many algorithms as possible, the algorithms from the literature review were investigated in forms of accessibility (documentation), relevance, performance and contrast from others. In some cases there are characteristics in several algorithms that may be merged because of their similarities. Since the implementations of the chosen algorithms are modified and their purpose have changed, they will henceforth be referred to as supervisors.

### 5.1.1 Motivation of Selected Supervisors

For convenience the supervisors will be named, and those names will be used to refer to the supervisors throughout the report. The names are shown in bold letters below.

#### NoveltyGAN

The overall impression from Schlegl et al. (2017) is that the algorithm presents an interesting approach to reconstruction-based anomaly detection utilizing both the discriminator and generator. The fact that the algorithm is evaluated on a medical dataset poses some challenges to adopt to the classification problem and datasets in this thesis. Lucas Deecke (2018) achieved impressive results and his algorithm is also more suitable for the experiments carried out in this thesis. Hence a combination of the two is considered.

#### Cascade

The algorithm proposed by Li & Li (2016) uses a relatively unique approach, leveraging information from several layers in the neural network. The information is coupled with a cascade classifier enabling different abnormalities in the activations to be detected at different stages. Hence, the algorithm offers the possibility to investigate the entire network and also show good results on out-of-sample inputs.

#### OpenMax

Bendale & Boult (2016) use statistics to investigate activations in the penultimate layer. A plus with the method is that it can be used to detect not only novelties, but also adversarials. The impression of the algorithm is that it is promising given the motivations of mean activation vectors by arguing that classes are related. The algorithm can be found, implemented by the authors, at the Github repository (Bendale 2016) which facilitates an implementation for this thesis.

#### Artifacts

Feinman et al. (2017) present an interesting approach combining layer statistics and network uncertainties with logistic regression. The promising results with good contrast between the training domain and adversarial images are confidently applicable on novelty detection and make the method interesting for the thesis. An implementation of the article code is available at the Github repository (Feinman 2018), but is modified to suit the thesis experiments.

#### BinaryNet

Chen et al. (2017) create a binary classifier in form of a guardian network, quite similar to the idea of a supervisor. However, the guardian network sends rejected samples to a modifier which perturbs the (adversarial) image and iterates the process until the guardian network accepts the image. The concept of using a neural network to detect novelties differs from the other algorithms implemented in the thesis and is therefore of interest.

### 5.1.2 Dismissed Approaches

To ensure that implementations of interesting algorithms are made thoroughly and as justifiably as possible, a robust documentation is demanded. Without it, the re-implementation is under risk of becoming too insecure and time-consuming. Even though several of the non-selected approaches would be interesting to implement, they are left for future work because they lack well-explained algorithms or are hard to re-implement.

## 5.2 Implementation of Selected Supervisors

In this section, the supervisors are explained in more detail. Note that they are not identical to the supervisors that they are based upon.

### 5.2.1 Baseline

To evaluate the performance of the algorithms, and have a performance to compare to, a simple baseline algorithm was implemented. The idea of this supervisor is to rely on the uncertainty of the CNN. The novelty score of an input is defined as

$$\text{Novelty Score} = 1 - \text{The maximum probability of the classes.} \tag{5.1}$$

The idea is that if a novelty is presented to the classifier, it should not be certain that it belongs to any of the known classes. For example, if a network outputs the probabilities $[0.3, 0.4, 0.3]$ in a three-class problem, the CNN is unsure on which class the input should belong to. Therefore the novelty score becomes $1 - 0.4 = 0.6$. On the other hand, if the network is very certain, it may output the probabilities $[0.1, 0.1, 0.8]$. The novelty score then becomes $1 - 0.8 = 0.2$.

### 5.2.2 NoveltyGAN

To identify novelties, a generative model in form of a DCGAN is constructed. This generative model is trained using normal data, that is, the same training data as the classifier is trained on. The DCGAN is trained according to the procedure explained in Section 2.1.5. Hence the generator, see Figure 5.1, learns how to map, in this example, samples from the standard normal distribution (latent space) $z \sim P(z) = N(0, 1)$, $z \in \mathbb{R}^{100}$. The generator constructs an image $G(z)$ with the objective to fool the corresponding discriminator, see Figure 5.2. Note that the images in the datasets are resized to $64 \times 64 \times \beta$, where $\beta$ is the number of channels in the images. Since the two networks are constantly trying to fool and not to be fooled by the other, it is not uncommon that the respective losses oscillate during training.

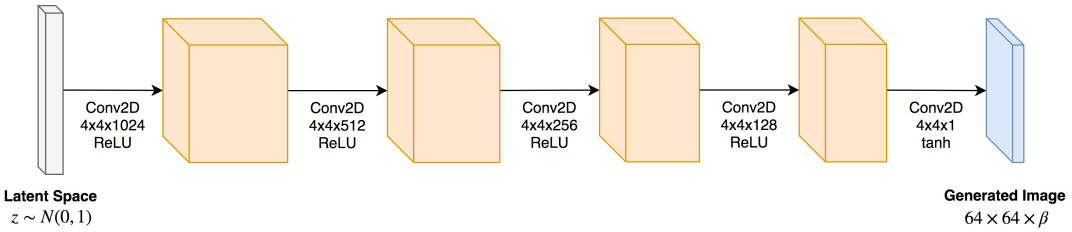



**Figure 5.1:** The GAN generator network architecture.

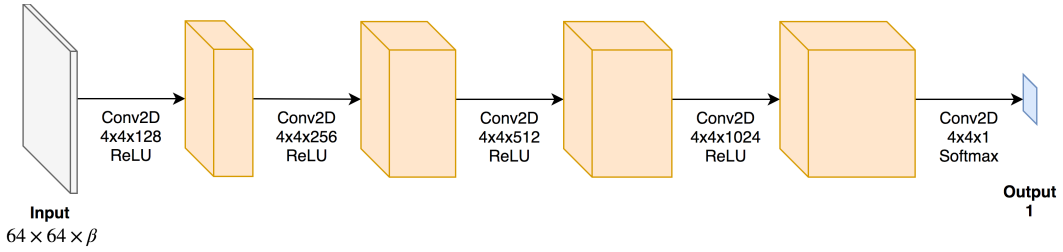



**Figure 5.2:** THe GAN discriminator network architecture.

When the generator and discriminator are done training, the generator and discriminator parameters are frozen and not allowed to change. Consider a new query image, $x$, and a latent vector, $w$. To decide if the input is a novelty or not, the latent vector is optimized to match the input according to a loss function consisting of two components:

- the residual loss $L_R(z_\gamma) = \sum |x - G(z_\gamma)|$ and
- the discrimination loss $L_D(z_\gamma) = \sum |f(x) - f(G(z_\gamma))|$.

Here, $f(\cdot)$ refers to the activations of some layer in the discriminator. For this algorithm the last layer logits are used. The total loss is then a convex combination of the two:

$$L(w) = (1 - \lambda)L_R(w) + \lambda L_D(w). \tag{5.2}$$

A novelty score given an input image $x$ is defined as

$$A(x) = (1 - \lambda)R(x) + \lambda D(x) \tag{5.3}$$

where the residual score $R(x)$ and discrimination score $D(x)$ are defined by the residual loss and discrimination loss, respectively. As Schlegl et al. (2017) recommend, $\lambda$ is set to 0.1 for the experiments. To address the non-convexity of the problem, $n_{seed}$ seeds or starting points are used to initiate the latent vector $w$ and each of them is then optimized with respect to the loss function in (5.2) (Lucas Deecke 2018). The novelty score is then given by the minimum loss/score corresponding to one of the $n_{seed}$ seeds.

### 5.2.3 Cascade

The neural network layers are of two different kinds: fully connected (dense) and convolutional. For each layer in the network, one PCA-transform is created using the activations produced by the training data. Assume the activations produced by a convolutional layer has the dimensions $W \times H \times \beta$, where $\beta$ is the number of feature maps. The activations is then treated as $W \times H$ different samples from $\beta$-dimensional distributions, yielding $\beta$ PCA components. A dense layer of dimension $n$ is considered to be $n$ samples from $n$ one-dimensional distributions yielding $n$ PCA components. The statistics extracted from each input and layer is the $L1$-norm of the normalized PCA coefficients, the maximal values and $25^{th}$, $50^{th}$ and $75^{th}$ percentiles. Hence, the statistics for a convolutional layer $s_{conv} \in \mathbb{R}^{5\beta}$ and for fully connected layers $s_{fc} \in \mathbb{R}^{n+4}$.

The Cascade supervisor consists of $n$ SVCs; one for each of the $n$ layers. Each classifier receives statistics from its designated layer and decides whether the corresponding input is a novelty or a normal. Given a set of normal training examples and adversarials, each classifier is trained on a balanced subset of the two classes. Then, the SVC classification threshold is set to only accept a predefined false positive rate (with normal being the positive class). At each stage in the cascade, the trained classifier predicts all normal images. The images classified as normal are removed from the set and not used for training the remaining classifiers. In addition to the predefined target false positive rate, a parameter that controls the fraction of normal inputs classified as a novelty exists. If the fraction exceeds the parameter, the training is restarted with a lower target false positive rate. An illustration of the Cascade supervisor is shown below in Figure 5.3.

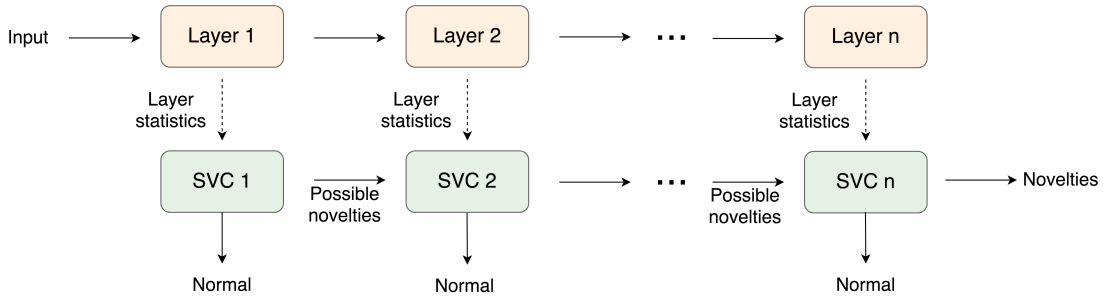



**Figure 5.3:** Illustration of the Cascade supervisor.

### 5.2.4 OpenMax

The OpenMax algorithm utilizes the Weibull distribution to fit activation vectors. In this implementation, as well as in the original by Bendale & Boult (2016), the libMR package by Scheirer, Rocha, Michaels & Boult (2011) is used to fit the data. The Weibull distribution is often used in failure rate analysis. For a further motivation of the Weibull distribution, see the work by (Scheirer, Rocha, Micheals & Boult 2011). The Weibull distribution is described as

$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}(\frac{x}{\lambda})^{k-1}e^{-(x/\lambda)^k}, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{5.4}$$

where $x$ is the quantity (in this case a distance), $\lambda$ the shape parameter and $k$ the scale parameter.

The OpenMax idea is to utilize that CNNs may give responses in the penultimate layer for classes that have similar features. These classes are considered to be related. For example, the top circle of a nine might show activations when the network is fed an eight.

Given a CNN predicting the class of 100 images, $x_i$, $i = 1, ..., 100$, with labels $j \in \{1, ..., 10\}$, there are 100 penultimate activation vectors of size $[10, 1]$. All correctly classified images' activation vectors are then collected in 10 matrices separated by class, $S_j$. For each class, $j$, a mean activation vector, $\mu_j$, is computed from samples in $S_j$. For each correctly classified image, the euclidean-cosine (eucos) distance from the class $j$ are calculated as

$$d_{eucos}(x_i|x_i \in S_j) = ||\mathbf{v}(x_i) - \mu_j|| + 1 - \frac{\mu_j \bullet \mathbf{v}(x_i)}{||\mu_j|| \cdot ||\mathbf{v}(x_i)||} \tag{5.5}$$

where $\mathbf{v}(\theta)$ is the penultimate layer activation vector given by the input image $\theta$. For the eucos distances, the $\eta$ largest distances are chosen to fit the Weibull distribution and create 10 Weibull models $\rho_j$ with parameters $\lambda_j, k_j$. $\eta$ is a hyper-parameter called tail length. The next step is to feed the network a test image.

For a new input $x$ with activation vector $\mathbf{v}(x)$, the values in $\mathbf{v}(x)$ are sorted from largest to smallest. The mapping between the indices of the original activation

vector and the sorted activation vector is denoted by $g(j)$. A weighted Weibull score (Weibull cumulative distribution function), $\omega$ is calculated with the distance between the input activation vector and the classes' mean activation vector, $\mu_j$, according to

$$\omega_j(x) = 1 - \frac{\alpha - g(j)}{\alpha} e^{-\left(\frac{||x-\mu_j||}{\lambda_j}\right)^{k_j}}. \tag{5.6}$$

where $\alpha = 10$ is the number of possible classes. A new, revised activation vector is then calculated with

$$\hat{v}(x) = \mathbf{v}(x) \circ \omega(x) \tag{5.7}$$

which leads to a new activation vector, weighted with the Weibull scores for all closed set classes. To enable the novelty detection, one has to introduce the possibility of open set classes or 'unknown unknowns'. The open set class activation is placed as an extra index in the existing new activation vector

$$\hat{v}_{\alpha+1}(x) = \sum_i \hat{v}_i(x)(1 - \omega_i(x)) \tag{5.8}$$

which enables a probability to be calculated. A normal Softmax calculation divides the exponent with the sum of all closed set class exponents. For OpenMax, the denominator is extended with the open set class which leads to

$$\hat{P}(j|x) = \frac{e^{\hat{\mathbf{v}}_j(x)}}{\sum_i e^{\hat{\mathbf{v}}_i(x)}},\ j = 1, ..., \alpha + 1, \tag{5.9}$$

where $\hat{P}(\alpha+1|x)$ is the open set class probability given input $x$. The predicted class is calculated by taking, just as in the Softmax, the index with the highest value. An input image is rejected if the open set class probability is over some threshold value.

### 5.2.5 Artifacts

Given the network with class labels $c_l$ where $l = 1, ..., N$, each class is said to have corresponding sub-manifolds in the last hidden layer of the network. These sub-manifolds can be estimated using Kernel Density Estimation on the feature space activations of the network layer. The feature space activations are gathered by running all training images through the network, and saving the activations of each node in the last hidden layer. For each class $c_l$, a kernel density model is fitted to the training images of the corresponding class, using a Gaussian kernel with a pre-tuned bandwidth. By sending a tuning set, $\hat{x}_i$ of correctly classified images through the network and gathering all hidden layer activations and the predicted class of each image, a log probability is calculated for the kernel corresponding to the predicted class. This log probability is also computed for an equal amount of novelty images.

To further robustify the supervising method, the previously computed log probabilities are used in a logistic regression model together with uncertainty scores. These scores are gathered by utilizing the dropout layer in the network, enabling the chosen dropout nodes to change between test runs, a Monte Carlo search is done to find

how much the logits change with different dropouts. The mean of the variance of these computed values is then saved for each image. The logistic regression model is then trained, using each sample's, both normal and novelty, normalized kernel density and uncertainty score.

When fed a new image, the method needs the normalized scores of the image's uncertainty and kernel density log probability to be able to use the logistic regression classifier model. Therefore, a scaling function is saved during training to enable the scaling of new inputs.

### 5.2.6 BinaryNet

While the concept of adjusting input images works for adversarial examples as explained in ReabsNet (Section 3.2.1), it is unnecessary to modify novelties. Hence, the perturbation module is dismissed in this implementation. A guardian network in form of a binary classifier is used, resembling the supervised CNN with the exception of a branched part of the network, as in Figure 5.4, inserted just before the fully connected (dense) layers and that the last fully connected layer has two nodes. The branch includes two fully connected layers with 200 nodes each, as in the mentioned guardian network. The network is trained on normal images and adversarial examples. In other terms, the network will function like a regular CNN, in this case predicting whether an input is novel or not.

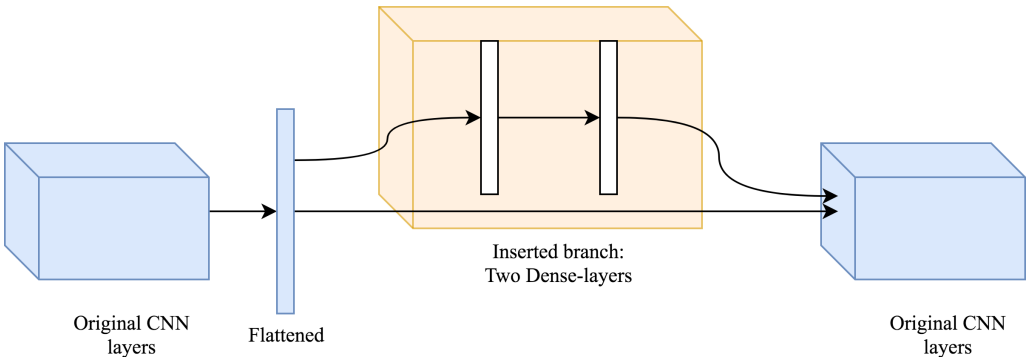



**Figure 5.4:** BinaryNet branch visualization.

## 5.3 Experiments on MNIST vs Omniglot

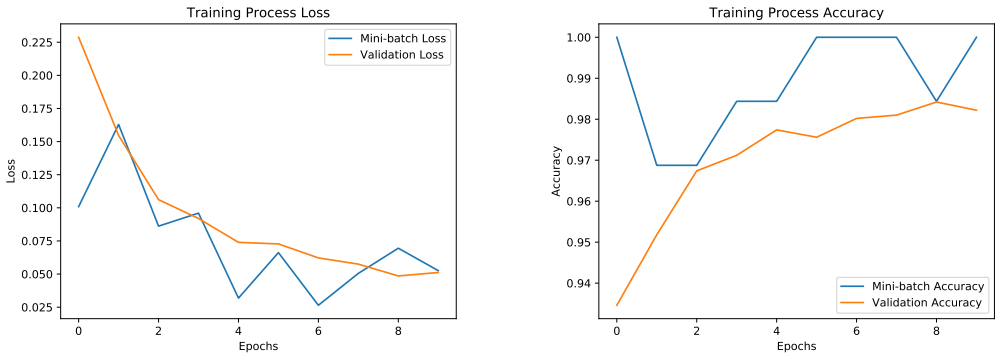
In the following section, results concerning the experiments on the MNIST and Omniglot dataset are presented. The normal images (MNIST) are denoted by the negative class and novelties (Omniglot) are denoted by the positive class. The ROC curves are very similar across all supervisors for this experiment (with the exception of the NoveltyGAN supervisor). They are nevertheless displayed to illustrate how and why different separations of novelties and normal examples yield similar curves. A summary of all supervisor metrics is provided in Section 5.3.8.

### Data Partition

The MNIST data was divided into a training, validation and test set. The training set contained 55000 examples, the validation set contained 5000 examples and the test set contained 20000 examples of which 10000 were Omniglot images (novelties). Since some of the supervisor algorithms require training with novelties, 15000 adversarials were generated, 5000 from each of the three algorithms: FSMA (Kurakin et al. 2016), JSMA (Papernot et al. 2015) and Deepfool (Moosavi-Dezfooli et al. 2015).

## 5.3.1 MNIST Neural Network

The network was trained for 10 epochs using a cross-entropy loss and a gradient descent optimizer with a learning rate of 0.01, a dropout rate of 0.4 and batch size of 64. As shown in Figure 5.5 both the loss and accuracy for the training and validation set improved during training.




**(a)** Loss. **(b)** Accuracy.

**Figure 5.5:** Loss and accuracy during training for the MNIST neural network.

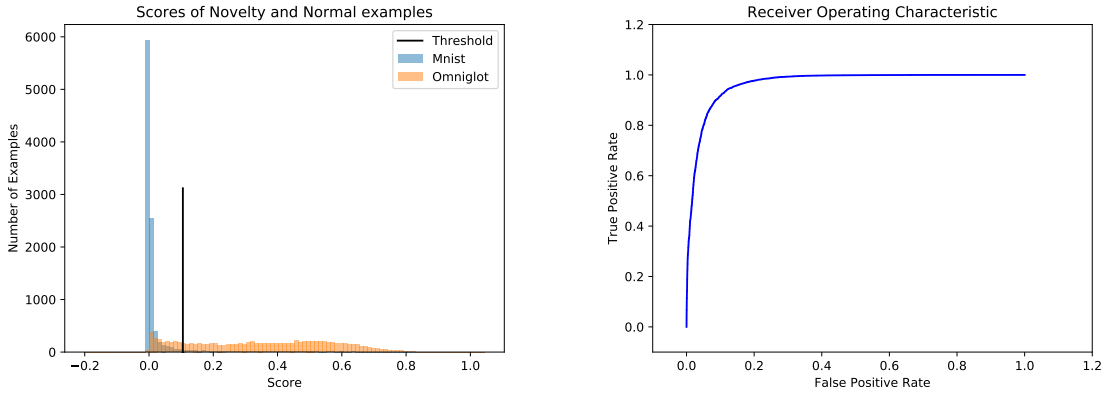
Evaluating the network on half the test set, only containing MNIST images and no novelties, yielded a test accuracy of 98%. Class-specific metrics are presented below in Table 5.1. It can be seen that the network performance is high across all different classes, only with minor deviations in metric scores.

**Table 5.1:** F1-score, Precision, Recall and Support (number of samples of class in test set) on the MNIST part of the test set for the network used in the MNIST vs Omniglot experiments.

| Class | F1-score | Precision | Recall | Support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 1.0 | 1001 |
| 1 | 0.98 | 0.98 | 0.98 | 1127 |
| 2 | 0.98 | 0.99 | 0.96 | 991 |
| 3 | 0.98 | 0.98 | 0.98 | 1032 |
| 4 | 0.99 | 0.98 | 0.99 | 980 |
| 5 | 0.98 | 0.98 | 0.99 | 863 |
| 6 | 0.99 | 0.99 | 0.99 | 1014 |
| 7 | 0.98 | 0.98 | 0.99 | 1070 |
| 8 | 0.97 | 0.97 | 0.98 | 944 |
| 9 | 0.98 | 0.99 | 0.97 | 978 |

### 5.3.2 Baseline

The threshold used to classify the test data was set to the $90^{th}$ percentile of the scores of the training set: 0.1. The separation between MNIST and Omniglot images in the test set in terms of novelty scores is illustrated in Figure 5.6a and the corresponding ROC curve in Figure 5.6b. The histogram shows a clear separation between Omniglot and MNIST, which is reflected in the sharply increasing ROC curve.




**(a)** Scores. **(b)** ROC curve.

**Figure 5.6:** Histogram of scores assigned to MNIST and Omniglot and the ROC curve for the Baseline supervisor on the test set in the MNIST vs Omniglot experiments.

To illustrate which classes are more commonly misclassified as novelties, the false positives relative to the class count of the 10 different classes are displayed below in Figure 5.7. Images of the number 2 followed by 9 and 8 are most commonly classified as novelties.

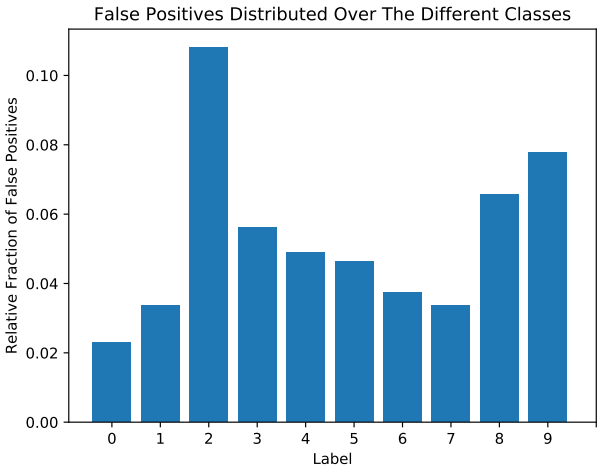



**Figure 5.7:** The false positives distributed over the 10 different classes relative to the class counts for the Baseline supervisor on the test set in the MNIST vs Omniglot experiments.

### 5.3.3 NoveltyGAN

The DCGAN was trained for 30 epochs using the loss described in Section 2.1.5 and the Adam optimizer (Kingma & Ba 2014) with a learning rate of 0.0002, beta1 of 0.5, and a batch size of 100. As seen in Figure 5.8 neither the discriminator nor generator losses are diverging from the other during training but the discriminator finishes with a lower loss.

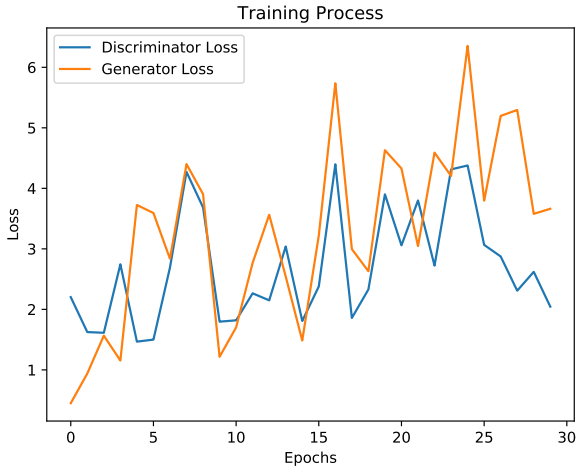
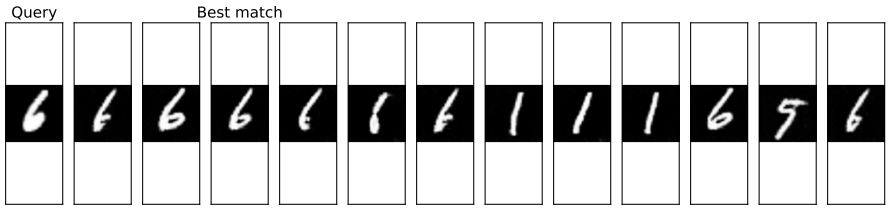


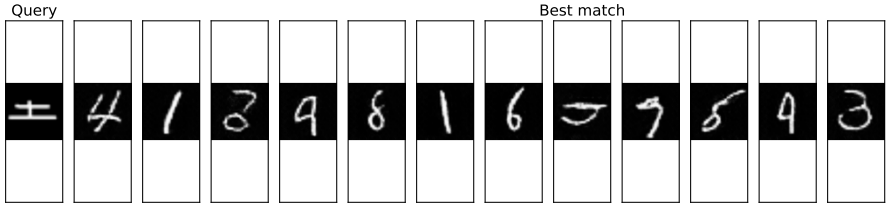

**Figure 5.8:** NoveltyGAN Discriminator and Generator loss during training in the MNIST vs Omniglot experiments.

After the training, the DCGAN was frozen and the only variable allowed to change was the random latent vector. Each query image from the test set was regenerated by changing the latent vector using the loss described in section 5.2.2. The Adam optimizer was used with a learning rate of 0.25 and beta1 of 0.5 for 7 iterations.

For each query, 12 starting points for the latent vector were tried and optimized separately. The final novelty score assigned to the query is the lowest score generated by the 12 latent vectors. A sample of how the regenerated images turned out is displayed below in Figure 5.9. The best match in terms of score is denoted with a title above the corresponding image. The supervisor manages to reconstruct the MNIST image accurately but not the Omniglot image as well.



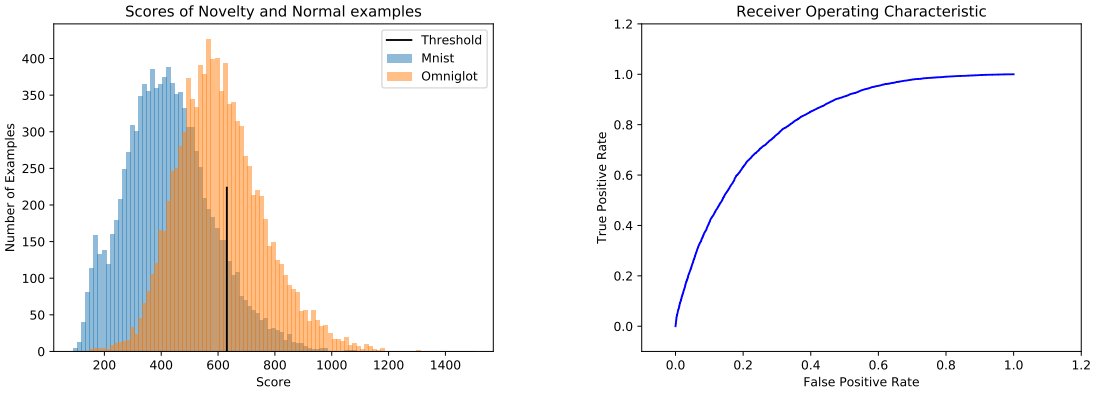

**(a)** MNIST query




**(b)** Omniglot query

**Figure 5.9:** Reconstructed images during the testing of the NoveltyGAN supervisor in the MNIST vs Omniglot experiments.

The threshold used to classify the test data was set to the $90^{th}$ percentile of the scores of the training set: 631. The separation between MNIST and Omniglot images in the test set in terms of novelty scores is illustrated by Figure 5.10a and the corresponding ROC curve in Figure 5.10b. The histogram shows harder separation between Omniglot and MNIST, which is reflected in the less steep ROC curve compared to curves achieved by the other supervisors.




**(a)** Scores. **(b)** ROC curve.

**Figure 5.10:** Histogram of scores assigned to MNIST and Omniglot and the ROC curve for the NoveltyGAN supervisor on the test set in the MNIST vs Omniglot experiments.

The false positives relative to the class count of the 10 different classes are displayed below in Figure 5.11. Images of the number 2 followed by 8 and 3 are most commonly classified as novelties.

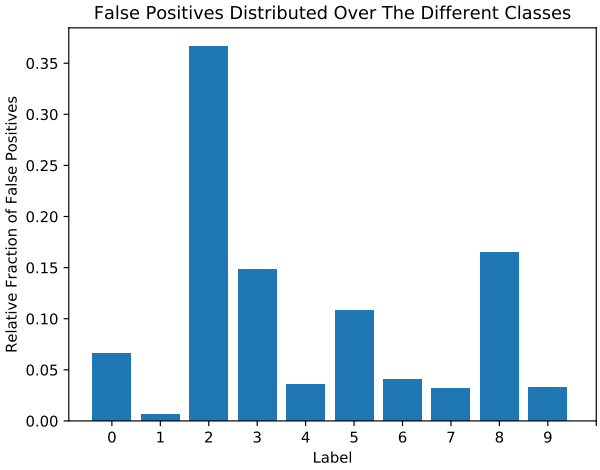
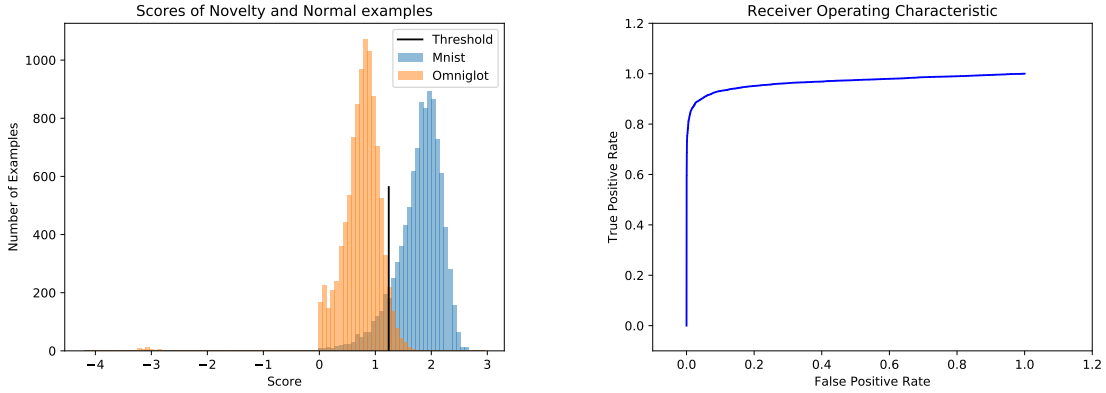



**Figure 5.11:** The false positives distributed over the 10 different classes relative to the class counts for the NoveltyGAN supervisor on the test set in the MNIST vs Omniglot experiments.

### 5.3.4 Cascade

Each stage of the cascade was trained on a subsample of all adversarials (abnormalities) and a corresponding amount of MNIST examples (normal). The algorithm target was that a maximum of 0.5% of the normal examples were to be classified as novelties. This resulted in a convergence at a false positive rate of 20% (with normal examples as the positive class). In the first stage of the classifier 99.9% of

the normal examples were detected. The threshold was set to the $10^{th}$ percentile of the scores on the training set: 1.24. Note that the classification rule with respect to the threshold is reversed here relative to the other supervisors since the decision function of the supervisor assigns high scores to normal examples and low to abnormal. The separation between Omniglot and MNIST images in the test set is illustrated by Figure 5.12a and the corresponding ROC curve in Figure 5.12b.




**(a)** Scores. **(b)** ROC curve.

**Figure 5.12:** Histogram of scores assigned to MNIST and Omniglot and the ROC curve for the Cascade supervisor on the test set in the MNIST vs Omniglot experiments.

The false positives relative to the class count of the 10 different classes are displayed below in Figure 5.13. The distribution between the classes are relatively uniform compared to other supervisors.

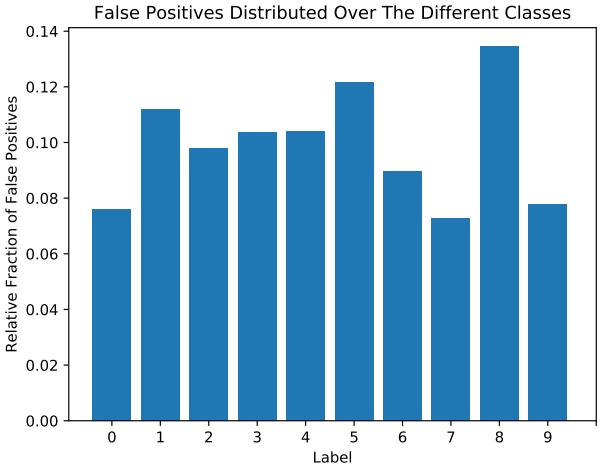
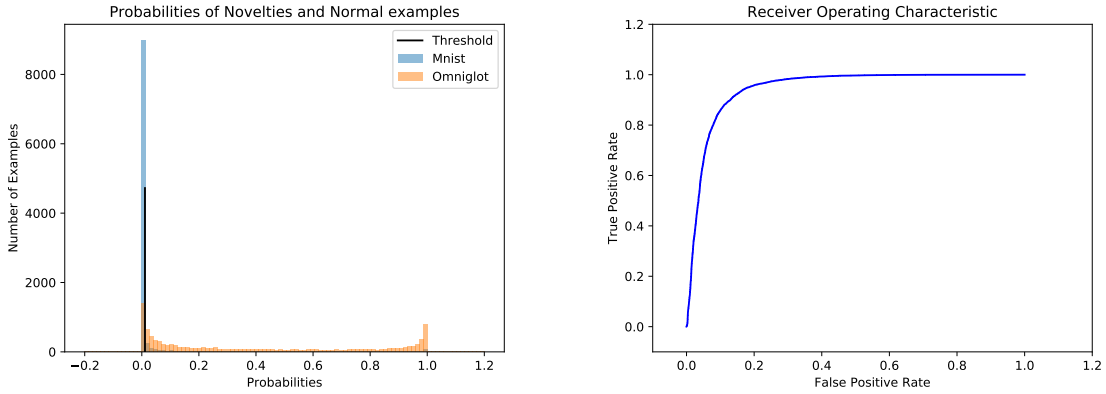



**Figure 5.13:** The false positives distributed over the 10 different classes relative to the class counts for the Cascade supervisor on the test set in the MNIST vs Omniglot experiments.

### 5.3.5 OpenMax

The OpenMax supervisor was fitted using the training data. The hyper-parameter tail length was tuned using the set of adversarial images and the training set with the AUC as a best-fit score, resulting in a tail length of 99. The classification threshold for the probability of a novelty was set using the $90^{th}$ percentile of the probabilities of the training set: 0.01. The separation between Omniglot and MNIST in the test set is illustrated by Figure 5.14a and the corresponding ROC curve in Figure 5.14b. The histogram shows a clear separation between Omniglot and MNIST, which is reflected in the sharply increasing ROC curve.




**(a)** Scores. **(b)** ROC curve.

**Figure 5.14:** Histogram of scores assigned to MNIST and Omniglot and the ROC curve for the OpenMax supervisor on the test set in the MNIST vs Omniglot experiments.

The false positives relative to the class count of the 10 different classes are displayed below in Figure 5.15. Images of the number 6 followed by 2 and 9 are most commonly classified as novelties, relative to the number examples of the class in the test set.

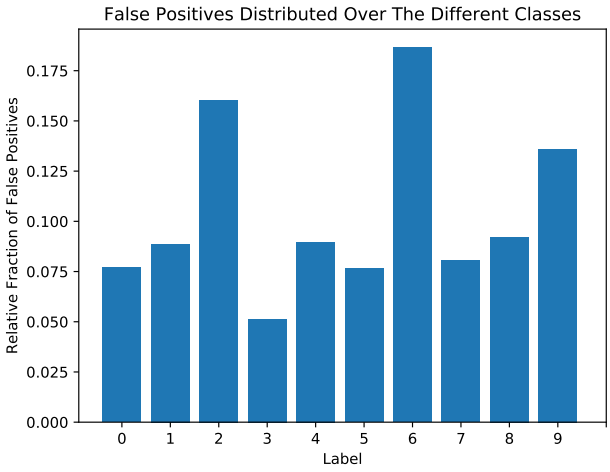
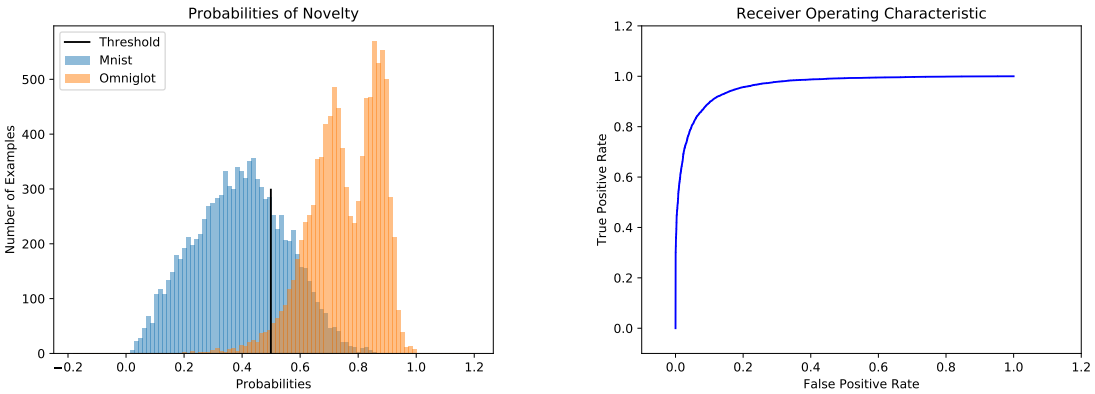



**Figure 5.15:** The false positives distributed over the 10 different classes relative to the class counts for the OpenMax supervisor on the test set in the MNIST vs Omniglot experiments.

### 5.3.6 Artifacts

The Artifacts supervisor was trained using a subset of the training set and adversarial images. The remainder of the training set were used for kernel density estimates. The hyper-parameter bandwidth was tuned using a subset of adversarial images and training set with the AUC as a best-fit score, resulting in a bandwidth of 1.1. The classification threshold for the probability of a novelty was set using the $90^{th}$ percentile of the probabilities of the training set: 0.5. The separation between Omniglot and MNIST images in the test set is illustrated by Figure 5.16a and the corresponding ROC curve in Figure 5.16b. The histogram shows a clear separation between Omniglot and MNIST, which is reflected in the sharply increasing ROC curve. The probabilities assigned to Omniglot are centered around two peaks.




**(a)** Scores. **(b)** ROC curve.

**Figure 5.16:** Histogram of scores assigned to MNIST and Omniglot and the ROC curve for the Artifacts supervisor on the test set in the MNIST vs Omniglot experiments.

The false positives relative to the class count of the 10 different classes are displayed below in Figure 5.17. Images of the number 8 followed by 9 are more often classified as novelties.

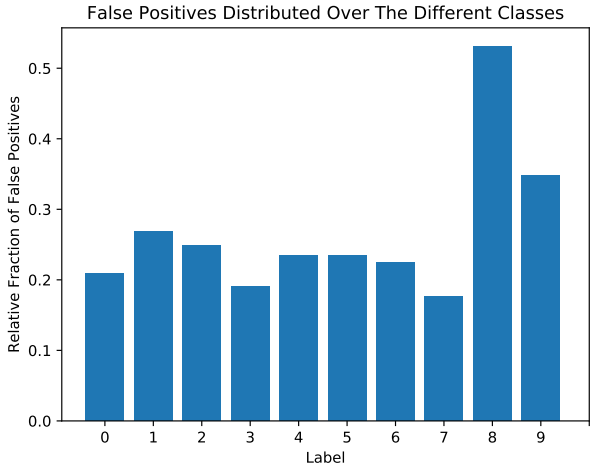
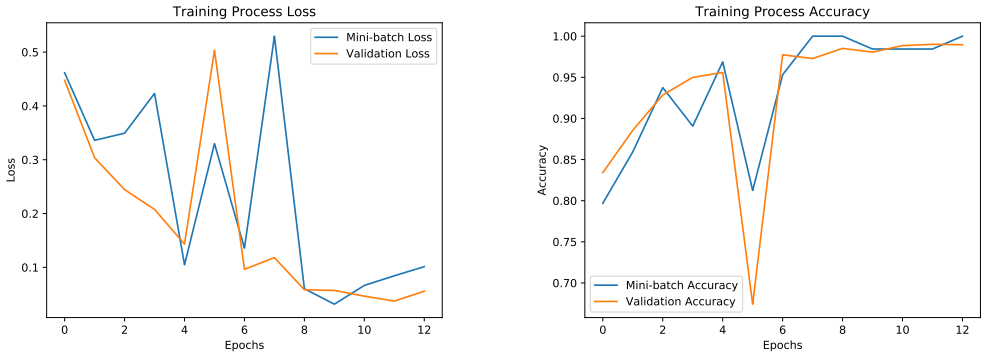



**Figure 5.17:** The false positives distributed over the 10 different classes relative to the class counts for the Artifacts supervisor on the test set in the MNIST vs Omniglot experiments.

### 5.3.7 BinaryNet

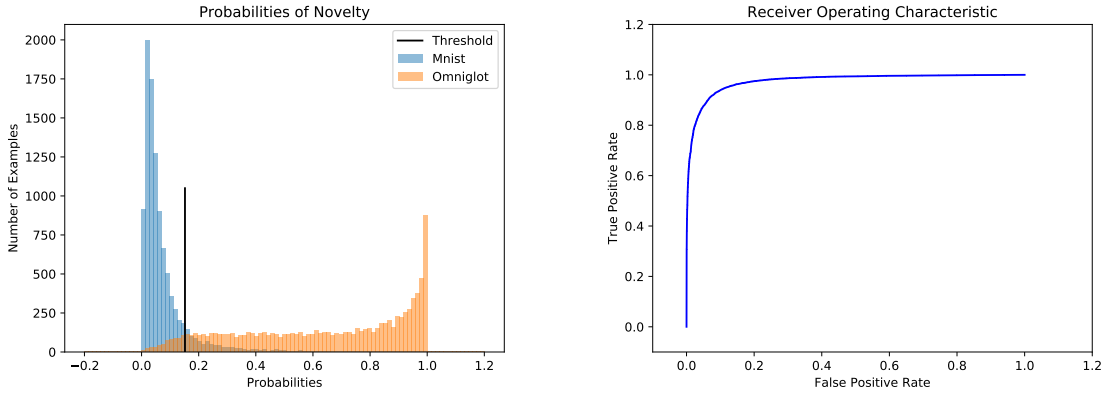
Using adversarial images and the training set the BinaryNet supervisor was trained for 13 epochs using a cross-entropy loss and a gradient descent optimizer with a learning rate of 0.01, a dropout rate of 0.5 and a batch size of 64. The validation set consisted of 20% of the data. As shown in Figure 5.18 both the loss and accuracy for the training and validation set improved during training, but with some volatility.




**(a)** Loss. **(b)** Accuracy.

**Figure 5.18:** Accuracy and loss during training of the BinaryNet supervisor in the MNIST vs Omniglot experiments.

The threshold on the novelty probability produced by the binary network was set to the $90^{th}$ percentile of the probabilities yielded by the training set: 0.15. The separation between Omniglot and MNIST images in the test set is illustrated by Figure 5.19a and the corresponding ROC curve in Figure 5.19b. The histogram shows a separation between Omniglot and MNIST, which is reflected in the sharply increasing ROC curve.




**(a)** Scores. **(b)** ROC curve.

**Figure 5.19:** Histogram of scores assigned to MNIST and Omniglot and the ROC curve for the BinaryNet supervisor on the test set in the MNIST vs Omniglot experiments.

The false positives relative to the class count of the 10 different classes are displayed below in Figure 5.20. Images of the number 8 followed by 3 and 9 are most commonly classified as novelties.

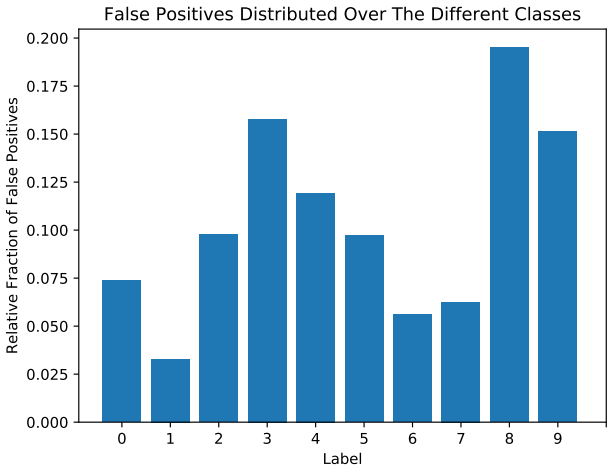



**Figure 5.20:** The false positives distributed over the 10 different classes relative to the class counts for the BinaryNet supervisor on the test set in the MNIST vs Omniglot experiments.

### 5.3.8 Comparison of Metrics

The performance metrics for all supervisors are displayed below in Table 5.2. The AUC is comparable for all different supervisors with the exception of NoveltyGAN, meaning the supervisors are able to separate MNIST from Omniglot. How they do it varies as can be seen in the corresponding separation histograms shown earlier in the section. The Baseline supervisor, which achieved the highest precision, is able to correctly classify Omniglot as novelties while also minimizing the number of MNIST examples classified as novelties. This is in contrast to the Artifacts supervisor which achieves the lowest score. The recall score is on the other hand highest for the Artifacts supervisor which means it is able to classify a large portion of the Omniglot examples as novelties. The NoveltyGAN supervisor has a significantly lower recall score. The F1-score is the weighted average of the two aforementioned metrics, taking both false positives and false negatives into account and is highest for the Cascade supervisor. The highest accuracy in classifying novelties is also achieved by the Cascade supervisor. The rest of the supervisors with the exception of the NoveltyGAN supervisor, achieves around 90% accuracy.

**Table 5.2:** Performance metrics for the six supervisors on the test set in the MNIST vs Omniglot experiments.

| Supervisor | AUC | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|---|
| Baseline | 0.97 | 0.94 | 0.81 | 0.87 | 0.88 |
| NoveltyGAN | 0.8 | 0.81 | 0.41 | 0.54 | 0.65 |
| Cascade | 0.97 | 0.91 | 0.95 | 0.93 | 0.93 |
| OpenMax | 0.94 | 0.89 | 0.87 | 0.88 | 0.88 |
| Artifacts | 0.96 | 0.79 | 0.97 | 0.87 | 0.85 |
| BinaryNet | 0.97 | 0.9 | 0.94 | 0.92 | 0.92 |

## 5.4 Experiments on CIFAR

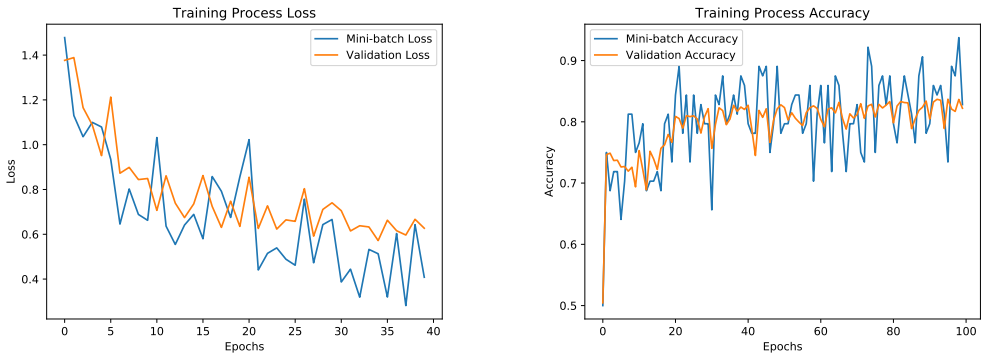
In the following section, results concerning the experiments on the CIFAR-10 and CIFAR-100 dataset are displayed. The normal images (CIFAR-10) are denoted by the negative class and novelties (four CIFAR-100 super classes) are denoted by the positive class. A summary of supervisor metrics for the CIFAR experiments are presented in Section 5.4.8.

#### Data partition

The data is divided into 45000 training images, 5000 validation images and 10000 test images from the CIFAR-10 dataset. From the CIFAR-100 dataset, 10000 test images are used containing the super classes: large carnivores, large man-made outdoor things, non-insect invertebrates, small mammals. For the training of supervisors, 15000 adversarial images were created using the FSMA, JSMA and Deepfool algorithms.

### 5.4.1 CIFAR-10 Neural Network

The network was trained for 40 epochs using a cross-entropy loss and a gradient descent optimizer with a learning rate of 0.001 and batch size of 64. The two first dropout layers use a rate of 0.4 and the third a rate of 0.5. As shown in Figure 5.21, both the loss and accuracy for the training and validation set improved during training.




**(a)** Loss. **(b)** Accuracy.

**Figure 5.21:** Loss and accuracy during training for the CIFAR neural network.

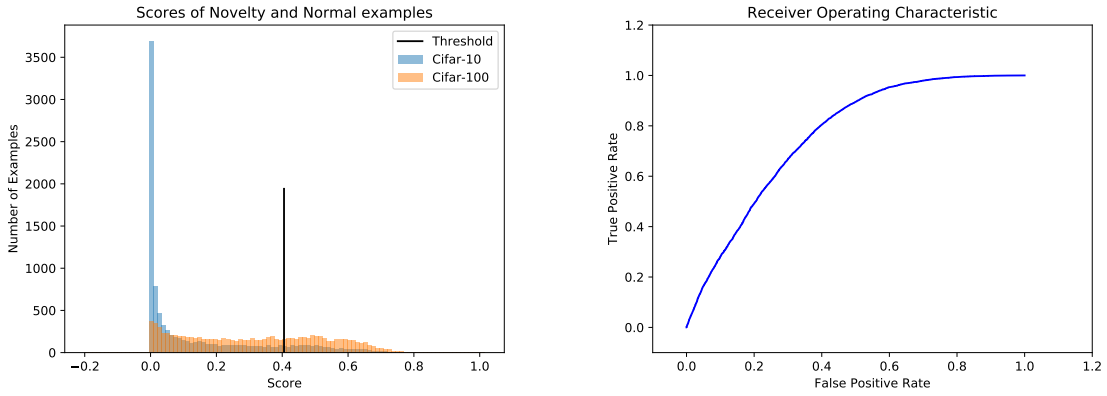
Evaluating the network on the CIFAR-10 test set yielded a test accuracy of 79%. Class-specific metrics are presented below in Table 5.3. The network achieves an F1 score of about 0.80 and above for most classes but have trouble with birds, cats, deer and dogs.

**Table 5.3:** F1-score, Precision, Recall and Support (number of samples of class in test set) on the CIFAR-10 part of the test set for the network used in the CIFAR experiments.

| Class | F1 score | Precision | Recall | Support |
|---|---|---|---|---|
| plane | 0.82 | 0.84 | 0.81 | 1000 |
| car | 0.9 | 0.94 | 0.87 | 1000 |
| bird | 0.7 | 0.85 | 0.59 | 1000 |
| cat | 0.59 | 0.72 | 0.5 | 1000 |
| deer | 0.76 | 0.69 | 0.84 | 1000 |
| dog | 0.7 | 0.63 | 0.78 | 1000 |
| frog | 0.84 | 0.82 | 0.85 | 1000 |
| horse | 0.84 | 0.79 | 0.9 | 1000 |
| ship | 0.89 | 0.86 | 0.92 | 1000 |
| truck | 0.88 | 0.85 | 0.9 | 1000 |

### 5.4.2 Baseline

The ROC curve for the Baseline supervisor in the CIFAR experiments is shown in Figure 5.22b. The curve is reflected in the histogram in Figure 5.22a. The histogram illustrates how the supervisor probabilities can be somewhat, visually, separated. The threshold used to classify the test data was set to the $90^{th}$ percentile of the scores from the training set: 0.41.




**(a)** Scores. **(b)** ROC curve.

**Figure 5.22:** Histogram of scores and the ROC curve for the Baseline supervisor on the test set in the CIFAR experiments.

To illustrate which classes are more commonly misclassified as novelties, the false positives relative to the class count of the 10 different classes are displayed below in Figure 5.23. Similar to the network performance seen in Table 5.3, the Baseline supervisor has more trouble with birds, cats and dogs.

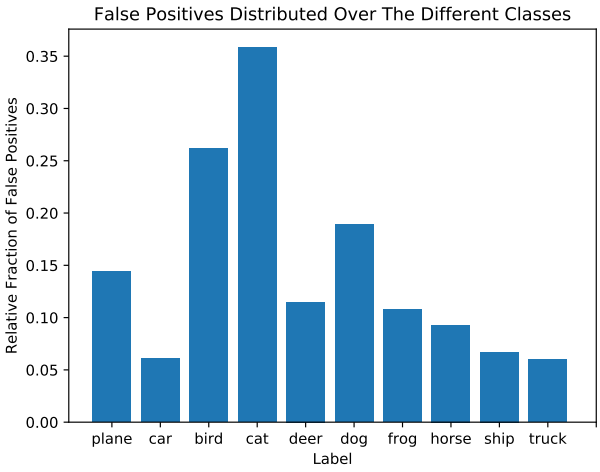



**Figure 5.23:** The false positives distributed over the 10 different classes relative to the class counts for the Baseline supervisor on the test set in the CIFAR experiments.

### 5.4.3 NoveltyGAN

The DCGAN was trained for 100 epochs using the loss described in Section 2.1.5 and the Adam optimizer with a learning rate of 0.0002, beta1 of 0.5, and a batch size of 100. How the discriminator and generator loss changed during the epochs are shown below in Figure 5.8.

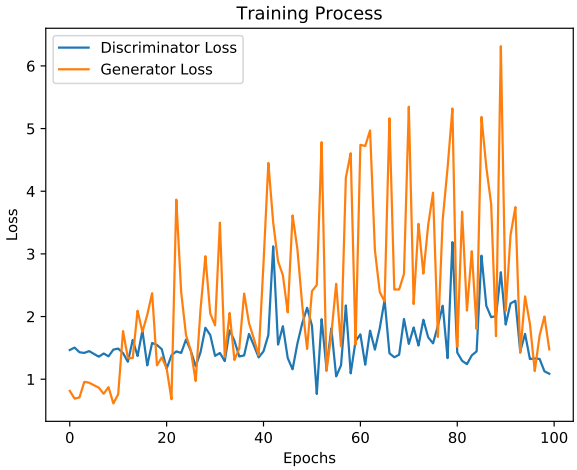
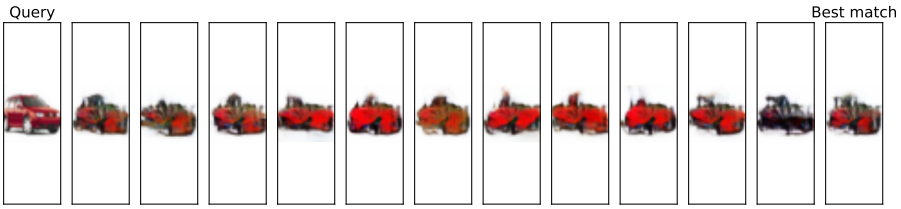



**Figure 5.24:** NoveltyGAN Discriminator and Generator loss during training in the CIFAR experiments.

The Adam optimizer was used with a learning rate of 0.25 and beta1 of 0.5 for 7 iterations. For each query, 12 starting points for the latent vector were tried and optimized separately. The final novelty score assigned to the query is the lowest score generated by the 12 latent vectors. A sample of how the regenerated images turned out is displayed below in Figure 5.25. The best match in terms of score is

denoted with a title above the corresponding image. The GAN manages to recreate the car somewhat but has more problems with the castle from CIFAR-100.

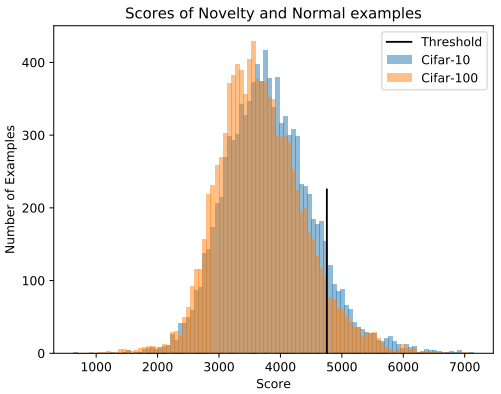



**(a)** CIFAR-10 query

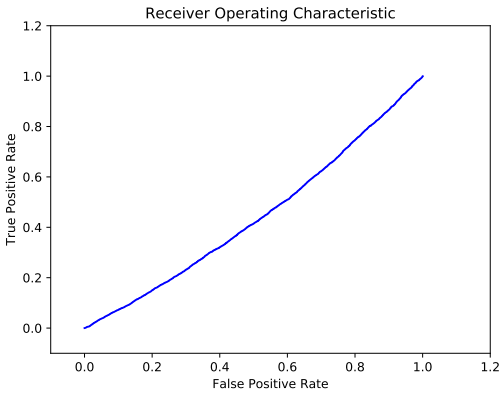

**(b)** CIFAR-100 query

**Figure 5.25:** Reconstructed images during the testing of the NoveltyGAN supervisor in the CIFAR experiments.

The threshold used to classify the test data was set to the $90^{th}$ percentile of the scores of the training set: 4755.87. The separation between CIFAR-10 and CIFAR-100 images in the test set in terms of novelty scores is illustrated by Figure 5.26a and the corresponding ROC curve in Figure 5.26b. The histogram shows the difficulties the supervisor has with separating novelties from normal inputs. The difficulties can also be seen in the gradually increasing ROC curve, close to a straight line.




**(a)** Scores.




**(b)** ROC curve.

**Figure 5.26:** Histogram of scores and the ROC curve for the NoveltyGAN supervisor on the test set in the CIFAR experiments.

The false positives relative to the class count of the 10 different classes are displayed below in Figure 5.11. Weaknesses can be seen in the classes car, cat, dog and truck.

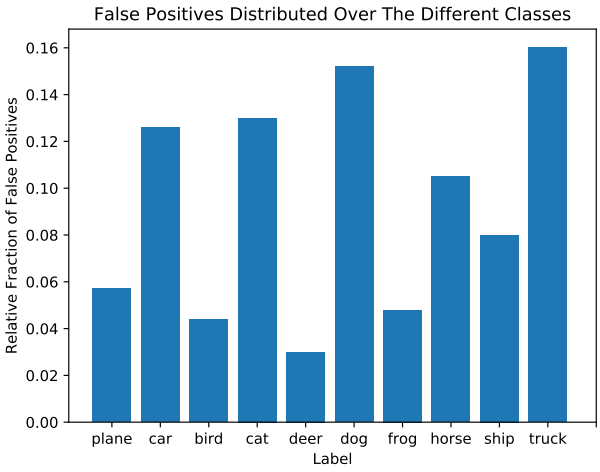
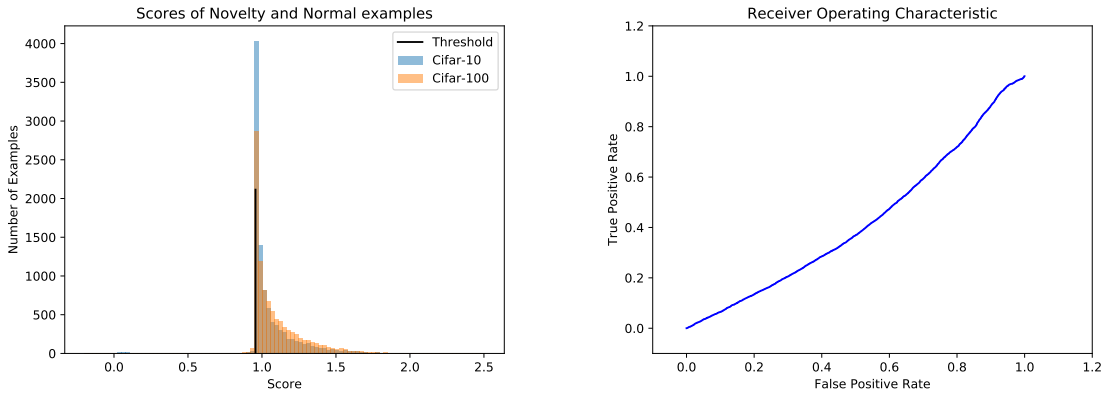



**Figure 5.27:** The false positives distributed over the 10 different classes relative to the class counts for the NoveltyGAN supervisor on the test set in the CIFAR experiments.

### 5.4.4 Cascade

Each stage of the cascade was trained on a subsample of all adversarials (abnormalities) and a corresponding amount of CIFAR-10 examples (normal). The supervisor target was that a maximum of 0.5% of the normal examples was to be classified as novelties. This resulted in a convergence at a false positive rate of 81% (with normal examples as the positive class). At the third stage of the classifier 100% of the normal examples had been detected. The threshold was set to the $10^{th}$ percentile of the scores on the training set: 0.96. Note that the classification rule with respect to the threshold is reversed here relative to the other supervisors since the decision function of the supervisor assigns high scores to normal examples and low to abnormal. The separation between CIFAR-10 and CIFAR-100 images in the test set is illustrated by Figure 5.28a and the corresponding ROC curve in Figure 5.28b. As can be seen by the histogram, the CIFAR-100 and CIFAR-10 examples are about the same scores, but with marginally higher scores for CIFAR-100. In reality the opposite should hold, this is reflected in the ROC curve which lies below the straight line indicating a poor separation.




**(a)** Scores. **(b)** ROC curve.

**Figure 5.28:** Histogram of scores and the ROC curve for the Cascade supervisor on the test set in the CIFAR experiments.

The false positives relative to the class count of the 10 different classes are displayed below in Figure 5.29. The car, plane and bird classes are most commonly misclassified as novelties.

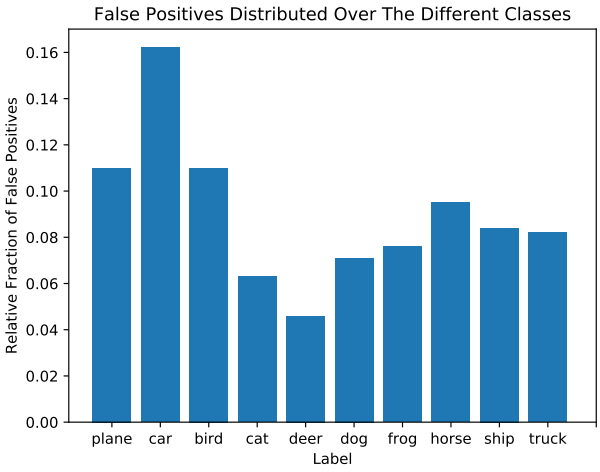
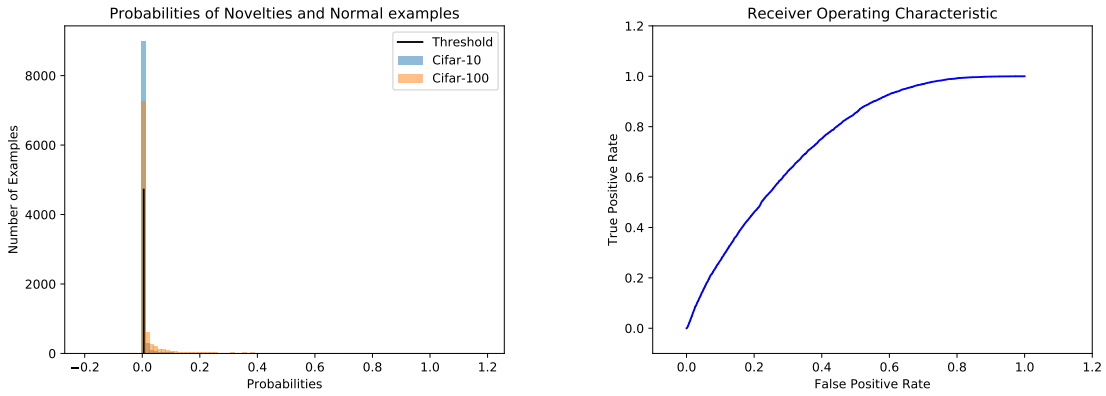



**Figure 5.29:** The false positives distributed over the 10 different classes relative to the class counts for the Cascade supervisor on the test set in the CIFAR experiments.

### 5.4.5 OpenMax

The OpenMax supervisor was fitted using the training set. The hyper-parameter tail length was tuned using the set of adversarial images and the training set with the AUC as a best-fit score, resulting in a tail length of 49. The classification threshold for the probability of a novelty was set using the $90^{th}$ percentile of the probabilities of the training set and resulted in 0.005. The separation between CIFAR-10 and CIFAR-100 in the test set is illustrated by Figure 5.30a and the corresponding ROC curve in Figure 5.14b. In the histogram it is visually hard to separate normal from novelty, but the ROC curve has a better form than most supervisors have on CIFAR.




**(a)** Scores. **(b)** ROC curve.

**Figure 5.30:** Histogram of scores and the ROC curve for the OpenMax supervisor on the test set in the CIFAR experiments.

The false positives relative to the class count of the 10 different classes are displayed below in Figure 5.31. As seen in the histogram, the plane, bird and cat classes are difficult for the supervisor.

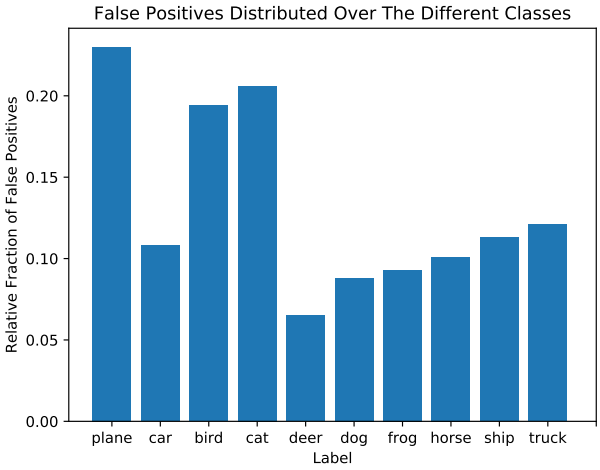
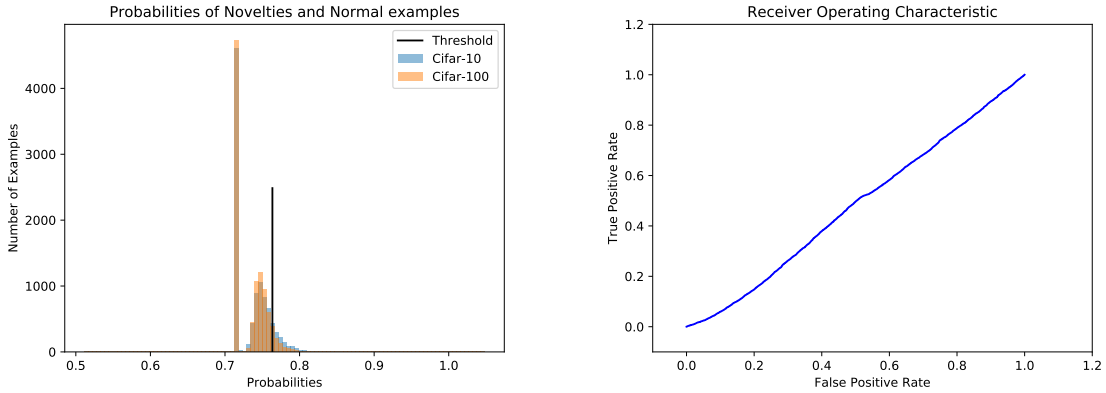



**Figure 5.31:** The false positives distributed over the 10 different classes relative to the class counts for the OpenMax supervisor on the test set in the CIFAR experiments.

### 5.4.6 Artifacts

The Artifacts supervisor was trained using a subset of the training set and adversarial images. The remainder of the training set was used for kernel density estimates. The hyper-parameter bandwidth was tuned using a subset of adversarial images and training set with the AUC as a best-fit score, resulting in a bandwidth of 0.5. The classification threshold for the probability of a novelty was set using the $90^{th}$ percentile of the probabilities of the training set: 0.76. The separation between

CIFAR-10 and CIFAR-100 images in the test set is illustrated by Figure 5.32a and the corresponding ROC curve in Figure 5.32b. The histogram shows two peaks but no separation between normal and novel images.




**(a)** Probabilities. **(b)** ROC curve.

**Figure 5.32:** Histogram of probabilities assigned to MNIST and Omniglot and the ROC curve for the Artifacts supervisor on the test set in the CIFAR experiments.

The false positives relative to the class count of the 10 different classes are displayed below in Figure 5.33. In the histogram it is clear that the supervisor has problems with the car class.

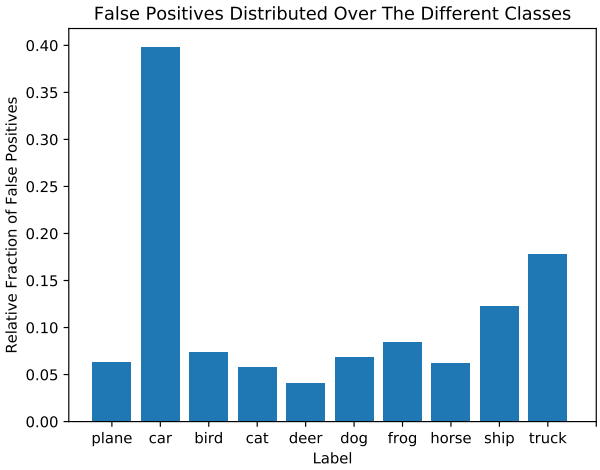
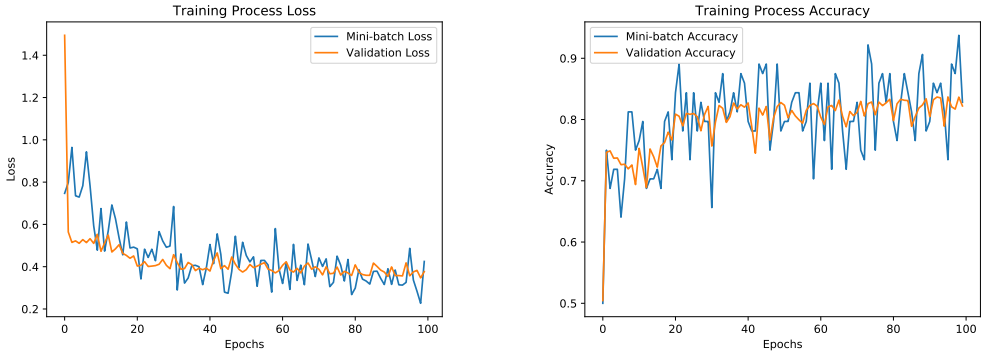



**Figure 5.33:** The false positives distributed over the 10 different classes relative to the class counts for the Artifacts supervisor on the test set in the CIFAR experiments.

### 5.4.7 BinaryNet

Using adversarial images and the training set, the BinaryNet supervisor was trained for 100 epochs using a cross-entropy loss and a gradient descent optimizer with a

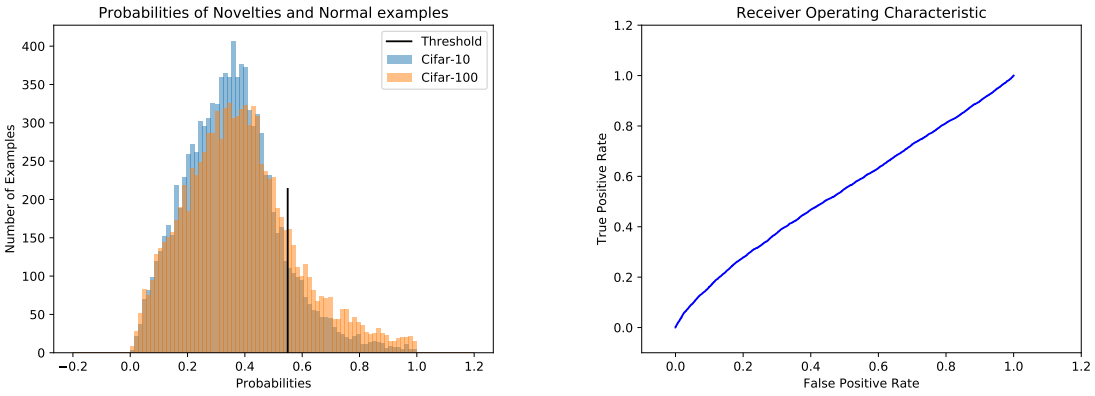
learning rate of 0.01, a dropout rate of 0.5 and a batch size of 64. The validation set consisted of 20% of the data. As shown in Figure 5.34, both the loss and accuracy for the training and validation set improved during training, but with some volatility.




**(a)** Loss. **(b)** Accuracy.

**Figure 5.34:** Accuracy and loss during training of the BinaryNet supervisor in the CIFAR experiments.

The threshold on the novelty probability produced by the binary network was set to the $90^{th}$ percentile of the probabilities yielded by the training set: 0.55. The separation between CIFAR-10 and CIFAR-100 images in the test set is illustrated by Figure 5.35a and the corresponding ROC curve in Figure 5.35b. No clear separation is visually seen in the histogram, and is also reflected in the ROC curve, which is close to a straight line.




**(a)** Scores. **(b)** ROC curve.

**Figure 5.35:** Histogram of scores and the ROC curve for the BinaryNet supervisor on the test set in the CIFAR experiments.

The false positives relative to the class count of the 10 different classes are displayed below in Figure 5.36. Largest difficulties are seen in the classes deer, frog, ship and truck.

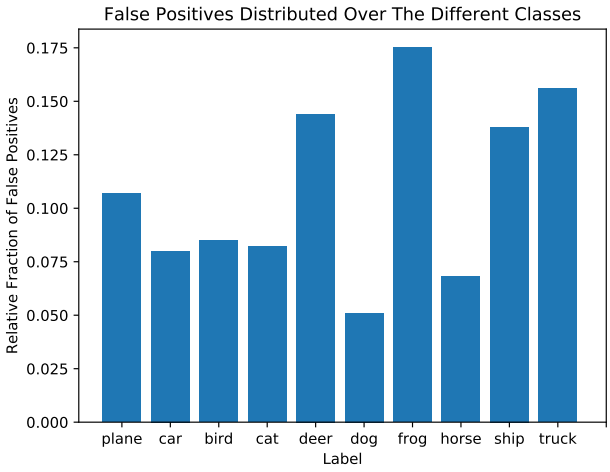



**Figure 5.36:** The false positives distributed over the 10 different classes relative to the class counts for the BinaryNet supervisor on the test set in the CIFAR experiments.

### 5.4.8 Comparison of Metrics

A summary of performance metrics for all supervisors are displayed below in Table 5.4. The supervisors show AUC scores of varying abilities. In contrast to the MNIST vs Omniglot experiments, only two supervisors show any ambition for separation: OpenMax and Baseline. Further details on supervisor-specific abilities for separation can be seen in the aforementioned sections. Three supervisors manage to achieve precision scores higher than 0.5. Hence, they correctly classify more CIFAR-100 images as novelties than falsely classify CIFAR-10 images as novelties. Worth noting is that the Artifacts supervisor manages to achieve a precision score far worse, which means it separates the normal from novel well but the probabilities are inverted. That means the probabilities are likely mixed up; normal images achieve high probabilities of being novelty and novel images the opposite. The recall score shows how well the supervisors classify novelties in relation to the number of available novelties there are. The best scores are achieved by the Baseline and OpenMax supervisors, which still achieve a low score. That means they are able to correctly classify some but not many of the CIFAR-100 images. The F1-score is the weighted average of the two aforementioned metrics, taking both false positives and false negatives into account. Since the Baseline and OpenMax supervisors achieve highest scores on precision and recall, they also do it for F1. Lowest F1 scores are achieved by the Cascade, Artifacts and NoveltyGAN supervisors since they all achieved low precision and, in specific, recall scores. The highest accuracies are achieved by the Baseline and OpenMax supervisors at about 60% while the other supervisors are about as good as random guessing.

**Table 5.4:** Performance metrics for the six supervisors on the test set in the CIFAR experiments.

| Supervisor | AUC | Precision | Recall | F1-score | Accuracy |
|---|---|---|---|---|---|
| Baseline | 0.76 | 0.72 | 0.38 | 0.49 | 0.61 |
| NoveltyGAN | 0.44 | 0.42 | 0.07 | 0.12 | 0.49 |
| Cascade | 0.42 | 0.49 | 0.08 | 0.14 | 0.50 |
| OpenMax | 0.74 | 0.72 | 0.34 | 0.46 | 0.60 |
| Artifacts | 0.48 | 0.38 | 0.07 | 0.12 | 0.48 |
| BinaryNet | 0.54 | 0.62 | 0.17 | 0.27 | 0.53 |

## 5.5 Experiments on Retinal OCT

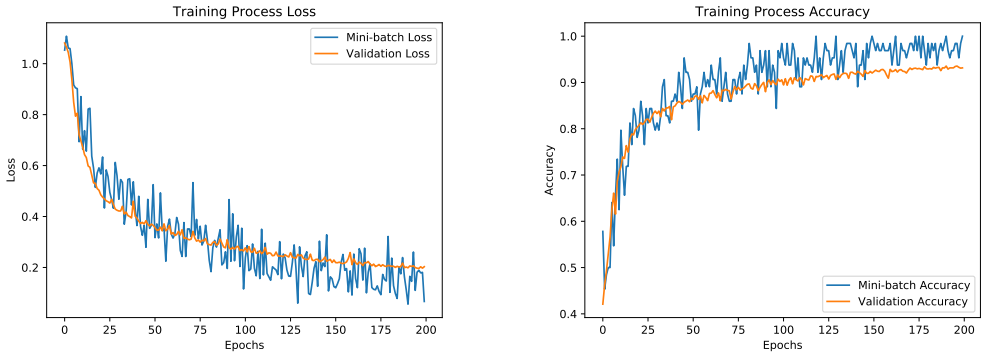
In the following section, results concerning the experiments on the Medical Retinal OCT dataset are presented. How the dataset is split and which classes are denoted as the positive class (novelty) and negative class (normal) are described below. A summary of supervisor metrics are presented in Section 5.5.8.

#### Data Partition

The class with the fewest number of occurrences, Drusen, was chosen to be the novelty class of the experiments. A balanced dataset of the remaining three classes was created. The size was hence dictated by the class with fewest number of occurrences: DME. The dataset contains 33000 images, 11000 of each of the three classes Normal, CNV and DME. The dataset was then divided into a training and validation set with a split of 10% for the latter. The test set of total size 16000 contains 8000 novelties in the form of the condition Drusen, 8000 known classes of which 598 DME, 3701 Normal and 3701 CNV. As in the MNIST vs Omniglot experiments, 5000 adversarials from each of the aforementioned algorithms were created.

### 5.5.1 Retinal OCT Neural Network

The network was trained for 200 epochs using a cross-entropy loss and a gradient descent optimizer with a learning rate of 0.003, a dropout rate of 0.5 and batch size of 64. Both the loss and accuracy for the training and validation set improved during training, as shown in Figure 5.37.




**(a)** Loss. **(b)** Accuracy.

**Figure 5.37:** Loss and accuracy during training for the Retinal OCT neural network.

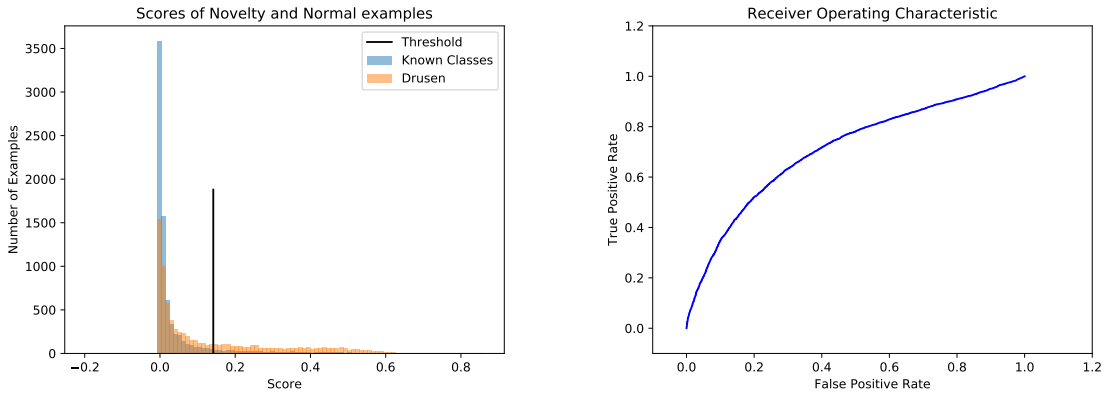
Evaluating the network on half the test set, only containing known classes, yielded a test accuracy of 96%. Note that the test set is not balanced in the case of known classes in this experiment, which effects the test accuracy. Class specific metrics are presented below in Table 5.5. The network achieves good scores with regard to the two classes Normal and CNV, but significantly lower scores for the DME class.

**Table 5.5:** F1 score, Precision, Recall and Support (number of samples of class in test set) on the known classes in the test set for the network used in the Retinal OCT experiments.

| Class | F1 score | Precision | Recall | Support |
|---|---|---|---|---|
| Normal | 0.97 | 0.97 | 0.97 | 3701 |
| CNV | 0.97 | 0.99 | 0.96 | 3701 |
| DME | 0.8 | 0.73 | 0.89 | 598 |

### 5.5.2 Baseline

The threshold used to classify the new data was set to the $90^{th}$ percentile of the scores of the training set: 0.14. The separation between the known classes and Drusen in the test set in terms of novelty scores is illustrated by Figure 5.38a and the corresponding ROC curve in Figure 5.38b. The histogram shows a separation between known classes and Drusen. The known classes are rarely being appointed higher scores. The separation is reflected in the increasing ROC curve.




**(a)** Scores. **(b)** ROC curve.

**Figure 5.38:** Histogram of probabilities assigned to the known classes and Drusen and the ROC curve for the Baseline supervisor on the test set in the Retinal OCT experiments.

To illustrate which of the classes most frequently are confused with the Drusen pathology, the false positives of the different classes relative their class counts are displayed below in Figure 5.39. Images of the class DME are most commonly misclassified as novelties followed by almost equal amount of normal and CNV.

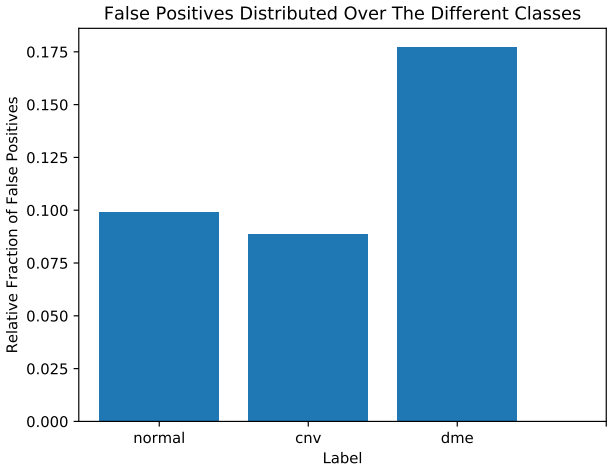



**Figure 5.39:** The false positives distributed over the 3 different classes relative to the class counts for the Baseline supervisor on the test set in the Retinal OCT experiments.

### 5.5.3 NoveltyGAN

The DCGAN was trained for 30 epochs using the loss described in Section 2.1.5 and the Adam optimizer (Kingma & Ba 2014) with a learning rate of 0.0002, beta1 of 0.5, and a batch size of 100. As seen in Figure 5.40, neither of the discriminator and generator losses diverges from the other during training.

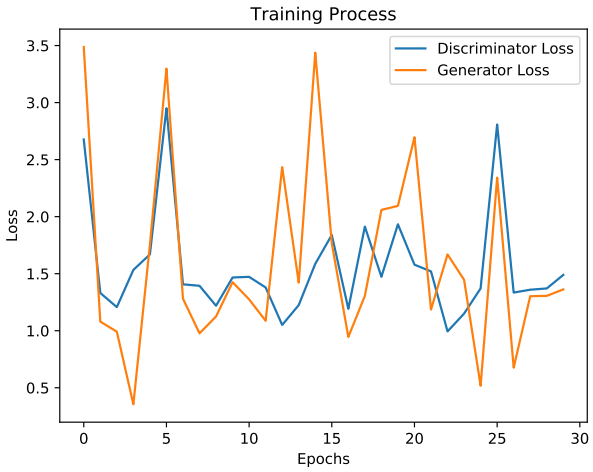
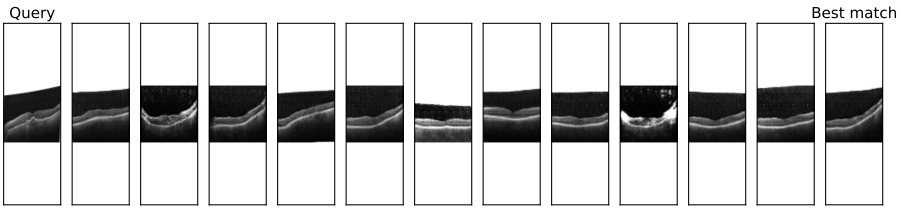
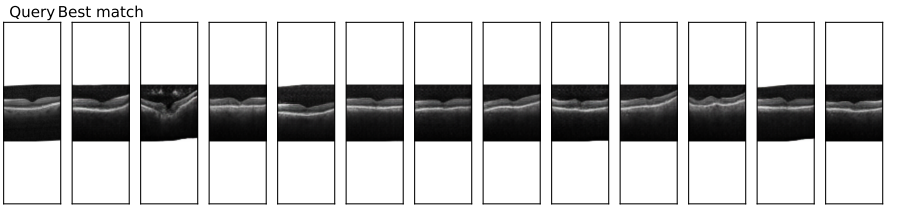



**Figure 5.40:** Discriminator and Generator loss during training in the Retinal OCT experiments.

After training, each query image from the test set was regenerated by changing the latent vector using the loss described in Section 5.2.2. The Adam optimizer was used with a learning rate of 0.25 and beta1 of 0.5 for 7 iterations. For each query, 12 starting points for the latent vector were tried and optimized separately. The final novelty score assigned to the query is the lowest score generated by the 12 latent vectors. A sample of how the regenerated images turned out is displayed below in Figure 5.41. By visual inspection, the supervisor manages to reconstruct the Retinal OCT images but not necessarily in the favor of the CNV class.




**(a)** CNV query

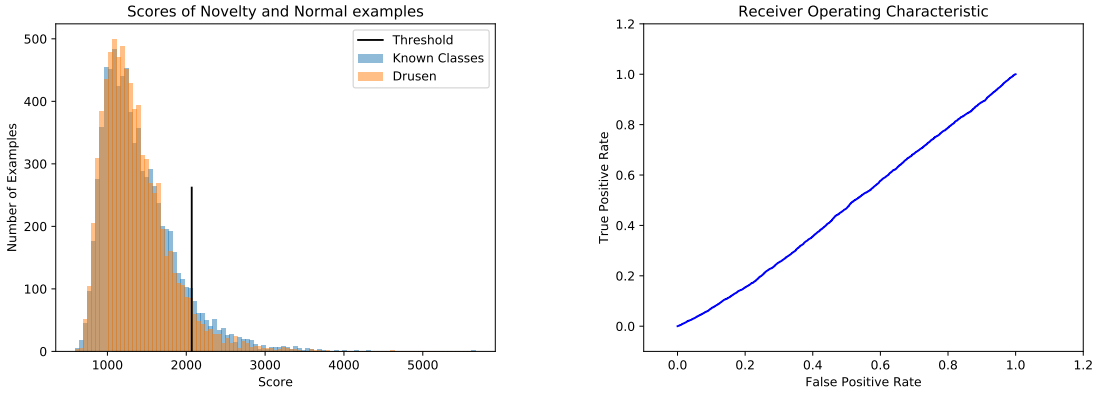



**(b)** Drusen query

**Figure 5.41:** Reconstructed images during the testing of the NoveltyGAN supervisor in the Retinal OCT experiments.

The scores on the entire test set were evaluated and a classification reached by setting the threshold to the $90^{th}$ percentile of the scores of the training set: 2069.27. The

separation between known classes and Drusen in the test set in terms of novelty scores is illustrated by Figure 5.42a and the corresponding ROC curve in Figure 5.42b. The histogram shows no separation between the known classes and Drusen. This is reflected in the ROC curve which approaches the straight line.




**(a)** Probabilities. **(b)** ROC curve.

**Figure 5.42:** Histogram of scores assigned to known classes and Drusen and the ROC curve for the NoveltyGAN supervisor on the test set in the Retinal OCT experiments.

The false positives of the different classes relative their occurrence are displayed below in Figure 5.43. Images of the classes DME and CNV are most commonly classified as novelties.

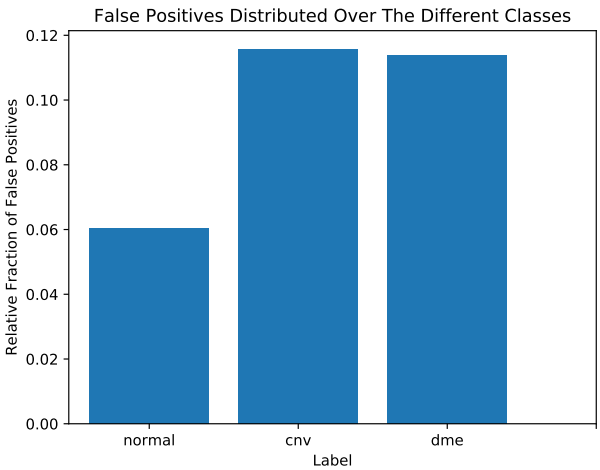
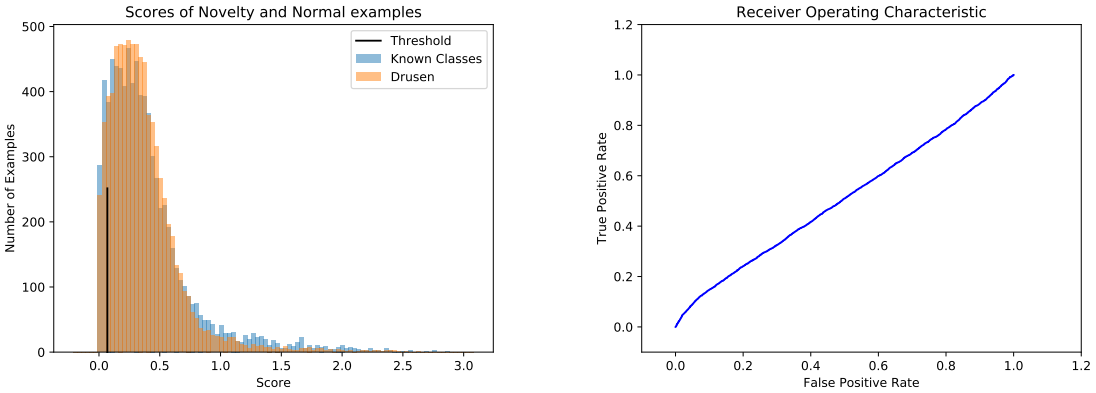



**Figure 5.43:** The false positives distributed over the 3 different classes relative to the class counts for the NoveltyGAN supervisor on the test set in the Retinal OCT experiments.

### 5.5.4 Cascade

Each stage of the cascade was trained on a subsample of all adversarials (abnormalities) and a corresponding amount of the known classes (normal). Note that normal is here meant to denote the non-adversarial examples belonging to one of the known classes. The algorithm target was that a maximum of 1% of the normal examples was to be classified as novelties. This resulted in a convergence at a false positive rate of 17% (with normal examples as the positive class). For this data set all stages of the cascade were required to detect all normal examples. The threshold was set to the $10^{th}$ percentile of the scores on the training set: 0.07. Note that the classification rule with respect to the threshold is reversed here relative to the other supervisors since the decision function of the supervisor assigns high scores to normal examples and low to abnormal. The separation between the known classes and Drusen in the test set is illustrated by Figure 5.44a and the corresponding ROC curve in Figure 5.44b. The histogram shows a small separation where a small proportion of the known classes is given a higher score than Drusen. The small separation is reflected in the ROC curve which approaches, but lies above, the straight line.




**(a)** Scores. **(b)** ROC curve.

**Figure 5.44:** Histogram of scores assigned to known classes and Drusen and the ROC curve for the Cascade supervisor on the test set in the Retinal OCT experiments.

The false positives of the different classes relative their occurrence are displayed below in Figure 5.43. The known classes are classified as Drusen about as frequently, relative to the other supervisors.

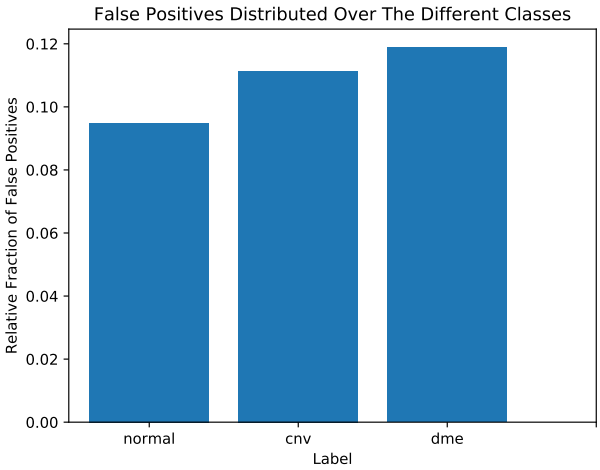
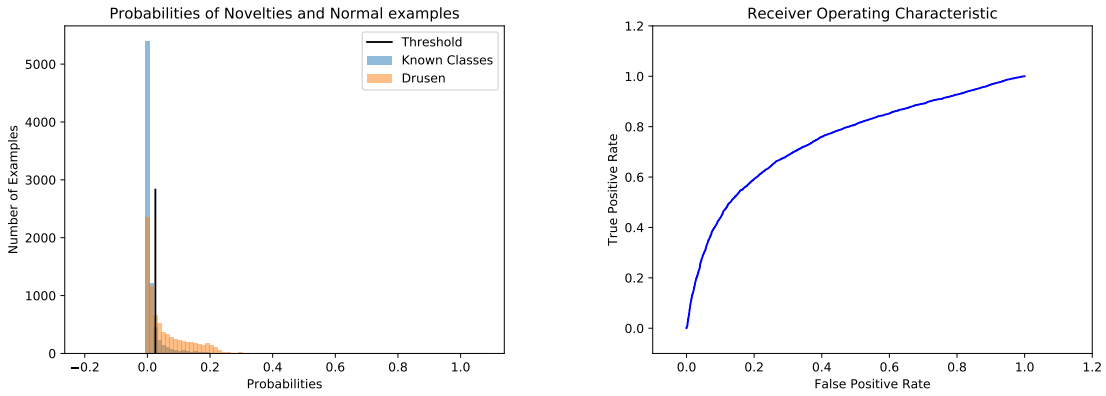



**Figure 5.45:** The false positives distributed over the 3 different classes relative to the class counts for the Cascade supervisor on the test set in the Retinal OCT experiments.

### 5.5.5 OpenMax

The OpenMax supervisor was fitted using the training data. The hyper-parameter tail length was tuned using the set of adversarial images and the training set with the AUC as a best-fit score, resulting in a tail length of 24. The classification threshold for the probability of a novelty was set using the $90^{th}$ percentile of the probabilities of the training set: 0.03. The separation between known classes and Drusen in the test set is illustrated by Figure 5.46a and the corresponding ROC curve in Figure 5.46b. The histogram shows a separation between known classes and Drusen. The known classes are rarely being appointed higher scores, which also is reflected in the small threshold. The separation is reflected in the increasing ROC curve which is similar as the one for the Baseline supervisor (Figure 5.38b).




**(a)** Probabilities. **(b)** ROC curve.

**Figure 5.46:** Histogram of scores assigned to known classes and Drusen and the ROC curve for the OpenMax supervisor on the test set in the Retinal OCT experiments.

The false positives of the different classes relative to their occurrence are displayed below in Figure 5.47. Images of the class Normal are most commonly classified as novelties followed almost equal fractions of DME and CNV.

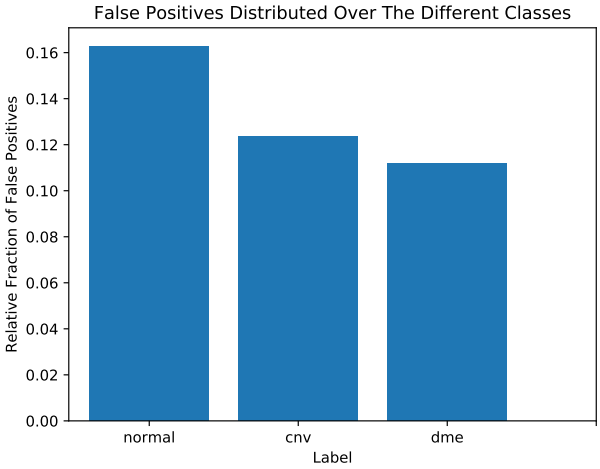
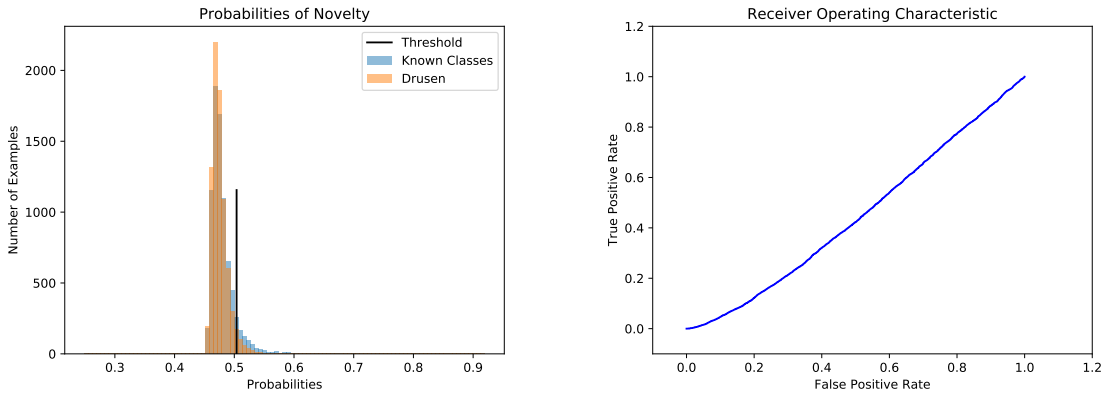



**Figure 5.47:** The false positives distributed over the 3 different classes relative to the class counts for the OpenMax supervisor on the test set in the Retinal OCT experiments.

### 5.5.6 Artifacts

The artifacts algorithm was trained using a subset of the training set and adversarial images, the remainder of the training set was used for kernel density estimates. The hyper-parameter bandwidth was tuned using a subset of adversarial images and training set with the AUC as a best-fit score, resulting in a bandwidth of 1.9. The classification threshold for the probability of a new class was set using the $90^{th}$

percentile of the probabilities of the training set: 0.5. The separation between known classes and the Drusen class in the test set is illustrated by Figure 5.48a and the corresponding ROC curve in Figure 5.48b. The histogram shows a poor separation between the known classes and Drusen, where some known classes are assigned higher probabilities for being a novelty than Drusen. This is reflected by the ROC curve being under the straight line.




**(a)** Scores. **(b)** ROC curve.

**Figure 5.48:** Histogram of scores assigned to known classes and Drusen and the ROC curve for the Artifacts supervisor on the test set in the Retinal OCT experiments.

The false positives of the different classes relative their occurrence are displayed below in Figure 5.49. Images of the classes CNV DME and most commonly classified as novelties followed by a small fraction of the Normal class.

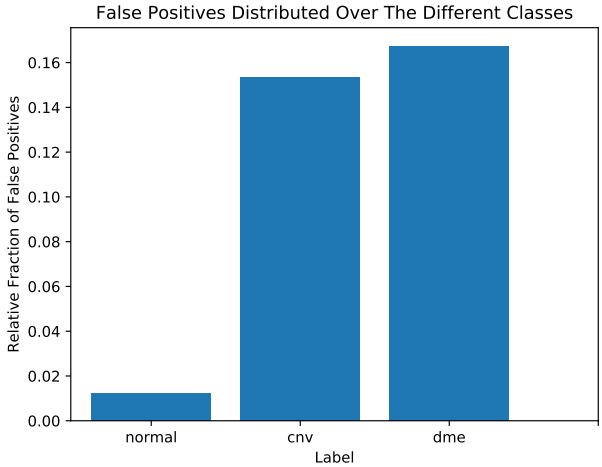
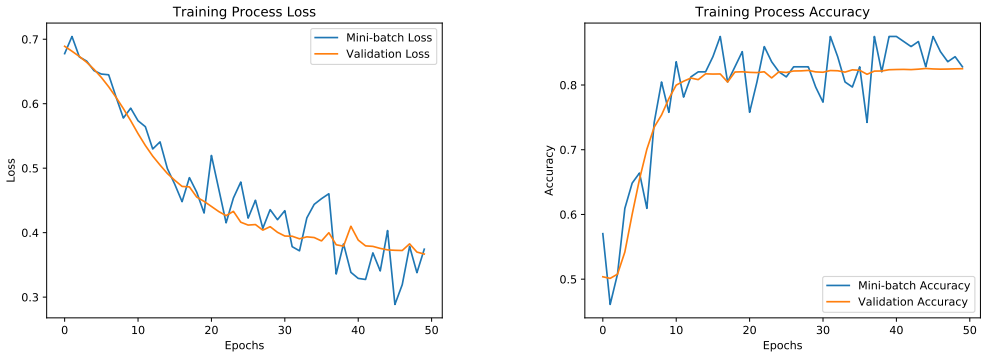



**Figure 5.49:** The false positives distributed over the 3 different classes relative to the class counts for the Artifacts supervisor on the test set in the Retinal OCT experiments.

### 5.5.7 BinaryNet

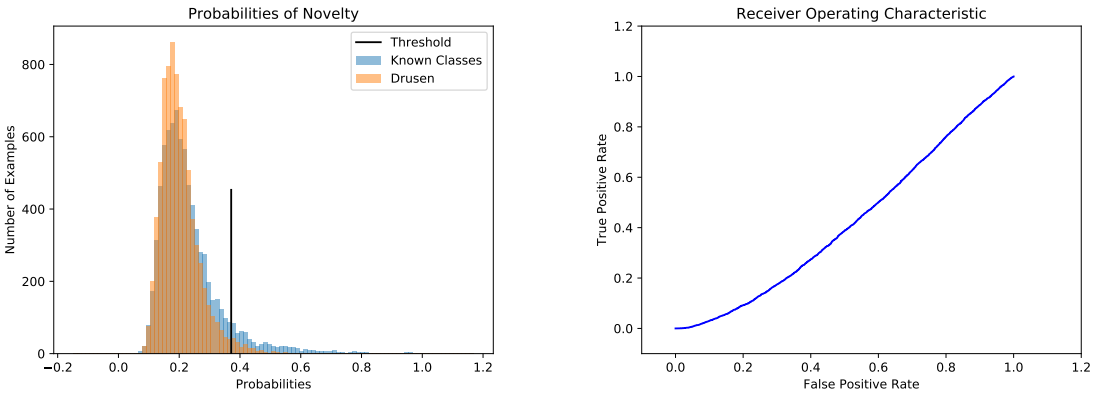
Using adversarial images and the training set the binary classifier was trained for 50 epochs using a cross-entropy loss and a gradient descent optimizer with a learning rate of 0.001, a dropout rate of 0.5 and batch size of 128. 20% of the data was used as the validation set. As shown in Figure 5.50, both the loss and accuracy for the training and validation set improved during training,




**(a)** Loss. **(b)** Accuracy.

**Figure 5.50:** Accuracy and loss during training of the BinaryNet supervisor in the Retinal OCT experiments.

The threshold on the output probability of novelty produced by the binary network was set to the $90^{th}$ percentile of the probabilities yielded by the training set: 0.37. The separation between known classes and the Drusen class in the test set is illustrated by Figure 5.51a and the corresponding ROC curve in Figure 5.51b. The histogram shows a poor separation between the known classes and Drusen, where some known classes are assigned higher probabilities for being a novelty than Drusen. This is reflected by the ROC curve being under the straight line.




**(a)** Scores. **(b)** ROC curve.

**Figure 5.51:** Histogram of scores assigned to known classes and Drusen and the ROC curve for the BinaryNet supervisor on the test set in the Retinal OCT experiments.

The false positives of the different classes relative their occurrence are displayed below in Figure 5.52. Images of the classes CNV DME and most commonly classified as novelties followed by a small fraction of the Normal class as for the Artifacts supervisor.

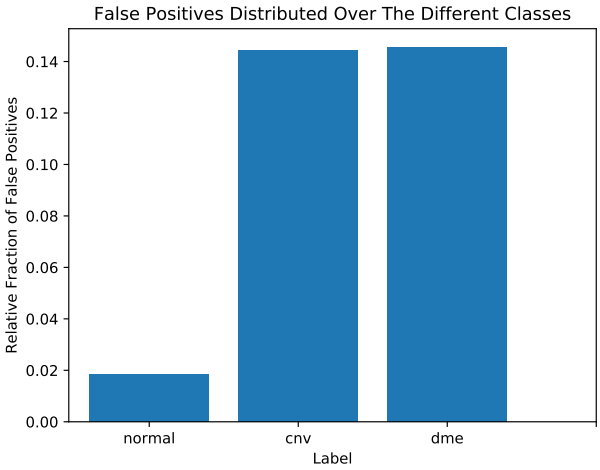



**Figure 5.52:** The false positives distributed over the 3 different classes relative to the class counts for the BinaryNet supervisor on the test set in the Retinal OCT experiments.

### 5.5.8 Comparison of Metrics

The performance metrics for all supervisors are displayed below in Table 5.6. Using the AUC the supervisors can be divided into two groups. Both the Baseline and OpenMax supervisor achieves, in the context, good scores over 0.5 which reflects that they can separate novelties from normal examples. The other supervisors however end up around 0.5, meaning that a poor separation is achieved. The same pattern is emerging for the other metrics as well. The Baseline and OpenMax supervisor achieve the highest precision and are able to correctly classify Drusen as novelties while also minimizing the number of known classes classified as novelties. This is in contrast to the other supervisors which are all achieving low precision scores. In the recall score there is a difference between the Baseline and OpenMax. OpenMax achieves the highest score which means it is to a higher degree able to classify a large portion of the Drusen examples as novelties. The Baseline supervisor is not far behind in comparison to the other supervisors achieving near-zero scores. The F1 score is highest for the OpenMax and the Baseline supervisor, outperforming the other supervisors. The pattern is present in the accuracy as well. The OpenMax supervisor achieves the highest accuracy at 69% followed by the Baseline at 62%. The rest of the supervisors all have an accuracy below 50%, meaning randomly classifying incoming examples with equal probabilities would perform better (at 50% accuracy).

**Table 5.6:** Performance metrics for the six supervisors on the test set in the Retinal OCT experiments.

| Supervisor | AUC | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|---|
| Baseline | 0.71 | 0.78 | 0.35 | 0.48 | 0.62 |
| Cascade | 0.51 | 0.46 | 0.09 | 0.15 | 0.49 |
| OpenMax | 0.75 | 0.79 | 0.52 | 0.62 | 0.69 |
| Artifacts | 0.45 | 0.3 | 0.04 | 0.07 | 0.47 |
| BinaryNet | 0.42 | 0.21 | 0.02 | 0.04 | 0.47 |
| NoveltyGAN | 0.47 | 0.4 | 0.06 | 0.11 | 0.49 |

## 5.6 Characteristics

In this section, observed characteristics in the supervisors are described. Valuable characteristics will, if possible, be utilized in the creation of a new supervisor in Section 5.7.

### 5.6.1 Baseline

The Baseline was among the best performing supervisors across all scenarios. The results confirm the hypothesis that a well-performing network's probabilities do contain information regarding the uncertainty of the network when presented with novel inputs. The separation is characterized by a histogram where the normal examples have a distinct peak at near zero probability of novelty, and are overlapped by a uniform-like distribution of the novelties. This means there is not a clear distributional separation like many of the other supervisors achieved in the MNIST vs Omniglot experiments. This is reflected in a lower recall score for the Baseline supervisor. The number of false positives for known classes of the Baseline supervisor has a clear connection to the recall score in the classifier network. A low recall score for a class in the classifier correspond to high fraction of false positives in the supervisor.

### 5.6.2 NoveltyGAN

The DCGAN component of the NoveltyGAN supervisor is able to generate realistic looking images but have a harder time separating between novelties and normal images. An apparent reason is that the more varied the dataset is, the harder it is to reconstruct images and hence separate between the normal and novel images. Another aspect is the number of iterations while optimizing over the latent vectors; a higher number of iterations would likely cause the reconstruction to converge closer to the normal query image. The supervisor is highly dependent on the performance of the DCGAN. Given a perfect DCGAN, able to only reconstruct images from the training set distribution, the novelty detection performance would be asymptotic. This is an undesirable trait since there is no guarantee that such a GAN exists for all datasets. The NoveltyGAN is however the only supervisor not dependent on the performance and activations of the CNN classifier.

### 5.6.3 Cascade

The Cascade supervisor performed well in the MNIST vs Omniglot experiments but, as many other supervisors trained using adversarials, did not succeed in the other scenarios. The layer statistics extracted from the PCA-transforms are fed into a cascade classifier using normal examples against adversarials. The cascade classifier's ability to separate adversarials from normal examples does not generalize good enough to be able to detect novelties. It is hard to determine if the failure is dependent of the information in the statistics or because of the adversarials. It is possible that replacing the support vector classifiers by another algorithm, that is commonly used for anomaly detection to find abnormalities in the activations, would yield more interpretable results.

### 5.6.4 OpenMax

Since the OpenMax supervisor creates its Weibull distributions from the mean activation vectors in the CNN, it is highly aligned with the invariability in the outputs. Bendale & Boult (2016) argue that classes are connected to each other and that such behaviours can be seen in the Softmax outputs. It is therefore feasible that since the CNN recall scores are low for birds and cats, then sometimes, similar images of planes are affected in the mean activation vectors. This behaviour is also observable in the MNIST vs Omniglot experiments with classes 2, 6 and 9. All three numbers have similarities in form. The differences in the false positive rates of the Retinal OCT experiments are too small to make such conclusions.

The supervisor achieves good results in all experiments in relation to the other supervisors. It is therefore reasonable to argue that mean activation vectors contain information utilizable for the detection of novelties.

### 5.6.5 Artifacts

The Artifacts supervisor utilizes dropout to find uncertainties in the network when fed both normal and adversarial images. The uncertainties are associated to the robustness of the CNN and are a part of the logistic regression predicting if an image is novel or not. To conclude the results in the CIFAR and Retinal OCT experiments, the Artifacts algorithm cannot be used for the detection of novelties when trained on normal and adversarial images even if it in the less complex MNIST vs Omniglot experiments manages to achieve a high AUC score of 0.96. In the CIFAR experiment the supervisor has particular problems with the car class, which cannot be connected to any of the CNN metrics.

### 5.6.6 BinaryNet

The BinaryNet supervisor is trained on normal and adversarial images. For the MNIST vs Omniglot experiments the supervisor achieves a high AUC score of 0.97 which means a good separation. In the CIFAR experiments though, it only achieves an AUC score of 0.54 and goes down to 0.42 for the Retinal OCT experiments.

The impression is that BinaryNet has more difficulties in separating novelties from normal, when facing more complex scenarios. Therefore it seems possible but hard to train a CNN on normal and adversarial images, for the use in supervising tasks.

## 5.7 Thesis Supervisor

### Motivation

The best performing supervisors (OpenMax and Baseline) utilized the uncertainties in the logits layer (the baseline does it implicitly). Hence, they are bound to miss novelties that the network thinks it knows. Some novel inputs simply yield a prediction with a high confidence. In the MNIST experiments where both supervisors reach near perfect metrics (together with the other supervisors). The cause of this is likely the well-defined dataset. With more varied datasets the problem gets harder. There may very well be ways to create even better supervisors based upon the late layer activations. However, the information regarding the novelties in the last layer is limited. As mentioned, there will be novelties in the open set that cannot be detected. To be able to rise over the problem, more information explicitly related to the image itself, such as earlier layer activations rather than late layer activations, need to be utilized. As shown in the supervisors Artifacts, Cascade and BinaryNet, using adversarials is not successful for representing novelties. Hence, it is more suitable to use the existing training data to, instead of training a supervisor, find scores or statistics that are characteristic for normal images.

### 5.7.1 Analysis of Neural Network Layers

#### Earlier Layers

To address the concern about information explicitly related to the image itself, earlier layers of the networks were investigated. Since the dimension of the layers prior to the logits is high, some kind of dimensionality reduction need to take place. Szegedy et al. (2013) highlight the importance of an activation of a feature (or neuron, meaning a high value) and its connection to similar inputs. Hence, the maximum values in the convolutional layers (feature maps) were inspected. Images in the training set should give rise to these high activations while unfamiliar should not. A meaningful score could then be $L_2$ norm of all maximum activations in all feature maps of a convolutional layer. Consider the second layer of a neural network; a $WxBx\beta$ layer with activations, $x$, the maximum value, $m^2$, of each filter would be

$$m_k^2 = \max_{\substack{i\in\{1,2...,W\},\\ j\in\{1,2,...B\}}} x_{ijk}, \quad k = 1, 2, ..., \beta. \tag{5.10}$$

The score for the second layer, $n^2$, is then calculated as

$$n^2 = ||m^2||_2. \tag{5.11}$$

For the fully connected layers prior to the logits the $85^{th}$ percentile was used as the score.

### Logits

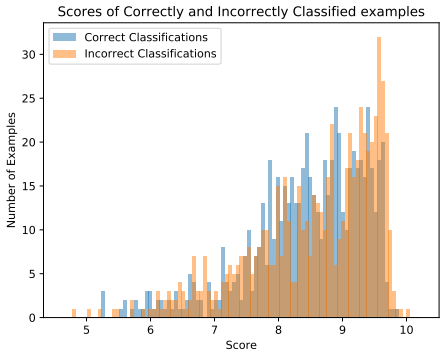
The OpenMax supervisor is based upon the distance from the logits to the mean activation vectors, and the Baseline on the uniformity of these values after the Softmax activation function; both proved to be effective. A similar approach would be to leverage the dropout-layer placed before the logits in all classification networks used in this thesis. First by recording the value of the without dropout. Then to run $num_{iter}$ iterations with dropout enabled recording the new logits. Novelties which were characterized by uniformity should have retained that uniformity throughout the iterations. This means that the difference between the logits during the iterations should not deviate from the original one. On the other hand, if the classification changes for known classes, the values in the logits should deviate. Consider an input, $x$, which yields logits, $l^o$ without dropout enabled. Running $num_{iter} = 10$ iterations would yield 10 other logits $l^i, i = 1, 2, .., 10$. The score can then be expressed as the mean-square difference between the originally observed logits and the ones observed during the iterations. The score, $n^{log}$, is expressed mathematically in (5.12).

$$n^{log} = \frac{1}{num_{iter}} \sum_{i=1}^{10} (l^o - l^i)^2 \tag{5.12}$$

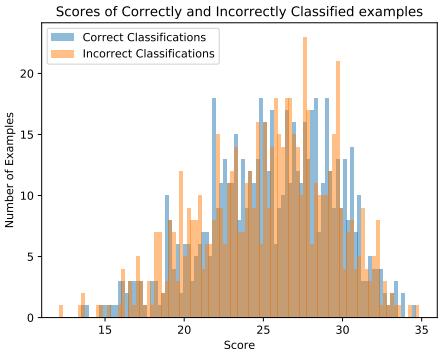
To see if the scores are suitable for separating unfamiliar from normal examples, adversarials were previously used. However, due to the ineffectiveness in using them for the CIFAR and Retinal OCT experiments, misclassified examples were used to represent unfamiliar examples instead.

### MNIST Neural Network

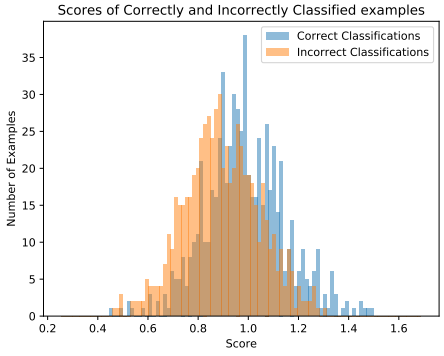
Only 650 misclassifications occurred in the training set. Hence, the data used for the experiments with MNIST contained 650 random samples that were correctly classified and the 650 misclassified. The separation between misclassifications and correctly classified examples in terms of layer-scores for each layer in the MNIST neural network is presented below in Figure 5.53 together with the AUCs. As seen in the figure, none of the two first layer-scores manage to separate the two categories. The first layer even assigns higher scores to incorrect classifications. The fully connected layer's and logits' layer-scores, does a better job and achieves higher AUC values. Note that the AUC scores are gradually increasing with the layers, and then decrease slightly in the logits layer.



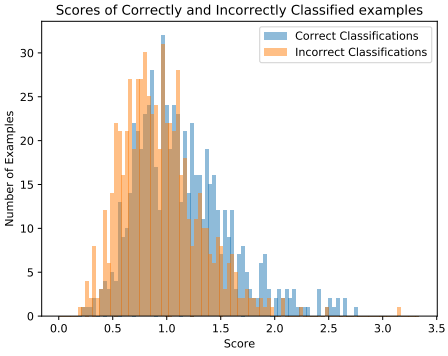

**(a)** 1st layer (score $n^1$) with an AUC of 0.45.




**(b)** 2nd layer (score $n^2$) with an AUC of 0.51.




**(c)** 3rd layer (score $n^2$) with an AUC of 0.66.

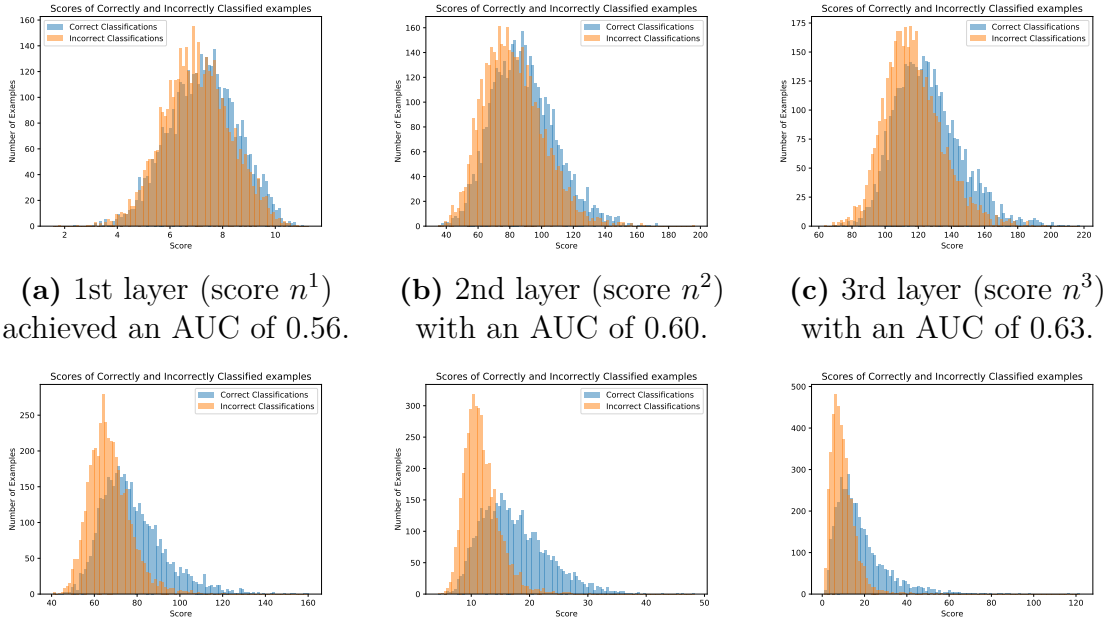



**(d)** Logits layer (score $n^{log}$) with an AUC of 0.63.

**Figure 5.53:** Histogram of scores assigned to correctly and incorrectly classified MNIST examples in the training set for the Thesis supervisor.

### CIFAR neural network

In the training set, 4283 misclassifications occurred. Hence the data used for the experiments with CIFAR contain 4283 random samples that were correctly classified, and 4283 misclassified. The separation between misclassified and correctly classified examples in terms of layer-scores for each layer in the CIFAR neural network is presented below in Figure 5.54 together with the AUC. As seen in the figure, the first layer-score does not manage to separate the two categories well. The AUC scores and separation for the next layers are gradually increasing with a peak in the fifth layer, second to last, and then decreasing in the logits layer.




**(a)** 1st layer (score $n^1$) achieved an AUC of 0.56.

**(b)** 2nd layer (score $n^2$) with an AUC of 0.60.

**(c)** 3rd layer (score $n^3$) with an AUC of 0.63.

**(d)** 4th layer (score $n^4$) with an AUC of 0.73.

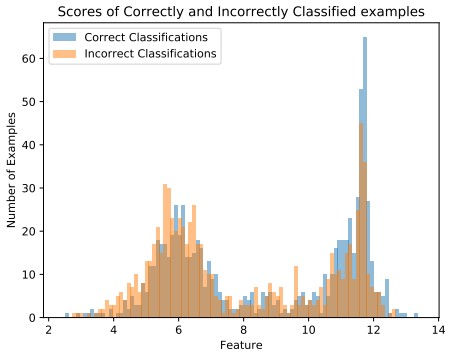
**(e)** 5th layer (score $n^5$) with an AUC of 0.82.

**(f)** Logits layer (score $n^{log}$) with an AUC of 0.74.

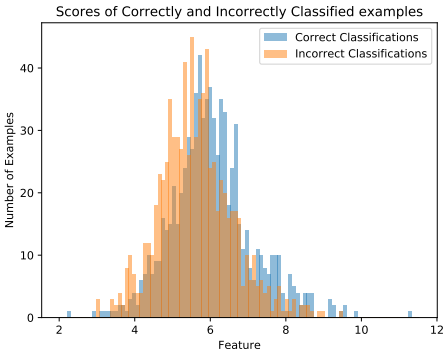
**Figure 5.54:** Histogram of scores assigned to correctly and incorrectly classified CIFAR examples in the training set for the Thesis supervisor.

### Retinal OCT Neural Network

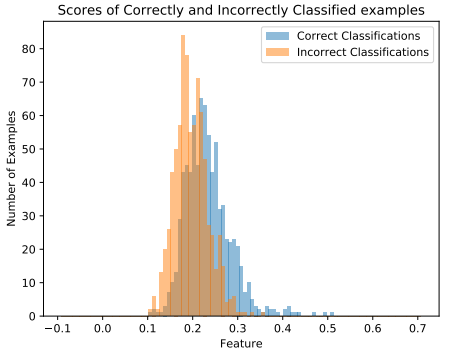
In the training set, 796 misclassifications occurred. Hence, the data used for the experiments with Retinal OCT contains 796 random samples, that were correctly classified, and 796 misclassified. The separation between misclassified and correctly classified examples in terms of layer-scores for each layer in the Retinal OCT neural network is presented below in Figure 5.55 together with the AUC. The same pattern as in the previous datasets occur here. The AUC scores and separation for the layers are gradually increasing and peaking in the second to last layer, and then decreasing in the logits layer.



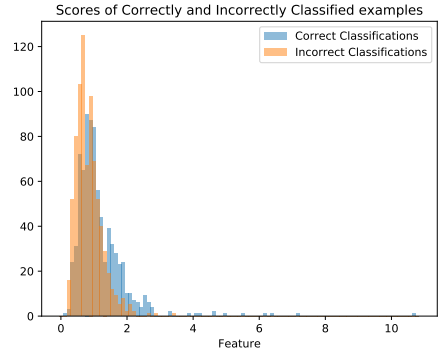

**(a)** 1st layer (score $n^1$) with an AUC of 0.59.




**(b)** 2nd layer (score $n^2$) with an AUC of 0.62.




**(c)** 3rd layer (score $n^2$) with an AUC of 0.72.




**(d)** Logits layer (score $n^{log}$) with an AUC of 0.66.

**Figure 5.55:** Histogram of scores assigned to correctly and incorrectly classified Retinal OCT examples in the training set for the Thesis supervisor.

### 5.7.2 Implementation

Given a dataset and a neural network with $l$ layers, the correctly and incorrectly classified examples are calculated. Assume the number of correctly classified examples is $n_c$ and incorrectly $n_w$, where $n_c > n_w$ (otherwise the classifier would not fulfill any purpose). A random subsample of size $n_w$ is drawn from the $n_c$ correctly classified examples and the layer-scores, explained earlier, are extracted for every layer using the subsample and the $n_w$ incorrectly classified examples. The AUCs calculated from these scores are denoted $AUC^1, AUC^2, ...AUC^l$. Using all the correctly classified examples, scaling values (maximum value) for each score are calculated and denoted by $s^1, s^2, ...s^l$, respectively.

Given a new query image, the total supervisor score, $q_s$, is calculated as the weighted sum of the layer scores as

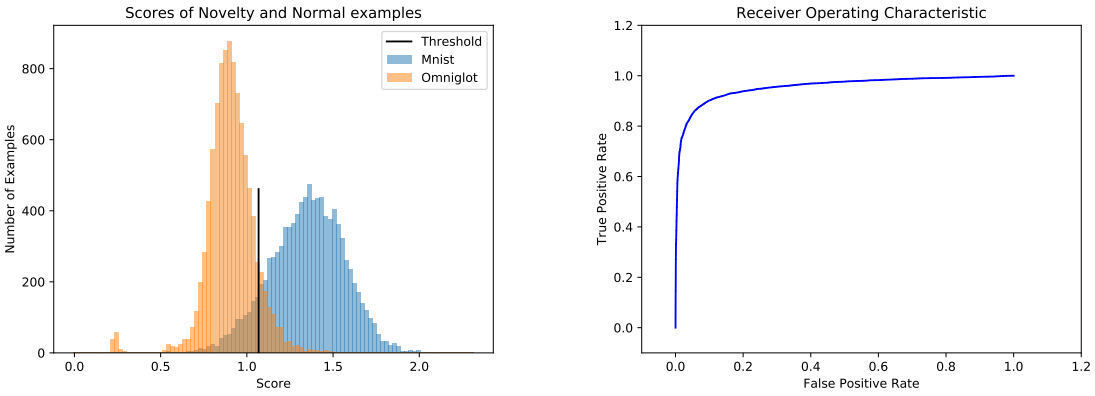
$$q_s = \sum_{i=1}^{l-1} \frac{AUC^i}{s^i} n^i + \frac{AUC^l}{s^l} n^{log}. \tag{5.13}$$

The idea is that layer scores that creates better separation for correctly and incor-

rectly classified examples should also separate novelties and normal examples better and are given a higher weight. Note that the classification rule with regard to a threshold will be reversed relative to the other supervisors (excluding the Cascade supervisor which also have a reversed classification rule).

### 5.7.3 Results for MNIST vs Omniglot

The 650 incorrectly classified MNIST examples and a subsample of the same size of the correctly classified examples were used to extract the AUC values. The threshold was set to the $10^{th}$ percentile of the scores (5.13) of the correctly classified examples from the training set: 1.07. The separation between Omniglot and MNIST images in the test set is illustrated by Figure 5.56a and the corresponding ROC curve in Figure 5.56b. The result is similar to the results for the other supervisors (excluding NoveltyGAN) with a histogram that shows a clear separation between Omniglot and MNIST and a sharply increasing ROC curve. Note that novelties are consistently given lower scores than misclassified examples creating a greater separation than the previous histograms indicated.




**(a)** Scores. **(b)** ROC curve.

**Figure 5.56:** Histogram of scores assigned to MNIST and Omniglot and the ROC curve for the Thesis supervisor on the test set in the MNIST vs Omniglot experiments.

To illustrate which classes are more commonly misclassified as novelties, the false positives relative to the class count of the 10 different classes are displayed below in Figure 5.57. Images of the number 1 followed by 8 and 9 are most commonly classified as novelties.

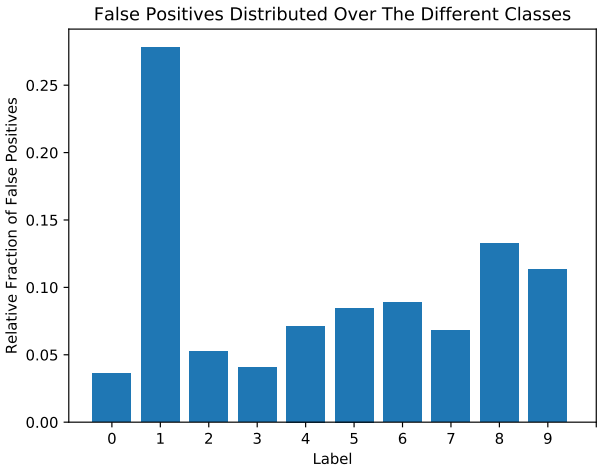



**Figure 5.57:** The false positives distributed over the 10 different classes relative to the class counts for the Thesis supervisor on the test set in the MNIST vs Omniglot experiments.

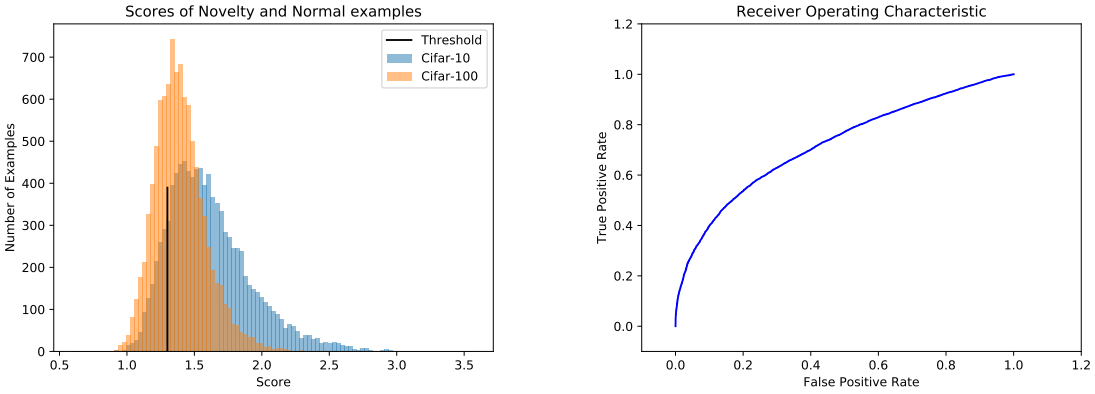
The performance metrics for the Thesis supervisor are displayed below in Table 5.7. The metrics are all comparable to the best performing supervisors (Table 5.2) and shows a clear balance in precision and recall, which is reflected in the F1-score.

**Table 5.7:** Performance metrics for the Thesis supervisor on the test set in the MNIST vs Omniglot experiments.

| | AUC | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|---|
| Thesis supervisor | 0.96 | 0.9 | 0.9 | 0.9 | 0.9 |

### 5.7.4 Results for CIFAR

The 4283 incorrectly classified CIFAR examples and a subsample of the same size of the correctly classified examples was used to extract the AUC values. The threshold was set to the $10^{th}$ percentile of the scores of the correctly classified examples from the training set: 1.30. The separation between CIFAR-100 and CIFAR-10 images in the test set is illustrated by Figure 5.58a and the corresponding ROC curve in Figure 5.58b. The result is a visible separation but with a significant overlap.




**(a)** Scores. **(b)** ROC curve.

**Figure 5.58:** Histogram of scores and the ROC curve for the Thesis supervisor on the test set in the CIFAR experiments.

To illustrate which classes are more commonly misclassified as novelties, the false positives relative to the class count of the 10 different classes are displayed below in Figure 5.59. The Thesis supervisor has more trouble with deers, birds and cats.

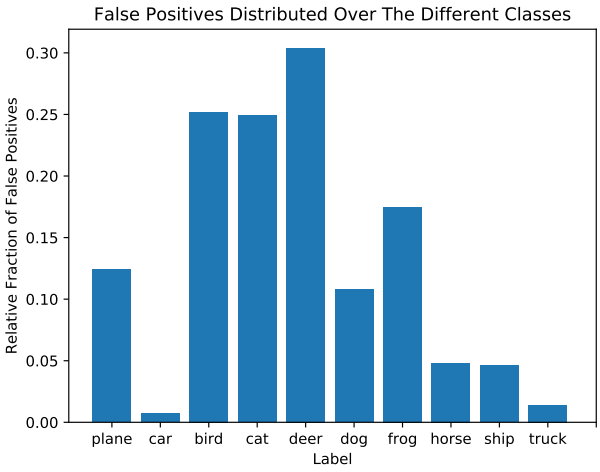



**Figure 5.59:** The false positives distributed over the 10 different classes relative to the class counts for the Thesis supervisor on the test set in the CIFAR experiments.

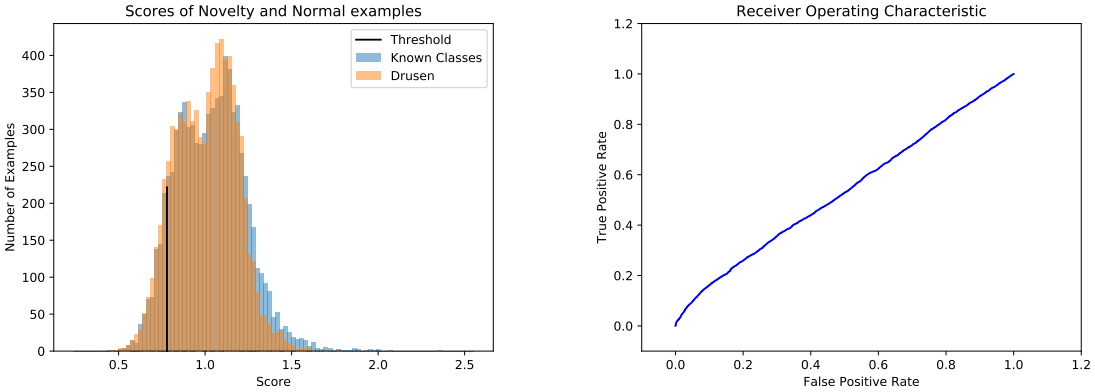
The performance metrics for the Thesis supervisor are displayed below in Table 5.8. The AUC is the same as the OpenMax supervisor and only 0.02 units worse than the Baseline. The same goes for the other metrics as well. They are all very similar to the OpenMax supervisor.

**Table 5.8:** Performance metrics for the Thesis supervisor on the test set in the CIFAR experiments.

| | AUC | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|---|
| Thesis supervisor | 0.74 | 0.71 | 0.32 | 0.44 | 0.60 |

### 5.7.5 Results for Retinal OCT

The 796 incorrectly classified Retinal OCT examples and a subsample of the same size of the correctly classified examples was used to extract the AUC values. The threshold was set to the $10^{th}$ percentile of the scores of the correctly classified examples from the training set: 0.78. The separation between Drusen and known classes in the test set is illustrated by Figure 5.60a and the corresponding ROC curve in Figure 5.60b. The histogram shows almost no separation, which is reflected in the ROC curve resembling the straight line.




**(a)** Scores. **(b)** ROC curve.

**Figure 5.60:** Histogram of scores assigned to known classes and Drusen and the ROC curve for the Thesis supervisor on the test set in the Retinal OCT experiments.

The false positives of the different classes relative to their occurrence are displayed below in Figure 5.61. Images of the class DME is most commonly classified as a novelty.

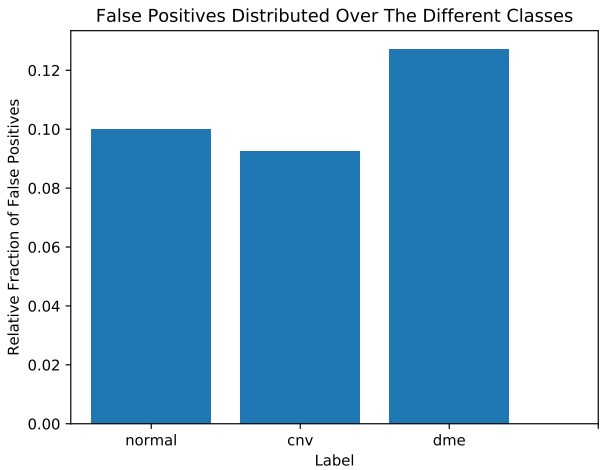



**Figure 5.61:** The false positives distributed over the 3 different classes relative to the class counts for the Thesis supervisor on the test set in the Retinal OCT experiments.

The performance metrics for the Thesis supervisor are displayed below in Table 5.9. As with many other supervisors the performance is poor and reduces to approximately randomly guessing whether a new input is a novelty or not. In comparison to the other supervisors the Thesis supervisor do however perform better than all of them in terms of all metrics, with the exception of the Baseline and OpenMax supervisors.

**Table 5.9:** Performance metrics for the Thesis supervisors on the test set in the Retinal OCT experiments.

| | AUC | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|---|
| Thesis supervisor | 0.56 | 0.53 | 0.11 | 0.18 | 0.51 |

### 5.7.6 Characteristics

As shown in the analysis of misclassifications, earlier layers prior to the logits do contain information about such occurrences. In specific, the score of the second to last layer achieved an AUC of 0.66, 0.82 and 0.72 separating correctly and incorrectly classified examples for the MNIST vs Omniglot, CIFAR and Retinal OCT scenarios, respectively. The results on the MNIST vs Omniglot experiments were, as for the other supervisors, very good. The Thesis supervisor did however show most difficulty in separating the number 1 from the novelties. The other supervisors did not experience the same issue. This means that the Thesis supervisor scores examples differently and that might be a complementary trait to other supervisors. For CIFAR, the Thesis supervisor performs almost identically to the OpenMax supervisor although the score distributions look differently. Again, the relative fraction of false positives differ. For the Retinal OCT experiment, the Thesis supervisor performs poorly relative to the OpenMax and Baseline supervisors, showing almost no separation at all. That is surprising due to the fact that the results for the other supervisors are similar over the Retinal OCT and CIFAR experiments. The reason for the under-performance may be that the accuracy for the Retinal OCT neural network is significantly higher than the CIFAR neural network. The supervisor might need a network that shows more uncertainty, even for the known classes, to be able to perform.

# 6
# Discussion

In this chapter, the methods and results of the thesis will be evaluated. Beyond major subjects, questions that have occurred during the project will be discussed in less details.

## 6.1 Background and Purpose

The thesis project was initiated as a part of the SMILE II project at Semcon. Part of the objective was to verify inputs sent to neural networks. Regardless of the SMILE II focus on autonomous driving, the thesis has found and evaluated supervisors on a mix of eligible scenarios consisting of different datasets. The findings are connected to the SMILE II project in terms of developing a supervisor able to detect novelties in CNNs and what key characteristics a supervisor should hold.

A thorough literature review has been made and presented in Chapter 3 of the thesis. Included are nine articles on novelty detection, nine on the defense against adversarial examples and five datasets. Therefore, a proper investigation of the current state of development has been made in accordance to the stated thesis objective. The review will be discussed in the following section.

## 6.2 Literature Review

In the search for related works to input verification, an associative search method was used. The method showed to be fruitful as 73 relevant articles were found in only two iterations. With that amount of articles, the most interesting ones were selected through two iterations of a screening process. The first iteration consisted of an evaluation of relevance through the reading of abstracts and the second iteration consisted of more thorough evaluation of algorithms and results. During the thesis process, discoveries of missed articles and articles published during the work have been found such as the publications by Ge et al. (2017), Hassen & Chan (2018) and Shu et al. (2018).

Since the thesis has, among other things, focused on using adversarial examples to gain knowledge about, or find boundaries to, the learned domain of a CNN, several papers in the literature review focus on the defense against such adversarials. While this thesis is not the first approach towards using adversarials to detect novelties,

see the work by Bendale & Boult (2016), the approach is still in the cradle.

## 6.3 Experiments

A collection of characteristics in the evaluated supervisors, found during the experiments, can be found in Section 5.6.

The results show that the adversarial hypothesis was true for the easier MNIST vs Omniglot experiments. However, for the CIFAR and Retinal OCT experiments, no such conclusion could be made, since only the Baseline and OpenMax supervisors achieved satisfying results. That means that the best performing supervisors were not the ones utilizing adversarials. The supervisors being trained on adversarials achieved almost no separation at all on the more complex datasets and often reduced to performance equal to randomly guessing if an input is a novelty or not.

Since the MNIST vs Omniglot scenario resulted in good results for every supervisor except NoveltyGAN, the impression is that the datasets are too simple in complexity to be used for evaluating supervisors. In future implementations towards input verification, the other scenarios, that are more complex, are more suitable. The supervisors implemented were modified and applied to different problems than originally designed for. Hence, it is hard to compare the results against the original papers. The supervisors using the adversarial examples all performed poorly on the two more complex datasets, addressing the issue or difficulty using adversarials as training data for novelty detection. To draw conclusions about the approaches, the supervisors use and the information they extract, it would be suitable to apply them without using adversarials. Instead, one should modify the underlying mechanisms to work only with the normal data provided to act as a traditional novelty detection algorithm. The best performing supervisors, OpenMax and Baseline, both used the final layer of the neural network, the logits, to classify novelties. The Baseline, which is a simple rule to produce a novelty score, works surprisingly well for all datasets. However, the baseline consistently classifies normal classes that have a low recall score in the network classifier as novelties across all experiments. The same pattern is not experienced for the OpenMax supervisor. Both supervisors are dependent on the network to display a certain pattern in the last layer which is different for novelties compared to normal examples. This trait cannot be guaranteed at this time but might be a problem to solve in future work. If the network can be trained to display these traits, a simple or more sophisticated supervisor can be used to accurately remove novelties from the deep learning system.

The thesis process has been limited to one laptop with a good graphics card until the fans broke and, luckily, the project was saved by the access to a high-end computer. As a consequence of the limitations in computational capacity, time has been of the essence as the implementations and experiments of the chosen supervisors took longer time than expected. The limitations have eventually led to delimitations in network depth and complexity, hence the smaller networks used in the experiments.

It is possible that deeper CNNs with more layers could open up to other supervisors, utilizing the increased complexity. Better computational capacity and time could also enable evaluations on datasets with larger images with more realistic scenarios. Additional resources would decrease the uncertainties in the results since there were not time enough to run the experiments multiple times and then average over the results. As of now, the results are achieved in a single run.

## 6.4 Development of A New Supervisor

A thorough motivation for the characteristics used in the development of the thesis supervisor is presented in the beginning of Section 5.7.

The analysis of the different networks show that attributes extracted from the different layers show different patterns depending on if the input was correctly or incorrectly classified. Using this fact, an attempt was made to utilize all layers of the network in the supervisor to perform novelty detection. The results were satisfying for the MNIST vs Omniglot and CIFAR experiments but not for the Retinal OCT experiment. As mentioned, the MNIST vs Omniglot results are hard to interpret as nearly all supervisors performed well. The difference in classification performance between the networks used for the other two datasets were significantly different. The CIFAR neural network achieved a test accuracy of 79% while the Retinal OCT neural network achieved 96%. This points towards the possibility that the supervisor might only be functional when the neural network itself is not completely certain of normal examples.

Depending on the problem, a simple supervisor that achieves decent results such as the Baseline supervisor is enough, but if higher performance is needed, a more thorough analysis of the neural network that is supposed to be supervised might be needed. As the results for the new supervisor shows, there might not be one best fit for all types of networks. It is important to investigate the network performance and in which layers patterns of uncertainties exist, then use them to extract meaningful attributes that separate novelties from normal examples.

# 7
# Conclusion

Connecting the literature review to the research questions, it can be concluded that the search for articles has resulted in a variety of algorithms with different approaches, making a good starting point for a possible future project on the same subject. As several different approaches were analyzed, a few promising and available algorithms were implemented, not only from the field of novelty detection but also from adversarial detection. Since the novelty space is infinite, adversarials were used as they are close to the decision boundary of the classes defined by the trained classifier. The results show, however, that adversarials do not provide the representation of novelties as hoped, but resulted in algorithms failing when presented with more complex and varied problems. As such, it is hard to judge the algorithms in the sense of the information they extract; the results might just be a consequence of the adversarials. A more thorough analysis where the algorithms can be trained either on the available normal data or something else, might provide valuable information.

The algorithms that achieved the best results across the scenarios are based on the final layer of the neural network being supervised. One of them being the Baseline supervisor which is a simple rule reflecting the networks uncertainty in predictions. The other being the OpenMax supervisor, also based upon the patterns in the last layer, but with a more sophisticated probabilistic approach. Both algorithms are very dependent on the network ability to show abnormalities in the final layer when receiving novelties. An undesirable trait since this ability cannot be guaranteed at this time. The simple Baseline is however very effective in comparison with its complexity and do provide a quick way to benchmark new algorithms and even use in a real system.

When developing the new supervisor, the characteristics were taken into account and not only the final layer was considered, but all of them. By extracting attributes from the layers, a separation was seen between incorrectly and correctly classified examples revealing that information can be found in earlier layers as well. While evaluating the supervisor, varying results were observed. The new supervisor performed similar to the two aforementioned supervisors in two scenarios while significantly worse in the other. Meaning that the information contained in layers, and how its represented, may vary with both network and problem, hence neural network-specific supervisors may prove to be needed.

# Bibliography


Abdi, H. & Williams, L. J. (2010), 'Principal component analysis', *Wiley Interdisciplinary Reviews: Computational Statistics* **2**(4), 433–459.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P. F., Schulman, J. & Mané, D. (2016), 'Concrete problems in AI safety', *CoRR* **abs/1606.06565**.
**URL:** *http://arxiv.org/abs/1606.06565*

Bendale, A. (2016), 'Osdn'. Accessed: 2018-04-24.
**URL:** *https://github.com/abhijitbendale/OSDN*

Bendale, A. & Boult, T. (2016), Towards open set deep networks, *in* 'Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on', IEEE.

Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J. & Zieba, K. (2016), 'End to end learning for self-driving cars', *CoRR* **abs/1604.07316**.
**URL:** *http://arxiv.org/abs/1604.07316*

Boudette, N. E. & Vlasic, B. (2017), 'Tesla self-driving system faulted by safety agency in crash - the new york times', `https://www.nytimes.com/2017/09/12/business/self-driving-cars.html`. (Accessed on 02/02/2018).

Chandola, V., Banerjee, A. & Kumar, V. (2009), 'Anomaly detection: A survey', *ACM computing surveys (CSUR)* **41**(3), 15.

Chen, J., Meng, Z., Sun, C., Tang, W. & Zhu, Y. (2017), 'Reabsnet: Detecting and revising adversarial examples', *arXiv preprint arXiv:1712.08250* .

Dau, H. A., Ciesielski, V. & Song, A. (2014), Anomaly detection using replicator neural networks trained on examples of one class, *in* 'Asia-Pacific Conference on Simulated Evolution and Learning', Springer, pp. 311–322.

Erfani, S. M., Rajasegarar, S., Karunasekera, S. & Leckie, C. (2016), 'High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning', *Pattern Recognition* **58**, 121–134.

Fawzi, A., Moosavi-Dezfooli, S., Frossard, P. & Soatto, S. (2017), 'Classification regions of deep neural networks', *CoRR* **abs/1705.09552**.
**URL:** *http://arxiv.org/abs/1705.09552*

Feinman, R. (2018), 'detecting-adversarial-samples'. Accessed: 2018-04-26.
**URL:** *https://github.com/rfeinman/detecting-adversarial-samples*

Feinman, R., Curtin, R. R., Shintre, S. & Gardner, A. B. (2017), 'Detecting adversarial samples from artifacts', *arXiv preprint arXiv:1703.00410* .

Gal, Y. & Ghahramani, Z. (2015), 'Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning', *ArXiv e-prints* .

Ge, Z., Demyanov, S., Chen, Z. & Garnavi, R. (2017), 'Generative openmax for multi-class open set classification', *CoRR* **abs/1707.07418**.
**URL:** *http://arxiv.org/abs/1707.07418*

*Getting Started | TensorFlow* (2018). Accessed: 2018-04-20.
**URL:** *https://www.tensorflow.org/get_started/*

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep Learning*, MIT Press. `http://www.deeplearningbook.org`.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014), Generative adversarial nets, *in* 'Advances in neural information processing systems', pp. 2672–2680.

Grosse, K., Manoharan, P., Papernot, N., Backes, M. & McDaniel, P. D. (2017), 'On the (statistical) detection of adversarial examples', *CoRR* **abs/1702.06280**.
**URL:** *http://arxiv.org/abs/1702.06280*

Grother, P. J. (1995), 'Nist special database 19', *Handprinted forms and characters database, National Institute of Standards and Technology* .

Hassen, M. & Chan, P. K. (2018), 'Learning a neural-network-based representation for open set recognition', *CoRR* **abs/1802.04365**.
**URL:** *http://arxiv.org/abs/1802.04365*

He, K., Zhang, X., Ren, S. & Sun, J. (2015), 'Deep residual learning for image recognition', *CoRR* **abs/1512.03385**.
**URL:** *http://arxiv.org/abs/1512.03385*

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2012*a*), 'Improving neural networks by preventing co-adaptation of feature detectors', *CoRR* **abs/1207.0580**.
**URL:** *http://arxiv.org/abs/1207.0580*

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. R. (2012*b*), 'Improving neural networks by preventing co-adaptation of feature detectors', *arXiv preprint arXiv:1207.0580* .

Huang, X., Kwiatkowska, M., Wang, S. & Wu, M. (2016), 'Safety verification of deep neural networks', *CoRR* **abs/1610.06940**.
**URL:** *http://arxiv.org/abs/1610.06940*

Ioffe, S. & Szegedy, C. (2015), 'Batch normalization: Accelerating deep network training by reducing internal covariate shift', *CoRR* **abs/1502.03167**.
**URL:** *http://arxiv.org/abs/1502.03167*

Kermany, D. S., Goldbaum, M., Cai, W., Valentim, C. C., Liang, H., Baxter, S. L., McKeown, A., Yang, G., Wu, X., Yan, F. et al. (2018), 'Identifying medical diagnoses and treatable diseases by image-based deep learning', *Cell* **172**(5), 1122–1131.
**URL:** *http://dx.doi.org/10.17632/rscbjbr9sj.2*

Kingma, D. P. & Ba, J. (2014), 'Adam: A method for stochastic optimization', *CoRR* **abs/1412.6980**.
**URL:** *http://arxiv.org/abs/1412.6980*

Krizhevsky, A. & Hinton, G. (2009), 'Learning multiple layers of features from tiny images'.

Kurakin, A., Goodfellow, I. J. & Bengio, S. (2016), 'Adversarial examples in the physical world', *CoRR* **abs/1607.02533**.
**URL:** *http://arxiv.org/abs/1607.02533*

Lake, B. M., Salakhutdinov, R. & Tenenbaum, J. B. (2015), 'Human-level concept learning through probabilistic program induction', *Science* **350**(6266), 1332–1338.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998), 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE* **86**(11), 2278–2324.

Li, X. & Li, F. (2016), 'Adversarial examples detection in deep networks with convolutional filter statistics', *CoRR* **abs/1612.07767**.
**URL:** *http://arxiv.org/abs/1612.07767*

Liu, D. C. & Nocedal, J. (1989), 'On the limited memory bfgs method for large scale optimization', *Mathematical programming* **45**(1-3), 503–528.

Liu, Y., Cukic, B. & Gururajan, S. (2007), 'Validating neural network-based online adaptive systems: A case study', *Software Quality Journal* **15**(3), 309–326.

Lu, J., Issaranon, T. & Forsyth, D. A. (2017), 'Safetynet: Detecting and rejecting adversarial examples robustly', *CoRR* **abs/1704.00103**.
**URL:** *http://arxiv.org/abs/1704.00103*

Lucas Deecke, Robert Vandermeulen, L. R. S. M. M. K. (2018), 'Anomaly detection with generative adversarial networks'.
**URL:** *https://openreview.net/forum?id=S1EfylZ0Z*

Metzen, J. H., Genewein, T., Fischer, V. & Bischoff, B. (2017), 'On detecting adversarial perturbations', *arXiv preprint arXiv:1702.04267* .

Moosavi-Dezfooli, S., Fawzi, A. & Frossard, P. (2015), 'Deepfool: a simple and accurate method to fool deep neural networks', *CoRR* **abs/1511.04599**.
**URL:** *http://arxiv.org/abs/1511.04599*

Murphy, K. P. (2014), *Machine learning: a probabilistic perspective*, MIT Press, Cambridge, MA.

Papernot, N., McDaniel, P. D., Jha, S., Fredrikson, M., Celik, Z. B. & Swami, A. (2015), 'The limitations of deep learning in adversarial settings', *CoRR* **abs/1511.07528**.
**URL:** *http://arxiv.org/abs/1511.07528*

Paultimothymooney (2018), 'Retinal oct images (optical coherence tomography)'. Accessed: 2018-05-09.
**URL:** *https://www.kaggle.com/paultimothymooney/kermany2018*

Pimentel, M. A., Clifton, D. A., Clifton, L. & Tarassenko, L. (2014), 'A review of novelty detection', *Signal Processing* **99**, 215–249.

Radford, A., Metz, L. & Chintala, S. (2015), 'Unsupervised representation learning with deep convolutional generative adversarial networks', *CoRR* **abs/1511.06434**.
**URL:** *http://arxiv.org/abs/1511.06434*

Richter, C. & Roy, N. (2017), Safe visual navigation via deep learning and novelty detection, *in* 'Robotics: Science and Systems XIII, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, July 12-16, 2017'.
**URL:** *http://www.roboticsproceedings.org/rss13/p64.html*

Sakurada, M. & Yairi, T. (2014), Anomaly detection using autoencoders with nonlinear dimensionality reduction, *in* 'Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis', ACM, p. 4.

Scheirer, W. J., de Rezende Rocha, A., Sapkota, A. & Boult, T. E. (2013), 'Toward open set recognition', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **35**(7), 1757–1772.

Scheirer, W. J., Rocha, A., Michaels, R. & Boult, T. E. (2011), 'Meta-recognition: The theory and practice of recognition score analysis', *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* **33**, 1689–1695.

Scheirer, W. J., Rocha, A., Micheals, R. J. & Boult, T. E. (2011), 'Meta-recognition: The theory and practice of recognition score analysis', *IEEE transactions on pattern analysis and machine intelligence* **33**(8), 1689–1695.

Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U. & Langs, G. (2017), 'Unsupervised anomaly detection with generative adversarial networks to guide marker discovery', *CoRR* **abs/1703.05921**.
**URL:** *http://arxiv.org/abs/1703.05921*

Shu, L., Xu, H. & Liu, B. (2018), 'Unseen class discovery in open-world classification', *CoRR* **abs/1801.05609**.
**URL:** *http://arxiv.org/abs/1801.05609*

Simonyan, K. & Zisserman, A. (2014), 'Very deep convolutional networks for large-scale image recognition', *CoRR* **abs/1409.1556**.
**URL:** *http://arxiv.org/abs/1409.1556*

Steinwart, I., Christmann, A. & (e-book collection), S. (2008), *Support vector machines*, 1st;1. aufl.; edn, Springer, New York.

Subramanya, A., Srinivas, S. & Babu, R. V. (2017), 'Confidence estimation in deep neural networks via density modelling', *CoRR* **abs/1707.07013**.
**URL:** *http://arxiv.org/abs/1707.07013*

Swanson, E. A. & Fujimoto, J. G. (2017), 'The ecosystem that powered the translation of oct from fundamental research to clinical and commercial impact (invited)', *Biomed. Opt. Express* **8**(3), 1638–1664.
**URL:** *http://www.osapublishing.org/boe/abstract.cfm?URI=boe-8-3-1638*

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J. & Fergus, R. (2013), 'Intriguing properties of neural networks', *CoRR* **abs/1312.6199**.
**URL:** *http://arxiv.org/abs/1312.6199*

Ting, D. S. W., Cheung, C. Y., Lim, G., Tan, G. S. W., Quang, N. D., Gan, A., Hamzah, H., Garcia-Franco, R., San Yeo, I. Y., Lee, S. Y., Wong, E. Y. M., Sabanayagam, C., Baskaran, M., Ibrahim, F., Tan, N. C., Finkelstein, E. A., Lamoureux, E. L., Wong, I. Y., Bressler, N. M., Sivaprasad, S., Varma, R., Jonas, J. B., He, M. G., Cheng, C.-Y., Cheung, G. C. M., Aung, T., Hsu, W., Lee, M. L. & Wong, T. Y. (2017), 'Development and validation of a deep learning system for diabetic retinopathy and related eye diseases using retinal images from multiethnic populations with diabetes', *JAMA* **318**(22), 2211–2223.

Wang, W., Wang, A., Tamar, A., Chen, X. & Abbeel, P. (2017), 'Safer classification by synthesis', *CoRR* **abs/1711.08534**.
**URL:** *http://arxiv.org/abs/1711.08534*

Xu, W., Evans, D. & Qi, Y. (2017), 'Feature squeezing: Detecting adversarial examples in deep neural networks', *CoRR* **abs/1704.01155**.
**URL:** *http://arxiv.org/abs/1704.01155*

Zheng, S., Song, Y., Leung, T. & Goodfellow, I. (2016), Improving the robustness of deep neural networks via stability training, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 4480–4488.