CHALMERS
UNIVERSITY OF TECHNOLOGY

Novel Scenario Detection in Road Traffic Images

A Comparative Evaluation of Novelty Detection Algorithms

Master's thesis in Complex Adaptive Systems

ERIK KRATZ

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Novel Scenario Detection in Road Traffic Images

A Comparative Evaluation of Novelty Detection Algorithms

ERIK KRATZ

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Communication and Antenna Systems
Communication Systems group
Chalmers University of Technology
Gothenburg, Sweden 2019

Novel Scenario Detection in Road Traffic Images
A Comparative Evaluation of Novelty Detection Algorithms
ERIK KRATZ

Supervisor: Roman Sokolovskii, Department of Electrical Engineering
Supervisor: Cristofer Englund, RISE Viktoria
Examiner: Giuseppe Durisi, Department of Electrical Engineering

Department of Electrical Engineering
Division of Communication and Antenna Systems
Communication Systems group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Images used for novelty detection experiments. Top row: samples from a simulator-generated dataset. Bottom row: samples from videos captured in road traffic.

Typeset in LaTeX
Gothenburg, Sweden 2019

Novel Scenario Detection in Road Traffic Images
A Comparative Evaluation of Novelty Detection Algorithms
Erik Kratz
Department of Electrical Engineering
Chalmers University of Technology

Abstract

For artificial neural networks to be deployed in safety critical applications, such as autonomous driving, there is a need for reliable detection and rejection of unfamiliar inputs, because of the black box nature of such algorithms. This thesis compares the performance of three recently published convolutional autoencoder-based novelty detection algorithms when applied to road traffic images. The algorithms were reimplemented for high-resolution images, and tested for detecting two types of previously unseen scenarios: unseen weather conditions and unseen type of landscape. Each use case was represented in two datasets: one simulated dataset with low scene variation, and one dataset captured in real road-traffic. Classification results in terms of area under receiver operating characteristic and area under precision-recall curve show that for low variability in the normal scenario, novelties can be reliably detected with two out of three approaches. For the real image dataset, performance is consistently lower, indicating that more complex and/or more well tuned models are needed for use in real-world applications.

Keywords: novelty detection, outlier detection, anomaly detection, artifical neural networks, machine learning, verification

Acknowledgements

First and foremost, I would like to thank Cristofer Englund and Boris Duran at RISE Viktoria and my supervisor Roman Sokolovskii at the Department of Electrical Engineering, for helping and guiding me throughout the entirety of this project. A thanks also goes out to Dr. Giuseppe Durisi at the Department of Electrical Engineering, for taking on the role of examiner and for reassuring me that I was on the right track during the most difficult part of the project.

I also want to thank everyone at RISE Viktoria and the participants of SMILE II, for giving me the opportunity to work on this exciting project. It has been both fun and challenging.

A final thanks to my friends at Chalmers and my family. Without you, my time at Chalmers would not have been the same, including this thesis project.

Erik Kratz, Gothenburg, March 2019

Contents

List of Figures

List of Tables

Abbreviations

AAE	adversarial autoencoder
AE	autoencoder
ALOCC	adversarially learned one-class classifier for novelty detection
ANN	artificial neural network
AUPRC	area under precision-recall curve
AUROC	area under receiver operating characteristic
BCE	binary cross entropy
CAE	convolutional autoencoder
CNN	convolutional neural network
DNN	deep neural network
DSVDD	deep support vector data-description
FFNN	feed-forward neural network
FN	false negatives
FP	false positives
FPR	false positive rate
GPND	generative probabilistic novelty detection
ML	machine learning
MLP	multilayer perceptron
MNIST	Modified National Institute of Standards and Technology
MSE	mean squared error
ND	novelty detection
OCC	one-class classification
PRC	precision-recall curve
ReLU	rectified linear unit

ROC	receiver operating characteristic
SMILE II	Safety analysis and verification/validation of MachIne LEarning based systems
SVDD	support vector data-description
TN	true negatives
TP	true positives
TPR	true positive rate
VAE	variational autoencoder

1 Introduction

1.1 Background

Artificial neural networks (ANNs) have in recent years become the state-of-the-art within pattern recognition and classification. They have successfully been used for tasks such as image classification [1, 2, 3], speech recognition [4], and even generation of realistic images from drawings [5].

The potential of finding complex mappings between input data, such as images from a front-facing vehicle camera, and desired outputs make ANNs, and specifically deep neural networks (DNNs), promising for use in autonomous driving applications. Examples of existing results include learning to output the vehicle steering angle [6] or a set of currently feasible actions [7] based on raw image input.

The power of DNNs lies in the high number of connections. However, the very same property also means that the flow of information inside a DNN is difficult to follow. This black box nature of DNNs is a severe drawback. Even networks that achieve high accuracy, in both training and testing, can misclassify inputs, sometimes with high confidence, and knowing what causes the errors is often practically impossible. This unreliability effectively disqualifies DNNs for use in safety-critical situations, such as autonomous driving, where an undetected misclassification can have dire conseqeunces.

Misclassifications can occur for a number of reasons, including the DNN being overfitted to training data and the DNN being poorly trained. Another reason are the recently discovered adversarial examples: small carefully designed perturbations of an input that would be classified correctly, which result in a severe misclassification. A situation where misclassifications will always occur is when there is a discrepancy between the distribution of the training data and the distribution of the test data. A property of DNN classifiers is that they will classify any input into one of the classes seen during training, simply because this is what they were designed to do. A simple example can be seen in Fig. 1.1, which shows a set of images with corresponding classifications generated by a classifier trained on the CIFAR-10 dataset [8]. The ground truth for all images is a type of novelty, i.e., an object class not present in CIFAR-10. For each of these inputs, the network provides high confidence classifications in one of the CIFAR-10 classes.

The preferred behaviour of the above classifier, if integrated in a safety-critical system, would be to detect the example images as unknown and subsequently pass control to another part of the system, e.g., a default safe behaviour which does not require knowledge of the image content. The research project Safety analysis and verification/validation of MachIne LEarning based systems (SMILE II) [9] aims to

(a) A lamb, classified as a bird with probability 0.995.

(b) A goldfish, classified as an airplane with probability 0.992.

(c) Some flowers, classified as a frog with probability 0.999.

Figure 1.1: Examples of high confidence misclassification of previously unseen objects.

solve this problem for the case of vehicular perception sensor inputs, and is led by the research institute RISE Viktoria. This thesis is conducted as a part of SMILE II, and will focus on the novelty detection (ND) step, where inputs that are not part of the training data distribution are detected and rejected.

1.2 Problem Description

The problem addressed in this thesis is that of performing ND in sets of raw pixel images from on-road traffic. More specifically, the goal is finding one or several algorithms capable of modelling a normal scenario class consisting of a set of images from a front-facing vehicle camera driving in a certain type of conditions. The objective during testing is to detect images taken in other driving conditions and mark them as novel, while still accepting normal images. A successful ND algorithm effectively becomes a safety cage for any DNN built for application using the normal class dataset, since the ND algorithm removes inputs not seen during training of the DNN. This would make the DNN more suited for use in safety-critical applications.

1.2.1 Aim

The aim of this project is to identify a set of existing state-of-the-art novelty detection algorithms that perform well on front camera input to self-driving systems. Algorithms were reimplemented, evaluated, and compared using relevant classification evaluation measures.

1.2.2 Scope and Delimitations

The following delimitations serve to further limit the scope of the project:

- For selected algorithms to be extendable to work with any type of sensor input and with large amounts of data, we demand that they are unsupervised: i.e., given a set of training inputs, there should not be any requirement for further annotation, such as subclasses or object labels.
- ND will be performed on single image frames. Algorithms analyzing video sequences are not considered in this thesis.

- Whole, raw pixel images will be used as inputs, so as to learn to detect an entire scenario as either normal or novel. No object detection or other type of image segmentation will be performed explicitly.
- Editing of selected algorithms is limited to making them compatible with the chosen dataset(s) and tuning of existing parameters and settings.
- When possible, open source code will be used.
- Hardware requirements for testing in real-time will not be taken into account when selecting algorithms for evaluation. However, these properties may be discussed when comparing the selected algorithms after evaluation.

1.3 Social and Ethical Aspects

This section reflects on possible negative social and ethical impacts of the completion of this thesis project.

1.3.1 Implementation

Our view is that there are no ethical or social problems related with the implementation of this project. The main activities will be a literature review and computer simulations, neither of which is going to have any unwanted impact on other people.

1.3.2 Outcome

However large or small the contribution, the goal of this project is to help in achieving safe autonomous driving applications. This goal likely would have a big impact on society if realized; one negative impact is the loss of employment for professional drivers. However, this issue is clearly outweighed by the expected benefits of increased traffic safety and reduced greenhouse gas emissions, making the project worth implementing.

1.4 Thesis Outline

First, theory relevant for understanding the problem and the proposed solutions is presented in Section 2. Then, the literature review conducted to find relevant self-driving image datasets and state-of-the-art ND algorithms is presented in Section 3. The reimplementation of selected algorithms and experimental results are presented in Section 4. The experimental results are then discussed in Section 5. Finally, the main conclusions of the thesis project are given in Section 6.

2
Theory

This chapter presents the basic theoretical concepts needed to understand the methodology and experiments in this thesis.

2.1 One-Class Classification

ND is often approached as a one-class classification (OCC) problem. In OCC, the aim is to determine whether an input belongs to the normal class or not. More formally, for a given point $x \in \mathbb{R}^n$, the goal is to determine if $x \in A \subset \mathbb{R}^n$, or if $x \in \mathbb{R}^n \setminus A$, where the only available information is a set of N known members of A, $\hat{A} = \{a_1, \ldots, a_n\} \subset A$. This is different from binary classification and multiclass classification, where all known classes have samples, and every possible input is assumed to belong to one of the known classes. The difference is illustrated in Fig. 2.1. In binary classification, it is enough to find a boundary separating the closest samples of different classes to separate all samples. In one-class classification there is no information about points outside the normal class, which makes it harder to find a suitable decision boundary.

2.2 Support Vector Data Description

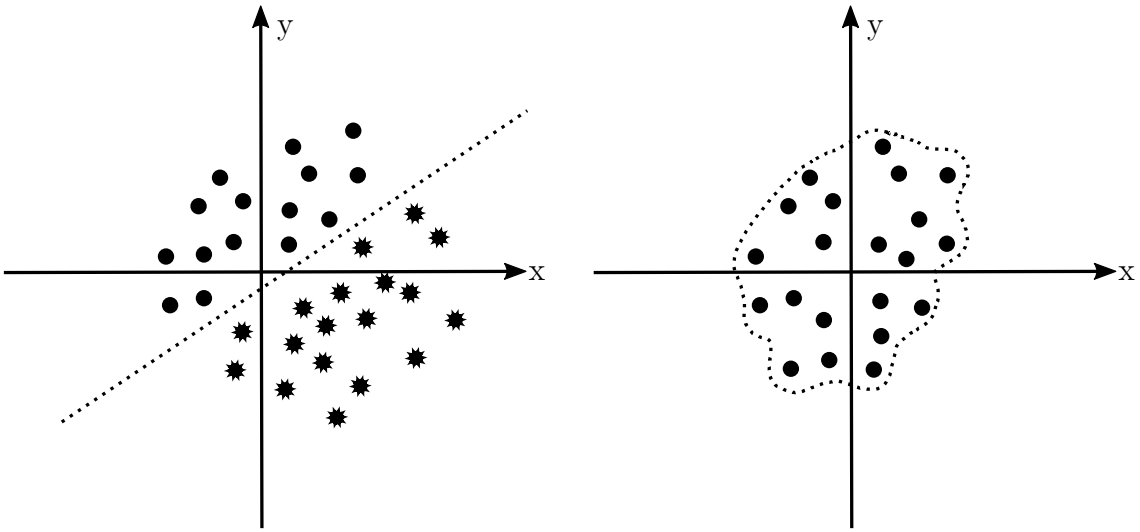
Support vector data-description (SVDD) [10] is an OCC algorithm for describing a set of n-dimensional column vector inputs $\mathbf{x_i} \in \mathbb{R}^n$, $i = 1, \ldots, N$. The aim is to find a hypersphere of minimum radius R enclosing feature space points $\Phi(\mathbf{x_i})$, where Φ is some mapping from the input space $\mathbb{R}^n$ to a feature space. In the simplest case, Φ is the identity mapping, so that $\Phi(\mathbf{x_i}) = \mathbf{x_i}$, but to obtain more flexible solutions, other functions can be used, e.g., kernel functions mapping inputs into a feature space of higher dimension $m > n$ [10]. For a given Φ, the SVDD objective can be defined as

$$\min_{R,\mathbf{c},\xi} \quad R^2 + \frac{1}{\nu N} \sum_{i=1}^{N} \xi_i, \tag{2.1}$$

with constraints

$$||\Phi(\mathbf{x_i}) - \mathbf{c}||^2 \leq R^2 + \xi_i, \qquad \xi_i \geq 0, \qquad i = 1, \ldots, N. \tag{2.2}$$

The variables ξ_i enable a soft boundary such that all feature points need not lie within distance R from the hypersphere center $\mathbf{c}$, and the parameter ν determines the trade-off between minimizing R and ξ_i. The above notation is the same as in [11], which is further elaborated on in Section 3.2.2.



(a) Example of a binary classification problem. The straight boundary separating the two well sampled classes is relatively easy to model.

(b) Example of an OCC problem. Since only the normal class is well sampled, it is relatively hard to model an optimal decision boundary.

Figure 2.1: Two dimensional example illustrating the difference between an OCC problem and a binary classification problem.

2.3 Artifical Neural Networks

An ANN is, as the name suggests, a network where the basic unit is an artificial version of the biological neuron. Each artificial neuron can be connected to other neurons via both incoming and outgoing connections. These connections correspond to the synapses in the brain. The basic action of a neuron is to:

1. Receive input signals from other neurons through incoming connections.
2. Compute an output signal, called activation, based on the inputs.
3. Send the activation signal to other neurons through outgoing connections.

A feed-forward neural network (FFNN) is an ANN where information flows one way: from the input in one end to the output in the other end. A common type of FFNN is the multilayer perceptron (MLP). In a MLP, the input signal is propagated through one layer of neurons at a time. Neurons in a given layer have incoming connections only from neurons in the previous layer and outgoing connections only to the subsequent layer. The activation $x_j^{(l)}$ of neuron j in layer l is determined as

$$x_j^{(l)} = g\left(\sum_i w_{ij}^{(l)} x_i^{(l-1)} + b_j^{(l)}\right), \tag{2.3}$$

that is, a weighted sum of the previous layer activations $x_i^{(l-1)}$, with weights $w_{ij}^{(l)}$ and bias $b_j^{(l)}$. The function g is called the activation function and is normally chosen depending on the type of network and layer, see Section 2.3.1. The bias can also be included as an extra neuron with constant activation equal to 1, and can therefore be excluded without loss of generality. In a MLP, the nodes representing the input signal are called the input layer, the nodes representing the output signal are called the output layer, while all intermediate layers are called hidden layers. An example

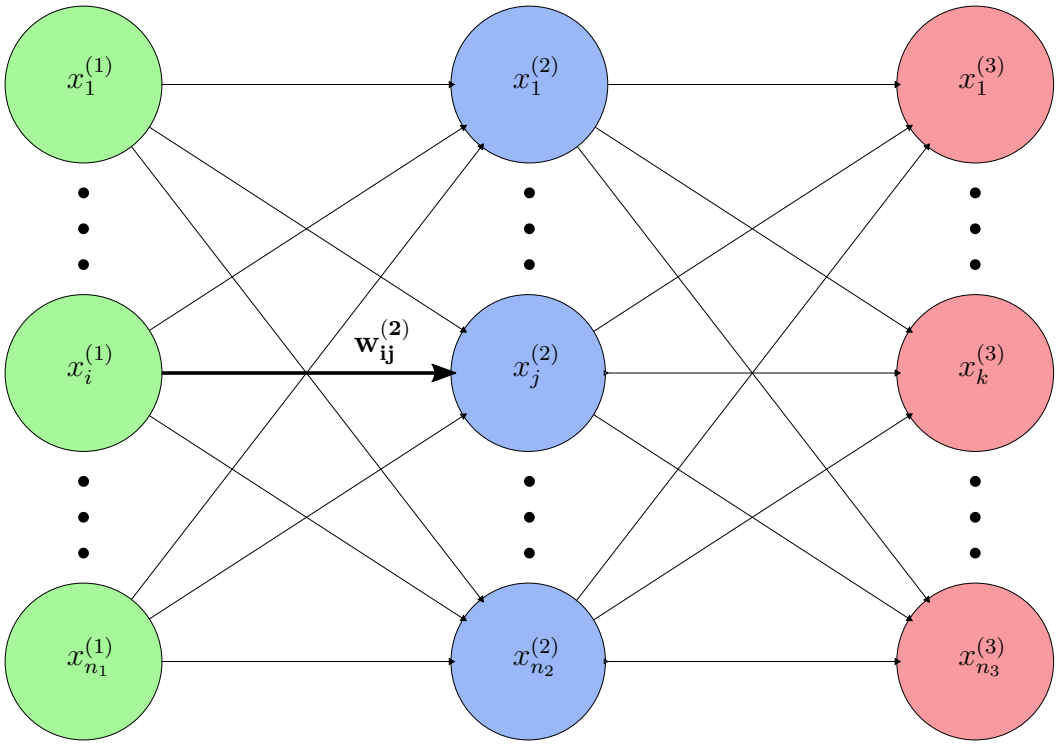


Figure 2.2: Example of a MLP with one hidden layer. The nodes represent neurons and the arrows represent connections, or weights.

of a MLP with n_1-dimensional input, a single hidden layer with n_2 neurons, and n_3-dimensional output can be seen in Fig. 2.2.

2.3.1 Activation Functions

Below are some common activation functions for ANNs. They are also plotted in Fig. 2.3.

Sigmoid

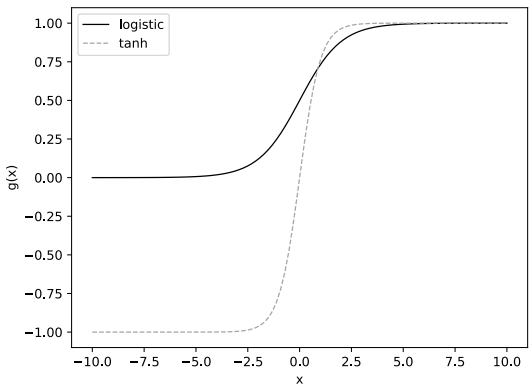
Sigmoid functions are defined by their S-shape, and are monotonically increasing functions between two real values. Often the output range is $(-1, 1)$ or $(0, 1)$. Examples are the logistic function,

$$g(x) = \frac{1}{1 + e^{-x}} \in (0, 1), \qquad \forall x \in (-\infty, \infty), \tag{2.4}$$

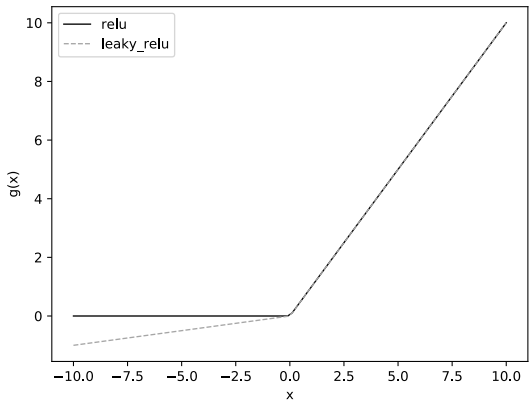
and hyperbolic tangent function,

$$g(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \in (-1, 1), \qquad \forall x \in (-\infty, \infty). \tag{2.5}$$

Because of its monotonic mapping and $(0, 1)$ output range, the logistic function (2.4) is normally used in layers where the output is interpreted as a probability.



(a) Plots of the logistic function and the hyperbolic tangent function.



(b) Plots of the ReLU function and the leaky ReLU function.

Figure 2.3: Plots of various ANN layer activation functions.

ReLU

The rectified linear unit (ReLU) activation is defined by

$$g(x) = \max(0, x), \tag{2.6}$$

and is very popular in deep learning applications since it has been demonstrated empirically [12] to lead to faster learning compared with traditional activations, such as sigmoids. A possible reason for this is that ReLU combines non-linearity with being computationally inexpensive. With ReLU, the problem of vanishing and exploding gradients is also avoided, since the derivative is always 1 for $x > 0$. A known problem with the ReLU activation is that neurons become permanently inactive if they at some point get negative input for all training set samples. Since their gradient is 0 for $x < 0$, ReLU neurons cannot recover in such situations and will output 0 indefinitely. This is known as the dying ReLU problem.

2.3.1.1 Leaky ReLU

The leaky ReLU function is defined by

$$g(x) = \begin{cases} x, & x \geq 0 \\ \alpha x, & x < 0, \end{cases} \tag{2.7}$$

where $\alpha > 0$ is a small number. Leaky ReLU is an option used to avoid the dying ReLU problem, since its gradient is nonzero also for $x < 0$.

Softmax

The softmax function is a generalization of the logistic function (2.4), outputting a vector of numbers in the range $(0, 1)$. It is defined by

$$g_j(\mathbf{x}) = \frac{e^{x_j}}{\sum_{i=1}^{N} e^{x_i}}. \tag{2.8}$$

which by definition yields g_j such that $g_j > 0$, $j = 1, \ldots, N$ and $\sum_{j=1}^{N} g_j = 1$. The softmax layer is commonly used for categorical probability distributions as the final output of multi-class classifiers, where the output g_j signifies the probability that the input belongs to class j.

2.3.2 Loss Function

The loss function is the optimization objective during training of an ANN. It is normally a measure of the difference between the output prediction p of the ANN and a corresponding target output y and is preferably differentiable w.r.t. the ANN's weights and biases. This way, ANN optimization can be performed by minimizing the loss function using, e.g., gradient descent. The loss functions used in this thesis are binary cross entropy (BCE),

$$\text{BCE}(y, p) = -y \log p - (1 - y) \log (1 - p) \tag{2.9}$$

and mean squared error (MSE),

$$\text{MSE}(\mathbf{y}, \mathbf{p}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - p_i)^2, \tag{2.10}$$

where the MSE is computed over a mini-batch of n predictions $\mathbf{p} = p_1, \ldots, p_n$ and corresponding targets $\mathbf{y} = y_1, \ldots, y_n$. Both functions are differentiable everywhere and have a global minimum at $y = p$, which make them suitable for regression. Plots for both BCE and MSE for a fixed $y = 0.5$ can be seen in Fig. 2.4.

2.3.3 Backpropagation

Backpropagation refers to the process of computing the loss in the output layer and then using this error to compute a similar loss for the previous layer. This process is then repeated until the input layer is reached. The parameters of all layers can hence be updated by propagating the output loss backwards through the ANN, yielding the name backpropagation.

2.3.4 Mini-Batch Training

Mini-batch training refers to processing a subset, called a mini-batch, of the training data between each update of the ANN parameters. For each input sample in a mini-batch, the loss and corresponding parameter update is computed separately. The parameters are then updated just once per mini-batch, using the sum of all parameter updates computed for the current mini-batch. The word batch refers to the entire set of training inputs, also called the training set.

2.3.5 Epoch

In machine learning, one training epoch is said to have passed each time all input samples in the training set have been processed one time and the ANN parameters have been updated according to the used optimization objective.

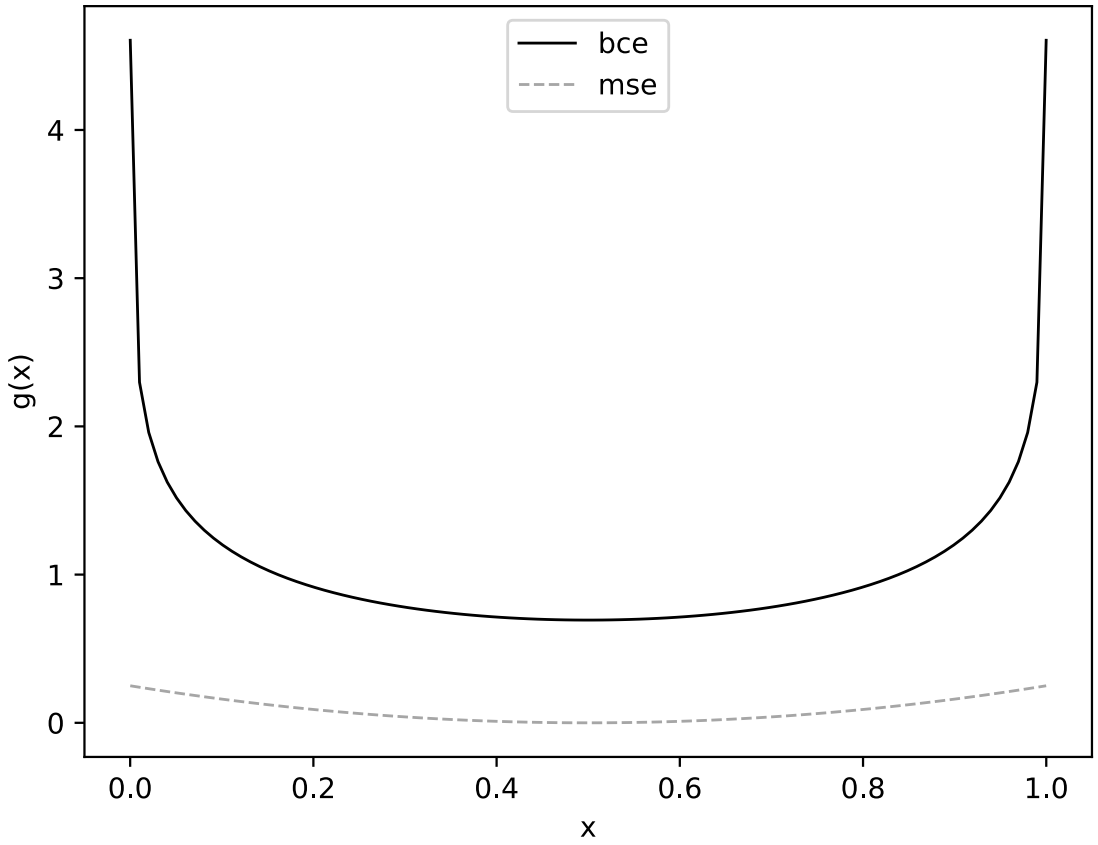
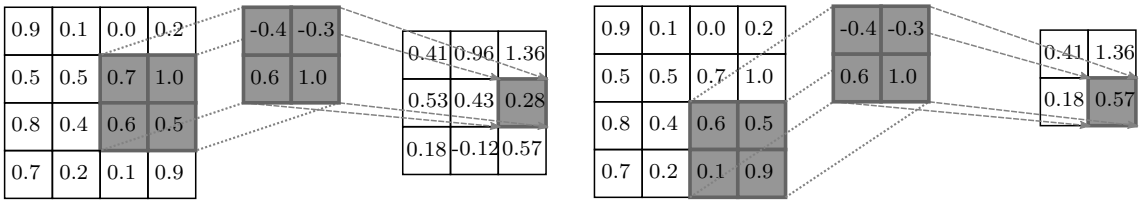


Figure 2.4: BCE and MSE plotted as functions of p, for a fixed $y = 0.5$. For the MSE function, this is the special case where $n = 1$.



(a) Convolution with stride $s = 1$ in both dimensions.

(b) Convolution with stride $s = 2$ in both dimensions.

Figure 2.5: Two examples of a two-dimensional convolutional filter with filter size $k = 2$ in both dimensions, applied to an input image of size 4×4. The filter is multiplied element-wise with different sections of the input image and the sum of the products is assigned to the corresponding position of the output map. The output size is determined not only by the input and filter shape but also the stride s, which is different for the two examples.

2.4 Convolutional Neural Networks

Digital image inputs are represented with one or more scalar values per pixel (usually three for color images). This means the input dimensionality is high, even for relatively small images. In a MLP, there are $(n + 1)m$ weights and biases between an input of dimension n and a subsequent layer of dimension m, which leads to a large number of parameters for MLPs built for processing images. Moreover, with unique weights for each pixel, the features learned by the MLP are local, meaning that objects seen during training will only be recognized if they are in the same part of the input image as they were in the corresponding training sample(s). These problems are either reduced or avoided with convolutional neural networks (CNNs), a type of FFNNs which use convolutional layers as part of the network. Below, the different parts of a typical CNN are described. In addition to CNNs, they are also applicable in similar types of networks, such as convolutional autoencoders (CAEs), described in Section 2.5.1.

2.4.1 Convolutional Layers

In a convolutional layer, a set of $k \times k$ filters is cross-correlated with the input image. Cross-correlation means sliding the filter across the input in steps of size s and performing element-wise multiplication between the filter and the input points currently covered by the filter. For each filter position, the k^2 products are added to yield a single data point in the output. The result is a new image, called an activation map or a feature map. The activation map has high pixel values in areas where the input is similar to the filter and low values where it is not. This way, a filter works as a feature extractor. Typically, a number of filters is used in each layer, to extract different features. The number of filters is equal to the number of output maps, and is called the depth of the layer. The shape of the output maps depend on the input shape, the filter size k, the stride s and the dilation d. The stride s is the step size used when moving filters across the image and can be different in different dimensions. The dilation d determines the spacing between the pixels the filter is applied to. The difference between $s = 1$ and $s = 2$ can be seen in Fig. 2.5. Setting $s > 1$ or $d > 1$ is a way of reducing the size of the output image, thereby performing dimensionality reduction.

2.4.2 Fully Connected Layers

Fully connected layers, also called dense layers, are equivalent to layers in a MLP. Each neuron in the input is connected to each neuron in the output. Fully connected layers have the advantage of being able to model complex mappings, and are thus often used after feature extraction, as a mapping from high level features to the desired output shape, e.g., a class prediction. The drawback with fully connected layers, compared with convolutional layers, is the high number of parameters (i.e., weights and biases) which can make them inconvenient and difficult to train.

2.4.3 Padding

Padding is the process modifying the shape of an image by adding pixels to it. Padding is normally applied before or after convolutional layers, since their output shape can not be explicitly specified.

2.4.4 Max Pooling

Pooling is used for dimensionality reduction, often after convolutional layers. A cluster of neurons in the input is represented as one neuron in the output. In max pooling, the maximum value of the cluster is chosen.

2.4.5 Dropout

In a dropout layer, each neuron is deactivated, meaning it outputs zero activation, with some nonzero probability. This means the total layer activation becomes less dependant on the outputs of specific neurons, which improves network robustness.

2.4.6 Batch Normalization

By using a batch normalization [13] layer, the activation of the preceding ANN layer, which is the input to the batch normalization layer, is normalized by having its mean and variance kept fixed during training. The mean and variance is computed over all samples in the batch, but independently for each point in the input. The details of the normalization process will not be covered here, however they can be found in the original paper [13].

The purpose of batch normalization is to reduce internal covariate shift, which is when the probability density functions of inputs to the intermediate layers of a DNN change during training, as the network parameters change. When the distribution of a layer activation changes, the learning is slowed down, since the subsequent layer has to learn to match a new input distribution to the training targets. Through batch normalization, internal covariate shift is reduced, which in turn reduces the total training time. The vanishing gradient problem, which refers to when the gradient of the loss function becomes so small that learning is effectively stopped, is also eliminated when using batch normalization. For common activation functions,

such as sigmoids, the gradient vanishes for inputs far from zero, something which normalizing the inputs to zero mean and unit variance counteracts.

2.4.7 Typical Convolutional Neural Network Architecture

In a typical CNN classifier, the above layers are combined in two parts. The first is a feature extraction part: this consists of a set of convolutional layers, each followed by one or several of the pooling, dropout and batch normalization layers. After the feature extraction, a classifier is used to model the mapping from feature space to desired output space. For a multiclass classifier, this is usually one or several fully connected layers, i.e., a MLP, followed by a final softmax layer. The activations in the final fully connected layer are real valued and are typically called logits, and the softmax layer converts the logits to probabilities, which are used as a class membership prediction. This is true also for a one-class classifier, with the softmax layer reducing to a simple sigmoid activation and the probability vector reducing to a scalar which denotes the probability of the input being a member of the normal class.

During training, backpropagation can be used for the whole network, however the convolutional layers require a special type of backpropagation, which will not be outlined here. For more information on CNNs, including a a derivation of backpropagation rules, the interested reader is referred to [14].

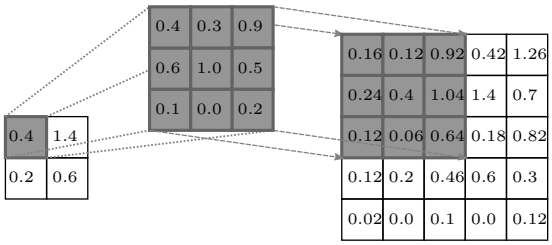
2.5 Autoencoders

An autoencoder (AE) is a FFNN which reproduces its inputs after first compressing them into a space of lower dimension. An AE has two parts: the encoder network and the decoder network. The encoder compresses the input into a latent space, and the decoder maps the latent representation back into input space. The decoder is normally a mirrored version of the encoder, meaning the autocoder architecture is symmetric. An AE is normally trained using backpropagation, using the reconstruction error, or reconstruction loss, i.e., some measure to evaluate how different the reconstruction is from the original input, as loss function. Since the target output is the input, no annotation of the training data is required, and thus the training can be said to be unsupervised. This property of AEs is useful for feature extraction when no labels for further classifying the input data are available, e.g., in OCC.

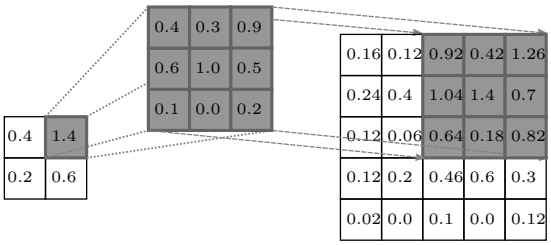
2.5.1 Convolutional Autoencoders

As the name suggests, a CAE is an AE with convolutional layers. Similarly to a CNN, it is suitable for image inputs. Where the encoder network contains convolutional layers, the decoder contains convolutional transpose layers, sometimes also called deconvolutional layers or fractionally strided convolutional layers.

Since the encoder reduces the size of the input image, the decoder needs to increase it to retain the original image shape in the output layer. Just as dimensionality reduction, dimensionality increase can be performed in different ways. Two of them are presented below.



(a) Convolutional transpose operation on the first data point in the input.



(b) Convolutional transpose operation on the second data point in the input, where the filter has been moved $s = 2$ steps in the horizontal dimension of the output, compared with the first operation in Fig. 2.6a.

Figure 2.6: Example of a convolutional transpose layer, with input of size 2, filter of size $k = 3$ and stride $s = 2$ in both dimensions. The two subfigures show the difference between the first and second position of the filter, which is moved in steps of s in the output image instead of the input image.

Unpooling layers

If the encoder network contains pooling layers, the decoder normally contains unpooling or upscaling layers in the corresponding position of the decoder. Max pooling is not reversible, but the index of the maximum value during the pooling operation can be stored. During the unpooling of a pixel, the pixel value is assigned to the output pixel corresponding to the stored index, while the rest of the unpooling output cluster is set to some constant value.

Upsampling by transposed convolution

If the encoder has convolutional stride $s > 1$ for dimensionality reduction, upsampling in the decoder can be performed in the convolutional transpose layers. In short, with a filter of size $k \times k$, 1 point in the input corresponds to k^2 points in the output, instead of the opposite, which is the case with convolutional layers. The stride s of a convolutional transpose layer refers to the step size in the output image and not the input. Another way of viewing this stride is that moving 1 point in the output corresponds to moving $1/s$ points in the input image, giving the optional name fractionally strided convolution.

Upsampling with convolutional transpose layers has the advantage of allowing more parameter optimization than unpooling, i.e., the upsampling operation is learnable. An example of a convolutional transpose layer can be seen in Figure 2.6. For a more detailed explanation of convolutional transpose layers, the interested reader is referred to [15].

2.5.2 Variational Autoencoders

In a variational autoencoder (VAE), a prior distribution is imposed on the latent space representations. This means that apart from training the encoder to extract relevant features for reconstruction and the decoder to reproduce the input, the encoder is trained to map inputs onto a specific distribution, typically a normal distribution, in latent space. Instead of encoding the input x into one latent vector $z(x)$, two vectors $\mu(x)$ and $\sigma(x)$ are generated. The latent representation fed to the

decoder is then sampled from the normal distribution $\mathcal{N}(\mu(x), \sigma(x))$. This means that during training, the same input x will generate different latent representations, making the latent space more well sampled than with deterministic encodings. The result is a latent space distribution which is continuous. This makes it possible to randomly sample latent space representations, feed them to the trained decoder, and retrieve new outputs which are similar to the training data: the VAE is said to have generative properties.

For a more thorough explanation of VAEs, we refer to [16].

2.6 Generative Adversarial Networks

A generative adversarial network consists of a generator $\mathcal{G}$ and a discriminator $\mathcal{D}$. $\mathcal{G}$ generates outputs, e.g., images, from inputs z that are sampled from some distribution $f(z)$. The purpose of $\mathcal{D}$ is to distinguish outputs of $\mathcal{G}$ from real images from some training dataset. $\mathcal{G}$ and $\mathcal{D}$ are trained jointly, but have separate loss functions: $\mathcal{G}$ is rewarded for tricking $\mathcal{D}$ into believing its outputs are from the training set and $\mathcal{D}$ is rewarded for correctly classifying inputs as either training set members or outputs of $\mathcal{G}$. Since the structure of $\mathcal{G}$ is similar to the decoder part of a VAE, $\mathcal{G}$ can be initialized as the decoder of a VAE pretrained on the training set.

2.7 Datasets

This section presents a list of publically available image datasets. Section 2.7.1 presents the image datasets that were used in the original experiments of algorithms reimplemented for this thesis. Self-driving datasets are presented in Section 2.7.2. The self-driving datasets used for experiments in this thesis are further discussed in Section 3.2.3.

2.7.1 Benchmarking Image Datasets

Modified National Institute of Standards and Technology

The Modified National Institute of Standards and Technology (MNIST) database of handwritten digits [17] is a frequently used dataset for benchmarking image classification algorithms. It consists of 70 000 images of handwritten digits: 60 000 in a training set and 10 000 in a test set, distributed across all 10 digit classes. The images are 28×28 pixels in grayscale. Examples of images from the MNIST database can be seen in Fig. 2.7.

Fashion-MNIST

The Fashion-MNIST database [18] was created to be a more challenging alternative to the original MNIST database and consists of images of garments instead of handwritten digits. It has the same format as MNIST: 70 000 grayscale images of size 28×28. Example images can be seen in Fig. 2.8.

Figure 2.7: Example images from the MNIST database of handwritten digits.

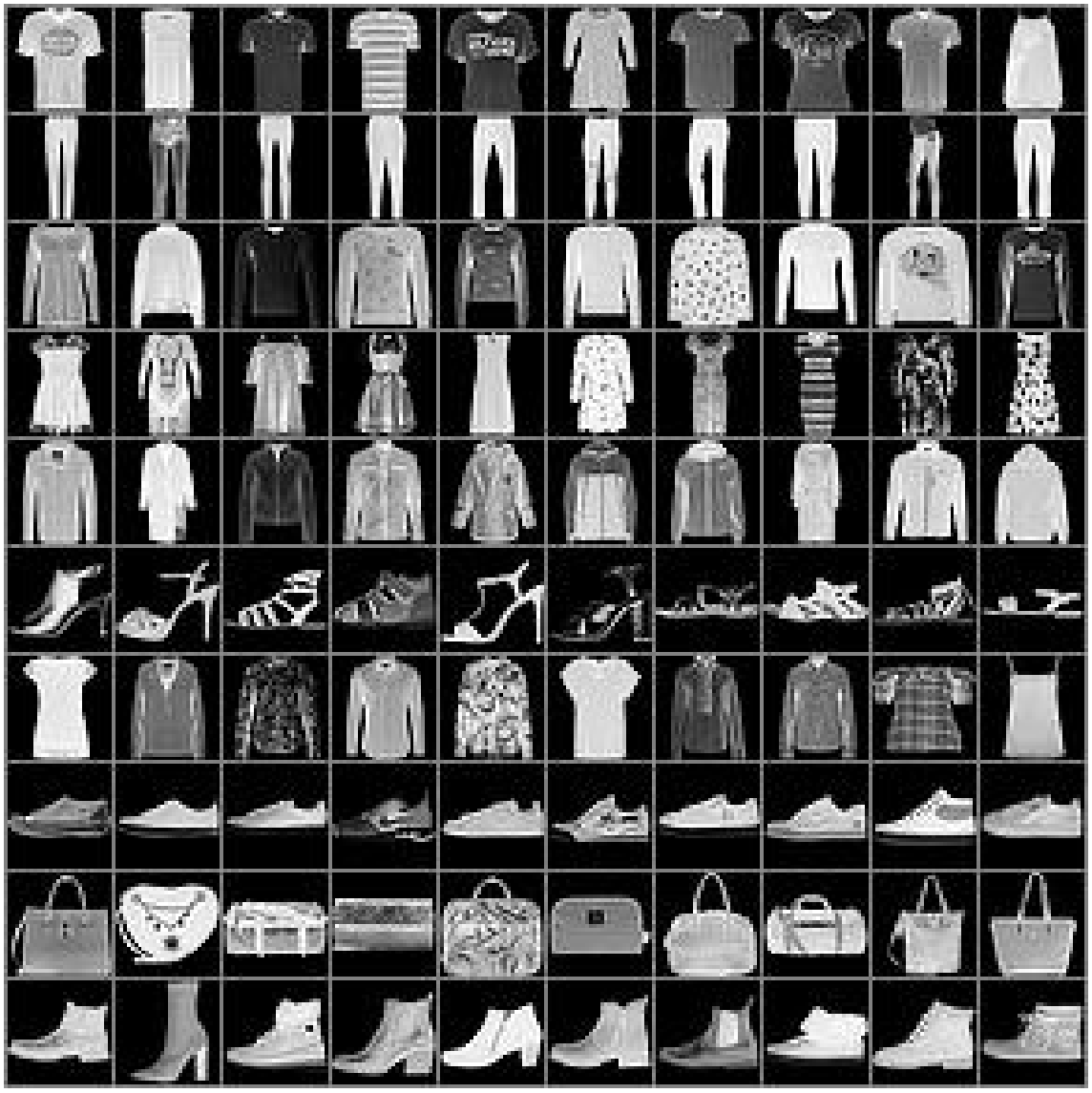


Figure 2.8: Example images from the Fashion-MNIST database.

Figure 2.9: Example images from the CIFAR-10 dataset.

CIFAR-10

The CIFAR-10 dataset [8] is a set of small color images, 32×32 pixels, in 10 different object categories. The images are downscaled from larger images of various sizes and aspect ratios. The object classes are airplane, automobile (but not truck or pickup truck), bird, cat, deer, dog, frog, horse, ship, and truck (but not pickup truck). The images depict different instances of each object class, on different backgrounds, and from different viewpoints. Each category has 6 000 samples. Example images can be seen in Fig. 2.9.

Caltech-256 Object Category Dataset

The Caltech-256 Object Category Dataset [19] is a dataset of 30 607 images of different sizes and aspect ratios, sorted into 257 different object categories. The 257th category is called "clutter" and is included to represent novelty samples. The

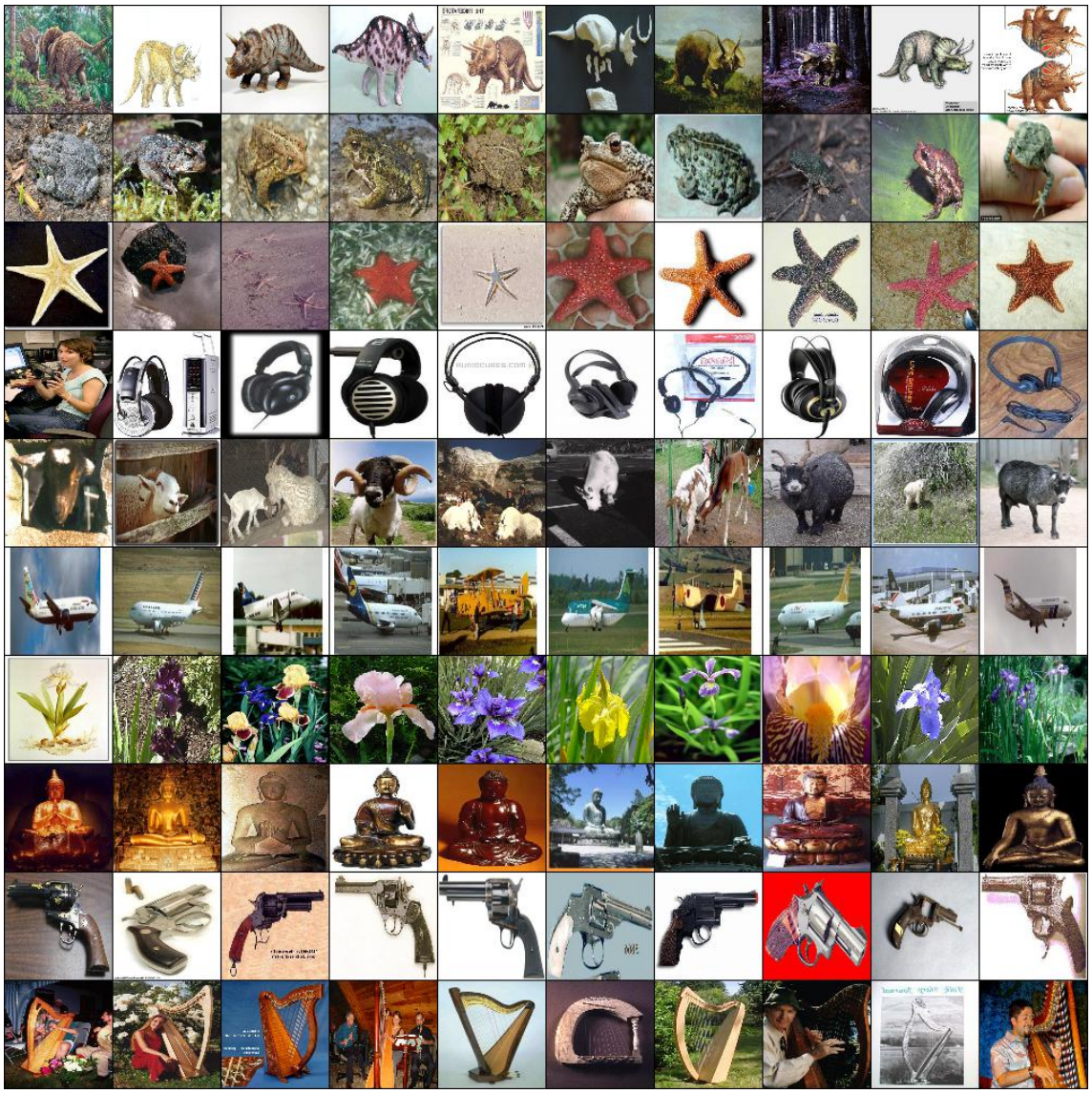


Figure 2.10: Example images from the Caltech-256 dataset. Each row shows a set of examples from one category, resized to a square image.

number of images per category varies from 80 to 807 with a mean of 119. Examples can be seen in Fig. 2.10.

Columbia Object Image Library (COIL-100)

The COIL-100 database [20] consists of 7 200 images distributed evenly across 100 different objects. Each object is depicted in 72 different angles, with a 5 degree difference between each image. Examples of 10 of the objects in 10 different poses can be seen in Fig. 2.11.

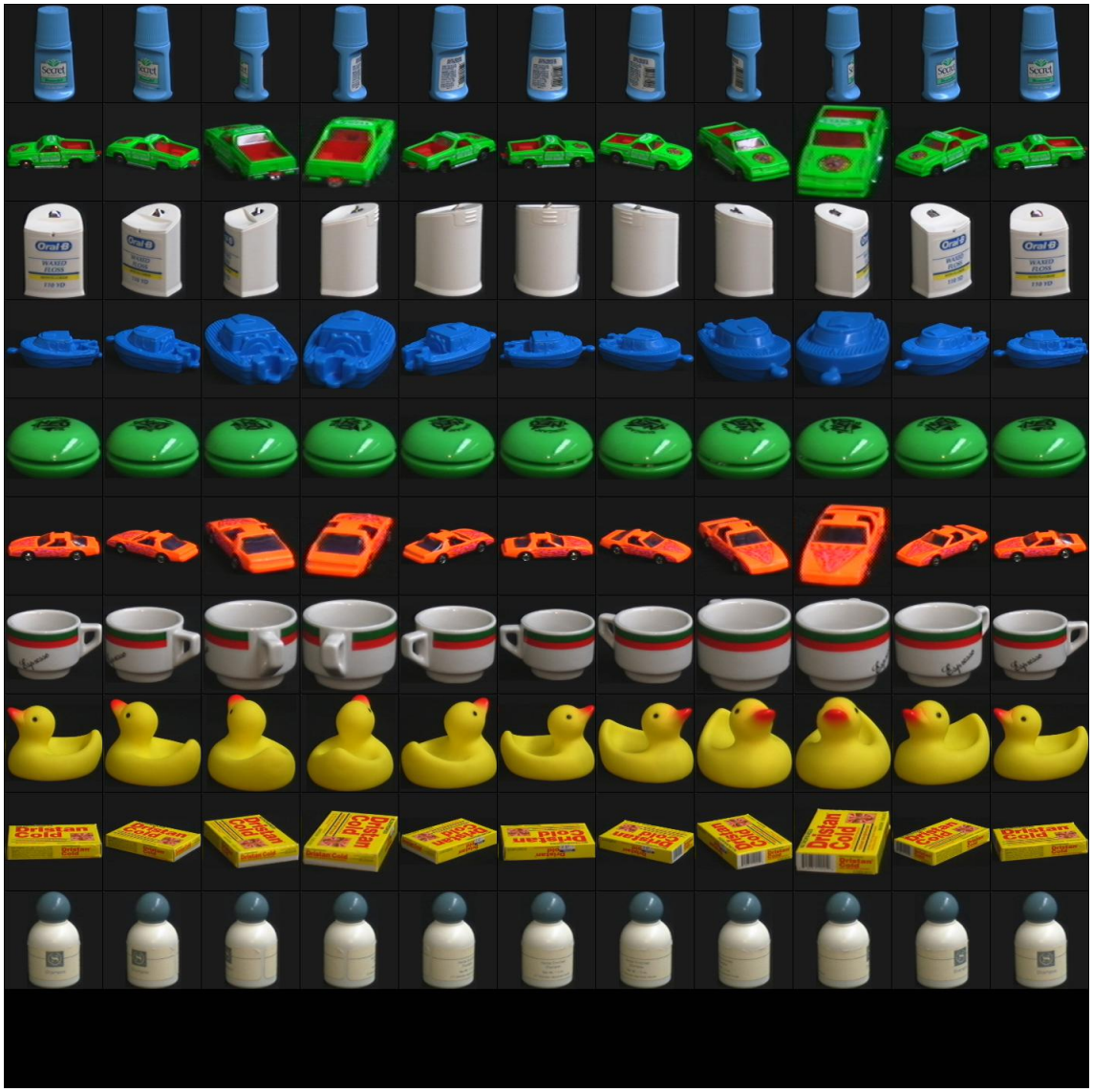


Figure 2.11: Example images from the COIL-100 dataset. Each row represents a new object, in a number of different angles.

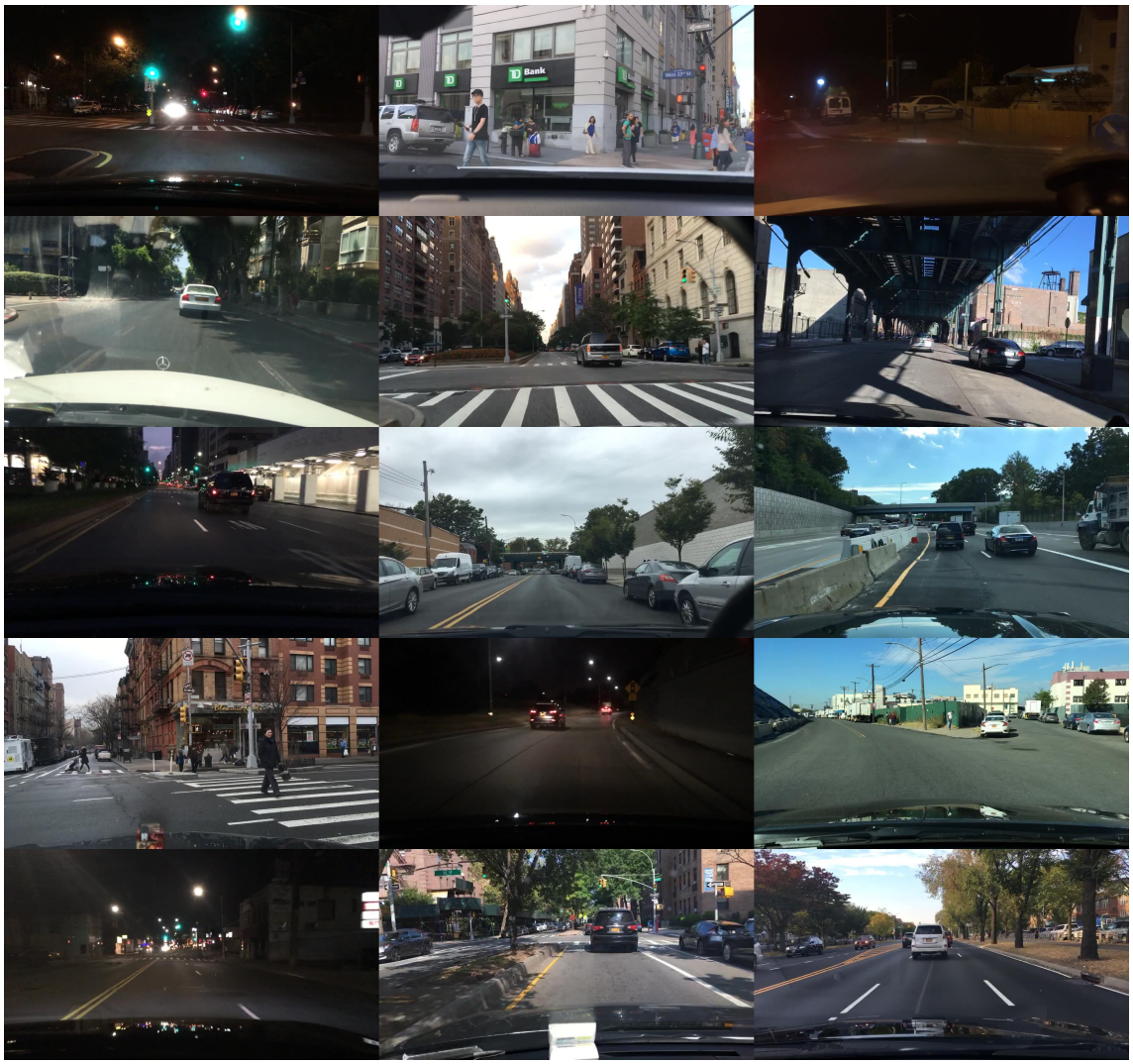


Figure 2.12: Example images from the Berkeley DeepDrive image dataset.

2.7.2 Self-driving Datasets

Berkeley DeepDrive

The Berkeley DeepDrive database [21] is a large video database for self-driving applications, consisting of 100 000 high definition videos in a range of locations and driving conditions. Each video is roughly 40 seconds long at 30 frames/s, resulting in 120 000 000 images. For each video sequence, there is metadata annotation including time of day, weather conditions and type of landscape. There is also a separate image dataset with one frame from each video sequence which, in addition to the metadata, is provided with annotation for tasks such as object detection and drivable area segmentation. Example images can be seen in Fig. 2.12.

Dr(eye)ve

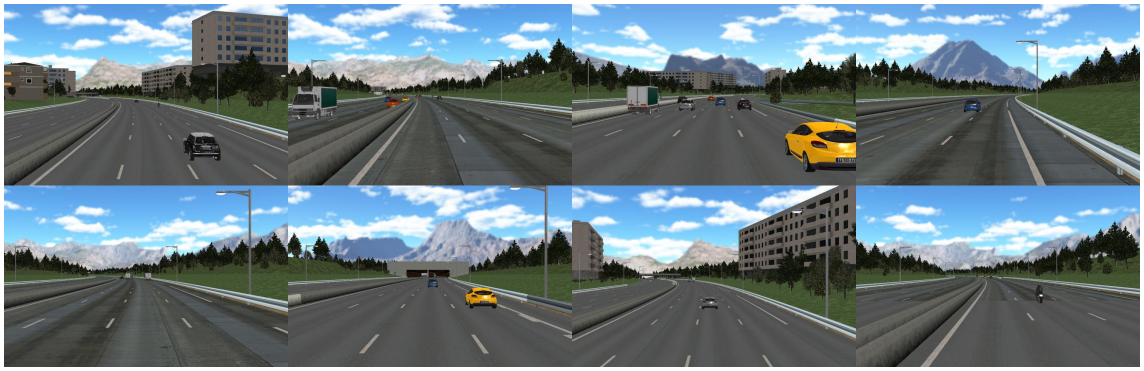
The Dr(eye)ve dataset [22] consists of 74 video sequences of 5 minutes each at 25 frames/s, totalling 555 000 frames. The videos were captured with a vehicle-

Figure 2.13: Example frames from a subset of the Dr(eye)ve dataset videos. Each image is the first frame of the corresponding video.

mounted, front-facing camera. Each video sequence is annotated with time of day (morning, evening, night), weather (sunny, cloudy, rainy) and type of landscape (highway, countryside, downtown). The data annotation also includes driver's gaze fixation, as the dataset was originally created for tasks regarding driver attention. Example frames can be seen in Fig. 2.13.

Pro-SiVIC highway scenario dataset

The Pro-SiVIC highway scenario dataset was created for the project SMILE II by QRTECH AB, and consists of images generated in the simulator ESI Pro-SiVIC™ [23]. The dataset is divided into three scenarios with different types of conditions: a highway in sunny weather, the same highway in heavy fog, and an urban setting in sunny weather. Example images from each of the three conditions can be seen in Fig. 2.14.



(a) Images from the Pro-SiVIC highway scenario dataset. The scenario is defined by sunny weather and a highway landscape.

(b) Images from the Pro-SiVIC highway scenario dataset. The scenario is defined by foggy weather and a highway landscape.

(c) Images from the Pro-SiVIC highway scenario dataset. The scenario is defined by sunny weather and an urban landscape.

Figure 2.14: Example images from the Pro-SiVIC highway scenario dataset.

3
Literature Review

This section presents the literature review which was conducted to get an understanding of the current state-of-the-art in ND in image data, and to identify the most suitable ND algorithms for evaluation in this thesis project. First, we present the applied method for finding and selecting ND algorithms and self-driving datasets. Afterwards, the results of the review are presented.

3.1 Methodology

The methodology for the literature review is presented in the following order: first, the principle for finding and selecting articles to read is presented. Then, the criteria for selecting algorithms for reimplementation and criteria for selecting datasets for evaluation are presented.

3.1.1 Finding Articles

The first articles were found through a keyword search, such as "novelty detection", using the search engine Google Scholar. Articles were chosen based on title and filtered after reading the abstracts. After fully reading the first round of articles, an associative search method was mainly used, inspired by [24]: new articles were found through association with those already analyzed, i.e., by being references in read articles or by appearing in the related or recommended articles section of the database page of a read article. This process replaced the earlier keyword search, and new articles were again filtered, first by title and then by abstract. During the reading process, other relevant keywords were encountered, such as synonyms for ND. The process would then start over with keyword search. Throughout the article selection process, filtering was based on the criteria in Section 3.1.2, where all articles were given the benefit of the doubt: if the article was not obviously irrelevant, it was deemed potentially relevant and would go on to the next stage.

The search for new articles was concluded when all of the articles selected through associative search had already been processed before, meaning they had either been read or discarded, indicating that the scientific field in question had been examined to a such an extent that publications presenting current state-of-the-art algorithms were unlikely to have been overlooked.

3.1.2 Ground criteria for algorithm selection

Algorithms were selected for reimplementation subject to the following criteria:

A1 Each selected algorithm must yield good results, in terms of area under receiver operating characteristic (AUROC) (see Section 4.2.3), for at least one well-known image dataset.
A2 Each selected algorithm must have source code readily available and licensed for use in research purposes.
A3 No selected algorithm should require the normal class to have labeled subclasses: they should be able to model a single normal class.
A4 The number of selected algorithms should be at least 3, so that a comparison can be made.
A5 The number of selected algorithms should not be too large so as to allow for the selected algorithms to be reimplemented and tested within the limited time of the project.

3.1.3 Ground criteria for dataset selection

Self-driving datasets were selected for use in this thesis subject to the following critera:

D1 All selected datasets must be image datasets for self-driving applications, with images taken in the forward direction of a vehicle in road traffic.
D2 Each selected dataset must have metadata available, e.g., weather conditions, such that the data can be divided into at least one normal class and at least one novelty class.
D3 Each dataset should represent a different level of novelty detection difficulty, i.e., if more than one dataset is selected they should have different variability in the elements of the normal class.
D4 It is preferable if all selected datasets can be evaluated with similar differences between the normal scenarios and the novelty scenarios.
D5 The number of selected datasets should not be larger than such that all selected algorithms can be tested with each dataset within the limited time of the project.

3.2 Results of Literature Review

The implemented method for finding relevant articles, presented in Section 3.1.1, resulted in a total of 17 which were deemed relevant enough for this thesis. In this section, they are presented as follows: first, there is a summary of the read material. Then, the three algorithms selected for reimplementation and experiments are explained in more detail. Finally, there are some remarks on interesting algorithms which, for various reasons, were dismissed.

3.2.1 Summary of Current State-of-the-art Novelty Detection

The latest complete review of the field of ND was made in 2014 by Pimentel et al. [25]. The majority of papers reviewed here were published later, however it serves

as a basis for understanding the broader field of ND. In [25], the authors sort ND algorithms into 6 different categories: probabilistic, distance-based, reconstruction-based, domain-based and information-theoretic ND. They further state that in the application domain of image processing, there exist algorithms in all but the last of these categories.

Since we concern ourselves only with ND in image data, the most common approach among those investigated is to use a CAE for feature extraction in some way. This includes both using regular CAEs [11, 26, 27, 28, 29] and using generative adversarial networks with convolutional encoder and generator [30, 31, 32, 33]. Other approaches use non-convolutional deep AEs, either by preprocessing the image data and thereby reducing its dimensionality [34] or by using the entire image in a fully connected AE [35]. Some algorithms use no AE at all, but instead use CNNs for feature extraction, either with transfer learning from pre-trained models [36, 37], or by only working on normal classes with subclasses in their original implementation [38, 39].

The main difference between algorithms with similar feature extraction methods is how the trained ANN is used for assigning novelty scores to testing inputs. Common ways of doing this are using CAE reconstruction error, using the discriminator output in a generative adversarial network, or applying a separate OCC algorithm to the feature space representation of a CAE.

In [40], the authors compare 20 different VAE statistics as novelty scores, and show that though some metrics yield higher AUROC than the most typical one, which is the output layer reconstruction error, there is no large variation between the top 10 AUROC scores, which are all in the range $[0.86, 0.881]$. Although for a VAE and not a regular CAE, it indicates that for the same trained model, different internal statistics of the model contain the same amount of information about the normal class features.

3.2.2 Algorithms Selected for Reimplementation

The algorithms selected for reimplementation and evaluation in this thesis are presented and motivated below. Each algorithm is presented under an acronym, primarily the one used by the original authors. If there was no such acronym, one has been devised here.

Adversarially learned one-class classifier for novelty detection (ALOCC)

Sabokrou et al. [32] train an adversarial autoencoder (AAE), which means the training objective is a weighted combination between reconstruction loss and an adversarial loss. The setup consists of two networks; the autoencoder, denoted $\mathcal{R}$, and a discriminator $\mathcal{D}$. $\mathcal{R}$ maps the input image x as

$$\mathcal{R}: \qquad \tilde{x} = (x \sim p_t) + (\eta \sim \mathcal{N}_\sigma) \longrightarrow x' \sim p_t, \tag{3.1}$$

where $\mathcal{N}_\sigma = \mathcal{N}(0, \sigma^2\mathbf{I})$ is normally distributed noise with zero mean and variance σ^2. Note that the reconstructed image x' is mapped to the same distribution p_t as

the input x, making $\mathcal{R}$ a denoising AE. The discriminator $\mathcal{D}$ maps the output of $\mathcal{R}$ as

$$\mathcal{D}: \qquad \mathcal{R}(\tilde{x}) \longrightarrow p \in (0,1), \tag{3.2}$$

$$\min_{\mathcal{R}} \max_{\mathcal{D}} \mathcal{L}_{\mathcal{R}+\mathcal{D}}, \tag{3.3}$$

where

$$\mathcal{L}_{\mathcal{R}+\mathcal{D}} = \mathrm{E}_{x\sim p_t}\left[\log\left(\mathcal{D}(x)\right)\right] + \mathrm{E}_{\tilde{x}\sim p_t * \mathcal{N}_\sigma}[\log\left(1-\mathcal{D}(\mathcal{R}(\tilde{x}))\right)]. \tag{3.4}$$

An intuitive way to explain (3.4) is that both terms train $\mathcal{D}$ to distinguish training images $x \sim p_t$ from reconstructed images $\mathcal{R}(\tilde{x})$, since outputting $\mathcal{D}(x) = 1$ and $\mathcal{D}(\mathcal{R}(\tilde{x})) = 0$ maximizes (3.4). The $\mathcal{R}$ network is only affected by the second term, where (3.4) is minimized for $\mathcal{D}(\mathcal{R}(\tilde{x})) = 1$, meaning that $\mathcal{R}$ successfully tricks $\mathcal{D}$ that $\mathcal{R}(\tilde{x})$ belongs to the training dataset distribution p_t.

The $\mathcal{R}$ network is also trained with a reconstruction loss

$$\mathcal{L}_{\mathcal{R}} = ||x - x'||^2 \tag{3.5}$$

giving the complete training objective

$$\min_{\mathcal{R}} \left\{ \max_{\mathcal{D}} \mathcal{L}_{\mathcal{R}+\mathcal{D}} + \lambda \mathcal{L}_{\mathcal{R}} \right\} \tag{3.6}$$

where $\lambda > 0$ is a tradeoff hyperparameter.

During testing of an image $\bar{x}$, the normal class likelihood $S_{\text{normal}}(x) = \mathcal{D}(\mathcal{R}(\bar{x}))$ is used to detect novelties. Note that since $\mathcal{D}$ outputs higher values for normal class images, the score is a normalcy score, so that low scores signify higher probability of $\bar{x}$ being a novelty.

In the original paper [32], the algorithm is tested on the MNIST database as well as the Caltech-256 Object Category Dataset, both presented in Section 2.7.1. The authors compare $\mathcal{D}(\bar{x})$ and $\mathcal{D}(\mathcal{R}(\bar{x}))$ as scoring functions, obtaining AUROC= 0.932 and AUROC= 0.942 respectively for the Caltech dataset, using one object class as normal class and 50% novelties, sampled from the 257th class, "clutter".

Deep support vector data-description (DSVDD)

Ruff et al. [11] pre-train a CAE for learning normal class features, and then use the encoder network as initialization for a CNN used for feature extraction. A SVDD classifier is attached to the final layer of the CNN to perform ND, and the resulting method is called DSVDD. The two parts of the new network are then trained jointly, with objectives designed to optimize network parameters W so that the CNN learns to map samples $\mathbf{x}$ from the normal class into a hypersphere of radius R. Two algorithms are proposed: soft-boundary DSVDD and one-class DSVDD. Soft-boundary DSVDD has the objective

$$\min_{R,W} \left\{ R^2 + \frac{1}{\nu n} \sum_{i=1}^{n} \max\{0, \|\phi(\mathbf{x_i}; W) - \mathbf{c}\|^2 - R^2\} + \frac{\lambda}{2} \sum_{l=1}^{L} \|\mathbf{W}^l\|_F^2 \right\}, \tag{3.7}$$

and one-class DSVDD has the objective

$$\min_{W} \left\{ \frac{1}{n} \sum_{i=1}^{n} |\phi(\mathbf{x_i}; W) - \mathbf{c}\|^2 \ + \frac{\lambda}{2} \sum_{l=1}^{L} \|\mathbf{W}^l\|_F^2 \right\}. \tag{3.8}$$

In both (3.7) and (3.8), $||\cdot||_F$ denotes the Frobenius norm

$$\|\mathbf{W}^l\|_F = \sqrt{\sum_{i,j} {W_{ij}^l}^2}, \tag{3.9}$$

i.e., the root square sum of all network parameters.

The soft-boundary DSVDD objective (3.7) optimizes parameters W and hypersphere radius R jointly. The first term aims to minimize the hypersphere volume. The second term penalizes the network for all points lying outside of the sphere, since the max operator sets the second term to zero for all points within the hypersphere. The hyperparameter ν controls the tradeoff between the two terms.

The one-class DSVDD objective (3.8) penalizes the distance of any feature point $\phi(\mathbf{x}; W)$ to the center $\mathbf{c}$, implicitly minimizing the radius of the smallest hypersphere enclosing all feature space representations. In both (3.7) and (3.8), the last term is a regularizer with hyperparameter λ, serving as a type of weight decay.

In the source code [41] for DSVDD algorithm, there is the option to optimize both (3.7) and (3.8) w.r.t. to the center $\mathbf{c}$ as well as R and W. However, the authors recommend not to do so, since it increases the risk for what they call hypersphere collapse: when the network learns the trivial solution to set $R = 0$ and a constant mapping $\phi(\mathbf{x}; W) = \mathbf{c}_0$ for any $\mathbf{x}$.

The novelty score of a new input $\bar{\mathbf{x}}$ is assigned in a similar way for both soft-boundary DSVDD and one-class DSVDD. For one-class DSVDD, it is simply the distance from the feature space point to the hypersphere center:

$$S_{\text{novelty}}(\bar{\mathbf{x}}) = s(\bar{\mathbf{x}}) = ||\phi(\bar{\mathbf{x}}; W) - \mathbf{c}||^2. \tag{3.10}$$

For soft-boundary DSVDD, the novelty score is set as $S_{\text{novelty}}(\bar{\mathbf{x}}) = s(\bar{\mathbf{x}}) - R$, as to get negative scores for normal class inputs and positive scores for novelties. In the original paper [11], the algorithm is benchmarked on, among others, the MNIST database and the CIFAR-10 dataset, see Section 2.7.1. For both datasets, one class at a time is used as normal class with samples from the all other classes used as novelties. For MNIST, the average AUROC is 0.935 and 0.948 for the soft-boundary and one-class methods, respectively, while the corresponding values for CIFAR-10 are 0.633 and 0.648.

Generative probabilistic novelty detection (GPND)

Pidhorskyi et al. [30] propose a probabilistic novelty score based on learning features of the normal class in an AAE. It is assumed that all normal class samples $x_i \in \mathbb{R}^m, i = 1, \ldots, N$, are sampled from a manifold $\mathcal{M}$ of dimension $n < m$, such that

$$x_i = f(z_i) + \xi_i, \tag{3.11}$$

where $z_i \in \Omega \subset \mathbb{R}^n$ and ξ_i denotes noise. The manifold $\mathcal{M}$ is then defined by

$$\mathcal{M} \equiv f(\Omega). \tag{3.12}$$

The authors further assume that $f : \Omega \to R^m$ is smooth and invertible with inverse $g : \mathbb{R}^m \to \mathbb{R}^n$ such that $x_i = f(g(x_i))$, $i = 1, \ldots, N$. By linearizing f on $\mathcal{M}$, using a

first order Taylor expansion, they express the probability $p_X(\bar{x})$ that a new input $\bar{x}$ is sampled from $\mathcal{M}$ in terms of entities whose computation only require numerical estimates of f and g. The full derivation of this probability estimation will not be covered here, but can be found in the original paper [30].

The mappings f and g are approximated using an AAE: the encoder network mapping of input x to latent representation z represents g, while the decoder network represents f. The encoder-decoder network is also trained in a way similar to a VAE, that is, a prior distribution is imposed on the latent space Ω. In this case it is a standard normal distribution $\mathcal{N}(0,1)$. In addition to the VAE objective, the adversarial setup consists of two discriminators, D_z and D_x, discriminating upon the latent space representation and the reconstructed image, respectively. The loss $\mathcal{L}_{\text{adv}-d_z}$ for D_z is defined as

$$\mathcal{L}_{\text{adv}-d_z}(x, g, D_z) = \mathrm{E}\left[\log\left(D_z(\mathcal{N}(0,1))\right)\right] + \mathrm{E}\left[\log\left(1 - D_z(g(x))\right)\right]. \tag{3.13}$$

Minimizing (3.13) w.r.t. the parameters of g trains g to map x onto z following the prior distribution $\mathcal{N}(0,1)$. Maximizing it w.r.t. the parameters of D_z trains D_z to distinguish between the mappings of x and random samples from the prior distribution.

The loss $\mathcal{L}_{\text{adv}-d_x}$ for D_x is defined as

$$\mathcal{L}_{\text{adv}-d_x}(x, f, D_x) = \mathrm{E}\left[\log\left(D_x(x)\right)\right] + \mathrm{E}\left[\log\left(1 - D_x(f(\mathcal{N}(0,1)))\right)\right]. \tag{3.14}$$

Minimizing (3.14) w.r.t. the parameters of f trains f to map samples from $\mathcal{N}(0,1)$ to reconstructed images that resemble the input images x. Maximizing it w.r.t. the parameters of D_x trains D_x to distinguish between the generated images $f(\mathcal{N}(0,1))$ and x.

To approximate the manifold $\mathcal{M}$ well, a reconstruction loss $\mathcal{L}_{\text{error}}$ is imposed on g and f:

$$\mathcal{L}_{\text{error}}(x, g, f) = \mathrm{E}\left[\mathrm{BCE}_{\text{image}}(x, f(g(x)))\right], \tag{3.15}$$

where

$$\mathrm{BCE}_{\text{image}}(x, f(g(x))) = \sum_{j=1}^{m} \mathrm{BCE}(\mathrm{y_j}, \mathrm{p_j}), \tag{3.16}$$

where y_j and p_j are data points in the images x and $f(g(x))$, respectively, and $BCE(p_j, y_j)$ is given by (2.9).

The AAE is trained using stochastic gradient descent, updating the networks in the following order:

1. Maximize $\mathcal{L}_{\text{adv}-d_x}$ w.r.t. parameters of D_x.
2. Minimize $\mathcal{L}_{\text{adv}-d_x}$ w.r.t. parameters of f.
3. Maximize $\mathcal{L}_{\text{adv}-d_z}$ w.r.t. parameters of D_z.
4. Minimize $\mathcal{L}_{\text{adv}-d_z} + \lambda\mathcal{L}_{\text{error}}$ w.r.t. parameters of g and f,

where λ is a hyperparameter controlling the tradeoff between adversarial loss and reconstruction loss.

After completed training, the approximations of f and g are used to estimate $p_X(\bar{x})$, which is in turn used as the normalcy score $S_{\text{normal}}(x)$ for any input $\bar{x}$. In the original paper [30], the algorithm is benchmarked on the MNIST, Fashion-MNIST and COIL-100 datasets. AUROC results, using one object class as normal class

and 50% novelties sampled from the other classes, are 0.932 for MNIST, 0.901 for Fasion-MNIST, and 0.968 for COIL-100.

Dismissed algorithms

All ND articles and related algorithms from the literature review were judged based on the conditions presented in Section 3.1.2. The algorithms selected for reimplementation were mainly selected for their source code availability and their relative difference in approach. Several [27, 28, 31, 35, 36, 37, 38] of the considered algorithms met most of the conditions on individual algorithms, and are worth investigating in future work.

3.2.3 Datasets Selected for the Evaluations

Three datasets were considered for use in experiments in this thesis: Berkeley DeepDrive, Dr(eye)ve and the Pro-SiVIC highway scenario dataset, all described in Section 2.7.2.

The datasets chosen for ND experiments were the Dr(eye)ve dataset and the Pro-SiVIC highway scenario dataset. The Pro-SiVIC highway scenario dataset provides a simple test case, with relatively low scene variation even though elements such as vehicles, bridges and buildings on the side of the road gives the scenes some complexity. The Dr(eye)ve dataset provides an increase in scene variation and complexity, since it consists of real world images and the images are captured across different runs, in different locations.

Dismissed datasets

The time frame of this thesis did not allow for three datasets to be used, specified in the final criterion in Section 3.1.3. The Berkeley DeepDrive database was omitted from experiments because it was deemed, through initial testing, to present a higher level of difficulty than the two selected datasets. The increased difficulty is likely caused by the number of runs used for the different datasets: a single run for the Pro-SiVIC highway scenario dataset, 74 runs for the Dr(eye)ve dataset, and 100 000 runs for the Berkeley DeepDrive dataset. The high number of runs in Berkely DeepDrive causes an increase in the variation within the training dataset due to, e.g., an increased number of landscape types and a larger variation in the camera angle.

4
Novelty Detection Experiments

This chapter presents all the experiments performed with the algorithms selected in Section 3.2.2 and datasets selected in Section 3.2.3. First, the reimplementation of the selected algorithms is detailed. Then, the experimental setup is outlined. Finally, the results of all experiments are reported.

4.1 Reimplementation of Selected Algorithms

All selected algorithms have full source code available online, which greatly facilitated reimplementation. Each algorithm was implemented using the original source code, with modifications. All three algorithms were originally implemented in Python, but in different Python versions and different machine learning (ML) frameworks, shown in Table 4.1. Any resulting differences regarding model performance, given identical ANN architectures and optimization settings, were assumed to be negligible.

Since all the selected algorithms were originally implemented for MNIST and datasets with images of approximately the same size, the main modification was to adapt each of them to larger input images. Since larger images contain more information, CAE architectures with more convolutional layers were needed in order to extract enough relevant features from the training datasets.

For all three selected algorithms a method was implemented, which enabled to easily change the number of convolutional layers and filters in each CAE, in order to evaluate how much different hyperparameters affected the ability of the CAEs to encode meaningful latent representations of the normal class samples. Since training a CAE until convergence with the full datasets took on a timescale of hours, a subset of 100 images was used for testing different architectures. The investigated hyperparameters were:

- Convolutional filter size: $k \in \{4, 5\}$.
- Number of convolutional layers: $2 \leq n_{\text{conv}} \leq 7$.
- Number of fully connected layers: $n_{\text{fc}} \in \{0, 1\}$.
- Number of filters in the first convolutional layer: $c_1 \in \{8, 16, 32, 64\}$.
- Dimensionality of the latent representation: $c_z \in \{256, 512, 1\,024, 2\,048\}$.

Table 4.1: Programming frameworks used in the implementations of the evaluated algorithms

Algorithm	**Python version**	**ML framework**
ALOCC	3.5.2	Keras 2.2.4
DSVDD	2.7.12	Lasagne 0.2.dev1
GPND	3.5.2	Torch 0.4.1

Table 4.2: Number of images in the dataset splits. The two values in the Pro-SiVIC highway scenario test set refer to the novelty scenarios weather/landscape

Dataset	Pro-SiVIC highway scenario	Dr(eye)ve
Number of training images	6 785	6 000
Number of validation images	840	600
Number of test normal samples	500/488	600
Number of test novelties	500/488	600

- Learning rate: $\eta \in \{0.0001, 0.001, 0.01\}$.

The number of convolutional layers n_{conv} and fully connected layers n_{fc} refer to either of the encoder and the decoder, meaning that the whole CAE had n_{conv} convolutional layers, $2n_{\text{fc}}$ fully connected layers, and n_{conv} transposed convolution layers. Settings which were kept constant are 2×2 stride, to get feature map dimensionality reduction in each convolutional layer, and that the number of filters was doubled for each new convolutional layer: e.g., with $n_{\text{conv}} = 3$ and $c_1 = 8$ the number of filters in each of the layers would be $\{8, 16, 32\}$. A full grid search was not done, as the hyperparameter options listed above yield 1152 combinations. Instead, one hyperparameter at a time was varied while keeping the others constant. Some combinations, such as $n_{\text{conv}} = 7, c_1 \geq 32$, yielded models too large to fit into the memory of the used hardware (see Section 4.2.4), and could not be evaluated. It was further assumed that the feature extraction capability of a CAE model with a certain architecture would be the same for all three algorithm implementations, so that testing different architectures for only one of them would suffice to determine a common architecture to be used with all three. The implementation used for this was ALOCC, because of the user-friendly functionality for changing the CAE architecture provided by the Keras ML framework.

For each hyperparameter setting, a CAE was trained until the average reconstruction error over the training data subset was no longer decreasing. Using the training set reconstruction error for this increases the risk for overfitted models. However, this risk was deemed low in practice, based on initial experiments with the DSVDD algorithm, where the average error for the validation and training sets started diverging after a considerably larger number of epochs than what was used in the experiments presented here. Settings to be used for the presented experiments were chosen using a tradeoff between the number of epochs needed for convergence and the lowest reconstruction error reached: a model with more layers and filters could, in theory, extract and reconstruct more complex and detailed features due to the increased number of neurons, but the weight search space is larger and it might not be practically feasible to find the optimal weight configuration.

4.2 Experimental Setup

4.2.1 Dataset Preparation

The number of images in the training, validation and testing subsets of each dataset are shown in Table 4.2. Below is a description of how images were arranged into normal and novelty sets for the respective datasets.

Table 4.3: Attributes used for splitting the Dr(eye)ve dataset into normal scenario and novelty scenarios, and the videos each set of images was sampled from

Scenario	Weather	Landscape	Sampled videos
Normal	Sunny	Higway or countryside	23, 25, 34, 45, 55
Novel weather	Rainy	Highway or countryside	14, 17, 22, 31, 32, 37, 44, 50, 63
Novel landscape	Sunny	Downtown	06, 40, 65

Pro-SiVIC highway scenario dataset splits

Since the Pro-SiVIC highway scenario dataset was created for the SMILE II project, the normal and novelty classes were designed specifically for the purpose of novel scenario detection. The normal class is set on a highway in sunny weather conditions. The two types of novelty scenarios are, in relation to the normal class:

1. same landscape, but with foggy weather conditions,
2. same weather conditions, but in urban conditions.

Example images from the normal class and the two novelty scenarios are shown in Fig. 4.1.

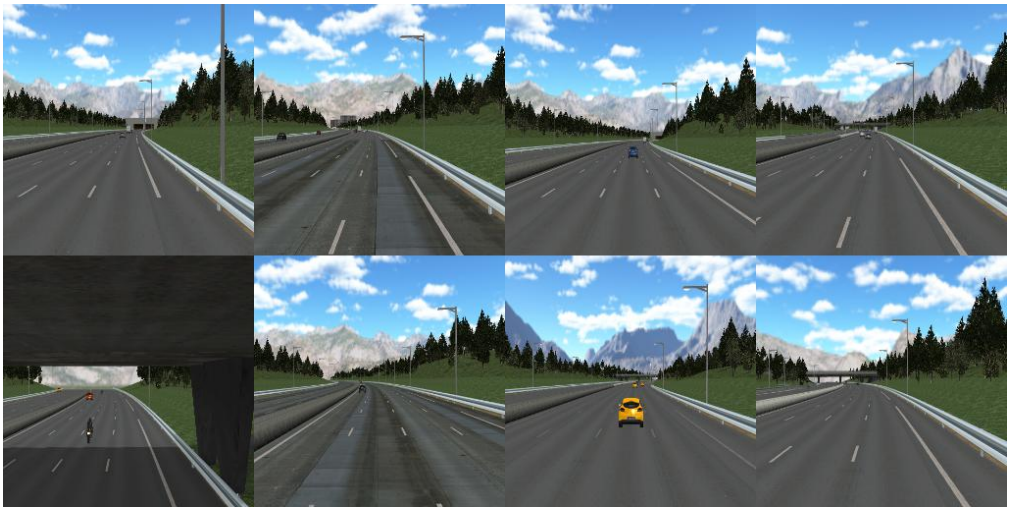
Dr(eye)ve dataset splits

To get a normal class scenario and novel scenarios which are similar for both datasets, a subset of the videos in the Dr(eye)ve dataset were selected based on the metadata attributes provided: time of day, weather and landscape.

Since labels for time of day were "morning", "evening" and "night", while the requirement for these experiments were only for all images to be in daylight conditions, an extra filtering of data was required. The relatively low number of videos, 74, in the Dr(eye)ve dataset, allowed for manual inspection of all videos. This resulted in a relabeling of each video as either having daylight conditions, eligible for use in this thesis, or being too dark and consequently left out of experiments. Out of the videos with daylight conditions, sets of weather and landscape attributes were chosen as similar as possible to the scenarios for the Pro-SiVIC highway scenario dataset. The attributes chosen for the different data subsets are shown in Table 4.3. Each of the 5 minute videos selected for the respective scenarios was then sampled at a rate of 5 frames/s. The images extracted from the normal class videos were randomly sorted into training, validation and testing splits. When the number of videos matching the attribute description of a scenario was so large that 5 frames/s sampling generated more images than required, a subset of the extracted frames was randomly selected. Example images from each of the three scenarios can be seen in Fig. 4.2.

Common preprocessing of both datasets

Images in both datasets were preprocessed by resizing to 256×256 pixels using OpenCV's resize function with INTER_AREA interpolation option, and rescaling all pixel values to the range $[0, 1]$. The image size 256×256 was chosen as a tradeoff between reducing the input dimensionality while still keeping much of the detail in the images.



(a) Resized images from the Pro-SiVIC highway scenario dataset normal scenario.

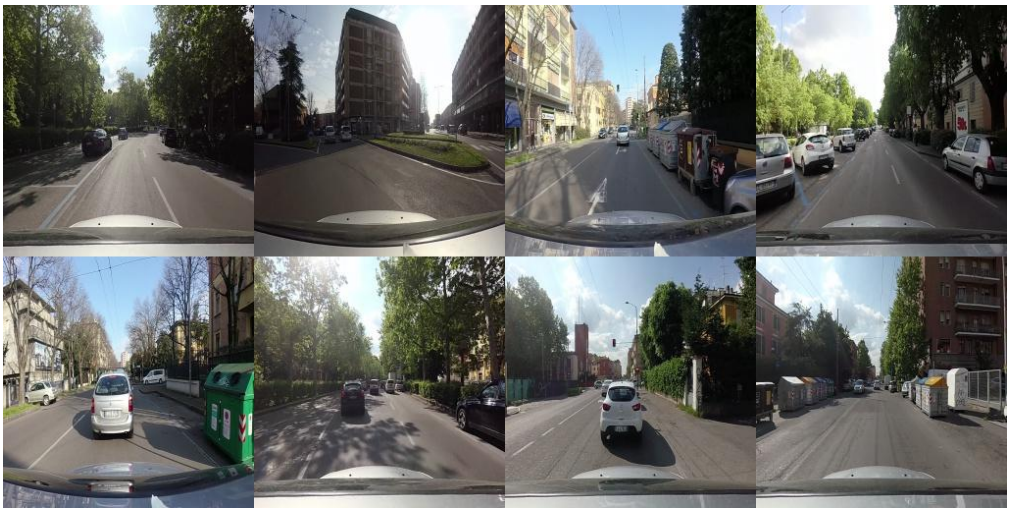
(b) Resized images from the Pro-SiVIC highway scenario dataset novel weather scenario.

(c) Resized images from the Pro-SiVIC highway scenario dataset novel landscape scenario.

Figure 4.1: Resized image samples from the normal class and the two novelty scenarios in the Pro-SiVIC highway scenario dataset.

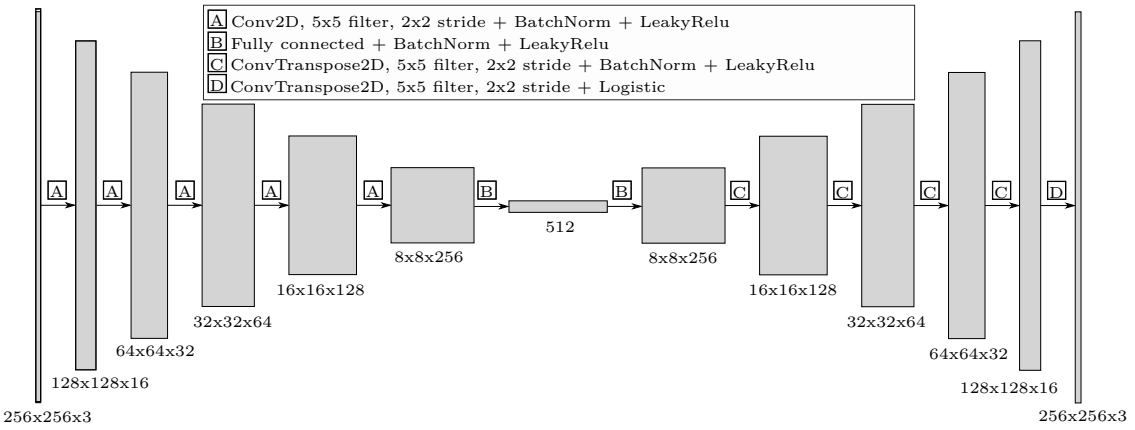
(a) Resized images from the Dr(eye)ve dataset normal scenario.

(b) Resized images from the Dr(eye)ve dataset novel weather scenario.

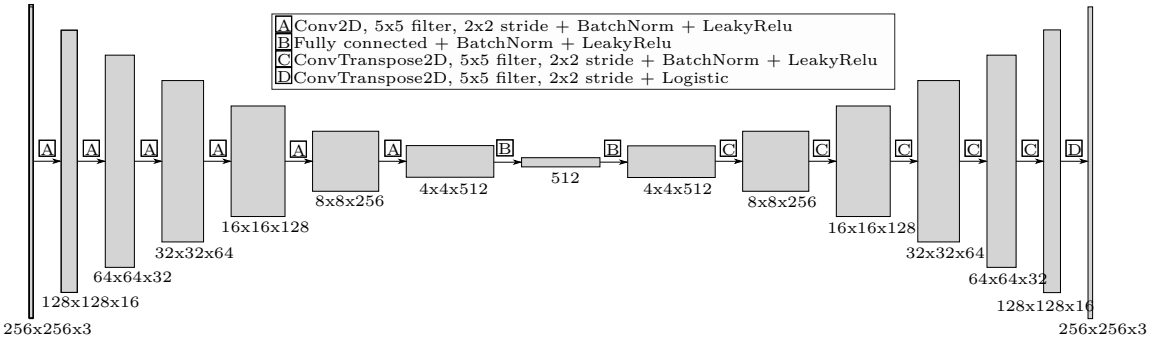


(c) Resized images from the Dr(eye)ve dataset novel landscape scenario.

Figure 4.2: Resized image samples from the normal class and the two novelty scenarios in the Dr(eye)ve dataset.



(a) Pro-SiVIC highway scenario CAE architecture.



(b) Dr(eye)ve CAE architecture.

Figure 4.3: CAE architectures used for experiments with the respective datasets. In both cases, an extra batch normalization layer was added after the output layer for the ALOCC algorithm, as this proved to improve results.

4.2.2 Optimization of Novelty Detection Models

For each dataset and algorithm, CAE optimization was performed with the algorithm specific optimization objective, presented in Section 3.2.2, and dataset specific architecture and training settings, which are listed in Table 4.4. The CAE architectures used for the two datasets are also depicted in Fig. 4.3. The discriminator architectures used with the ALOCC and GPND are shown in Figs. 4.4–4.5. For the GPND algorithm, the architecture is identical to the CAE encoder network for the respective datasets, except for the output being a single scalar. For the ALOCC algorithm, an additional convolutional layer is used. The reason for the difference is that in the original implementations, the GPND discriminator has the same depth as the CAE encoder, while the ALOCC discriminator has one more convolutional layer.

4.2.3 Evaluation metrics

For evaluating ND experiments as a type of OCC, we define the prediction *novelty* to be a positive prediction in this thesis, since that is what the algorithms are aiming to detect. Consequently, normal class membership is labeled as a negative prediction. For each input x in a testing set, each algorithm assign a novelty score $S_{\text{novelty}}(x)$. For the two algorithms which in their original implementation output a

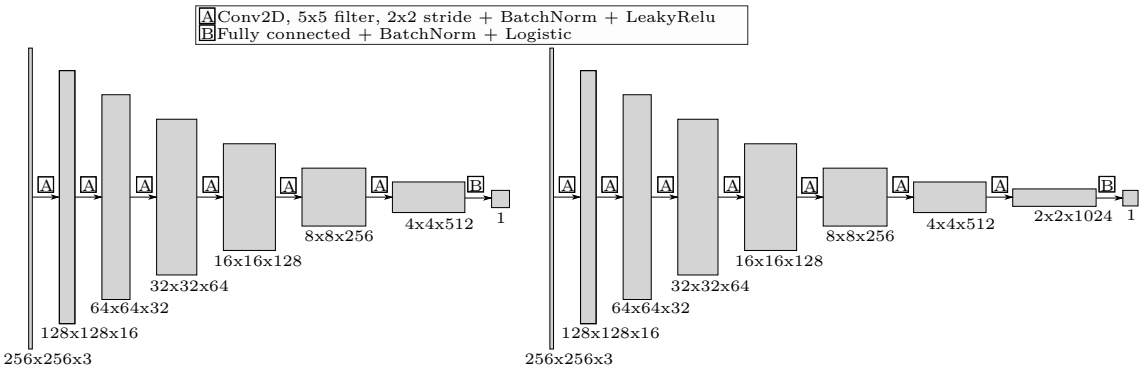


Figure 4.4: Architectures used for the discriminators in the ALOCC algorithm. Left: architecture for the Pro-SiVIC highway scenario dataset. Right: architecture for the Dr(eye)ve dataset.

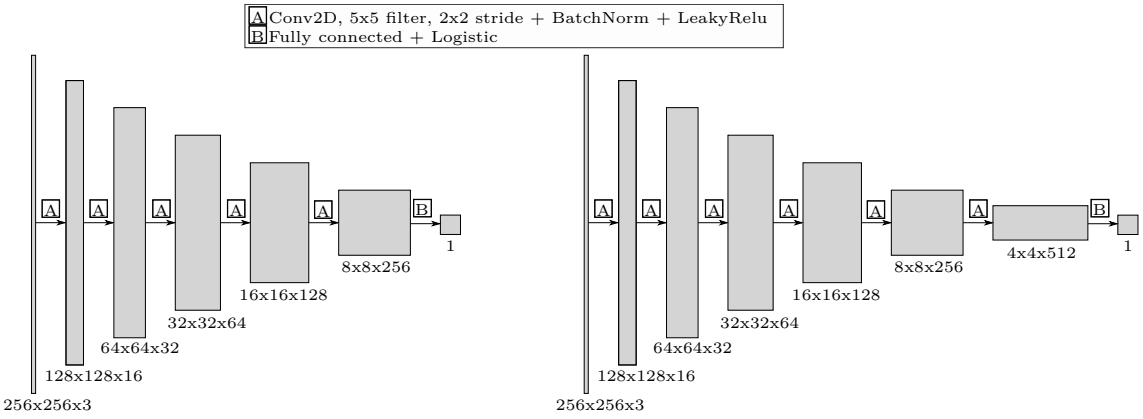


Figure 4.5: Architectures used for the discriminators in the GPND algorithm. Left: architecture for the Pro-SiVIC highway scenario dataset. Right: architecture for the Dr(eye)ve dataset.

Table 4.4: CAE architectures and training settings

Architecture		
Dataset	Pro-SiVIC highway scenario	Dr(eye)ve
k	5	5
s	2	2
n_{conv}	5	6
n_{fc}	1	1
c_1	16	16
c_z	512	512
Training settings		
Dataset	Pro-SiVIC highway scenario	Dr(eye)ve
η	$0.001 \rightarrow 0.0001$	$0.001 \rightarrow 0.0001$
Epochs	500	500
η change epoch	250	250
Mini-batch size	64	64
Optimizer	adam	adam
Weight initialization	xavier uniform	xavier uniform

Table 4.5: A confusion matrix, showing the relation between classifier prediction and true value of normal class membership

		Prediction	
		Positive	**Negative**
Actual value	**Positive**	True positive	False negative
	Negative	False positive	True negative

normalcy probability $S_{\text{normal}}(x)$, which is higher for normal samples, we simply use the probability complement as novelty score, so that $S_{\text{novelty}}(x) = 1 - S_{\text{normal}}(x)$. For a given threshold τ, all inputs for which $S_{\text{novelty}}(x) \geq \tau$ are classified as novelties, i.e., a positive result. All inputs for which $S_{\text{novelty}}(x) < \tau$ are classified as normal, i.e., a negative result. This allows us to define:

- true positives (TP): number of actual novelties correctly classified as novelties,
- true negatives (TN): number of actual normal samples correctly classified as normal,
- false positives (FP): number of actual normal samples wrongly classified as novelties,
- false negatives (FN): number of actual novelties wrongly classified as normal.

The total number of each of the above can be represented in the confusion matrix of a classifier, and is shown in Table 4.5. From this, we can define several other classification metrics. The true positive rate (TPR), also called recall, is defined as

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{4.1}$$

which in this context means the fraction of all actual novelties that were correctly detected. The false positive rate (FPR) is defined as

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}, \tag{4.2}$$

which in this context means the fraction of all actual normal samples that were wrongly classified as novelties. The precision is defined as

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{4.3}$$

which in this context means the fraction of all inputs classified as novelties that are actually novelties.

Results for all experiments in this thesis are presented in Section 4.3, using two types of binary classification evaluation curves: receiver operating characteristic (ROC) and precision-recall curve (PRC), as well as their corresponding area under curve measures: AUROC for the ROC and area under precision-recall curve (AUPRC) for the PRC.

The ROC curve is obtained by plotting the TPR against the FPR of an experiment for all possible threshold levels τ for the novelty score $S_{\text{novelty}}(x)$. Similarly, the PRC is obtained by plotting the precision against the recall for all possible values of τ. The AUROC measure can be viewed as the average probability that a positive sample, in this case a novelty, is also classified as a positive. This means that a perfect classifier will yield AUROC = 1, while the opposite case, predicting all

positives as negatives and vice versa, will yield AUROC = 0. A random baseline classifier, classifying any input as positive or negative with equal probability, would yield AUROC = 0.5.

Since the AUROC measure is independent of the threshold τ, it is convenient for comparison of different classifiers. A drawback with ROC is that the curve remains unchanged for unbalanced datasets, i.e., when the number of negative samples N is significantly larger than number of positive samples P, or vice versa. For a high N, the lowest threshold τ, corresponding to correctly classifying all positives, meaning FN = 0 and thereby TPR = 1, might still lead to a relatively low FP, resulting in a low FPR.

In such cases, the AUPRC measure is more suitable. AUPRC is also threshold independent, and furthermore, the baseline AUPRC value of a random classifier is equal to the fraction $P/(P + N)$. Since $P = N$ in this thesis, the baseline value is 0.5 for both AUROC and AUPRC.

Results are also presented as histograms of the novelty scores for each experiment. Histograms allow a more close examination of the difference between the scores for normal samples and the scores for novelty samples.

4.2.4 Hardware

All experiments, including training and testing of all models, were performed using a single Nvidia GeForce GTX 1080 Ti with 11GB memory.

4.3 Experimental Results

Results for all experiments, in terms of AUROC and AUPRC, are shown in Table 4.6. Plots of ROC curves and PRCs as well as histograms of S_{novelty} are shown in separate subsections for each combination of dataset and implemented algorithm. For visibility, ROC curves and PRCs are also plotted in separate windows for each type of novel scenario. In each plot, the classification performance using the respective AE reconstruction error as novelty score is shown, in addition to the algorithm specific novelty score. For the DSVDD algorithm, both the soft-boundary and one-class scores are provided. Note that the vertical scale is logarithmic in all histograms, to show scores of low frequency more clearly. To enable comparison between experiments, the novelty scores for each experiment were scaled and shifted to the range $[0, 1]$ before the creation of the corresponding histogram.

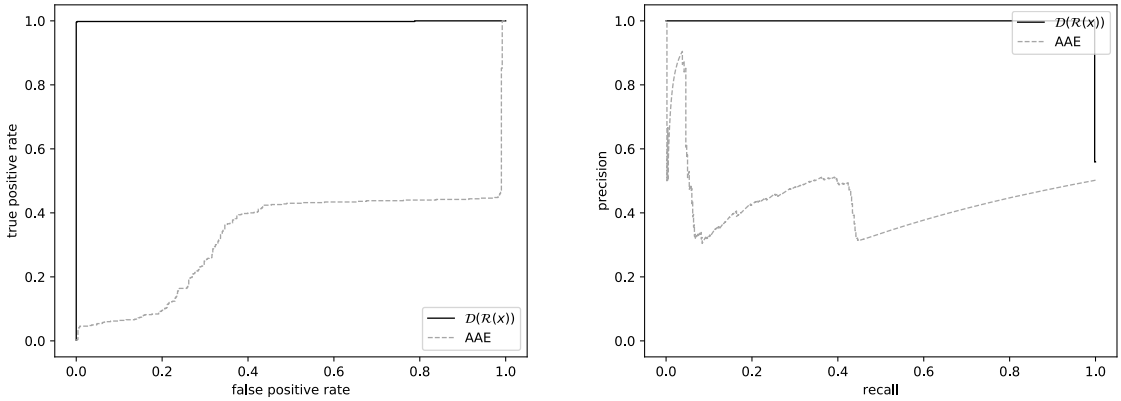
4.3.1 Results for Experiments on the Pro-SiVIC Highway Scenario Dataset

ALOCC

Classification results for the ALOCC algorithm on the Pro-SiVIC highway scenario dataset are shown in Figs. 4.6–4.7 for the novel weather scenario and Figs. 4.8–4.9 for the novel landscape scenario. For the novel weather scenario, the ALOCC novelty score $\mathcal{D}(\mathcal{R}(x))$ yields almost perfect separation of the two classes, with AUROC

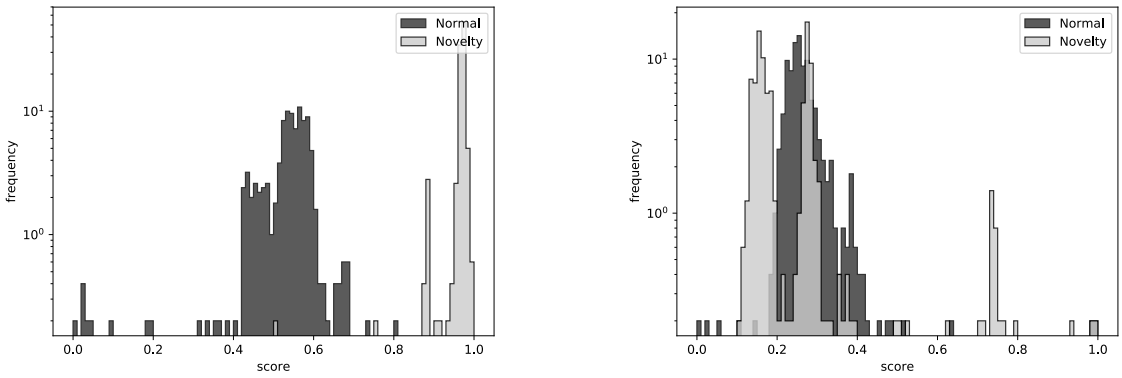
Table 4.6: ND performance metrics for all experiments

Dataset	Pro-SiVIC highway scenario				Dr(eye)ve			
Outlier type	Weather		Landscape		Weather		Landscape	
Metric / Algorithm	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC
ALOCC $\mathcal{D}(\mathcal{R}(x))$	0.998	0.999	0.999	0.999	0.498	0.748	0.507	0.741
ALOCC AAE	0.330	0.443	0.999	0.999	0.560	0.519	0.705	0.641
DSVDD soft-boundary	0.970	0.904	0.994	0.992	0.808	0.747	0.781	0.679
DSVDD one-class	0.977	0.926	0.992	0.990	0.807	0.747	0.781	0.680
DSVDD CAE	0.969	0.906	0.997	0.996	0.748	0.671	0.948	0.921
GPND $p_X(x)$	0.955	0.954	0.021	0.308	0.427	0.434	0.385	0.434
GPND AAE	0.516	0.531	0.543	0.530	0.514	0.499	0.486	0.497



(a) ROC curves. **(b)** PRCs.

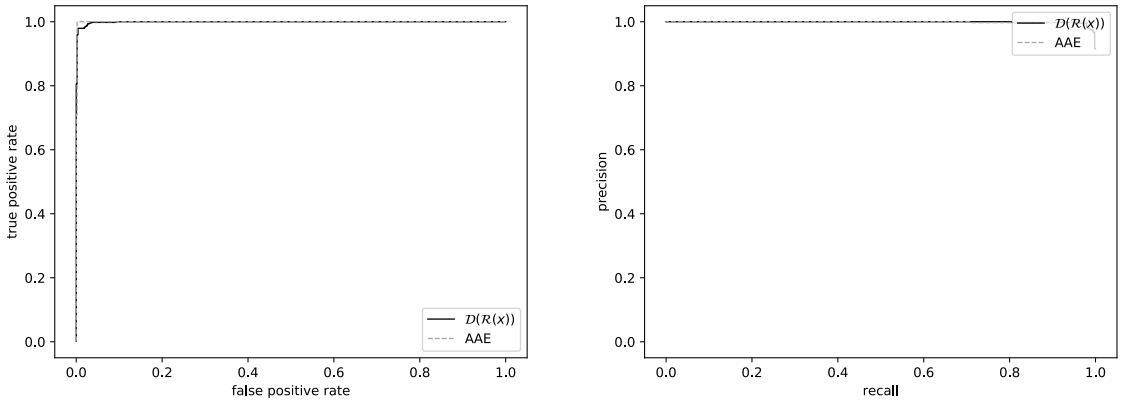
Figure 4.6: ROC and PRCs for the ALOCC algorithm, on the Pro-SiVIC highway scenario dataset, with unseen weather as novelties.



(a) Histograms displaying the distributions of $\mathcal{D}(\mathcal{R}(x))$ scores for normal class and novelties.

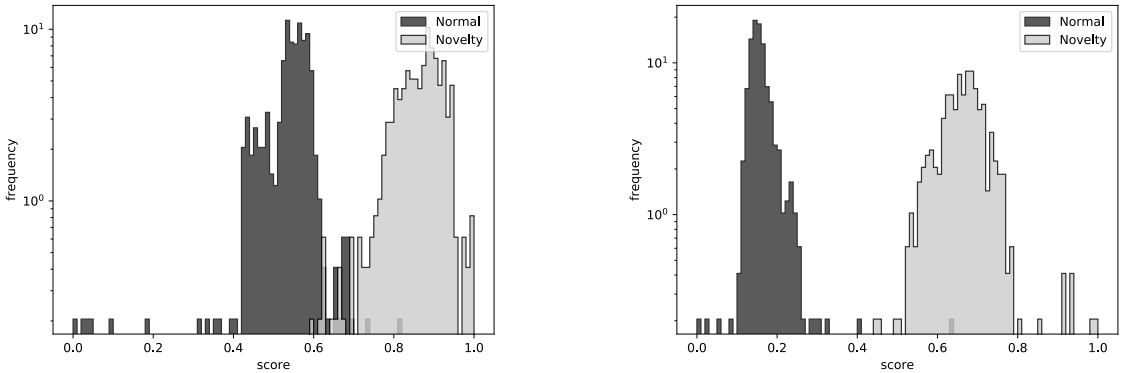
(b) Histograms displaying the distributions of AAE scores for normal class and novelties.

Figure 4.7: Histograms of novelty scores for the ALOCC algorithm, on the Pro-SiVIC highway scenario dataset, with unseen weather as novelties.



(a) ROC curves. **(b)** PRCs.

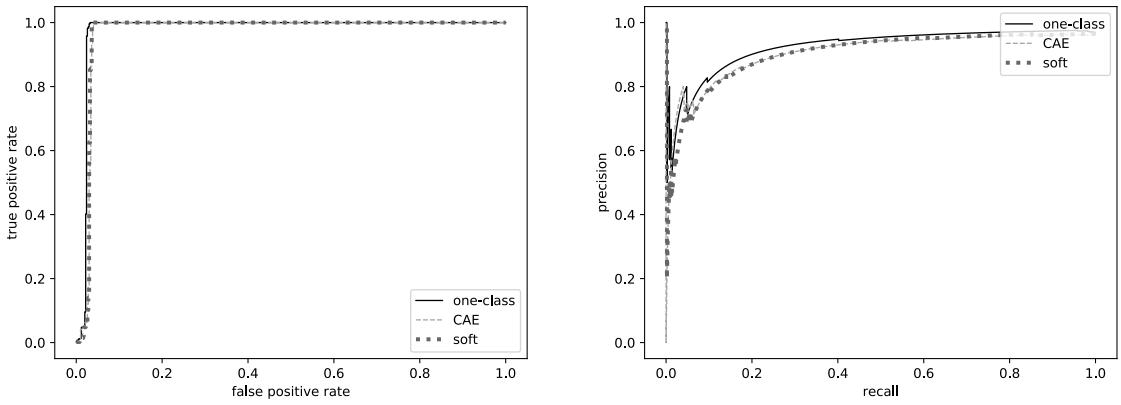
Figure 4.8: ROC and PRCs for the ALOCC algorithm, on the Pro-SiVIC highway scenario dataset, with unseen landscape as novelties.



(a) Histograms displaying the distributions of $\mathcal{D}(\mathcal{R}(x))$ scores for normal class and novelties.

(b) Histograms displaying the distributions of AAE scores for normal class and novelties.

Figure 4.9: Histograms of novelty scores for the ALOCC algorithm, on the Pro-SiVIC highway scenario dataset, with unseen landscape as novelties.



(a) ROC curves. **(b)** PRCs.

Figure 4.10: ROC and PRCs for the DSVDD algorithm, on the Pro-SiVIC highway scenario dataset, with unseen weather as novelties.

= 0.998 and AUPRC = 0.999. The distributions of AAE reconstruction errors are somewhat overlapping, yielding lower errors than the normal class for some novelties and higher errors for others. This is also reflected in the classification metrics AUROC = 0.330 and AUPRC = 0.443, which is worse than a random classifier.

For the novel landscape scenario, both the $\mathcal{D}(\mathcal{R}(x))$ score and reconstruction error yield near perfect separation of novelties.

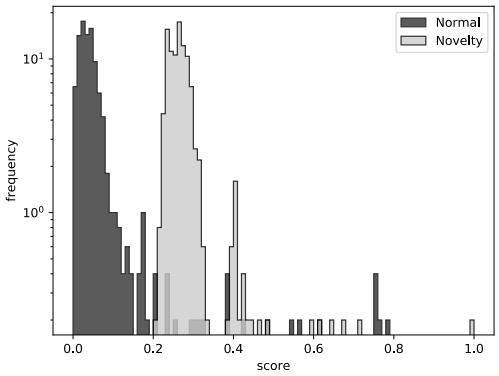
DSVDD

Classification results for the DSVDD algorithm on the Pro-SiVIC highway scenario dataset are shown in Figs. 4.10–4.11 for the novel weather scenario and Figs. 4.12–4.13 for the novel landscape scenario. The three score types, soft-boundary DSVDD, one-class DSVDD and CAE reconstruction error, yield very similar score distributions for the novel weather scenario, with the one-class classifier having a slight edge in terms of both AUROC and AUPRC. The separation is better in the novel landscape scenario for all three score types, but instead the CAE reconstruction error yields higher AUROC and AUPRC.

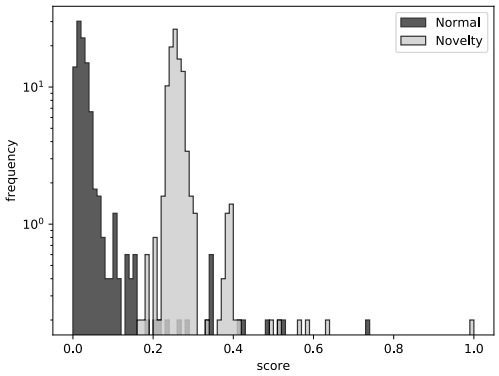
GPND

Classification results for the GPND algorithm on the Pro-SiVIC highway scenario dataset are shown in Figs. 4.14–4.15 for the novel weather scenario and Figs. 4.16–4.17 for the novel landscape scenario. For the novel weather scenario, the GPND novelty score $p_X(x)$ yields almost completely separated distributions, with AUROC= 0.955 and AUPRC = 0.954. The AAE reconstruction error yields heavily overlapping contributions, resulting in AUROC and AUPRC near those of a random classifier.

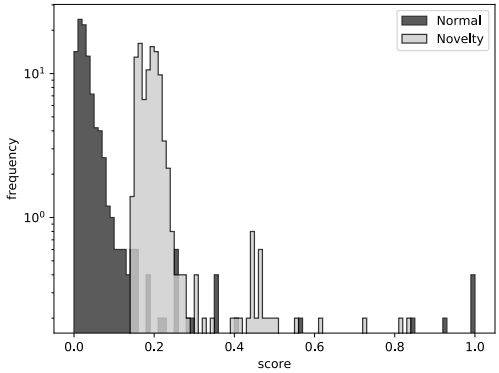
For the novel landscape scenario, the $p_X(x)$ score again yields good separation of distributions, however scoring most novelties lower than normal class samples, yielding AUROC = 0.021. The reconstruction error contains very little information also in this case, resembling a random classifier.



(a) Histograms displaying the distributions of soft-boundary DSVDD scores for normal class and novelties.

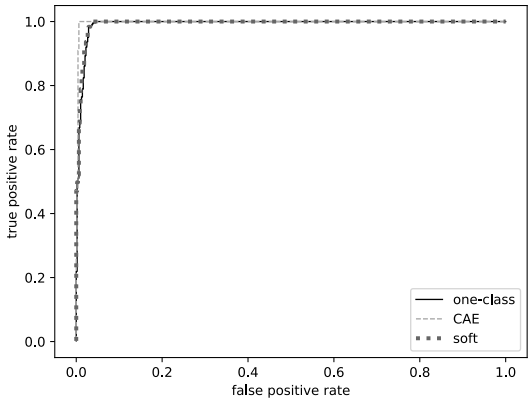


(b) Histograms displaying the distributions of one-class DSVDD scores for normal class and novelties.

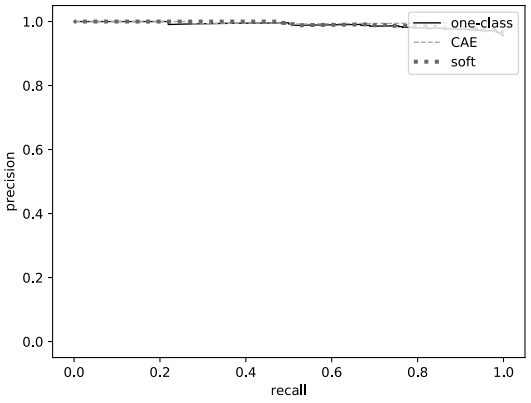


(c) Histograms displaying the distributions of CAE scores for normal class and novelties.

Figure 4.11: Histograms of novelty scores for the DSVDD algorithm, on the Pro-SiVIC highway scenario dataset, with unseen weather as novelties.

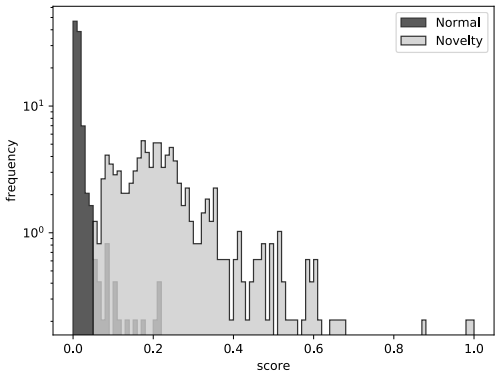


(a) ROC curves.

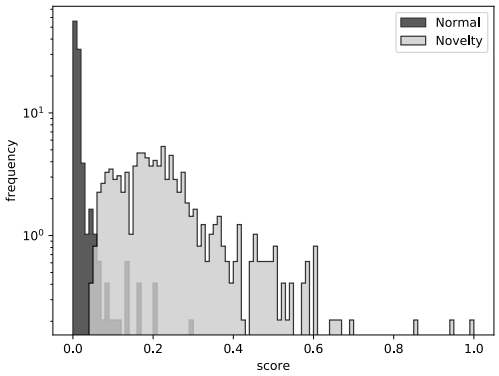


(b) PRCs.

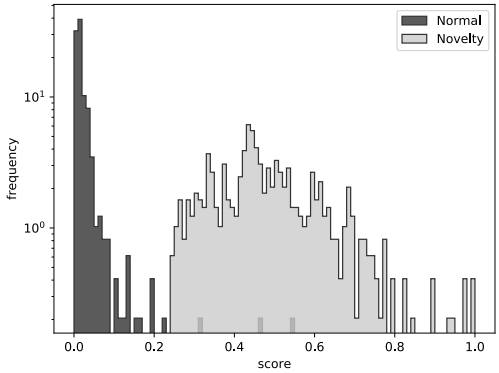
Figure 4.12: ROC and PRCs for the DSVDD algorithm, on the Pro-SiVIC highway scenario dataset, with unseen landscape as novelties.



(a) Histograms displaying the distributions of soft-boundary DSVDD scores for normal class and novelties.

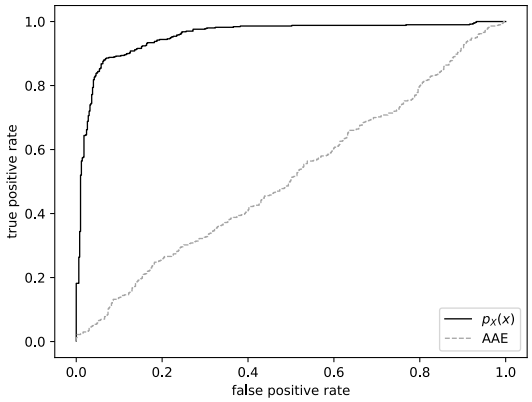


(b) Histograms displaying the distributions of one-class DSVDD scores for normal class and novelties.

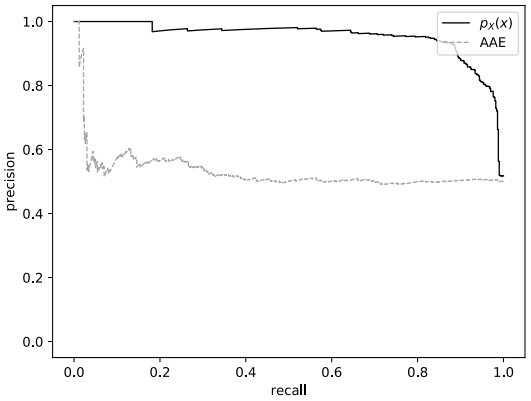


(c) Histograms displaying the distributions of CAE scores for normal class and novelties.

Figure 4.13: Histograms of novelty scores for the DSVDD algorithm, on the Pro-SiVIC highway scenario dataset, with unseen landscape as novelties.

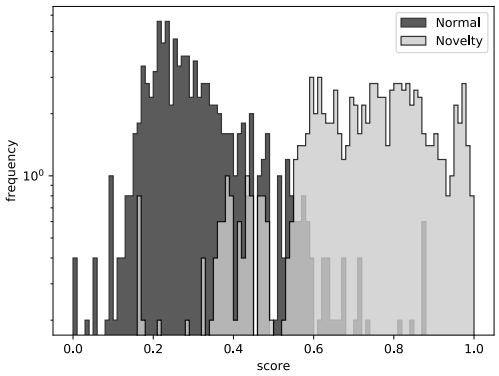


(a) ROC curves.

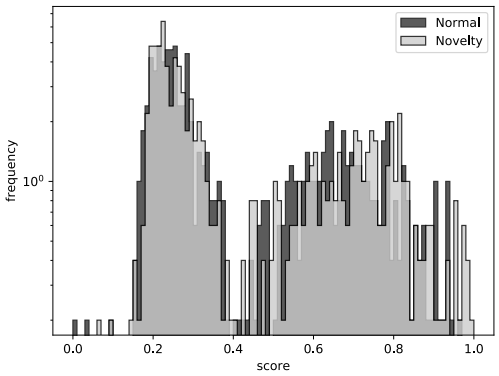


(b) PRCs.

Figure 4.14: ROC and PRCs for the GPND algorithm, on the Pro-SiVIC highway scenario dataset, with unseen weather as novelties.

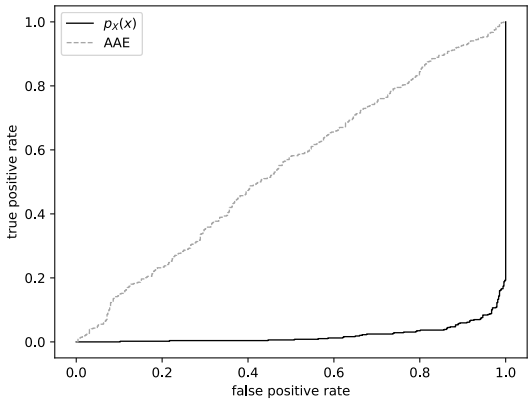


(a) Histograms displaying the distributions of $p_X(x)$ scores for normal class and novelties.

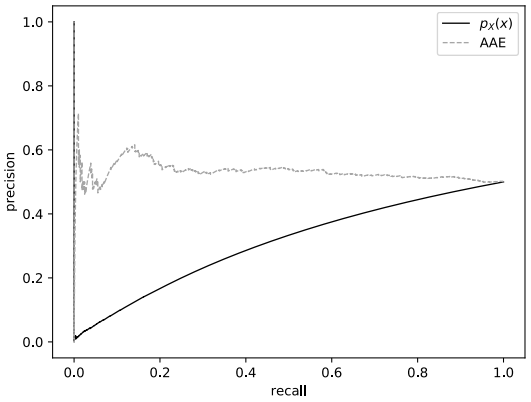


(b) Histograms displaying the distributions of AAE scores for normal class and novelties.

Figure 4.15: Histograms of novelty scores for the GPND algorithm, on the Pro-SiVIC highway scenario dataset, with unseen weather as novelties.

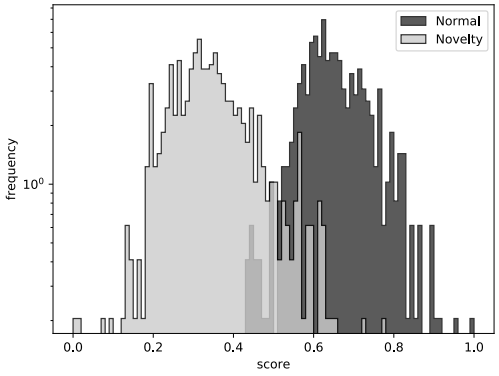


(a) ROC curves.

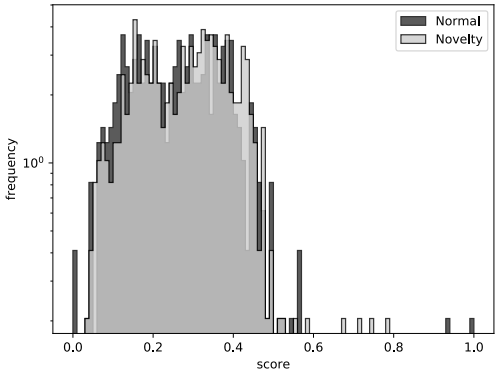


(b) PRCs.

Figure 4.16: ROC and PRCs for the GPND algorithm, on the Pro-SiVIC highway scenario dataset, with unseen landscape as novelties.

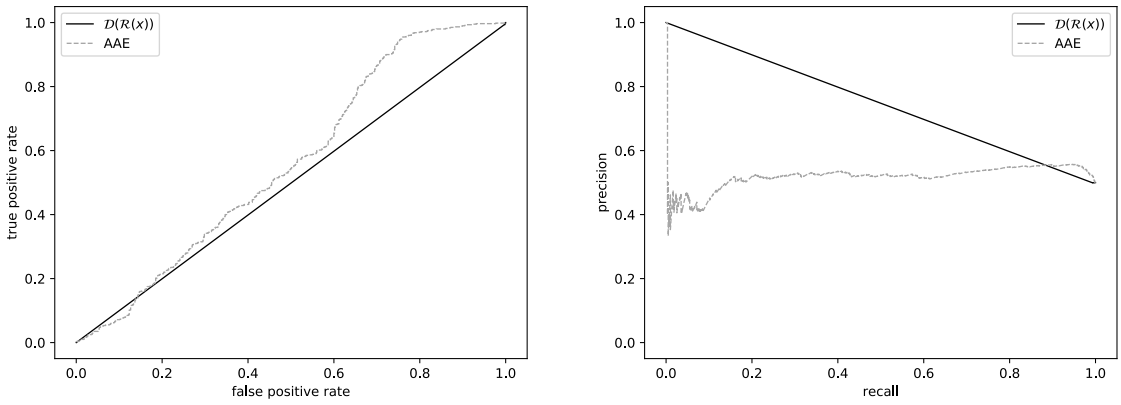


(a) Histograms displaying the distributions of $p_X(x)$ scores for normal class and novelties.



(b) Histograms displaying the distributions of AAE scores for normal class and novelties.

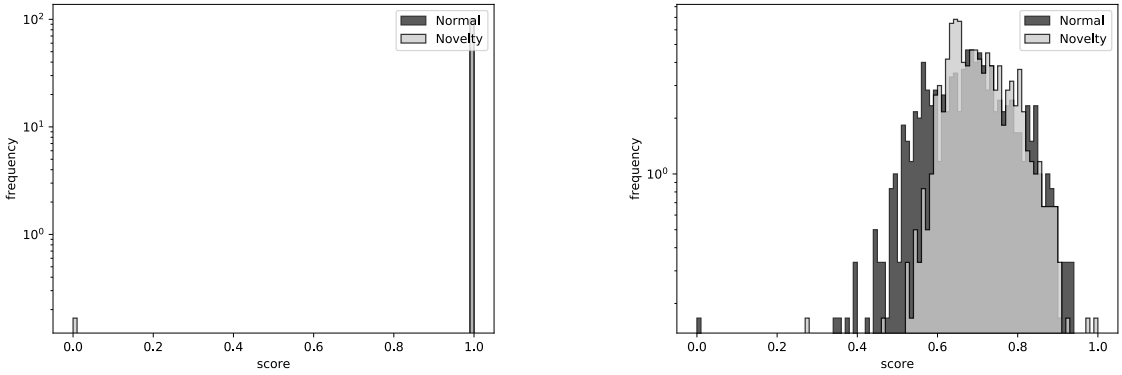
Figure 4.17: Histograms of novelty scores for the GPND algorithm, on the Pro-SiVIC highway scenario dataset, with unseen landscape as novelties.



(a) ROC curves.

(b) PRCs.

Figure 4.18: ROC and PRCs for the ALOCC algorithm, on the Dr(eye)ve dataset, with unseen weather as novelties.



(a) Histograms displaying the distributions of $\mathcal{D}(\mathcal{R}(x))$ scores for normal class and novelties.

(b) Histograms displaying the distributions of AAE scores for normal class and novelties.

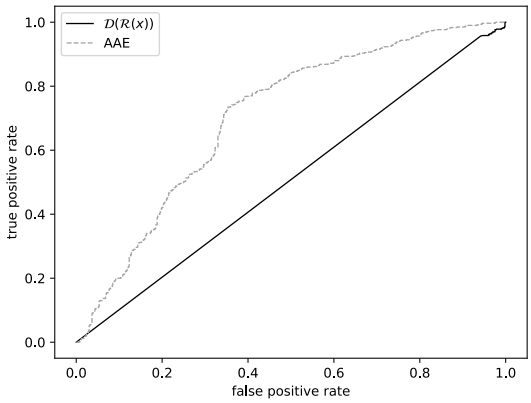
Figure 4.19: Histograms of novelty scores for the ALOCC algorithm, on the Dr(eye)ve dataset, with unseen weather as novelties.

4.3.2 Results for Experiments on the Dr(eye)ve Dataset

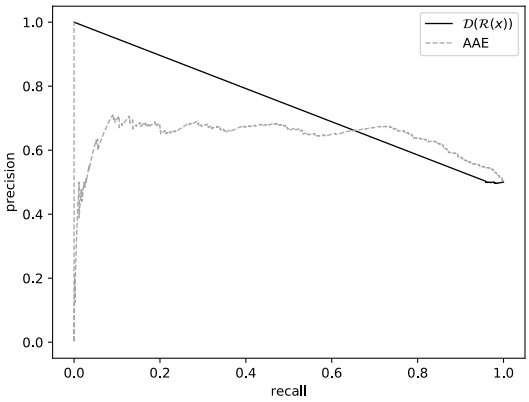
ALOCC

Classification results for the ALOCC algorithm on the Dr(eye)ve dataset are shown in Figs. 4.18–4.19 for the novel weather scenario and Figs. 4.20–4.21 for the novel landscape scenario. For the novel weather scenario, the ALOCC novelty score $\mathcal{D}(\mathcal{R}(x))$ is very similar for almost all samples, yielding nearly completely overlapping distributions and AUROC = 0.498. The AAE reconstruction error performs slightly better, but still does not manage to separate novelties from the normal class to a large extent.

For the novel landscape scenario, the results are very similar, with the $\mathcal{D}(\mathcal{R}(x))$ score being almost constant for all inputs, and the AAE reconstruction error containing little relevant information.

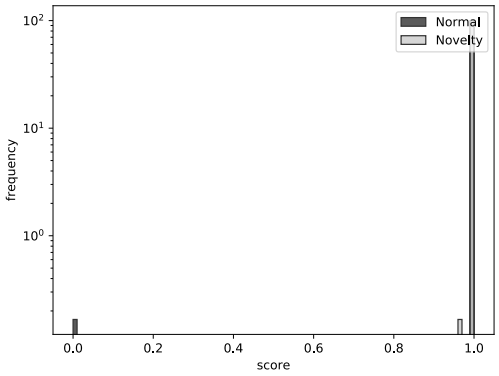


(a) ROC curves.

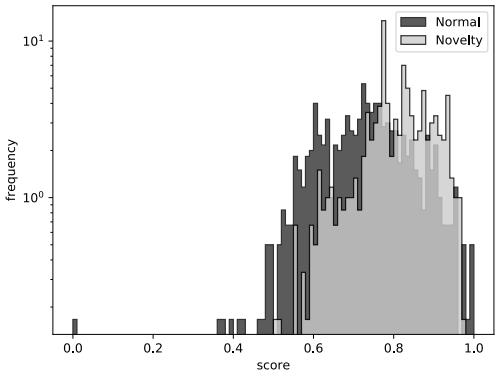


(b) PRCs.

Figure 4.20: ROC and PRCs for the ALOCC algorithm, on the Dr(eye)ve dataset, with unseen landscape as novelties.

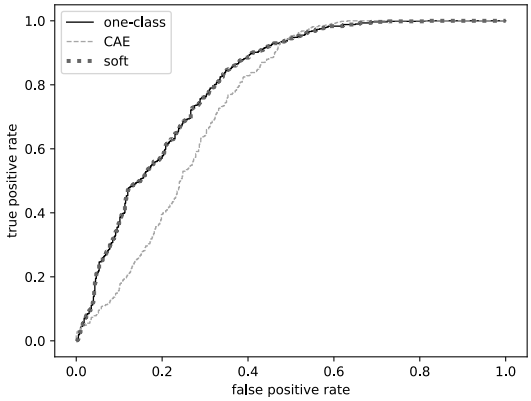


(a) Histograms displaying the distributions of $\mathcal{D}(\mathcal{R}(x))$ scores for normal class and novelties.

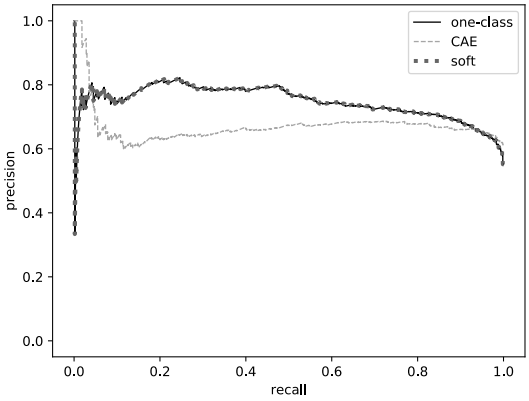


(b) Histograms displaying the distributions of AAE scores for normal class and novelties.

Figure 4.21: Histograms of novelty scores for the ALOCC algorithm, on the Dr(eye)ve dataset, with unseen landscape as novelties.

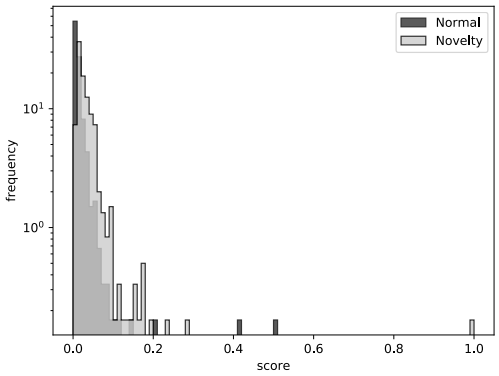


(a) ROC curves.

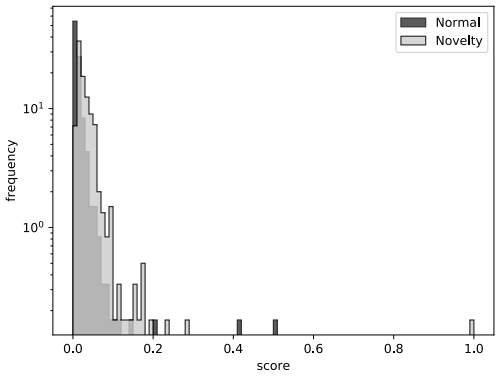


(b) PRCs.

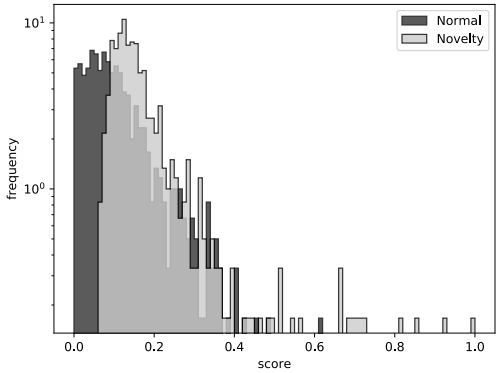
Figure 4.22: ROC and PRCs for the DSVDD algorithm, on the Dr(eye)ve dataset, with unseen weather as novelties.



(a) Histograms displaying the distributions of soft-boundary DSVDD scores for normal class and novelties.

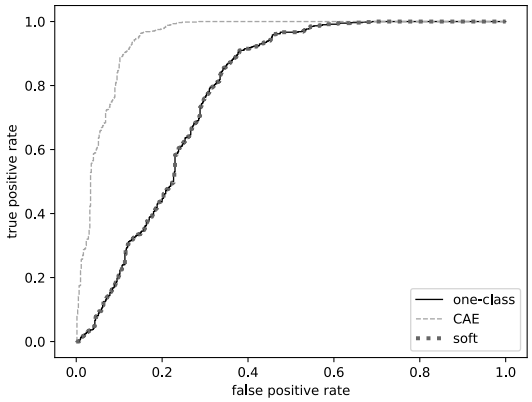


(b) Histograms displaying the distributions of one-class DSVDD scores for normal class and novelties.

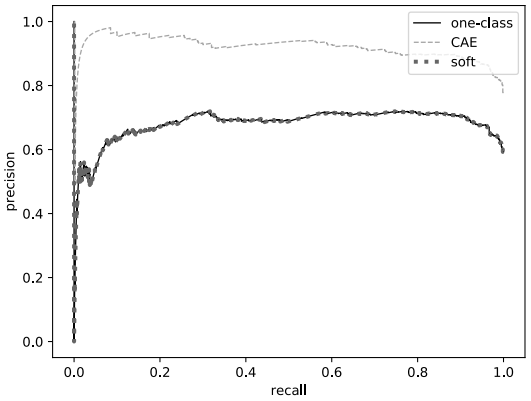


(c) Histograms displaying the distributions of CAE scores for normal class and novelties.

Figure 4.23: Histograms of novelty scores for the DSVDD algorithm, on the Dr(eye)ve dataset, with unseen weather as novelties.

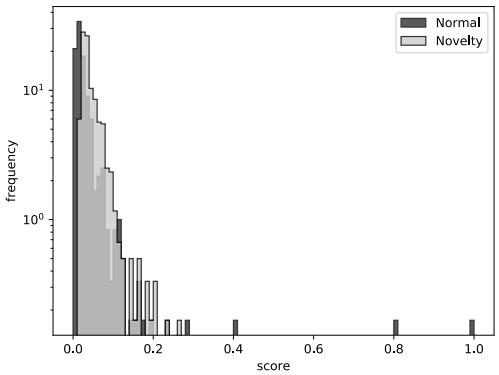


(a) ROC curves.

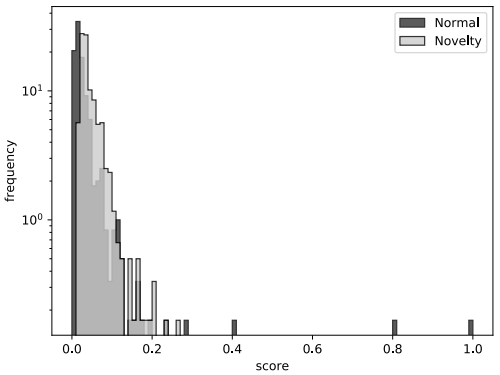


(b) PRCs.

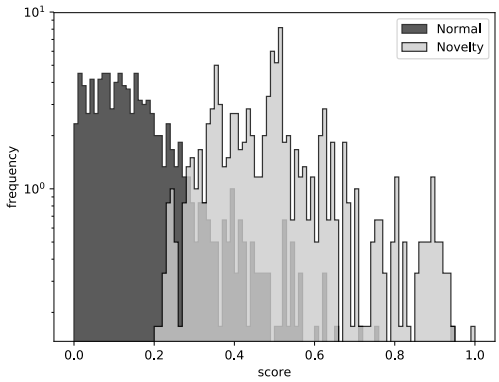
Figure 4.24: ROC and PRCs for the DSVDD algorithm, on the Dr(eye)ve dataset, with unseen landscape as novelties.



(a) Histograms displaying the distributions of soft-boundary DSVDD scores for normal class and novelties.

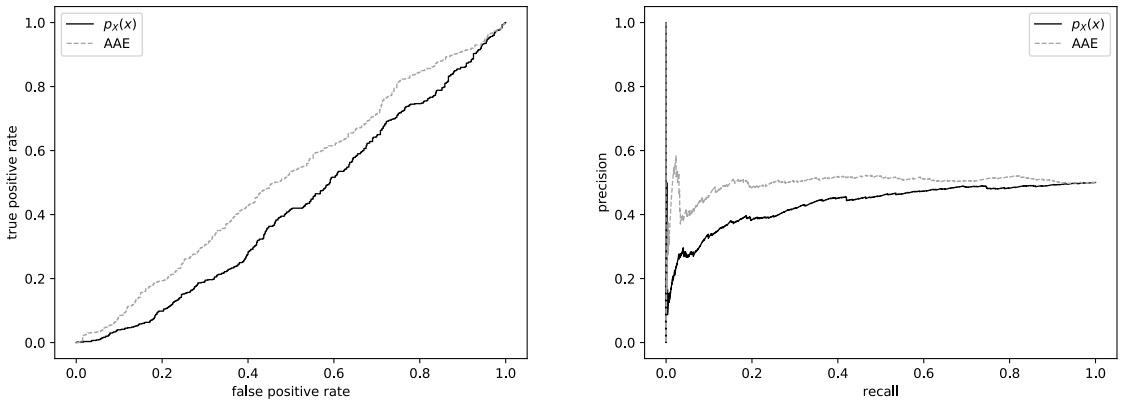


(b) Histograms displaying the distributions of one-class DSVDD scores for normal class and novelties.



(c) Histograms displaying the distributions of CAE scores for normal class and novelties.

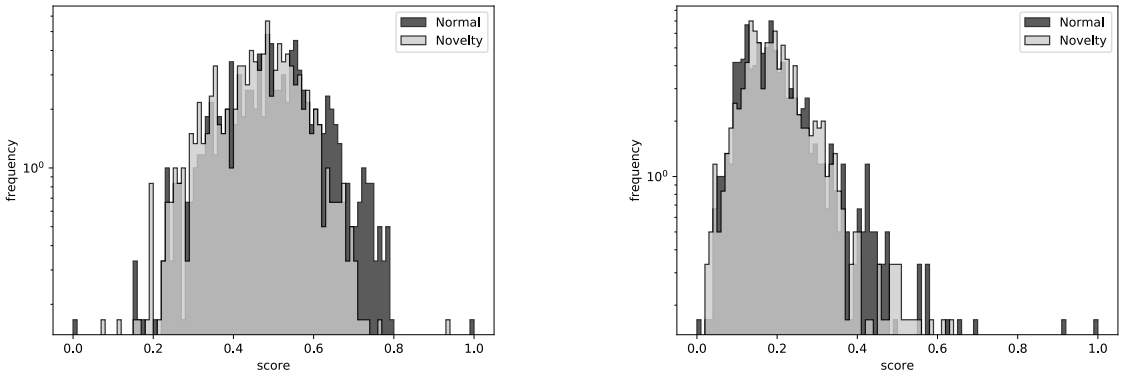
Figure 4.25: Histograms of novelty scores for the DSVDD algorithm, on the Dr(eye)ve dataset, with unseen landscape as novelties.



(a) ROC curves.

(b) PRCs.

Figure 4.26: ROC and PRCs for the GPND algorithm, on the Dr(eye)ve dataset, with unseen weather as novelties.



(a) Histograms displaying the distributions of $p_X(x)$ scores for normal class and novelties.

(b) Histograms displaying the distributions of AAE scores for normal class and novelties.

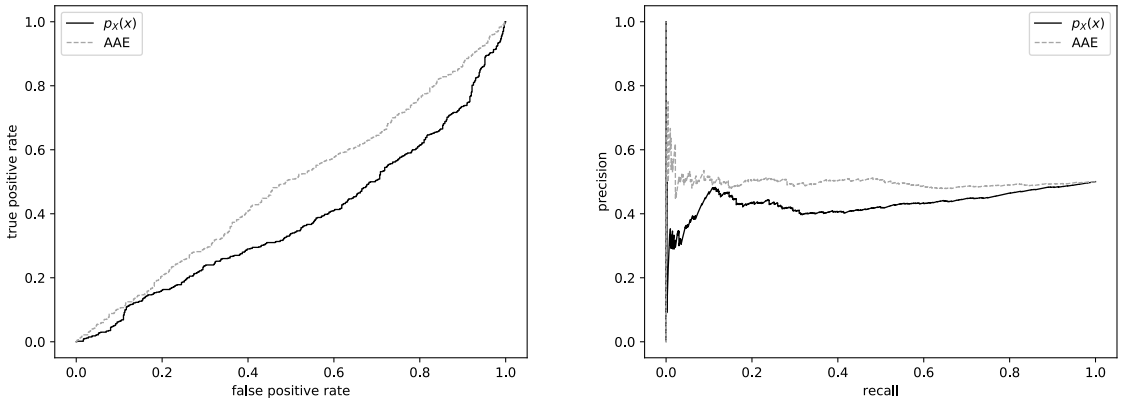
Figure 4.27: Histograms of novelty scores for the GPND algorithm, on the Dr(eye)ve dataset, with unseen weather as novelties.

DSVDD

Classification results for the DSVDD algorithm on the Dr(eye)ve dataset are shown in Figs. 4.22–4.23 for the novel weather scenario and Figs. 4.24–4.25 for the novel landscape scenario. The three DSVDD score types give similar results also for the Dr(eye)ve dataset. The soft-boundary DSVDD and the one-class DSVDD are almost indistinguishable from each other, for both the novel weather scenario and the novel landscape scenario. As seen in Figs. 4.22a–4.22b, both methods yield clearly better results than the CAE reconstruction error for the novel weather scenario, with AUROC ≈ 0.8, the highest out of all experiments on that scenario. For the novel landscape scenario, the reconstruction error yields the best results out of all experiments, with AUROC $= 0.948$ and AUPRC $= 0.921$.

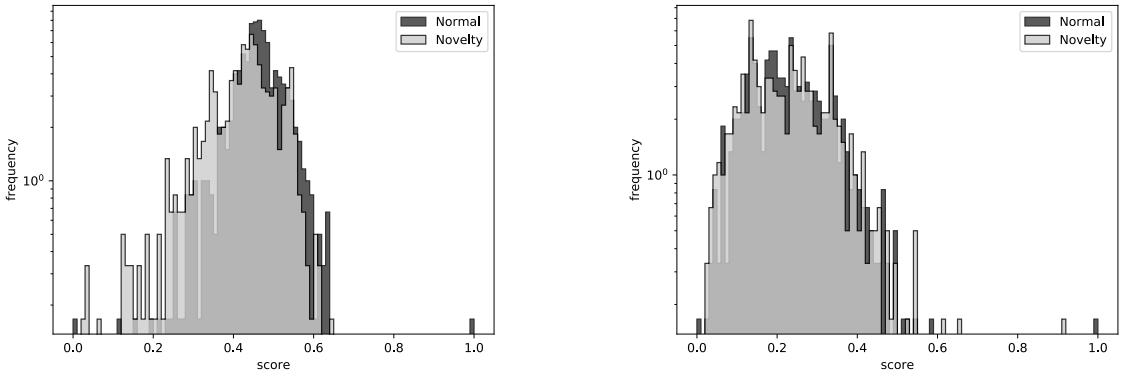
GPND

Classification results for the GPND algorithm on the Dr(eye)ve dataset are shown in Figs. 4.26–4.27 for the novel weather scenario and Figs. 4.28–4.29 for the novel



(a) ROC curves. **(b)** PRCs.

Figure 4.28: ROC and PRCs for the GPND algorithm, on the Dr(eye)ve dataset, with unseen landscape as novelties.



(a) Histograms displaying the distributions of $p_X(x)$ scores for normal class and novelties.

(b) Histograms displaying the distributions of AAE scores for normal class and novelties.

Figure 4.29: Histograms of novelty scores for the GPND algorithm, on the Dr(eye)ve dataset, with unseen landscape as novelties.

landscape scenario. For both novelty scenarios, both the GPND score $p_X(x)$ and the reconstruction error yield results close to or worse than a random classifier, with all values of AUROC and AUPRC in the range $[0.38, 0.52]$.

5
Discussion

In this section, the results of the thesis project are discussed, in terms of validity and relevance. Then, possible improvements to the experiments are discussed. Finally, possible further work is proposed.

5.1 Implications of Experiment Results

5.1.1 Comparison of Evaluated Algorithms

On the Pro-SiVIC highway scenario dataset, all algorithms managed to extract normal class features well enough to separate it from the novelty scenarios: using the original algorithm novelty score types, the lowest AUROC was 0.955 for the novel weather scenario and 0.021 for the novel landscape scenario, both yielded by the GPND algorithm $p_X(x)$ score. We note that AUROC = 0.021 is an almost perfectly bad score, i.e., if all predictions were inverted it would produce an AUROC of 0.979. This shows that, although the model learns relevant features, the GPND scoring method did not produce higher novelty scores for novelties, but rather the opposite. Since it assigns higher scores than the normal class for one type of novelties and lower scores for another, it is not safe to assume that it would perform well in a safety-critical situation, where novelties can come from a much larger distribution than those tested here.

For the more difficult Dr(eye)ve dataset, there is a notable difference between the three models, seen in Table 4.6: all the DSVDD score types yield higher AUROC and AUPRC than the other two algorithms. The only exception is that that the ALOCC score $\mathcal{D}(\mathcal{R}(x))$ gets the highest AUPRC for both novelty scenarios. The two DSVDD score types perform clearly better than the simple CAE reconstruction loss on the novel weather scenario, indicating that this type of algorithm is a good option to using AEs for ND. On the novel landscape scenario, the DSVDD CAE even more clearly outperforms the rest, producing AUROC and AUPRC of 0.948 and 0.921 compared with the second best 0.781 and 0.741, respectively. A possible reason that the reconstruction error works better in one novelty scenario, but not the other, is that the reconstruction error is computed pixel wise, which makes it sensitive to the background changing from trees and fields to buildings, as they have different colors and shapes. In contrast, the difference between sunny and rainy weather is marked by, e.g., clouds in the sky, where the color of clouds can be more similar to a bright sky, leading to a lower pixel by pixel difference and worse classification based on reconstruction error.

The superior performance of the DSVDD algorithm on the Dr(eye)ve dataset

coincides with the fact that the reconstruction error based classification results are consistently better for the DSVDD CAE than for the AAEs used in ALOCC and GPND. This leads to two observations: the first one is that an AE which is itself better for ND might yield better results also for other ND algorithms using the latent representation of that AE. The second observation is that in this case, the simple approach of optimizing a CAE with only the reconstruction loss, as for DSVDD, tends to be superior in comparison with using adversarial training objectives. A possible reason is that with the large image size used here, generating realistic reconstructions is sufficiently hard that using an adversarial loss for the CAE appears as noise and does not improve the model. A possible improvement to the ALOCC and GPND experiments could be varying the hyperparameter λ, thereby controlling the tradeoff between reconstruction loss and adversarial loss, to give more weight to the reconstruction loss in the initial part of training. In the experiments here, λ was simply left unchanged compared with the original source code of the respective algorithms.

No thorough comparison of the suitability for real time execution of the algorithms has been made, regarding either memory requirements or the testing time per frame. However, one clear difference between the three is that the GPND algorithm is considerably slower than the other two during testing. Computing the novelty score $1-p_X(x)$ of a single 256×256 pixel image took, on average, around 20s on the used hardware, which effectively disqualifies the current GPND implementation from use in real-time systems. The algorithm step which causes this is a singular value decomposition of the Jacobian matrix of f, which is part of the linearization process mentioned in Section 3.2.2. This step accounts for over 99.9% of the testing time. The Jacobian matrix is of size $m \times n$, where $m = 256^2 \times 3 = 196608$ is the image dimensionality and $n = 512$ is the latent space dimensionality. An explanation for this step taking a long time would be if the common Golub-Kahan algorithm [42], with a time complexity of $\mathcal{O}(mn^2)$ for $m > n$, was being used for computing the singular value decomposition.

5.1.2 Relation to Similar Work

Given that the Pro-SiVIC highway scenario dataset is relatively new, and that the Dr(eye)ve dataset is primarily aimed at tasks regarding driver gaze, there is not much earlier work using the same datasets for benchmarking similar types of algorithms, at least to the best of our knowledge. In [27], the authors benchmarked their algorithm for ND in image and video, autoregressive novelty detectors, on the Dr(eye)ve dataset. Differences between this thesis and [27], in addition to having different ND algorithms, include:

- their experiments are more extensive, in terms of varying both the normal class and the novelty samples,
- they rescale the Dr(eye)ve images to 160×256 pixels instead of 256×256, and
- they use video sequences of 16 frames instead of single frames.

Note that there is no result for which the data in normal and novelty data splits where the same in this thesis and [27], so the results are not directly comparable. Still, the setups of normal class versus novelty scenarios in [27] which are most

similar to the ones in this thesis are sunny weather versus all other weathers, and highway landscape versus all other landscapes. The corresponding setups in this thesis are the weather and landscape data splits, respectively, both described in Section 4.2.1. Autoregressive novelty detectors achieved an AUROC of 0.585 and 0.718, respectively, for those experiments. This is to be compared with 0.808 and 0.948, which were the corresponding best results for algorithms evaluated in this thesis.

5.2 Validity of Experiment Results

This section discusses the validity of the results in relation to drawbacks in how the experiments were performed. This is presented together with possible improvements of the experiments.

The main issue with the presented results is that they were not averaged over several independent runs of the experiments, which makes it possible that all results would differ to some extent if the experiments were reproduced. The reason for this is because of limited time: running all model training sessions and different tests took over 72 hours on the used hardware.

Another problem is that there are few test cases, meaning that there is only one type of normal class for each dataset, and two types of novel scenarios. This is partially caused by a lack of data, i.e., the annotations of the chosen datasets were limited to a few options per attribute. However, the main reason is, once more, limited time: since each normal class requires training separate models, the total time for all experiments is directly proportional to the number of tested normal classes.

Since each algorithm was reimplemented using the source code from the corresponding article, possible errors in one or several of the implementations may have caused biased results, even though the CAE architectures were intended to be identical, as outlined in Section 4.2.2. One such error was likely present in the implementation of the GPND algorithm. This is indicated by the fact that the CAE image reconstructions could not be made as accurate in the corresponding framework as with the other two algorithms, even when applying the same optimization objectives. This most likely affected the normal class modeling capability and, as a consequence, also the ND results using the GPND algorithm.

5.3 Further Work

Below, we list some ideas for further work along the lines of this thesis project.

- **Other algorithms**. Several of the articles reviewed in Section 3.2.1 present interesting approaches to ND, and should be evaluated in a way similar to the experiments performed here.
- **Other input formats**. All three algorithms evaluated here could easily be extended to handle video feed: it only adds a dimension to the input tensors. It would also be interesting to build models analyzing not just camera input, but input from an entire autonomous vehicle sensor setup.

- **Deeper networks**. Given more computational resources and/or time for experiments, the evaluated algorithms, or other AE based approaches, should be implemented with even deeper networks. The deepest CAE tested here had 7 layers (6 convolutional and 1 fully connected) in the encoder network, to be compared with famously successful networks such as VGG [43] and GoogleNet [3], using 16 and 22 layers, respectively. Deeper networks, along with video feed, giving a temporal dimension, would allow the models to extract even more detail and subsequently perform better in ND experiments.
- **Larger datasets**. A required next step towards a more realistic use case is to use normal classes with a wider distribution, such as the Berkeley DeepDrive dataset.

6
Conclusions

In this thesis, we have found that most state-of-the-art, unsupervised ND algorithms for image input employ CAEs in some way. Evaluating three such algorithms on two self-driving image datasets of different complexity shows that an approach performing CAE optimization by minimizing reconstruction loss performs better than models with identical ANN architecture, but that also employ adversarial optimization objectives. This does not prove that the use of adversarial losses is ineffective, but it indicates that it is not necessarily superior either. The overall ND results show that although several CAE approaches are promising, models which separate a larger fraction of novel inputs from normal samples are required for these algorithms to be used as safety cages for ANNs in real-world autonomous drive systems.

Bibliography

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[4] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[5] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arxiv*, 2016.

[6] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue *et al.*, "An empirical evaluation of deep learning on highway driving," *arXiv preprint arXiv:1504.01716*, 2015.

[7] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[8] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[9] C. Englund, B. Duran, and M. Borg, "Projectsmile ii - safety analysis and verification/validation of machine learning based systems," Oct 2017. [Online]. Available: https://www.viktoria.se/projects/smile-ii

[10] D. M. Tax and R. P. Duin, "Support vector data description," *Machine learning*, vol. 54, no. 1, pp. 45–66, 2004.

[11] L. Ruff, N. Goernitz, L. Deecke, S. A. Siddiqui, R. Vandermeulen, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *International Conference on Machine Learning*, 2018, pp. 4390–4399.

[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[13] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[14] Jefkine, "Backpropagation in convolutional neural networks," Sep 2016. [Online]. Available: https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

[15] B. Sahu, "A fractionally-strided convolution a.k.a transposed convolution," Sep 2018. [Online]. Available: https://beerensahu.wordpress.com/2018/04/10/pytorch-a-fractionally-strided-convolution-or-a-deconvolution/

[16] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[17] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[18] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[19] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

[20] S. Nayar, S. A. Nene, and H. Murase, "Columbia object image library (coil 100). department of comp," *Science, Columbia University, Tech. Rep. CUCS-006-96*, 1996.

[21] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving video database with scalable annotation tooling," *arXiv preprint arXiv:1805.04687*, 2018.

[22] S. Alletto, A. Palazzi, F. Solera, S. Calderara, and R. Cucchiara, "Dr (eye) ve: a dataset for attention-based tasks with applications to autonomous and assisted driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 54–60.

[23] "Esi pro-sivic - 3d simulations of environments and sensors," May 2016. [Online]. Available: https://www.esi-group.com/software-solutions/virtual-environment/virtual-systems-controls/esi-pro-sivictm-3d-simulations-environments-and-sensors

[24] M. Landgren and L. Tranheden, "Input verification for deep neural networks," Master's thesis, Chalmers University of Technology, 2018.

[25] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.

[26] R. Chalapathy, A. K. Menon, and S. Chawla, "Anomaly detection using one-class neural networks," *arXiv preprint arXiv:1802.06360*, 2018.

[27] D. Abati, A. Porrello, S. Calderara, and R. Cucchiara, "And: Autoregressive novelty detectors," *arXiv preprint arXiv:1807.01653*, 2018.

[28] R. Chalapathy, A. K. Menon, and S. Chawla, "Robust, deep and inductive anomaly detection," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2017, pp. 36–51.

[29] M. Ribeiro, A. E. Lazzaretti, and H. S. Lopes, "A study of deep convolutional auto-encoders for anomaly detection in videos," *Pattern Recognition Letters*, vol. 105, pp. 13–22, 2018.

[30] S. Pidhorskyi, R. Almohsen, D. A. Adjeroh, and G. Doretto, "Generative probabilistic novelty detection with adversarial autoencoders," *arXiv preprint arXiv:1807.02588*, 2018.

[31] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar, "Efficient gan-based anomaly detection," *arXiv preprint arXiv:1802.06222*, 2018.

[32] M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli, "Adversarially learned one-class classifier for novelty detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3379–3388.

[33] K. Lee, H. Lee, K. Lee, and J. Shin, "Training confidence-calibrated classifiers for detecting out-of-distribution samples," *arXiv preprint arXiv:1711.09325*, 2017.

[34] A. Dairi, F. Harrou, M. Senouci, and Y. Sun, "Unsupervised obstacle detection in driving environments using deep-learning-based stereovision," *Robotics and Autonomous Systems*, vol. 100, pp. 287–301, 2018.

[35] M. Gutoski, M. Ribeiro, N. M. R. Aquino, A. E. Lazzaretti, and H. S. Lopes, "A clustering-based deep autoencoder for one-class image classification," in *Computational Intelligence (LA-CCI), 2017 IEEE Latin American Conference on*. IEEE, 2017, pp. 1–6.

[36] C. You, D. P. Robinson, and R. Vidal, "Provable selfrepresentation based outlier detection in a union of subspaces," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1–10.

[37] J. T. Andrews, T. Tanay, E. J. Morton, and L. D. Griffin, "Transfer representation-learning for anomaly detection," 2016.

[38] P. Perera and V. M. Patel, "Learning deep features for one-class classification," *arXiv preprint arXiv:1801.05365*, 2018.

[39] P. Smagghe, J.-L. Buessler, and J.-P. Urban, "Novelty detection in image recognition using irf neural networks properties." in *ESANN*, 2013.

[40] A. Vasilev, V. Golkov, I. Lipp, E. Sgarlata, V. Tomassini, D. K. Jones, and D. Cremers, "q-space novelty detection with variational autoencoders," *arXiv preprint arXiv:1806.02997*, 2018.

[41] L. Ruff, "Deep-svdd," https://github.com/lukasruff/Deep-SVDD, 2018.

[42] G. H. Golub, "Least squares, singular values and matrix approximations," *Aplikace matematiky*, vol. 13, no. 1, pp. 44–51, 1968.

[43] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.