CHALMERS
UNIVERSITY OF TECHNOLOGY

Uncertainty-Aware Weather-Related Road Surface Condition Classification

Master's thesis in Complex Adaptive Systems

JAKOB BERGMAN

ERMAN KULAGLIC

DEPARTMENT OF PHYSICS

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Uncertainty-Aware Weather-Related Road Surface Condition Classification

JAKOB BERGMAN
ERMAN KULAGLIC

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2026

Uncertainty-Aware Weather-Related Road Surface Condition Classification
JAKOB BERGMAN
ERMAN KULAGLIC

Supervisor: Pontus Andersson, Klimator
Examiner: Daniel Midtvedt, Department of Physics, University of Gothenburg

Master's Thesis 2026
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: A wet winter road through the forest into the mountains.
Photo by Vince Traveller, licensed under CC BY 2.0 [1].

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Uncertainty-Aware Weather-Related Road Surface Condition Classification
JAKOB BERGMAN
ERMAN KULAGLIC
Department of Physics
Chalmers University of Technology

Abstract

Autonomous vehicles rely on collaborative systems for decision-making during operation, one of which is machine learning algorithms. Using cameras and sensors mounted on the car, these algorithms can analyse captured data to interpret surroundings, enabling it to make informed and safe driving decisions. The algorithms are trained on data gathered in the field. In real-time use, samples that do not belong to the training distribution, referred to as Out-Of-Distribution (OOD) samples, can occur, which the algorithm then confidently uses to make driving decisions. In a worst-case scenario, this can lead to injuring passengers or property. The primary objective of this thesis is to implement and analyse methods for detecting OOD samples in a road surface condition model. The machine learning algorithm was a convolutional neural network, trained to detect dry, wet and snowy road surface conditions. The methods used to detect samples were Maximum Softmax Probability, Energy-Based OOD detection, Outlier Exposure, Virtual Outlier Synthesis, Rectified Activations, Deep Nearest Neighbour, and Virtual Logit Matching. Evaluation of the methods involved three datasets - Cars, Slush and Glare - constructed from the same database as the training data, thus considered as near OOD datasets. Additionally, two publicly available datasets, CIFAR-10 and Texture were used as far OOD datasets. Deep Nearest Neighbour and Virtual Logit Matching performed the best, achieving near-perfect results, when evaluated on the far OOD datasets. When tested on the cars dataset, Virtual Logit Matching exhibited a notable deviation in performance compared to the other methods, although it still performed poorly. On the rest of the near OOD data, the detectors did not outperform the two baseline detectors, Energy-Based OOD detection and Maximum Softmax Probability.

Keywords: Artificial Neural Networks, Convolutional Neural Network, Machine Learning, Out-Of-Distribution Detection, Road Climatology

Acknowledgements

We would like to thank Klimator for the opportunity to work with such an interesting project and Daniel Midtvedt for being our Examiner.
We also want to thank our supervisor Pontus Andersson, who has been answering our questions and provided feedback in our work.

And a special thanks to all friends and family who have helped us through our years at Chalmers and during this thesis. Without your support, all of this would not have been possible.

Jakob Bergman & Erman Kulaglic, Gothenburg, May 2025

Use of AI tools

This thesis have used the AI tool ChatGPT for inspiration during the writing of code and text for the report [2].
The AI tool Grammarly was used to refine already written text to improve the writing-style and language in the thesis [3].

The generated text or code was never trusted blindly or used directly in the thesis or programs. Instead, the text or code was thoroughly analysed and rewritten by the authors before being implemented in the thesis or programs.

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AUROC	Area Under the Receiver Operating Characteristic curve
CNN	Convolutional Neural Network
d-kNN	Deep Nearest Neighbour
FN	False Negative
FP	False Positive
FPR	False Positive Rate
FPR95	False Positive Rate at 95% true positive rate
ID	In-Distribution
MSP	Maximum Softmax Probability
OE	Outlier Exposure
OOD	Out-Of-Distribution
ReAct	Rectified Activations
ResNet	Residual Neural Network
ROC	Receiver Operating Characteristic
RSC	Road Surface Condition
TN	True Negative
TP	True Positive
TPR	True Positive Rate
ViM	Virtual logit Matching
VOS	Virtual Outlier Synthesis

Contents

List of Figures

List of Tables

1

Introduction

Most vehicles on the road today are still manually driven and rely heavily on the driver's judgment, awareness, and decision-making skills. However, we are experiencing a paradigm shift as we move toward an era of autonomous vehicles. For this transition to succeed, autonomous vehicles must take greater responsibility for ensuring safety. A key aspect of this shift is the ability to identify adverse road surface conditions (RSC), such as water, snow, and ice, to reduce the risk of accidents. In Sweden, 19% of traffic fatalities occur on roads affected by snow, frost, ice, or slush, which is a higher rate than observed in the European Union [4]. This statistic underscores the importance of identifying RSC.

Detecting RSC enables an autonomous vehicle to adjust its control strategy based on road conditions. Real-time monitoring equipment facilitates adaptive decision-making, allowing key parameters such as speed, following distance, and other control systems to be modified dynamically to maintain safety under varying conditions. For example, a low-friction surface may require a more cautious braking approach compared to a high-friction surface. Thus, developing systems that provide information about the current RSC is crucial. In autonomous vehicles, safety is best achieved through collaboration among multiple systems, one of which involves computer vision [5].

In the context of autonomous vehicles, computer vision allows the car to see and analyse its surroundings with visual information from images, similar to how humans use their eyes. One effective way to process these images is through deep learning, a type of machine learning that uses artificial neural networks to learn from data. Among these, Convolutional Neural Networks (CNNs) have been proven to be highly efficient in image classification tasks because they automatically identify essential features that distinguish between different classes [6]. Features such as edges, textures and surface patterns are learned directly from the pixel values of the images to classify an image into its corresponding class. By providing the network with labelled images for training, it can take new images as input, and output probabilities for each possible class it has been trained to recognise. These predictions can guide the vehicle's decision-making and control systems to adjust driving behaviour based on the information received from the network.

However, image classification models rely entirely on the data they have been trained on. Each prediction is made by comparing the input with what the network has learned during training. If the model encounters unexpected or unfamiliar inputs,

classification will occur within the known categories, leading to incorrect classification. For example, a model trained on wet and dry RSCs should not classify a snowy road condition as wet or dry. Ideally, the model should be able to tell if an image falls outside the training distribution and raise an alert, indicating that it cannot confidently assign the condition to the known classes. This sort of data is said to be Out-Of-Distribution (OOD) [7].

A straightforward way to detect OOD samples is by analysing the softmax probability distribution, a result from the neural network's final activation function. A neural network should output a high probability for one class for In-Distribution (ID) samples and equal probabilities for all classes, resulting in low probabilities for OOD samples. One could flag the sample as OOD if it is below some probability threshold for the maximum probability class. It has been shown that this is not always the case, and high-confidence predictions might not be reliable [8]. To address this issue, ongoing research has explored various methods to improve the reliability of OOD detection. These methods aim to provide more reliable measures of uncertainty beyond simply relying on softmax probability distribution.

This project is in collaboration with Klimator, a company that specialises in detecting and forecasting different weather-related RSCs.

1.1 Aim

The main objective of this thesis is to implement and analyse OOD detection methods in RSC classification. An ideal detector should be able to find OOD samples or distribution shifts in the RSC data.

Klimator is interested in integrating OOD detection methods into their classification models to enhance RSC classification reliability.

1.2 Objectives

The work is structured into three distinct phases to develop an uncertainty-aware deep learning model for RSC prediction. The initial phase focuses on a literature review of OOD detection and its associated terminology. Various methods will be examined and compared to gain a theoretical understanding and an intuition for their potential in the specific context of this work. Based on the initial analysis, the most promising methods will be selected for further study. An additional component of this phase involves analysing the available data from the company to determine its characteristics and suitability for the task. This involves specifying which subsets of the data should be used as ID and OOD.

The second phase focuses on constructing ID and OOD datasets, implementing deep learning models and OOD detection methods. This involves preprocessing the raw data into suitable formats, training the deep learning model, and integrating the selected OOD detection methods into the model architecture. To ensure the correctness and reliability of the implementation, similar existing work in the field will

be replicated as a validation step. This replication verifies the proposed methods' functionality and establishes a solid foundation for further experimentation.

The third phase focuses on evaluating the OOD detection methods across the different datasets. This involves analysing each method's performance, with the aim of gaining deeper insight into their underlying mechanisms and understanding why certain methods perform better on specific datasets. The methods will be compared to assess the strengths and limitations of each approach, thereby concluding how practical they could be in a real-world application.

1.3 Limitations

This thesis's primary dataset comes from Klimator's database and consists of sparsely annotated RSC estimates. Each estimate is provided with an associated probability, and while some misclassification may occur, no extensive effort will be made to correct these. Therefore, they will be treated as ground truth.

Dry, wet, and snowy RSC images will be the only data considered as ID; hence, these will be the three classes used when training the model.

Training models can be both time-consuming and computationally demanding, with model performance heavily influenced by the choice of architecture and hyperparameters. Therefore, only one model architecture will be used to balance performance with efficiency.

2

Theory

The upcoming chapter will present the key concepts and theories related to the thesis. First, the underlying theory of the model architecture will be introduced, followed by the theoretical foundations of the OOD detectors. Finally, the metrics used to evaluate these OOD detectors will be presented.

2.1 Artificial Neural Network

An artificial neural network is a machine-learning model inspired by the structure and function of the human brain. It is composed of layers of interconnected nodes, called neurons, which are usually organised into input, hidden, and output layers. Each neuron receives inputs from neurons in the previous layer and sends its output to the neurons in the following layer through an activation function. The network is trained by adjusting the strengths of the connections between neurons, known as weights and biases, allowing it to perform various information-processing tasks. Figure 2.1 shows a visual representation of this structure.

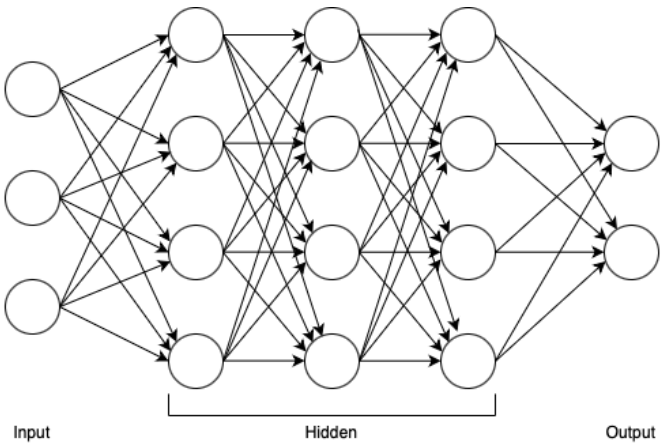


Figure 2.1: Feed-forward network with three hidden layers. The input layer consists of three neurons, each hidden layer consists of four neurons and the output layer consists of two neurons.

2.1.1 Supervised Training

Supervised learning refers to algorithms that use labelled datasets to train machine-learning models. The training process in neural networks usually follows the steps

1. Forward feed
2. Loss computation
3. Backward feed
4. Parameter update

2.1.1.1 Forward Feed

A forward feed is the process where input data passes through the input layer to the output layer. When the neural network has been trained on a given task, a forward feed is used to make its prediction.

Suppose the input layer L has N neurons and the next layer $L+1$ has M neurons. The values of the input neurons are denoted as $x_i^{[L]}$ and the next layer neurons are denoted as $x_j^{[L+1]}$, the connection between the neurons is denoted with some weight $w_{ij}^{[L]}$ and bias $b_j^{[L]}$ where $i \in 1, ..., N$ and $j \in 1, ..., M$. The activation function takes some function σ. The values for the neurons in the next layer are computed as

$$x_j^{[L+1]} = \sigma(\sum_i w_{ij}^{[L]} x_i^{[L]} + b_j^{[L]})$$

This computation is repeated for all layers. The network's output is the values in the last layer, denoted as logits [9].

The activation function allows the network to learn nonlinear relations in the data. Without nonlinear activation functions, the layers would reduce to a simple linear transformation. When choosing an activation function, it is important that the function is differentiable because neural networks are often trained by using gradient-based optimisation, such as backpropagation. A common activation function used is the Rectified Linear Unit (ReLU). It helps avoid gradients vanishing through the network, a common problem in backpropagation. The ReLU activation function is defined as

$$\text{ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Another activation function that is commonly used is the softmax probability function. This activation function is often used at the end of a network to convert the logits into a probability distribution. Given a vector of logits $\boldsymbol{l} = [l_1, l_2, ..., l_C]$, where C is the number of classes, the softmax function is defined as

$$\text{softmax}(\boldsymbol{l})_i = \frac{e^{l_i}}{\sum_{i=1}^{C} e^{l_i}} \tag{2.1}$$

2.1.1.2 Loss Function

The loss function is used to give a mathematical expression of a model's classification performance. This is done through a computation by comparing the model's prediction of a sample with its label. The output serves not only as a measurement of how well a model performs but also as a guide to optimising the parameters. Various loss functions exist, each suited for different objectives. For classification tasks, a common loss function is the cross-entropy loss. The cross-entropy loss is defined as

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{C} y_i \log\left(\hat{y}_i\right) \tag{2.2}$$

where C is the number of classes, $\boldsymbol{y}$ is a vector containing the true labels and $\hat{\boldsymbol{y}}$ is the predictions from the model [9].

2.1.1.3 Backwards Feed

The backwards feed, also known as backpropagation, is the process by which the network learns how to adjust its parameters to improve the model's performance. It uses the chain rule to calculate the gradient of the loss function with respect to each weight and bias.

Consider a neural network composed of N layers. Each layer does the following computations in a forward feed

$$\begin{aligned}\boldsymbol{z}^{[L]} &= \boldsymbol{W}^{[L]}\boldsymbol{a}^{[L-1]} + \boldsymbol{b}^{[L]} \\ \boldsymbol{a}^{[L]} &= \sigma(\boldsymbol{z}^{[L]})\end{aligned}$$

where $\boldsymbol{W}^{[L]}$ is the weight matrix at layer L, $\boldsymbol{a}^{[L]}$ is the activation vector at layer L, $\boldsymbol{b}^{[L]}$ is the bias vector at layer L and σ is some activation function [9]. Let the loss function for the neural network take some function $\mathcal{L}$. Define an error term at each layer and apply the chain rule

$$\boldsymbol{\delta}^{[\boldsymbol{L}]} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}^{[L]}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{a}^{[L]}} \frac{\partial \boldsymbol{a}^{[L]}}{\partial \boldsymbol{z}^{[L]}}$$

For the output layer, N, the error term is defined as

$$\boldsymbol{\delta}^{[N]} = \nabla_a \mathcal{L} \odot \sigma'(\boldsymbol{z}^{[N]})$$

and for the hidden layer, the error term is defined as

$$\boldsymbol{\delta}^{[L]} = (\boldsymbol{W}^{[L+1]^\intercal}\boldsymbol{\delta}^{[L+1]}) \odot \sigma'(\boldsymbol{z}^{[N]})$$

where $\odot$ is the Hadamard product (element-wise product). The gradients of the loss with respect to the weights or biases can be written as

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^{[L]}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}^{[L]}} \frac{\partial \boldsymbol{z}^{[L]}}{\partial \boldsymbol{W}^{[L]}} = \boldsymbol{\delta}^{[L]} \frac{\partial \boldsymbol{z}^{[L]}}{\partial \boldsymbol{W}^{[L]}} = \boldsymbol{\delta}^{[L]} \boldsymbol{a}^{[L-1]^\intercal}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{b}^{[L]}} = \boldsymbol{\delta}^{[L]}$$

2.1.1.4 Parameter Update

The parameter update for the weights and biases can be done in multiple ways. The most common method is gradient descent, which uses the gradients of the loss with respect to the weights or biases to perform the updates [10]. This can be written as

$$\begin{aligned} \boldsymbol{W}^{[L]}_{new} &= \boldsymbol{W}^{[L]} - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^{[L]}} \\ \boldsymbol{b}^{[L]}_{new} &= \boldsymbol{b}^{[L]} - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{b}^{[L]}} \end{aligned}$$

where η is the learning rate.

Another common method is the Adam optimiser, which leverages the advantages of the other methods, AdaGrad and RMSProp. Let θ be the parameter to be optimised over. The objective function is the loss function $\mathcal{L}$. Then, at each timestep t, the first and second moment estimates are updated with

$$\begin{aligned} \boldsymbol{m}_t &= \beta_1 \cdot \boldsymbol{m}_{t-1} + (1 - \beta_1) \nabla_\theta \mathcal{L} \\ \boldsymbol{v}_t &= \beta_2 \cdot \boldsymbol{v}_{t-1} + (1 - \beta_2)(\nabla_\theta \mathcal{L} \odot \nabla_\theta \mathcal{L}) \end{aligned}$$

where β_1, β_2 are decay rates. In the first timestep, the moments are initialised to zero. To correct the bias towards zero, the moments are corrected with

$$\begin{aligned} \hat{\boldsymbol{m}}_t &= \frac{\boldsymbol{m}_t}{1 - \beta_1^t} \\ \hat{\boldsymbol{v}}_t &= \frac{\boldsymbol{v}_t}{1 - \beta_2^t} \end{aligned}$$

Then, the parameter can be updated with

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha \cdot \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon}$$

where are the α is the learning rate and ϵ is a small constant to avoid division by zero.

2.1.2 Convolutional Neural Network

CNN is a type of neural network that is efficient in image classification tasks. An image can be constructed as a three-dimensional matrix where the height and width correspond to the size of the image, and the depth corresponds to the number of colour channels. A CNN is to take this input [height × width × depth] to some output $[1, 1, C]$ where C is the number of classes.

A typical CNN has three types of layers, which are

1. Convolutional layer
2. Pooling layer
3. Fully-connected layer

2.1.2.1 Convolutional Layer

A convolutional layer is where most of the computation in a CNN occurs. Convolution is an operation in which a small matrix, known as a filter or kernel, slides across the input image and computes the dot product at each position [11]. The filter can be seen as a feature detector or a pattern recogniser, each specialised in identifying specific visual patterns, such as edges, textures or other patterns. During training, the network learns to recognise these patterns, represented as the filters' weights.

The filter moves across the matrix with some value called stride. Another technique used is padding, which refers to adding extra pixels (usually zeros) around the border of the matrix before the convolution. This allows filters to look at the border regions equally and also helps control the output size, which is called an activation map (also a feature map). An activation function can then be applied to the activation map before sending it to the pooling layer. Figure 2.2 shows a visualisation of this operation.

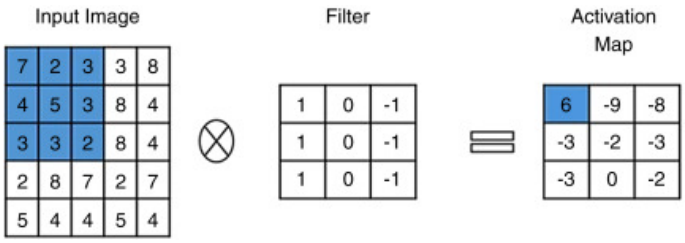


Figure 2.2: An example visual representation of the convolution operations. Input matrix is of size 5x5, filter is of size 3x3, stride equals one and no padding occurs. One operation is marked with blue background. The activation map is of size 3x3. Image taken from [12].

2.1.2.2 Pooling Layer

The pooling layer usually comes after the convolutional layer in a CNN. The purpose of this layer is to reduce the dimension (height and width) of the activation map while preserving most of the information. The pooling layer slides a small matrix, similar to the convolutional layer filter, across the layer's input matrix. The most common pooling operation is max pooling, which takes the maximum value inside each window. Another common pooling operation is average pooling, which averages the values inside the window. Figure 2.3 shows an example of how the pooling layer works.

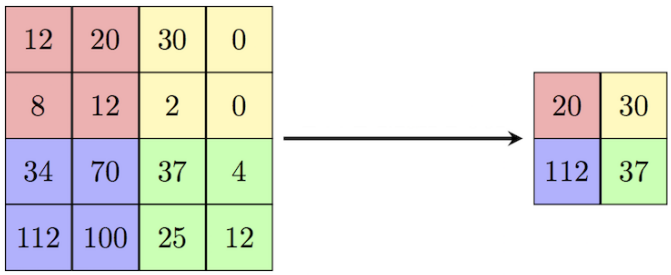


Figure 2.3: An example visual representation of the pooling layer. A max pooling filter of size 2x2 with a stride of 2 is slid across the matrix. The size of the input matrix is 4x4, and the output becomes 2x2.

2.1.2.3 Fully Connected Layer

A fully connected layer is the same as one layer in an artificial neural network, which is described in section 2.1

2.1.3 Residual Neural Network

Residual Neural Network (ResNet) is a CNN that uses skip connections. Skip connections allow the input to skip over layers and be added directly to a deeper layer, which helps the gradients to flow better since, in deep neural networks, the gradient can become tiny, a problem called the vanishing gradient problem. If a skip con-

nection is used, a residual block is created. A visual representation can be seen in Figure 2.4.

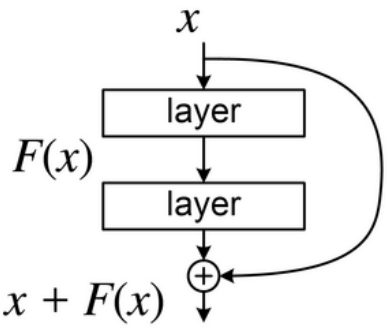


Figure 2.4: A residual block with two skipped layers. The input passes through the two layers but also skips them to be added together in a deeper layer.

2.1.3.1 ResNet-18

ResNet-18 is a residual neural network architecture consisting of 18 layers. After each layer, batch normalization is applied, followed by a ReLU activation function [13]. The architecture can be seen in Table 2.1 and Figure 2.5.

In this thesis, all the layers, excluding the last layer, will be referred to as the feature extractor. The neuron activations of an input after the feature extractor will be referred to as the input's feature space embedding.

Table 2.1: ResNet-18 architecture. Brackets correspond to residual blocks. Downsampling is done with a stride of 2. After each layer, batch normalisation is applied, followed by a ReLU activation function. Pooling does not count as a layer.

Layer	Operation	Output Size
Image		224 × 224 × 3
Conv1	7×7, 64 filters	112 × 112 × 64
Pooling	7×7, avg pooling	56 × 56 × 64
Conv2	$\begin{bmatrix} 3\times3, 64 \text{ filters} \\ 3\times3, 64 \text{ filters} \end{bmatrix} \times 2$	56 × 56 × 64
Conv3	$\begin{bmatrix} 3\times3, 128 \text{ filters} \\ 3\times3, 128 \text{ filters} \end{bmatrix} \times 2$	28 × 28 × 128
Conv4	$\begin{bmatrix} 3\times3, 256 \text{ filters} \\ 3\times3, 256 \text{ filters} \end{bmatrix} \times 2$	14 × 14 × 256
Conv5	$\begin{bmatrix} 3\times3, 512 \text{ filters} \\ 3\times3, 512 \text{ filters} \end{bmatrix} \times 2$	7 × 7 × 512
Pooling	7×7, avg pooling	1 × 1 × 512
FC	Fully connected	1 × 1 × #classes

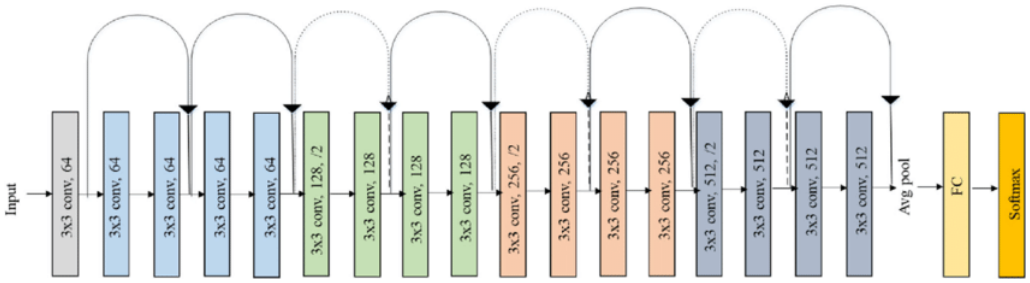


Figure 2.5: Visualisation of the ResNet-18 architecture. Downsampling is done with a stride of 2. After each layer, batch normalisation is applied, followed by a ReLU activation function. The softmax activation function at the end of a model does not count as a layer. Image taken from [14].

2.1.4 Transfer Learning

Transfer learning is a technique where knowledge from one task is transferred to another. In machine learning, this is often translated to transferring the weights from an already trained model with a specific task to another model with a different task [15]. This leverages the general patterns the source model has already learned and adapts them to a new problem.

A common way of transfer learning in neural networks is to transfer over all the layers of a pre-trained network and replace the last, fully connected layer with a new layer. Then, the weights of all layers except the last, new one are locked, followed by training the network on the new task.

After training the new layer, fine-tuning can be done by unfreezing some layers and training them with the final layer on a lower learning rate. In this work, fine-tuning is applied to all layers of the model.

2.1.5 Dropout

Dropout is a method used to avoid overfitting, a common problem during neural network training. The core idea is randomly setting a fraction, called the dropout rate, of the neurons in a layer to zero during a forward pass. Dropout prevents the neurons from co-adapting too much [16].

$$x_i^{[L]} = \begin{cases} 0, \text{ with dropout probability } P_d \\ x_i^{[L]}, \text{ otherwise} \end{cases}$$

2.1.6 Network Metrics

Different metrics can be used to evaluate the performance of a neural network. In classification tasks, some standard terms that describe the metrics are the following. For each class, the model's predictions can be evaluated as

- True Positive (TP): Instances that belong to the class and are correctly predicted as that class
- True Negative (TN): Instances that do not belong to the class and are correctly predicted as not belonging to that class
- False Positive (FP): Instances that do not belong to the class but are incorrectly predicted as belonging to that class.
- False Negative (FN): Instances that do belong to the class but are incorrectly predicted as not belonging to that class.

Accuracy is the most common metric used to evaluate a neural network and is defined as the percentage of correct predictions made by the neural network. It can be expressed as

$$P_{acc} = \frac{\text{Total Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

A confusion matrix can be used to visualise the performance of a multi-class neural network. The matrix plots the predicted labels against the true labels. Figure 2.6 shows an example of a confusion matrix.

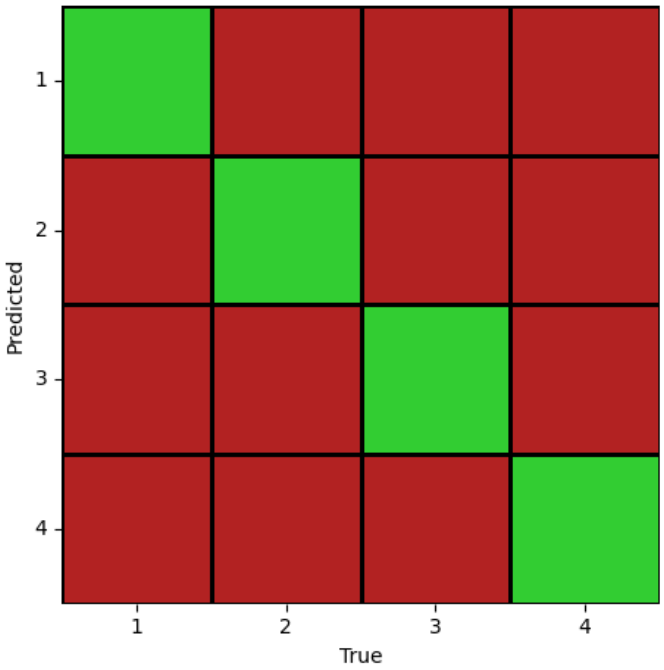


Figure 2.6: Visualisation of an example confusion matrix with four classes. The x-axis represents the true label, and the y-axis represents the predicted label. Green squares are correct predictions by the network, while red squares are incorrect predictions.

2.2 Out-Of-Distribution Metrics

OOD detection can be framed as a binary classification problem, where a positive prediction indicates an ID input and a negative prediction indicates an OOD input. The two most common metrics for evaluating OOD detectors stem from the receiver operating characteristics (ROC) curve. The ROC curve is created by plotting the False Positive Rate (FPR) against the True Positive Rate (TPR). Recall the metric from section 2.1.6. Let positive correspond with ID and negative with OOD.

TPR is then defined as

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

and FPR is defined as

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}.$$

A perfect classifier will have a square ROC curve where the TPR is always 100% for all FPRs, while a completely random classifier will have a diagonal ROC curve where the TPR equals the FPR for all values. Figure 2.7 (a) illustrates the ROC curve.

The FPR95 metric quantifies the percentage of OOD inputs incorrectly classified as ID when the TPR is maintained at 95%. A lower FPR95 corresponds with a better-performing classifier, and a perfect classifier would have an FPR95 of 0%. The purple dashed lines in Figure 2.7 (b) show how the FPR95 can be read from the ROC curve.

The AUROC metric can be interpreted as the average TPR across all possible FPRs. A perfect classifier will have an AUROC of 100%, and a random classifier will have an AUROC of 50%.

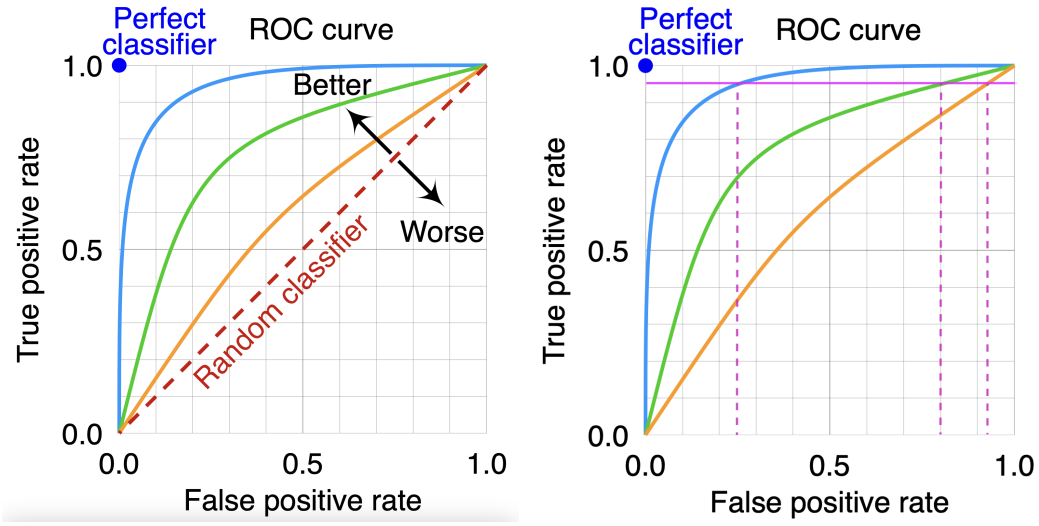


Figure 2.7: (a) Illustration of the receiver operation characteristics curve. (b) Illustration how the false positive rate at 95% true positive rate is read from the receiver operation characteristics curve. Image taken from [17].

2.3 Out-Of-Distribution Detectors

All OOD detectors proposed in this thesis use a thresholding function on their corresponding OOD score to determine if a sample is ID or OOD. The thresholding function is either

$$G_\tau(S) = \begin{cases} \text{ID}, & \text{if } S \geq \tau \\ \text{OOD}, & \text{if } S < \tau \end{cases} \tag{2.3}$$

or

$$G_\tau(S) = \begin{cases} \text{ID}, & \text{if } S \leq \tau \\ \text{OOD}, & \text{if } S > \tau \end{cases} \tag{2.4}$$

depending on which detector was used to calculate the OOD score, S is the OOD score and τ is the threshold.

2.3.1 Maximum Softmax Probability

Maximum Softmax Probability (MSP) is a simple but effective way of detecting OOD data. It operates by utilising the maximum confidence score produced by the softmax activation function (2.1). Hence, it can be written as

$$M = \max_i \text{softmax}(l_i). \tag{2.5}$$

MSP can be used for OOD detection using the thresholding function (2.3) with $S = M$.

2.3.2 Energy Based Out-Of-Distribution Detection

Energy based OOD detection was proposed by Liu et al. in 2020 [18] and introduces an Energy score. It can be derived by combining the softmax activation function with Gibbs distribution and is defined as

$$E(\boldsymbol{l}) = -T \cdot \log \sum_{i=1}^{C} e^{l_i/T} \tag{2.6}$$

where $\boldsymbol{l}$ is the logit vector and T is a constant temperature parameter. The Energy score can be used for OOD detection using the thresholding function (2.3) with $S = E(\boldsymbol{l})$.

2.3.3 Outlier Exposure

Outlier Exposure (OE) was proposed by Hendricks et al. in 2019 [19]. The core idea is explicitly training the model with auxiliary outliers, samples from an external dataset disjoint from the training data.

Consider an ID training dataset $D_{in} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ where $\boldsymbol{x}_i$ is the input vector, $\boldsymbol{y}_i$ is the onehot true label vector and N is the number of instances in the training dataset. Denote the softmax probability vector given an input $\boldsymbol{x}$ as $\hat{\boldsymbol{y}}$ and the logit output from the neural network as $\boldsymbol{l} = f(\boldsymbol{x})$. Also consider a outlier dataset $D_{out} = \{(\boldsymbol{x}_j)\}_{j=1}^{M}$. Then, we want to minimize the function

$$\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_{\text{in}}}\Big[\mathcal{L}\Big(f(\boldsymbol{x}), \boldsymbol{y}\Big) \; + \; \lambda\, \mathbb{E}_{\boldsymbol{x}'\sim\mathcal{D}_{\text{out}}^{\text{OE}}}\Big[\mathcal{L}_{\text{OE}}\Big(f(\boldsymbol{x}'), f(\boldsymbol{x}), \boldsymbol{y}\Big)\Big]\Big] \tag{2.7}$$

over the parameters of f where λ is a constant parameter. In the case of unlabelled data, $\boldsymbol{y}$ can be ignored.

The design of $\mathcal{L}_{\text{OE}}$ is a modelling choice and depends on the OOD detector being used. In this work, we consider only the combination of OE and MSP.

2.3.4 Outlier Exposure using Maximum Softmax Probability

For convenience, write equation (2.7) as

$$\mathcal{L} = \mathcal{L}_{in} + \lambda \mathcal{L}_{out}$$

Recall the cross-entropy loss for one sample equation (2.2). Then, for a batch consisting of N instances, the cross-entropy loss is averaged and can be written as

$$\mathcal{L}_{in} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}(y_i)_j \log (\hat{y}_i)_j \tag{2.8}$$

Since we want to regularise the MSP for the outliers, we want the target vector to be uniform $\mathbf{u} = (\frac{1}{C}, ..., \frac{1}{C})$ where C is the number of classes. Hence, the loss function for the outliers is defined as

$$\mathcal{L}_{out} = -\frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{C} u_j \log (\hat{y}_i)_j \tag{2.9}$$

By inserting equations (2.8) and (2.9) into (2.7), the loss function used for model training is obtained. The parameter λ is chosen to balance the ID and OOD terms. The MSP OOD detection described in section 2.3.1 determines if a given input is ID or OOD.

2.3.5 Virtual Outlier Synthesis

Virtual Outlier Synthesis (VOS) was proposed by Du et al. in 2022 [20] and works by synthesising outliers. Synthesising data in the high-dimensional pixel space can be challenging to optimise. Thus, VOS synthesises data in the lower-dimensional feature space.

VOS assumes that the feature space embeddings of the ID training dataset form a class-conditional multivariate Gaussian distribution

$$p_\theta(\phi(\boldsymbol{x})|y = k) = \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}),$$

where $\boldsymbol{\mu}_k$ is each class mean, $\boldsymbol{\Sigma}$ is a covariance matrix and $\phi(\boldsymbol{x})$ is the inputs feature space embeddings. The mean values of the class-conditional Gaussian are given by

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{N_k}\sum_{i:y_i=k}\phi(x_i),$$

where N_k is the number of images belonging to class k in the training set, $\phi(\boldsymbol{x}_i)$ is the i-th input image's feature space embedding. The covariance matrix is given by

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N}\sum_{k}\sum_{i:y_i=k}(\phi(x_i) - \hat{\boldsymbol{\mu}}_k)(\phi(x_i) - \hat{\boldsymbol{\mu}}_k)^\intercal,$$

where N is the total number of images in the training set.

VOS now samples virtual outliers from the ϵ-likelihood region of the estimated class-conditional distribution.

$$\boldsymbol{\nu}_k = \{\boldsymbol{v}_k | \frac{1}{(2\pi)^{m/2}|\hat{\boldsymbol{\Sigma}}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{v}_k - \hat{\boldsymbol{\mu}}_k)^T \hat{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{v}_k - \hat{\boldsymbol{\mu}}_k)\right) < \epsilon\}$$

where $\boldsymbol{\nu}_k \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\Sigma}})$ is the sampled virtual outliers for class k.

For the use in training, VOS introduces a new loss,

$$\mathcal{L}_{uncertainty} = \mathbb{E}_{\boldsymbol{v}\sim\boldsymbol{\nu}}\left[-\log\frac{1}{1+e^{-\eta(E(\boldsymbol{v}))}}\right] + \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[-\log\frac{e^{-\eta(E(\boldsymbol{x}))}}{1+e^{-\eta(E(\boldsymbol{x}))}}\right] \tag{2.10}$$

where $\eta(\cdot)$ is a linear regression model which allows learning of a flexible energy surface. The energy $E(\boldsymbol{x}^*)$ is the Energy score defined in section 2.3.2.

The overall training objective for the network now becomes

$$\min_{\theta} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\ \mathcal{L}_{cls} + \lambda\mathcal{L}_{uncertainty} \tag{2.11}$$

where $\mathcal{L}_{cls}$ is the cross-entropy loss and λ is the weight of the uncertainty regularization.

During testing, VOS produces an OOD uncertainty score for a given test input $\boldsymbol{x}^*$ through

$$p_\theta(g|\boldsymbol{x}^*) = \frac{e^{-\eta(E(\boldsymbol{x}^*))}}{1+e^{-\eta(E(\boldsymbol{x}^*))}}. \tag{2.12}$$

For OOD detection VOS uses the thresholding function (2.4) with $S = p_\theta(g|\boldsymbol{x}^*)$ to determine if any given input is ID or OOD.

2.3.6 Rectified Activations

Rectified Activations (ReAct) was proposed by Sun et al. in 2021 and works by rectifying unit activations in the feature space of the model [21]. A rectification parameter $c > 0$ is chosen to sufficiently preserve the activations for ID data while rectifying that of OOD data. In practice, c is set to the q-th percentile value of feature space activations in the ID data.

The ReAct operation can be described as

$$\bar{\phi}(\boldsymbol{x}) = ReAct(\phi(\boldsymbol{x}); c) = min(\phi(\boldsymbol{x}); c) \tag{2.13}$$

and is applied element-wise to the output of the feature extractor $\phi(\boldsymbol{x})$.

The logits for the rectified network are then given by

$$\bar{\boldsymbol{l}} = \boldsymbol{W}^\intercal\bar{\phi}(\boldsymbol{x}) + \boldsymbol{b} \tag{2.14}$$

where the final classification and OOD detection is given by the MSP described in section 2.3.1.

2.3.7 Deep Nearest Neighbours

Deep Nearest Neighbours (d-kNN) was proposed by Bergman et al. in 2022 [22]. d-kNN works by calculating the Euclidean distance between the given input and

all training data in feature space. Thresholding is performed on the k-th shortest distance to determine if the input is ID or OOD.

d-kNN passes each image in the training dataset $\boldsymbol{x}_i$, $i \in [1, N]$ where N is the number of images in the dataset, through the feature extractor $\phi(\cdot)$ to get each sample's feature space embedding. All embeddings are then L2 normalised and stored as

$$\boldsymbol{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, ...\boldsymbol{z}_N] = [\frac{\phi(\boldsymbol{x}_1)}{\|\phi(\boldsymbol{x}_1)\|_2}, ..., \frac{\phi(\boldsymbol{x}_N)}{\|\phi(\boldsymbol{x}_N)\|_2}].$$

For OOD detection, d-kNN passes the sample $\boldsymbol{x}^*$ through the feature extractor ϕ and normalises the feature space embedding to obtain $\boldsymbol{z}^*$. The Euclidean distances between the sample $\boldsymbol{z}^*$ and each sample in $\boldsymbol{Z}$ are calculated to form a new vector $\boldsymbol{d}$ where each element represents the distance from $\boldsymbol{z}^*$ to $\boldsymbol{z}_i$.

$$\boldsymbol{d} = [d_1, d_2, ..., d_n].$$

The vector, $\boldsymbol{d}$ is then reordered in increasing distance, which forms the vector $\boldsymbol{d}^{'}$. The distance to the k-th index in $\boldsymbol{d}^{'}$ is considered as the OOD score for $\boldsymbol{x}^*$. d-kNN performs thresholding using equation (2.4) with $S = d'_k$.

2.3.8 Virtual Logit Matching

Virtual logit matching (ViM) was first proposed by Wang et al. in 2022 [23]. ViM creates a virtual logit in the feature space which is the compared against the original logits to calculate an OOD score.

ViM extracts the features space embeddings from all images in the training set and constructs a matrix $\mathbf{X} \in \mathbb{R}^{N \times M}$ where N is the number of images in the training set and M is the dimension of the feature space. The layer that transform the input from the feature space to the logits contain a bias term. To avoid this bias term, the feature space is moved to a new coordinate system. A point,

$$\boldsymbol{o} := -(\boldsymbol{W}^T)^+ \boldsymbol{b}$$

where $(\cdot)^+$ is the Moore-Penrose inverse, is set as the origin of a new coordinate system. Then, the logits are given by

$$\boldsymbol{l} = \boldsymbol{W}^T(\boldsymbol{x} - \boldsymbol{o}) = \boldsymbol{W}^T\boldsymbol{x}'. \tag{2.15}$$

A covariance matrix between all images in the feature space is calculated as

$$\boldsymbol{X}^\intercal \boldsymbol{X} = \Sigma$$

on which eigendecomposition is performed as

$$\Sigma = \boldsymbol{Q}\Lambda\boldsymbol{Q}^{-1}. \tag{2.16}$$

The eigenvalues, Λ, are sorted decreasingly and the corresponding eigenvector matrix, $\boldsymbol{Q}$, is arranged accordingly. The principal space $\boldsymbol{P} \in \mathbb{R}^{M \times D}$ is the span of the D first columns. Let the $(D+1)$-th column to the last, M-th, column in $\boldsymbol{Q}$ be a new matrix $\boldsymbol{R} \in \mathbb{R}^{M \times (M-D)}$.

The residual, $\boldsymbol{x}^{\boldsymbol{P}^\perp}$, is the projection of $\boldsymbol{x}$ onto the null-space $\boldsymbol{P}^\perp$ given by

$$\boldsymbol{x}^{\boldsymbol{P}^\perp} = \boldsymbol{R}\boldsymbol{R}^T\boldsymbol{x}. \tag{2.17}$$

The norm of the residual, $||\boldsymbol{x}^{\boldsymbol{P}^\perp}||$, can not be used directly as a new logit since the softmax operation normalises over the exponential of the logits and is, therefore, highly sensitive to the scale of the logits. For the virtual logit to match the scale of the logits, the average norm of the virtual logit is divided by the mean of the maximum logit from the training dataset as

$$\alpha := \frac{\Sigma_{i=1}^{K} max_{j=1,...,C}\{l_i^j\}}{\Sigma_{i=1}^{K} ||\boldsymbol{x}_i^{\boldsymbol{P}^\perp}||} \tag{2.18}$$

where $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_K$ is K uniformly sampled training images, l_i^j is the j-th logit of $\boldsymbol{x}_i$, and C is the number of classes. The norm of the residual is scaled with α resulting in the virtual logit:

$$l_0 = \alpha ||\boldsymbol{x}_i^{\boldsymbol{P}^\perp}||. \tag{2.19}$$

An OOD score can be calculated using the softmax probability for the virtual logit l_0 as

$$\text{ViM}(x) = \frac{e^{l_0}}{\sum_{i=0}^{C} e^{l_i}}. \tag{2.20}$$

The thresholding function (2.3) is used to determine if the input is either ID or OOD by replacing S with the ViM(x).

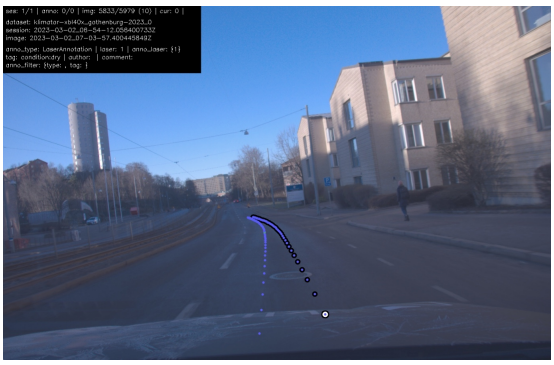
3

Method

This chapter introduces the thesis's overall workflow. It explains how the data was gathered and assembled into datasets, all parameters and the training pipeline of the models, and a detailed view of the implementation of each OOD detector.

3.1 Datasets

Klimator holds a database containing images with laser annotations. The data was gathered with a camera inside the car, together with one or more lasers under the car. The laser takes measurements of the road surface and, through an algorithm, returns a probability of the surface condition under the car [24]. These laser annotations are then superimposed onto the road during the post-processing of the data. Figure 3.1a shows an example of the graphical user interface.



(a) Camera view of the RSC with laser annotations superimposed.

(b) Road condition predictions illustrating both the left (upper) and right (lower) lasers.

Figure 3.1: Comparative displays of camera and laser-based road assessments: (Left) Camera view with superimposed laser annotations, and (Right) Road condition predictions illustrating both the left (upper) and right (lower) lasers.

3.1.1 In-Distribution Dataset

A three-class dataset was used as the ID dataset. An algorithm was created to check each image in the database to assemble this dataset. If an image contained

only dry, at least 90% wet or at least 80% snowy annotations, it was added to the ID dataset labelled with its corresponding class. Since the data is collected from a vehicle driving on the road, capturing ten images per second, consecutive images look almost the same. Therefore, if an image was saved to the ID dataset, all captured images over the next 40 meters of driving were skipped. The final ID dataset contained 36223 images with the class distributions: 21706 dry, 18437 wet and 5128 snow.

The full dataset was then split into training, validation, and testing datasets, with an 80%/10%/10% split. The training set contained 35876 images, and validation and testing contained 5434 images, with an equal distribution of the classes in each subset. Since images taken 40 meters apart still appeared almost identical, an algorithm was developed to separate the data. This algorithm ensures that images from the same hour can not appear in more than one subset.

(a) Class: Dry **(b)** Class: Dry **(c)** Class: Dry

(d) Class: Wet **(e)** Class: Wet **(f)** Class: Wet

(g) Class: Snow **(h)** Class: Snow **(i)** Class: Snow

Figure 3.2: Example images from the ID dataset

3.1.2 Out-Of-Distribution Datasets

The OOD datasets used in this thesis can be divided into two categories of data: near OOD and far OOD. The far OOD datasets were the publicly available datasets

CIFAR-10 [25] and Textures [26]. For near OOD data, images close to the ID dataset in pixel space were needed. Therefore, images from the database were used under the condition that they did not overlap with any of the classes in the ID dataset. Three variants of such data were gathered into OOD datasets, referred to as the Cars, Glare, and Slush datasets. The Slush dataset was assembled using the same method as in section 3.1.1 with the condition that at least 40% of the laser annotations were slush, and no more than 10% of the annotations were either dry, wet or snow. The Cars and Glare datasets were manually assembled from the database. Figure 3.3, 3.4, 3.5, 3.6, and 3.7 each show three examples of images sampled from the Car, Glare, Slush, CIFAR-10, and Textures datasets, respectively.

Figure 3.3: Example images from the Cars dataset

Figure 3.4: Example images from the Glare dataset

Figure 3.5: Example images from the Slush dataset

Figure 3.6: Example images from the CIFAR-10 dataset [25].

Figure 3.7: Example images from the Textures dataset [26].

3.2 Neural Network and Feature Extractor

The base neural network was created using transfer learning. A pre-trained ResNet-18 model with its fully connected layer removed was used as the feature extractor, transforming the images from pixel space into a 512-dimensional feature space. A new fully connected layer was constructed, which transforms the image feature space embeddings into three logits, one for each class. The maximum logit value was used to classify a given input.

The parameters in the feature extractor were frozen, and the final layer was trained for 20 epochs using the training dataset with a batch size of 128. The Adam algorithm was used for backpropagation on the cross-entropy loss with a learning rate of 0,01. A scheduler was used to avoid early convergence to a suboptimal solution, which would lower the learning rate by a factor of ten if there was no improvement for 50 batches. The evaluation dataset was used to evaluate the training every fifth epoch to avoid overfitting. This network will be referred to as the Base network.

A second network was created by fine-tuning the Base network for another 20 epochs with a starting learning rate of 0,01. This network will be referred to as the Fine-tuned network.

3.3 Metrics

The ID test dataset was passed through the network and OOD detector to calculate the ROC curve presented in section 2.2. Each image in the ID test dataset was then given an OOD score, which was saved in a scores list for later use. All OOD datasets were passed through the network and detector, and each input was given an OOD score. These scores were saved into five different score lists, one for each dataset.

The ROC curve was then created by setting the threshold τ in the thresholding operation to the first percentile highest or lowest value in the ID scores list, meaning that 1% of the ID data was accepted as ID, which is equivalent to a TPR of 1%. The OOD detector's thresholding operation was then used on all five OOD score lists to give an FPR for 1% TPR for each OOD dataset. τ is then set to the second percentile value of the ID scores, and a corresponding FPR is calculated for 2% TPR. This operation was repeated until the TPR reached 100%, at which point a final data point was manually created for both 100% TPR and FPR. All the created data points were then used to plot the ROC curve.

FPR95 was given by the FPR value corresponding to the 95-th percentile value of the ID scores list.

The AUROC was then given by calculating the area under the ROC curve by integrating along the FPR axis using the composite trapezoidal rule.

3.4 Out-Of-Distribution Detectors

The following section describes in detail the implementation of the detectors and the parameters used. The detectors can be categorised into two categories: trainers and post-processors. All detectors are implemented at least twice, once using the Base network and once using the Fine-tuned network. All parameters for the detectors were chosen through multiple runs of testing.

3.4.1 Maximum Softmax Probability

MSP is a parameter-free detector categorised as a post-processor and is implemented with only an ID/OOD threshold. This thesis considers MSP a baseline detector since it is the most widely used OOD detector. The test dataset was fed forward through the network, and the confidences for each image were then calculated using equations (2.1) and (2.5). The threshold, τ, in the thresholding equation (2.3) was set to the value corresponding to the p-th percentile lowest ID test dataset confidence.

To evaluate the detector, all OOD datasets were passed through the network, and their images corresponding MSP confidences were calculated. Each image was then flagged as either ID or OOD using the thresholding equation (2.3).

3.4.2 Energy Based Out-Of-Distribution Detection

Energy based OOD detection is categorised as a post-processor and is, in this thesis, considered as a baseline detector since it is widely used. For convenience, the negative Energy score, $-E(\boldsymbol{l})$, was calculated for each ID test dataset input using equation (2.6) with $T = 1$. The thresholding equation (2.3) is used where τ was set to the p-th percentile lowest test dataset Energy score.

To evaluate the detector, all OOD datasets were passed through the network, and their images corresponding Energy scores were calculated. Each image was then flagged as either ID or OOD using the thresholding equation (2.3).

3.4.3 Outlier Exposure

OE is categorised as a trainer since it operates mainly during training. It relies on MSP to calculate ID confidences for each input during classification.

Since OE depends on a large OOD dataset during training, it was implemented twice per network using both the Textures and CIFAR-10 datasets. For all implementations, the network's fully connected layer was further trained for 20 epochs using the same algorithm for backpropagation and learning rate as the Base network. During training, one ID batch and one OOD batch, both of size 128, were forward fed through the network and the loss was then calculated with equations (2.7), (2.8) and (2.9). λ in equation (2.7) was set to 0,5.

For evaluation of OE, MSP is used in the same way as described in section (3.4.1) with the sole exception that a new τ in equation (2.3) had to be set. The reason for a new τ was that the network parameters were updated during training, resulting in a new distribution of logit values that differed from the Base and Fine-tuned networks' logit distributions.

3.4.4 Virtual Outlier Synthesis

OE was categorised as a trainer since it operates mainly during training. It relies on Energy based OOD detection to calculate OOD scores for each input during classification.

Six outliers were synthesised in feature space for each batch of 128 ID images, with two belonging to each class. To synthesise these outliers, three class-dependent queues were created, consisting of 1000 feature space embeddings each. Whilst filling up the queues, the network was further trained as in section 3.2 whilst enqueueing an input feature space embedding to its corresponding class queue. If one or more queues were full when a new embedding for the class appeared, the oldest embedding was dequeued, and the new embedding was enqueued.

When all three queues were full, outliers began to be synthesised. For each batch, the enqueueing/dequeueing process continues before using the queues to construct three class-dependent multivariate Gaussian distributions. 10000 embeddings were sampled from each Gaussian, and each sample's log probability was calculated. These

probabilities were sorted in ascending order, and the two samples with the lowest log probability were selected as that class's outliers. After each class has had two outliers synthesised, the full batch plus the six outliers are passed through the fully connected layer to calculate each embedding's logits. The Energy score for each embedding was then calculated using equation (2.6), and a target vector was created where the ID embeddings got target 1, and outliers got target 0. The Energy score for each embedding was then passed through a linear regression, and the cross-entropy loss of the linear regression was calculated using the target vector. This loss corresponds with $\mathcal{L}_{uncertainty}$ in equation (2.10). The normal cross-entropy loss, $\mathcal{L}_{cls}$, was then calculated without the outliers before using equation (2.11), in which λ was set to 0,1, to calculate the final loss.

The full ID test dataset was then passed through the network, and OOD scores, $p_\theta(g|\boldsymbol{x}^*)$, were calculated for each image using equation (2.12). τ in equation (2.3) was set to the p-th percentile lowest OOD score for the ID test dataset.

To evaluate the detector, all OOD datasets were passed through the network, and their images corresponding Energy scores were calculated. Each image was then flagged as either ID or OOD using the thresholding equation (2.3).

3.4.5 Rectified Activations

ReAct is categorised as a post-processor and relies on Energy based OOD detection to calculate OOD scores for each input during classification.

The full ID train dataset was passed through the network, and the feature space embeddings of each image were saved to a list. c in equation (2.13) was then set to the q-th percentile highest value in this list. q was set to 95% for the Base network and 99% for the Fine-tuned network.

The fully connected layer for the network was then replaced by the operation described in equation (2.14) where $\boldsymbol{W}^\intercal$ and $\boldsymbol{b}$ are the weights and biases of the network's fully connected layer, respectively.

To evaluate the detector, all OOD datasets were passed through the network, and their images corresponding Energy scores were calculated. Each image was then flagged as either ID or OOD using the thresholding equation (2.3).

3.4.6 Deep Nearest Neighbour

D-kNN is categorised as a post-processor and operates entirely in feature space.

The ID train dataset was passed through the network, and all feature space embeddings were L2 normalised before being saved to an index.

Each image in the ID test dataset was then passed through the network, and the k-th shortest distance, with $k = 1$, from the image feature space embedding the data points in the index is the OOD score. The threshold, τ in equation (2.4), was set to the p-th percentile highest distance from the ID test dataset.

To evaluate d-kNN, all OOD sets were passed through the network, and each image's

distance to the k-th nearest data points in the index was calculated. Each image was then flagged as either ID or OOD using the thresholding equation (2.4).

3.4.7 Virtual-logit Matching

ViM is categorised as a post-processor and produces a virtual logit, which is compared against the original logits.

A subset of the training dataset was passed through the network to construct the covariance matrix, $\boldsymbol{X}$, in equation (2.16). The subsets' feature space embeddings were saved to a matrix. The logits were calculated using equation (2.15) and then saved to another matrix to omit the bias term in the fully connected layer of the network. The size of the subset was the first 1000 occurrences of each class in the ID train dataset, which was shuffled. The embeddings in feature space were moved to the new coordinate system with an origin in $\boldsymbol{o}$ before the empirical covariance matrix, $\boldsymbol{X}$, was constructed. The null-space, $\boldsymbol{P}^{\perp}$, in equation (2.17) was calculated by sorting the eigenvalues of $\boldsymbol{X}$ in ascending order and removing the first 60 dimensions for both the Base- and the Fine-tuned network. To obtain the unscaled virtual logit $||\boldsymbol{x}_i^{\boldsymbol{P}^{\perp}}||$ in equation (2.18), each image embedding in feature space was projected onto the null-space by matrix multiplication before being normalised. The scaling parameter, α, was calculated using equation (2.18), where the numerator was given by the logits saved during the forward pass of the subset.

With the null-space and alpha created, the ID test dataset was passed through the network. The feature space embedding for each image was used to calculate its corresponding virtual logit using equation (2.19) and OOD score using equation (2.20). A threshold, τ, was then set to the p-th percentile highest score from the ID test dataset.

The OOD datasets were passed through the feature extractor, and an OOD score for each image was produced. Then, the thresholding operation in equation (2.4) was applied to each image OOD score to flag it as either ID or OOD.

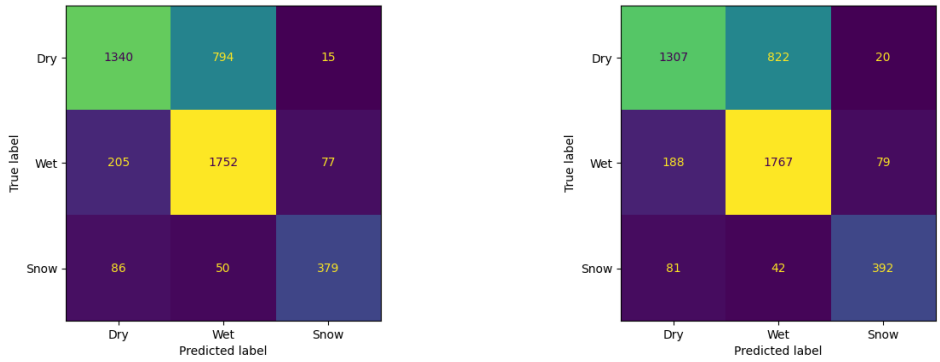
4

Results

This chapter presents the findings from the evaluation of both the neural networks and OOD detectors. First, the confusion matrices and accuracy of the networks are presented, followed by the AUROC and FPR95 for each OOD detector across the various OOD datasets.

4.1 Networks

Figure 4.1 shows the confusion matrices for both networks, Figure 4.1a shows the base model which has a accuracy of 73,88% on the ID test dataset and Figure 4.1b shows the Fine-tuned network which has a accuracy of 73,78% on the ID test dataset.



(a) Base network **(b)** Fine-tuned network

Figure 4.1: Confusion matrices for the Base (Left) and Fine-tuned network (Right).

4.2 Out-Of-Distribution Detectors using the Base Network

Table 4.1 shows the final FPR95 and AUROC scores for all OOD detectors using the Base network. ReAct alters the network's feature space, and OE and VOS further train the network, all of which affect its accuracy. Therefore, ReAct has an accuracy of 73,82%, OE Cifar has an accuracy of 80,21%, OE Textures has an accuracy of 74,89%, and VOS has an accuracy of 75,01% on the ID test dataset.

The ROC curves, density plots of the ID/OOD scores and confusion matrices for the OOD detectors that alter the network can be seen in appendix A.

Table 4.1: AUROC and FPR95 results for all OOD detectors using the Base network on the five OOD datasets. ↓ or ↑ indicate if a small or large value represents a better result. The best result is emphasised in bold, and the second-best result is underlined.
*The dataset was used during the training of the detector.

	Cars		Glare		Slush		Cifar 10		Textures	
OOD Detector	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑
MSP	97.62	57.83	90.87	58.38	**86.11**	**68.31**	83.49	67.69	83.51	68.98
Energy	92.06	60.43	92.78	61.72	86.81	63.76	94.17	59.34	89.79	63.57
OE Cifar	96.83	53.60	93.16	52.34	90.74	61.27	*87.17	*65.79	86.01	65.27
OE Textures	95.24	47.04	97.34	42.53	93.75	57.94	92.55	55.40	*85.21	*71.13
VOS	92.06	66.82	95.06	53.90	95.14	56.64	98.20	47.68	94.20	47.55
ReAct	90.48	60.99	**90.49**	**65.42**	86.34	66.88	83.92	82.52	80.05	78.88
ViM	**78.57**	**75.91**	93.16	58.10	93.52	58.61	0.84	99.77	21.86	93.90
d-kNN	93.65	56.43	93.92	48.24	95.60	50.78	**0.14**	**99.89**	**1.81**	**99.27**

4.3 Out-Of-Distribution Detectors using the Fine-tuned Network

Table 4.2 shows the final FPR95 and AUROC scores for all OOD detectors using the Fine-tuned network. ReAct alters the network's feature space, and OE and VOS further train the network, all of which affect its accuracy. Therefore, ReAct has an accuracy of 73,63%, OE Cifar has an accuracy of 79,29%, OE Textures has an accuracy of 74,24%, and VOS has an accuracy of 71,40% on the ID test dataset.

The ROC curves, density plots of the ID/OOD scores and confusion matrices for the OOD detectors that alter the network can be seen in appendix B

Table 4.2: AUROC and FPR95 results for all OOD detectors using the Fine-tuned network on the five OOD datasets. ↓ or ↑ indicate if a small or large value represents a better result. The best result is emphasised in bold, and the second-best result is underlined.
*The dataset was used during the training of the detector.

	Cars		Glare		Slush		Cifar 10		Textures	
OOD Detector	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑
MSP	96.03	59.09	**90.11**	59.82	86.81	**68.93**	84.06	67.04	83.30	69.09
Energy	90.48	61.59	**90.11**	63.72	**85.19**	64.04	93.40	58.50	89.89	62.96
OE Cifar	96.83	53.83	91.63	51.91	90.97	63.88	*83.76	*71.68	83.83	69.71
OE Textures	96.83	48.71	97.72	45.40	94.21	59.09	87.39	65.81	*82.23	*72.48
VOS	93.65	68.19	90.87	65.52	85.88	63.17	88.88	64.35	96.60	53.95
ReAct	90.48	62.16	**90.11**	**67.31**	85.42	67.07	83.70	81.99	80.21	78.39
ViM	**76.98**	**75.75**	94.30	59.33	93.75	58.23	0.91	99.74	21.91	93.77
d-kNN	92.86	56.88	93.16	49.24	96.06	50.61	**0.16**	**99.88**	**2.02**	**99.21**

5

Discussion

In this chapter, we discuss the results and decisions made throughout this thesis. We start by discussing the difference between the results of the Fine-tuned and Base model before analysing the results of each OOD detector on the near and far OOD datasets. Finally, we discuss the boundaries between ID, near OOD, and far OOD and how they influence our results.

5.1 Difference between the Base and Fine-tuned model

When comparing the confusion matrices of the Base networks and the Fine-tuned networks, as illustrated in Figures 4.1a and 4.1b, we observe that their accuracy rates are nearly identical. Additionally, if we examine each cell in Table 4.1 alongside its counterpart in Table 4.2, the maximum difference is only two percentage points. This leads us to conclude that, surprisingly, fine-tuning the model did not result in any significant improvement in the effectiveness of either the network or the detectors. Consequently, we will focus solely on the results presented in Table 4.1 for the remainder of our discussion regarding detector performance.

Typically, fine-tuning is expected to enhance the transformation from pixel space to feature space, resulting in a more well-defined feature space. Therefore, we anticipated improved performance from the detectors operating in feature space and higher overall accuracy for all fine-tuned networks. However, we observed the opposite, with similar or even slightly worse results for the detectors following fine-tuning.

This outcome may be explained by the training of the fully connected layer. During this phase, the optimiser quickly identifies the minimum loss for the initial learning rate, prompting the scheduler to reduce the learning rate. The optimiser continues working within that optimum until the learning rate is lowered again. The scheduler keeps decreasing the learning rate until training concludes after 20 epochs, which means the model is trained with a very low learning rate by the end.

In contrast, during fine-tuning, the model has many more parameters to adjust with each batch, which does not trigger the scheduler as often as it did during the training of the base network. As a result, the model tends to move away from the optimum established during the training of the fully connected layer due to the higher learning rate. It searches for a new optimum for all model parameters, and

the loss decreases for each batch. Therefore, the scheduler is not activated as often, and training concludes while still at a relatively high learning rate, leading to a worse outcome. To achieve improved results for the fine-tuned network compared to the base network, it would be beneficial to extend the fine-tuning process over more epochs. This approach ensures that the training concludes with a learning rate similar to that used when training the fully connected layer.

5.2 Effectiveness of the Detectors on the Near Out-Of-Distribution datasets

The near OOD datasets include three datasets to the left of the dashed line in Tables 4.1 and 4.2: Cars, Glare, and Slush. We anticipated that the Slush dataset would be closest to the border of the distribution, representing the limit between ID and OOD samples, while the Cars dataset would be most distant from the distribution.

When comparing all results of the detectors on the near OOD datasets, we find that they are about the same, except for ViM on the Cars datasets. The best AUROC score is 18 percentage units above 50% which is the same as guessing if a sample is ID or OOD. We can also see that the more advanced detectors do not significantly outperform the baseline detectors on any of the near OOD datasets. The second-best FPR95 on all near OOD datasets (after ViM on Cars) is the baseline detector MSP on the Slush dataset with an FPR95 of 86,11%. When using an OOD detector in a real scenario, the threshold is often drawn at 95% TPR of the training data. MSP's FPR95 of 86.11% would, therefore, in a real scenario, mean that the detector classifies 5% of all ID images as OOD whilst still classifying 86,11% of the OOD data as ID. Whether this trade-off is worth it depends on the scenario in which the detector is used.

When considering ViM's result on the Cars dataset, it is interesting that the FPR95 and AUROC greatly differ against the other detectors on the Cars dataset and ViM's score on the two other near OOD datasets. This likely depends on the fact that ViM calculates its score from a subset of the eigenspace of the covariance matrix created using training samples' feature space embeddings. The subset is created by discarding the 60 eigenvectors corresponding to the 60 largest eigenvalues, effectively removing the 60 dimensions that contribute the most to the final classification. If we compare this aspect of ViM with d-kNN, which also operates entirely in feature space, then d-kNN compares its sample with the training data on all dimensions in feature space. Since the 60 largest dimensions contribute the most to the d-kNN score, the smaller details in the feature space get overlooked, whilst ViM compares the sample with the smaller, less influential dimensions. Somewhere in these less influential dimensions, ViM recognises that the pattern of a car does not overlap with the patterns of a road surface. However, the patterns from Slush and Glare are not apparent enough for ViM to find those differences.

Our main takeaway from the detectors on the near OOD datasets is that these datasets closely match the ID dataset in pixel space. Since the near OOD and ID pixel spaces are so close, this translates further to the feature space and onto the

logits, making the border between ID and OOD hard to distinguish for the detectors.

5.3 Effectiveness of the Detectors on the Far Out-Of-Distribution datasets

The two far OOD datasets, shown to the right of the dashed line in Table 4.1 and 4.2, consist of the CIFAR-10 and Textures dataset, where we thought that the Textures dataset would be closer to the border of the distribution than the CIFAR-10 dataset.

The two baseline detectors perform the same on the far OOD datasets as on the near OOD datasets. Both OE models have a small bump in performance, whilst VOS has a worse result. One interesting result is the two OE models on the dataset they were trained on. We expected to see a larger difference in performance for OE Cifar on the CIFAR-10 dataset compared to it on the Textures dataset and vice versa for OE Textures; however, the most significant difference is less than 10 percentage units. The most likely reason all trainers underperform on all OOD datasets is that they were only trained for 20 epochs on a relatively small network. In the original paper where VOS was first presented, Du et al. trained VOS for at least 18,000 epochs on a ResNet-50 model [20]. In the original paper for OE, Hendrycks et al. train their OE model for 100 epochs using CIFAR-10 as their ID dataset and 1% of the *80 million tiny images* as their OOD dataset [19]. They, therefore, have trained for 80 epochs more than in this thesis, where each epoch also consisted of around 10 times more data. The main reason the trainers were only trained for 20 epochs each is that one epoch for VOS took around 50 minutes, and one epoch for OE took around 80 minutes. Therefore, training for more than 20 epochs for each trainer was not feasible. The difference between the epoch training time for VOS and OE stems from the fact that each epoch of OE contains twice the number of images as each epoch of VOS, since OE passes through an entire extra dataset.

The three best detectors on the far OOD datasets are the three detectors that operate somewhat or entirely in the feature space: ReAct, ViM and d-kNN. For ReAct, we can see that the AUROC score is high, but it is still struggling on the FPR95. Since ReAct uses Energy to calculate its OOD score, the difference between ReAct's and Energy's metrics indicates abnormally large activations in feature space. If the activations were of the same magnitude for both ID and OOD, then ReAct's and Energy's metrics would be the same. This is further emphasised in the metrics for ViM and d-kNN on the far OOD datasets, where both these detectors have good metrics on the Textures dataset and are almost perfect detectors on the CIFAR-10 dataset.

If we continue the discussion at the end of section 5.2 but consider the metrics for d-kNN and ViM on the CIFAR-10 dataset instead of on the Cars dataset, the conclusion that can be drawn on the CIFAR-10 dataset is the opposite of the conclusion drawn on the Cars dataset. In this case, the feature space embeddings for the OOD samples differ in all dimensions, where d-kNN picks up on the significant differences, whilst ViM looks at the more minor details. Both differ enough for the two detectors to separate ID and OOD nearly entirely.

In the original paper where ViM was first proposed, Wang et al. discuss what information is lost when a detector operates solely in the logits or in the feature space [23]. Their finding was that the information lost or gained between the feature space and the logits highly depends on the character of the OOD dataset. Depending on the OOD dataset used, detectors operating in the logits sometimes outperform those operating in feature space and vice versa. From these findings, they developed ViM to extract information from the feature space and then compare it to the logits to get the best information from both the feature space and the logits. Two OOD datasets used to emphasise this are the Textures dataset and the Openimage-O dataset, which overlap significantly with CIFAR-10 used in this thesis [27]. In this thesis, the d-kNN method operates solely in the feature space and is expected to outperform ViM on the Textures dataset. Interestingly, it also outperforms ViM by a small margin on the CIFAR-10 dataset. Noteworthy is that d-kNN was not used as a baseline detector in the ViM paper; instead, two similar feature-based detectors were used as the baseline.

All the advanced detectors show promising results on the far OOD datasets compared to the baseline detectors, and the trainers could possibly deliver good results if they had been trained for more epochs on more diverse data. As in the paper that proposed ViM, we also see that information is lost between the feature space and the logits. Implementing a detector in the feature space can be a lightweight, post hoc solution to finding significant distribution shifts or OOD samples in the data fed to the network.

5.4 Limit of the Distribution

When comparing the results of the detectors on the near- against the far OOD datasets, we can see a significant difference. As discussed in section 5.2, the near OOD closely match the ID dataset in pixel space, which is also reflected in the results of the detectors. Most detectors work as intended on the far OOD datasets, but are nearly useless on the near OOD datasets. Therefore, we ask ourselves: Can the near OOD datasets even be considered OOD? In a real-world scenario, OOD samples are, by nature, entirely unknown to the designer of a model, but to evaluate the model, OOD data is needed. In the case of this thesis, the near OOD datasets were constructed from samples gathered in the same sessions as the ID data, and the question asked is therefore further emphasised about the limit of the distribution. From the results, we believe that the near OOD dataset either overlaps with the ID dataset or is too close to the distribution limit for the detectors to notice any difference between the ID and the OOD samples.

6

Conclusion

In this thesis, we compared seven OOD detectors on three near OOD and two far OOD datasets to detect OOD samples or distribution shifts for an RSC classifier. Some detectors show promising results on the far OOD datasets, and there are suitable use cases for these in real-world applications. However, we do not believe that the results from the near OOD datasets convey enough information to decide on whether these detectors could be used for real-world RSC classification. From the nature of the problem, we cannot predict what OOD samples will occur in a real-world scenario, and the OOD data used for the evaluation of the detectors in this thesis covers such a narrow part of all OOD data that could occur in such scenarios. Therefore, further assessment of the detectors on more diverse OOD data is needed before dismissing or implementing any detectors for real-world RSC classification use. There are still many areas for improvement of the detectors and the evaluation, which is further discussed in Chapter 7.

7

Future work

7.1 Longer Training of Trainer Models

As discussed in section 5.3, all trainers were trained for fewer epochs than in their original research papers. To fully evaluate the trainers, they should be trained until convergence before conducting a new evaluation. The trainers' results still outperform most of the results obtained solely from the detector they used to produce an OOD score. Therefore, while the trainers are effective, they have not yet reached their full potential due to the limited training conducted in this thesis. More extended training periods would likely yield better results, allowing for a more accurate and fair evaluation of the trainers.

7.2 Evaluation on Diverse Out-Of-Distribution Datasets

As mentioned in section 5.4, the near OOD datasets may overlap with the ID dataset and only represent a limited portion of the distribution considered near OOD. Therefore, it is important to gather more OOD data to evaluate the detectors. Additionally, collecting a broader range of OOD data would help determine each detector's ID/OOD boundary and provide a more comprehensive assessment of their performance.

7.3 Real World Evaluation

Evaluating detectors in real-world scenarios is challenging because it's hard to foresee how OOD data might look. This makes precise assessment difficult using conventional methods. One potential solution is to deploy OOD detectors in real-world settings and monitor their performance during regular operation.

By analysing the images flagged as OOD, we can evaluate how effectively the detectors function in practice. For instance, if an image is labelled as OOD and is also misclassified, this indicates that the OOD detector is working as intended.

This approach offers a more realistic assessment of the detectors' performance. Gathering data from real-world use can help to evaluate each detector more fairly.

Bibliography

[1] V. Traveller, *Winter road-tripping*, https://www.flickr.com/photos/vincetraveller/51955487884, Licensed under CC BY 2.0, 2022.

[2] OpenAI, *Chatgpt*, https://chat.openai.com, 2024.

[3] Grammarly Inc., *Grammarly writing assistant*, https://www.grammarly.com, 2025.

[4] European Road Safety Observatory (ERSO), *Country overview 2023: Sweden*, Accessed: 2025-03-27, 2023. [Online]. Available: https://road-safety.transport.ec.europa.eu/document/download/cff9bbb1-ead6-4855-ad29-82c7755accda_en?filename=erso-country-overview-2023-sweden_0.pdf.

[5] Rapid Innovation, *Computer vision for advanced driver assistance systems (adas)*, https://www.rapidinnovation.io/post/computer-vision-for-advanced-driver-assistance-systems-adas, Accessed: 2025-05-26, 2024.

[6] K. Kanwal, K. T. Ahmad, A. Shabir, *et al.*, "Efficient cnn architecture with image sensing and algorithmic channeling for dataset harmonization," *Scientific Reports*, vol. 15, p. 7552, 2025. DOI: 10.1038/s41598-025-90616-w. [Online]. Available: https://www.nature.com/articles/s41598-025-90616-w.

[7] A. Bonnet, *What is out-of-distribution (ood) detection?* https://encord.com/blog/what-is-out-of-distribution-ood-detection/, Accessed: 2025-05-26, Sep. 2023.

[8] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 427–436. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Nguyen_Deep_Neural_Networks_2015_CVPR_paper.pdf.

[9] B. Mehlig. "Machine learning with neural networks." version 4. Revised version, 241 pp. arXiv: 1901.05639 [cs.LG]. (Oct. 27, 2021), [Online]. Available: https://arxiv.org/abs/1901.05639 (visited on 06/04/2025).

[10] J. Lu, "Gradient descent, stochastic optimization, and other tales," 2022, arXiv preprint arXiv:2205.00832. arXiv: 2205.00832 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2205.00832.

[11] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015, Accessed: 2025-05-06. [Online]. Available: https://arxiv.org/abs/1511.08458.

[12] *Convolutional layer - an overview | sciencedirect topics*, https://www.sciencedirect.com/topics/engineering/convolutional-layer, Accessed: 2025-05-05.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[14] F. Ramzan, M. U. Khan, A. Rehmat, *et al.*, "A deep learning approach for automated diagnosis and multi-class classification of alzheimer's disease stages using resting-state fmri and residual neural networks," *Journal of Medical Systems*, vol. 44, Dec. 2019. DOI: 10.1007/s10916-019-1475-2.

[15] G. Vrbančič and V. Podgorelec, "Transfer learning with adaptive fine-tuning," *IEEE Access*, vol. 8, pp. 196 197–196 211, Oct. 2020, Received September 2, 2020; accepted October 8, 2020; published October 28, 2020; current version November 10, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3034343. [Online]. Available: https://ieeexplore.ieee.org/document/9241777 (visited on 05/08/2025).

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html.

[17] M. cmglee, *Roc-draft-xkcd-style.svg*, Creative Commons Attribution-Share Alike 4.0, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=109730045, 2023. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/3/36/Roc-draft-xkcd-style.svg.

[18] W. Liu, X. Wang, J. D. Owens, and Y. Li, "Energy-based out-of-distribution detection," *arXiv preprint arXiv:2010.03759*, 2020. DOI: 10.48550/arXiv.2010.03759. arXiv: 2010.03759 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2010.03759.

[19] D. Hendrycks, M. Mazeika, and T. Dietterich, *Deep anomaly detection with outlier exposure*, 2019. arXiv: 1812.04606 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1812.04606.

[20] X. Du, Z. Wang, M. Cai, and Y. Li, *Vos: Learning what you don't know by virtual outlier synthesis*, 2022. arXiv: 2202.01197 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2202.01197.

[21] Y. Sun, C. Guo, and Y. Li, *React: Out-of-distribution detection with rectified activations*, 2021. arXiv: 2111.12797 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2111.12797.

[22] Y. Sun, Y. Ming, X. Zhu, and Y. Li, "Out-of-distribution detection with deep nearest neighbors," *arXiv preprint arXiv:2204.06507*, 2022. [Online]. Available: https://arxiv.org/abs/2204.06507.

[23] H. Wang, Z. Li, L. Feng, and W. Zhang, *Vim: Out-of-distribution with virtual-logit matching*, 2022. arXiv: 2203.10807 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2203.10807.

[24] J. Casselgren, "Road surface characterization using near infrared spectroscopy," Doctoral thesis, ISSN 1402-1544, Ph.D. dissertation, Luleå University of Technology, Luleå, Sweden, 2011, ISBN: 978-91-7439-273-9. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:999512/FULLTEXT01.pdf.

[25] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

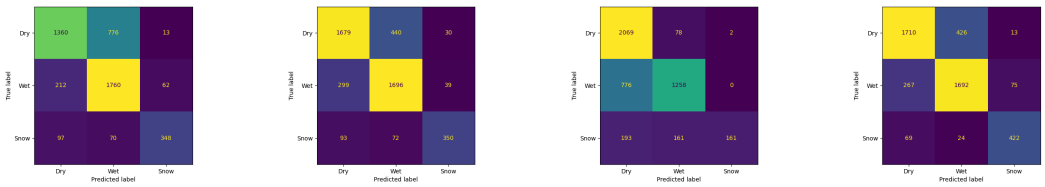
[26] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 3606–3613. DOI: 10.1109/CVPR.2014.460. [Online]. Available: https://www.robots.ox.ac.uk/~vgg/data/dtd/.

[27] D. Hendrycks, S. Basart, N. Mu, *et al.*, "Natural adversarial examples," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

A

Confusion matrices, ROC curves and density plots for all OOD detectors on the Base network

A.1 Confusion matrices for the OOD detectors that alter the Base network



(a) ReAct **(b)** OE on Cifar 10 **(c)** OE on Textures **(d)** VOS

Figure A.1: Confusion matrices for the four OOD detectors that alter the Base network.

A.2 Cars dataset

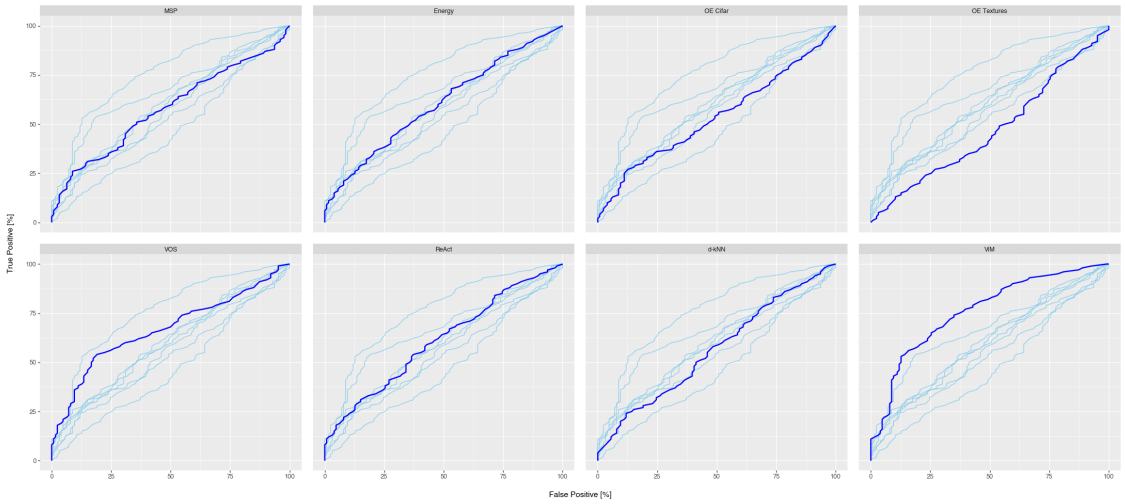


Figure A.2: Operating receiver characteristics for all OOD detectors on the Cars dataset.

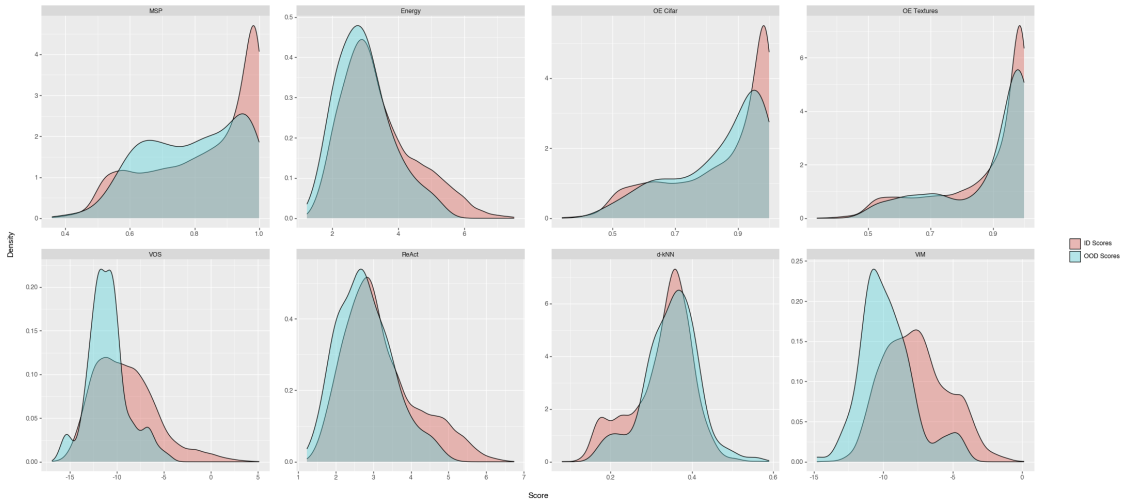


Figure A.3: Density plots of the ID/OOD scores for all OOD detectors on the Cars dataset.

A.3 Glare dataset

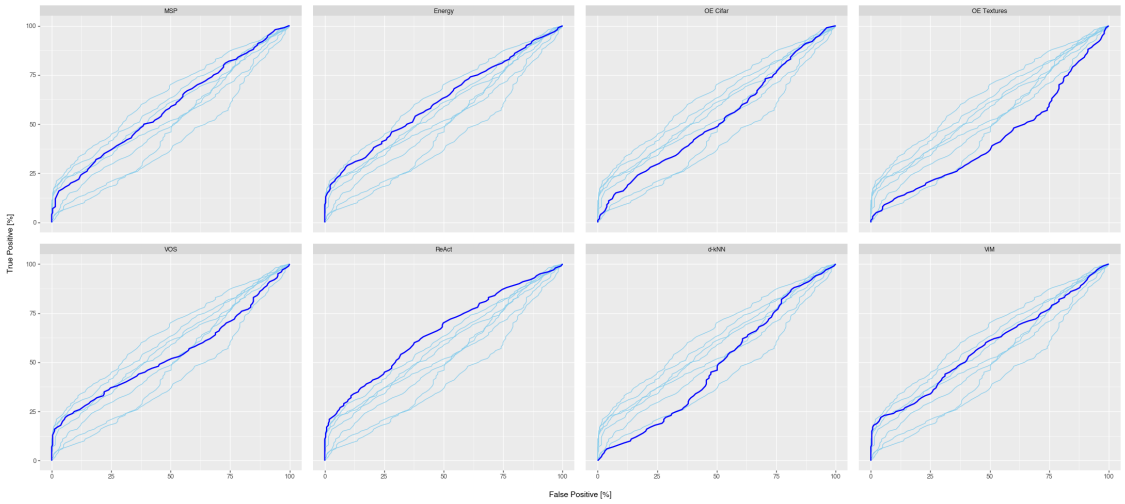


Figure A.4: Operating receiver characteristics for all OOD detectors on the Glare dataset.

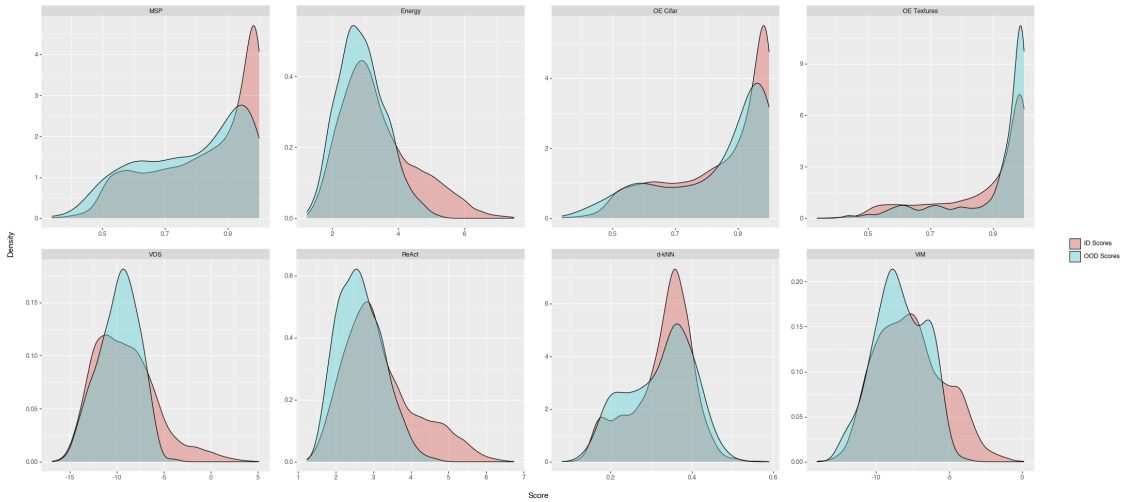


Figure A.5: Density plots of the ID/OOD scores for all OOD detectors on the Glare dataset.

A.4 Slush dataset

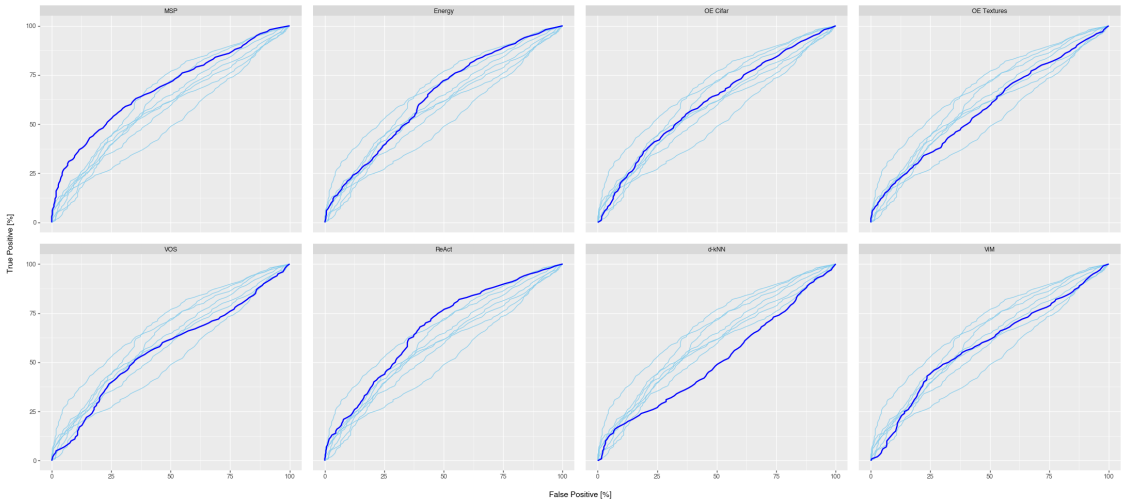


Figure A.6: Operating receiver characteristics for all OOD detectors on the Slush dataset.

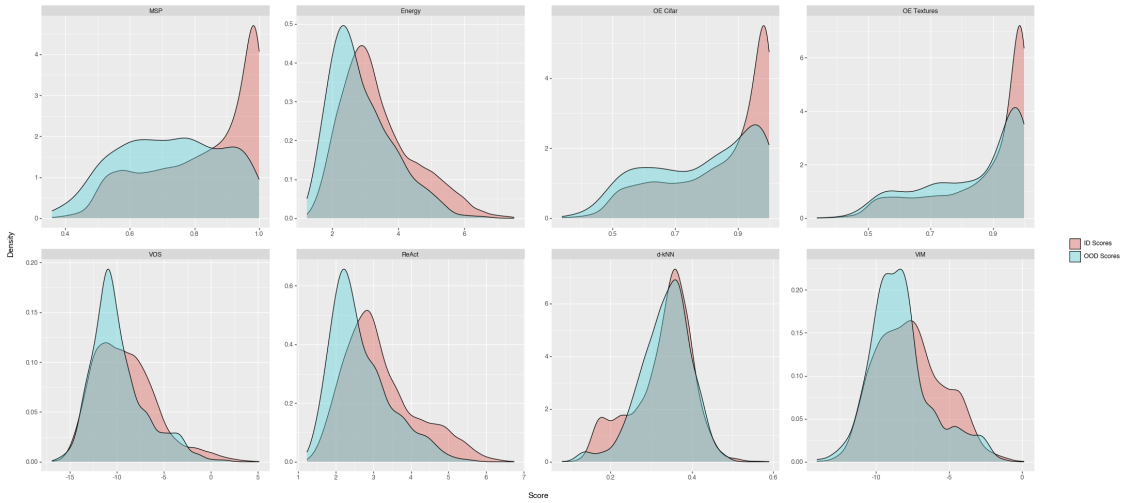


Figure A.7: Density plots of the ID/OOD scores for all OOD detectors on the Slush dataset.

A.5 CIFAR-10 dataset

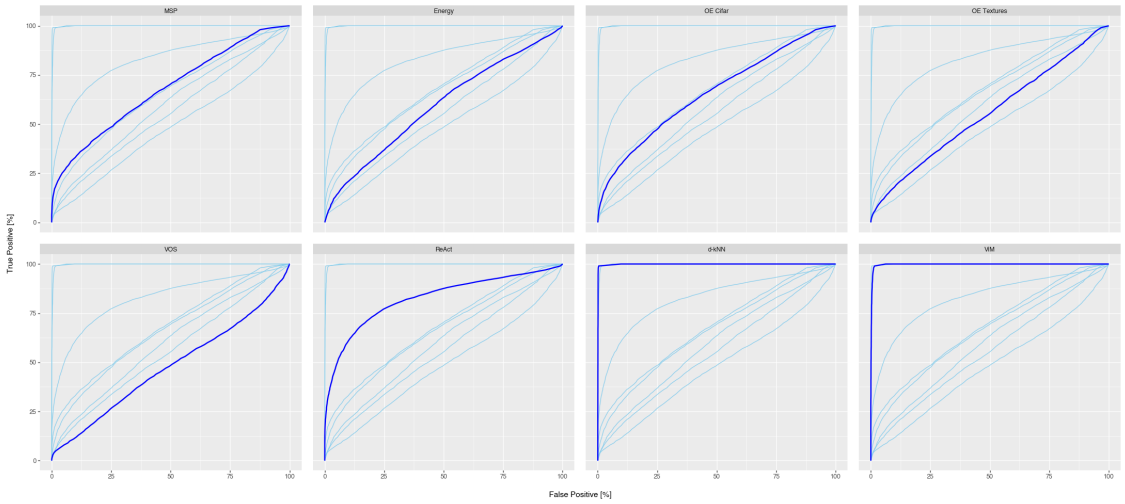


Figure A.8: Operating receiver characteristics for all OOD detectors on the Cifar 10 dataset.

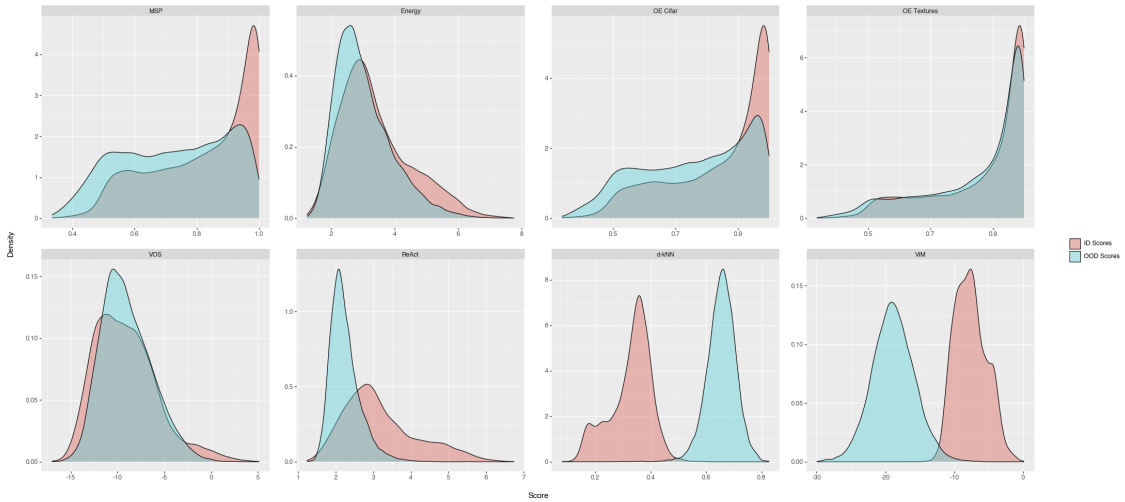


Figure A.9: Density plots of the ID/OOD scores for all OOD detectors on the Cifar 10 dataset.

A.6 Textures dataset

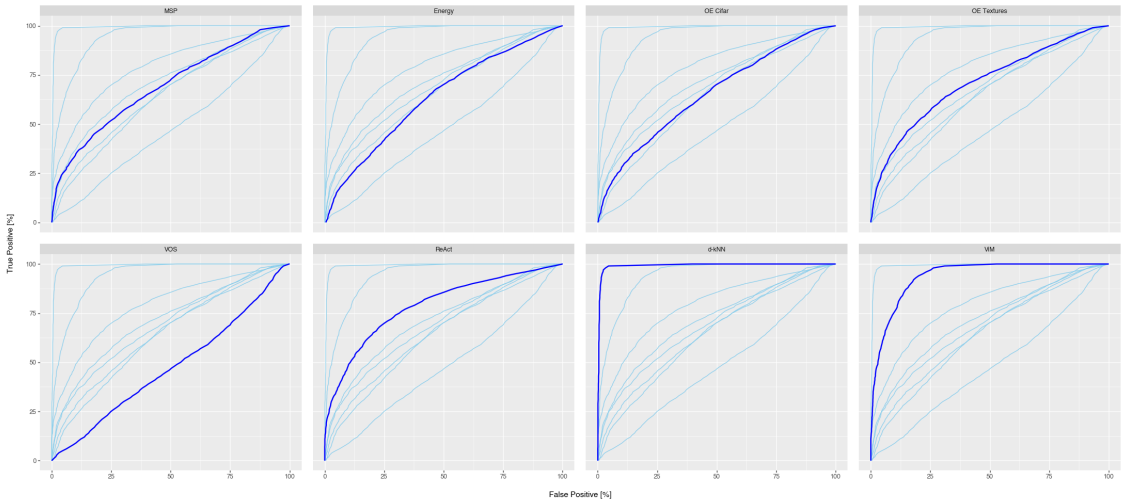


Figure A.10: Operating receiver characteristics for all OOD detectors on the Textures dataset.

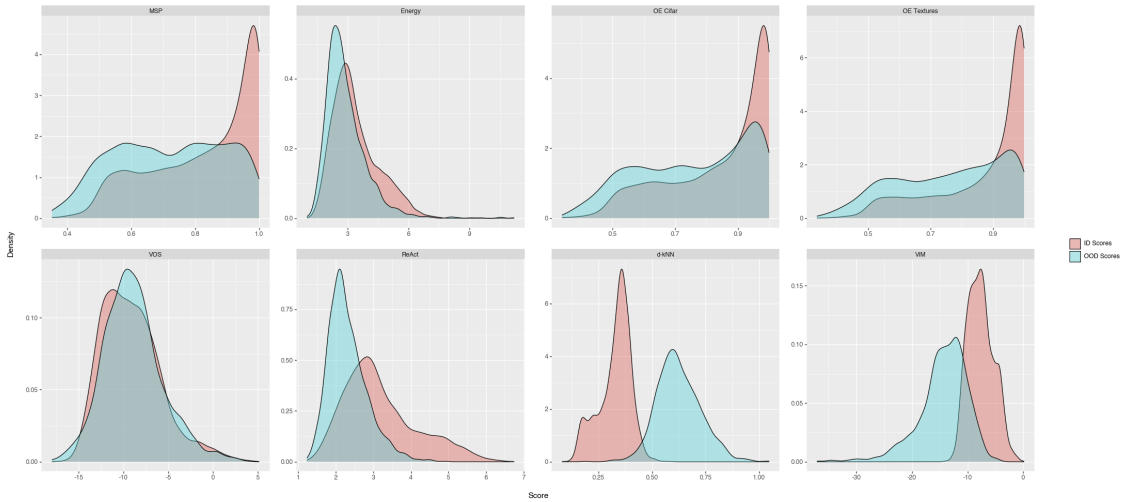
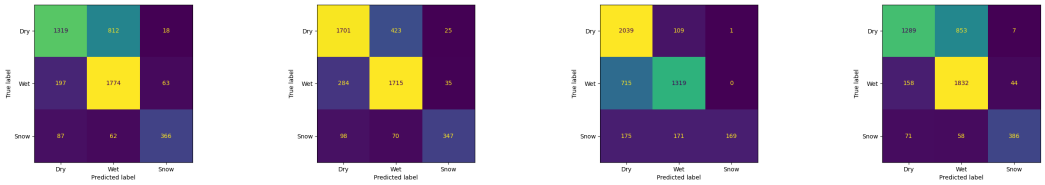


Figure A.11: Density plots of the ID/OOD scores for all OOD detectors on the Textures dataset.

B

Confusion matrices, ROC curves and density plots for all OOD detectors on the Fine-tuned network

B.1 Confusion matrices for the OOD detectors that alter the Fine-tuned network



(a) ReAct **(b)** OE on Cifar 10 **(c)** OE on Textures **(d)** VOS

Figure B.1: Confusion matrices for the four OOD detectors that alter the Fine-tuned network.

B.2 Cars dataset

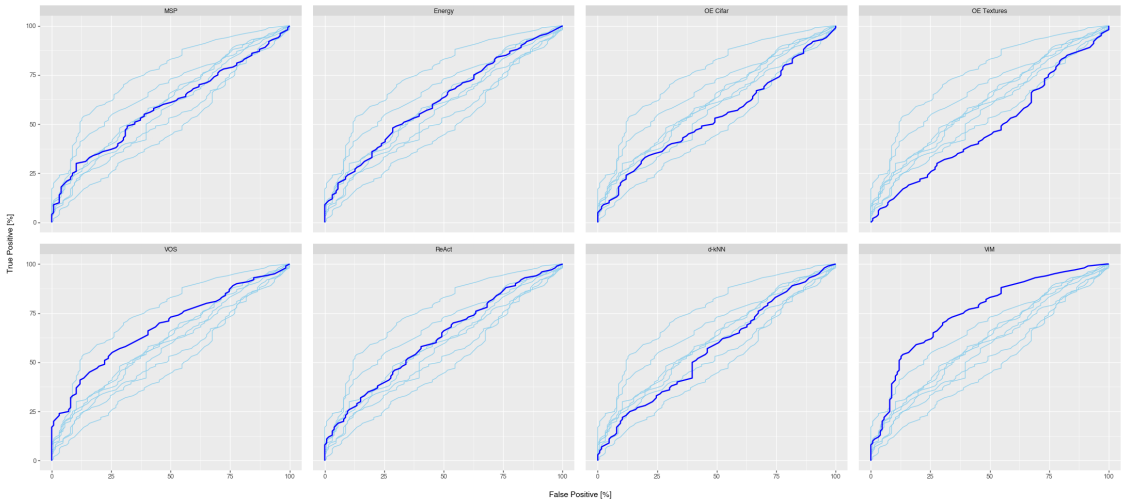


Figure B.2: Operating receiver characteristics for all OOD detectors on the Cars dataset.

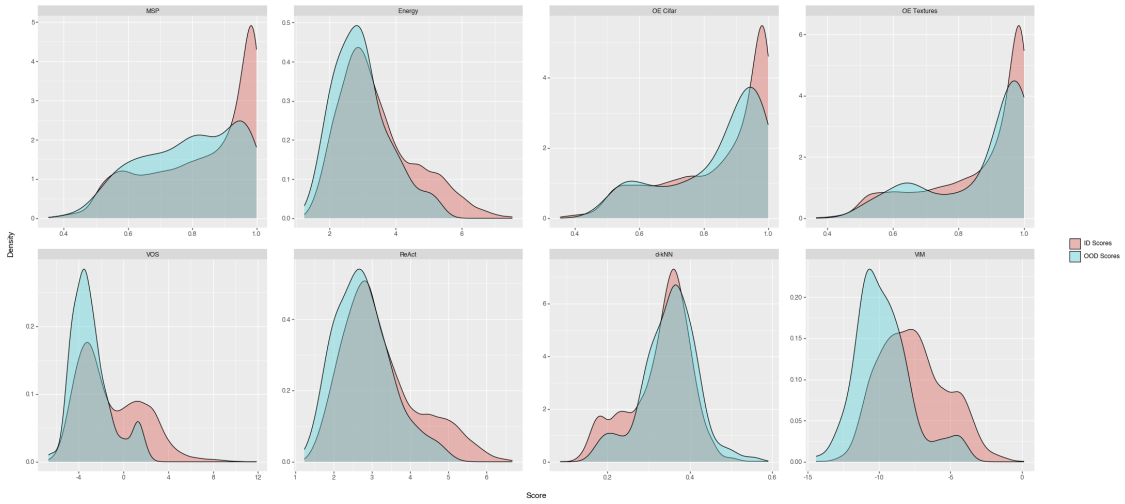


Figure B.3: Density plots of the ID/OOD scores for all OOD detectors on the Cars dataset.

B.3 Glare dataset

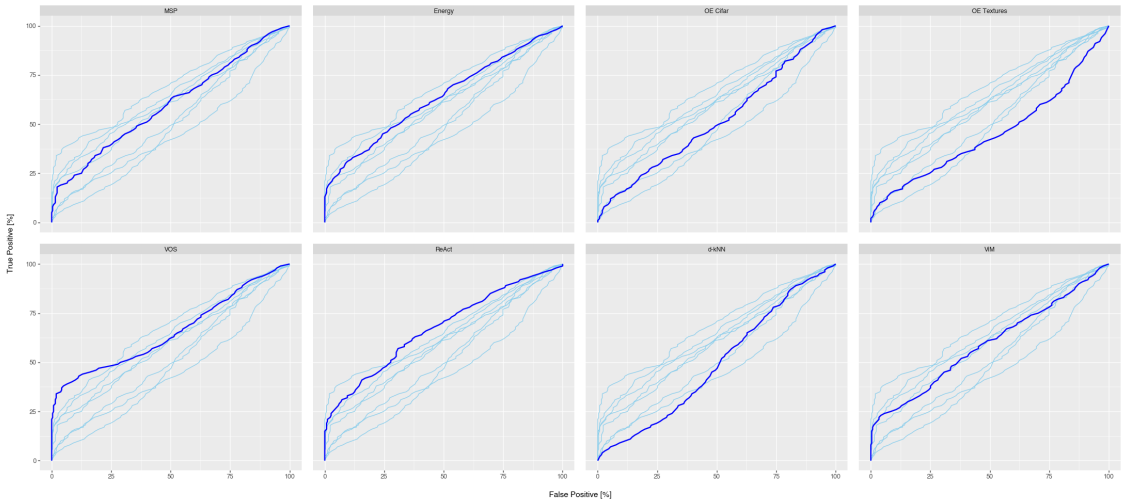


Figure B.4: Operating receiver characteristics for all OOD detectors on the Glare dataset.

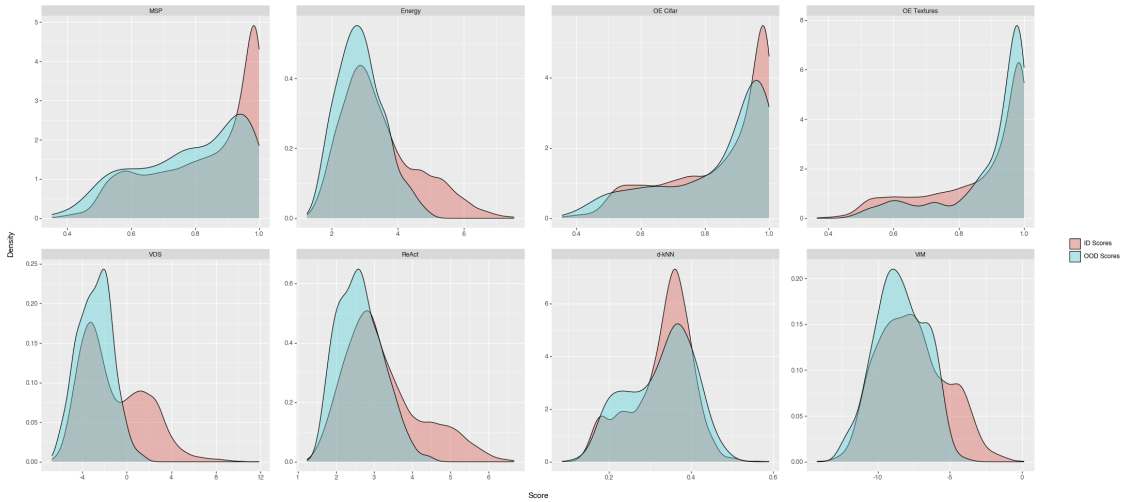


Figure B.5: Density plots of the ID/OOD scores for all OOD detectors on the Glare dataset.

B.4 Slush dataset

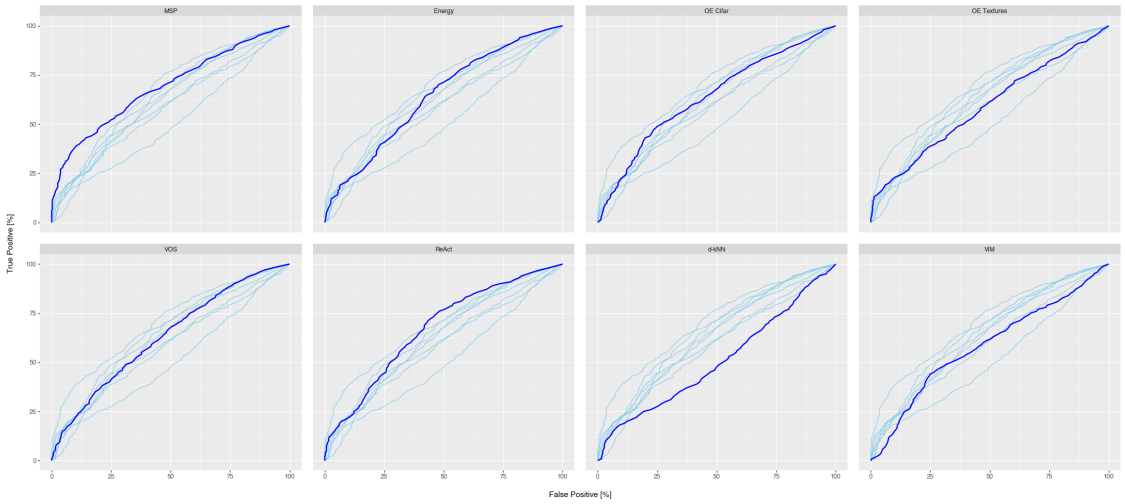


Figure B.6: Operating receiver characteristics for all OOD detectors on the Slush dataset.

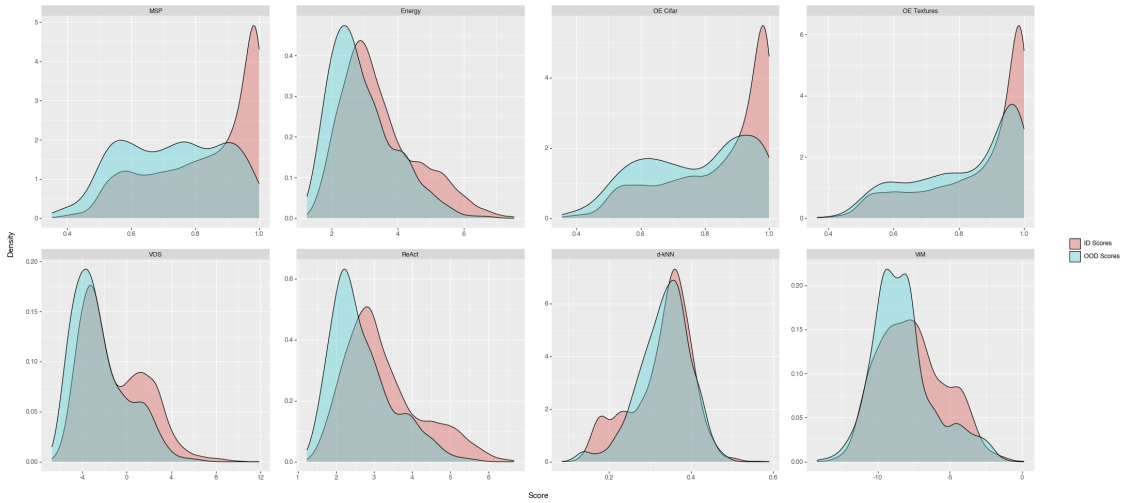


Figure B.7: Density plots of the ID/OOD scores for all OOD detectors on the Slush dataset.

B.5 CIFAR-10 dataset

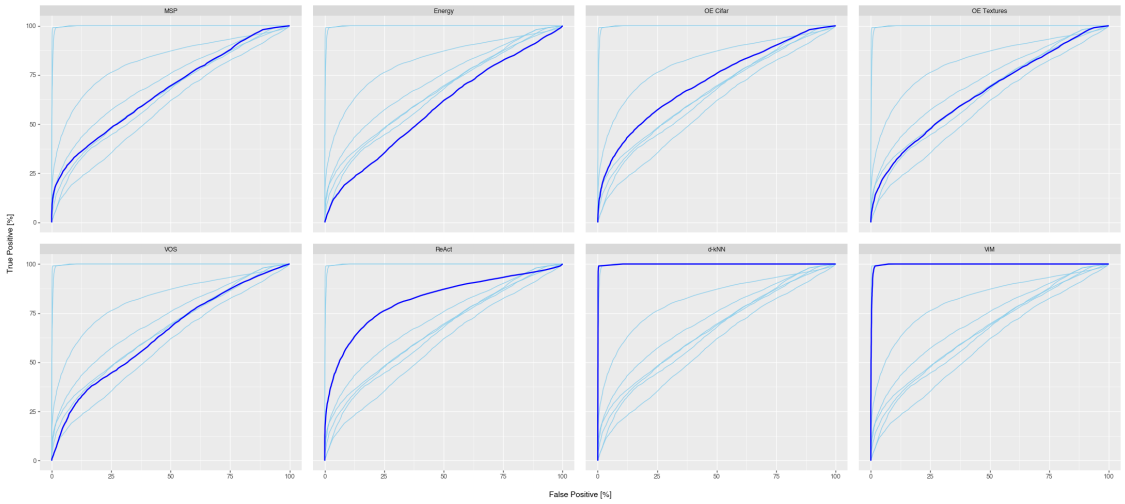


Figure B.8: Operating receiver characteristics for all OOD detectors on the Cifar 10 dataset.

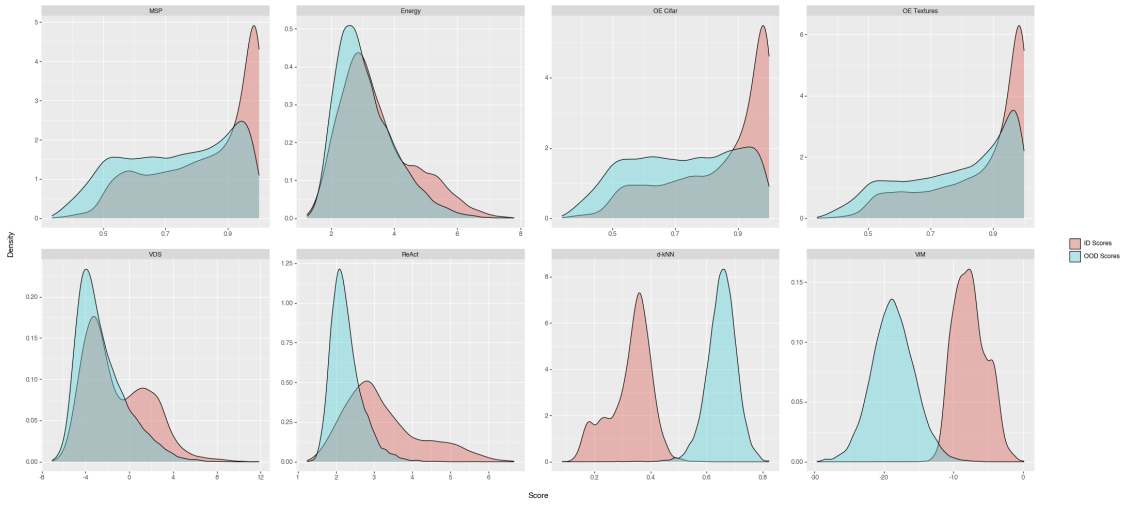


Figure B.9: Density plots of the ID/OOD scores for all OOD detectors on the Cifar 10 dataset.

B.6 Textures dataset

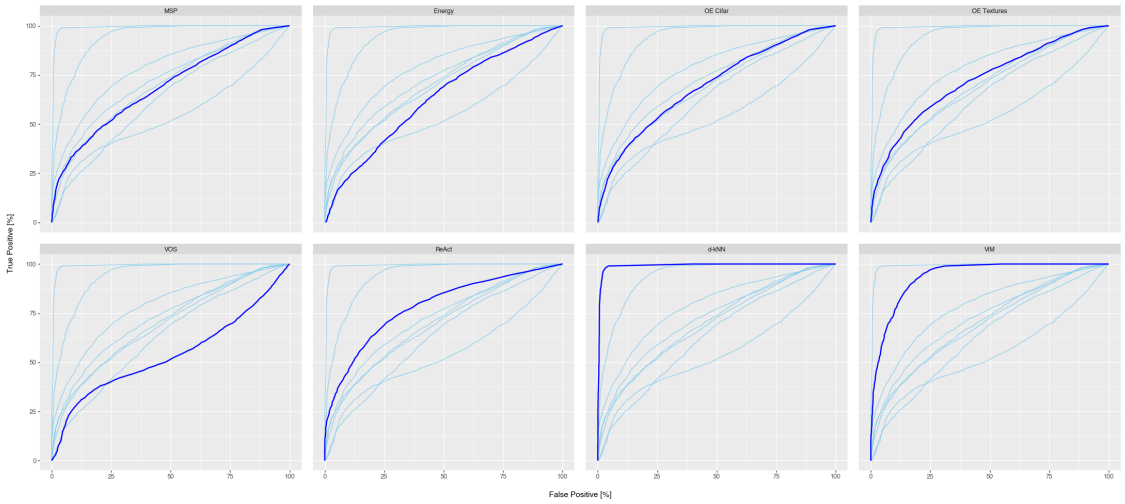


Figure B.10: Operating receiver characteristics for all OOD detectors on the Textures dataset.

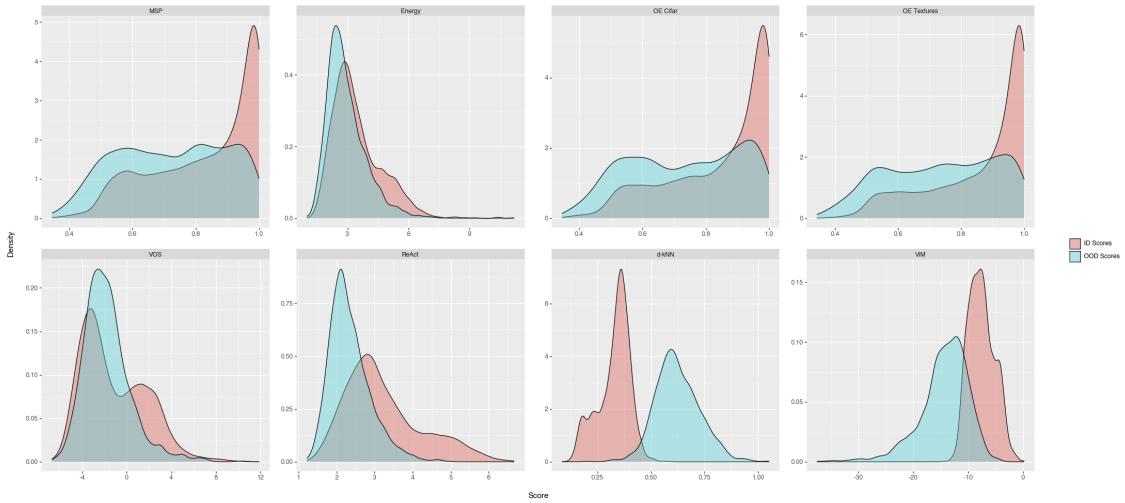


Figure B.11: Density plots of the ID/OOD scores for all OOD detectors on the Textures dataset.

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY