Image classifiers in automated vehicles:

An evaluation framework for robustness

Lovis Heindrich
6629164
July 17, 2020

Master Artificial Intelligence,
Utrecht University

Supervisors:
Jan-Pieter Paardekooper (TNO)
Jeroen Manders (TNO)
Chris Janssen (UU)

Abstract

This thesis reports the development and evaluation of a general safety evaluation procedure for image classifiers that can be applied to automated vehicles. Artificial Intelligence (AI) systems are widely used in automated vehicles, for example to process camera inputs. For the deployment of safety-critical systems, their safety assessment is of critical importance. However, current type approval and safety tests in cars are not suitable for evaluating machine learning components due to their black-box nature and general instability. To fill this research gap, I developed a methodological evaluation framework that tests the robustness of image classifiers. The testing procedure consists of two separate tests. The first test evaluates the ability to function in challenging and novel conditions. The second test evaluates the system's behavior when confronted with out-of-distribution (OOD) examples. Evaluation measures for these tests are accuracy, separability of OOD examples, and overall quality of certainty estimates. I tested the framework on two different classifier architectures: a baseline Deep Convolutional Neural Network (DCNN) and Monte Carlo (MC) Dropout, which is used due to its more accurate certainty predictions. Results of the evaluation showed that different image transformations provide critical tests for both evaluating the classifier on challenging in-distribution images and realistic OOD examples. In this way, the evaluation framework can be used to identify weak points in the classifier's robustness and show their unsuitability for safety-critical applications. Comparing the two classifiers with the evaluation shows the better generalization of MC Dropout, which was not detectable when evaluating with a traditional test set. The evaluation can ultimately be used to define acceptance criteria for the safe use of AI systems, for example, in type-approval for automated vehicles.

Contents

1 Introduction

Artificial Intelligence (AI) systems are increasingly used in the real world [17]. This includes safety-critical applications such as automated vehicles where the AI physically interacts with the real world. Because failures of such systems can have disastrous consequences, their evaluation and safety assessment is becoming increasingly important. This thesis focuses on automated vehicles as it is a representative example of AI facing highly complex environments and challenging scenarios. The complexity of the requirements we place on these AI systems raises a significant challenge: how can we properly evaluate their safety?

Automated vehicles typically rely on individual components for perception, prediction, and decision [27]. The perception component can, for example, be a camera that records the environment. From this camera input, the system has to detect and correctly identify a wide number of objects, such as traffic signs. Then, the information is passed to a prediction component which anticipates future developments of its environment. Finally, a decision component chooses the best action in the current situation. My focus within this framework is the evaluation of the perception component, which is detrimental to the safe functionality of the vehicle since following components rely on its output [27]. While my evaluation focuses on this specific use case, it is defined in a generalizable way that can potentially be applied to other domains as well.

In Europe, all vehicles have to go through a type-approval process before they are allowed on public roads. During this process, the safety of software components used in the vehicles is evaluated. If the vehicle uses AI systems, their evaluation should be part of the type-approval process as well. This, however, is problematic since existing safety standards for software in the industry are not fit for evaluating machine learning systems [19, 37]. Currently, there is no widely accepted testing method for machine learning software components in autonomous systems [20]. The lack of a testing framework makes it hard to assess and compare the proprietary systems used by different manufacturers. Despite that computer vision is a crucial initial first step in many of the processes of automated vehicles, so far, there has been little research on systematic test procedures for such systems in this domain. Instead, current safety evaluation frameworks are mostly designed with the whole vehicle in mind [19, 20] or focus explicitly on decision-making and planning systems [39, 38, 6].

For AI systems in general, robustness to distributional shift is one of the main safety problems [1, 36]. When used in real-world environments, AI systems will inevitably encounter novel and challenging situations. In these situations a distributional shift occurs: the distribution of the situations the AI encounters during deployment is different to the distribution it has been trained on. In these situations, the AI is required to generalise to novel examples. Additionally, the AI should be aware of the difficulty of the new situation by displaying uncertainty [1]. These robustness properties are needed for safely using AI in the real world. To make sure the AI is sufficiently robust, manufacturer's software systems need to be systematically assessed and tested before they are allowed to be deployed.

1.1 Research question

This thesis is a contribution to the bigger scientific question: What should an evaluation pipeline for AI-functions in safety-critical applications look like? As an instantiation of this question, I investigate the specific problem: How can the robustness of computer vision systems in safety-critical environments be automatically and systematically evaluated in the domain of automated cars? To answer this question, I develop a systematic evaluation pipeline for AI systems that evaluates their accuracy and uncertainty estimates in the context of distributional shift. More specifically, I test the robustness of computer vision systems in the form of convolutional neural networks (CNN). The proposed evaluation pipeline consists of a series of detection and classification tasks that quantify how well the tested classifier can generalize to novel situations.

My proposed evaluation pipeline is split into two parts that evaluate different properties of the tested classifier: classification evaluation and out-of-distribution (OOD) detection. The classification evaluation tests, if the classifier functions on difficult in-domain images. Out-of-distribution detection is the task of identifying whether an image belongs to the domain of the classifier (this is further explained in section 2.3). Figure 1

visualizes the two types of tests performed within my evaluation.

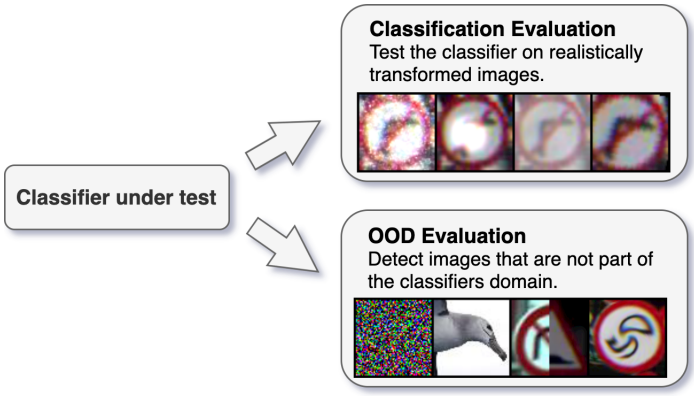


Figure 1: The pipeline evaluates two different properties: Classification performance and OOD detection.

This thesis is relevant to the field of Artificial Intelligence because it directly investigates evaluation techniques for machine learning classifiers. The developed evaluation methodology contributes to the broader field of AI safety and has practical uses in the domain of automated vehicles.

1.2 Relevance to safety in automated vehicles

Since computer vision systems are widely used in automated vehicles, the evaluation of image classifiers is highly relevant to their safety. Automated vehicles that make use of cameras in systems such as lane-keeping assistants or automated parking are already commercially in use. In the future, the importance of computer vision systems and the software of vehicles, in general, will only increase as driving becomes increasingly automated. For image classification, machine learning and, more precisely, neural networks are the unrivalled state-of-the-art performer [21, 43]. There are no viable alternatives to using CNNs. Neural-network-based approaches have therefore been widely used in safety-critical applications [7].

The lack of robustness in AI components causes problems in the domain of automated vehicles. The number of different situations a car can encounter on the road and needs to respond to is extremely large, making exhaustive testing impossible [19]. Having a set amount of experience on the road does not mean the car has encountered all possible situations and gives no guarantee for correct behavior in all situations [19]. The sequential nature of different components causes an additional security risk because a critical error in the perception of an automated car has the potential to cause more errors in later decision-making stages [27]. A proposed solution to this are components that can give accurate uncertainty measures instead of only the class prediction which allows later components to take the error potential into account and act accordingly [27]. Salay et al. (2020) have made use of these perceptual uncertainties to define a set of safe behavior rules [38]. This shows the importance of achieving meaningful certainty estimates in the system's perception, which can be evaluated with my proposed evaluation.

In the automotive industry, mainly two relevant standards for developing functionally safe software exist: ISO 26262 [15] and ISO/PAS 21448 (SOTIF) [16]. The safety standard ISO 26262 addresses safety by assessing risks caused by hazards or malfunctions of the system. The standard also addresses software used in vehicles, where it ensures safety through a rigorous development process, including extensive verification and testing [37]. ISO 26262 has recently been extended by the SOTIF standard, which covers unsafe behaviors when the system is functioning as intended [37].

These current safety standards are not well equipped to deal with AI components due to their black-box nature, and the current testing procedure will have to be adapted to incorporate the evaluation of machine learning software [20, 37]. Two central problems with applying current safety standards are specification and interpretability. Since neural networks learn from data instead of being explicitly engineered, it is impossible to apply the ISO standard software developing process that requires the software's requirements to be precisely defined. Additionally, in machine learning systems the ability to trace faults and errors is limited [37]. Since current safety standards cannot be applied to machine learning methods, the current testing process relies heavily on road experience where errors are eliminated by testing the system during deployment, which is impractical due to the large amount of required experience [20].

The Dutch Safety Board released a report on the safety of automated vehicles and found a lack of legislation regarding advanced driver-assistance systems (ADAS) [3]. They investigated several accidents of which some can be directly linked to failures in the car's perception such as a collision where a Volvo truck failed to engage its automatic emergency braking system because the camera system did not recognize the truck in front of it correctly. The causes of these accidents show the need for an evaluative framework that assesses software components in autonomous or semi-autonomous vehicles.

1.3 Scope and contribution

The main contribution of this thesis is the definition of the evaluation tests the pipeline consists of. In the OOD evaluation, a large number of OOD datasets are created automatically to provide different test scenarios. The majority of these datasets are novel test sets and aim to make the evaluation more varied, extensive and challenging. The classification evaluation consists of image transformations. Here, I extend existing data augmentation techniques to be used during test time. This approach is used to evaluate the performance of the classifier beyond the existing test set, approximating difficult real-world scenarios the system might encounter during deployment. The whole pipeline is designed to require a low amount of human interaction to be used on a new dataset. The OOD datasets and classification evaluation data transformations can be created programmatically without requiring many parameters to be tuned. The automatic generation of challenging conditions separates this work from other datasets that rely on hand-crafted image manipulations [44].

To demonstrate the functionality of the pipeline, an experiment using two different classifier architectures is run. The experiment is meant to show how the pipeline evaluates the different tested criteria (OOD detection, classification performance, calibration). The actual performance of the classifiers is not relevant and newer and better performing OOD detectors exist (e.g. [4, 12, 24, 26, 28]).

The verification of AI systems such as neural networks is a fundamental challenge [7]. Existing solutions for the robustness evaluation of neural networks are mostly based on adversarial attacks [8]. These evaluations focus on the ability of the network to function under small changes to the input [7, 13]. In real-world applications, however, robustness against small perturbations is not enough. Due to the infinitely large number of potential situations, perturbations can not realistically be bounded as they are in the framework of adversarial attacks. In real-world environments, complete testing is infeasible because the creation of an exhaustive dataset containing all realistic situations is not possible [19]. In my evaluation, I approximate this problem of infinite test cases by choosing a wide range of different data augmentations that try to achieve a meaningful overview of the classifier's performance. This approach offers a scalable way to evaluate the classifier without requiring additional data.

An additional problem is that classifiers not only often fail to predict examples far from the training distribution correctly, but also the failure often happens unnoticed [11]. These silent failures happen because predictions of neural networks are generally poorly calibrated [10]. While neural networks usually give predictions as a probability for each class, these probability estimates are often not representative of the true probability of the prediction being correct. Classifiers are often overconfident in their prediction, giving a very high activation for the predicted class, even if they are wrong [1, 41]. My evaluation pipeline takes

this into account and evaluates not only the correctness of the classifier's prediction but also the classifiers calibration: how meaningful the actual predicted probability values are.

The classification pipeline makes heavy use of data augmentation, which is a common training strategy that aims to increase the variety of training examples by applying transformations such as a rotation to the images. Data augmentation is widely used to improve the training of neural networks and make the network more robust and mostly used during the training of the network [40]. Data augmentation can also be applied during test time [40], which has, for example, been utilized in the medical domain to improve classifier robustness [46, 34]. Minh et al. (2018) [29] improved the performance of image classifiers by training the classifier with a chain of transformations discovered by using reinforcement learning and then testing the network with random transformations. In this work, I use test time augmentation with a different purpose. Instead of improving the classifiers, the data augmentation is used to be able to test them more extensively in realistic scenarios. The classifier's training procedure is not relevant. The use of test time augmentation allows the automatic creation of novel and more difficult test cases.

The task of OOD detection has been approached frequently (e.g. [11, 25, 26]) and is crucial to deal with the safety concern of distributional shift: the ability to identify inputs that are different to training inputs. The goal of the OOD evaluation is to test the classifier's behavior on images that are very different from its training data. This includes realizing when encountering a novel situation and not giving a confident prediction for images that do not belong to the classifier's domain. Usually, classifiers simply predict which of the available classes a given input most likely belongs to. However, the behaviour on inputs that do not belong to any class is undefined. As a domain-relevant example, we consider a classifier trained to recognize different traffic signs. This classifier should not output a prediction with a high certainty for any class when shown images that don't contain a traffic sign at all. These OOD images can be anything else that the car's camera records such as pedestrians, cyclists, other cars or an empty road. It is not uncommon to see very high certainty in a classifier's prediction, even when given pictures of pure noise [25].

In the area of OOD detection, previous work has shown the limitation that the used OOD datasets are too easy, it is not uncommon to see very high Area Under the Receiver Operating Characteristics (AUROC) scores of around 99% (e.g. [4, 12, 24, 26, 28]). In general, existing research mostly focuses on finding good detection algorithms and not so much on evaluating the detection performance. Due to this fact, most proposed detection algorithms are only evaluated with relatively simple, publicly available datasets (e.g. [4, 12, 24, 26, 28]). Even not particularly robustly trained classifiers (such as the baseline used by Hendrycks et al. (2016) [11]) perform well on these datasets, which indicates to me that the used datasets are not a good representation of the space of possible OOD images. Since the set of possible OOD images is too large to test exhaustively, it is necessary to find meaningful subsets that give representative performance estimates. Showing that a classifier can distinguish real examples from easily detectable OOD examples does not properly assess the classifiers ability to do so in later deployment in the real world. Additionally, easy test sets make improvements through new approaches to OOD detection hard to measure and compare. I overcome these issues by using novel and more challenging test sets beyond using publicly available image data. I believe this is beneficial to further evaluate the detection mechanisms on increasingly difficult tests.

The combination of these the two evaluation aspects (classification evaluation and OOD detection) is a novel approach to classifier evaluation because these two properties have not been looked at in combination. The goal of this approach is to test the classifier in a way that shows its robustness and suitability for use in safety-critical applications. For this, both the OOD detection and classification evaluation are highly relevant and only performing well in one of the two is not sufficient. To be sufficiently safe, a classifier needs to give meaningful certainty estimates, show uncertainty in challenging cases, and still maintain the ability to generalize well and correctly classify difficult images.

1.4 Importance

This research is useful for society because it will contribute to making the use of AI in real-world applications safer. As more automated functionality (such as lane-keeping assistants) is used in vehicles, the safety of these systems must be ensured. Ultimately, the complete pipeline can be used to evaluate AI components in automated vehicles and other safety-critical applications. In the domain of automated vehicles, this could contribute to the approval procedure (type-approval) of cars with automated components. The main advantage of the pipeline is that it offers a standardized and automated safety evaluation, which is needed for the safe use of AI in the real world.

Additionally, this work can contribute to science and engineering by extending the scope of testing done in machine learning and offers a novel approach to evaluate the robustness of a pre-trained image classifier. The evaluation of classifiers is often limited to the used test set, which does, in many cases, not reflect the full operational domain of the AI. Since we require the AI to function in unknown environments, the tests we use to evaluate the AI also need to go beyond known examples. The pipeline evaluation tries to offer a more general approach applicable across domains, classifiers, and datasets. While not offering a way to test classifiers exhaustively, the pipeline extends the test set significantly and offers a more comprehensive evaluation through the automatic generation of challenging examples.

1.5 Outline of the pipeline

The general purpose of the pipeline is to evaluate the robustness of a given image classifier by creating a number of increasingly difficult detection and classification tasks. Since the tasks in the pipeline can be generated automatically, this can be applied to any pre-trained image classifier and any dataset. The generated tasks are split into two categories: out-of-distribution detection and classification. This leads to the following structure: The input of the evaluation pipeline is the classifiers that are being tested. Then, the classifier solves several OOD detection and classification tasks of increasing difficulty. The results of the tasks are summarized in the analysis, giving a comprehensive performance overview of the classifier.

In the first part of the evaluation pipeline, the classification evaluation (reported in detail in section 3.1.1), the classifier's behavior on difficult in-domain examples is tested. In real-world applications, classifiers encounter novel situations and need to be able to generalize well beyond the known training examples. For example, a classifier that has been trained to detect traffic signs needs to recognize traffic signs in a wide number of different lighting conditions and different viewing angles. If however all the training images were recorded during broad daylight the classifier will probably not function well at night.

This problem can be overcome using test time data augmentation techniques to create more difficult images the classifier can be tested on. Applying different image transformations during test time allows evaluating the classifier in unseen scenarios. The data augmentation is only applied during test time and not during training, which is motivated by the fact that the pipeline is meant to evaluate classifiers regardless of their training procedure. Additionally, to be able to use this kind of evaluation during type-approval, the evaluation needs to be independent of the creation and training of the machine learning systems. The data augmentation is used to create challenging test images under distributional shift. This evaluates how sensitive to perturbations the classifier is and how well it can generalize to unknown examples that show structural differences to known training examples. Additionally, the classifier's raw performance and robustness may affect each other [45]. Achieving higher robustness could, therefore, negatively impact the raw accuracy of the classifier. The possible interaction emphasizes the necessity to examine the classification and the OOD evaluation simultaneously.

The OOD evaluation (reported in section 3.1.2) in the pipeline evaluates the classifier by generating different OOD datasets on which the classifier is tested. These consist of images that are not part of the domain the classifier is trained on and are generated in several different ways. The OOD images are then mixed with the original images and, using Hendrycks et al. [11] detection mechanism, the classifier has to separate real in-domain test images from the OOD images. One example of the OOD detection task

is the use of random noise as out-of-distribution examples. The classifier then has to distinguish between actual in-distribution and noise images. Other ways to generate OOD datasets include the incorporation of the transformations from the classification evaluation, and generative adversarial network (GAN) generated images. These datasets can be generated automatically, which means they can be easily adapted to new domains since the test conditions do not need to be hand-crafted.

A novel approach to create more difficult OOD datasets is to use the transformations introduced in the classification evaluation. Reusing these transformations on both the in-distribution and OOD images increases the difficulty of separating them considerably since it creates a lot of visual similarity between the two. The intuition behind this is that since we expect the classifier to still classify correctly in challenging conditions (as evaluated in the classification evaluation), we also need to expect it to detect OOD examples in challenging conditions as well. As an example, the black-and-white effect from the classification evaluation can be applied to both the noise OOD image and the original test image. Another approach used to create OOD images is the use of GANs. Repurposing GAN networks from the literature ([25, 41]) allows generating novel images that fulfil specific properties such as vaguely resembling an in-distribution image visually or causing an unusually high response from the classifier.

In the final pipeline output, the performance evaluation of the classifier is summarized and gives indications regarding the following questions about the robustness of the classifier:

- How reliably can the classifier detect out-of-distribution examples?
- How well can the classifier function under difficult conditions?
- Does the classifier show reliable certainty estimates for its predictions?

Running the pipeline with multiple classifier candidates gives a detailed comparison regarding these factors. Ultimately, this could be used during type-approval when assessing whether a classifier is safe to be deployed. Defining different use-case-dependant thresholds or baseline measures the classifier has to surpass allows assigning pass/fail criteria to specific tests.

The whole pipeline is designed to function as a black-box evaluation, which means the pipeline does not require access to the inner working of the classifier under test. Instead, the classifier is treated as an oracle, where the only information needed to test the classifier are the test data and the classifier's predictions on the different tests. The main advantage of this is that the tested classifier does not have to be made public. This could potentially allow testing classifiers used by a company in systems such as automated cars without requiring the company to make their internal research public. As long as the evaluation can be used to assess the system's functional safety, knowledge about its implementation is not required.

2 Technical Background

In the scope of this thesis, a general understanding of deep neural networks is assumed. Especially deep convolutional neural networks (DCNN) are used extensively throughout the experiments. The most relevant aspects for this thesis will be explained in the following sections.

2.1 Neural networks

I will explain two layer types commonly used in neural networks that are relevant for this thesis: dropout layers and the softmax activation function.

Dropout is a type of layer that is often used in neural networks to prevent overfitting [42]. The dropout layer introduces a stochastic element to the network by randomly deactivating a proportion of the network's neurons. In this thesis, I will use the concept of dropout layers to to implement the MC Dropout architecture [5] (see section 2.4.1), that is more robust and has more accurate certainty estimates. On a technical level, the

dropout layer works by randomly setting a proportion of the previous layer's output to zero. The proportion of zeroed neurons is controlled by a factor that represents the chance of ignoring a neuron's output. Usually, dropout is only used during training. During a models deployment, the dropout layers are deactivated to make use of the full power of the neural network.

Another important concept is the softmax activation function, which is commonly used in the last layer of neural networks. It regularizes the raw outputs of the neurons so that all output neurons together sum to one. This effectively transforms the numerical output of the network to class probabilities. The softmax layer is essential for the function of my evaluation since the analysis of the networks certainty estimates as well as the OOD detection is based on this layer's output.

2.2 GANs

Generative adversarial networks (GAN) are a category of generative models first proposed by Goodfellow et al. (2014) [9]. The main advantage of GANs is that they can automatically generate novel data by learning the structure of the training examples. The ability to automatically create new examples is relevant since the generated examples are used in the pipeline to evaluate the classifiers. In the context of OOD detection, I apply modified GAN architectures (see section 2.4.2) to generate novel OOD examples.

Classical GANs try to generate novel examples that are indistinguishable from the real images. This is achieved by jointly training two different networks, a discriminator and a generator. The generator is tasked to create new examples from random noise. The generator does not have access to any real examples of the training dataset. The discriminator works in tandem with the generator. Its inputs are fake examples from the generator as well as real examples. During training, the discriminator's predictions are backpropagated to the generator, which allows updating both networks simultaneously: The discriminator is trained to distinguish fake from real examples. In contrast, the generator's task is to produce examples that lead to a high loss for the discriminator. Over the last years, GANs have been extended in many ways. A relevant extension is the Wasserstein loss [2], an improved loss function. This loss function has been shown to stabilize the training procedure and improve the quality of the results. While the original GAN used networks of only fully connected layers, Radford et al. (2015) [35] proposed a modified architecture using convolutional layers (DCGAN) and gave practical tips on the training procedure.

2.3 OOD detection

Hendrycks et al. (2016) [11] first introduced the OOD detection task. Here, a classifier has to decide if given examples belong to the original data distribution or not. In the paper, this is evaluated simultaneously to the related task of predicting erroneous outputs, where correctly and incorrectly predicted examples have to be separated. The author introduces a baseline on the OOD detection task, which is based on the classifier's softmax output. Even though the classifier's softmax predictions generally don't produce accurate predictive certainties, as a baseline measure, they can be used to classify OOD examples without any adjustments to the classifier. This is done by taking the maximum softmax prediction, which can be interpreted as the classifier's certainty in its prediction. The baseline detector is motivated by the observation that, on average, the output for in-distribution examples is higher than the output for out-of-distribution examples. The classifier is generally more certain of its prediction when predicting a real example than of an OOD example. The described approach to OOD detection is extensively used for evaluating the OOD detection on my own test datasets.

The AUROC score (a measure for the classifier's performance in binary tasks) is used to measure the performance on the OOD detection task [11]. For this, in-distribution examples are treated as the positive class and OOD examples as the negative class. Using the AUROC score allows measuring the separability of the two without the need to define an explicit threshold. Intuitively, the measure represents the probability that

an in-distribution example has a higher activation than an out-of-distribution example [25].

Hendrycks et al. (2016) [11] test their OOD detection method using random noise and publicly available image datasets as OOD datasets. Ideally, a classifier outputs a uniform distribution for OOD examples and a high certainty for in-distribution examples. However, this is not required for a high AUROC score which only evaluates how well the two classes can be separated from the prediction. An example of this is a classifier that is given a testset of real examples and random noise. Taking the average certainties of the classifier could, for example, result in an average activation of 95% for in-distribution examples and an average activation of 75% for out-of-distribution examples. This allows to relatively easily distinguish between the two by, for example, using 85% certainty as a threshold.

2.4 Confident classifiers

Many approaches to creating more confident classifiers that can detect OOD examples exist (e.g. [11, 25, 4, 12, 24, 26]). In the next two sections, I will go into detail about two relevant techniques: MC Dropout [5] and a modified GAN architecture [25].

2.4.1 MC Dropout

The goal of the MC Dropout (Monte-Carlo Dropout) architecture is to output more accurate certainty values. To achieve this, Gal et al. (2016) [5] developed the MC Dropout framework that allows representing model certainty by approximating a Gaussian Process with the extensive use of dropout layers. The main idea is to sample the network multiple times while keeping the dropout layers active. This leads to slight variations in the prediction (since different neurons are activated in each forward pass). The resulting predictions are then averaged to obtain the final prediction of the model. In my evaluation, the improved certainty estimates of MC Dropout should lead to a better performance in comparison to the baseline classifier for the OOD detection task and the evaluation of certainty estimates. This method is, in addition to the baseline method by Hendrycks et al. [11], used in the experiments.

2.4.2 Lee GAN

Lee et al. (2017) [25] have implemented a confident classifier that is trained with the help of a GAN. The GAN is used to improve the classifier's robustness by generating images that lie on the edge between in-distribution and OOD examples. This is achieved by using a modified loss term that minimizes the Kullback-Leibler (KL) divergence between the classifier's prediction and a uniform distribution. The new loss function steers the generator towards images the classifier is unsure about. Combining the KL loss term with the original GAN loss results in images that are visually similar to the original training images (due to the GAN loss) but still OOD (due to the KL loss). The authors argue that this effectively creates images on the boundary of the training distribution. The classifier is jointly trained with the GAN to predict the correct class for in-distribution examples while predicting a uniform distribution for the generated OOD examples. Using these boundary examples to train the classifier translates well to other OOD examples and achieves good robustness in the classifier.

Using the same GAN architecture, Sricharan et al. (2018) [41] extend the previous work with a modified loss function. While the generator as used by Lee et al. (2017) [25] was incentivized to generate visually similar images the classifier is unsure about, Sricharan et al. (2018) [41] generate images without visual similarity that the classifier is sure about. This is implemented by training the generator to maximize the KL divergence instead of minimizing it, steering the generator towards examples the classifier is sure about. The standard GAN loss is inverted as well to avoid simply generating in-distribution examples. The generator, therefore, creates examples that are easily distinguishable from in-distribution examples but cause a high activation in the classifier.

In the context of this thesis, the GANs proposed by Lee et al. (2017) [25] and Scricharan et al. (2018) [41] are used solely for their capabilities to generate OOD examples. The generated images are then used as

OOD test datasets.

3 Method

3.1 Architecture of the pipeline

This section will discuss the design of the two pipeline components highlighted in Figure 1. Firstly, I discuss the classification evaluation. This part of the full pipeline evaluates how well the tested classifier performs on difficult images created through transformations such as adding a fog effect. Secondly, I describe the OOD detection task where the same tested classifier has to detect which images belong to its domain and which images are out-of-distribution (OOD).

The input of the pipeline is the pre-trained classifier that is being evaluated and a test dataset of images. The evaluation is designed to function in a black-box setting. Accessing the classifiers weights and architecture is not necessary for the evaluation.

3.1.1 Classification evaluation

The classification evaluation aspect of the pipeline aims to test classifiers ability to correctly classify images under challenging conditions. This is implemented using image transformations to realistically distort the test examples in a way that makes them harder to identify. Ten different image transformations have been implemented to test the classifier.

The used image transformations (see Figure 3) loosely follow Temel et al. (2017), who introduce a traffic sign recognition challenge where a complex image dataset of traffic signs is created by transforming the images with varying effects such as simulated rain or noise [44]. Contrary to Temel et al. (2017), the transformations in my pipeline are used to evaluate pre-trained classifiers instead of creating a more difficult training set. The actual training procedure of the classifiers under test is irrelevant; the pipeline aims to evaluate their generalization capabilities without taking the classifier's architecture or training procedure into account. Additionally, the image transformations used in the pipeline are generated automatically by programmatically applying image transformations created in code, while the image transformations by Temel et al. (2017) were hand-crafted. This allows using the classification evaluation in the pipeline on arbitrary datasets of any size without increasing the necessary amount of manual work.

Firstly, the classification evaluation is concerned with the tested classifier's accuracy on the transformed images. Also evaluated are the certainty estimates output by the classifier. This is relevant since machine learning classifiers have a tendency to be overconfident in their predictions [25] and are generally bad at producing accurate predictive certainties [22]. Expressing truthful certainty values is not only useful in the context of OOD prediction, but it is also highly relevant in any real-world systems where decisions are based on the certainty of the perception. For example, in the domain of autonomous cars, Shalev-Schwartz et al. (2017) [39] propose the Responsibility-Sensitive Safety (RSS) model, which defines safety assurance requirements of an autonomous car's actions. These rules rely on the perception component of the car and assume that all perceptions are correct [39]. This framework has later been extended by Salay et al. (2020) to allow for uncertainties in the vehicles perception [38]. This extended framework still requires correct uncertainty values from the perception module to function.

The test procedure of the classification evaluation is the following: The image transformations are applied to the test dataset, resulting in a new test set for each applied transformation. Additionally, each image transformation is created in five different levels of strength, creating different difficulty levels within each transformation. The classifier under test is then used to classify each difficulty level of all image transformations. The classifier does not need to be trained with these transformations; the transformations are meant to test the generalization capabilities of the classifier under realistic scenarios. In real-world scenarios AI

systems can always encounter novel situations their training did not adequately prepare them for. Especially in safety-critical environments such as autonomous cars, it is still expected that the classifier functions correctly, which is why this ability is evaluated.

The classification evaluation is motivated by the fact that in real-world scenarios, training and test datasets are never a full representation of all the situations the system might encounter during deployment. Since such an AI system can always encounter a novel situation, it has to be capable of generalizing to unknown examples. In cases where a novel example can not be identified correctly, the AI needs to be able to recognize the image as a difficult case and output a lower confidence score, so other components of the system can react accordingly. The common testing procedure of image classifiers does not reflect these requirements well. Usually, a classifier is evaluated using only a test set which contains a portion of training examples that was held back during training. These test examples serve the purpose of evaluating the AI, but since they come from the same distribution as the training examples, they cannot fully show the AI's capabilities in novel situations.

Image transformations in the classification evaluation

The following ten image transformations have been implemented and used as part of the classification evaluation. All transformations are defined in five strength levels and can be applied to the test dataset automatically. Examples of the effects can be seen in Figure 2 and 3. The parameters for the transformations are chosen arbitrarily by manual inspection of the resulting images. The parameters were chosen to achieve a minimal transformation at level 1 and a strong transformation that is still identifiable by humans at level 5. Remaining levels are interpolated between those two manually set values. Example images of all levels and transformations can be found in Figure 20 in the appendix.

Figure 2: Different strengths of transformations are defined in 5 levels of increasing difficulty.

Implemented transformations (see Figure 3) are:

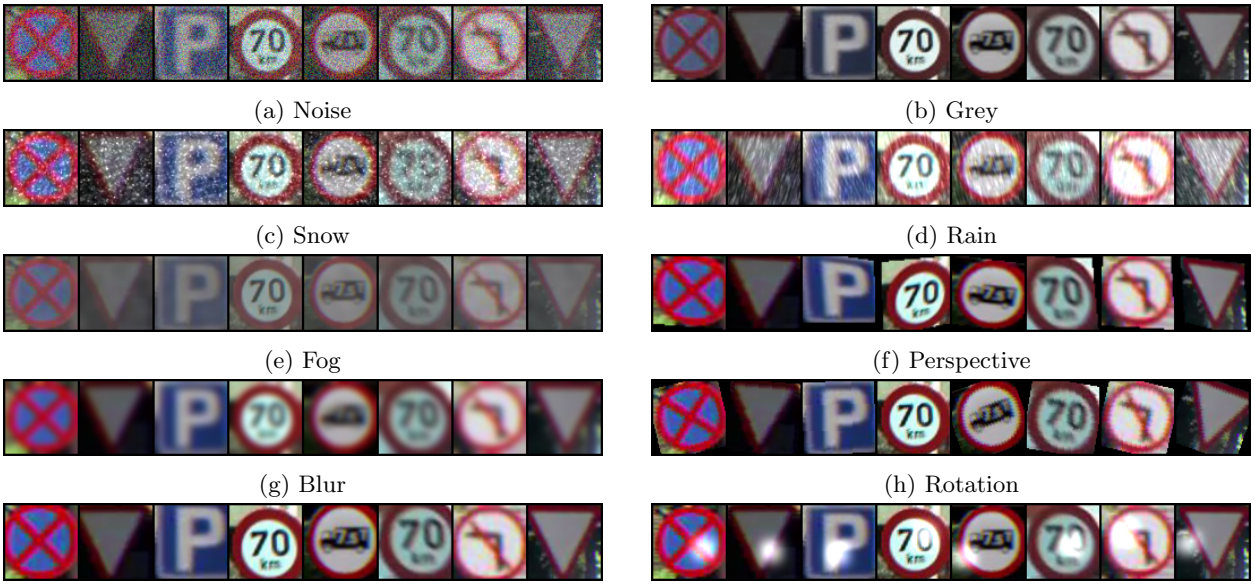
- Noise: Generates a layer of random (uniform) noise and blends the noise (n) with the original image (img). A variable factor (f) determines what proportion of noise is added to the image to create different levels of the effect.
$$(1 - f) * img + f * n$$
- Greyscale: A copy of the image is transformed to greyscale. Then the grey image ($grey$) is blended with the original image (img) depending on a factor (f) to achieve different levels of the effect.
$$(1 - f) * img + f * grey$$

- Snow: A snow-like effect is created by using a texture image that is added to the original image. The snow texture is created by hand and can be reused on any dataset. The texture image is randomly cropped, scaled and rotated to achieve a natural effect and variation. The snow texture is added to the original image (and clipped to prevent illegal values). Different levels of the effect are created by scaling the intensity of the added texture applying the effect multiple times on the same image. The input to this transformation is a list of numbers where each entry represents the intensity of adding the texture a single time.
- Rain: A rain effect is generated in an identical way to the snow texture using a different texture image.
- Fog: A fog effect is created using a random noise input. The noise is blurred using a Gaussian blur filter to achieve a fog-like cloud pattern using the random noise. The fog-like noise image is then scaled by a factor determining the strength of the effect and blended with the original image in the same way the noise effect was applied.
- Perspective: A random perspective transformation is applied to the image. A factor determines the strength of the transformation. The image is then slightly rescaled to fit the original image dimensions better. The input of this transformation are two numbers, the first specifying the strength of the crop effect and the second specifying the size of the scaling effect.
- Blur: A Gaussian blur effect is applied to the image. To achieve different levels of the effect, the size of the blur filter (in pixels) is adjustable.
- Rotation: A random rotation in a given range of degrees is applied to the image. Increasing the maximum angle of rotation creates different strength levels of the effect.
- Crop: A random crop of a few pixels is applied to the image. The cropped image is then scaled back to the original image size. This causes a number of pixels at the boundaries of the image to be lost. This is used to effectively zoom into the image a bit, changing the size and border location of the image. The input to this transformation is the size of the image after cropping (in pixels). Different sizes are used to create different effect strengths.
- Reflection: A reflection effect is applied by generating a white square that is blurred with a Gaussian filter to achieve a circular shape with soft and rounded edges. The square is added to the original image at a random position. The size of the square (in pixels) can be adjusted for different effect strengths.

Evaluation measures

The performance is assessed on three metrics. The first and most important is classification accuracy. The accuracy of the classifier is measured across the five different difficulty levels of each effect. Due to the increasing difficulty level of the effects, it is expected to observe a decline in the classification performance. The strength of this effect determines how well the classifier is able to generalize. The reason for the importance of this metric is that it expresses how well the classifier is able to perform its designated task. Without the ability to classify inputs correctly, the classifier loses its usefulness in the task it was designed to perform.

The other metrics are used to evaluate the classifier's certainty estimate based on the strength of the classifier's activation. The second metric is the Brier score. This score evaluates the calibration of the classifier in a multi-class scenario [47]. The Brier score is calculated by calculating the mean squared error (MSE) between the classifier's prediction and the ideal prediction of assigning a certainty of 1 to the correct class and a certainty of 0 to all other classes. Since the Brier score expresses the error the classifier makes, a lower score is better than a larger score. Guo et al. (2017) discover that neural networks are generally not well-calibrated [10]. The Brier score is used to evaluate how well the classifier's certainty estimates correlate to the probability of the prediction being correct. The intuition behind this measure is that a classifier predicting a class with 80% certainty should also be correct about 80% of the time [47].



(a) Noise (b) Grey

(c) Snow (d) Rain

(e) Fog (f) Perspective

(g) Blur (h) Rotation

(i) Crop (j) Reflection

Figure 3: Image transformations used in the classification evaluation. Only the level 3 transformation is shown.

Lastly, the AUROC score is used to measure the difference between certainties for correctly and incorrectly classified examples [11]. Similarly to calibration, this measure investigates the plausibility of the certainty values the classifier outputs. The difference between the two measures is that here, it is not relevant to achieve accurate certainty values. Instead, this measure looks at the separability of correctly and incorrectly classified examples. The measure is based on the maximum class certainty of the classifier. To calculate the AUORC score, all examples where the prediction is correct are assigned the positive class while incorrect predictions are assigned the negative class. Intuitively, this measure tells us the probability that a correctly classified example has a higher certainty/activation than an incorrectly classified example [11]. A binary classifier that, for example, always assigns a certainty of 0.51 to the correct class and 0.49 to the incorrect class will achieve a perfect accuracy and AUROC score but a low calibration score.

3.1.2 OOD evaluation

The out-of-distribution (OOD) evaluation in the pipeline tests the detection of OOD examples. The classifier has to detect, which images belong to the classifier's domain and which images are OOD. To test this, 23 OOD datasets have been implemented: Three datasets known from the literature [11, 4, 26, 25]; four additional novel datasets, ten datasets based on my image transformations from section 3.1.1 and six datasets generated by different GAN architectures.

The task of OOD detection is defined as a binary classification problem: Given two kinds of input images, in-distribution as the positive class and OOD examples as the negative class, the classifier has to separate them. The task is evaluated with the AUROC score that expresses how well the classifier's predictions on the fake and real examples can be used to separate the two sets [11].

This part of the pipeline is implemented in the following testing procedure:

1. The OOD datasets are selected or generated.
2. The in-distribution test set and the OOD datasets are shuffled together. The original classification labels are replaced by new binary labels: OOD examples as the negative class and in-distribution

examples as the positive class.

3. The classifier under test is used on the new dataset; this produces class probabilities for the original classification task. The classifier does not have to be changed or retrained to fit the binary OOD detection task.

The task is motivated by the fact that in real-world scenarios, image classifiers encounter novel situations regularly. The response to novel situations that are not part of the AI's design domain is of high importance in safety-critical applications. In domains such as automated driving, falsely recognizing unrelated objects as, for example, a stop sign, can cause serious safety problems. The problems of OOD detection and meaningful certainty estimates are highly related [25]. Since classifiers are often overconfident in their predictions, OOD inputs can lead to a highly confident prediction [25]. Intuitively, when shown an image that does not belong to any class, the classifier should be uncertain of its prediction. In a real-world scenario, this uncertainty could be utilized by alerting the human driver about the unrecognized situation.

One problem with current OOD evaluation approaches is that they are relatively easy to perform well on. Used test sets are always limited to a specific small subsection of the complete OOD space, running into the same problem of infinite testing as the classification evaluation. Since exhaustive evaluation is not an option, the used test sets should represent difficult edge cases instead of random selections. Additionally, the current and easy OOD datasets allow many different classifiers to achieve high scores and make it difficult to compare different architectures and measure improvements.

Examples of previously used OOD datasets are CIFAR, SVHN and LSUN by Hendrycks et al. (2016) [11], TinyImagenet and LSUN by DeVries et al. (2018) [4] and Liang et al. (2018) [26], or all of these by Lee et al. (2018) [24]. On these datasets, classifiers often perform very well, and results of over 99% AUROC are not uncommon. In the pipeline, it is therefore important to find more difficult OOD datasets to ensure a wider set of images that includes more difficult edge cases.

One way of creating more difficult OOD datasets is to find examples that are more similar to the in-distribution images while still being identifiable as OOD by humans. Defining the properties of these images is not entirely clear: How much does the image have to look like an in-distribution example to be considered in-distribution? This is difficult to measure on a technical level. The distinction between challenging in-domain and out-of-domain examples is unclear, but the required behaviour of the classifier is different. For challenging and perturbed in-distribution examples, the classifier is expected to correctly identify the traffic sign while for OOD examples, the classifier should output a uniform distribution. What images qualify as OOD and which images are in-distribution is therefore based on human judgement [20]. In real life this is a sliding scale - a traffic sign photographed during snow should still be recognized as a traffic sign. At some point, however, the snow will completely cover the traffic sign making it impossible to classify it correctly and therefore making it OOD. Similar problems exist in the challenging conditions of the CURE-TSD dataset [44]. Sometimes, the perturbations applied to the images are so disruptive that the image is no longer identifiable.

Evaluation measures

To evaluate the OOD task, the method proposed by Hendrycks et al. (2017) is used [11]. This means the classifier does not have to be adjusted to output OOD predictions, and the original softmax output is sufficient. The classifier is evaluated using the binary classification task where the positive class are in-distribution, and the negative class are out-of-distribution examples. This measure is only obtained indirectly by using the predictions of the original classifier. Using this indirect measure of OOD classification allows testing a wide range of different classifier architectures without needing to adapt them specifically. If an architecture already provides specific certainty or out-of-distribution estimates separate from the softmax prediction that value can be used instead.

The main evaluation measure is, analogous to Hendrycks et al. (2017), the AUROC score[11]. In addition, Hendrycks et al. use several additional scores that are not considered relevant for this pipeline to avoid giving too many confusing measures. The motivation of focusing on the AUROC score is that it offers an intuitive explanation of the measure: The probability with which a real example has a higher activation than an OOD example [25].

OOD datasets

The goal of creating OOD datasets is to offer a more nuanced evaluation of OOD detection than the datasets used in the literature. 23 OOD datasets have been implemented which are roughly split into three categories:

1. Standard OOD datasets: OOD datasets that are more difficult by resembling in-distribution examples more closely.
2. OOD image transformations: OOD datasets that are more difficult by transforming both the OOD and the in-distribution images to be more similar.
3. GAN OOD example generation: OOD datasets generated by a generative adversarial network trained to specifically create examples that work well as OOD.

Standard OOD datasets

This category contains three OOD datasets (see Figure 4) I implemented based on existing literature. Similar to Hendrycks et al. (2017), two versions of randomly generated noise (using either a normal or uniform distribution) are used as OOD datasets. As done before, the public TinyImagenet dataset [23] is used as a pre-existing OOD dataset [4, 24]. These previously used measures are chosen to provide a point of comparison and a difficulty baseline.

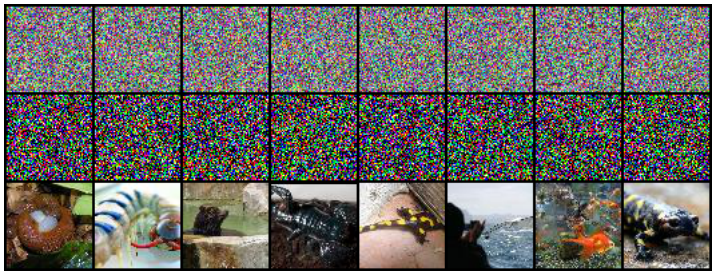


Figure 4: OOD datasets used in the literature. Each row shows examples from one OOD dataset. From top to bottom: Uniform noise, normal noise, Tiny ImageNet.

To find more difficult OOD test sets, I experimented with different ways of distorting the in-distribution images strongly enough to make them no longer recognizable. These transformations are distinct to the previously used transformations in the classification evaluation aspect of the pipeline since these transformations had a different goal: In the classification evaluation, transformations are meant to make the classification more difficult while still clearly representing the original class. In this part of the pipeline, the transformations are meant to transform in-distribution images to OOD images while still keeping some of the visual properties of the original image. I am not aware of any existing work using this approach to create OOD examples. The implemented transformations (visualized in Figure 5) are:

- Shuffled images: The OOD examples are created by taking an original test example and randomizing the order of its pixels. This effectively creates an image that looks like random noise but uses the same color profile as the original image. In cases where the classifier bases its decision heavily on the images color values, this should intuitively make the OOD detection task harder. Since the images look like noise, they clearly qualify as OOD examples.

- Mixed images: An OOD example is created by combining two in-distribution of different classes. Two random in-distribution examples are selected while ensuring the class of the second image does not match the first one. Then, both images are vertically cut in half, and two halves from different source images are combined to a new OOD image. These images can be considered as OOD because they clearly show identifying features of two different classes, the combined image does, however, not exist.
- Swirled images: An image effect that applies a circular rotation in the center of the image. This obscures the original image heavily, making it unrecognizable. Since the distribution colors and their general positions stay similar to the original image, this provides a good candidate for an effective OOD dataset.
- Color swap: Swapping the color channels of an RGB image creates a new image with the same edges, but different colors. Whether this can be considered an OOD example is domain-dependent. In the case of traffic signs, all class instances are colored in the same way (stop signs are always red). Swapping these colors (for example by creating a blue stop sign) therefore qualifies as an OOD image.

Figure 5: Novel OOD datasets used in the evaluation. Each row shows examples from one OOD dataset. From top to bottom: Shuffled images, mixed images, swirled images and color shift.

OOD image transformations

The OOD datasets in the previous section followed the intuition to create more challenging OOD datasets by creating images that resemble in-distribution data more closely. Conversely, it should also be possible to make the in-distribution images look less like in-distribution images to increase the difficulty of the OOD detection.

To explore this idea, I combined the transformations used in the classification evaluation with the OOD evaluation. Using the TinyImagenet [23] dataset as OOD data, previous image transformations (such as Fog, Rain and Reflection) are applied to both the in-distribution and OOD examples, resulting in 10 new test datasets. Figure 6 shows this at the example of my Reflection transformation. More detailed examples for all image transformations can be found in the appendix in Figure 21. This use of data augmentation techniques in the context of OOD evaluation is a novel contribution of this research.

Figure 6: Example use of data transformations to create more difficult OOD datasets. In this case, a reflection effect is applied to both the in-distribution and OOD images, making the detection of OOD examples more difficult.

GAN OOD example generation

Generative models such as GANs can be utilized in the context of data augmentation to generate novel examples [34]. In this work, the generated examples will be used as OOD examples that offer an additional test case to evaluate the classifier on the task of OOD detection.

To utilize a GAN for OOD example generation, the architecture proposed by Lee et al. (2017) [25] is used. Using this architecture allows generating novel OOD examples that have some visual similarity to in-distribution examples. The same architecture is also used to create OOD examples that have no visual similarity to in-distribution examples but cause a high network activation [41]. Both networks have been implemented from scratch using the CURE-TSD dataset [44].

In addition to the original GAN loss function, the improved Wasserstein loss function [2] has been implemented. Using the two different architectures with both loss functions gives a total of four different GAN configurations. To allow the GAN networks to generate examples of specified classes, the networks have been modified to allow conditional example generation [30].

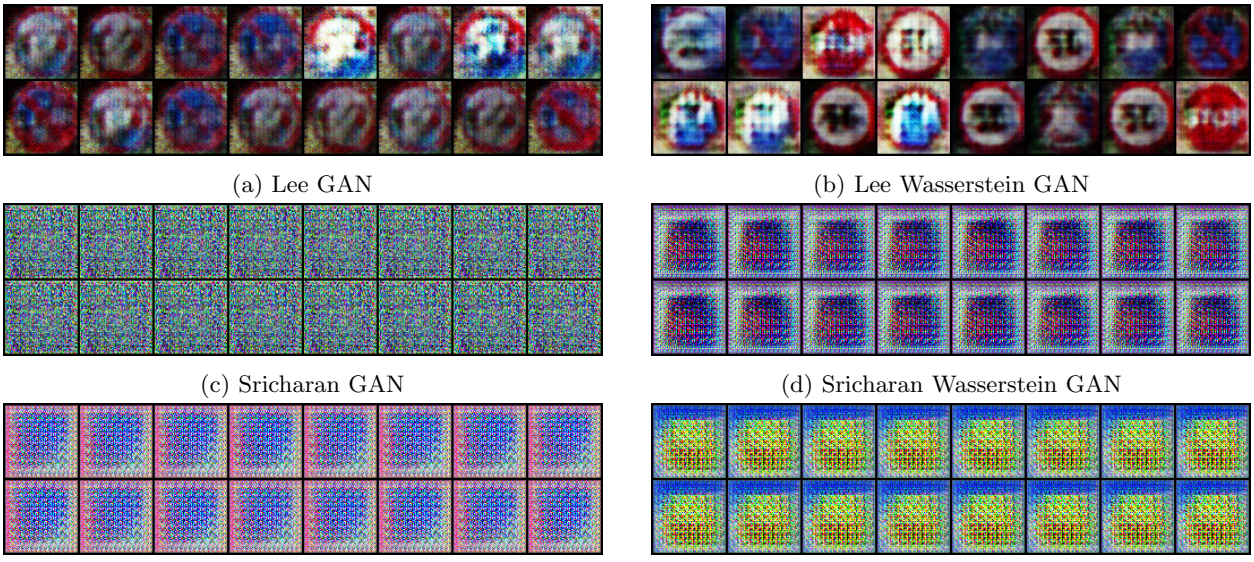
In the original use-case by Lee et al. (2017) [25], the GAN network was used to train a robust classifier. In this case, however, the GAN should not interact with the classifier under test directly since the whole pipeline process is meant to evaluate a pre-defined classifier whose weights are not necessarily known (this is referred to as a black-box setting earlier). To circumvent this issue, the GAN architecture is trained on a surrogate classifier which is created solely for the purpose of training the GAN network. This classifier does not share weights with the classifier under test and does not even have to follow the same network architecture. To highlight this, the GAN classifier is implemented without fully connected layers which are used in the two classifier architectures. My hypothesis for this procedure was that the generated images using the surrogate classifier should still be able to produce a response in the classifier under test, similar to black-box adversarial attacks [31].

To use the GAN network by Lee et al. (2017) [25] to generate OOD data, it is necessary to stop the training procedure at the right time. When the training is stopped too early, the resulting images do not show much resemblance to actual in-distribution images and therefore do not provide a challenging OOD test set. When the training is carried out for too long, the images start resembling in-distribution images and cannot be used as an OOD example anymore. This is visualized in Figure 7, which shows GAN examples at different stages of the network's training process.
To utilize the GAN for OOD example generation, I add a new stopping criterion to the training of the GAN network. Since I modified the GAN to generate images of specific classes, it is possible to calculate the accuracy of the classifier on the generated images. Taking the accuracy of a classifier on the generated images allows to judge their visual quality automatically: If the classifier predicts similarly well to random

guessing, the images look too dissimilar to the original images and if the classifier has a high accuracy the generated images are too similar. This allows defining a threshold between those two extreme points where the GAN training is stopped. For the CURE-TSD dataset, this threshold has been set to 30% accuracy by manually inspecting the resulting image quality. Only the best performing GAN (under the threshold) is then used in the pipeline.

Figure 7: Visualization of the GAN training progress over 30 epochs. Starting in the top left corner (and moving along the row), each image represents one training epoch. The threshold of 30% accuracy is reached after 14 epochs (highlighted in red).

A different stopping criterion is needed to train the modified architecture proposed by Sricharan et al. (2018) [41]. Since the generated examples are not visually similar to the original images, calculating the accuracy on the generated images does not work in this case. Instead, the classifier loss from the GAN architecture is used to determine when to stop training. In initial experiments, this value quickly converges to 0 and gives a good indication in the strength of the generated images. The threshold is, based on initial tests, set to 0.005. For the architecture using the Wasserstein loss function, two additional configurations were trained since this architecture showed the most promising results. The additional stopping criteria for these configurations are a classifier loss of 0.002 and a set training period of two epochs. In general, all measures used for this GAN architecture were chosen by experimentation and will probably not work well on different datasets. Example images for all GAN configurations can be found in Figure 8.



(a) Lee GAN

(b) Lee Wasserstein GAN

(c) Sricharan GAN

(d) Sricharan Wasserstein GAN

(e) Sricharan GAN (0.002)

(f) Sricharan GAN (2 epochs)

Figure 8: GAN generated OOD examples for different GAN architectures and configurations.

3.2 Experimental setup

The pipeline as defined in section 3.1 is tested on two classifier architectures: the baseline classifier and MC Dropout. Applying the pipeline leads to two main experiments that are being evaluated. First, the classification evaluation, in which the classifiers are tested on difficult in-distribution examples. These test cases are created using the data transformations introduced in section 3.1.1. The exact parameters of the transformations are chosen by manual inspection of the images and are given in Table 12 in the appendix. In total, 10 transformations are evaluated over 5 different levels, leading to 50 different test cases. Secondly, the OOD evaluation, in which the classifiers have to detect OOD examples. The OOD evaluation consists of three separate tests which are described in detail in section 3.1.2. These tests use a total of 23 test sets: three OOD test cases from the literature, four OOD test cases created by perturbing in-distribution images, ten OOD test cases created using image transformations, and six OOD test cases created using GANs. OOD test datasets are always created using equal proportions of in-distribution and OOD examples.

Since the evaluation introduces some randomness due to the initial weights of the classifier networks as well as the applied data transformations, the performance is evaluated over multiple runs. Each evaluation is repeated 20 times, averaging the results and reporting the mean and standard deviation. In each repeated evaluation the classifier is retrained and the tested transformation is generated again. To ensure the reproducibility of the results, the randomness has been fixed with a seed of 0 in the experiments.

The used dataset is CURE-TSD created by Temel et al. (2017) [44]. The original dataset contains a number of videos with 14 different classes of traffic signs. As a pre-processing step, the traffic signs are extracted from the existing video files and stored as 64x64 images. For this thesis, only the unmodified videos in the dataset are used, resulting in a total of 8960 images of traffic signs.

The classifiers follow a relatively simple convolutional structure: Two convolutional layers with 32 filters each, followed by a pooling layer and two fully connected layers. For the MC Dropout classifier, an additional dropout layer (p=0.6) is added after each hidden network layer. To train the classifiers, the dataset is split into 80% training data and 20% testing data. All later evaluation steps such as the image transformations or OOD tests are based on the test data the classifier has not seen before. To train the classifiers, the Adam optimizer [18] (lr=0.0002) and a cross entropy loss function is used. The baseline classifier is trained for 4 epochs while the MC Dropout classifier is trained for 8 epochs (batch size = 128). This training setup results in a similar test accuracy for both networks since the MC Dropout classifier learns slower due to its dropout layers.

The implemented GANs consist of a generator network, a discriminator network and a classifier network. The generator and discriminator networks consist of 5 convolutional layers with contrasting filter sizes: [64, 128, 256, 512, 1] for the discriminator and [512, 256, 128, 64, 3] for the generator. Both networks use an additional embedding layer to encode class information into the input of discriminator and generator. The classifier used in the joint GAN training process uses 4 convolutional layers of filter sizes [64, 128, 256, 14], where the 14 output neurons represents the classes of the CURE-TSD dataset. All networks used for the GAN make use of batch normalization [14] to accelerate and stabilize the training. The GANs are trained using the Adam optimizer [18] with a learning rate of 0.0002 and a batch size of 96.

All experiments are implemented in Python using the PyTorch deep learning framework [32]. All machine learning models are implemented and trained from scratch, no pre-trained models are used.

4 Results

4.1 Classification evaluation

To evaluate the classification evaluation under the different image transformations, I will first look at the raw performance of the classifiers without any applied transformations in comparison to the average results of the pipeline evaluation (across all transformations and levels). This is done to show the main effect of using the pipeline evaluation in comparison to the initial test set. Table 1 summarizes the average scores for accuracy, AUROC, and Brier for the Baseline model and the MC dropout both when tested on the initial test dataset without transformation (Table 1a) and with the evaluation pipeline (Table 1b). Results of the pipeline evaluation are averaged over all transformations and effect levels. Visualizations of the results can be found in Figure 9.

Since my main goal is evaluating the pipeline itself – and not evaluation of individual classifier performance – the focus of my analysis is on relative effects and trends, and not on absolute performance scores. To this end, I analyze the results with ANOVAs and post hoc tests to determine the relative effects.

Classifier	Baseline	MC Dropout
Accuracy	0.9899 (0.0030)	0.9866 (0.0019)
AUROC	0.9874 (0.0070)	0.9917 (0.0022)
Brier	0.0011 (0.0003)	0.0026 (0.0003)

(a) No transformations

	Baseline	MC Dropout
Accuracy	0.9132 (0.1108)	0.9302 (0.0978)
AUROC	0.9342 (0.0621)	0.9517 (0.0567)
Brier	0.0095 (0.0112)	0.0110 (0.0099)

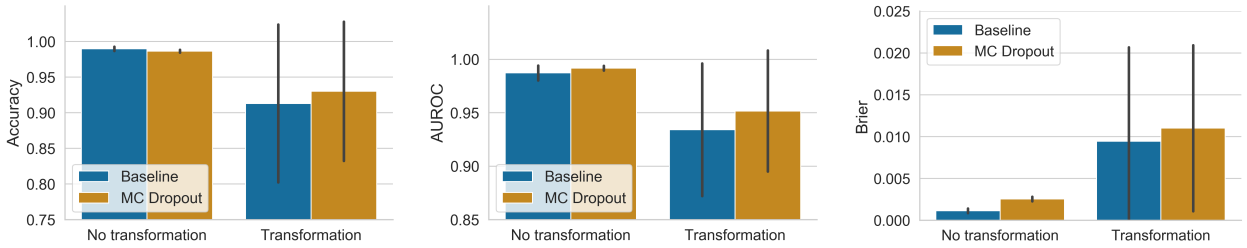
(b) Evaluation pipeline

Table 1: Results for all three evaluation measures are shown (Accuracy, AUROC and Brier). On the left Table are the base measures of baseline and MC Dropout classifiers without applying any transformation on the test set (n=2000). On the right are results for both classifier when using the evaluation pipeline and averaging over all transformations and levels (n=2000). This number is chosen since I sampled the transformations 20 times for each combination of classifier (2 levels), transformation (10 levels) and level (5 levels), resulting in a total of 2000 samples. The evaluation without transformation was run 2000 times to match that number. Results for accuracy and AUROC are given as a fraction, and all results are rounded to the fourth decimal. Standard deviations are reported in brackets.

I analysed the effects of using the evaluation pipeline compared to the original test set where no transformations were applied in a 2 (Classifier: Baseline or MC Dropout) x 2 (Transformation applied: Yes or No) ANOVA. The analysis is repeated three times for the different dependent variables: accuracy, AUROC, and Brier score. For all three measures – accuracy ($F(1,3996)=8.53$, $p<.01$), AUROC ($F(1,3996)=67.44$, $p<.001$), and Brier ($F(1,3996)=39.36$, $p<.001$) – there is a significant main effect of the used classifier. On accuracy and AUROC, MC Dropout performs better compared to the Baseline. On the Brier score, the Baseline performs better. Applying the transformations (which means using the pipeline instead of using the original test set) also has a significant effect for all three measures: accuracy ($F(1,3996)=812$, $p<.001$), AUROC ($F(1,3996)=1219.65$, $p<.001$), and Brier ($F(1,3996)=1250.69$, $p<.001$). Analysing the performance by applying the image transformations leads to significantly worse performance of the classifiers across all measures than using the original test set and not applying the transformations. There is an interaction effect between the classifier and transformation for accuracy ($F(1,3996)=18.84$, $p<.001$) and AUROC ($F(1,3996)=24.38$, $p<.001$) but not for Brier ($F(1,3996)=0.1$, $p=.75$).

Looking at the classifier and whether the pipeline has been applied in combination shows the true power of the pipeline. A post hoc comparison between the two factors has been applied to analyse the different effects on both classifiers. The results show that the differences between the two classifiers are significant with $p<.001$ for accuracy and AUROC when using the image transformations. When the original test set is used, the differences between the two classifiers are insignificant with $p=.75$ for accuracy and $p=.1$ for AUROC.

This means that both classifiers perform very similar on the original test set; none of the classifiers had a clear advantage over the other. Only if the pipeline transformations are used, MC Dropout significantly outperforms the baseline classifier. For the Brier score, there is no gained knowledge in terms of comparing the classifiers. Instead, the Baseline classifier consistently outperforms MC Dropout for both the initial test set and the pipeline evaluation.



(a) Accuracy (b) AUROC (c) Brier

Figure 9: Comparison between pipeline evaluation and the initial test set of untransformed images. Results are split between the two classifiers: Baseline (blue) and MC Dropout (orange). The error bars indicate the standard deviation. The large size of the error bars when applying transformations is caused by averaging results over all transformations and levels, which are very different effects.

Detailed results looking at the effects of image transformations and different levels are split into three sections for the three different measures: Accuracy in section 4.1.1, AUROC in section 4.1.2, and the Brier score in section 4.1.3. In each of the sections, the results have been analyzed in a single 3-way ANOVA with the classifier, the transformation and the strength of the transformation as the independent variables. This results in a 2 (Factor: Classifiers) x 10 (Factor: Transformations) x 5 (Factor: Levels) ANOVA. In the following sections, I will investigate the general effects of the main factors, individual results and more detailed plots can be found in the appendix.

4.1.1 Accuracy

The results of the ANOVA are given in Table 2 and show that all variables and interactions have significant effects. The individual effects of the transformations, levels and interactions with the classifier are analyzed in post hoc multiple comparisons.

In the appendix, Table 13 shows the results of the experiment split into individual groups by classifier, transformation and level of the transformation. A visualization of these results can be found in Figure 22 in the appendix as well.

Factor	d.f.	F	p
Classifier	1	144.83	<.001
Transformation	9	726.64	<.001
Level	4	1968.66	<.001
Classifier*Transformation	9	27.46	<.001
Classifier*Level	4	25.92	<.001
Transformation*Level	36	145.45	<.001
Error	1936		

Table 2: Statistical analysis of the accuracies given the used classifier, transformation and level.

Comparing the results for the five different levels of transformations, the consistent pattern is that accuracy is lower when the image transformation level increases. Each level of strength produces significantly different results to all other levels ($p<.001$ for all individual comparisons).

Figure 10 shows the main effect of the transformations. For each transformation, the mean score is shown, with error bars that depict comparison intervals. The dotted lines in Figure 10 show a boundary between two conditions that differ significantly from one another. In that way, 6 groups can be identified. The models struggle the most with manipulations of Perspective, followed by Reflection, Fog, and Crop. For all other metrics, accuracy is relatively high (above 95%).

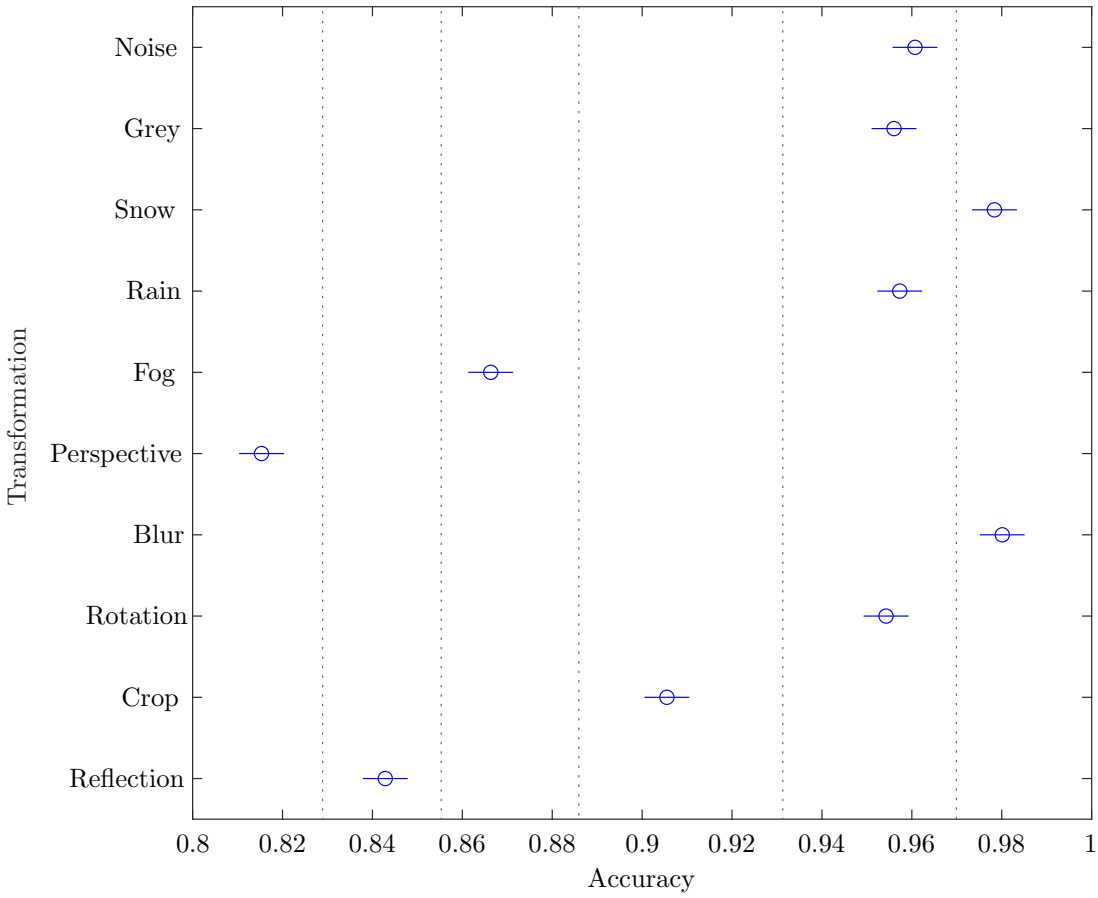


Figure 10: Multiple comparison post hoc analysis of the effects of different transformations. Vertically overlapping error bars for different transformations indicate non-significant differences. Vertical separations between the non-overlapping and significantly different groups have been added.

Figure 11 shows the performance of the two classifiers, Baseline (blue) and MC Dropout (orange) for the different strength levels with which the transformations have been applied. For the first level (and the initial result on the original test set), the baseline classifier achieves a higher score than MC Dropout. This effect is reversed when looking at the other transformation levels (2 to 5), where MC Dropout has a higher accuracy than the baseline classifier. In post hoc statistical analyses, the effect of the interaction becomes clear: Comparing the performance of the two classifiers for the different levels, there is no significant difference

between Baseline and MC Dropout for Level 1 and 2. The classifiers differ significantly for the levels 3 ($p<.001$), 4 ($p<.001$) and 5 ($p<.001$), where MC Dropout outperforms the Baseline. This shows that there are differences in the two classifier architectures, that are only visible when evaluating at higher levels of transformations.

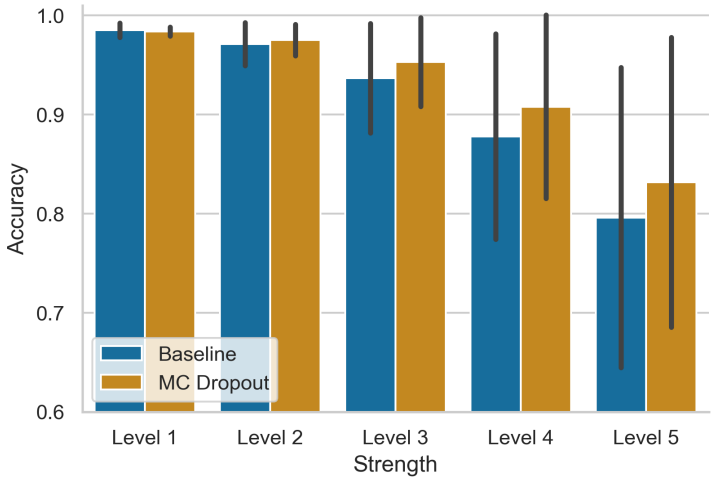


Figure 11: This plot shows the main effect of the different strength levels of the transformations. Within each level, the different transformations are averaged. The hue of the bars represent the two classifiers: Baseline (blue) and MC Dropout (orange). The error bars show the standard deviation. The standard deviations are quite high, especially in higher levels, because the results are averaged over all transformations which have different effects.

The joint analysis of classifier and transformations is shown in Figure 12. Not all transformations show significant differences between the two classifiers when averaging over the different strength levels of the transformations. Non-significant differences between the Baseline and MC Dropout occur for the following transformations: Snow, Perspective, Blur, Rotation and Crop. In all remaining transformations, MC Dropout significantly outperforms the Baseline: Noise ($p<.001$), Grey ($p<.05$), Rain ($p<.001$), Fog ($p<.001$), Reflection ($p<.001$).

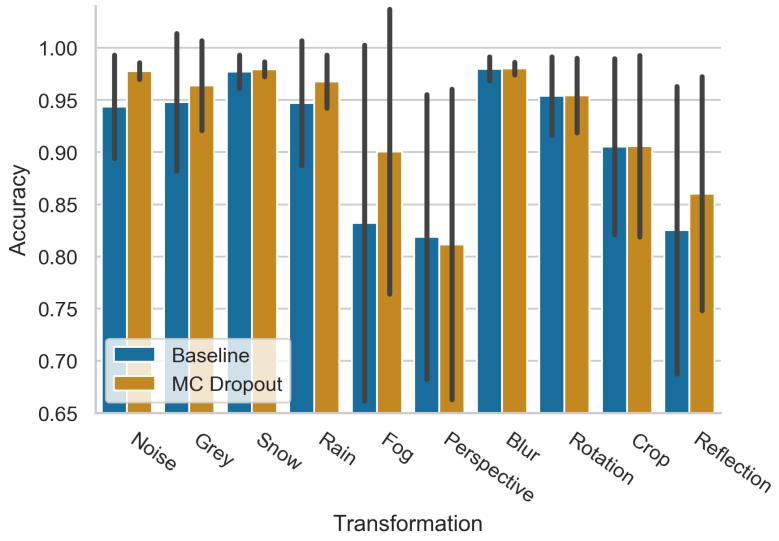


Figure 12: This plot shows the main effect of the different transformations. Within transformations, results for the 5 levels are averaged. The hue of the bars represent the two classifiers: Baseline (blue) and MC Dropout (orange). The error bars show the standard deviation. The standard deviations are quite high, especially in higher levels, because the results are averaged over all levels which have quite different effects.

The analysis demonstrates that performance assessments of classifiers require testing on a range of different transformation with different levels of strength. For example, while for low levels of transformations both classifiers perform similarly, the higher robustness of MC Dropout becomes only evident when looking at effects of strength 4 or 5. Similarly, only looking at a subset of transformations is insufficient because the classifiers respond differently to different transformations. For example, both classifiers respond similarly to Rotation, but show significant differences on the Fog transformations. There are subtle differences between the classifiers that are not detectable by only looking at the original test set or even individual transformations.

The statistical analysis shows an additional interaction effect between the transformation type (e.g. Snow, Fog) and the level of transformation. This is not surprising since the different levels of transformation are implemented based on the way the transformation works, and it is not guaranteed that the levels between different transformations scale identically. Therefore, I will not look into this interaction effect in detail.

Given the detailed result plot (Figure 13 in the appendix), an additional interaction effect between all three factors is likely to occur. For example, in the Snow transform, the Baseline performs better than MC Dropout at levels 1-3 while MC Dropout performs better at levels 4 and 5. Looking at this interaction in detail might reveal additional significant differences between the classifiers given the specific transformation and level. However, due to the large number of total factors considered (2*10*5), this level of analysis is too detailed and will not be discussed.

4.1.2 AUROC

The AUROC score shows how separable correctly and incorrectly classified examples are. This correlates to the quality of the classifier's certainties in the following sense: We would expect a classifier that outputs accurate certainties for its prediction to be more confident of correctly predicted examples than of incorrectly predicted examples.

In the same manner as the results for accuracy are reported, results of the ANOVA can be found in Table

3. Detailed results split by classifier, transformation, and level can be found in Table 15 and Figure 23 in the appendix.

Factor	d.f.	F	p
Classifier	1	306.79	<.001
Transformation	9	527.40	<.001
Level	4	1229.60	<.001
Classifier*Transformation	9	5.85	<.001
Classifier*Level	4	8.05	<.001
Transformation*Level	36	66.30	<.001
Error	1936		

Table 3: Statistical analysis of the AUROC with the factors classifier, transformation and level.

The analysis shows significant effects on all factors and interactions. Summarizing the main effects for the AUROC, MC Dropout outperforms the Baseline, higher levels lead to a lower AUROC, and different transformations cause different results. These results are in line with the previous results for accuracy and show that my results are consistent across measures.

The effect of the different levels on both classifiers is visualized in Figure 13. Across all levels, MC Dropout performs better than the Baseline classifier. Comparing the levels, all five levels differ significantly from all other levels with p<.001, with the effect that higher levels cause lower AUROC scores. When comparing the classifiers over the different levels, both classifiers differ significantly on all five levels with p<.05 for level 1 and p<.001 for levels 2-5. Differences between the classifier are therefore detectable in all levels, but they become larger at higher levels.

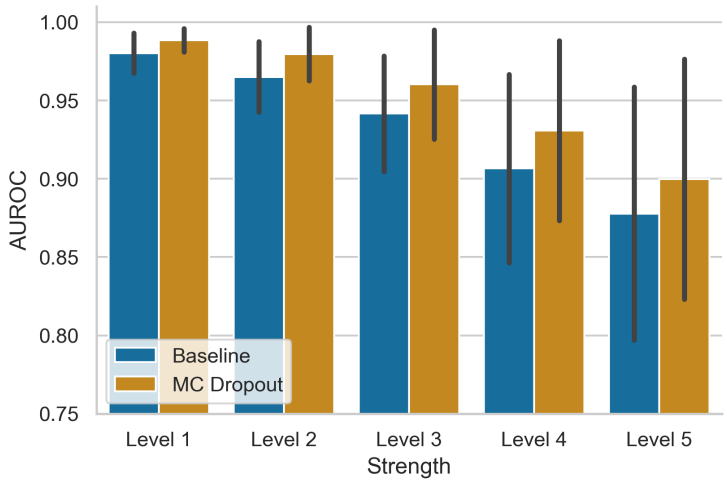


Figure 13: This plot shows the main effect of the different strength levels of the transformations on the AUROC. Within each level, the different transformations are averaged. The hue of the bars represents the two classifiers: Baseline (blue) and MC Dropout (orange). The error bars show the standard deviation. The standard deviations are quite high, especially in higher levels, because the results are averaged over all transformations which have quite different effects.

AUROC results for different image transformations are given in Figure 14. A Figure showing significant effects between the different levels analogue to Figure 10 for accuracy can be found in the appendix (Figure 24). The four transformations that differ significantly from all other transformations and cause the lowest

AUROC values are: Perspective, Reflection, Crop, and Fog. Comparing the two classifiers based on the transformations shows significantly ($p<.001$) different performance on the following transformations: Noise, Grey, Perspective, Crop, and Reflection. In these transformations, MC Dropout significantly outperforms the Baseline classifier. MC Dropout still performs better in the remaining transformations Rotation, Blur and Fog, but these differences were non-significant.

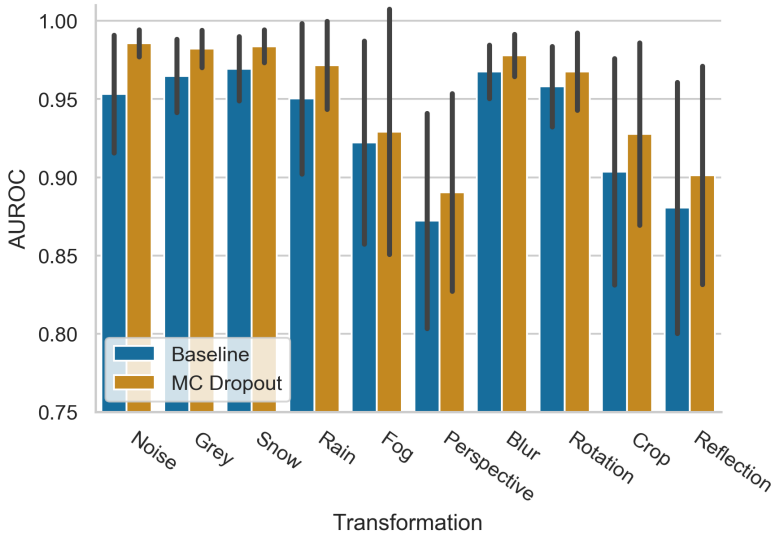


Figure 14: This plot shows the main effect of the different transformations on the AUROC. Within each transformations, results for each level are averaged. The hue of the bars represent the two classifiers: Baseline (blue) and MC Dropout (orange). The error bars show the standard deviation. The standard deviations are quite high, especially in higher levels, because the results are averaged over all levels which have quite different effects.

4.1.3 Brier

The Brier score, similarly to the AUROC, evaluates the quality of the classifier's certainties. Instead of comparing activations of correct and incorrect examples, the calibration is evaluated. The calibration describes how well the predicted certainties correlate to the probability of predicting correctly. Since the Brier score is a loss value, a low score means that the classifier is well-calibrated while a high score means that the classifier is not well calibrated. The resulting score does not offer an intuitive interpretation and therefore relies on comparing scores of different classifiers, baselines, or predefined thresholds.

Analogous to accuracy and AUROC, the results of the ANOVA can be found in Table 4. Detailed results split by classifier, transformation, and level can be found in Table 17 and Figure 25 in the appendix.

Factor	d.f.	F	p
Classifier	1	189.00	<.001
Transformation	9	1253.16	<.001
Level	4	3592.35	<.001
Classifier*Transformation	9	38.53	<.001
Classifier*Level	4	7.61	<.001
Transformation*Level	36	188.81	<.001
Error	1936		

Table 4: Statistical analysis of the Brier scores with the factors classifier, transformation and level.

For the Brier score, as for the other measures, the analysis shows significant effects on all factors and interactions. The main result for the Brier score is that (conversely to the previous results) the Baseline classifier performs better than MC Dropout. In line with previous observations, higher levels lead to worse (higher) Brier scores and different transformations cause different results.

Comparing results of different levels (Figure 15), higher levels are more difficult for both classifiers and therefore cause higher Brier scores. The difference between the levels is significant with p<.001 for all five levels. Comparing the classifiers on the different levels, an inverse observation to the results on the other two measures can be made. On level 1 to 4, the Baseline classifier performs significantly better than MC Dropout (with p<.001 for all four levels). At level 5, however, this difference is no longer significant.

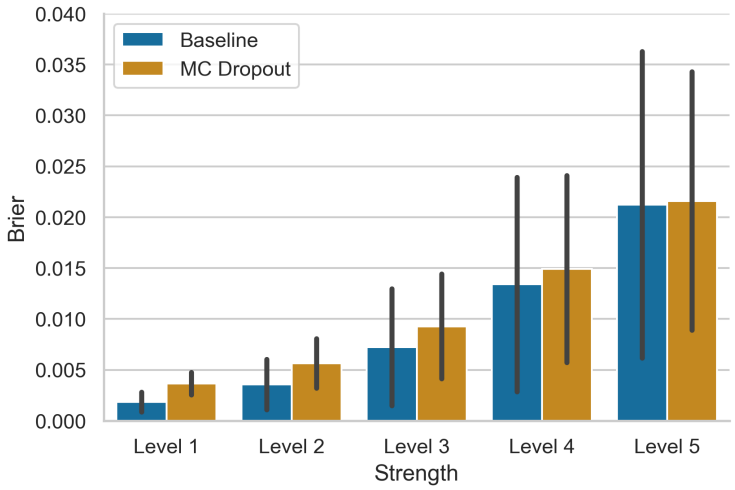


Figure 15: This plot shows the main effect of the different strength levels of the transformations on the Brier score. Within each level, the different transformations are averaged. The hue of the bars represents the two classifiers: Baseline (blue) and MC Dropout (orange). The error bars show the standard deviation. The standard deviations are quite high, especially in higher levels, because the results are averaged over all transformations which have quite different effects.

Comparing Brier scores on the different transformations (see Figure 16), the scores are varied, showing that different transformations have different effects on the Brier score (similar to previous results). Results of the statistical comparison are plotted in Figure 26 in the appendix. The most difficult transformations are again Fog, Perspective, Crop, and Reflection, showing a consistent effect across all three measures. Comparing the two classifiers in combination with the transformation, the Baseline classifier outperforms MC Dropout with p<.001 on the following measures: Noise, Snow, Rain, Fog, Blur, Rotation, and Crop. On the Perspective transformation, the Baseline classifier outperforms MC Dropout with p<.05. For the grey transformation,

there is no significant effect. For the Reflection transformation, MC Dropout outperforms the Baseline classifier ($p<.001$).

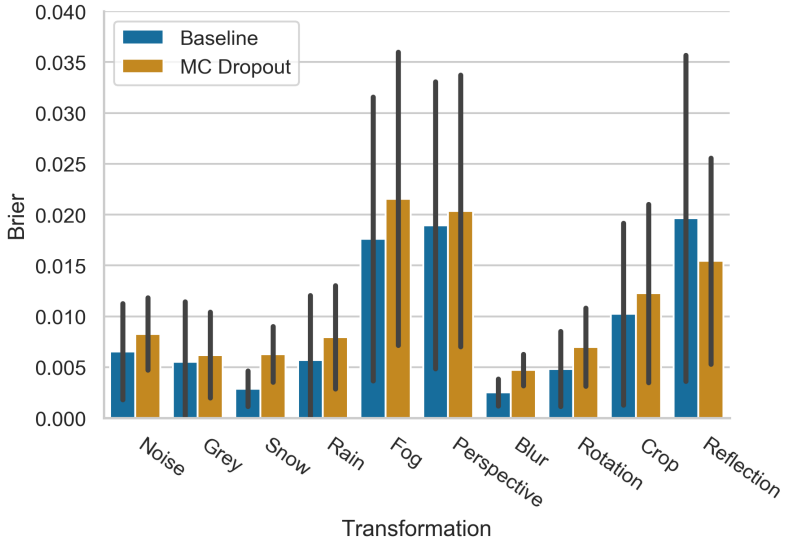


Figure 16: This plot shows the main effect of the different transformations on the Brier score. Within each transformations, results for each level are averaged. The hue of the bars represents the two classifiers: Baseline (blue) and MC Dropout (orange). The error bars show the standard deviation. The standard deviations are quite high, especially in higher levels, because the results are averaged over all levels which have quite different effects.

4.2 OOD evaluation

The OOD evaluation tests, how well in-distribution images can be separated from out-of-distribution (OOD) images. This is evaluated using the AUROC, following [11]. I will split my results into three sections:

1. Standard OOD datasets: The datasets from the literature and novel OOD datasets.
2. OOD data transformations: OOD datasets generated by the image transformations.
3. OOD GAN examples: OOD datasets generated by GAN examples.

4.2.1 Standard OOD datasets

The standard OOD datasets consist of three datasets used in the literature and four novel datasets. I will compare these based on the performance of both classifiers: Baseline and MC Dropout. Results are given in Table 5 and visualized in Figure 17.

Using the datasets from the literature (top 3 rows in Table 5), both classifiers perform far better than 90% AUROC. The noise datasets are very easy to distinguish from the real images, and both classifiers can do so with over 98% accuracy. There is no significant difference between the two classifiers for the noise datasets. For the Tiny ImageNet dataset, the classifiers perform worse, and the difference between them is significant. However, the classifiers still perform very well on the Tiny ImageNet dataset. The baseline classifier is able to achieve an AUROC of 94.9% without being specialized or trained in OOD detection at all. The difference between the classifiers is not very large either (less than 4%).

	Baseline mean	MC Dropout mean
Uniform noise	0.9862 (0.0208)	0.9880 (0.0053)
Normal noise	0.9846 (0.0115)	0.9906 (0.0045)
Tiny ImageNet	0.9490 (0.0128)	0.9821 (0.0036)
Shuffled images	0.9697 (0.0213)	0.9893 (0.0040)
Mixed images	0.8985 (0.0187)	0.9252 (0.0067)
Swirled images	0.8283 (0.0337)	0.9359 (0.0128)
Color swap	0.8487 (0.0745)	0.8946 (0.0391)

Table 5: AUROC values on the standard OOD datasets. Standard deviations are given in brackets. Values are given as a fraction and rounded to the fourth decimal.

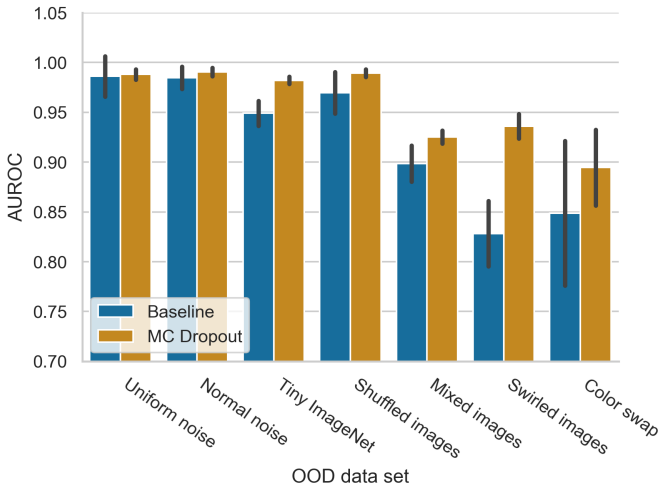


Figure 17: AUROC scores of both classifiers on the standard OOD datasets. The y-axis shows the AUROC score as a fraction, the x-axis the used OOD data set. Classifiers are indicated by color: Baseline (blue) and MC Dropout(orange). Error bars show the standard deviation.

To analyse the statistical effects of the results in Table 5, a 2 (Factor: Classifier) x 7 (Factor: OOD dataset) ANOVA has been applied. Results of the ANOVA can be found in Table 6. Significant effects are present for both factors and the interaction. This indicates that, in general, MC Dropout performs significantly better on these test sets. The choice of OOD test set and the interaction between OOD test set and classifier also have significant effects which will be analysed in post hoc analysis.

Factor	d.f.	F	p
Classifier	1	115.55	<.001
OOD test set	6	144.37	<.001
Classifier * OOD test set	6	17.78	<.001
Error	266		

Table 6: Statistical analysis of the standard OOD datasets with the factors classifier and OOD test set.

Comparing these OOD datasets to the ones created by me (lower four rows in Table 5), the difficulty of the tests has clearly increased. Three of the four added OOD datasets are significantly harder to identify

than the Tiny ImageNet dataset (third row in Table 5), which is the hardest of the datasets taken from the literature. Significantly more difficult ($p<.001$) are: Mixed images, Swirled images, and Color swap. The OOD dataset with shuffled images is easier than Tiny ImageNet, but this difference is non-significant.

Comparing the classifiers on the different test sets shows significant differences for Tiny ImageNet ($p<.01$), Swirled images ($p<.001$), and Color swap ($p<.001$). Color swap and Swirled images improve upon Tiny ImageNet in the sense that the difference between the two classifiers is larger and both classifiers perform worse on them. The other four OOD datasets did not produce significantly different responses in the classifiers. This indicates that my own OOD examples not only provide more challenging test cases, they also show a clearer distinction in the performance of the two compared classifiers.

4.2.2 OOD data transformations

The OOD data transformations use the ten transformations from the classification evaluation to produce ten new OOD test sets. The transformations have been applied to Tiny ImageNet and the in-distribution traffic sign images to create difficult OOD test sets. Results for both Baseline and MC Dropout are given in Table 7 and visualized in Figure 18.

	Baseline	MC Dropout
None	0.9478 (0.0135)	0.9827 (0.0031)
Noise	0.8018 (0.0721)	0.9312 (0.0346)
Grey	0.7466 (0.0451)	0.8362 (0.0196)
Snow	0.8243 (0.0614)	0.9443 (0.0296)
Rain	0.7519 (0.0974)	0.8916 (0.0653)
Fog	0.4825 (0.1465)	0.5406 (0.1779)
Perspective	0.6054 (0.0357)	0.7170 (0.0292)
Blur	0.8483 (0.0273)	0.9575 (0.0116)
Rotate	0.8178 (0.0216)	0.9079 (0.0188)
Crop	0.7200 (0.0351)	0.8141 (0.0308)
Reflection	0.6560 (0.0407)	0.7870 (0.0272)

Table 7: AUROC values on the data transformation OOD datasets. Values are given as a fraction. For all transformations, only strength level 5 is evaluated. The "None" dataset is equivalent to the Tiny ImageNet result in Table 5. Since the results measured in different independent experiments, they differ slightly.

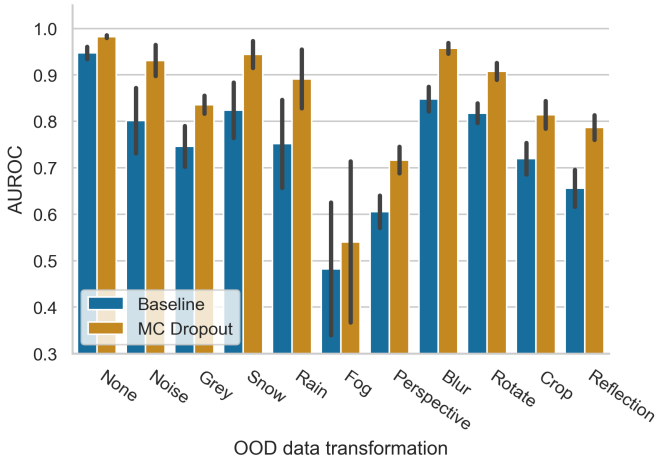


Figure 18: AUROC scores for the different OOD datasets created by data transformations. The y-axis shows the AUROC and the x-axis the used transformation. Results for the Baseline classifier (blue) and MC Dropout (orange) are indicated by color. Error bars show the standard deviation.

Statistical effects of the results in Table 7 are analyzed in a 2 (Factor: Classifier) x 10 (Factor: Transformation) ANOVA. The results (table 8) show that MC Dropout performs significantly better ($p<.001$) than the Baseline classifier. The type of the applied transformation and the interaction between the two factors are significant as well and are analysed in a post hoc comparison.

Factor	d.f.	F	p
Classifier	1	275.86	$<.001$
Transformation	10	161.74	$<.001$
Classifier * Transformation	10	2.54	$<.01$
Error	418		

Table 8: Statistical analysis of the transformed OOD datasets with the factors classifier and the applied transformation on Tiny ImageNet.

For all transformations, the classifiers perform significantly worse ($p<.001$ for all transformations) compared to using no transformations. The lowest results occur for the transformations Fog, Perspective, Reflection and Crop, which were already the most challenging transformations in the classification evaluation. Especially Fog is extremely difficult, causing the Baseline classifier to perform worse than chance (48.25%). MC Dropout is affected similarly, performing only slightly better than chance (54.05%). Since the in-distribution and OOD examples were mixed in equal proportions, random guessing would result in a performance of 50%. In line with previous results, the different transformations lead to differences in the performance of the classifiers. This shows that even within Tiny ImageNet as an OOD dataset, it is useful to look at multiple different transformations within the dataset.

Comparing the classifiers on the individual transformations shows significant differences for Noise, Snow, Rain, Perspective, Blur, Crop, and Reflection with $p<.001$ and for Grey and Rotate with $p<.01$. No significant differences were found for Fog and using no transformations. The average difference between the two classifiers on the image transformations (excluding None) is 10.73% while the average difference on Tiny ImageNet is 3.49%. This shows that using the image transformations in the context of OOD detection leads to larger differences between the classifiers.

4.2.3 OOD GAN examples

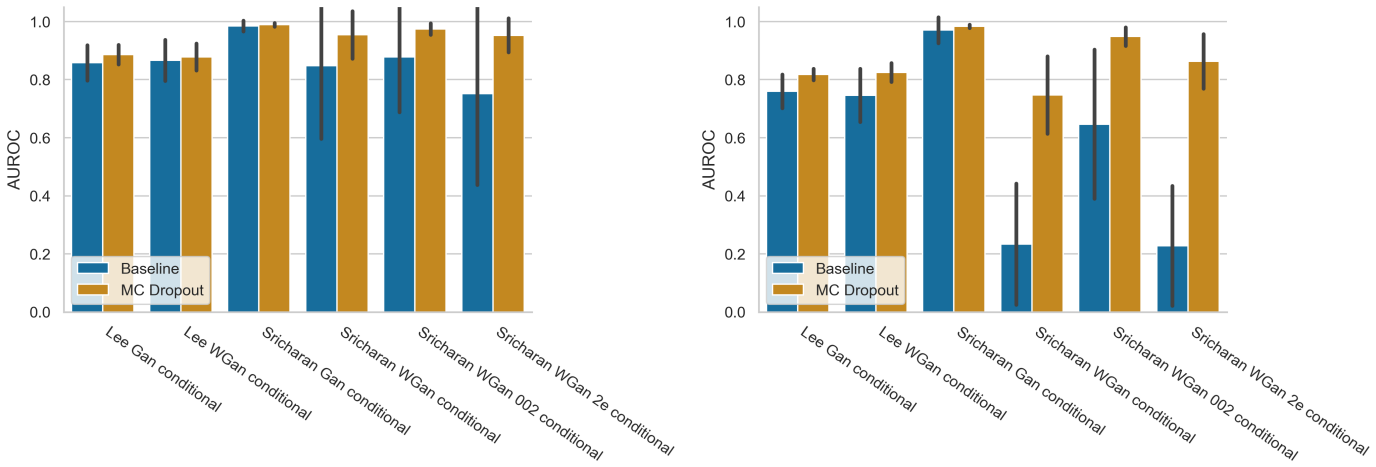
Initial results with GAN generated datasets showed a very high amount of variance. To deal with this, not only have the classifiers been tested 20 times (as with the other tests), the individual GANs have also been trained ten times from scratch, resulting in 200 measurements for each GAN generated dataset. These results are displayed in Table 9. The names of the different architectures show which loss function, or, in the case of the Sricharan architecture, which stopping criteria has been used.

The average GAN results, as shown in Table 9, do not provide many insights. Generally, the GAN datasets are not especially difficult, showing higher AUROC values than the OOD datasets created by data transformations in Table 7. Additionally, the identical GAN networks trained for each architecture show widely different results as indicated by the standard deviations of up to 0.31 in Table 9. This effect seems to be especially present for the Baseline classifier; MC Dropout performs more consistent with standard deviations of up to 0.08. Detailed results for the individually trained GANs can be found in Figure 27 in the appendix.

	Baseline	MC Dropout
Lee Gan	0.8579 (0.0606)	0.8863 (0.0337)
Lee WGan	0.8666 (0.0704)	0.8783 (0.0468)
Sricharan Gan 005	0.9849 (0.0183)	0.9893 (0.0064)
Sricharan WGan 002	0.8779 (0.1901)	0.9742 (0.0202)
Sricharan WGan 2e	0.7523 (0.3149)	0.9528 (0.0584)
Sricharan WGan 005	0.8480 (0.2520)	0.9544 (0.0821)

Table 9: Mean and standard deviation of the AUROC values on the GAN generated OOD datasets. Values are given as a fraction. AUROC results are averaged over all 10 GANs trained for the specified architecture.

Due to the high standard deviations of the individual GANs and the overall high performance of the averaged GAN networks, I will not perform a statistical analysis of these results and instead focus on the single GAN network for each architecture that produced the lowest AUROC. These results can be found in Table 10. A visualization of both the average results and the singled out GAN can be found in Figure 19. The results have been analyzed in a 2 (Factor: Classifier) x 6 (Factor: GAN architecture) ANOVA, whose results (Table 11) show strong significant differences between the classifiers and the different GAN architectures. Additionally, a significant interaction effect between the two is present, showing that the classifiers respond differently to the GAN datasets.



(a) Average GAN

(b) Hardest GAN

Figure 19: Comparison between the averaged GAN results for each architecture compared to the best performing GAN for each architecture.

	Baseline	MC Dropout
Lee Gan	0.7601 (0.0592)	0.8177 (0.0198)
Lee WGan	0.7462 (0.0942)	0.8250 (0.0335)
Sricharan Gan 005	0.9703 (0.0459)	0.9839 (0.0060)
Sricharan WGan 002	0.6469 (0.2632)	0.9486 (0.0326)
Sricharan WGan 2e	0.2278 (0.2116)	0.8631 (0.0960)
Sricharan WGan 005	0.2340 (0.2141)	0.7467 (0.1367)

Table 10: Using only the best performing GAN within each architecture, mean and standard deviation of the AUROC values are given.

Factor	d.f.	F	p
Classifier	1	249.78	<.001
GAN architecture	5	76.68	<.001
Classifier * GAN architecture	5	39.91	<.01
Error	228		

Table 11: Statistical analysis of the GAN generated OOD datasets with the factors classifier and the used GAN architecture. Classifiers are indicated by color: Baseline (blue) and MC Dropout(orange). Error bars show the standard deviation.

Comparing MC Dropout to the Baseline classifier, MC Dropout performs better across all GAN architectures (see Table 10). This difference is significant ($p<.001$) for all three "Sricharan WGan" architectures (last 3 rows in Table 10). Comparing the performance on these three GAN architectures in detail shows that the Baseline classifier has been affected much stronger by the GAN examples, dropping far below random chance for "Sricharan WGan 2e" and "Sricharan WGan 005". These results show that the Baseline classifier was even more certain of OOD examples than the actual in-distribution images. MC Dropout has been affected much less, never scoring lower than 74% AUROC.

In general, while not all GANs had the anticipated effects on the classifiers, finding a GAN that works well

allows generating OOD examples that achieve the lowest overall AUROC scores of all the experiments.

5 General Discussion

5.1 Interpretation of results

The goal of this thesis was to develop an evaluation methodology for image classifiers. Returning to the questions I tried to answer with developing this evaluation (see section 1.5), the pipeline has added value to the evaluation. It highlights that the robustness of classifiers can not be assessed properly when only using a standard test set. Instead, different image transformations have been applied successfully to find conditions the classifiers struggle with. The image transformations worked especially well for the conditions Fog, Perspective, Crop and Reflection across all three measures. Comparing performance between different conditions helps in answering the question of how well the classifier functions in challenging environments.

Both the AUROC and the Brier score help to assess the classifier's ability to give accurate certainty predictions. Comparing the performance with the use of my image transformations showed that for certainty estimates, the same conditions caused problems for both classifiers. The usefulness of looking at both scores is highlighted by the fact that in my experiments, MC Dropout had a better performance on the AUROC but the Baseline classifier performed better when looking at the Brier score. Analyzing the classifier's performance using these two measures in combination with the image transformations helps to answer the question of how reliable a classifier's certainty estimates are.

When analyzing the classifier's ability to detect out-of-distribution (OOD) examples, my own OOD datasets showed success in evaluating this task under challenging conditions and exposed vital weak spots in the classifier. Applying image transformations to this task shows the same difficulties as before: Fog, Perspective, Crop and Reflection cause major problems for the classifiers. Additionally, GAN examples proved useful in identifying edge cases that the classifier identifies incorrectly.

In the following sections, I will interpret my experimental results in more detail and then discuss the relation to existing work, limitations, and directions for future research.

5.2 Classification evaluation

The general effect of the different levels of the transformations is that increasing the level of a transformation increases the difficulty, resulting in lower overall performance of the classifiers (see Figures 11, 13, 15). In terms of accuracy, this is to be expected since applying a stronger transformations makes the image less alike to the original in-distribution image. The increase in difficulty also carries over to the measures evaluating the certainty of the classifier. This shows the classifiers shortcomings in generalizing to predicting novel examples and giving meaningful certainty estimates.

The ten different transformations produce very different results (Figures 12, 14, 16). Some transformations like Blur and Snow do not seem to have a big impact on the classifiers while other transformations impact the performance drastically. Reflection, Fog and Perspective have an especially big impact. Different results for different transformations indicate that testing the classifiers with multiple different transformations is necessary to evaluate them properly. The difficulty of different transformations is mostly consistent across the three measures (accuracy, AUROC and Brier). For example, Perspective, Reflection, Crop and Fog were the most difficult transformations for all three evaluation measures. When the classifier starts to have bigger problems classifying the transformed examples correctly, it also loses the ability to give high-quality certainty estimates. This highlights the problem of overconfident classifiers, where the classifier not only cannot classify the examples correctly any more, but it is also unaware of that fact.

Taking into account that the transformations had consistent effects across evaluation measures and that both classifier architectures are affected similarly by the different transformations, it seems like the transformations have an overarching level of difficulty. This pattern is only visible in higher levels of transformations (see Figures 22, 23 and 25 in the appendix), making the importance of a step-wise evaluation (in which multiple levels of transformation or difficulty are used) clear. Since my experiments have only been performed on the CURE-TSD [44] dataset, it is not clear if these difficulties with specific transformations are persistent when applying the pipeline to a different domain. The fact that both classifiers are affected similarly by the different transformations can, at least in part, also be attributed to the fact that the architectures and training procedures of both classifiers are quite similar.

Interestingly, the transformations that cause the most problems for the classifiers are not necessarily harder to classify for humans. Subjectively, I would identify Snow and Fog as the most challenging transformations. For Fog, this is consisted with the classifier's performance. However, snow is one of the transformations that were the most unchallenging for the classifiers. Conversely, Reflection, Crop and Perspective would be among the subjectively easy transformations for humans. This indicates potential differences between how humans perceive their environment while driving to the perception of automated vehicles. Ultimately, this suggests that humans have different weaknesses in their perception than image classifiers and that a cooperation between human and machine might be beneficial to assess a driving environment properly, which also fits the trends of human-automation interaction identified by Janssen et al. (2019) [17].

When comparing the Baseline classifier and MC Dropout using the pipeline (see Figure 9), it becomes clear that MC Dropout is the more robust classifier since it significantly outperforms the Baseline classifier on the accuracy and AUROC measures. On the Brier measure, the Baseline classifier performs better. For the AUROC and the Brier score, these findings are in line with the performance on the untransformed test set. For accuracy, however, this effect is only visible when looking at the pipeline evaluation. In the original test set, the Baseline performed better (although non-significantly) while only in the pipeline evaluation, MC Dropout achieves a significantly higher accuracy score. This shows the value of comparing classifiers in this way; the pipeline evaluation is able to show differences in the robustness of the classifiers that are not detectable without it.

For the Brier score, the Baseline classifier performs better than MC Dropout (see Table 9). The poor performance of the MC Dropout classifier shows problems in its calibration, which is unexpected since the network was specifically designed to output more accurate certainties [5]. A possible explanation is the low difficulty level of the used dataset. Since both classifiers can achieve scores close to 99% accuracy on the initial test set, the overconfidence of the Baseline classifier is, in this case, warranted. MC Dropout outputs are automatically lower than the outputs of the baseline since the prediction is generated by essentially averaging multiple stochastic passes through the network. This is supported by the fact that in higher transformation levels (which cause lower performance scores) this difference is reduced and at level 5, the Brier score between the classifiers is no longer significantly different.

Since the Brier score evaluates a similar property to the AUROC, it might be surprising to see results this different on the two measures. However, this can be explained by the fact that the measures are independent of each other. For the AUROC measure, the actual certainty values are not relevant. Instead, only the difference in the certainty values for correctly and incorrectly classified examples is relevant. The Brier score, however, evaluates only the concrete certainty value of the prediction, which can be utilized differently: To avoid classifiers classifying examples incorrectly without noticing, a high AUROC is more relevant than the Brier score. However, when the classifier's certainty is used directly, the Brier score is more important. An example of this are the later processing stages in automated vehicles, where decisions can rely the classifiers confidence in its perceptions.

Across measures, the actual scores achieved by classifiers carry much useful information for potential real-world use cases as well. Since the transformations were designed to evaluate the classifiers in situations they can potentially encounter during their deployment, poor performance scores indicate that the classifier is not robust enough to be used in the real world. For example, the Baseline classifier's accuracy drops from

around 98% for Fog at level 1 to under 57% for Fog at level 5 (see Table 13). In the domain of automated vehicles, this arguably shows that the classifier is unfit to be used on the road since driving during fog is a realistic scenario to occur and the vehicle needs to be able to function under this condition.

While my experiments and results are often focused on comparing the two classifiers, in a more realistic scenario, it is likely that only a single classifier is evaluated at a time. In that case, as outlined in the previous paragraph, the actual performance scores of the classifier become more important. I have not analysed these scores in detail since my classifiers are only used to demonstrate the functionality of the pipeline, their actual performance is less of importance than the effects of the transformations and test sets I introduced.

To summarize these results, the different image transformations and the multiple levels with which each transformation is applied give new insights into the performance of the classifiers that were not visible before. When comparing the two classifiers, the different transformations and levels are especially important to show detailed differences between the classifiers. These are mostly visible when looking at higher levels of transformations, while lower levels show mostly similar responses (see Figures 11, 13 and 15). In the same fashion, specific transformations show larger differences between the classifiers than others (see Figures 12, 14, 16). Assessing the performance across the different transformations and levels shows blind spots in the classifier's robustness that were not clear before. Looking at the performance across different levels gives nuanced comparison points between multiple tested classifiers, even when they initially perform very similarly. This shows the general usefulness of the pipeline evaluation when judging the generalization of a classifier. Evaluating the classifiers without any transformations on the original test set, the baseline classifier outperforms MC Dropout by a small amount (see Table 9). In the results for the image transformations (see Figure 22), the two classifiers perform increasingly different from one another.

5.3 OOD evaluation

In my experiments using OOD test sets from the literature (see the first three rows in Table 5), both classifiers perform well and also similarly. The high performance is problematic because it shows that the test sets are too easy and do not give the classifiers a realistic and challenging evaluation of the OOD detection task. Therefore, they are not very useful when determining if the classifier is able to do this task in a more challenging situation during deployment. This is also highlighted by the fact that both classifiers perform similarly even though MC Dropout should be more robust and better at this task by design. Therefore, I tried to find more challenging test sets that show weaknesses in the classifiers robustness and additionally also highlight differences between the classifiers.

In both cases, the results from my own OOD test datasets show clear improvements in the form of performance drops and stronger differences between the Baseline classifier and MC Dropout. I consider worse performance from the classifiers as positive for two reasons: Firstly, both used classifiers, and especially the Baseline classifier, are not trained to do this kind of task at all. High performance is therefore not expected and indicates to me that the applied test is insufficient. My test sets show shortcomings of the classifiers in the area of OOD detection that are simply not visible in previous test sets. Secondly, the OOD datasets I introduced are relatively easy to distinguish for humans (see the example images in Figure 6). Since the task is doable for humans and also a realistic example of a situation a classifier might encounter in the real world, poor performance on this task gives a strong argument for the classifiers being unsafe to deploy. This shows the usefulness of the created OOD datasets and the OOD evaluation in general when assessing the robustness of classifiers. Comparing the OOD datasets in more detail, nearly all newly introduced OOD datasets show a much lower average performance score than the datasets used in the literature (e.g. [11, 4, 28]).

These OOD data transformation test datasets introduce the novel idea of using data augmentations on both the in-distribution and OOD images. This shows to be highly effective to create more difficult OOD test cases while also being very relevant to evaluate real-world capabilities. An automated vehicle that is, for example, in use during foggy conditions will also need to perform OOD detection with fog obscuring both

the traffic signs and other objects. The test sets generated with data augmentations show lower AUROC scores and bigger differences between the two tested classifiers (compared to my previous results in Table 5). Therefore, the OOD data transformation test sets help when evaluating the classifiers since they give a clearer picture of the expected performance in realistic scenarios. The high performance scores on the undistorted Tiny ImageNet dataset (see Figure 5) is misleading and does not represent the actual OOD detection performance when considering that in-distribution, as well as OOD images, are potentially subject to distributional shift.

The GAN generated data sets did, on average, not work as well as the other newly introduced test sets. This can be explained by the black-box nature of the evaluation. Instead of directly training a GAN to generate examples that work well on the tested classifier, a surrogate model is used in the GAN training procedure. This extra step is necessary since the pipeline should function in a black-box setting without accessing the weights or architecture of the classifier under test. Transferring the GAN examples to the tested classifier does not always work well and is highly dependent on how similar the pattern both classifiers learn during training is. It also causes an especially high variance in this experiment.

However, looking at the GAN networks that worked well still provides useful insights. Some of the GANs were extremely difficult, causing the classifier to perform much worse than random chance (see Table 10). Even though the GAN examples do not always cause problems, the existence of some GANs that cause scores that are this low already indicates a safety risk in itself. Especially in the case of the Sricharan GAN architecture, the images look like noise patterns to us humans and show no visual similarity to traffic signs (see Figure 8). For humans, distinguishing these GAN images from in-distribution images is very easy. In the classifiers that struggle with the same task, the GAN images expose critical blind spots and edge cases in the classifier's learned patterns about the domain. These can cause severe safety-risks since they show that the classifier can be fooled even by images that have no visual resemblance to in-distribution images. Therefore, the GAN images provide useful counterexamples that disprove the robustness of the classifier in edge cases.

5.4 Relation to existing work

Koopman et al. (2018) [20] outline fundamental problems of testing automated vehicles. One of their findings was that current testing procedures are not fitting for machine learning components due to their black-box nature and the general infeasibility of testing every possible scenario. Instead, he proposes a test centered approach. This is what this thesis offers as a proof-of-concept in a subdomain of automated vehicles: perception using machine learning classifiers.

Comparison to existing robustness evaluation frameworks shows two main advantages of the pipeline. Firstly, this thesis proposes an evaluation that considers unbounded and strong variations to the input images while accepting that proving robustness will not be possible anymore. The extend of the transformations contrast with existing frameworks that are mostly focused on adversarial attacks [13, 7]. These frameworks can prove the network's robustness, albeit only in a small, defined area of acceptable perturbations limited in strength and numbers of affected pixels. In real-world scenarios, however, perturbations occurring to the inputs are unlikely to be bounded in this way. Secondly, my evaluation framework is designed as a black-box evaluation where only a test dataset and the option to get predictions of the classifier are required. This offers practical advantages over existing white-box evaluations [13, 7, 33], since companies will be more likely to allow potential testers access to their proprietary classification network in a black-box setting.

The use of test time data augmentation is, in general, not widespread in machine learning. Conventional evaluation methods of neural networks consist of a separate test set that is withheld during training. The problem with this is that the test images come directly from the training distribution and are therefore contain the same biases and limitations present in the initial dataset. This method is therefore not suitable to evaluate the classifier situations with a distributional shift.

When test time data augmentation is used, it is done to train a more robust network (e.g. [46, 34, 29]), while my work focuses on evaluating the robustness of a pre-existing network. This means current applications of test time augmentations are using the same kind of transformations during training and testing. In this thesis, test time augmentation is used differently: Instead of training the classifier with the augmented images, the data augmentation is purely used during the evaluation to test whether the classifier is able to generalize to the new images. This creates a stronger separation between training and evaluation of classifiers and emphasizes the testing of difficult and unforeseen scenarios. The implemented transformations (such as Fog and Rain) have the purpose of transforming the image in a realistic way that evaluates the classifiers ability to perform in similar real-life environments.

Looking at existing work in the area of OOD detection, the task is primarily evaluated with relatively simple datasets (e.g. [11, 25, 4]). An explanation for this is that existing work focuses on finding new methods to detect OOD examples and not on the extensive evaluation of OOD detection in general. While my work does not propose a new detection method and follows the evaluation method by Hendrycks et al. (2016) [11], it extends existing tests by introducing more challenging and varied test datasets. The proposed new test sets can be generated automatically based on the provided test set, which allows evaluation on different datasets. The new OOD examples include the use of data augmentation techniques and GAN generated images. Improvements from these datasets are more challenging test cases and stronger differences between the performance of different classifiers. Finding difficult test cases is of high importance since we are trying to evaluate how well a classifier can carry out the task in a real-world scenario. Since it is not possible to exhaustively test all possible OOD images, it is detrimental to find meaningful and difficult test sets that evaluate how the classifier reacts in challenging real-world conditions.

5.5 Implications for theory and practice

The main contribution of this work is methodological. It offers a novel method of evaluating the robustness and safety of machine-learning image classifiers. This method can be applied to both scientific problems and practice. The purpose of the evaluation is to systematically test the classifier in difficult and realistic scenarios, which helps to expose vital weaknesses and blind spots in the classifier's training. A classifier passing all the proposed test cases is therefore not necessarily safe in all possible real-world scenarios, but a classifier failing some of the tests shows unsafe behavior and should, therefore, be investigated in more detail.

In addition, the evaluation method offers three contributions to science and, specifically, machine learning. Firstly, the joint evaluation of difficult in-distribution images and OOD examples creates a first approach to an extensive evaluation of different occurring effects of distributional shift. Secondly, the classification evaluation combines existing research in the area of data augmentation, test time data augmentation [34], calibration [10] and misclassification identification [11]. Combining both model certainty and accuracy leads to an extensive evaluation of the classification performance, ability to generalize and the quality of the classifiers certainty estimates. Thirdly, in the area of OOD detection, this work extends the evaluation of the task through the introduction of new and more varied test cases. Novel contributions in this context are the introduction of multiple new OOD test sets and the use of data augmentation to create more difficult OOD test sets automatically based on the provided test data.

In offering a feasible and automatic way to test AI software components, the pipeline can ultimately contribute to practical safety evaluations for real-world systems. For example, this can be applied in type-approval procedures where authorities could use the pipeline to evaluate the safety of the AI systems used in automated vehicles. Practically, this can be implemented by creating use-case dependent pass/fail criteria a vehicle has to fulfill to be allowed on public roads. The construction of the pipeline as a black-box evaluation method creates an acceptable solution for both industry and authorities because the pipeline does not require companies to make their proprietary AI software accessible.

5.6 Limitations and future work

The current work has three limitations and associated paths for future work. The first of these is caused by the CURE-TSD dataset [44]. The cropped images I used show a low variance and are therefore quite easy to perform well on. It is possible that, when using a more complex dataset, the pipeline has different effects on the classifiers. The low initial variance also potentially made the OOD detection task easier since the concept of what should be identified as a traffic sign was (visually) very clearly defined. In general, the choice of this dataset was intentional since it allowed me to use relatively small CNNs and kept the training times short. This allowed me to focus on the evaluation itself instead of optimizing performance on a difficult dataset, which was not the goal of the experiments.

Secondly, to test the pipeline procedure's generalization, it would be necessary to apply the whole pipeline to multiple domains using different datasets of images. It would be especially interesting to see if the same datasets and data transformations lead to major difficulties that caused problems on the traffic sign dataset. Applying the transformations to a different domain will likely require the strength of the transformations to be adjusted since they were tuned to produce difficult, but identifiable examples on the CURE-TSD dataset. On a different dataset, especially if that dataset is already harder to classify, the transformations might need to be weakened. This also means that performance scores can not be directly compared across domains.

Lastly, a limitation for the practical use of the proposed pipeline is the focus on image classification. Real-world applications in the domain of automated vehicles are likely to be more complex. Instead of identifying the class of an image (image classification), the more likely task is detecting (multiple) objects within an image (object recognition). Additionally, image data is likely available in the form of videos and augmented by additional sensory data such as LiDAR. However, pure image classifiers are still used and relevant in practice, for example, in a modular system where traffic signs are identified via object detection and then classified with an image classifier. In this scenario, the classifier would have a very similar task to the one used in my experiments. Additionally, the reduced domain of images allows me to better assess the function of my evaluation methodology in a constrained and simplified setting. The evaluation can be extended to more complex settings in future work.

In addition to widening the evaluation domain, potential future work can be carried out in multiple directions. It is possible to extend the range of evaluations done in the pipeline to get an even more extensive picture of the evaluated classifier. One possible inclusion would be to test the robustness of adversarial examples. Adversarial examples are small perturbations that are imperceptible to humans but can cause big changes in the targeted classifier [8].

Instead of applying the evaluation to different domains and datasets, it would also be interesting to see if the pipeline can be extended to other types of machine learning models where robustness is of relevance. The most similar and relevant one would be object detection, to which the current approach could be adapted relatively quickly. Another option would be to try using a similar approach based on data-augmentation in reinforcement learning, where robustness is also a highly relevant criterion.

To use the pipeline in a real-world setting, it would also be necessary to define safe thresholds of performance the classifier has to achieve to be deemed safe enough. For example, which accuracy does the classifier need when looking at the snow image transformation? This is highly dependent on the use case and can not be answered easily. It is also closely connected to ethical questions and should probably be solved by policy-makers instead of AI researchers.

Another extension for real-world application are the used image transformations. The current transformations were designed through expert knowledge and offer a range of different scenarios the classifier might encounter. When using the evaluation in practice, this could be part of a much larger database of transformations that is continuously updated with new problem cases recorded by vehicles currently in use. Ultimately, this combination of expert knowledge and gathered data could allow a stronger evaluation of new classifiers: it would potentially be possible to quantify what percentage of expected situations the classifier is suitable for.

5.7 Conclusion

In conclusion, this thesis proposed a method for evaluating image classifiers used in safety-critical real-world scenarios and tested this in a proof-of-concept implementation. Since the majority of transformations were applied with realism in mind, lacking performance on these tests gives direct indications of scenarios the classifier is not equipped to deal with. Within the evaluation pipeline, both cases of safety-critical behavior are tested: classifying challenging in-distribution images correctly and not falsely identifying difficult out-of-distribution images as in-distribution. Safety assessment is an essential component of the deployment of safety-critical systems, and this research could ultimately contribute to the safety approval procedure for real-world systems such as automated vehicles.

References

[1] Dario Amodei et al. "Concrete problems in AI safety". In: *arXiv preprint arXiv:1606.06565* (2016).

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein gan". In: *arXiv preprint arXiv:1701.07875* (2017).

[3] Dutch Safety Board. *Who is in control? Road safety and automation in road traffic*. `https://www.onderzoeksraad.nl/en/page/4729/who-is-in-control-road-safety-and-automation-in-road-traffic`. 2019.

[4] Terrance DeVries and Graham W Taylor. "Learning confidence for out-of-distribution detection in neural networks". In: *arXiv preprint arXiv:1802.04865* (2018).

[5] Yarin Gal and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning". In: *international conference on machine learning*. 2016, pp. 1050–1059.

[6] Bernd Gassmann et al. "Towards Standardization of AV Safety: C++ Library for Responsibility Sensitive Safety". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 2265–2271.

[7] Timon Gehr et al. "Ai2: Safety and robustness certification of neural networks with abstract interpretation". In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018, pp. 3–18.

[8] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2014).

[9] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[10] Chuan Guo et al. "On calibration of modern neural networks". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 1321–1330.

[11] Dan Hendrycks and Kevin Gimpel. "A baseline for detecting misclassified and out-of-distribution examples in neural networks". In: *arXiv preprint arXiv:1610.02136* (2016).

[12] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. "Deep anomaly detection with outlier exposure". In: *arXiv preprint arXiv:1812.04606* (2018).

[13] Xiaowei Huang et al. "Safety verification of deep neural networks". In: *International Conference on Computer Aided Verification*. Springer. 2017, pp. 3–29.

[14] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[15] ISO. *Road vehicles – Functional safety*. Norm. 2011.

[16] ISO ISO. "PAS 21448-Road Vehicles-Safety of the Intended Functionality". In: *International Organization for Standardization* (2019).

[17] Christian P Janssen et al. "History and future of human-automation interaction". In: *International journal of human-computer studies* 131 (2019), pp. 99–107.

[18] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[19] Philip Koopman and Michael Wagner. "Challenges in autonomous vehicle testing and validation". In: *SAE International Journal of Transportation Safety* 4.1 (2016), pp. 15–24.

[20] Philip Koopman and Michael Wagner. *Toward a framework for highly automated vehicle safety validation*. Tech. rep. SAE Technical Paper, 2018.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[22] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles". In: *Advances in neural information processing systems*. 2017, pp. 6402–6413.

[23] Ya Le and Xuan Yang. "Tiny imagenet visual recognition challenge". In: ().

[24] Kimin Lee et al. "A simple unified framework for detecting out-of-distribution samples and adversarial attacks". In: *Advances in Neural Information Processing Systems*. 2018, pp. 7167–7177.

[25] Kimin Lee et al. "Training confidence-calibrated classifiers for detecting out-of-distribution samples". In: *arXiv preprint arXiv:1711.09325* (2017).

[26] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. "Enhancing the reliability of out-of-distribution image detection in neural networks". In: *arXiv preprint arXiv:1706.02690* (2017).

[27] Rowan McAllister et al. "Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning". In: International Joint Conferences on Artificial Intelligence, Inc. 2017.

[28] Alexander Meinke and Matthias Hein. "Towards neural networks that provably know when they don't know". In: *arXiv preprint arXiv:1909.12180* (2019).

[29] Tran Ngoc Minh et al. "Automated Image Data Preprocessing with Deep Reinforcement Learning". In: *arXiv preprint arXiv:1806.05886* (2018).

[30] Mehdi Mirza and Simon Osindero. "Conditional generative adversarial nets". In: *arXiv preprint arXiv:1411.1784* (2014).

[31] Nicolas Papernot et al. "Practical black-box attacks against machine learning". In: *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. 2017, pp. 506–519.

[32] Adam Paszke et al. "Automatic differentiation in pytorch". In: (2017).

[33] Kexin Pei et al. "Deepxplore: Automated whitebox testing of deep learning systems". In: *proceedings of the 26th Symposium on Operating Systems Principles*. 2017, pp. 1–18.

[34] Fábio Perez et al. "Data augmentation for skin lesion analysis". In: *OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis*. Springer, 2018, pp. 303–311.

[35] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[36] DeepMind Safety Research. *Building safe artificial intelligence: specification, robustness, and assurance*. `https://medium.com/@deepmindsafetyresearch/building-safe-artificial-intelligence-52f5f75058f1`. 2018.

[37] Rick Salay and Krzysztof Czarnecki. "Using machine learning safely in automotive software: An assessment and adaption of software process requirements in iso 26262". In: *arXiv preprint arXiv:1808.01614* (2018).

[38] Rick Salay et al. "PURSS: Towards Perceptual Uncertainty Aware Responsibility Sensitive Safety with ML". In: ().

[39] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. "On a formal model of safe and scalable self-driving cars". In: *arXiv preprint arXiv:1708.06374* (2017).

[40] Connor Shorten and Taghi M Khoshgoftaar. "A survey on image data augmentation for deep learning". In: *Journal of Big Data* 6.1 (2019), p. 60.

[41] Kumar Sricharan and Ashok Srivastava. "Building robust classifiers through generation of confident out of distribution examples". In: *arXiv preprint arXiv:1812.00239* (2018).

[42] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[43] Farhana Sultana, Abu Sufian, and Paramartha Dutta. "Advancements in image classification using convolutional neural network". In: *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*. IEEE. 2018, pp. 122–129.

[44] Dogancan Temel et al. "CURE-TSR: Challenging unreal and real environments for traffic sign recognition". In: *arXiv preprint arXiv:1712.02463* (2017).

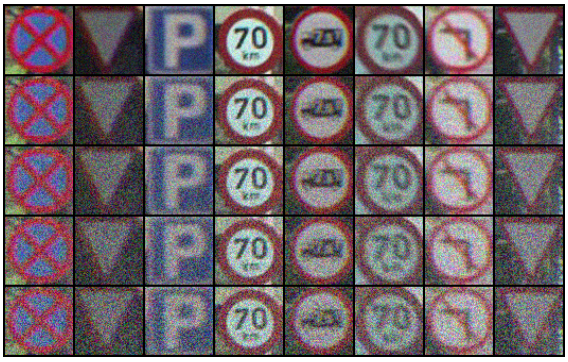
[45] Dimitris Tsipras et al. "Robustness may be at odds with accuracy". In: *arXiv preprint arXiv:1805.12152* (2018).

[46] Guotai Wang et al. "Automatic brain tumor segmentation using convolutional neural networks with test-time augmentation". In: *International MICCAI Brainlesion Workshop*. Springer. 2018, pp. 61–72.

[47] Bianca Zadrozny and Charles Elkan. "Transforming classifier scores into accurate multiclass probability estimates". In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2002, pp. 694–699.

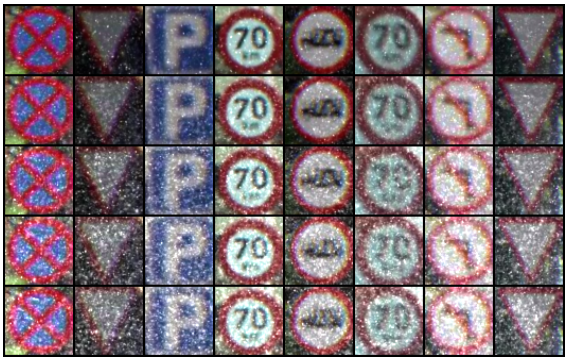
6 Appendix

6.1 Image transformations



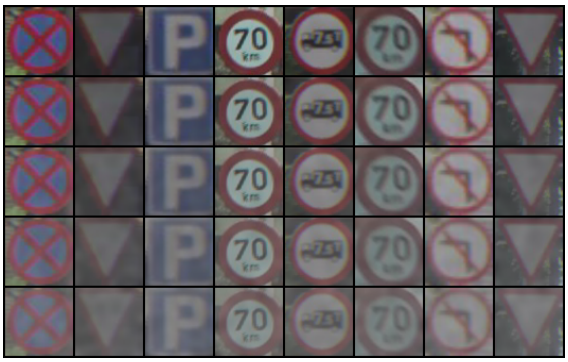
(a) Noise

(b) Grey

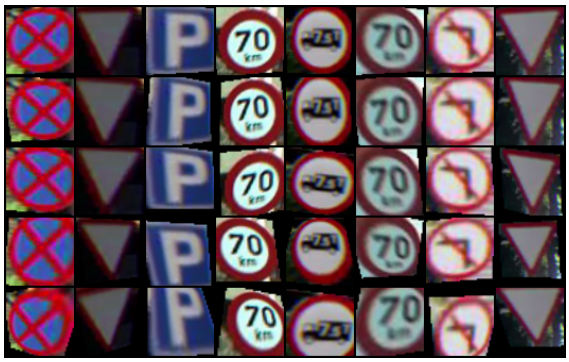


(c) Snow

(d) Rain



(e) Fog



(f) Perspective

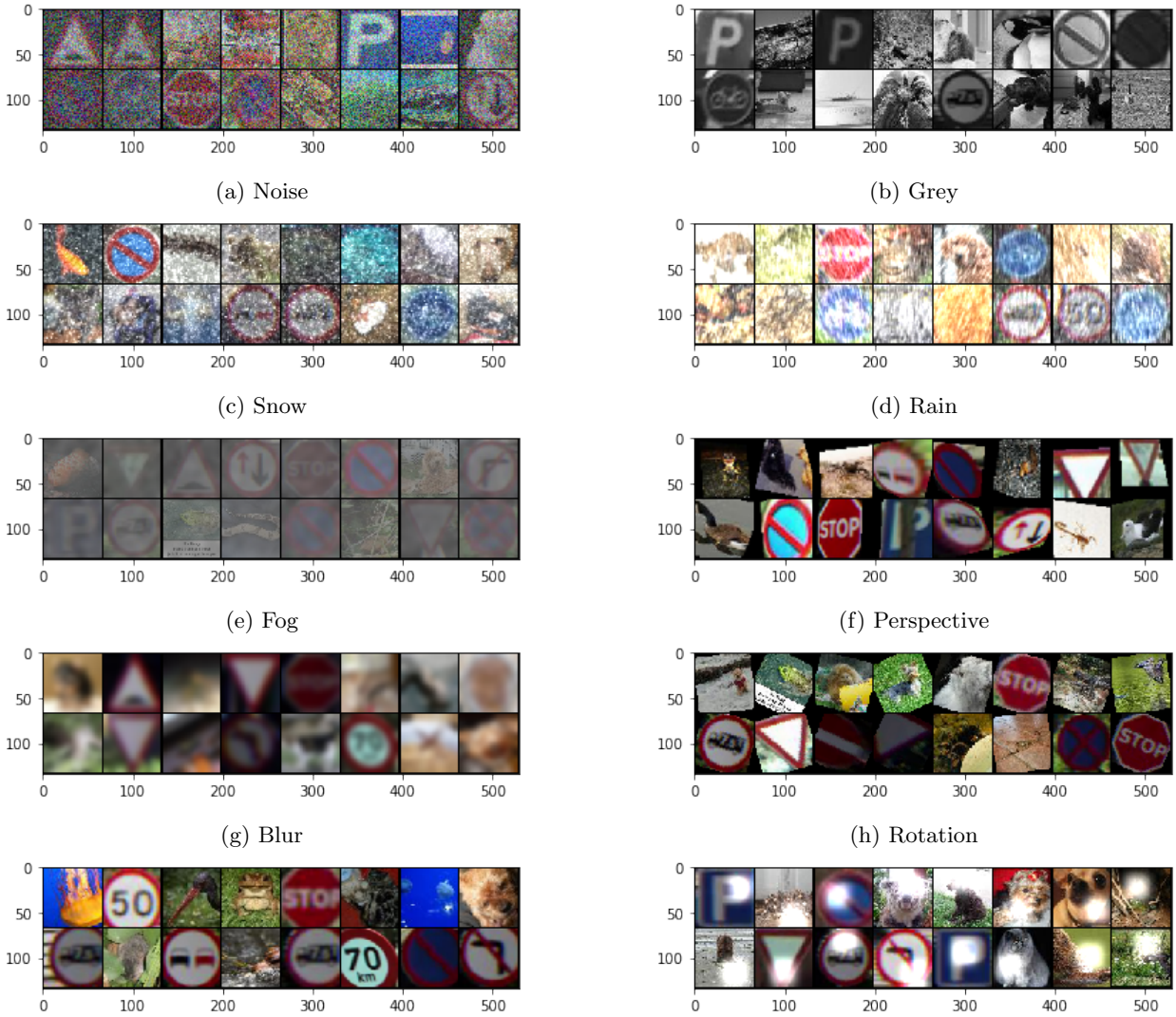
(g) Blur

(h) Rotation

(i) Crop

(j) Reflection

Figure 20: All image transformations with examples from 5 levels are shown. Within each transformation, the first row of images shows Level 1, subsequent rows show Levels 2, 3, 4 and 5 of the transformation.



(a) Noise (b) Grey

(c) Snow (d) Rain

(e) Fog (f) Perspective

(g) Blur (h) Rotation

(i) Crop (j) Reflection

Figure 21: All 10 OOD image transformation data sets. Transformations are applied with strength level 5. Examples are randomly selected from in-distribution and OOD images.

	Level 1	Level 2	Level 3	Level 4	Level 5
Noise	0.2	0.35	0.4	0.45	0.5
Grey	0.2	0.4	0.6	0.8	1
Snow	1	1, 1	1.5, 1.5	1.5, 1.5, 1	1.5, 1.5, 1.5
Rain	0.1, 0.1	0.2, 0.2	0.3, 0.3	0.4, 0.4	0.5, 0.4, 0.2
Fog	0.3	0.4	0.5	0.6	0.7
Perspective	0.2, 0.9	0.3, 0.85	0.4, 0.8	0.5, 0.75	0.6, 0.7
Blur	1.5	2	2.5	3	3.5
Rotation	10	15	20	25	30
Crop	62	60	58	56	54
Reflection	8	12	16	20	24

Table 12: Transformation parameters used in my experiment. For an explanation of the different transformations, see section 3.1.1.

6.2 Classification results

6.2.1 Accuracy

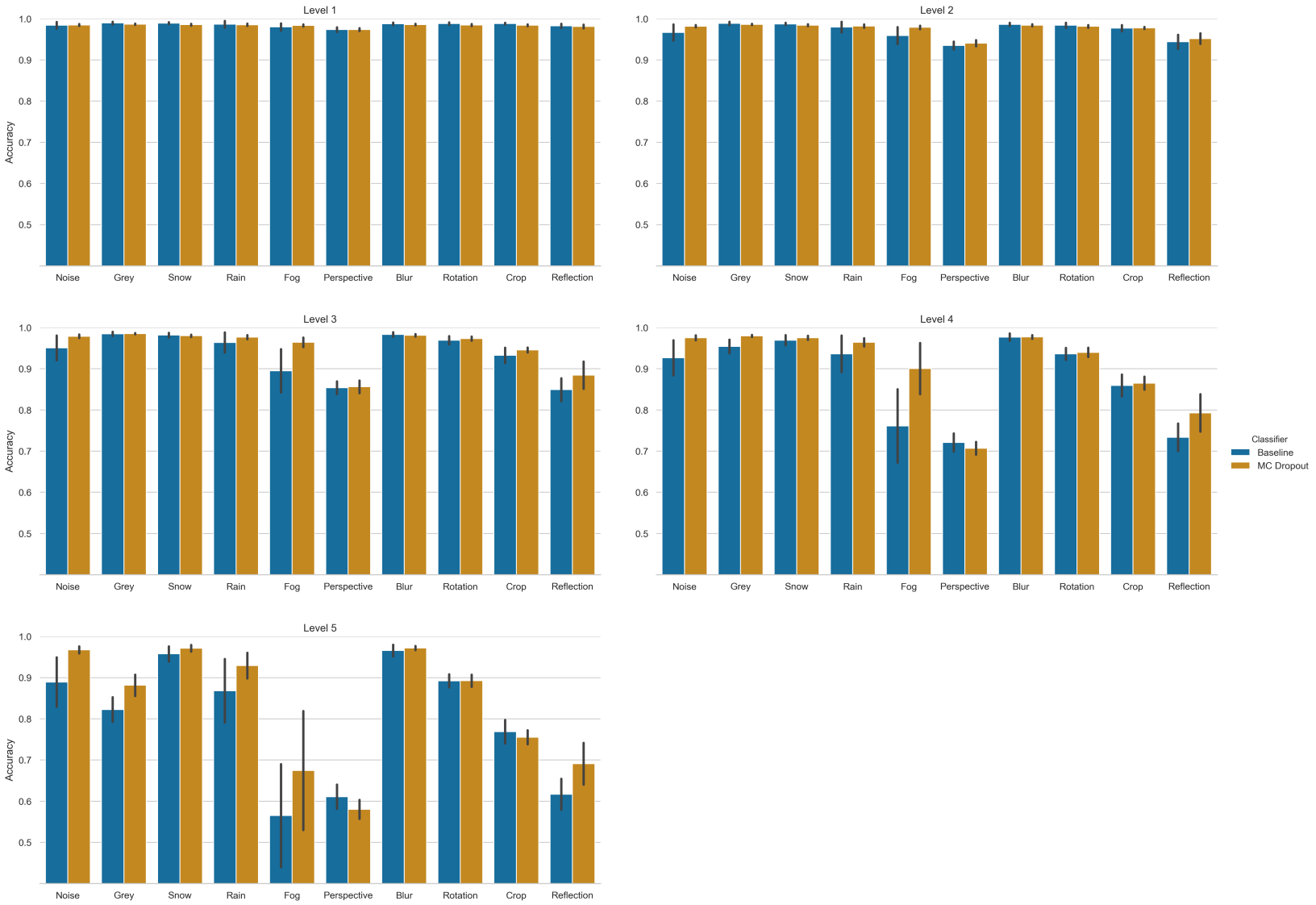


Figure 22: Each individual plot visualizes the results across all transformations for one strength level. Within each plot, the different transformations are shown on the x axis and the accuracy (as a fraction) on the y axis. The hue of the bars represent the two classifiers: Baseline (blue) and MC Dropout (orange). The error bars show the standard deviation.

	Baseline					MC Dropout				
	Level 1	Level 2	Level 3	Level 4	Level 5	Level 1	Level 2	Level 3	Level 4	Level 5
Noise	0.9843	0.9667	0.9509	0.9268	0.8893	0.9850	0.9819	0.9790	0.9754	0.9676
Grey	0.9896	0.9888	0.9848	0.9545	0.8228	0.9868	0.9863	0.9854	0.9799	0.8815
Snow	0.9895	0.9875	0.9817	0.9696	0.9579	0.9856	0.9844	0.9803	0.9755	0.9717
Rain	0.9866	0.9796	0.9641	0.9363	0.8684	0.9855	0.9822	0.9769	0.9643	0.9292
Fog	0.9800	0.9592	0.8950	0.7613	0.5652	0.9837	0.9790	0.9645	0.9007	0.6746
Perspective	0.9737	0.9351	0.8540	0.7212	0.6110	0.9738	0.9406	0.8564	0.7071	0.5804
Blur	0.9877	0.9863	0.9833	0.9770	0.9658	0.9859	0.9842	0.9815	0.9773	0.9720
Rotation	0.9882	0.9842	0.9693	0.9362	0.8924	0.9847	0.9816	0.9734	0.9400	0.8927
Crop	0.9883	0.9772	0.9328	0.8594	0.7689	0.9842	0.9779	0.9458	0.8653	0.7553
Reflection	0.9826	0.9444	0.8493	0.7337	0.6170	0.9814	0.9518	0.8845	0.7929	0.6909

Table 13: Accuracies of baseline and MC Dropout classifiers. Results are given as a fraction and rounded to the fourth decimal. A detailed visualization of the results can be found in Figure 22 in the appendix.

	Baseline					MC Dropout				
	Level 1	Level 2	Level 3	Level 4	Level 5	Level 1	Level 2	Level 3	Level 4	Level 5
Noise	0.0081	0.0201	0.0309	0.0438	0.0616	0.0021	0.0028	0.0043	0.0061	0.0088
Grey	0.0035	0.0040	0.0051	0.0170	0.0306	0.0017	0.0016	0.0015	0.0028	0.0267
Snow	0.0023	0.0030	0.0061	0.0126	0.0188	0.0023	0.0022	0.0028	0.0051	0.0082
Rain	0.0082	0.0134	0.0251	0.0456	0.0788	0.0027	0.0040	0.0051	0.0104	0.0319
Fog	0.0090	0.0209	0.0541	0.0916	0.1284	0.0025	0.0044	0.0119	0.0638	0.1483
Perspective	0.0056	0.0096	0.0159	0.0226	0.0302	0.0036	0.0077	0.0157	0.0159	0.0238
Blur	0.0031	0.0039	0.0058	0.0093	0.0147	0.0019	0.0024	0.0030	0.0045	0.0050
Rotation	0.0031	0.0061	0.0101	0.0148	0.0161	0.0026	0.0032	0.0052	0.0115	0.0152
Crop	0.0019	0.0075	0.0191	0.0272	0.0295	0.0022	0.0024	0.0064	0.0163	0.0174
Reflection	0.0051	0.0175	0.0286	0.0343	0.0387	0.0045	0.0132	0.0339	0.0465	0.0520

Table 14: Standard deviations of Table 13.

6.2.2 AUROC

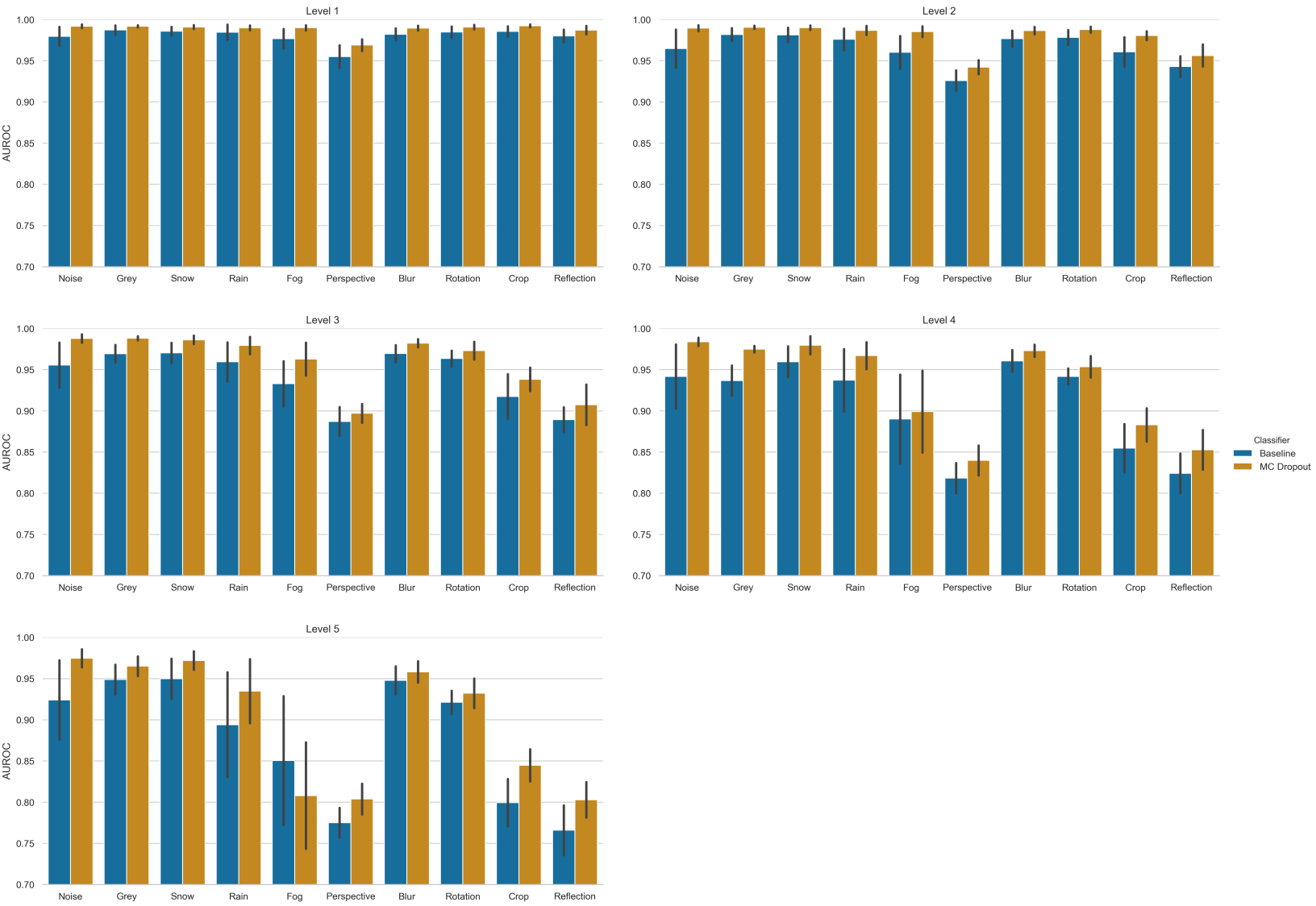


Figure 23: Each individual plot visualizes the results across all transformations for one strength level. Within each plot, the different transformations are shown on the x axis and the AUROC on the y axis. The hue of the bars represent the two classifiers: Baseline (blue) and MC Dropout (orange). The error bars show the standard deviation.

	Baseline					MC Dropout				
	Level 1	Level 2	Level 3	Level 4	Level 5	Level 1	Level 2	Level 3	Level 4	Level 5
Noise	0.9796	0.9647	0.9555	0.9418	0.9240	0.9919	0.9896	0.9878	0.9840	0.9747
Grey	0.9873	0.9819	0.9695	0.9367	0.9489	0.9919	0.9907	0.9881	0.9748	0.9651
Snow	0.9859	0.9814	0.9703	0.9595	0.9499	0.9910	0.9901	0.9863	0.9797	0.9720
Rain	0.9845	0.9761	0.9595	0.9372	0.8941	0.9898	0.9868	0.9794	0.9670	0.9348
Fog	0.9769	0.9603	0.9330	0.8900	0.8506	0.9900	0.9854	0.9628	0.8992	0.8080
Perspective	0.9550	0.9260	0.8871	0.8183	0.7751	0.9690	0.9422	0.8971	0.8399	0.8038
Blur	0.9822	0.9769	0.9696	0.9606	0.9479	0.9896	0.9867	0.9821	0.9731	0.9582
Rotation	0.9849	0.9784	0.9636	0.9419	0.9213	0.9908	0.9878	0.9731	0.9535	0.9322
Crop	0.9857	0.9607	0.9175	0.8549	0.7995	0.9923	0.9806	0.9383	0.8830	0.8448
Reflection	0.9802	0.9431	0.8894	0.8242	0.7660	0.9871	0.9563	0.9074	0.8528	0.8030

Table 15: AUROC scores of baseline and MC Dropout classifiers. Results are given as a fraction and rounded to the fourth decimal.

	Baseline					MC Dropout				
	Level 1	Level 2	Level 3	Level 4	Level 5	Level 1	Level 2	Level 3	Level 4	Level 5
Noise	0.0119	0.0236	0.0282	0.0399	0.0494	0.0023	0.0037	0.0053	0.0051	0.0113
Grey	0.0057	0.0079	0.0111	0.0189	0.0186	0.0014	0.0021	0.0026	0.0041	0.0122
Snow	0.0055	0.0089	0.0126	0.0193	0.0250	0.0024	0.0028	0.0052	0.0113	0.0116
Rain	0.0095	0.0134	0.0243	0.0389	0.0651	0.0031	0.0056	0.0108	0.0169	0.0399
Fog	0.0121	0.0202	0.0282	0.0555	0.0802	0.0031	0.0066	0.0206	0.0509	0.0663
Perspective	0.0142	0.0127	0.0180	0.0189	0.0184	0.0073	0.0087	0.0117	0.0187	0.0191
Blur	0.0073	0.0101	0.0106	0.0137	0.0174	0.0031	0.0046	0.0051	0.0076	0.0133
Rotation	0.0070	0.0093	0.0099	0.0100	0.0144	0.0029	0.0037	0.0110	0.0134	0.0183
Crop	0.0065	0.0184	0.0277	0.0299	0.0296	0.0018	0.0055	0.0147	0.0206	0.0202
Reflection	0.0080	0.0127	0.0155	0.0247	0.0310	0.0053	0.0138	0.0252	0.0247	0.0222

Table 16: Standard deviations of Table 15.

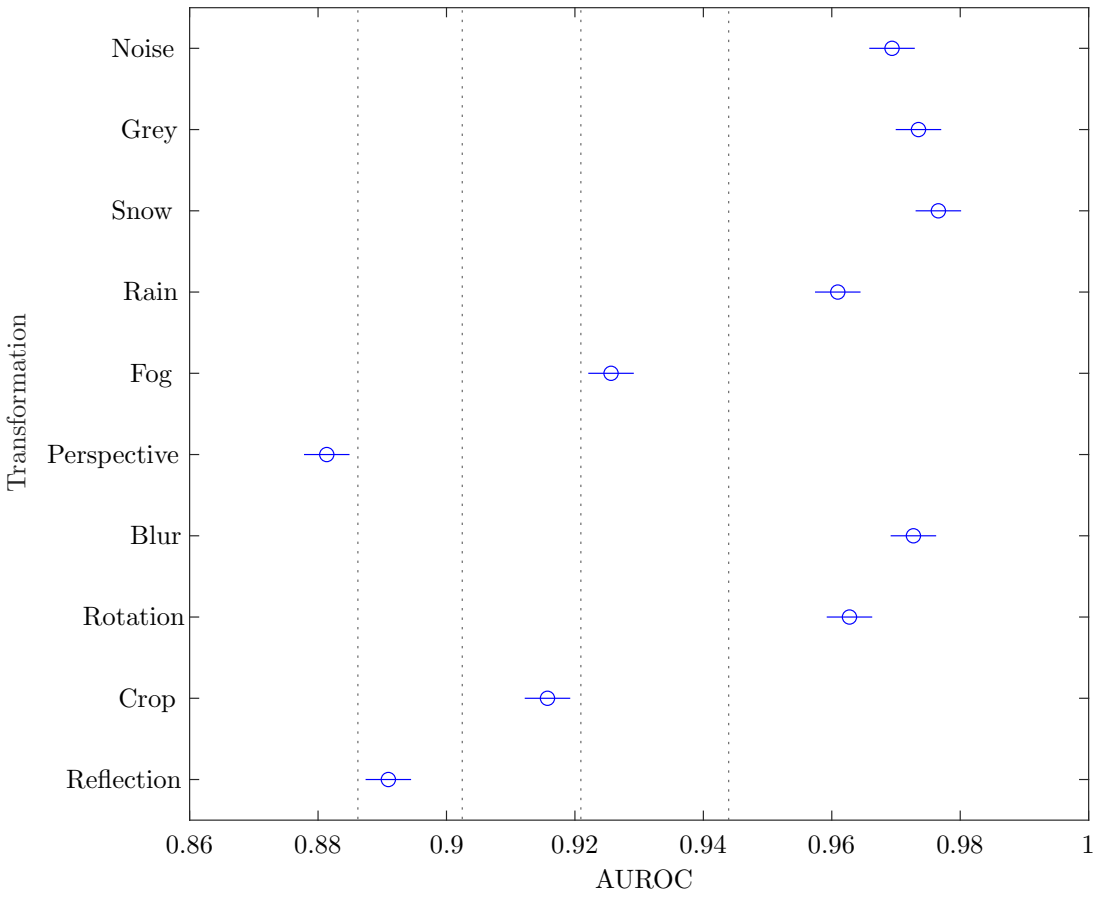


Figure 24: Multiple comparison post hoc analysis of the effects of different transformations on the AUROC. Vertically overlapping error bars for different transformations indicate non-significant differences. Vertical separations between the non-overlapping and significantly different groups have been added.

6.2.3 Brier

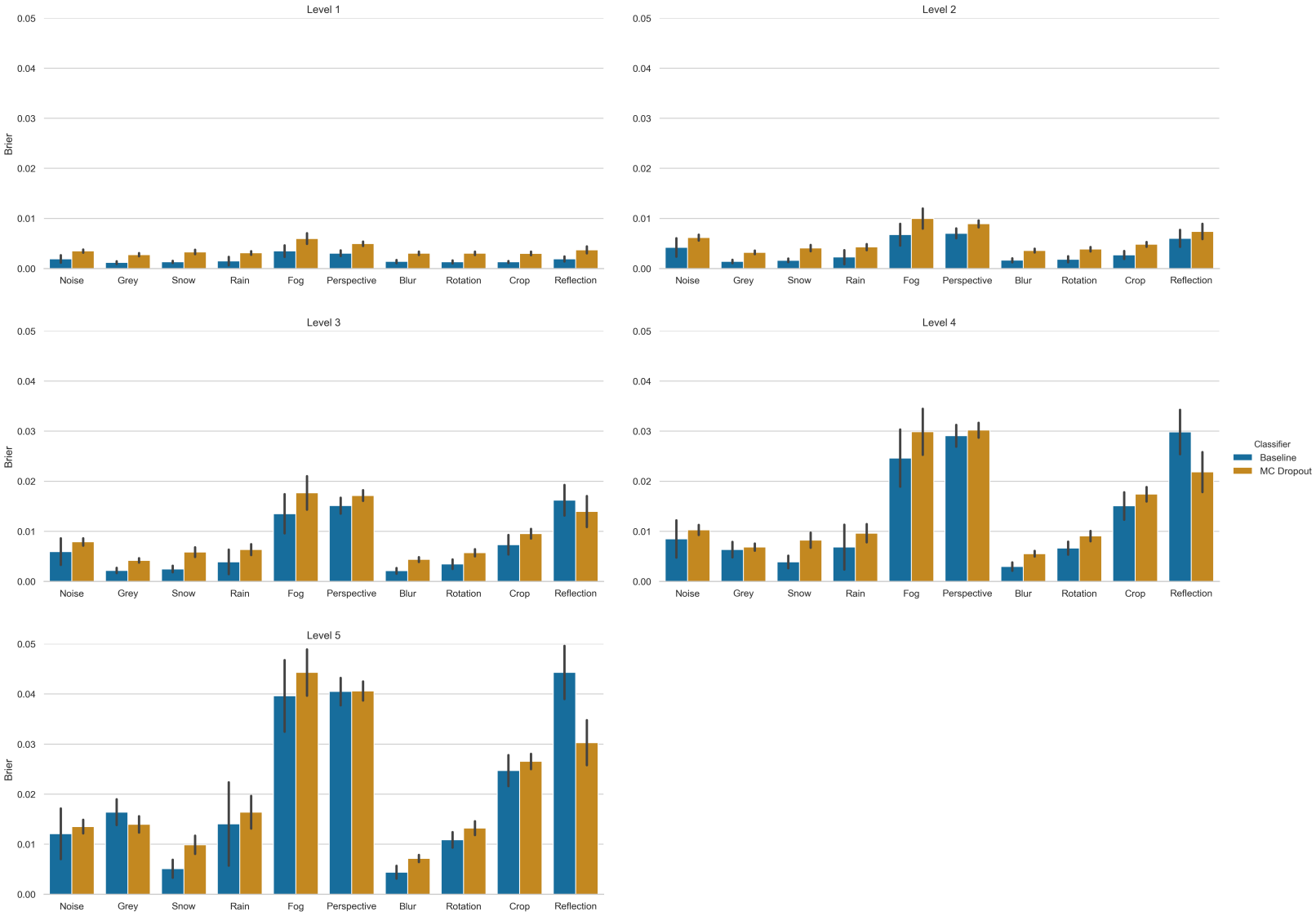


Figure 25: Each individual plot visualizes the results across all transformations for one strength level. Within each plot, the different transformations are shown on the x axis and the Brier score (lower is better) on the y axis. The hue of the bars represent the two classifiers: Baseline (blue) and MC Dropout (orange). The error bars show the standard deviation.

	Baseline					MC Dropout				
	Level 1	Level 2	Level 3	Level 4	Level 5	Level 1	Level 2	Level 3	Level 4	Level 5
Noise	0.0019	0.0042	0.0060	0.0085	0.0121	0.0035	0.0062	0.0079	0.0103	0.0135
Grey	0.0012	0.0014	0.0022	0.0063	0.0164	0.0028	0.0032	0.0042	0.0069	0.0140
Snow	0.0013	0.0016	0.0025	0.0039	0.0051	0.0033	0.0041	0.0059	0.0082	0.0099
Rain	0.0015	0.0023	0.0039	0.0069	0.0141	0.0031	0.0043	0.0064	0.0096	0.0164
Fog	0.0035	0.0068	0.0135	0.0246	0.0397	0.0060	0.0100	0.0177	0.0299	0.0443
Perspective	0.0031	0.0070	0.0151	0.0291	0.0405	0.0050	0.0089	0.0172	0.0302	0.0406
Blur	0.0014	0.0017	0.0021	0.0030	0.0044	0.0031	0.0036	0.0044	0.0055	0.0072
Rotation	0.0013	0.0019	0.0035	0.0066	0.0109	0.0030	0.0039	0.0057	0.0091	0.0132
Crop	0.0013	0.0027	0.0073	0.0151	0.0247	0.0030	0.0048	0.0095	0.0174	0.0265
Reflection	0.0019	0.0060	0.0162	0.0298	0.0443	0.0037	0.0074	0.0139	0.0218	0.0303

Table 17: Brier loss scores of baseline and MC Dropout classifiers. Results are rounded to the fourth decimal.

	Baseline					MC Dropout				
	Level 1	Level 2	Level 3	Level 4	Level 5	Level 1	Level 2	Level 3	Level 4	Level 5
Noise	0.0008	0.0019	0.0027	0.0038	0.0052	0.0003	0.0006	0.0008	0.0010	0.0014
Grey	0.0003	0.0004	0.0005	0.0016	0.0026	0.0003	0.0004	0.0005	0.0007	0.0017
Snow	0.0002	0.0004	0.0007	0.0013	0.0018	0.0004	0.0006	0.0010	0.0015	0.0018
Rain	0.0009	0.0014	0.0025	0.0046	0.0085	0.0004	0.0006	0.0011	0.0019	0.0033
Fog	0.0012	0.0022	0.0040	0.0059	0.0073	0.0011	0.0021	0.0034	0.0047	0.0047
Perspective	0.0006	0.0010	0.0016	0.0022	0.0028	0.0004	0.0007	0.0011	0.0015	0.0019
Blur	0.0003	0.0004	0.0006	0.0009	0.0013	0.0003	0.0004	0.0005	0.0006	0.0007
Rotation	0.0003	0.0006	0.0010	0.0013	0.0016	0.0003	0.0005	0.0007	0.0010	0.0014
Crop	0.0002	0.0008	0.0020	0.0028	0.0032	0.0004	0.0005	0.0010	0.0015	0.0016
Reflection	0.0005	0.0017	0.0031	0.0045	0.0054	0.0007	0.0016	0.0032	0.0041	0.0046

Table 18: Standard deviations of Table 17.

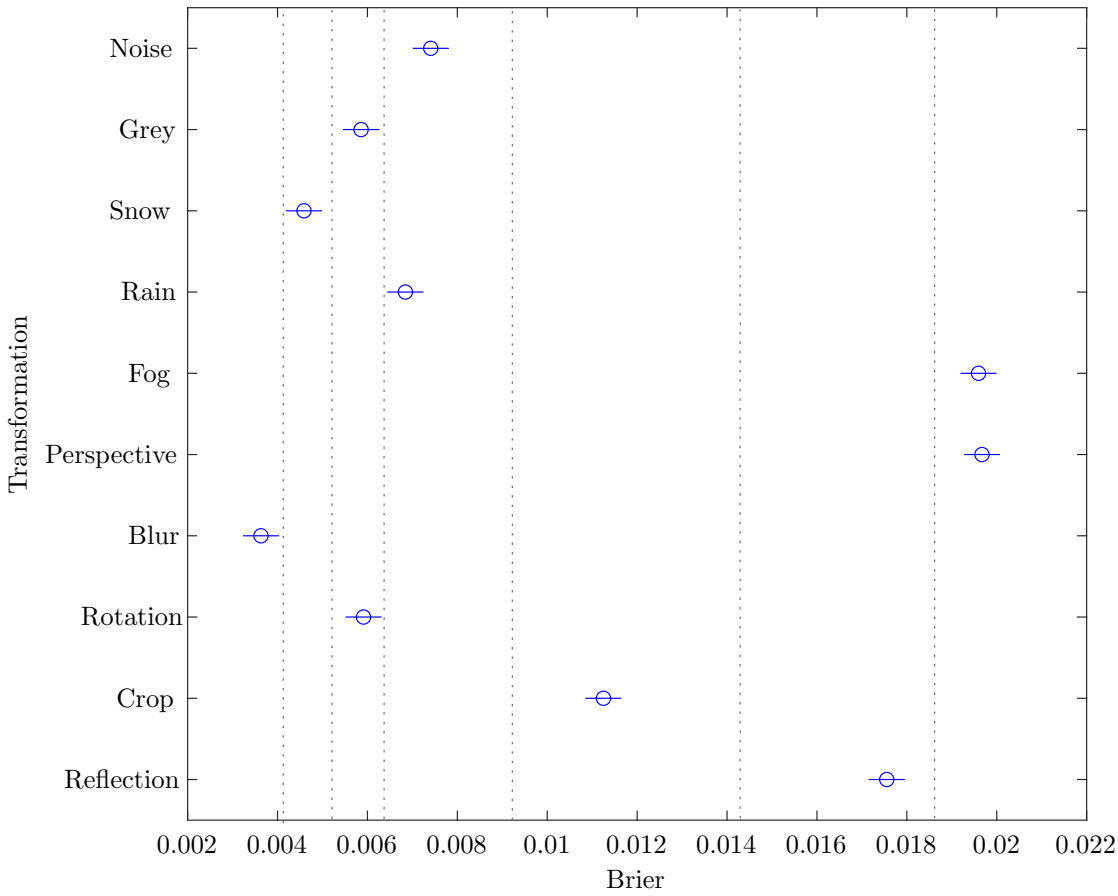


Figure 26: Multiple comparison post hoc analysis of the effects of different transformations on the Brier score. Vertically overlapping error bars for different transformations indicate non-significant differences. Vertical separations between the non-overlapping and significantly different groups have been added.

6.3 OOD evaluation

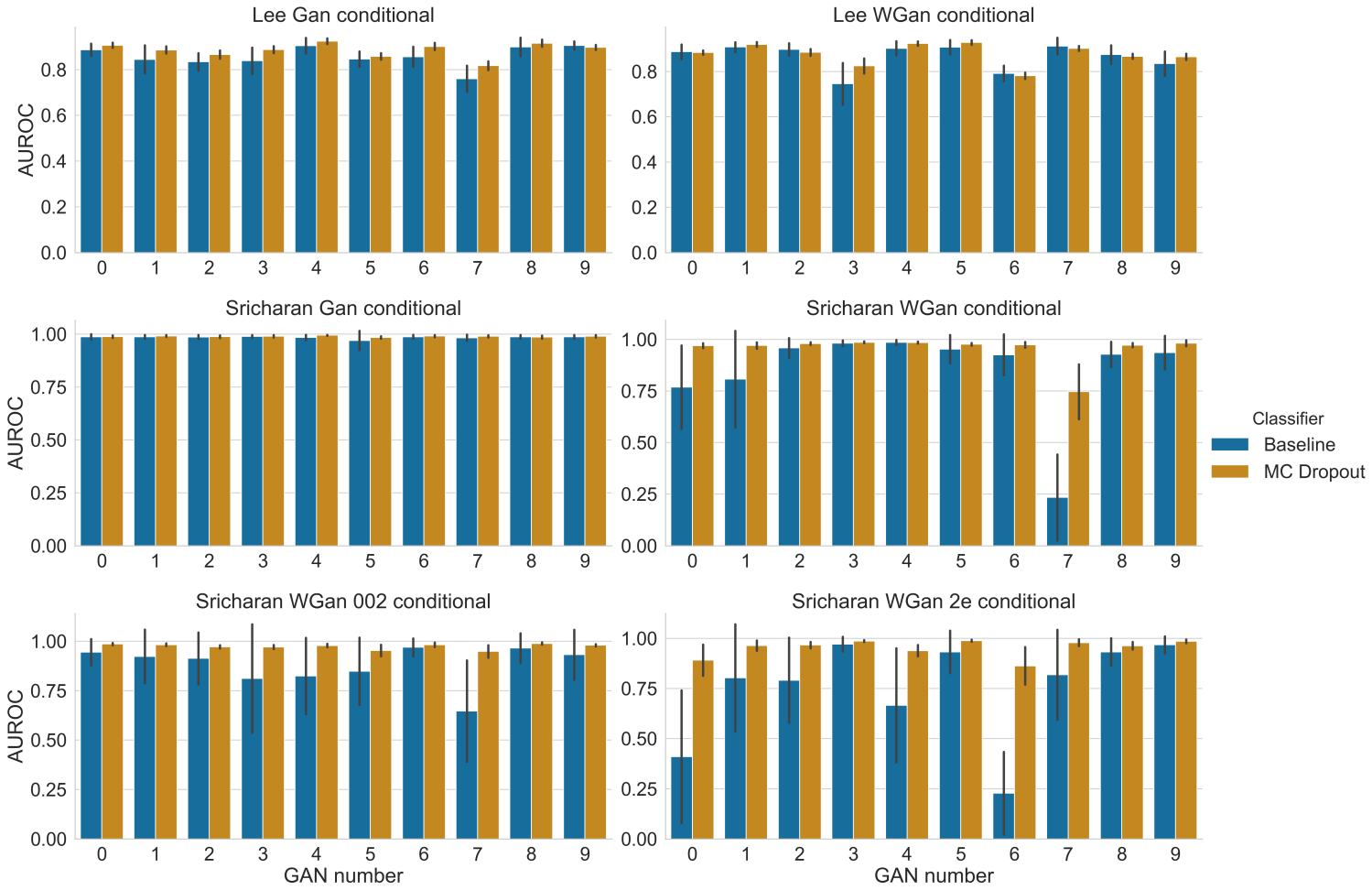


Figure 27: Each individual plot visualizes the results across all trained GANs for one of the six architectures. Within each plot, the individual networks are shown on the x axis and the AUROC on the y axis. The hue of the bars represent the two classifiers: Baseline (blue) and MC Dropout (orange). The error bars show the standard deviation.