CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Using elementary disturbances for testing of machine learning models

A general method for testing of machine learning models based on elementary disturbances: An evaluation with image and audio data

Master's thesis in Computer science and engineering

Arvid Hast

Fredrik Lindevall

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

Master's thesis 2020

Using elementary disturbances for testing of machine learning models

A general method for testing of machine learning models based on elementary disturbances: An evaluation with image and audio data

Arvid Hast, Fredrik Lindevall

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2020

Using elementary disturbances for testing of machine learning models
A general method for testing of machine learning models based on elementary disturbances: An evaluation with image and audio data
Arvid Hast, Fredrik Lindevall

Supervisor: Robert Feldt, Department of Computer Science and Engineering
Examiner: Jennifer Horkoff, Department of Computer Science and Engineering

Master's Thesis 2020
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2020

Using elementary disturbances for testing of machine learning models
A general method for testing of machine learning models based on elementary disturbances: An evaluation with image and audio data
Arvid Hast
Fredrik Lindevall
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

This thesis explores the testing of machine learning models. The problem with current testing methods is that testing often is case-specific and require significant additional effort to perform. A novel method of adding simple elementary disturbances to the input data is used. The method is done in a general way that should work for different kinds of data and different types of machine learning models. The simple disturbances can be used to predict how well a machine learning model handles unseen disturbances. A general testing methodology could be useful as a simple prediction of a machine learning model's resilience to unseen disturbances.

Keywords: Computer science, Software engineering, elementary, disturbance, machine learning, evaluation, testing, classification, image, audio.

Acknowledgements

We want to thank the open-source community for making machine learning accessible to more people. The neural networks used in this thesis have been modified versions of the open-source code. An open-source project also provided the tools used for statistical analysis.

We would also like to thank our supervisor, Robert Feldt, for his experience and input.

Arvid Hast and Fredrik Lindevall, Gothenburg, June 2020

Contents

List of Figures

List of Tables

List of Algorithms

1
Introduction

Different machine learning models are used in a wide range of fields and have become an important part of computer science. Machine learning involves systems that can be improved through experience. Machine learning models can be created and used in many different ways. One of the biggest problems with machine learning compared to conventional, non-machine learning solutions, is that machine learning solutions generally are difficult to test. Evaluation is usually performed by seeing how good a machine learning model is at classifying unseen data. This is data that the tested machine learning model has not had a chance to see beforehand. Generally, it is hard to get more information about what causes errors without doing more complex analysis of the data and the machine learning model tested.

1.1 Background

The traditional way to evaluate machine learning models is through holdout validation. This is when the data is split into different parts. One part of the data is for training. One part of the data is used for validation. The last part of the data that is for testing. The validation and testing parts of the data are not used while training. The validation part is used during training to see how well a solution works on unseen data during the development process, this data is used when optimizing parameters. The testing part of the data is used as a final check to confirm the accuracy of the solution, this is done when the development process is over.

By separating the data into different parts and introducing them step-wise, overfitted results can be avoided. Overfitting is when a solution memorizes rather than learns. This is when the accuracy of the training data increases, but the accuracy of unseen data decreases. It is the accuracy on the unseen data that is important. This gives a better representation of how well a machine learning model will perform outside of the training environment. In practice, this means that a model never trains with the data that is used to evaluate its performance.

In terms of testing or evaluating, no solution is perfect. A proof is only as strong as our trust in the method used for verifying the proof. This is because the verification itself needs to be verified. This verification loop never stops. Therefore there can never be a 100% guarantee that something is correct or not. There can only be a prediction with a varying probability of correctness. At the end of the chain

of verification of the proof, there is usually one single human [6]. Therefore, the proof must be as informative as possible. Instead of having one value describing the performance of the machine learning model, it would be better to have a set of weaknesses. It would be good to know how the machine learning model handles disturbances and what limitations it has. There are tools made to test how well trained machine learning solutions work. Testing tools for machine learning today are, however, mostly task-specific. The tools are generally focused on a specific domain and a specific type of data.

It is hard to do accurate and insightful tests about how well a machine learning model works. It would be optimal if testing could be done fast in an automated way. Currently, the testing methods available are generally case-specific and often require a significant amount of work. Some audio tests in the past have been done through speakers, for example, while adding real-world like disturbances [7]. The tests performed were to test how well their solution worked in different situations, emulating situations that could happen during real usage. Another example is testing of voice recognition with anesthetists (nurses who administer anesthesia) [8]. They tested voice recognition in different working environments to test how viable a voice recognition system would be in the work process.

1.1.1 Research goal

The research goal is to see if a general automated method can be used to test how well machine learning models can resist disturbances in the input data. The main questions to explore are the following:

1. Is it possible to generate elementary disturbances and use them for testing?
    - The method should ideally be able to create elementary disturbances and use them for testing, this would allow for a more dynamic fast and less data-dependent solution.
2. Is it possible to perform testing while handling a machine learning model as a "Black Box"?
    - This means that the method ideally should be able to work without knowing the internal structure of the machine learning model. It should use only the input and output from the machine learning model.
3. Is it possible to create a general solution for evaluating different machine learning models and different types of data?
    - The method should ideally work on a wide array of machine learning models. If this is possible, it will make the method more viable, and it would be easier to compare different models used for the same problem.
4. Is it possible to use simple elementary disturbances to predict the performance on unseen disturbances?
    - The goal is to see if the simple elementary disturbances can be used to predict the performance of a machine learning model subjected to an unseen disturbance. Unseen disturbances are disturbances the machine learning model have not trained on or seen before.

1.1.2 Scope and Delimitations

The methods described are focused on machine learning methods used explicitly for classification of labeled data. This means that the machine learning models used have to be supervised. In other words, the input data has predetermined, correct categories. The goal of a machine learning model is to predict an input category correctly for a data point. In terms of the methods described in this thesis, three levels of testing complexity were identified as significant steps. In order of complexity, the identified levels of testing are:

1. Using one type of data with one type of machine learning model
2. Using multiple types of data with one type of machine learning models
3. Using multiple types of data with multiple types of machine learning models

In this thesis, the second level is reached. For the third, in terms of evaluation, the same tools and methods should work. The third does, however, require a significant increase in computational power required and some additional time for implementation. For the current level of implementation, there is some prioritizations:

- The method is tested on existing open-source resources. Implementation of new machine learning models will not be performed. The machine learning models used are based on existing examples.
- There will be no direct analysis of network architecture. The analysis will be of trained neural network performance. Every machine learning model will be handled as a "black box".
- Computation will be on consumer-grade computers with limited time. Limited computational power means that there is a limitation in what types of networks and what type of data can be used. More straightforward, simple examples are used to test the methods.

2
Related work

In the field of machine learning, a vast amount of papers are released frequently, but seemingly papers on actually testing, studies about the accuracy, and evaluation methods are few. There are plenty of metrics proposed in different studies regarding the testing of machine learning models; examples of these are Accuracy, Error Rate, Sensitivity, Specificity [9]. All these metrics share a common flaw. If the data that the models are tested on is incomplete or is not representative of all the cases encountered in the target system, there is no way of telling how accurate a model will be. This fact is the fundamental problem with metrics used for evaluating machine learning models. If the data is not verified, and if it lacks all the test-cases, the metrics themselves become insufficient and arbitrary. Because of this, a testing methodology and its results has to be as as informative as possible. Instead of having one quantitative test, like testing the accuracy, it would be better to know a set of weaknesses for the tested machine learning model.

The related work in the field of testing can, for the most part, be classified as two groups. One group describes the problem of understanding and overcoming weaknesses in machine learning models. This group could be methods to explore, exploit, or solve the weaknesses in machine learning models. Another group describes different testing methodologies and how to tackle the difficulties of understanding machine learning models. Essentially there is one group exploring weaknesses, like the adversarial examples [3] and [4], seen in Figure 3 and 4. Then there are papers focusing more on understanding of learned behavior, like [2], seen in figure 2.

The typical basic evaluation method when training a supervised machine learning model is to look at the results from holdout validation. This means that the data used is split into two or more different parts, generally one part for training and one or more part for evaluating the performance. During training, the machine learning model uses the training part of the data to improve, increasing the correctness of the model. To make sure the machine learning model works as intended, it is tested on the evaluation part of the data. This is data the machine learning model has not encountered during training. The reason this is done is to test how well the model performs with unseen data.

Often, it is useful to have some additional methods of evaluation when it comes to machine learning models. The currently available methods vary in methodology and are often case-specific. They are created for a specific task, a specific type of data, or a specific type of machine learning model.

2.1 Existing methods of evaluation

Several papers show a sizable difference when it comes to lab performance and real-world performance. Lab performance is the performance achieved when developing the solution. Real-life performance is the performance achieved in the intended use case. The disparity between the two can be so huge that the lab performance becomes nearly useless when it comes to comparing different solutions. In one paper [10], the difference in lab performance and "in the wild" performance is 80%. The paper mainly focuses on malware detection in Android applications and shows the inherent problem of the unknown. Even if an algorithm can get high results in a lab environment, it does not necessarily mean that the performance "in the wild" will be the same.

Recent research shows that the classification ability of deep neural networks (DNNs) can be easily altered by adding relatively small perturbations to the input vector. In a study, when it comes to image recognition, many models can get an incorrect classification after only changing one pixel in the image [11]. Testing this and showing the robustness of a system is therefore very important, especially as machine learning gets implemented in safety-critical systems such as self-driving in cars.

One paper [1] proposes a general solution for understanding classification decisions on images. Each pixel's contribution can be visualized. An example can be seen in Figure 1, a neural network trained on numbers in the MNIST dataset [12]. One comparison made is between the numbers "3" and "8". When looking at the activation for a "3", there a clear negative activation can be seen on the parts of the "3" that could be added to form an "8". This example, shown in Figure 1, is very simple. Visualization can help for smaller, less complex problems. Through mapping the activation weight of the input, there can be some understanding of what a machine learning model has learned, and what possible weaknesses exist. The simple example with numbers is intuitive for a person to examine and understand. When the problem is larger, and there is a higher amount of classifications, a method like this is likely, less helpful, since it would be hard to understand.

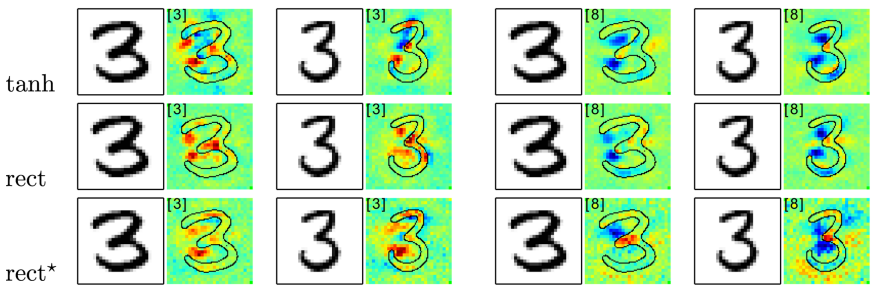


Figure 1: Showing pixel contribution of classification [1]

Visualization of how the input is interpreted can also be done with more sophisticated methods. One example [2] is Image-Specific Class Saliency Visualisation, see Figure 2, for example. These visualizations of trained networks give some understanding of what a network has learned. It does, however, give almost no information about what can go wrong. Why specific changes would alter the output is, in this case, significantly harder to interpret than the previous example.

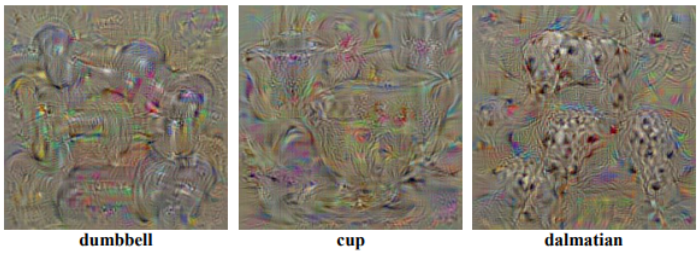
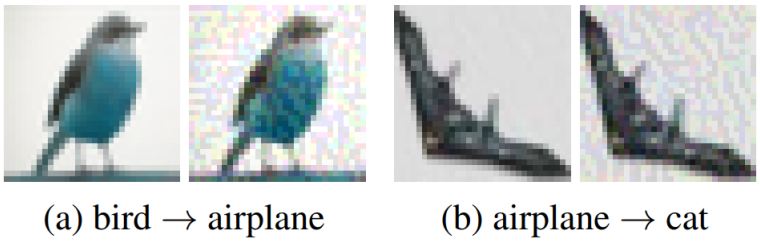


Figure 2: Saliency map examples [2]

Zeiler [13] shows numerous ways of visualizing what a machine learning model does with an input image and an occlusion. Essentially, different parts of an input image are covered to make the output change. This method is the usage of occlusions to test what parts of an image are most important, and what can cause adversarial effects. Partially covering parts of the image gives insight into what parts of the image that are most important for classification and how misclassifications could happen. The illustrations can be useful in understanding why something can go wrong in a classifier. In the examples given by the author, it is clear that for classifications, some parts of an image are more important than others.

There can be tiny changes in input that change the outcome of a neural network [14]. There are some methods for making adversarial attacks. These show that there can be a minimal change in input that gives an unwanted output. Misclassification comes in different types. Specific adversarial attacks might not be realistic; it is, however, a shown problem that minor changes in input can cause an unwanted change in output.

In figure 3, two adversarial examples are shown that can be hard to interpret. The pictures are altered, but there is no intuitive insight into why the classification is wrong. In figure 4, the small difference can be explained. The network interprets the Pong ball to be lower than it is, therefore, tricking the neural network. Different situations will have different levels of comprehensibility for humans. In the second example, Figure 4, the adversarial example is easy to understand, even though the change is smaller than in Figure 3. One way of interpreting this is that generating adversarial examples can give insight in some cases, but can also not be helpful at all in other cases, when trying to understand a certain model.



(a) bird → airplane (b) airplane → cat

Figure 3: Example of adversarial input changing the classification of images [3]

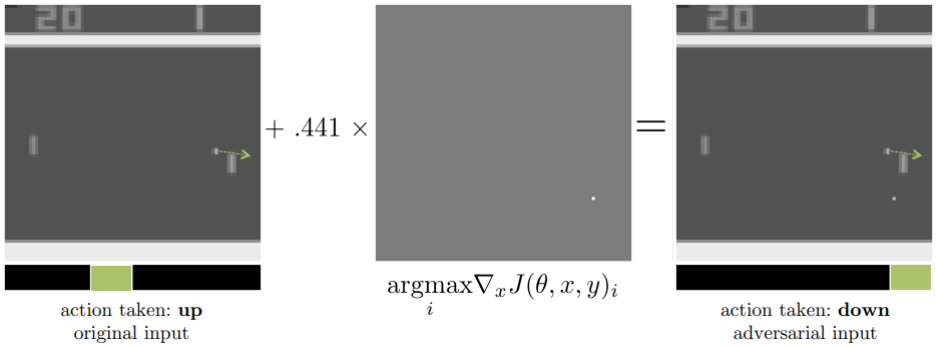


Figure 4: Example of adversarial input affecting Pong game action [4]

It is crucial to find methods to improve reliability and precision in machine learning solutions. Machine learning has an increasing role in the day-to-day life of humans. In some of these cases, like autonomous driving, safety is critical. Deep neural networks can, for example, be used to read traffic signs [15]. If a traffic sign or a traffic light is not recognized correctly, it could put people in a dangerous situation. One paper [16] suggests distillation as a defense to adversarial perturbations. With one deep neural network, the average minimum number of features that had to be modified to get adversarial results, increased by 800%. Solutions like these could be useful to improve the reliability of machine learning solutions.

One paper [17], proposes a test criterion using adequacy metrics. This method calculates a surprise value by comparing new input to training data. The authors claim that "a good test input should be sufficiently but not overtly surprising compared to training data". One test, using additional adversarial examples in training, sampled using surprise adequacy values, showed an improvement in accuracy of up to 77.5%.

There are many papers about generating training data for different supervised learning methods. There is however not as many papers about generating disturbances. However, some work has been done. In the field of grammatical error correction, there have also been attempts to create artificial grammatical errors that could be used for training [18]. Creating data with disturbances can be useful to train supervised learning methods and making them more reliable.

2.2 Large scale testing

Machine learning is a very rapidly growing field of computer science. Every year, there is significant progress both academically, and in commercial products. The companies or entities that are in charge of developing these large complex models generally have resources to create very specific testing tools that can be used to evaluate these systems. Likely, the types of methods described in this thesis do not apply to larger, more complex, machine learning models. It is, however, useful to understand the problems of these larger systems to get a broader understanding of problems in the machine learning field.

Currently, many large companies use machine learning to improve systems and develop new products. For many companies, improving these machine learning models is the main way to increase revenue. In the example of Google, they mention that:

- 40% of app installs on Google Play come from recommendations [19]
- 60% of watch time on YouTube comes from recommendations [19]

In both of these examples, giving better recommendations correlate to higher revenue directly. In services like these, Google is free to test different models and radically change methods of performing recommendations. While there could be sub-optimal suggestions that lose money or inappropriate suggestions, there is no larger safety risk involved. A simpler example of this is the statistical analysis in A/B testing that is widely used among online websites [20].

The company Tesla, a manufacturer of cars, are developing a self-driving system. When developing a self-driving system, there is a larger safety concern. Tesla uses prediction as a way to test new models safely [21]. Without performing any actions, the cars can run in "shadow mode", performing predictions, without doing any action. One example brought up is the example of cars cutting in front of Tesla cars. In this case, they have a neural network to predict when this will happen. In this case, they mention that both false positives and false negatives can be found. A false positive in this case would be a prediction that another car would cut in front but does not. A false negative would be a prediction that another car would not cut in front but does.

Automating the processes when working large datasets is essential. Assuming that the same quality can be achieved, using automated systems to handle the data is optimal. Using humans for manually working with the data costs significantly more than if it can be done automatically. A large part of the processing that has to be done is labeling the input data, so that machine learning models have data to train on. While manual labeling is still used for self-driving systems [21] [22], some parts can be automated.

Often there is a lot of data available, this is useful since more data generally can give better performance. One problem, though, is that all edge cases rarely are covered. There can be rare circumstances that will result in types of data that have not been seen before. Generally, there is a limitation to the data available. It will rarely contain all the possible edge cases. The problem with this is that the edge cases often can be the most interesting and important part of the data, as previously discussed. Processing and improving these edge cases is a large part of what has to be solved to improve machine learning models.

In the example of a self-driving car system, there could be millions of kilometers of peoples' driving data available. The majority of this data will be a car driving in one lane without many changes. A smaller, but still significant, part of the data would be more advanced maneuvers, like changing lanes, stopping at traffic lights. The smallest, but potentially most important type of data is the rare type, like accidents, people losing control of the car, and wild animal encounters. For example, Tesla has a trigger infrastructure [21] to strategically lets them pick the specific type of data that they want. This type of problem exists for all datasets. There will almost always be types of data that are significantly more infrequent.

3

Methodology

The overall goal is to create simple elementary disturbances that can be used to evaluate machine learning models. Disturbances are applied to the input data. The new, modified data is used to test the machine learning models. Testing a machine learning model with elementary, simple disturbances could give some insight into how well a machine learning would perform with unseen disturbances. The goal is to see if these simple disturbances can be used to predict how well a machine learning model performs with unseen disturbances. This chapter describes the creation and usage of these simple disturbances.

The goal is a general automatic way of evaluating how well a machine learning model is at handling unknown disturbances. The method should ideally be a more effective and accurate way to evaluate machine learning models with unknown disturbances than basic holdout validation, and less time consuming than manual evaluation and analysis. Testing how well a machine learning model would perform with unknown disturbances is difficult; basic holdout validation can only be used to evaluate known data. The objective is to test if the results from the simple disturbances can be a better predictor for unseen disturbances than holdout validation.

The evaluation method should work on all kinds of data, and work on all machine learning model that use labeled data. This removes the need for specialized methods and makes the workflow more straightforward and more effective. It should also require minimal setup before testing various machine learning models. This is because the method is meant to make it easier to evaluate the stability of different machine learning models against unknown disturbances. If the setup process is too complicated or difficult, it will limit the value of the proposed method compared to more traditional case-specific analysis.

The method needs to account for the fact that different types of disturbances might have different effects on different types of data. Evaluating the stability with as many automatically generated disturbances as possible might, therefore, be beneficial to get a more comprehensive overview of how well a machine learning model will handle unknown disturbances.

3.1 Summary

The method works by creating simple disturbances that are applied to data. Machine learning models can be tested with the modified data, to see how well they handle disturbances. Different levels of each disturbance are tested, to find at which point the disturbance causes the machine learning model to misclassify the data. The level for each disturbance is saved. The data saved can be looked at manually to see what type of disturbances alter the classifications the most.

In the method, the data is generalized, making it possible to use different kinds of input data. This enables the method to have the same elementary disturbances no matter the input. The data could be everything from single variables to multidimensional data, such as images and audio.

3.2 Prerequisites to use method

For the proposed method to work, there are a few criteria that have to be fulfilled. These are required to be compatible with the proposed general testing method of a machine learning model.

1. Input data

The input data can be in virtually any format. The automatic elementary disturbances used for testing are independent of the input data structure. This is further explained in 3.3.1.

2. Method for classification prediction

This means that regardless of the classification method used, there should be a method available for giving a classification prediction with new data. There should be an available way to get the machine learning model to classify new data. For example, a model used for image classification should have an available method to classify new images.

3. Intended classifications

These are the correct classifications (labels) for the sample data. The correct classifications should be available for each data point. If, for example, the dataset is of images, there should be a list with what the correct classification is for each image.

Assuming these three prerequisites are fulfilled, the method should be compatible with any machine learning model for classification and for any type of data.

3.3 Automatic elementary testing methodology

3.3.1 Methodology overview

The methodology is fairly complex and the flow is circular, therefore, it is hard to represent. But as a simplified version the following steps are done. A visual representation of the method can be seen in Figure 5.

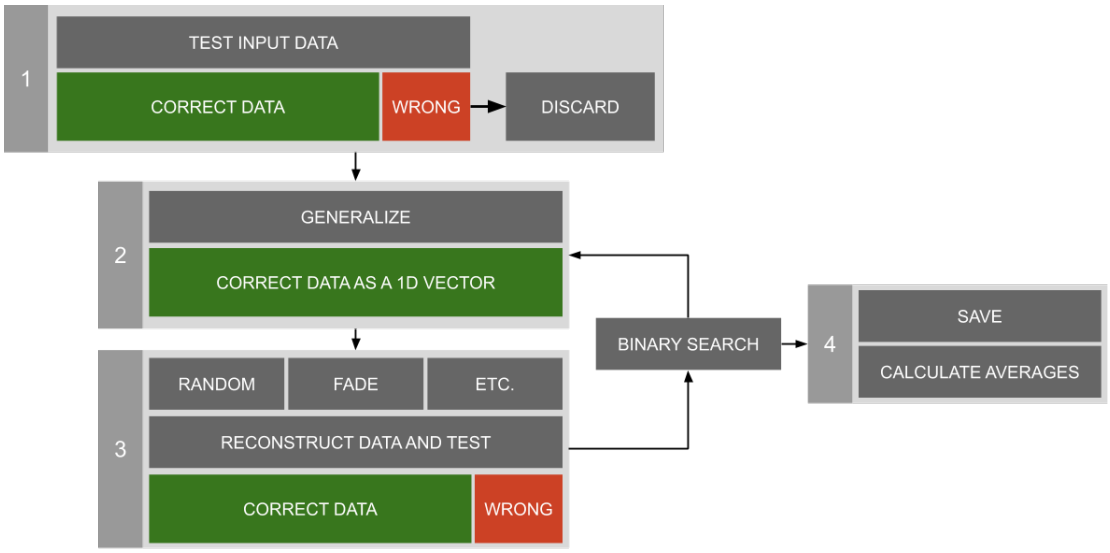


Figure 5: Simplified overview of the evaluation method that evaluates the proposed method. The evaluation method is divided into four stages.

3.3.1.1 1: Discarding misclassified classified data

The first step is to test the data. The data that is correctly classified by the machine learning model will be used in the next steps. The misclassified data will be discarded. This is because the method works by finding the breaking points of the machine learning models. The breaking point here refers to the point where a disturbance causes data to be misclassified.

Since misclassified data is wrong from the beginning, the level of disturbance needed to get a misclassification is zero. Therefore the data that is misclassified without disturbances is not used. After the misclassified data is discarded, the method proceeds to the next step.

3.3.1.2 2: Generalize

In this step, the data is generalized by turning it into a one-dimensional vector. This enables the elementary disturbances to work on any kind of data that can be represented as a vector. The input structure is also saved so the data can be reconstructed after the disturbances are applied. For further explanation of the generalization process, read section 3.3.3.

3.3.1.3 3: Find breaking points

The third step is to find the level of disturbance that is required to change the classification from correct to wrong. Disturbances are added to the generalized data. This is done at different levels to find where the limit is. For each different level, the data is reconstructed and input to the machine learning model, to see if the classification is changed. To efficiently find the limit, binary search is used. This is further explained in section 3.3.5. When the breaking point is found, the results are saved. This process is repeated for each data point and each disturbance.

A list of elementary disturbance implemented seen in section 3.3.6.

3.3.1.4 4: Save results and calculate averages

The saved data from the previous step is the input data, output data (data after added disturbance), original classification, new classification, and the disturbance level for each disturbance. After the testing is done, the method calculates the average for each automatic elementary disturbance that was needed to make the correctly classified input to be misclassified. These values are later used to assess the stability of unknown disturbances. A more in-depth explanation can be found in section 3.5.

3.3.2 Data

For testing, three datasets are used, two datasets of images and one dataset of audio. The datasets used are:

- MNIST [12]
- CIFAR-10 [5]
- URBANSOUND8K [23]

Datasets are collections of data that are used for the machine learning models. The MNIST and CIFAR-10 datasets are of small images. The URBANSOUND8K dataset is of short audio snippets.

The visualizations in this paper will be of images for the sake of making it easier to understand. The visualizations are with 28x28 pixel images from the MNIST database of hand-drawn digits [12], and 32x32 pixel images from the CIFAR-10 dataset [5]. The image below, Figure 6 is a spectrogram [24] of a sound effect that comes with a Windows installation, celled "Speech Disambiguation.wav". It is hard to visualize sound in a meaningful way. A spectrogram is arguable one of the better ways to visualize sound since it can be shown as a 2D image. This is why images are used instead.

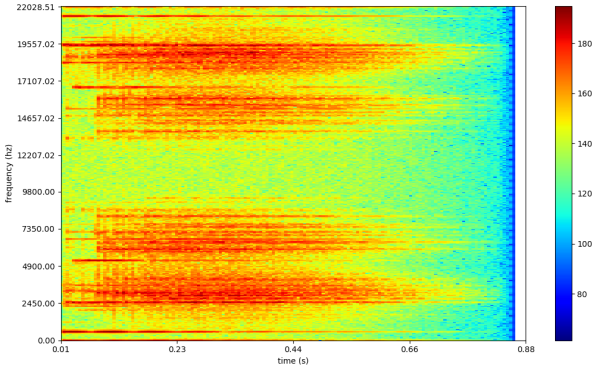


Figure 6: Example of a spectogram representation of an audio file.

3.3.3 Generalization of data

Data can be represented in many different ways. It can be in any number of dimensions. Data could be one single number or a video consisting of millions of individual numbers in a four-dimensional structure. To handle different types of data structures, the data needs to be generalized. This is done so that data can be used the same way, independently of what type of data is used. To generalize data, the data is converted to a one-dimensional vector. In a sense, the data is flattened. In the example of an image from the MNIST dataset [12], the input dimensions would be 28x28x1 (the image is 28x28 pixels, and the pixels are in greyscale). In this example, the input is flattened to the dimensions 784x1 to perform the different disturbances. The dimensions of the original input are saved so that the input can be restored to its initial structure. The downside with this approach is the fact that the method only can see the data in an unstructured form. This means that it will not be able to create complex disturbances spanning more than one dimension; it can just alter each value independently without knowledge of the original data structure.

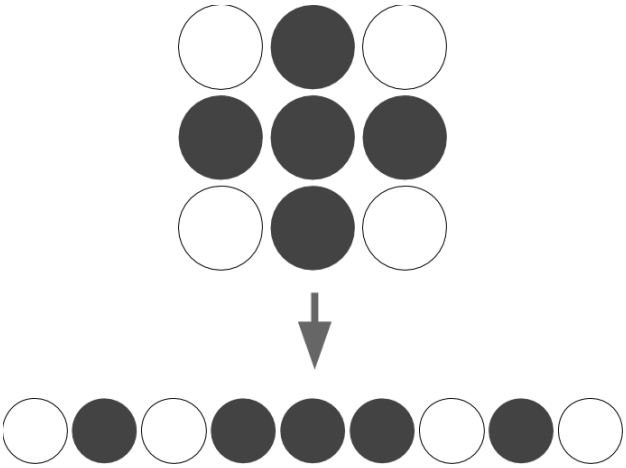


Figure 7: Visual representation of a 3x3x1 matrix being converted into a 9x1 vector

3.3.4 Normalization

The input is normalized so that each input element is between the values 0 and 1. Values are normalized so there can be a general way of handle data. The input is normalized such that $0 \leq i_n \leq 1$ for all n in i. Where i is the input vector, and n is the index of an element in the input vector. As previously mentioned, the data is reshaped to be one dimensional, such that disturbances can easily be applied independently of the input. This means that the final processed input is a one-dimensional vector that has elements with values between 0 and 1.

3.3.5 Adding disturbances

The method works by finding a breaking point of the classification. For example, if you have an image, how much disturbance can you add before the picture no longer get the correct classification. By doing this, the method can find out how much of a certain disturbance the machine learning model can handle before it breaks.

To find out how much disturbance can be added before the classification changes, the limit has to be found. An efficient method to find the limit is to use binary search [25], rather than gradually increasing the disturbance level.

For each step of the binary search algorithm, the uncertainty is halved. If the start is at 50% disturbance, for example, it can be determined if the breaking point is above or below 50%. If it is under 50%, the next search step will be at 25%, determining if the breaking point is between 0-25% or 25%-50% and so on. When performing this cycle ten times, the uncertainty drops to below 0.1%. This process substantially speeds up computation compared to gradually increasing the disturbance level to find the limit.

The different disturbances all use similar code. The process of testing every disturbance is very similar; the only difference is the disturbances themselves, which have slightly different implementations. The tests refer to finding the breaking point of each disturbance for each data point tested. This is the point where the classification changes from the initially correct classification to a new wrong classification. The pseudo-code of the testing process of a disturbance is shown below in Algorithm 1. Simplified versions of each disturbance can be seen in the appendix, Algorithm 2-9.

```
save original data structure ;
generalize (flatten) input ;
initialize output disturbance level = 0 ;
initialize output new classification = original classification ;
initialize output data with disturbance = original data ;
begin binary search;
while binary search precision ≤ 0.1% do
    dataWithDisturbance ← addDisturbance(flattened input, disturbance level) ;
    change dataWithDisturbance to original data structure, fitting the model ;
    modelClassificationPrediction ← getModelPrediction(dataWithDisturbance) ;
    if modelClassificationPrediction ≠ original classification then
        save disturbance level into output (override) ;
        save new classification into output (override) ;
        save data with added disturbance into output, using the original saved
          data structure (override);
    end
    update disturbance level in correct direction for higher precision ;
    continue binary search ;
end
```

Algorithm 1: Simplified version of algorithm for finding a disturbance limit

3.3.6 Disturbance types

The types of disturbances chosen for testing have a few common features. They are chosen and designed in such a way that they should be usable for any type of data. Since the data is generalized before disturbances are added (as described in 3.3.3), disturbances have to work with this type of flattened data. This means that there is no existing knowledge of the structure of the data, which means that there are no created disturbances that are specific to any type of data. The disturbances are created in such a way that they work with the flattened, generalized data structure (see Figure 7, for example).

The disturbances are simple modifications. The modifications are elementary, modifying the smallest building blocks of the data. These smallest building blocks are the individual elements of data, for example, the red, green, and blue colors of a pixel.

The automatic elementary disturbances tested are:

- Setting random element to a random value
- Setting random elements to 1
- Setting random elements to 0
- Fading between two data points
- Exponential mapping decrease of elements (black shift)
- Exponential mapping increase of elements (white shift)

Disturbances strictly for analysis are also made:

- Taking max elements from two data points (combination)
- Taking min elements from two data points (common features)

The examples will be on simpler images for the sake of demonstration. The methods should work for any type of data. The more detailed procedure of testing is described for each disturbance below. Additionally, pseudocode for each disturbance is in Algorithms 2-9, in the Appendix.

3.3.6.1 Setting random element to a random value

This disturbance sets a random input to a random value between 0 and 1. $x_r = d$ where $0 \leq d < 1$ and d is a uniform distribution random function. r is a random number identifying the element in the input vector ($r \in \mathbb{N}, 0 \leq r < N$), N is the amount of elements in the input vector. Binary search is performed to find at what level of disturbance the machine learning model misclassifies the original data. Conceptually this could, for example, mean that a pixel in an image becomes a different color or something similar. The effect of this can be seen in Figure 8. In the example, random pixels are altered into different random values of grey until the machine learning algorithm misclassifies the data.

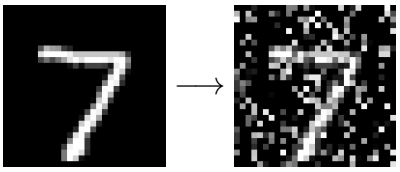


Figure 8: Example of an random disturbance being applied to an image.

Setting random elements of an input vector to a random value can represent multiple different things. It could be inaccuracy in measurements, lossy compression, decay, or other loss of data accuracy.

3.3.6.2 Setting random elements to 1

$x_r = 1$ where r is a random number identifying the element in the input vector ($r \in \mathbb{N}, 0 \leq r < N$), N is the amount of elements in the input vector. This disturbance sets a random input to 1. Binary search is performed to find at what level of disturbance the machine learning model misclassifies the original data. Setting random elements of an input vector to 1 can represent several disturbances. One example is audio, where a phenomenon known as white noise can occur. White noise is where all frequencies are at similar intensity. The effect of this can be seen in Figure 9. In the example, random pixels get set to white until the machine learning algorithm misclassifies the input.

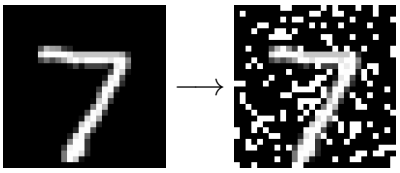


Figure 9: Example of an element setting disturbance being applied to an image.

Depending on the structure of data, setting elements to 1 could have the same effect as setting them to 0 if the representation is reversed. This means that setting to 1 and setting to 0 could be interchangeable, depending on the circumstance.

3.3.6.3 Setting random elements to 0

$x_r = 0$ where r is a random number identifying the element in the input vector ($r \in \mathbb{N}, 0 \leq r < N$). This disturbance sets a random input to 0. Binary search is performed to find at what level of disturbance the machine learning model misclassifies the original data. Conceptually this could, for example, mean in the removal of data, such as missing pixels in an image, gaps in an audio file, or stutter in a video. The effect of this can be seen in Figure 10. In the example, random pixels get set to black until the machine learning algorithm misclassifies the input.

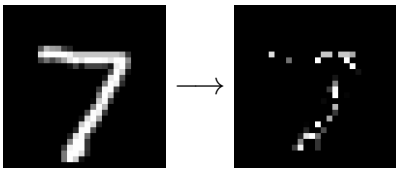


Figure 10: Example of of an element unsetting disturbance being applied to an image.

Setting random elements in the input vector to 0 represents the removal of random data. Depending on the structure of data, setting elements to 0 could have the same

effect as setting them to 1, if the representation is reversed. This means that setting to 0 and setting to 1 could be interchangeable, depending on the circumstance.

3.3.6.4 Fade between two datapoints

This disturbance fades between two classifications.

$\forall n(n \in \mathbb{N}, 0 \leq n < N) : x_n = i_{1n} * (1 - f) + i_{2n} * f$ where x is the output vector, i_1 and i_2 are different entries from the data-set. f is the fade, 0 means that $x = i_1$, and 1 that $x = i_2$. n is the index for each vector element, for example a pixel. N is the amount of elements in the input vector. Binary search is performed to find at what level of fade (f) the machine learning model misclassfies the original data, or is cancelled when $x = i_2$. Conceptually this could, for example, mean when two sounds are picked up simultaneously, or when a transparent object is placed in-front of something. The effect of this can be seen in Figure 11.

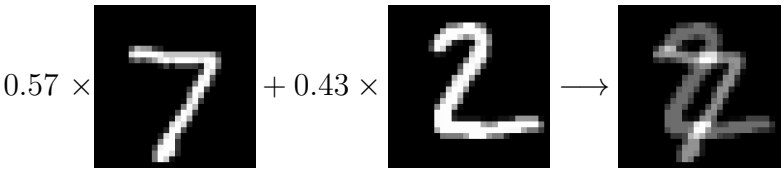


Figure 11: Example of a fade disturbance being applied to an image.

In this example, there is a 43% fade from the 7 to the 2.

3.3.7 Exponential mapping decrease of elements (black shift)

This disturbance is similar to changing the white balance in an image or video. The disturbance maps element values on an exponential curve towards 0. Each input element, x_n, in the input vector, x, has a value y_n in the output vector y. The value is shifted according to $y_n = x_n^z$ where $z \leq \mathbb{R}, 0 \leq z < 1$). Binary search is performed to find at what exponent (z) the machine learning model misclassifies the original data. This is a disturbance that is hard to explain conceptually for data other than images, and a disturbance like this would most likely not occur in any real-world situation. The reason for the implementation of these kinds of disturbances is to see if there is any value in more obscure alterations. This disturbance would, in essence, minimize the difference between values rapidly until they slowly all become 0, making more extreme values less prevalent. An example of this can be seen in Figure 13.

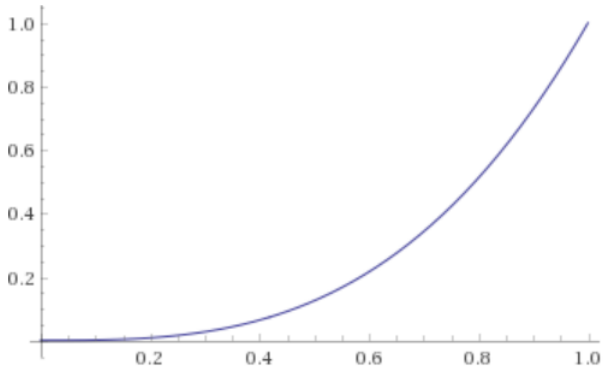


Figure 12: Example of an exponential mapping function, $y = x^3$.

Each value the x-axis corresponds to a new value on the y-axis that is the elements' new value. For instance, a black and white image might have 0 as a black pixel and 1 as a white pixel. In this case, the pixels would get a darker shade. This is because all the values in the red green and blue channels are changed according to, for example, $y_n = x_n^3$.

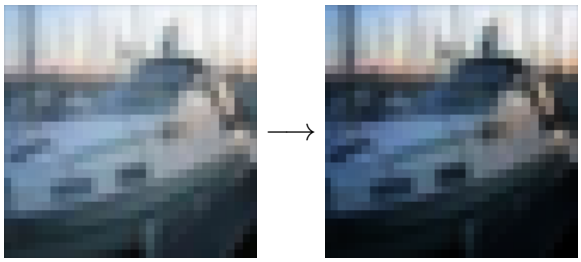


Figure 13: Example of an image from the CIFAR-10 dataset [5] being shifted towards a darker shade.

3.3.8 Exponential mapping increase of elements (white shift)

This disturbance is similar to changing the white balance in an image or video. The disturbance maps element values on an exponential curve towards 1. Each input element, x_n, in the input vector, x, has a value y_n in the output vector y. The value is shifted according to $y_n = x_n^z$ where $z \leq \mathbb{R}, 1 \leq z < 100$). Binary search is performed to find at what exponent (z) the machine learning model misclassifies the original data. This is a disturbance that is hard to explain conceptually for data other than images, and a disturbance like this would most likely not occur in any real-world situation. The reason for the implementation of these kinds of disturbances is to see if there is any value in more obscure alterations. This disturbance would, in essence, minimize the difference between values rapidly until they slowly all become 1, making more extreme values less prevalent. An example of this can be seen in Figure 15.

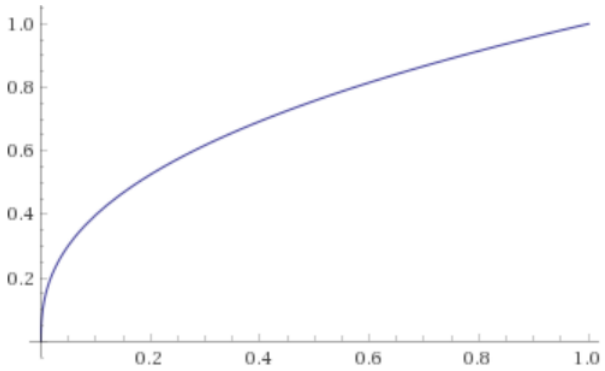


Figure 14: Example of an exponential mapping function, $y = x^{0.4}$.

Each value the x-axis corresponds to a new value on the y-axis that is the elements' new value. For instance, a black and white image might have 0 as a black pixel and 1 as a white pixel. In this case, the pixels would get a lighter shade. This is because all the values in the red green and blue channels are changed according to, for example, $y_n = x_n^{0.4}$.

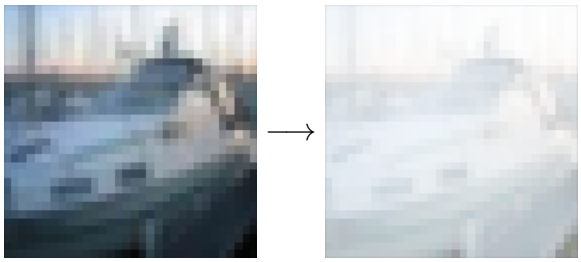


Figure 15: Example of an image from the CIFAR-10 dataset [5] being shifted towards a lighter shade.

3.3.9 Disturbances strictly for manual analysis

The disturbances previously described have different levels, for example, 50% random elements. In these two following cases, max and min, there is no level, it is merely a disturbance performed. Results from these are stored for manual review but not used for the statistical analysis, since there is no variable value, as the disturbance level for the other disturbances.

3.3.9.1 Take max elements from two data points

$\forall n(n \in \mathbb{N}, 0 \leq n < N) : x_n = max(i_{1n} + i_{2n})$ where x is the output vector, i_1 and i_2 are two input images, n is the index for each vector element, for example a pixel. N is the amount of element in the input vector.

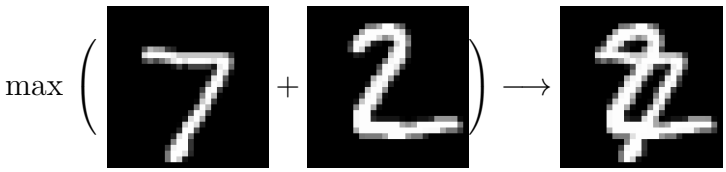


Figure 16: Example of combining two images with the max operator, elementwise

This disturbance could be used to analysis overlapping data, such as audio or energy levels It could be argued that this would not generally apply to image classification since combinations are hard to do logically, for most situations.

3.3.9.2 Take min elements from two data points

$\forall n(n \in \mathbb{N}, 0 \leq n < N) : x_n = min(i_{1n} + i_{2n})$ where x is the output vector, i_1 and i_2 are two input images, n is the index for each vector element, for example a pixel. N is the amount of element in the input vector.

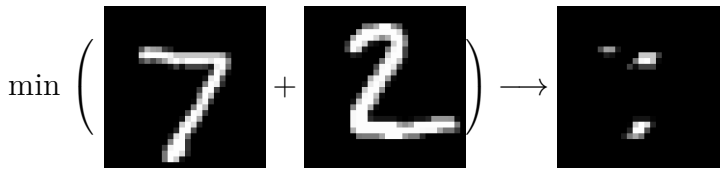


Figure 17: Example of combining two images with the min operator, elementwise

This disturbance could be used for finding common features in data. For example, it could identify common features for sounds. Running this output through a machine learning model might not be useful. It could, however, give insight into similarities in data and how potential errors could arise in machine learning models.

3.4 Implementation

A small amount of work has to be done for each machine learning model to run the evaluation. The analysis tools are not specific to any specific machine learning model or structure of data. In theory, it should be simple to adapt to most implementations. If the three prerequisites previously mentioned are fulfilled, see section 3.2, it should be possible to implement with most types of machine learning models and types of data. The method described is implemented in the programming language Python. Implementation of machine learning models should be in Python to fit with the current implementation. The same method could, however, be implemented in other languages as well. The following, Figure 18 is an example of an implementation done. In this example, the prerequisites are fulfilled. The function "networkPrediction" is the method that returns the classification prediction from this particular machine learning model. This fulfills the second requirement of having a "Method for classification prediction". The "dataPoints" and "classifications" fulfill the first and third requirements, i.e., the sample data and intended classifications, respectively.

```
# Example of network specific classification function
def networkPrediction(dataPoint):
    netOutput = conv_net(dataPoint)
    return np.argmax(netOutput.numpy()[0])

# Import the python code for the analysis
from networkAnalyzer import exampleAnalysis

# Run analysis (and give 5 examples)
exampleAnalysis(dataPoints, classifications, networkPrediction, 5)
```

Figure 18: Example of code added to a machine learning model, to run the analysis.

3.5 Saving data for interprentation

As described in the method overview methodology (3.3.1.4), the input data, output data (data after added disturbance), correct initial classification, new classification, and the disturbance level are saved for each disturbance of each data point analyzed. The total average level for when the model breaks for each disturbance is also saved in a CSV file for easy statistical analysis. This data can be used for both statistical analysis and manual interpretation.

For analysis, a folder structure is created to enable exploration of all the modified data, see Figure 19. One folder is created for each possible disturbance. In each of these disturbance folders, one folder for each possible classification is added to represent the correct original classification. In each folder of each possible classification, additional folders for each classification are created for the new classifications. After each disturbance test, a file is saved with the output data. The breaking point for the disturbance level is added to the file name, so the user can have a quick overview in their file system. For example, there could be an image originally classified as a 0. After a fade disturbance (described in 3.3.6.4), the new classification could be 2. In that case, an image file would be added with the output image in the directory *output/fade/originallyClassification_0/now_2*. The data that is wrong initially can also be saved this way, in case the user wants to see how the original data gets misclassified, see Table A.1 for an example of this.

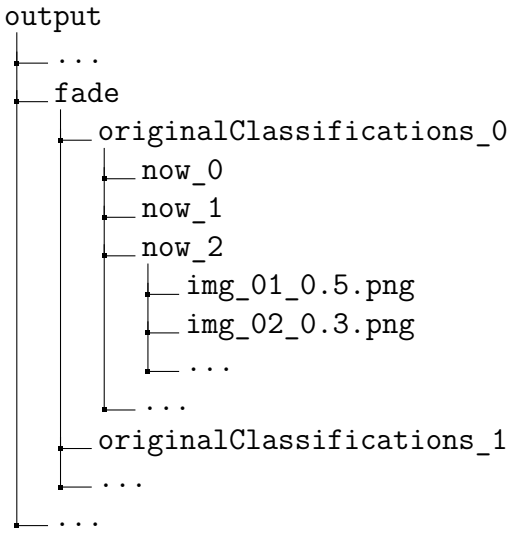


Figure 19: Example of how a generated file structure could look

An upper limit to the amount of data saved can be specified to avoid saving an unwanted amount of data. To avoid long file paths, naming simplifications of disturbances is performed according to Table 3.1 below.

Table 3.1: Simplified names of elementary disturbances

Disturbance	Simplified name
Setting random element to a random value	random
Setting random elements to 1	randomSet
Setting random elements to 0	randomUnset
Fade between two datapoints	fade
Exponential mapping decrease of elements	blackShift
Exponential mapping increase of elements	whiteShift
Take max elements from two data points	max
Take min elements from two data points	min

3.6 Specific methodology

Some data require special processing to work ideally. In the cases explored, only the audio needed some extra processing before it could be used. This processing is not done for the proposed method. Instead, it is done for the machine learning models. Often data is simplified to work with existing machine learning models, this is to make the data more easily interpreted for a computer, and this, in turn, results in more accurate and effective models. This can be done done in, for example, audio, video, and text.

3.6.1 Audio feature extraction

Audio files are generally rather difficult to work with without doing some feature extraction. The reason for this is because the audio format is difficult to represent in 1 or 2 dimensions. To tackle this problem and making audio into a format that is more easily used in machine learning, the audio can be converted into an understandable format using feature extraction. It is a process that converts the audio into feature values that try to explain the data in a more simplistic way while losing as little information in the process as possible. This also has the added benefit of minimizing the size of the data making the training and evaluation process of the machine learning models significantly faster.

There are several ways to do feature extraction. One way is to use Mel-frequency cepstral coefficients (MFCC). The MFCC collectively make up the Mel-frequency cepstral (MFC) and are derived from a type of cepstral representation of the audio file (a nonlinear "spectrum-of-a-spectrum"). MFCC's have been shown to have good performance when identifying audio [26]. MFCC is used when handling audio in the machine learning models.

4
Evaluation Methodology

Getting the proposed method to work with multiple different machine learning models is relatively simple. It is, however, harder to evaluate if the proposed method is useful for testing of machine learning models. Getting information about the simple general elementary disturbances is a simple process. Seeing if the information holds any significant value is much more complicated and requires a more thorough evaluation. It can not be assumed that the simple elementary disturbances correlate with more complex disturbances that occur in the real world. To evaluate the method, tests have to be made to see if the simple automatic disturbances can be used to predict the resilience of unknown disturbances in machine learning models. This chapter describes the evaluation methodology that is used to evaluate the prediction ability. This includes the more advanced disturbances as well as the process of performing statistical analysis. The statistical analysis is made to explore the prediction ability of the simple disturbances. The statistical analysis consists of two main parts, correlation analysis, and linear regression models.

To see if the proposed method works, data gathered from the automatic testing with simple disturbances is compared to unknown disturbances. For this comparison, more advanced disturbances are created. They are more advanced in the sense that modifications are not strictly on elementary components. Additionally, the disturbances are data type-specific, trying to emulate realistic disturbances.

4.1 Choosing advanced disturbances

The more advanced should be:

1. Without an inherent correlation to the simple elementary disturbances.
2. Data dependent, meaning that the advanced disturbances should be made for the particular data structure.
3. The modifications should not be on elementary components, and the changes should be done on multiple components simultaneously.

The goal is to see if the simple disturbances can be used to predict the performance of a machine learning model subjected to an unseen disturbance. For an accurate evaluation, the new, more advanced disturbances should have no implicit correlation with the simple disturbances. If the new disturbances are similar to the simple disturbances, the testing is not useful for predicting unseen disturbances, since there

is an inherent correlation. This would essentially just prove that the simple automatic tests can predict how stable a model is against similar disturbances. It would not show how stable the model is against unknown disturbances. It is, therefore, important that the proposed method is evaluated on data with disturbances that are not similar to the simple elementary disturbances described in the methodology section.

The advanced disturbances should be data-dependent. This is to make sure that they represent realistic disturbances to a higher degree. Having as realistic disturbances as possible is important to get a result that actually represents the performance outside of the lab environment. The elementary disturbances are specifically data-independent, and therefore the inherent correlation between the advanced disturbances and the simple disturbances will be minimized.

Modification should be more advanced and encompass more than elementary disturbances. This is because the new advanced disturbances should have a minimal inherent correlation with the simple disturbances. The simple disturbances are explicitly made on the elementary level. The advanced disturbances use more complex modifications of data; the inherent risk or correlation should be smaller.

4.2 Correlations

If two values correlate strongly, it is possible to use one of the values to roughly calculate the other value. For example, if the number of calories a person has burned during a running session is known, it is possible to calculate roughly how far the person ran. If a correlation can be seen between the automatic elementary disturbances and the more complex disturbances, there is a plausibility that these automatic values can be used to calculate how well the tested models will handle unknown disturbances. If no correlation can be found between the automatic elementary disturbances and the more complex disturbances, it an indication that the proposed method is insufficient as an estimate of how well a machine learning model will work on real disturbances in data. The method used for measuring correlation is linear correlation, this is measured using JASP [27] a free and open-source program for statistical analysis. The results from the tests are saved in a CSV file which is then used in JASP.

4.3 Data collection

The datasets chosen for the advanced disturbances and the statistical analysis, are the CIFAR-10 dataset and the URBANSOUND8K dataset. These are the two with larger, arguably more complex, data points out of the three datasets used in the automatic tests described in the methodology chapter. The MNIST dataset is simpler in the sense that each datapoint is smaller, and that is it black and white images that have less complex data than the other two. CIFAR-10 is of images, URBAN-

SOUND8K is for audio. Testing multiple types of data can indicate if the proposed method is useful for a general use-case.

To perform the statistical comparison, the following data is collected for each machine learning model:

1. The accuracy of the test data set in holdout validation, i.e., the original unmodified data is tested with the machine learning model to get the initial accuracy
2. The average disturbance level tolerated before a wrong classification for each simple disturbance described in the method chapter
3. The percentage of correct classifications after applying new more advanced disturbances

For the evaluation process, many machine learning models are created. For both the audio and the images, 24 hours of testing was performed with 24 CPU threads (AMD Ryzen 5 1600X and AMD Ryzen 5 3600) and a GTX 1060 graphics card. For images 267 different models were trained and tested. For audio, 480 different models were trained and tested. Having a large sample size of results from machine learning models will make it possible to create better statistical models.

4.3.1 Data from holdout validation

For each machine learning model, testing with the initial input data is performed. This is the accuracy of the model before any disturbances are added. This accuracy is saved to see how it compares to the numbers from testing the different disturbances. This data is also used to discard the initially wrong classifications since the interesting data is the one that gets misclassified by adding disturbances. The overall accuracy is still saved, but the already wrong data is not used for the simple disturbances.

4.3.2 Data from simple disturbances

For each machine learning model, tests are conducted with the input having the added simple disturbances. These are the disturbances described in the method chapter. The model's accuracy with this modified data is saved to compare with the initial unmodified data and the more advanced disturbances. These simple disturbances are described in the methodology chapter.

4.3.3 Data from advanced disturbances

For each machine learning model, tests with the input having the more advanced disturbances are performed. The model's accuracy with this modified data is saved to compare with the initial unmodified data and the data with simple disturbances. These advanced disturbances are described in 4.3 and 4.4.

4.4 Evaluation process

To evaluate if the proposed method can be useful when predicting the performance for unseen disturbances, the testing methodology shown in Figure 20 is used. The method uses two datasets. One dataset contains data without any disturbances; on this dataset, automatic elementary disturbances are added. The second dataset is a data set filled with data that have complex data specific disturbances. The complex disturbances are mentioned in section 4.5, and 4.6. The automatic elementary disturbances are discussed in section 3.3, with a list of disturbances in section 3.3.6.

Evaluation is conducted in the following 4-steps:

- **Step 1:** Train multiple machine learning models on the data-set without disturbances.
- **Step 2:** Analyze how well the machine learning models can handle the automatic elementary disturbances.
- **Step 3:** Test the machine learning models on the data-sets with real/complex disturbances.
- **Step 4a:** Compare results and see if there is any correlation.
- **Step 4b:** Create simple linear regression models to see the improvement

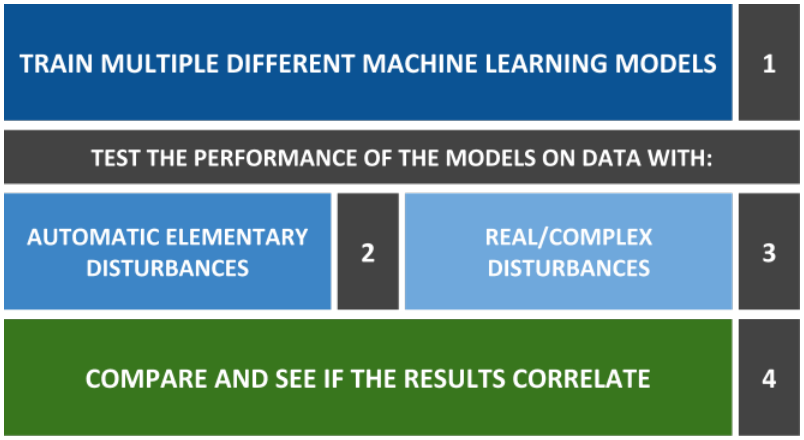


Figure 20: 4-step process of evaluation the proposed method

The machine learning models used for both the CIFAR-10 dataset and the URBAN-SOUND8K dataset are different convolutional neural networks based on the same open-source example [28]. The reason for using convolutional neural networks is because of that it is relatively easy to train compared to some other machine learning types. This is important since many models are tried, meaning that training has to be done many times. As mentioned in the scope and delimitation section 1.1.2, the testing is performed on the second significant level, i.e. one type of network with different types of data.

To generate data for statistical analysis, the networks are trained many times with different parameters to get varying performance.

The testing is performed on many, differently trained, two-layer convolution neural networks. This means that they are trained on the same data but with different parameters. Too see the exact parameters used for each test see the raw results [29].

List of parameters:
- Batch size: 128-512
- Convolutional layer 1 size: 32-256
- Convolutional layer 2 size: 64
- Fully connected layer size: 512-4096
- Training steps: 200-8000

For each machine learning model tested, the simple disturbances are applied individually to 1000 data points to test the limit of each classification. The 1000 data points are picked from the 8732 sound excerpts in the URBANSOUND8K dataset when testing audio and from the 60000 images from the CIFAR-10 dataset when testing images, the first 1000 data points are used for each test. For the advanced disturbances, the average accuracy is saved, here the average is calculated after testing all the data points of the dataset.

After data collection, statistical analysis can be made using the results. This analysis clarifies the viability of the method for that machine learning model. Ideally, there should be a correlation between the simple disturbances and the advanced disturbances. Having this correlation would indicate that the simple disturbances can be used to predict performance on unseen advanced disturbances.

As a baseline a linear regression model using only the hold out validation accuracy is created. If a better linear regression model can be created with the added information from the proposed method, this would indicate that the simple disturbances can be used to improve the prediction ability of the machine learning model's resilience to unseen disturbances. Getting a more accurate linear regression model would mean that the added information from the proposed method can not be extrapolated from only the accuracy from hold out validation. This would, in turn, indicate that the simple disturbances in addition to the accuracy from the holdout validation can be used in a general testing methodology for predicting how well a machine learning model will handle unseen disturbances.

4.5 Advanced disturbances - Images

The goal is to have more realistic disturbances that the simple disturbances can be compared to. For this, data specific disturbances are chosen. For images, there are three different disturbances chosen, that emulate realistic disturbances that can happen.

4.5.1 Distorted data (Fish Eye)

The goal is to see if there is a correlation between the simple and the advanced disturbances. By doing a fish-eye inspired distortion effect on the pictures, we could simulate focal distortion on the pictures used for testing. This essentially means that the information of the pixels gets distorted in a mathematical way that is dependant on the input format. To understand why this can not be done in the proposed elementary automatic testing method, read 3.3.3 on page 15. This has little in common with the elementary disturbances in the proposed method. Therefore, there is little inherent correlation with the elementary disturbances. In Figure 21, three examples of this can be shown.

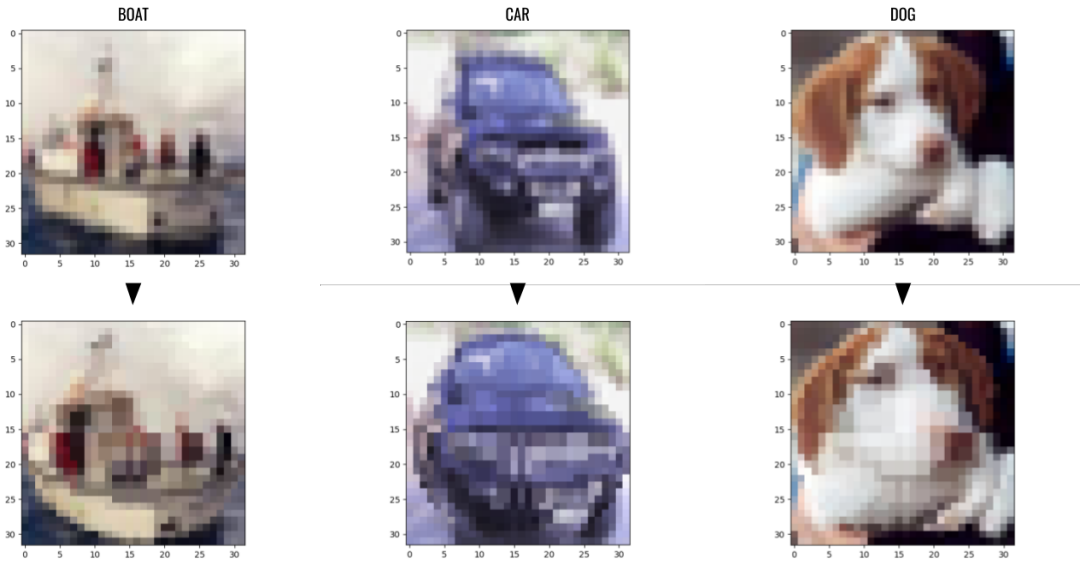


Figure 21: Examples of modified data using the fisheye distortion.

4.5.2 Occluded data (Colored boxes)

A second disturbance that can occur in many real-world situations is occluded data. In pictures, this corresponds to an element that occludes the object in the picture. To test data like this a simple data set with boxes in different colors that hid a part of the pictures was generated. This is an easy and effective way to see how the algorithm performs under circumstances when the data isn't fully visible. This is, for example, prevalent in face recognition when the face can be occluded by, for example, sunglasses [30]. Examples of this can be seen in figure 22. This is similar to all the random functions in the proposed method, to make the similarities less prevalent the boxes have uniform colors and are structured in a way that uses the input format in a way that can't be replicated in the automatic elementary disturbance testing. This is further explained in 3.3.3.

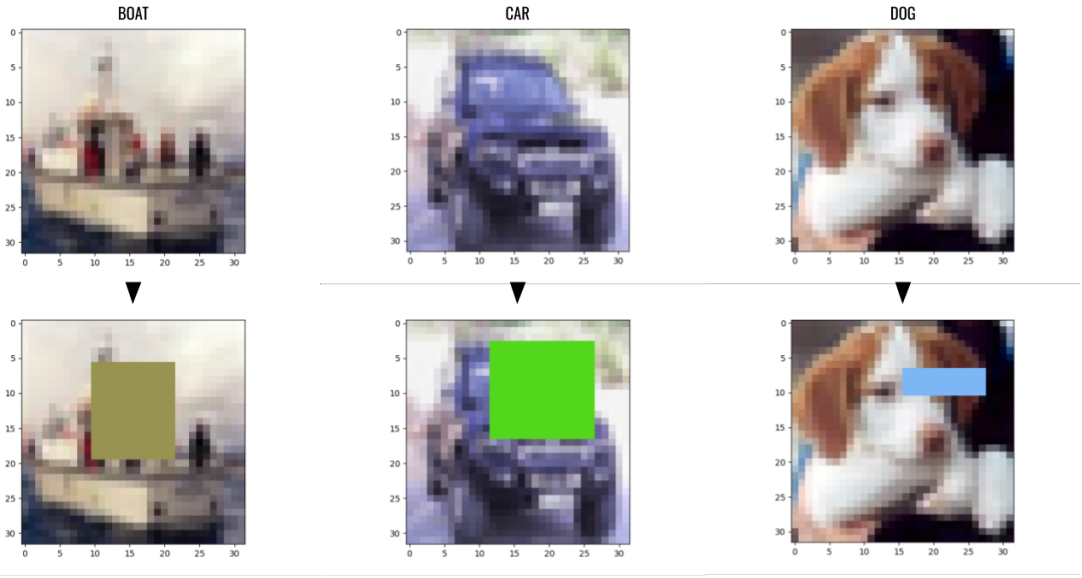


Figure 22: Examples of modified data using a random box block method.

4.5.3 Filtered data (Color shift)

To test the stability when it came to input nodes, we added a filter on top of the pictures. This would correspond to a scenario where the camera is behind a tainted glass, or in circumstances when the lighting conditions change drastically. This was made by adding a set filter over the data; in essence, a taint was added to change the color of the pictures. Figure 23. Color is structure dependant, and adding a uniform color shift on three different nodes in a repeating pattern can not be replicated in the automatic elementary disturbance testing without knowing the data type and structure. This is further explained in 3.3.3.

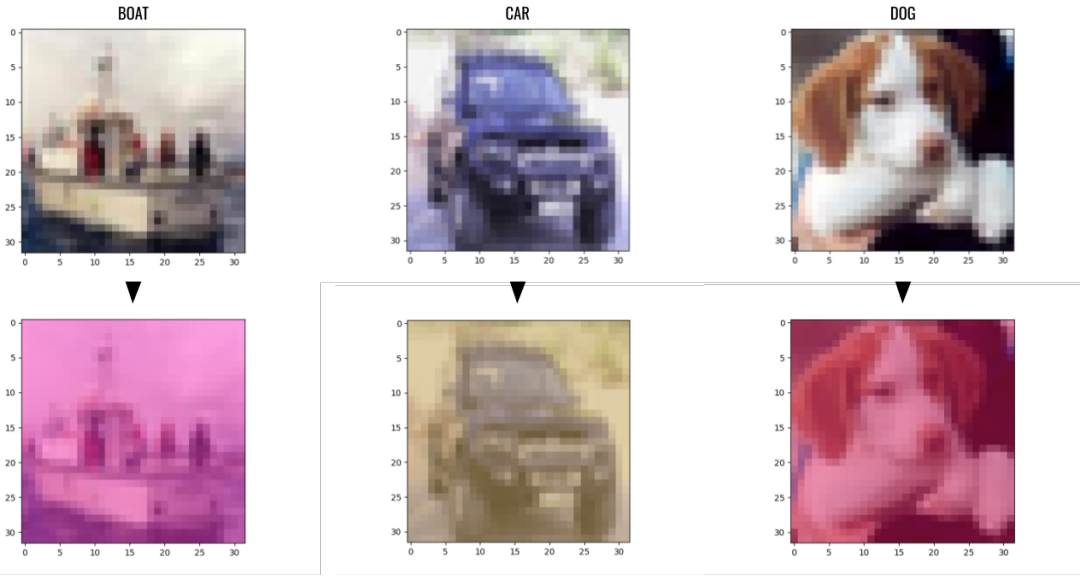


Figure 23: Examples of modified data using an overlaying colour layer.

4.6 Advanced disturbances - Audio

When testing audio compared to images, many of the elementary disturbances correlate very well with real-world disturbances. Fade correlates well with audio overlap. The random disturbances correlate well with noise, stutter, and similar disturbances that often occur in a real scenario. This was offset by the feature extraction in the audio model, to avoid having clear, obvious correlations with the disturbances evaluated. By having the automatic elementary disturbances added on the extracted values, this obvious similarity between the simple and advanced disturbance mostly disappears. Read more about feature extraction in 3.6.1.

4.6.1 Distorted data (Low bit-rate)

In audio bit rate is the number of bits for a certain amount of time, a high bit rate means that there is more data to describe the audio. Having higher bitrate gives higher quality audio but also require larger files. To test the proposed method on audio, we choose modifying bitrate as a disturbance due to the close link to reality. In situations where either the bandwidth of an information stream is very low or when having a small file-size is essential are cases when modifying the bit rate can help. Bit rate is commonly measured in kbps or kbit/s (kilobits per second). Following this, "k" will be used as a shortened version of kbit/s in figures.

Three different bit rates are chosen as disturbances:

- 24 kbit/s: Original audio bit rate. This is to have a reference when using the overlap data sets.
- 4 kbit/s: Very compressed audio, it is roughly the limit where you in this data set can classify the sounds with relative ease.
- 2 kbit/s: Heavily compressed audio can be hard for a human to classify the sound at this bit rate.

The choice of having such low bit rates for testing is to make classification harder for the machine learning models. Having a higher bit rate than 4k is too easy for the models trained to handle. There can, for example, exist situations where a significant reduction in quality is needed, for example, a connection over satellite. One satellite network solution has a connection bandwidth of 2.4 kbit/s [31], that can be used for voice calls anywhere on earth.

In theory the distortion effect could be compared to the random simple disturbances. Since feature extraction is performed, as mentioned before, this effect is not as prominent.

4.6.2 Distorted data (Overlap)

Overlapping audio is also a common occurrence. For example, when somebody records audio in a noisy room. We made two different data sets to test this. By making the labeled original class sound more dominant, it has a higher amplitude than the overlapping sound files.

- 24k bit rate with low amplitude overlap: A data set with disturbance sound that is much lower than the dominant sound.
- 24k bit rate with higher amplitude overlap: A data set with disturbance sound that is higher amplitude than the previous data set.

The reason for having two data sets is to be able to compare an easy disturbance (low extra noise) and a hard disturbance (extra high noise) and see if that changes the results. In theory the overlap effect could be compared to the fade simple disturbances. Since feature extraction is performed, as mentioned before, this effect is not as prominent.

4.6.3 Distorted data (Examples)

Examples of the disturbances can be listened to by following the links listed below:

- original: https://zenodo.org/record/3898510/files/DogOriginal.wav
- 24k bit rate: https://zenodo.org/record/3898510/files/Dog24k.wav
- 24k bit rate with low amplitude overlap: https://zenodo.org/record/3898510/files/DogOverlappedLow.wav
- 24k bit rate with higher amplitude overlap: https://zenodo.org/record/3898510/files/DogOverlappedHigh.wav
- 4k bit rate: https://zenodo.org/record/3898510/files/Dog4k.wav
- 2k bit rate: https://zenodo.org/record/3898510/files/Dog2k.wav

4.7 Exploring correlations

Correlations are useful because they can indicate a predictive relationship. This ability to predict means that if one set of values is known, the second set of values can be predicted. This ability is useful when creating a predictive model. Correlation is usually referred to when a pair of variables are linearly related. An example of this can be that the thrust to weight ratio of an airplane correlates to the maximum acceleration of an airplane.

There are many ways to measure correlation, one the most used metric being the Pearson correlation coefficient (PCC). PCC is a statistic that measures the correlation between variables. It can have a value between -1 and +1. 1 is a perfect positive linear correlation, a 0 means no correlation, and a -1 is a perfect negative correlation. The reason the Pearson correlation coefficient was chosen over the Spearman

correlation coefficient (SCC) is that the data points collected follow a bivariate normal distribution. This distribution works well with PCC. SCC is better in cases when the points do not follow a bivariate normal distribution [32]. JASP [27], an open-source tool for statistics, is the primary tool used to perform the statistical analysis.

In the evaluation of the method proposed, the correlation between the elementary disturbances, the advanced disturbances, and the basic holdout validation accuracy on data without disturbances are calculated. This means that every correlation between these values is calculated.

The result from this shows how well the simple elementary disturbances correlate with the advanced disturbances. A high absolute correlation would be a good result. Having a low absolute correlation would be bad. Correlation is a very basic form of finding relations between two different variables, it is, therefore, a good +core indicator of if the values are useful or not.

4.8 Linear regression

To show the usefulness of the correlations shown in the correlation testing, linear regression models were evaluated. Linear regression is an approach to model relationships between a dependent variable and one or more explanatory variables. Linear regression, when having one explanatory variable, is called simple linear regression. In this case, the relationship between variable y (dependent variable) and the variable x (explanatory variable) is represented as a vector. An example of this can be seen in Figure 24.

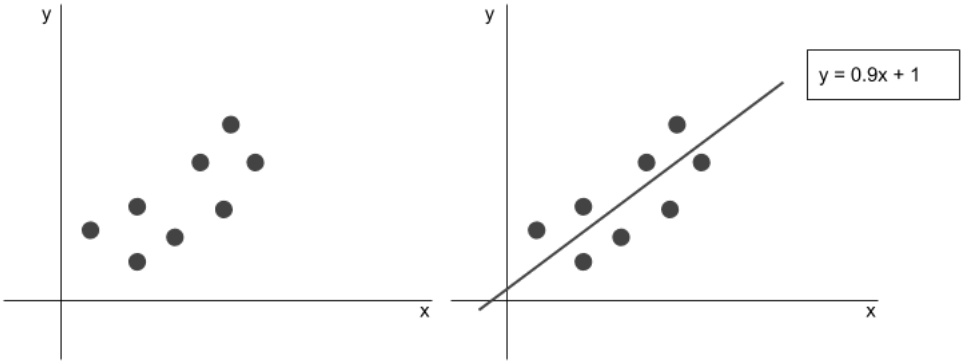


Figure 24: Example of linear regression fitting data

By using linear regression, it is possible to show the benefit of the added information from the simple disturbances when predicting the performance in the advanced disturbances. Two linear regression models are created. For both models, the dependant variable (the predicted value) is the accuracy on advanced disturbances. The explanatory variables (values used to predict the dependant variables) are different. For one model, only the accuracy from the holdout validation is used as the

explanatory variable. The other model uses the results from the tests with simple disturbances in addition to the accuracy from holdout validation.

Linear regression models are used as a complement to the correlation matrix. By using multiple linear regression, it is possible to show the effectiveness of the proposed method when it comes to predicting the accuracy in unseen data. In this particular case, the simple elementary disturbances are used to predict the different advanced disturbances. The benefit to this compared to only using correlation as a proof is that the difference in the methods can be shown. This gives a more solid foundation to draw conclusions from and also gives a rough estimate of how effective the proposed method is. When comparing the models, the difference in the accuracy of the prediction can be measured.

Due to the fact that linear regression is widely used in statistics, and it suited the situation, we choose this an approach to statistically show the potential of the proposed method. By using linear regression, the value of the proposed method can be evaluated. Linear regression is, in this case, only relevant in a research perspective and is not practical as a complete model for predicting how well a machine learning model will handle unseen disturbances.

One problem with linear regression is that it assumes a linear relationship between the dependent variable and the explanatory variables. But due to the fact that the models only will be used as a method of evaluation, this will not be a problem. The ideal model might not be linear. Linear regression could, however, give a good indication of performance. Because the goal is merely to evaluate the difference between the two cases, a more complex model is not needed. If new information can improve the prediction ability, that means that the new information is, in fact, new and can not be extrapolated from the holdout validation accuracy. An improvement in the linear regression model would also indicate an improvement in more complex models.

One other problem with linear regression is that it only tries to minimize the error of all predicted results. Minimizing the average error in predictions is useful, but not always the information that is most valuable. In some cases, it is more important for a model to be able to compare different entities (for example different machine learning models) and decide what entity is better, accurately predicting performance is not always as important as picking the best performance. For example, a person must choose a machine learning model. That person uses a statistical model to predict which machine learning model will be the best from a set of results. It does not matter what the mean error of the predictions from the statistical model is. It only matters which machine learning model is the best. In this case, using linear regression as the statistical model will most likely not be optimal.

In Table 4.1 a hypothetical scenario showing this can be seen. In the hypothetical example, the actual accuracy of the two machine learning models are 80 and 79 percent respectively. The linear regression model 1 is very accurate, it only gets a mean error of 1 percent, It does, however, not correctly predict which machine learning model that is the best. The linear regression model 2 is worse at predicting the actual accuracy and has a mean error of 4 percent. The second model does, however, correctly predict which model has the highest accuracy. This means that both linear regression models can be seen as better than the other depending on the situation. Using linear regression to predict the best model is tried in section 5.5.2.

	ML Model 1	ML Model 2	Mean error	Correct best model
Actual Accuracy	80%	79%	-	-
LR Model 1 Prediction	79%	80%	1	No
LR Model 2 Prediction	85%	76%	4	Yes

Table 4.1: Hypothetical example showing the prediction of two models. ML stands for machine learning, LR stands for linear regression

5
Results

In this chapter, both the results from the saved data of simple disturbances and the statistical analysis are shown. For the saved data, both the correlations and the data that can be manually reviewed are shown. For the statistical analysis, the result from both correlations and different statistical models are explored. To see the raw test results, see [29].

5.1 Analysing data from tests

Some general analysis can be done with the saved data for each time a model is tested. This analysis can be useful, for example, for seeing which classification changes to what new classification for each disturbance. In this example, the data consists of 10000 images from the MNIST dataset [12]. This is one example of what can be collected from a test with a machine learning model. As described in the method chapter, for each disturbance, the limit of how much of a disturbance can be added is tested. When the disturbance causes a wrong classification, that new wrong classification is saved. The full result of this particular test can be seen in the Appendix, Tables A.1-A.18.

Automated generation of tables exist for the distribution and specifics about change in classifications. The disturbance level is saved for statistical analysis but tables are not generated. Generating additional tables could be done fairly simply, but is currently not done with the performed implementation of the method. To see individual disturbance levels the file names of saved disturbances can be used (see section 3.5).

The following, Figure 5.1, is an aggregate from the tables A.10-A.18, showing the distribution of classifications after disturbances are added. Each row is a simple disturbance, the columns 0-9 are the new classifications after the data is modified.

Table 5.1: Classification distribution after simple disturbances

	0	1	2	3	4	5	6	7	8	9
Setting random element to a random value	0%	0%	47%	39%	3%	3%	0%	1%	3%	2%
Setting random elements to 1	0%	0%	53%	33%	4%	3%	0%	1%	3%	1%
Setting random elements to 0	0%	4%	15%	24%	3%	31%	0%	14%	5%	3%
Fade between two datapoints	9%	8%	9%	10%	9%	7%	7%	8%	15%	18%
exponential mapping increase of elements	10%	11%	10%	10%	9%	9%	9%	10%	10%	10%
exponential mapping decrease of elements	9%	11%	10%	11%	10%	11%	8%	11%	7%	9%
max	15%	0%	12%	11%	6%	7%	8%	5%	18%	17%
min	3%	11%	11%	16%	11%	20%	2%	20%	1%	4%

Each row is the distribution of new classifications, after the limit for the wrong classification is reached. For example, after performing the "setting random element to a random value" disturbance, 47% of the new, wrong, classifications were classified as "2". As mentioned before, the more detailed data of the new distributions can be seen in the Appendix. The initial, correct data, can also be seen in the Appendix in Figures A.10-18.

The raw output saved can be seen in a folder called output. The method of how files are saved is described in 3.6. The following example shows images that were originally correctly classified as a 0, then after the random disturbance was classified as a 2.

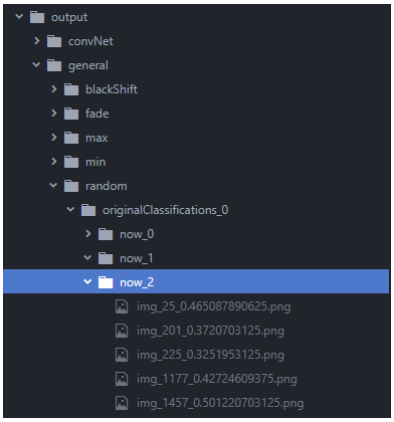
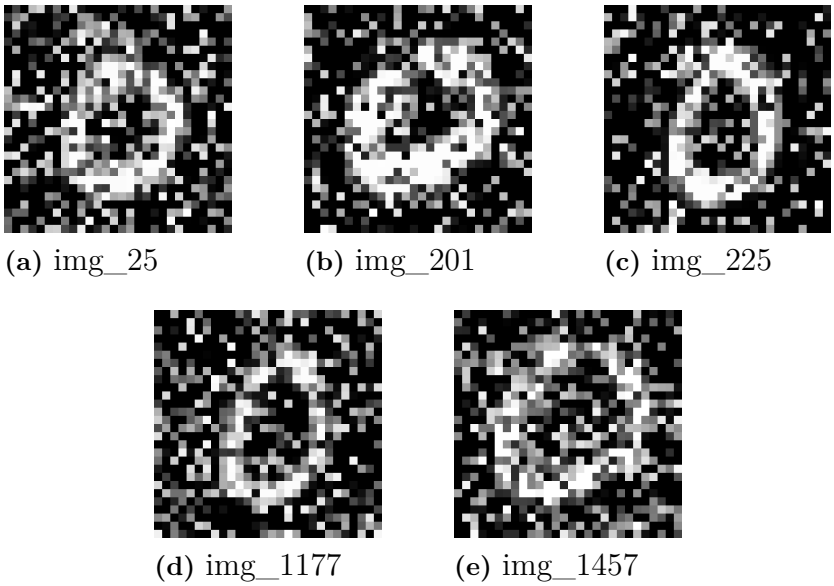


Figure 25: Example of output file folder structure, after analysis of the MNIST dataset

The first part of the filename is to ensure each file has an original file name, for example, img_25_, for the first image. After that, the level of disturbance that made the image change classification is added, for example, 0.465..., i.e., $\approx 47\%$ of the image is changed in this case. As can be seen in Table A.2, 977 images originally had the classification 2, and 84% of those were classified as 2 after the distortion, the images shown are a small amount of the real images. Here are the first five images classified initially as 0, now classified as 2:



(a) img_25 (b) img_201 (c) img_225

(d) img_1177 (e) img_1457

Figure 26: Examples of images originally classified as 0, now classified as 2

5.2 Correlation testing overview

As described in 4.2, variables can be correlated. There are situations where one variable can be predicted with one or more other variables. For both the images and audio tested, correlations are calculated. These correlations are used to analyze if the simple disturbances could be used to predict resilience to more realistic disturbances. The variables, in this case, are:

- accuracy with unmodified data
- accuracy with each advanced (realistic) disturbance
- tolerated levels of each simple disturbance before incorrect classification

5.3 Results correlation - Images

5.3.1 Collection of data

To collect the data points necessary to do a statistical analysis, 267 tests were conducted on the CIFAR-10 data set using both the simple and complex disturbances. All the tests are individually trained convolutional neural networks. These networks are trained with different parameters (see section 4.4) to see how the method performs in various situations. The number 267 is not derived from a special criterion. It was the number of tests that ran in the time allocated.

5.3.2 Interpreting the image correlation heatmap matrix

The matrix below, Figure 27, shows the result of computing the Pearson correlation coefficient (PCC) for all variable pairs. The variables are the results gathered from the different disturbances and the initial accuracy from holdout validation. The variables have shortened names in the figures, according to Table 5.2. Every cell in the matrix shows a PCC for a variable pair. For example, "testAcc" and "distortionAcc" have a PCC of 0.978, meaning that they have a high positive correlation since the value is close to 1. In this example, it means that the testing accuracy from holdout validation has a strong correlation with the accuracy of fish-eye distortion. This strong correlation might indicate that the holdout validation is good enough to predict the stability in this particular case. However, this merely means that the advanced disturbance, in this case, might just have been to easy for the machine learning model to handle. As described in section 4.7, a number near -1 would indicate a high negative correlation, a number near 0 would indicate no correlation.

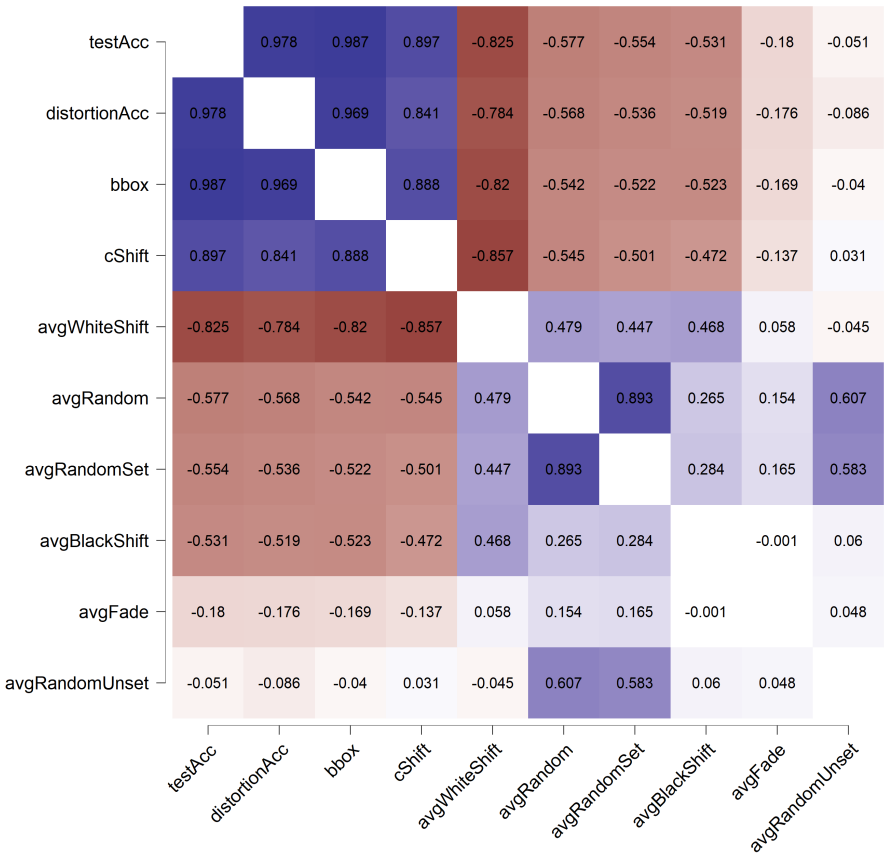


Figure 27: Correlation heatmap matrix of image testing variables

Table 5.2: Explanation of image variables

Disturbance	Variable	Meaning
none	testAcc	accuracy averages from holdout validation (unmodified data)
advanced	distortionAcc	accuracy averages with the fish eye distortion (4.5.1)
advanced	bbox	accuracy averages with the occlusion distortion (4.5.2)
advanced	cShift	accuracy averages with the color shift distortion (4.5.3)
simple	avgRandom	averages of resilience to disturbance "Setting random element to a random value" (3.3.6.1)
simple	avgRandomSet	averages of resilience to disturbance "Setting random elements to 1" (3.3.6.2)
simple	avgRandomUnset	averages of resilience to disturbance "Setting random elements to 0" (3.3.6.3)
simple	avgRandomFade	averages of resilience to disturbance "Fade between two datapoints" (3.3.6.4)
simple	avgWhiteShift	averages of resilience to disturbance "Exponential mapping increase of elements" (3.3.8)
simple	avgBlackShift	averages of resilience to disturbance "Exponential mapping decrease of elements" (3.3.7)

5.3.2.1 Correlation with white shift

The automatic elementary disturbance that had the highest correlation with the accuracy on the different advanced disturbances was the white shift disturbance (explained in 3.3.8). The advanced disturbance that had the least correlation with the holdout validation accuracy was the color shift disturbance. This can be seen in Figure 27. The fish-eye lens disturbance and colored box disturbance have a very high correlation with the holdout validation accuracy. And because of the high correlation with the holdout validation accuracy, the proposed method adds very limited benefits in these cases. The advanced disturbance that stands out is the color shift; in this case, the white shift disturbance has a strong correlation that is not directly correlated with the holdout validation accuracy, meaning that information that otherwise could be overlooked is shown with the method. This correlation can be seen in Figure 28. To see the correlations between all the variables for images, see Figure 37 in the Appendix.

5.3.2.2 Statistical model with test results

Using linear regression, a simple model can be made using the automatic elementary disturbances in addition to the accuracy of holdout validation. These tests show the two different models made using linear regression. One model uses only the accuracy from holdout validation. The other model uses the accuracy of holdout validation and the results from the testing with simple disturbances. The test is made to show the difference in performance between the models, seeing if there is an added value to running tests with the multiple elementary disturbances. As seen below, the model can be accurate during these circumstances and show value in automatic tests. Even if the benefits are small with models like this compared to only using the accuracy from holdout validation, the data shows that additional tests can improve the prediction ability. In the model shown in the table below, the value of the different tests can be seen, the automatic test with the most correlation to the result is the white shift mentioned earlier. As seen in Figure 29 both of the models performed very well, but the model using all the additional information from the tests performed 15% better than the model only using the accuracy from the holdout validation. This means that only using the accuracy had a decent results, but adding information from the proposed method (the simple disturbance tests) improved the performance.

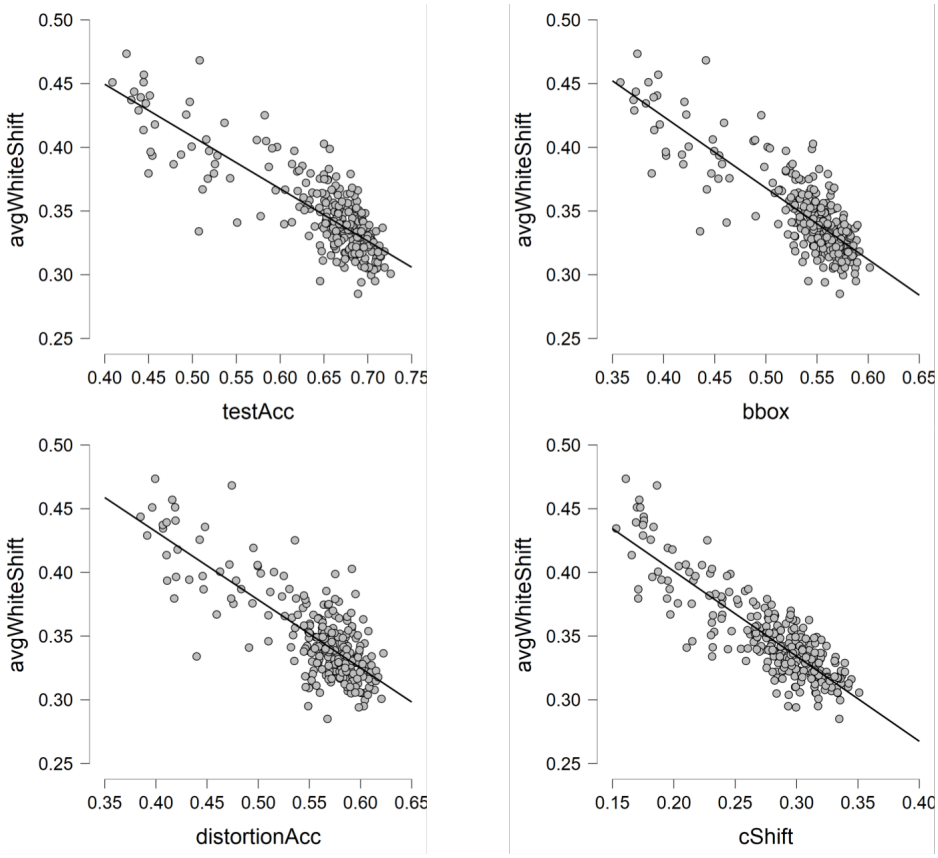


Figure 28: Correlations between advanced disturbances and the simple white-shift (3.3.8) disturbance

It is not clear that the relationship between the input and the output is linear. The real relationship between the input and the output does not matter because the model is only meant to show the benefit of the added information. In essence, it shows that the added information from the proposed method can not be extrapolated from the holdout validation. This means that the proposed method adds new information that can improve the prediction ability.

The two linear regression models are shown below, in Figure 29. The real difference is roughly 15% higher accuracy using the automatic tests and 27% improvement (reduction) in mean squared error. More in-depth statistics are shown for the model, including all the results in the Appendix. See Tables A.19-21.

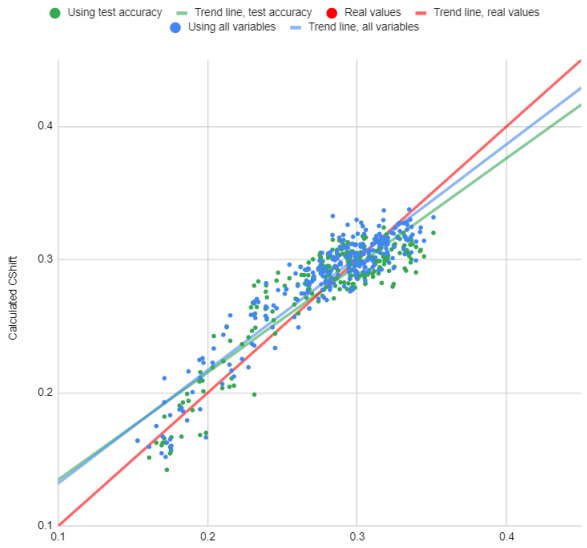


Figure 29: Predictions using all the information (holdout validation accuracy and simple disturbance tests) in blue and only using the holdout validation accuracy in green.

5.4 Results correlation - Audio

5.4.1 Collection of data

To collect the data points necessary to do a statistical analysis, 480 tests were conducted on the UrbanSound8k dataset with both the simple and advanced disturbances. All the tests are on individually trained convolutional neural networks. These networks are trained with different parameters (see section 4.4) to see how the method performs in various situations. The number 480 is not derived from a special criterion. It was the number of tests that ran in the time allocated.

5.4.2 Interpreting the audio correlation heatmap matrix

The matrix below, Figure 30, shows the result of computing the Pearson correlation coefficient (PCC) for all variable pairs. The variables are the results gathered from the different disturbances and the initial accuracy from holdout validation. The variables have shortened names in the figures, according to Table 5.3. Every cell in the figure shows a PCC for a variable pair. For example, "testAcc" and "distortionAcc24k" have a PCC of 0.931, meaning that they have a high positive correlation since the value is close to 1. In this example, it means that the testing accuracy from holdout validation has a strong correlation with the accuracy of the audio disturbance that has 24kbit/s bitrate. This means that the accuracy when classifying the original data and the accuracy when classifying the data with bitrate changed to 24kbit/s are highly correlated. As described in section 4.7, a number near -1 would indicate a high negative correlation, a number near 0 would indicate no correlation.

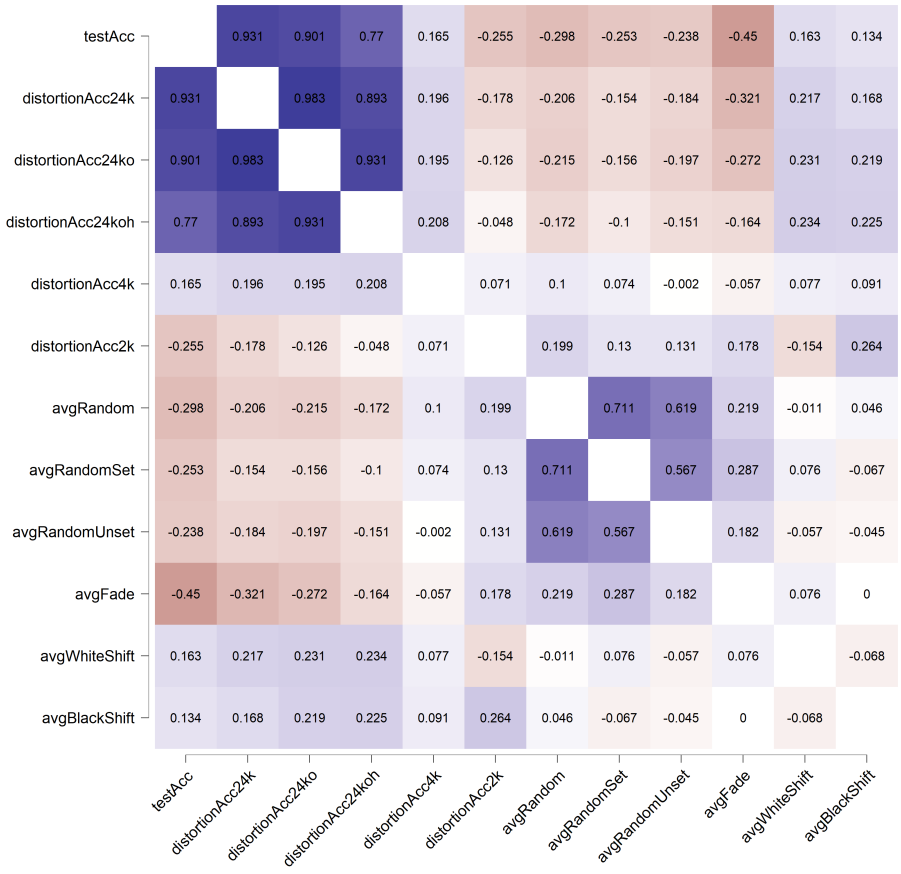


Figure 30: Correlation heatmap matrix of audio testing variables

Table 5.3: Explanation of audio variables

Disturbance	Variable	Meaning
none	testAcc	accuracy averages from holdout validation (unmodified data)
advanced	distortionAcc24k	accuracy averages from the original 24 kbit/s audio (4.6.1)
advanced	distortionAcc24ko	accuracy averages from the 24 kbit/s audio with overlap (4.6.2)
advanced	distortionAcc24koh	accuracy averages from the 24 kbit/s audio with high amount of overlap (4.6.2)
advanced	distortionAcc4k	accuracy averages from the 4 kbit/s audio (4.6.1)
advanced	distortionAcc2k	accuracy averages from the 2 kbit/s audio (4.6.1)
simple	avgRandom	averages of resilience to disturbance "Setting random element to a random value" (3.3.6.1)
simple	avgRandomSet	averages of resilience to disturbance "Setting random elements to 1" (3.3.6.2)
simple	avgRandomUnset	averages of resilience to disturbance "Setting random elements to 0" (3.3.6.3)
simple	avgRandomFade	averages of resilience to disturbance "Fade between two datapoints" (3.3.6.4)
simple	avgWhiteShift	averages of resilience to disturbance "Exponential mapping increase of elements" (3.3.8)
simple	avgBlackShift	averages of resilience to disturbance "Exponential mapping decrease of elements" (3.3.7)

One interesting variable to note is 2 kbit/s bit-rate sound distortion. This variable has the shortened name "distortionAcc2k" in the correlation matrix in Figure 30, see Table 5.3 row 6. When the sound is compressed to this low of a bitrate, to 2 kbit/s, the automatic elementary disturbances have a significant correlation to the accuracy in the 2 kbit/s data set compared to the holdout validation accuracy. Both the accuracy "testAcc" and the simple disturbances have a low correlation to the 2k disturbance, the simple disturbances do, however, have a high correlation in

comparison with the test accuracy. Since the combined statistical model uses both the accuracy and the results from the simple disturbances, this is a good sign.
One other interesting thing to note is that the 2 kbit/s bitrate has an opposite correlation compared to most of the other complex disturbances. Another interesting variable is the exponential mapping decrease of elements. This variable has the shortened name "avgBlackShift" in the correlation matrix in Figure 30, see Table 5.3 on the last row. The black shift disturbance test correlates better towards the distorted 2 kbit/s bit rate data than the performance on the test data set.

In the heatmap, it can be seen that the simple elementary disturbances correlate to a varying degree with all the more complex disturbances. In most cases, it is only a slight positive or negative correlation.

Below, in Figure 31, the improvement in prediction is shown. The labels used are from the variables starting with distortionAcc in Table 5.3, for example, "2k," meaning the accuracy from the 2 kbit/s audio.

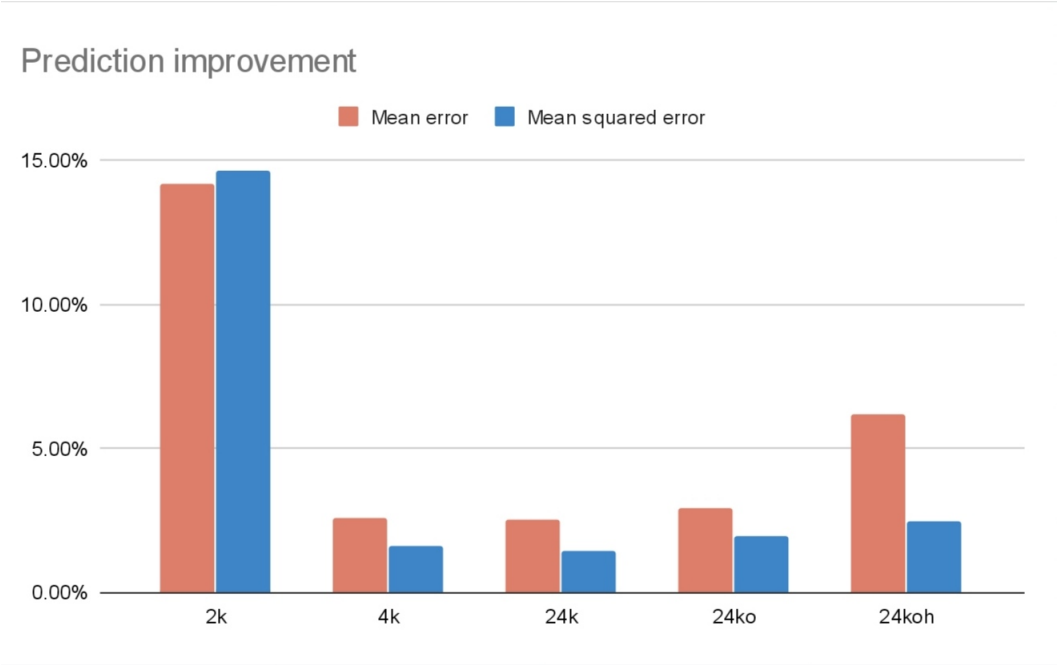


Figure 31: Prediction improvement in linear regression model using the information from the proposed method (simple disturbance tests); reduction of error in statistical models compared to only using holdout validation as prediction. This is like Figure 29, but for all the advanced audio disturbances.

5.4.3 Improvements compared to holdout validation

Two linear regression models are created for each complex disturbance. One made for only using the information from the holdout validation, and one made with all the added information from the simple elementary disturbance tests. This is done to see if the simple disturbances can be used to predict the performance of unseen

advanced disturbances better than only using holdout validation.

The data that sees most improvement by adding the automatic elementary disturbance tests is the 2 kbit/s bit rate data. It can be seen that by using automatic elementary disturbance tests makes the mean error of the prediction accuracy 14.21% lower, and the mean squared error of the prediction accuracy 14.64% lower. This is a significant improvement compared to all the other tests.

The second most improved is the 24 kbit/s bit rate with high volume overlapped disturbance sound (24koh). The result was an improvement in the mean error, but not as big of an improvement in the mean squared error. The result concerning the 4 kbit/s bitrate data is the most surprising, it has a low correlation with the automatic elementary disturbances and the accuracy from holdout validation "testAcc". This can be seen in figure 31

5.4.4 Deeper into the 2 kbit/s bit rate disturbance

The arguably most interesting result can be seen with the audio reduced down to a bit rate of only 2 kbit/s. This bit rate makes the whole file very low quality, and it can be hard for humans to interpret the sound. Most trained models have a hard time of archiving higher accuracy than merely guessing. Only guessing would get roughly 10% correct answers, since there are 10 different classes. The average achieved accuracy is only roughly 13%. Only a few models achieve higher accuracy than this. While interpreting the results, we can see that the automatic elementary disturbance tests are beneficial when it comes to predicting the accuracy of the machine learning model compared to only using the normal holdout validation method. The best performing automatic elementary disturbance test is the black shift (3.3.7) disturbance. Even on its own, it has a higher correlation than the standard holdout validation method. This can be seen in the plot below, Figure 32. To see the correlation between all the variables for the audio testing, see Figure 38 in the Appendix. When looking at the correlation with black shift disturbance, it can be noted that the points are less scattered and follow a more predictable pattern than with the test accuracy. One thing to note is that the worst improvement is seen in the 4 kbit/s data disturbance, even though it is the second most disturbed data. Therefore it can not be concluded that the method works better when data is more disturbed.

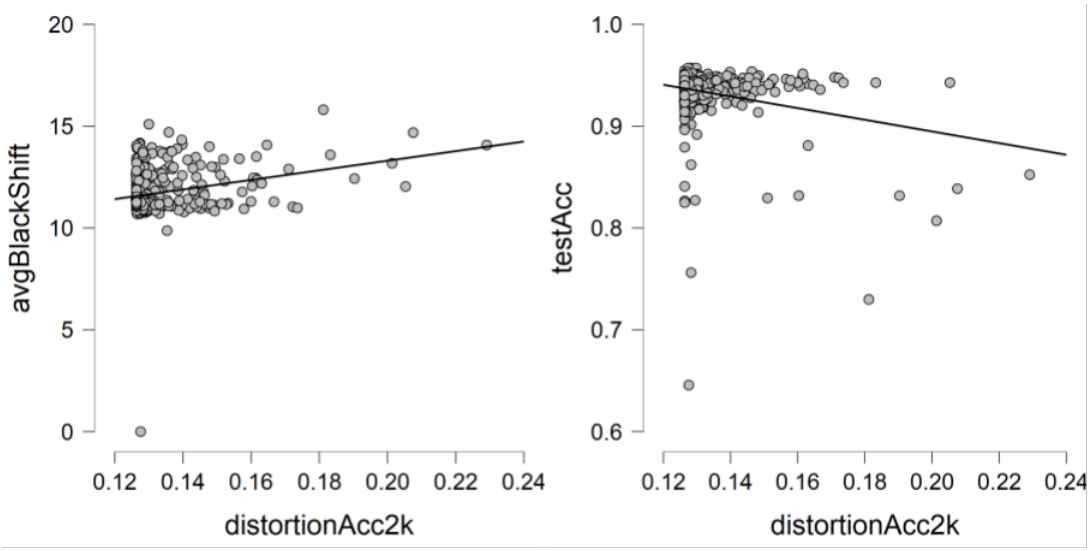


Figure 32: Comparing black shift (3.3.7) disturbance with holdout validation in correlation to 2kbit/s disturbance.

5.5 Using method as prediction

To test a general linear regression model, a way of determining the overall stability of a machine learning model against unseen disturbances is required. This is done by normalizing the accuracy value of each disturbance and adding them together. This creates a "total accuracy value". These values were calculated for both the audio and the images.

Using simple linear regression, a single model for the audio and images was created. This was to determine if there is a possibility to have a uniform model for multiple different input data and if the values could be accurately used to determine stability without regard to the input data. The combined correlation between variables can be seen in the Appendix, Figure 39.

5.5.1 Error prediction capability

By using the linear regression models to predict the accuracy in the advanced disturbances and compare it to the real accuracy of the machine learning model, an error value can be calculated. One linear regression model only uses the accuracy in the hold out validation test, the other linear regression model uses all the information from the proposed method in addition to the accuracy in the holdout validation test. Afterwards, these two linear regression models prediction accuracy is compared.

The old method of only using the holdout validation accuracy has a mean squared error of roughly 1180% higher than the proposed method, and a mean error of nearly 322% higher. This makes the proposed model much better in a general situation. This can be seen in Figure 33.

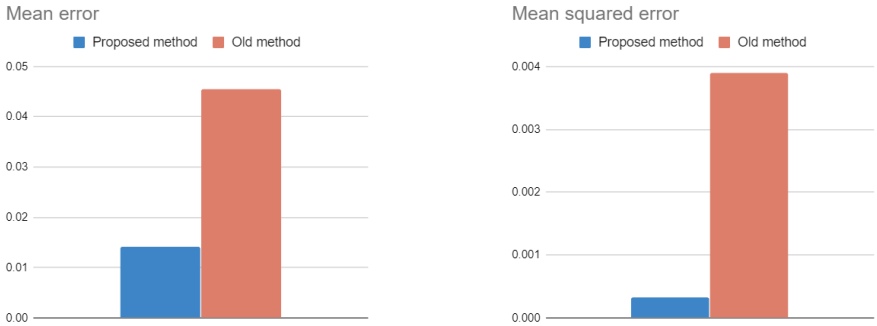


Figure 33: Error and mean squared error of statistical models; proposed model includes disturbance test result, old method only includes holdout validation

5.5.2 Best model prediction

In the field of machine learning, it is often beneficial to compare models. When looking at performance, error in predicted accuracy is not always the best metric. While lower error is generally better for a statistical model, picking the best machine learning model is sometimes more important. In this case, the value predicted by the linear regression is the accuracy of a machine learning model. What could be more interesting in a real-life use case is which machine learning model has the highest accuracy. A comparison could be a race. If someone wants to bet on the race, predicting who wins is more important than accurately predicting how long it takes for each person racing.

There is data gathered from the tests previously conducted. This data is about how well the machine learning models handled the different disturbances. When adding the information from the proposed method to a linear regression model, it can be seen that the mean error when predicting accuracy is lower, compared to only using the accuracy from holdout validation, see Figure 5.5.1. This however does not mean that the linear regression models necessarily are a good indicator when comparing two different machine learning models. To see how well linear regression models are at predicting the best out of two machine learning models an additional test was made. This was done to mainly see if there was a weakness of the linear regression approach.

The test of which model predicts the best accuracy is performed for all the machine learning models created. Data from each machine learning model is compared with the next machine learning model. The results can be seen below in Figure 34. It shows the prediction accuracy, i.e. how good the linear statistical models are at predicting what machine learning model out of two different will perform the best. As can be seen in the figure, both statistical models are about equal. Only using the accuracy from the holdout validation is slightly better. The additional information from the tests does not seem to help when using a linear statistical model when predicting which of the two models is the best. There could potentially be another type of statistical model that is more suitable for this, using the additional

information to increase the prediction ability.

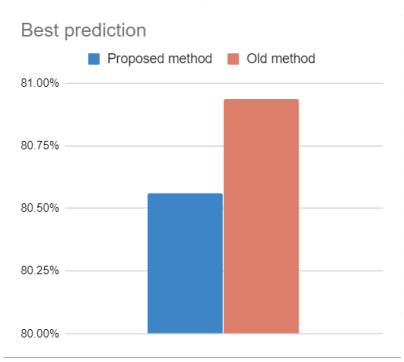


Figure 34: Prediction ability, comparing proposed method and holdout validation

6
Discussion

Due to the open nature of the thesis, it is hard to draw any solid conclusions without further discussion. The thesis mainly revolves around proving the possibility of general testing and the discussion will mainly focus on that aspect. The benefits, validity, and problems will be discussed concerning the method in general and the tests performed with images and audio.

6.1 Current evaluation methodology

During the evaluation phase, some alterations were performed to the different machine learning models to get more diverse results, but the structures of the different models were very similar. Essentially the goal was trying to create machine learning models with varying results. This was done by changing the parameters the determine how the neural networks were trained. This is problematic due to the fact that the machine learning models learned in very similar ways and had very similar performance, even though the attempted variability. Having a much wider array of networks and different machine learning models would most likely have yielded results that were more diverse. This would have given more data to work with hence increasing the possibility of getting a more interesting result. For example, all networks performed similarly in both automatic elementary disturbances and on the advanced disturbances. We can not say if having more different networks would give a better or worse result, but it is a weakness in the evaluation methodology that could be explored further.

As mentioned in the scope and delimitations section 1.1.2, three significance levels were identified, where the second level was explored in this thesis. The second level is using multiple types of data with one type of machine learning model. The third level would be to use multiple types of data with multiple types of machine learning models. Since the third level was not achieved, it is unclear whether the methods proposed would work as a general method. A truly general method would work no matter the type of machine learning model and no matter the type of data.

To get more conclusive results testing on this third level would have to be tried, adding more different kinds of models. This additional research would give more clarity to what you can expect from the values provided from the general elementary disturbances and in what cases they are the most useful. Currently, variations of convolutional neural networks are used to test all three datasets. Many types of

models can be used for classification; testing other ones would give more data for further analysis. A few examples of types that could be tested are:

- Linear classifiers
- Support vector machines
- Feed forward neural networks
- Recurrent neural networks
- Stochastic optimization

6.2 Testing with images

The testing shows a strong correlation between the accuracy of the holdout validation data and the accuracy of the dataset with advanced disturbances. As can be seen in Figure 35. These correlations are taken from the correlation matrix in the Appendix, see Figure 37. If there is a direct correlation between the accuracy of data without disturbances and the accuracy of the data with advanced disturbances, it might not be necessary to use the proposed method. Additional evaluation of the method in other cases is required, where the correlation between accuracy on data without disturbances and with disturbances is less significant. Finding advanced disturbances that do not correlate with other the accuracy from holdout validation would yield more interesting results and a stronger foundation to draw a conclusion from. When the correlation is as strong as it is now, the benefits of the proposed method is limited. The proposed method always gives a better prediction because it only adds information to the holdout validation, but the benefits are limited since the holdout validation accuracy already has a strong correlation with the advanced disturbances.

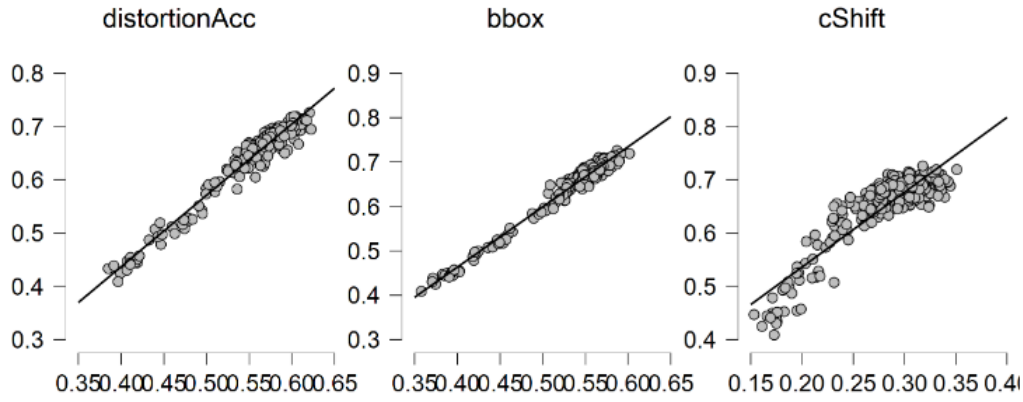


Figure 35: Correlation between the initial test accuracy and the accuracy with the advanced disturbances

6.3 Testing with audio

There is a significant difference in correlation between the different automatic elementary disturbances and the different advanced disturbances. This means that there is a difference in benefits depending on the situation. For example, it could be seen that the proposed method was super beneficial when predicting the accuracy in the 2k bit rate audio files, but not beneficial at all in the very similar 4k bit rate audio files. Even if it is the same type of disturbance, the extremity gives two completely different results. This makes it hard to predict how and when the proposed method is worth looking at. In all the cases, we performed the statistical analysis, we found some added value. However, the correlations could both be positive and negative, and it is hard to know when a certain value (the average tolerance to an automatic elementary disturbance) is good or bad. Minor changes in the disturbance can impact the usefulness of the proposed method dramatically. In some cases, we think that using the proposed method can hinder more than it can help due to the confusion it can cause. For example, white shift disturbance and black shift disturbance have a positive correlation with all the advanced disturbances in audio, but in images, they have a negative correlation. This means that for audio, it is good to have a high value in the black shift and white shift, while in images, the opposite is true. It can also be seen in the audio correlation heatmap (Figure 38) that the 2k bit audio files have the opposite correlation of all other disturbances.

6.4 Using the proposed method to predict the accuracy in unseen disturbances

The results, in section 5.5.1, were more effective than we suspected beforehand. The results in accuracy showed that the method with the automatic elementary disturbances could get a better prediction using linear regression than the holdout validation method. This meant that the predicted accuracy in the advanced disturbances was closer to reality than in the linear regression model that only used holdout validation accuracy. It is not sufficient to prove that the general method proposed can be used. The testing has to produce useful results. From the tests performed, there is an indication that this might be achievable. The statistical models that can be created with the data from the proposed method show some promising results.

When predicting what model that would handle unseen disturbances best (advanced disturbances) based on one linear regression model using only the holdout validation accuracy, and one linear regression model using all the information from the proposed method. The general linear regression model that used data from the proposed method had a similar performance to the linear regression model that only used data from the holdout validation method. This can be attributed mostly to the fact that the linear regression we used to predict the results tried to minimize the error in the prediction, not trying to maximize the correctness while comparing two models. Ideally, a better evaluation method should have been used. Using linear

regression and minimizing the error is likely not the best method to compare the different statistical models.

6.5 Viability and Research Questions

Based on the results, we can see that there is some benefit to performing the proposed method, but there are also many drawbacks. We can see that the values obtained from the proposed method can build different linear regression models that predict the accuracy of different kinds of disturbances. The linear regression model using all the data from the proposed method have lower error when predicting the accuracy in advanced disturbances, compared to only using the accuracy from holdout validation. The problem is that the linear regression models can only be accurately constructed after the disturbances are known, this defeats the purpose of the general solution proposed.

With that said, the evaluation of the method also shows that it might be possible to create a robust method that can predict the stability of a machine learning model in unknown disturbances. But to do this, a wide array of automatic elementary tests would have to be constructed and implemented, and if a more general solution was constructed, it would also require more testing on different machine learning models. It is not outside of the realm of possibility that a solution similar to the one proposed in the thesis can give key information when it comes to testing different machine learning models. For example, it was shown that already in the current state that the proposed method was more accurate than basic holdout validation.

When it comes to the question if it is possible to predict the performance in unseen disturbances only using simple elementary disturbances the results seem positive. It is hard to say to what extent the added information from the proposed method can be helpful. The results clearly indicate it is possible. The results show clear correlations even in the most simplistic form of correlation, and the linear regression models all have better results than the corresponding linear regression models without the information from the proposed method. The question if there is any value in a method like this that is not something that can be easily answered at this stage. The method proposed is merely to show that there is scientific value in researching similar methods. A methodology similar to this could definitely be useful, but at the current state, it is not developed enough.

The methods explored should not be used to get a false sense of security or reassurance of a working system. Results from these methods should be used as a tool for understanding and improvement. When evaluating safety-critical solutions, a deeper, more thorough evaluation has to be done than the methods explored in this thesis. The proposed method could be effective in many cases, but it will never replace case-specific testing. It is merely a way to get more insight and help during the development process.

Looking at the research questions, the individual answers are:

1. **Is it possible to generate elementary disturbances and use them for testing?**
   Yes, but usefulness is case-specific and the benefits vary depending on the situation.

2. **Is it possible to perform testing while handling a machine learning model as a "Black Box"?**
   For the proposed method this is possible. Modifying input and looking at the output is sufficient, it is all the proposed method needs.

3. **Is it possible to create a general solution for evaluating different machine learning models and different types of data?**
   Different machine learning models: We tested one type of machine learning model, but it should work for all models used for classification of labeled data
   Different types of data: Yes

4. **Is it possible to use simple elementary disturbances to predict the performance on unseen disturbances?**
   Unclear, more research is needed

6.6 Problems with current evaluation

A flaw in only evaluating the method using mainly linear regression and correlation is the problem of overlapping correlations; for example, if one of the automatic tests correlates very well with the holdout validation accuracy and an advanced disturbance at the same time, the correlation can be overlapping. Overlapping meaning that the same information that the automatic elementary disturbance test brings can be extrapolated from only looking at the holdout validation accuracy. That would mean that the automatic simple disturbance tests would not give any additional information to the model even if it had a correlation with the advanced unseen disturbances due to the fact that all that information could be extrapolated from the holdout testing accuracy on its own.

Another problem is that many of the general elementary disturbances are so general and diffuse that even if you can get a result, it is hard to get any real insight into how you can improve the model. Therefore it is hard to understand the real weakness of the model. Without understanding the weakness completely, it is hard to gain insight into how to or what to improve to get a better result and a more robust machine learning model. It is also difficult to know what real-world data that would be good to train more on.

Because of the limitations of the simple elementary disturbances, there could be some value in developing more case-specific disturbances. We can see that maybe a library of data-dependent disturbances for testing would be a better approach. This

would, on the other hand, make the solution less general, which is a disadvantage. A general method is by default less likely to give the best possible results. It is, however, the easiest to apply to different situations. Further research into specific types of disturbance for a certain type of data could be worth doing.

The proposed method can give more insight into the robustness of a machine learning model. But like with most cases in machine learning it is not recommended to trust the solutions completely, different disturbances can be handled in unpredictable ways, and it is impossible to account for all plausible situations that can occur in the wild. Having more insight into the robustness and more values to do an accurate analysis of the models can still be valuable, it can be used as an analytical way of deciding the best models. It can also guide the developers when improving the machine learning models.

6.7 Interpreting saved data

Interpreting saved data refers to the results of the testing results in 5.1. One thing worth noting is that the results from this example are just from one example, training the same networks different times could give different results. The results from a test have to be interpreted manually and the usefulness would depend on the situation and the person evaluating a system.

Looking at the new distributions of classifications from tables A.10-A.18, all the original classifications are almost uniformly distributed around at 10% for each of the 10 classifications. This is expected since the data is randomly sampled. The classifications, after adding disturbances, however, are for the most part not uniformly distributed in the test performed. See Table 5.1 or the more detailed breakdown in Tables A.10-A.18 to see this. After adding disturbances, the classifications change. The distribution refers to the distribution of new (wrong) classifications. How these classifications change is dependant on how the model interprets the data with disturbances. The data in Table 5.1 and Tables A.10-A.18 are from testing an individual model.

For the exponential mapping increase of elements disturbance 3.3.8, the images are shifted towards a lighter shade. The data tested is the MNIST dataset, where the data is images of black and white hand-drawn numbers. Since most of the pixels are completely white or completely black, there is no major change when applying the exponential increase. Looking at the new distribution of classifications, they generally stay the same, see Table A.6. It is predictable that the classifications generally stays the same since not many elements (pixels) are changed. The same phenomenon can be seen for the exponential mapping decrease of elements (3.3.7) disturbance, see Table A.7. In this case, the decrease, or darkening, also seem to have little effect on the images' new classification.

Table A.11 has the distribution of new values and the old values from the random

disturbance (3.3.6.1). It can be seen that a shift to the classifications 2 and 3 are common when randomly modifying elements in the input vector. This implies that random noise added to input, for this neural network, shifts classifications towards the numbers 2 and 3. In a sense, this could be interpreted as that the 2 and 3 classifications are the weakest for this type of disturbance. When a disturbance is added, almost all images change the classification to either 2 or 3.

All the disturbances changing individual elements, i.e., random (3.3.6.1), setting random to 1 (3.3.6.2), and setting random to 0 (3.3.6.3), have a significant shift in classifications. Especially the random, and the setting random elements to 1, have a changed distribution of new classifications. Almost all of the new classifications after applying these disturbances fall into the "2" or "3" classification. The smallest amount of disturbance has to be made to change the classifications to the "2" and "3" classifications. This is why these are over-represented in the new, wrong classifications.

Looking into classifications that are over-represented after disturbances could give some insight into possible alterations that could be made to improve a software system that uses classification. One interesting experiment that could be done in this case would be to remove the "2" and "3" classifications from the data. The "2" and "3" classifications could also be classified manually. This should, in theory, increase the average amount of disturbances that can be added before a wrong classification is made. If, for example, the 2 and 3 classifications here would be classified manually, it is possible that the other classifications could handle more disturbance before the machine learning model misclassifies the data. Tests like these were not performed, and it could give some insight into how the data used could be used to give more information. It could indicate how useful the methods are, in terms of manually looking at the saved data. In a larger system, there could be a situation where a machine learning model has a lower accuracy for identifying a specific classification, and it would be useful to do that classification manually or through another method. It is unclear whether the reasoning from the simple examples holds true for more complex types of data that could be tested.

It is arguable if this type of data is useful, likely, it sometimes would be and sometimes would not be. There could be multiple ways to interpret the saved data. Both looking at the data with disturbance and the distribution of classifications after adding disturbances could be useful in some cases, gathering more information should in most cases be better. Having information about different weaknesses is more useful than having a simple accuracy metric. In a safety-critical, like self-driving, seeing shifts in classifications could be useful. If there is a system identifying colors of traffic lights, for example, seeing how many red lights are wrongfully classified as green lights is likely more useful than seeing how many yellow lights are identified as green lights. Seeing the total distribution of classifications, in this case, could also be useful. Incorrectly classifying the lights as red might be better than incorrectly classifying them as green. It most likely would be better if a self-driving system thinks a green light is red rather than the other way around.

6.8 Threats to validity

6.8.1 Internal validity

Threat 1 - It could be argued that some of the testing methodologies are similar to the disturbances themselves. Testing overlapping sound, for example, in audio, is mostly similar to the fade disturbance. This could potentially mean that some of the advanced disturbance inherently are correlated to the simple disturbances tested. This would partly invalidate the ability to predict advanced unseen disturbances with the simple disturbances.

Threat 2 The effects measured was conducted in such a way that the statistical analysis occurred after the advanced data was tested. This creates a dilemma due to the fact that the statistical models were created after collecting data. It is easy to improve a statistical model when adding more variables.

Threat 3 - The advanced disturbances created, to simulate realistic disturbances, might not represent real-world disturbances. With other disturbances, the results may differ. The disturbances might not reflect real world situations, this would partly invalidate the point-of the statistical analysis.

Threat 4 The viability of the proposed method was mainly shown with linear correlation and linear regression models. This however might not be an optimal way to evaluate the proposed method. The weakness of linear regression is shown in the results. The raw results from the tests are available [29], and other evaluation methods could be explored.

6.8.2 External validity

Threat 1 - This method is not reliable for all situations. Different scenarios have different circumstances, and not all methods are suitable for all situations. For reliable results, any method should be used in combination with other evaluation methods if a thorough evaluation is wanted.

Threat 2 - It is unclear whether the methods described are useful and accurate in real-world use cases. The tests performed in this thesis should be considered lab testing; they were all using simple machine learning models and simple types of data. It is unclear whether the results would be similar with more complex types of data.

6.9 Future studies

The methods described are broad and meant to work as a tool to help while doing a general analysis, independent of what type of machine learning model is used, and no matter what type of data is used. A problem with focusing on the broader picture is that less focus is given to the smaller details. There are a few improvements that could be made to make this type of testing easier to use and more adaptable.

6.9.1 Usability

In terms of using the created code base for analysis, there could be some usability improvements. The code is meant to work with any type of data and data-structure that can be converted to a one-dimensional vector. The analysis can currently be performed for all types of data, and numbers from the analysis can be given. The visualization, however, is currently done manually for each type of data tested. There could be some improvement by saving output data and examples in the same format that they were taken in.

6.9.2 Separation of outliers

When calculating the average of the added disturbances, there is a potential improvement that could be made. With the current code structure, if the maximum level of a disturbance is added and the classification remains the same, that level is included in the calculation of the average, even though the classification does not change. Ideally, the group where maximum levels are added without a change in classification should be analyzed separately, or removed from the results. The interest is incorrectly classified data that changes classification by added disturbances. The data that is not changed by maximizing disturbances should be treated separately since it is not in that group of data. An example of this can be seen in Table A.6 and Table A.7 in the Appendix, where most of the classifications remain the same after disturbances are added.

6.9.3 Disturbances

Currently, the different disturbances on input data are tested individually. Having combinations of different disturbances could be worth exploring.

Another way to expand on the methods described is to add more disturbances. When creating new disturbances, the same basic principles should be followed. This means that the disturbances created should work regardless of what type of data is used. The input is converted to a one-dimensional input vector, so any domain-specific knowledge can not be used since all the input is flattened. It is also recommended that the disturbances are simple and modify each elementary element individually. In theory, this should keep it so that the disturbances are general, and be useful for any machine learning model and any data.

Two additional disturbances were started but not finished due to extensive training time and limited benefits over the more simple disturbances:

6.9.3.1 Auto encoder fade

While the compression is often not lossless, it can be an efficient way to reconstruct data. One paper suggests the use of variational autoencoders [33] to test neural networks. Autoencoders could create a more natural transition between classifications in the input. This effect is done by fading between autoencoder representations, rather than fade directly in the data (like the simple fade disturbance in 3.3.4.4).

A disadvantage is that an auto-encoder has to be trained. This can take a significant amount of time, depending on the data. For some domains, depending on the computational resources available, an autoencoder compression might not give a sufficiently accurate representation. The code used for the auto-encoder was based on an open-source example [34]. The main reason autoencoders were not used is the necessary training of them. The performance of the autoencoder would likely depend on which type of data used. It is unlikely that a consistent method could be developed without adapting the autoencoder for each type of input data.

Figure 36: An example of an 7 being faded into a 5 using an implementation of an autoencoder

6.9.3.2 Stochastic optimized disturbances

To find a subset of input nodes that have the most effect on the outcome and the result of the algorithm, an optimization algorithm can be used. The benefit of this is that an arbitrary number can be used, comparing how stable different algorithms are, based on how easily they can be manipulated into giving a faulty result. For example, if only 10% of all input nodes are changed with disturbances, how accurate will the algorithm be?

Given that the proposed method should work for any type of data and any type of machine learning model, a general algorithm is needed for optimization. One option would be to use a genetic algorithm. Using a genetic algorithm with each of the genome representing a level of disturbance on every input node, the approach can be scaled without needing any other fixes. The solution can, therefore, work on a magnitude of different machine learning approaches as long as the input is a vector of elements. The problem is for more complex input data, but if the same generalization is used as mentioned in section 3.3.3 this would not be a problem. Given the fact that the algorithm needs to optimize itself for each machine learning

model individually and that the data set it needs to train on will be rather large, the optimization will take a long time. It was, therefore, discarded in this thesis but could prove beneficial in future studies.

7
Conclusion

In this thesis, the benefit of using modified data for evaluating the performance of trained machine learning models is explored. The data is modified using simple general elementary disturbances so that it works for different types of data and different types of machine learning models. The results are looked at manually and the benefit of this approach is explored using statistical analysis, consisting of mainly linear correlation, and linear regression models.

It is possible to create many different elementary disturbances and use them for testing. All the created disturbances can be seen in the method section. There is some statistical evidence that the elementary disturbances can be a good indication of how well a machine learning model can resist unseen disturbances. It is possible to see a correlation between the elementary disturbances and unseen more complex data-dependent disturbances. Even if the correlations are not very strong when using simple linear correlation it is easy to conclude that the simple elementary disturbances give more information than only using holdout validation accuracy as a lone metric. The methods described show some potential, and it would be worth further exploring how these could be applied. There is some commentary about potential ways to further this research in the discussion chapter.

To have a generalized testing methodology, any type of data should be able to be used. With the proposed method, the input data can successfully be handled with no knowledge of the structure of the data. This can be done by generalizing the data to a one-dimensional input vector. The testing results show that both audio and images can be used. The same method of generalization should also work for other types of data. For a general testing methodology, any type of machine learning model should also be able to be used. This was not tested but is worth exploring. For testing, solely convolutional neural networks are used. Additional research is required to evaluate if the proposed method works for other types of machine learning models.

In terms of the general analysis and the data saved, see section 3.5, the usefulness is hard to assess. This type of manual evaluation is more subjective than statistical analysis. It is hard to say how useful this kind of analysis would be for general evaluation. Notably, there are significant differences when it comes to relatively similar types of disturbances, and the method can both have very high usefulness or barely any at all depending on small factors. The usefulness would vary depending on the situation and the preferences of the people implementing the software

system. One disadvantage with the proposed methods is that manual evaluation and significant computational power is required, compared to just looking at traditional holdout validation results. It most likely depends on the situation, whether it is worth the extra time or computational power to perform the methods described.

From the results, it can be seen that simple elementary disturbances most likely can be used as an indicator of how well a machine learning model will handle unseen disturbances. From a scientific perspective, the results were promising and show that there is a reason for continuing researching methods constructed in similar ways. In the current state, the developed method is not practical enough to be used, and more research and development is needed.

When it comes to the viability of the method, there is no clear conclusion. The methodology works well in the sense that it is general. The method should work with any type of machine learning model and any type of data. A general method is unlikely the best evaluation method for specific situations. In most situations, task-specific testing likely gives the best results. A general method could, however, be easy to use and be effective to use as a complement to more case-specific methods. Further research, specifically trying more machine learning models, would help give a better indication of the viability of the methods described.

Bibliography

[1] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. MÃ¼ller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLOS ONE*, vol. 10, no. 7, pp. 1–46, 07 2015. [Online]. Available: https://doi.org/10.1371/journal.pone.0130140

[2] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013. [Online]. Available: https://arxiv.org/abs/1312.6034

[3] Y. Sun, X. Huang, and D. Kroening, "Testing deep neural networks," *CoRR*, vol. abs/1803.04792, 2018. [Online]. Available: http://arxiv.org/abs/1803.04792

[4] S. H. Huang, N. Papernot, I. J. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *CoRR*, vol. abs/1702.02284, 2017. [Online]. Available: http://arxiv.org/abs/1702.02284

[5] "The cifar-10 dataset," https://www.cs.toronto.edu/~kriz/cifar.html, accessed: 2020-03-14.

[6] R. V. Yampolskiy, "What are the ultimate limits to computational techniques: verifier theory and unverifiability," *Physica Scripta*, vol. 92, no. 9, p. 093001, jul 2017. [Online]. Available: https://doi.org/10.1088%2F1402-4896%2Faa7ca8

[7] J. Vanus, M. Smolon, R. Martinek, J. Koziorek, J. Zidek, and P. Bilik, "Testing of the voice communication in smart home care." *springer*, 2015. [Online]. Available: https://doi.org/10.1186/s13673-015-0035-0

[8] Smith, N.T., Brien, R.A., Pettus, D.C. et al., "Recognition accuracy with a voice-recognition system designed for anesthesia record keeping," *springer*, 1990. [Online]. Available: https://doi.org/10.1007/BF02842489

[9] M. Hossin and M. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International Journal of Data Mining & Knowledge Management Process*, vol. 5, no. 2, p. 1, 2015.

[10] K. Allix, T. F. Bissyandé, Q. Jérome, J. Klein, Y. Le Traon *et al.*, "Empirical assessment of machine learning-based malware detectors for android," *Empirical Software Engineering*, vol. 21, no. 1, pp. 183–211, 2016.

[11] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, 2019.

[12] "THE MNIST DATABASE of handwritten digits," http://yann.lecun.com/exdb/mnist/, accessed: 2020-03-02.

[13] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833. [Online]. Available: https://arxiv.org/pdf/1311.2901.pdf

[14] V. Tjeng and R. Tedrake, "Verifying neural networks with mixed integer programming," *CoRR*, vol. abs/1711.07356, 2017. [Online]. Available: http://arxiv.org/abs/1711.07356

[15] D. Cire¸san, U. Meier, J. Masci, and J. Schmidhuber, "Multi-Column Deep Neural Network for Traffic Sign Classification," 2012. [Online]. Available: https://doi.org/10.1016/j.neunet.2012.02.023

[16] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy (SP)*, 2016, pp. 582–597. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7546524

[17] J. Kim, R. Feldt, and S. Yoo, "Guiding Deep Learning System Testing using Surprise Adequacy," 2018. [Online]. Available: https://arxiv.org/abs/1808.08444

[18] P. M. Htut and J. R. Tetreault, "The unbearable weight of generating artificial errors for grammatical error correction," *CoRR*, vol. abs/1907.08889, 2019. [Online]. Available: http://arxiv.org/abs/1907.08889

[19] Google, "Recommendations: What and why?" https://developers.google.com/machine-learning/recommendation/overview, accessed: 2020-05-24.

[20] Y. Xu, N. Chen, A. Fernandez, O. Sinno, and A. Bhasin, "From infrastructure to culture: A/b testing challenges in large scale social networks," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 2227–2236. [Online]. Available: https://doi.org/10.1145/2783258.2788602

[21] Tesla, "Tesla autonomy day," https://youtu.be/Ucp0TTmvqOE, 2019.

[22] comma ai, "Crowdsourced Segnet (you can help!)," 2020. [Online]. Available: https://medium.com/@comma_ai/crowdsourced-segnet-you-can-help-2e683244a039

[23] "The urbansound8k dataset," https://urbansounddataset.weebly.com/urbansound8k.html, accessed: 2020-05-02.

[24] F. Zalkow, "Create audio spectrograms with python." [Online]. Available: http://www.frank-zalkow.de/en/code-snippets/create-audio-spectrograms-with-python.html?i=1

[25] J. L. Bentley and R. Sedgewick, "Fast algorithms for sorting and searching strings," in *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, 1997, pp. 360–369.

[26] M. R. Hasan, M. Jamil, M. G. Rabbani, and M. S. Rahman, "Speaker identification using mel frequency cepstral coefficients," https://www.researchgate.net/profile/Golam_Rabbani4/publication/255574793_Speaker_Identification_Using_Mel_Frequency_Cepstral_Coefficients/links/55f05d5908ae0af8ee1d1894.pdf.

[27] "Jasp," https://github.com/jasp-stats/jasp-desktop.

[28] A. Damien, "Convolutional neural network example," https://github.com/aymericdamien/TensorFlow-Examples/blob/master/tensorflow_v2/notebooks/3_NeuralNetworks/convolutional_network_raw.ipynb, accessed: 2020-02-28.

[29] "Raw testing results," https://zenodo.org/record/3902373#.Xu5nhmgzYuU.

[30] T. Kurita, M. Pic, and T. Takahashi, "Recognition and detection of occluded faces by a neural network classifier with recursive data reconstruction," in *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance, 2003.* IEEE, 2003, pp. 53–58.

[31] T. Andersson, L. Blomkvist, A. Hast, F. Karlsson, J. Lindström, and T. Sundell, "Very low bandwidth (marine) web surfing a fault-tolerant content streaming web browsing solution," B.S. thesis, 2018.

[32] R. Artusi, P. Verderio, and E. Marubini, "Bravais-pearson and spearman correlation coefficients: meaning, test of hypothesis and confidence interval," *The International journal of biological markers*, vol. 17, no. 2, pp. 148–151, 2002.

[33] S. Kang, R. Feldt, and S. Yoo, "Sinvad: Search-based image space navigation for dnn image classifier test input generation," https://arxiv.org/abs/2005.09296.

[34] A. Damien, "Convolutional neural network example," https://github.com/aymericdamien/TensorFlow-Examples/blob/master/tensorflow_v2/notebooks/3_NeuralNetworks/autoencoder.ipynb, accessed: 2020-02-26.

A

Appendix 1

These following tables show the original classifications in the first column, and what their new classifications when adding random elements. For example, in table A.2, 84% of the 977 images originally classified as 0, are classified as a 2 after the disturbance is added. The values are rounded to the nearest percent. If there are no cases, the symbol (-) is used. To see even more in depth data, like disturbance level required to change classifications, the modified data can be looked at manually. See examples in 5.1 for this. Automatic generation of tables for disturbance levels were not implemented but can be looked at individually for each datapoint.

Table A.1: Showing the initially wrong classifications (removed from dataset)

Original classification	Now 0	Now 1	Now 2	Now 3	Now 4	Now 5	Now 6	Now 7	Now 8	Now 9
0 (n = 3)	-	-	-	-	-	33%	-	33%	33%	-
1 (n = 10)	-	-	20%	10%	-	-	10%	20%	40%	-
2 (n = 12)	-	8%	-	-	-	-	-	67%	25%	-
3 (n = 11)	9%	-	9%	-	-	36%	-	9%	9%	27%
4 (n = 8)	-	-	13%	-	-	-	-	13%	13%	63%
5 (n = 9)	22%	-	-	22%	-	-	22%	-	-	33%
6 (n = 20)	45%	10%	-	-	5%	40%	-	-	-	-
7 (n = 13)	-	8%	15%	-	-	8%	-	-	15%	54%
8 (n = 36)	33%	-	8%	3%	11%	11%	3%	8%	-	22%
9 (n = 16)	6%	19%	-	6%	38%	-	-	25%	6%	-

Table A.2: New classifications after adding a (random) distortion

Original classification	Now 0	Now 1	Now 2	Now 3	Now 4	Now 5	Now 6	Now 7	Now 8	Now 9
0 (n = 977)	-	-	84%	3%	1%	3%	2%	0%	1%	5%
1 (n = 1125)	-	-	68%	22%	1%	2%	0%	2%	4%	-
2 (n = 1020)	-	-	-	99%	0%	0%	-	0%	0%	-
3 (n = 999)	-	-	95%	1%	-	2%	-	-	2%	-
4 (n = 974)	-	0%	45%	41%	-	2%	1%	0%	5%	5%
5 (n = 883)	-	0%	21%	68%	0%	-	0%	-	10%	1%
6 (n = 938)	0%	0%	63%	16%	5%	9%	-	-	6%	0%
7 (n = 1015)	-	1%	35%	54%	1%	3%	-	-	1%	5%
8 (n = 938)	0%	0%	51%	45%	0%	3%	0%	0%	-	0%
9 (n = 993)	0%	1%	14%	46%	26%	4%	-	5%	4%	-

Table A.3: New classifications after adding a (Setting random elements to 1) distortion

Original classification	Now 0	Now 1	Now 2	Now 3	Now 4	Now 5	Now 6	Now 7	Now 8	Now 9
0 (n = 977)	-	-	84%	4%	1%	2%	2%	1%	1%	6%
1 (n = 1125)	-	-	67%	19%	3%	2%	0%	4%	5%	-
2 (n = 1020)	-	-	23%	77%	-	-	-	0%	0%	-
3 (n = 999)	-	-	99%	-	-	1%	-	0%	0%	0%
4 (n = 974)	-	0%	53%	37%	-	3%	0%	0%	3%	4%
5 (n = 883)	0%	0%	23%	64%	0%	-	-	0%	11%	1%
6 (n = 938)	0%	0%	66%	10%	8%	10%	-	-	5%	0%
7 (n = 1015)	-	1%	46%	46%	1%	2%	-	-	0%	4%
8 (n = 938)	0%	0%	59%	37%	1%	2%	-	0%	-	0%
9 (n = 993)	-	0%	17%	42%	25%	4%	-	7%	3%	-

Table A.4: New classifications after adding a (Setting random elements to 0) distortion

Original classification	Now 0	Now 1	Now 2	Now 3	Now 4	Now 5	Now 6	Now 7	Now 8	Now 9
0 (n = 977)	-	-	50%	3%	1%	27%	1%	6%	11%	2%
1 (n = 1125)	-	-	6%	34%	3%	30%	0%	3%	18%	6%
2 (n = 1020)	-	8%	1%	35%	6%	29%	0%	17%	2%	1%
3 (n = 999)	-	3%	17%	-	1%	50%	-	27%	0%	3%
4 (n = 974)	0%	1%	10%	7%	0%	35%	0%	36%	4%	7%
5 (n = 883)	0%	23%	10%	37%	5%	2%	0%	19%	0%	4%
6 (n = 938)	1%	0%	43%	5%	5%	43%	-	3%	2%	-
7 (n = 1015)	0%	9%	7%	40%	3%	34%	0%	1%	1%	3%
8 (n = 938)	-	-	6%	41%	0%	47%	-	1%	-	4%
9 (n = 993)	0%	0%	3%	34%	9%	21%	-	28%	5%	-

Table A.5: New classifications after adding a (fade) distortion

Original classification	Now 0	Now 1	Now 2	Now 3	Now 4	Now 5	Now 6	Now 7	Now 8	Now 9
0 (n = 977)	0%	8%	13%	7%	3%	6%	12%	9%	13%	27%
1 (n = 1125)	10%	-	11%	9%	14%	7%	9%	10%	25%	6%
2 (n = 1020)	12%	11%	0%	16%	7%	2%	9%	14%	21%	8%
3 (n = 999)	7%	9%	14%	0%	4%	12%	4%	12%	15%	24%
4 (n = 974)	4%	10%	9%	4%	0%	4%	11%	11%	12%	34%
5 (n = 883)	7%	7%	2%	17%	6%	0%	8%	3%	25%	26%
6 (n = 938)	15%	8%	12%	5%	19%	14%	0%	3%	17%	6%
7 (n = 1015)	10%	10%	14%	15%	9%	3%	3%	0%	8%	26%
8 (n = 938)	10%	13%	11%	13%	7%	9%	8%	6%	0%	22%
9 (n = 993)	12%	6%	6%	14%	19%	10%	5%	15%	14%	0%

Table A.6: New classifications after adding a (exponential mapping increase of elements) distortion

Original classification	Now 0	Now 1	Now 2	Now 3	Now 4	Now 5	Now 6	Now 7	Now 8	Now 9
0 (n = 977)	100%	-	-	-	-	-	-	-	0%	-
1 (n = 1125)	0%	96%	0%	-	-	-	-	0%	3%	-
2 (n = 1020)	0%	0%	99%	0%	0%	-	-	0%	0%	-
3 (n = 999)	-	-	0%	98%	-	1%	-	0%	1%	0%
4 (n = 974)	-	-	0%	-	97%	-	0%	-	0%	2%
5 (n = 883)	-	-	-	0%	-	99%	-	-	1%	0%
6 (n = 938)	0%	0%	-	-	-	-	100%	-	-	-
7 (n = 1015)	-	-	0%	0%	-	-	-	97%	0%	2%
8 (n = 938)	1%	-	-	0%	0%	0%	0%	-	99%	0%
9 (n = 993)	-	-	-	-	-	-	-	-	0%	100%

Table A.7: New classifications after adding a (exponential mapping decrease of elements) distortion

Original classification	Now 0	Now 1	Now 2	Now 3	Now 4	Now 5	Now 6	Now 7	Now 8	Now 9
0 (n = 977)	91%	-	3%	1%	-	3%	0%	1%	1%	1%
1 (n = 1125)	-	97%	0%	0%	0%	2%	-	0%	0%	0%
2 (n = 1020)	0%	0%	94%	2%	0%	1%	-	1%	1%	0%
3 (n = 999)	-	-	1%	96%	-	2%	-	1%	-	0%
4 (n = 974)	-	-	0%	0%	94%	1%	-	3%	-	1%
5 (n = 883)	-	0%	0%	2%	0%	96%	-	0%	-	0%
6 (n = 938)	0%	0%	2%	0%	1%	6%	90%	0%	0%	-
7 (n = 1015)	-	0%	0%	1%	-	1%	-	97%	-	1%
8 (n = 938)	-	0%	2%	9%	1%	11%	0%	0%	74%	3%
9 (n = 993)	-	0%	0%	1%	3%	3%	-	7%	-	86%

Table A.8: New classifications from Taking max elements from two data points

Original classification	Now 0	Now 1	Now 2	Now 3	Now 4	Now 5	Now 6	Now 7	Now 8	Now 9
0 (n = 977)	69%	-	4%	2%	1%	1%	3%	1%	6%	14%
1 (n = 1125)	12%	2%	10%	11%	10%	8%	10%	8%	18%	10%
2 (n = 1020)	8%	0%	57%	6%	2%	1%	3%	3%	15%	5%
3 (n = 999)	8%	0%	8%	48%	1%	6%	3%	1%	11%	15%
4 (n = 974)	5%	0%	6%	6%	28%	3%	5%	6%	14%	27%
5 (n = 883)	8%	0%	3%	8%	2%	39%	4%	1%	20%	15%
6 (n = 938)	14%	0%	7%	4%	8%	6%	41%	4%	10%	5%
7 (n = 1015)	10%	1%	9%	12%	3%	4%	3%	26%	12%	20%
8 (n = 938)	9%	0%	5%	4%	1%	2%	4%	1%	68%	7%
9 (n = 993)	5%	0%	6%	6%	5%	4%	4%	3%	12%	55%

Table A.9: New classifications from Taking min elements from two data points

Original classification	Now 0	Now 1	Now 2	Now 3	Now 4	Now 5	Now 6	Now 7	Now 8	Now 9
0 (n = 977)	13%	6%	13%	9%	5%	28%	4%	19%	1%	3%
1 (n = 1125)	1%	44%	8%	14%	6%	17%	0%	10%	1%	1%
2 (n = 1020)	2%	11%	37%	15%	6%	10%	2%	16%	1%	0%
3 (n = 999)	1%	7%	5%	49%	6%	15%	1%	14%	0%	2%
4 (n = 974)	1%	6%	5%	5%	46%	10%	1%	22%	1%	2%
5 (n = 883)	0%	3%	2%	15%	4%	64%	1%	8%	-	2%
6 (n = 938)	5%	8%	17%	10%	15%	17%	12%	11%	2%	3%
7 (n = 1015)	1%	4%	4%	11%	4%	11%	0%	64%	0%	1%
8 (n = 938)	2%	11%	10%	18%	9%	20%	3%	12%	7%	8%
9 (n = 993)	1%	4%	5%	17%	14%	16%	0%	26%	0%	17%

Table A.10: Classification distribution from initially wrong classifications (the percentages here are rounded to 0 because they are based on the total amount of images i.e. 10000 images).

	0	1	2	3	4	5	6	7	8	9
original classifications	3	10	12	11	8	9	20	13	36	16
new classifications	25	7	9	5	11	18	4	20	13	26
original classification part	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
new classification part	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%

Table A.11: Classification distribution from (random) disturbance

	0	1	2	3	4	5	6	7	8	9
original classifications	977	1125	1020	999	974	883	938	1015	938	993
new classifications	4	19	4726	3871	349	294	25	87	326	161
original classification part	10%	11%	10%	10%	10%	9%	9%	10%	9%	10%
new classification part	0%	0%	47%	39%	3%	3%	0%	1%	3%	2%

Table A.12: Classification distribution from (Setting random elements to 1) disturbance

	0	1	2	3	4	5	6	7	8	9
original classifications	977	1125	1020	999	974	883	938	1015	938	993
new classifications	5	20	5319	3293	389	250	20	130	292	144
original classification part	10%	11%	10%	10%	10%	9%	9%	10%	9%	10%
new classification part	0%	0%	53%	33%	4%	3%	0%	1%	3%	1%

Table A.13: Classification distribution from (Setting random elements to 0) disturbance

	0	1	2	3	4	5	6	7	8	9
original classifications	977	1125	1020	999	974	883	938	1015	938	993
new classifications	13	418	1462	2350	325	3140	17	1392	451	294
original classification part	10%	11%	10%	10%	10%	9%	9%	10%	9%	10%
new classification part	0%	4%	15%	24%	3%	31%	0%	14%	5%	3%

Table A.14: Classification distribution from (fade) disturbance

	0	1	2	3	4	5	6	7	8	9
original classifications	977	1125	1020	999	974	883	938	1015	938	993
new classifications	862	811	895	977	893	674	679	835	1483	1753
original classification part	10%	11%	10%	10%	10%	9%	9%	10%	9%	10%
new classification part	9%	8%	9%	10%	9%	7%	7%	8%	15%	18%

Table A.15: Classification distribution from (exponential mapping increase of elements) disturbance

	0	1	2	3	4	5	6	7	8	9
original classifications	977	1125	1020	999	974	883	938	1015	938	993
new classifications	988	1085	1021	988	949	876	938	989	989	1039
original classification part	10%	11%	10%	10%	10%	9%	9%	10%	9%	10%
new classification part	10%	11%	10%	10%	9%	9%	9%	10%	10%	10%

Table A.16: Classification distribution from (exponential mapping decrease of elements) disturbance

	0	1	2	3	4	5	6	7	8	9
original classifications	977	1125	1020	999	974	883	938	1015	938	993
new classifications	895	1110	1049	1121	966	1132	844	1121	713	911
original classification part	10%	11%	10%	10%	10%	9%	9%	10%	9%	10%
new classification part	9%	11%	10%	11%	10%	11%	8%	11%	7%	9%

Table A.17: Classification distribution from Taking max elements from two data points

	0	1	2	3	4	5	6	7	8	9
original classifications	977	1125	1020	999	974	883	938	1015	938	993
new classifications	1457	42	1171	1063	599	681	772	541	1826	1710
original classification part	10%	11%	10%	10%	10%	9%	9%	10%	9%	10%
new classification part	15%	0%	12%	11%	6%	7%	8%	5%	18%	17%

Table A.18: Classification distribution from Taking min elements from two data points

	0	1	2	3	4	5	6	7	8	9
original classifications	977	1125	1020	999	974	883	938	1015	938	993
new classifications	253	1069	1063	1628	1116	1992	232	2001	129	379
original classification part	10%	11%	10%	10%	10%	9%	9%	10%	9%	10%
new classification part	3%	11%	11%	16%	11%	20%	2%	20%	1%	4%

Table A.19: Model Summary - color shift

Model	R	R^2	Adjusted R^2	RMSE
H_0	0.000	0.000	0.000	0.044
H_1	0.910	0.828	0.823	0.019

Table A.20: ANOVA

Model		Sum of Squares	df	Mean Square	F	p
H_1	Regression	0.430	7	0.061	178.549	< .001
	Residual	0.089	260	3.440e-4		
	Total	0.519	267			

Table A.21: Linear regression coefficients

Model		Unstandardized	Standard Error	Standardized	t	p
H_0	(Intercept)	0.280	0.003		103.928	< .001
H_1	(Intercept)	0.587	0.053		10.996	< .001
	trainAcc	0.029	0.004	0.293	7.547	< .001
	avgRandom	-0.197	0.084	-0.156	-2.357	0.019
	avgRandomSet	-0.046	0.075	-0.036	-0.610	0.542
	avgRandomUnset	0.184	0.067	0.111	2.758	0.006
	avgFade	-0.111	0.110	-0.027	-1.012	0.313
	avgWhiteShift	-0.674	0.051	-0.525	-13.339	< .001
	avgBlackShift	-0.001	4.500e-4	-0.087	-2.948	0.003

The following are simplifications of the different disturbance algorithms. They refer to the interchangeable "addDisturbance" function of the pseudocode in Algorithm 1. The disturbance names are simplified, according to Table 3.1.

flattenedData ← input from addDisturbance function ;
disturbanceLevel ← input from addDisturbance function ;
randomIndexes ← input from addDisturbance function ;
output ← flattenedData ;
while *elements modified* ≤ *disturbanceLevel* **do**
    randomIndex ← next element from randomIndexes ;
    value of output at randomIndex ← 0 ≤ random value ≤ 1 ;
end

Algorithm 2: Simplified version of random disturbance

```
flattenedData ← input from addDisturbance function ;
disturbanceLevel ← input from addDisturbance function ;
randomIndexes ← input from addDisturbance function ;
output ← flattenedData ;
while elements modified ≤ disturbanceLevel do
    randomIndex ← next element from randomIndexes ;
    value of output at randomIndex ← 1 ;
end
```

Algorithm 3: Simplified version of random set disturbance

```
flattenedData ← input from addDisturbance function ;
disturbanceLevel ← input from addDisturbance function ;
randomIndexes ← input from addDisturbance function ;
output ← flattenedData ;
while elements modified ≤ disturbanceLevel do
    randomIndex ← next element from randomIndexes ;
    value of output at randomIndex ← 0 ;
end
```

Algorithm 4: Simplified version of random unset disturbance

```
flattenedData ← input from addDisturbance function ;
flattenedData2 ← input from addDisturbance function ;
disturbanceLevel ← input from addDisturbance function ;
output ← flattenedData ;
fade ← disturbanceLevel ;
while not all elements have been modified do
    value of output element = ← flattenedData element value * (1-fade) +
     flattenedData2 element value * fade ;
end
```

Algorithm 5: Simplified version of fade disturbance

```
flattenedData ← input from addDisturbance function ;
disturbanceLevel ← input from addDisturbance function ;
output ← flattenedData ;
exponent ← disturbanceLevel
while not all elements have been modified do
    value of output element ← flattenedData element value ^ exponent ;
end
```

Algorithm 6: Simplified version of blackShift disturbance

The process is identical to Algorithm 7

Algorithm 7: Simplified version of whiteShift disturbance

flattenedData ← input from addDisturbance function ;
flattenedData2 ← input from addDisturbance function ;
output ← flattenedData ;
while *not all elements have been modified* **do**
  value of output element ← max(flattenedData element value,
   flattenedData2 element value) ;
end

Algorithm 8: Simplified version of max disturbance/analysis

flattenedData ← input from addDisturbance function ;
flattenedData2 ← input from addDisturbance function ;
output ← flattenedData ;
while *not all elements have been modified* **do**
  value of output element ← min(flattenedData element value,
   flattenedData2 element value) ;
end

Algorithm 9: Simplified version of min disturbance/analysis

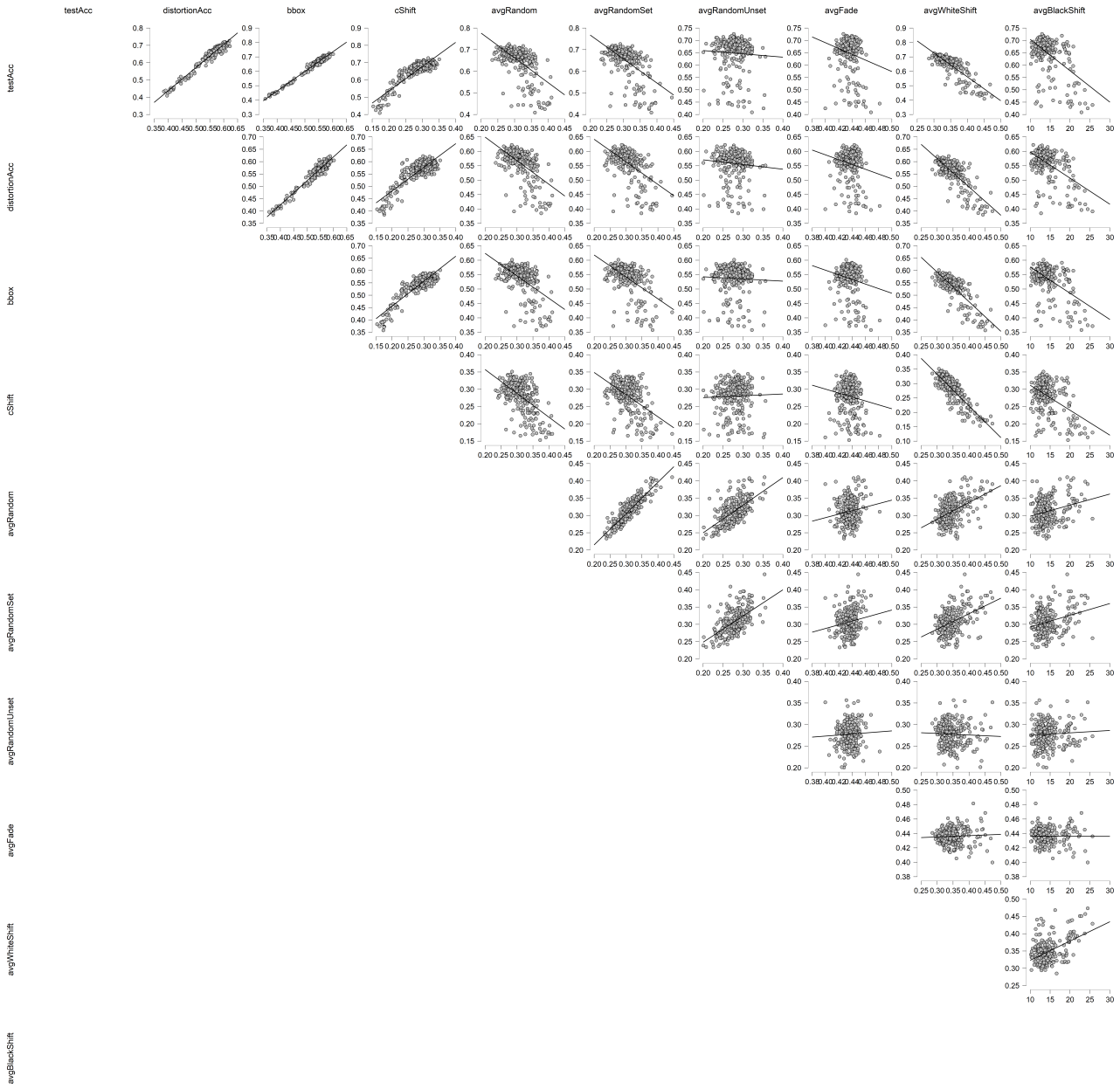


Figure 37: Image Correlation Matrix showing the correlation between all different variables in scatter plots

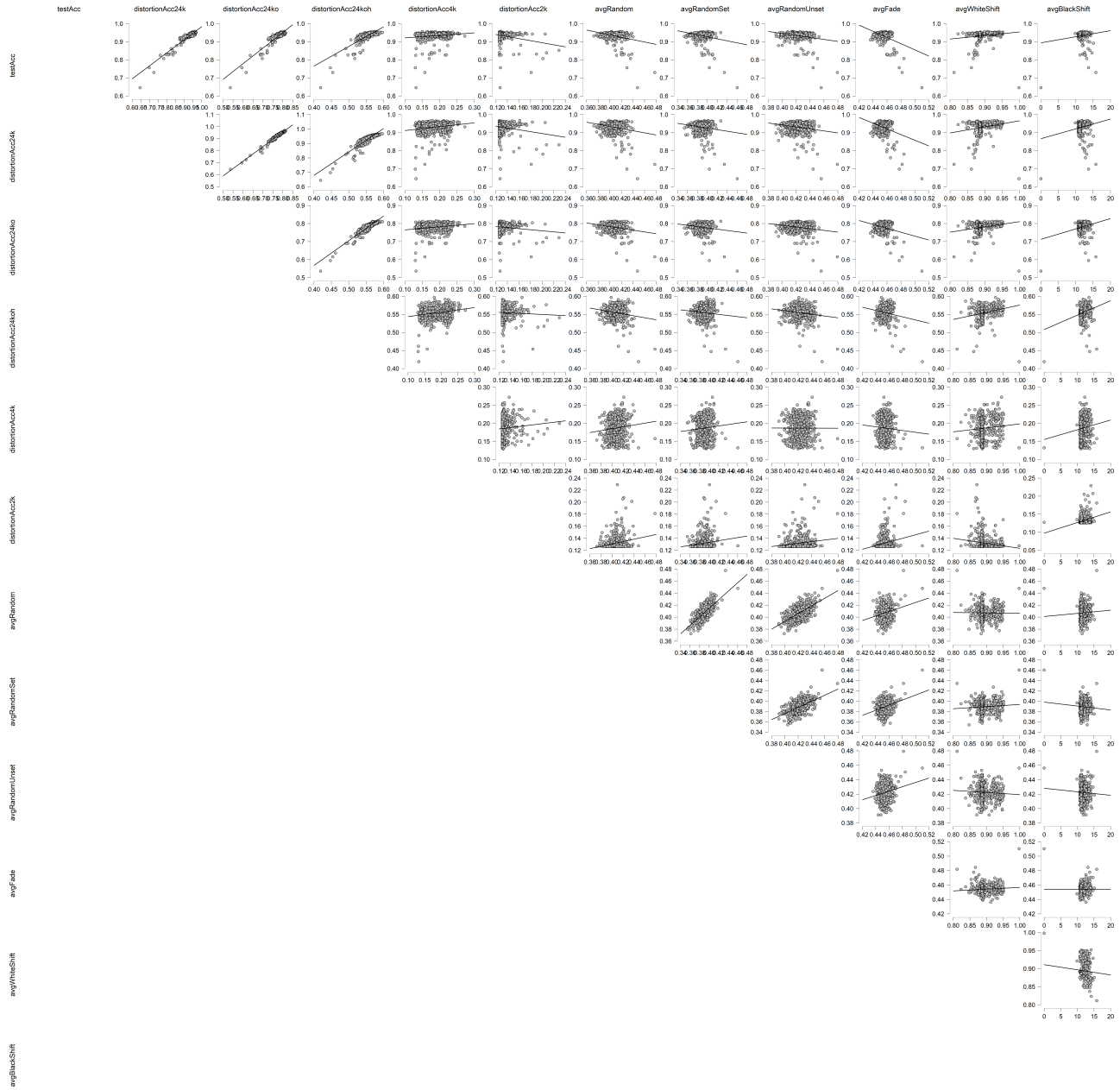


Figure 38: Audio Correlation Matrix showing the correlation between all different variables in scatter plots

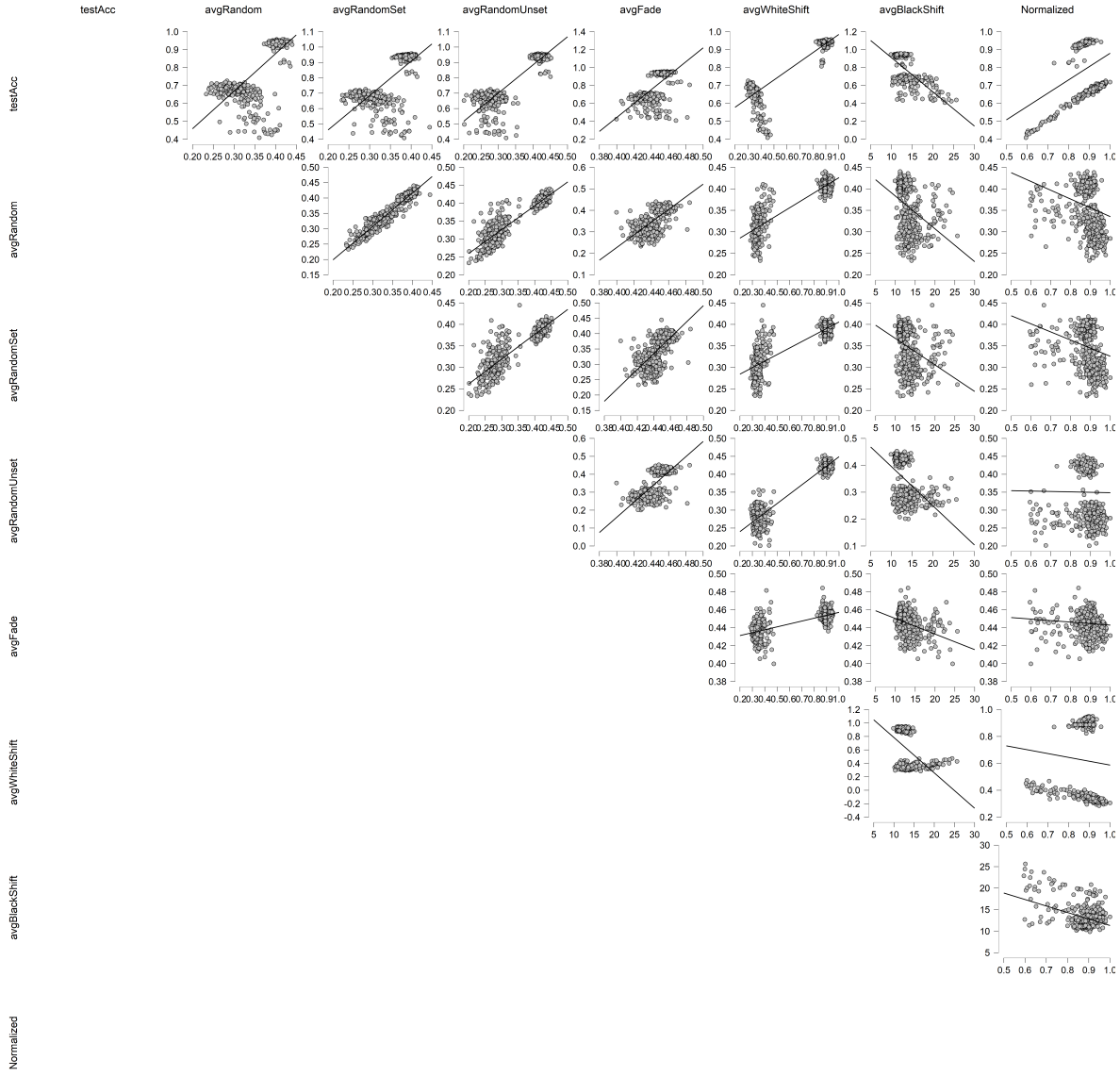


Figure 39: General Correlation Matrix showing the correlation between all different variables in scatter plots