Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Nikita Ostrovskiy

Comparison of Data Augmentation Techniques for efficient Training of Image-Text Classification Algorithms

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Informatics

Nikita Ostrovskiy

Comparison of Data Augmentation Techniques for efficient Training of Image-Text Classification Algorithms

Bachelor's thesis submitted as part of the Bachelor's examination
in the *Bachelor of Science Business Informatics*
at the Department of Computer Science
of the Faculty of Technology and Computer Science
at the University of Applied Sciences Hamburg

Supervising examiner: Prof. Dr. Tropmann-Frick
Second examiner: Prof. Dr. Schultz
Submitted on: 08 June 2022

Autor:

Nikita Ostrovskiy

Thema der Dissertation:

Vergleich von Datenerweiterungstechniken für effizientes Training von Bild-Text-Klassifikationsalgorithmen

Stichworte:

Maschinelles Lernen, CNN, Neuronales Netzwerk, Datenerweiterung, Bildverarbeitung, Bild-Text-Klassifikation

Kurze Zusammenfassung:

In diesem Beitrag werden verschiedene Methoden zur Datenerweiterung vorgestellt, die beim Training eines CNN zur Bild-Text-Klassifikation eingesetzt werden. Die Prinzipien, Konzepte, Anwendungen sowie die Effektivität dieser Methoden werden auf der Grundlage bereits durchgeführter wissenschaftlicher Arbeiten diskutiert. Dabei werden sowohl gängige Methoden als auch komplexere und noch weniger erforschte Methoden behandelt. Im experimentellen Teil dieser Arbeit wird ein CNN erstellt, das den GTSRB-Datensatz zum Erlernen der Bild-Text-Klassifikation verwendet. Einige der in dieser Arbeit besprochenen Methoden werden angewendet, um den Datensatz zu erweitern, so dass wir ihre Auswirkungen auf das Training und das Ausmaß, in dem sie bei der Aufgabe, für die sie entwickelt wurden, helfen, sehen können.

Author:
Nikita Ostrovskiy

Topic of the thesis:

Comparison of Data Augmentation Techniques for efficient Training of Image-Text Classification Algorithms

Keywords:

Machine Learning, CNN, Neural Network, Data augmentationm, Image Processing, Image-Text Classification

Short Abstract:

This paper introduces the reader to the different data augmentation methods used in training of a convolutional neural networks that conducts image-text classifcations. The principles, concepts, applications, as well as the effectiveness of these methods will be discussed on the basis of already conducted scientific work. Commonly used methods as well as more complex and even less researched methods will be covered in this work. In the experimental part of this paper, a convolutional neural network will be created that will use the GTSRB dataset to learn image-text classification. Some of the methods discussed in this paper will be applied in order to augment the dataset, allowing us to see their effect on training and the extent to which they help with the task for which they were developed.

Table of contents

Table of contents **i**

List of figures **iii**

List of tables **iv**

1. Introduction **1**

1.1 Problem definition **1**

1.2 Objectives **1**

1.3 Structure of the work **2**

2. Introduction to Machine Learning **3**

2.1 What is Machine Learning? **3**

2.2 Neural Networks **4**

2.2.1 Theoretical basics of a neural networks **4**

2.2.2 General functioning of a an artificial neuron **6**

2.2.3 Learning **6**

2.2.4 Convolutional Neural Network (CNN) **8**

2.2.5 Problematics of Machine Learning **9**

2.2.5.1 Generalizability **10**

2.2.5.2 Underfitting and Overfitting **10**

3. Data Augmentation **12**

3.1 General concept of Data Augmentation **12**

3.2 Image Data Augmentation Techniques **12**

3.2.1 Geometric Transformations **13**

3.2.1.1 Horizontal and vertical Flipping **14**

3.2.1.2 Cropping **14**

3.2.1.3 Rotation **14**

3.2.1.4 Translation (Shifting) **14**

3.2.1.5 Noise Injection **14**

3.2.1.6 Color space 15

3.2.1.7 Advatages and disadvantages of geometrical transformations 15

3.2.2 Colour space augmentations 16

3.2.3 Kernel filters 17

3.2.4 Mixing images 17

3.2.5 Random erasing 18

3.3 Deep Learning based Data Augmentation Methods 19

3.3.1 Adverserial Network 19

3.3.2 GAN-based Data Enhancement 21

3.4 Test-time Data Augmentation 22

4. Practical Experiment 24

4.1 Concept of the practical experiment 24

4.2 Experiment 24

4.2.1 Data 25

4.2.2 Implementation 27

4.2.2.1 The model 27

4.2.2.2 Data preprocessing 27

4.2.2.3 Training 28

4.2.2.3.1 Data augmentation 28

4.2.2.3.2 Training initialization 29

5. Evaluation and results 29

5.1 Experiment conclusion 35

6. Conclusion 36

Bibliography 38

List of figures

1. Figure 1: A taxonomy of feed-forward and recurrentlfeedback network Architectures- Source: [32 5
2. Figure 2: Overfitting (left) and a good fit (right) depicted with error rates on both training (blue) and testing (orange) dataset – Source: [22 10
3. Figure 3: Examples of machine learning algorithm output scatter in comparison to test data in case of underfitting, good fit and overfitting – Source: https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76 - researched on 10.04.2022 11
4. Figure 4: Classification of (some) Data Augmentation Techniques – Source: [45 13
5. Figure 5: Mixing images through random image cropping and patching – Source: [48] 17
6. Figure 6: the concept of Generative Adversarial Network – Source: - https://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html - researched on 09.05.2022 21
7. Figure 7: Snippet of meta data from GTSRB dataset - Source: Own representation 25
8. Figure 8: representation of class distribution in GTSRB dataset - Source: Own representation 26
9. Figure 9: Implementation of class that performs geometric transformation augmentations – Source: Own representation 28
10. Figure 10: Training accuracy, Training loss, Validation accuracy, Validation loss plotted after learning with geometric data augmentations in feature space – author’s work –Source: Own representation 35
11. Figure 11: Snippet of example image data samples in “examples” directory with predicted labels written on respective sample – Source: Own representation 36

List of tables

1. Table 1: Learning results with no data augmentation - Source: Own representation 30
2. Table 2: Learning result with Noise Injection in preprocessing - Source: Own representation 31
3. Table 3: Learning result in geometric transformations in preprocessing phase - Source: Own representation 33
4. Table 4: Learning with geometric augmentations in feature space - Source: Own representation 34

1. Introduction

1.1 Problem definition

Image processing and classification of images and texts are now becoming particularly popular in various computer vision systems. There are now many areas of research in which scientists hope to improve current results by applying deep convolutional networks to computer vision tasks. Excitement and demand have been at very high levels for a long time and will remain so for the foreseeable future. More broadly, almost every other type of classification and training of neural systems for recognition is attracting the attention of scientists and investors. Recognition of text, images, speech, music, computer-generated images, music again, photos, faces, speech-all the different facets of machine learning cover a huge range of tasks and research. But they all require data for training, for validation, for testing. Data that is designed for specific tasks or provides a wide range of information and therefore can be used in a variety of learning domains. Providing data is not an easy process in itself, but that data must also be optimally suited to the task, otherwise the neural network risks not being able to learn optimally from that data to perform its task. Sometimes there just isn't enough data or not enough variety. Often the prerequisites for unsuccessful learning arise already at the data phase alone. Even the most expertly constructed neural network can suffer from a poor data set and show excellent results on training data, but not as strong results on test data.

Data augmentation techniques have been invented to solve these problems. They not only increase the amount of data, but also improve the quality of the data, for example by changing (in our case) the image to a human-unrecognizable state, but keeping the label, thereby improving the learning process of the neural network. There is a wide range of methods of data augmentation and for different tasks. In this paper, we will focus mainly on methods related to image and text classification tasks, as well as how they affect the learning of the neural network and how this effect differs.

1.2 Objectives

In this paper the reader will be given a broader introduction and explanation of the concept known as Data Augmentation, the problems this concept struggles with, and the different variations (techniques) of it. The aim of this paper is not only to discuss the factors mentioned above, but also to demonstrate the work of some of these data augmentation methods (techniques) with a practical example from the field of image-text classification.

1.3 Structure of work

Chapter 2 introduces the reader to the theoretical background of machine learning and image-text classification, such as the learning process, neural network structure, and so on.

Chapter 3 introduces the concept of data augmentation, which is central to this paper, as well as the neural network training issues that the concept was invented to deal with. The reader will be given a wide range of different methods (techniques) of data augmentation, more or less relevant in the context of image-text classification. Some methods will be covered in more detail then others, especially those that will be later implemented and tested on a dataset. The other techniques will be partially mentioned as well. Chapters 4 and 5 will deal with the practical part:

- Chapter 4 will introduce the reader to the practical part of this paper – the experiment. It will introduce the reader to the the dataset used for the experiment. It will go into detail about the implementation of the experiment and the model used for it and about how the experiment is conducted.
- Chapter 5 covers in detail the results of the experiment and provides a summary. It will compare different iterations and different versions ran as well as the results of those iterations.

Chapter 6 will serve as the conclusion to this paper and will discuss potential ramifications of the main topic. It will talk about what the results mean for general understanding as well as which topics discussed in the paper have potential for further research and in what way.

2. Introduction to Machine Learning

2.1 What is Machine Learning?

Machine learning is a branch of computational algorithms that are designed to mimic human intelligence by learning from the environment. It is a component of artificial intelligence, although it seeks to solve problems based on historical examples [1]. Its methods and algorithms are being widely used in the fields of Data Science and Software Developement in order to create systems or programs that can learn and improve it‘s performance based on the given data and it‘s past experiences without the need to be explicitly programmed to do so. However, in contrast to artificial intelligence applications, machine learning involves learning hidden patterns in data (data mining) and then using these patterns to classify or predict events related to a problem [2]. Basically, processes and machines deemed „intelligent“ require knowledge to maintain their performance and functionality. In essence, machine learning algorithms are embedded in machines, and data streams are provided so that knowledge and information are extracted and fed into the system to manage processes more quickly and efficiently [3].

Also commonly referred to as driving force during the current era of Big Data, as methods based on machine learning have been successfully applied in fields ranging from pattern recognition, computer vision, spacecraft design, finance, entertainment and computational biology anc chemistry to medical applications. The ability of machine learning algorithms to learn from current context and generalize to unseen tasks proves its immense value in any field it is deployed to as it reaches the levels of accurate performance humans cannot. Using mathematical analysis methods, ML algorithms look for specific patterns in given data and use them in order to imrpove their performance and decision making. ML is often mixed with Deep Learning (DL) when it comes to a discussion about artificial intelligence and its algorithms. An introduction to the field of machine learning requires clarification of the essential differences between these concepts. In fact, these terms are very codependent of one another as deep learning is a sub-field of machine learning. A significant difference persists when comparing the ways ML differentiates itself from DL in the ways its algorithms learn:

DL relieves a substantial amount of human intervention into its learning processes by automating its data extraction processes, making it scalable when compared to classical ML [4]. It requires less manual activity, but also enables us to work on larger data sets, as classical ML algorithms require more structure in its data sets, because it is necessary to determine the set of features to understand the differences between data inputs, which can usually be done manually. This leads to smaller data sets used, as the process of labeling data is not suitable for larger data sets. DL does not require labeled datasets in order to extract information, as it can work with raw data form and identify patterns, that help it destinguish different categories of data from one another [5]. DL is widely used for solving problems in areas like speech recognition, natural language processing and computer vision. The learning of both ML and DL algorithms is made possible by the use of neural networks.

2.2 Neural networks

2.2.1 Theoretical basics of a neural network

Nerve cells of the human neural system basically allow our brain to react and make decision based on the input of information they receive. From a purely technical point of view, the underlying principles of the human neural network can basically be imitated in an artificial neural network. Decisive for the enormous performance of the brain are on the one hand the large number of nerve cells (estimated 10^{14}) and on the other hand, the extremely high number of connections. Far fewer connections are possbile in an artificial neural network [6].

A neural network, or in our case, an artificial neural network (ANN), consists of multiple (in this case, artificial) interconnected neurons (also called nodes). An average ANN can be broken down to 3 entities that build it: the individual neurons, connections between them (network topology) and the learning rules defined in the network [7].

The neurons of an ANN are grouped in and follow each other in layers, forwarding information from the neurons of one layer to the neurons of the next. There are 3 general layer-types to be seen in an ANN:

- Input layer – consisting of input nodes, this is the layer that get the information and prepares it for the transfer to the next layer. Similar to the human nervous system, input neurons receive the external stimuli or variables that are to be processed and transferred to the next set of inner (hidden) layers.
- Hidden layer – one to (usually) multiple layers of neurons that interract differently with each other based on one’s architecture type. Based on the signal received, each inner node carries out calculations and sends the result to the next node in the next layer (next hidden layer or the output layer).

- Output layer – layer of output nodes work as a network response (output) as well as evaluation of this response. [6]

When talking about the architecture of an ANN, the reader may picture it as a connected directed graph, as the theory behind the structure of both is very similar. The neurons would be the nodes, and the connections between them would be directed (weighted) edges [8]. It is important to identify the two most common structures of an ANN:

- feed-forward networks, in which graphs have no loops. These networks are classified as static and produce one set of output values rather than a sequence of values from a given input. This type of network is also memory-less, as it's response to an input does not differ from the previous network state. Among many other learning algorithms, the "back propagation algorithm" is the most popular and most commonly used for training of feed-forward neural networks, as it is essentially a means of updating the synaptic weights of the network by the back propagation of a gradient vector, in which each element is defined as the derivative of an error measure with respect to a parameter [47]. Error signals are usually defined as the difference between the actual outputs of the network and the desired outputs. Therefore, it is necessary to have a set of desired outputs for learning. For this reason, back propagation is a supervised learning rule [9].
- recurrent (or feedback) networks, in which loops are present and occur because of feedback connections. These networks are dynamic. When a new input pattern is presented, the neuron outputs are computed. Because of the feedback paths, the inputs to each neuron are then modified, which leads the network to enter a new state [10].

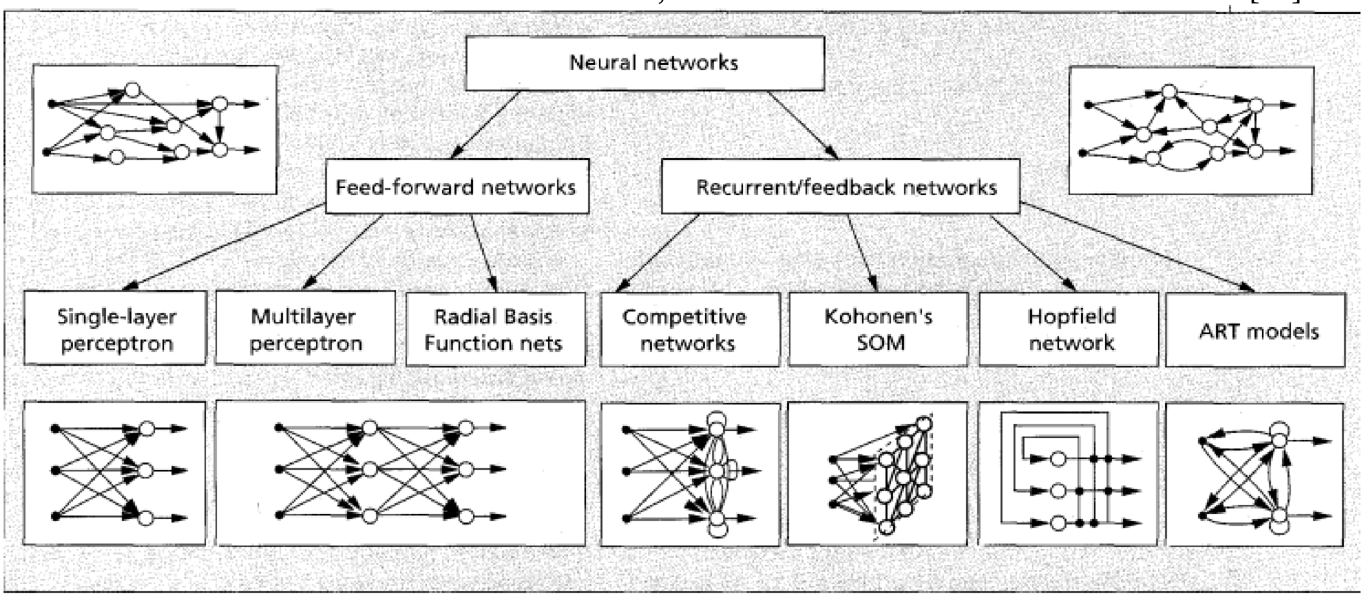


Figure 1: A taxonomy of feed-forward and recurrentlfeedback network architectures

Neural networks come in many different forms, each of which is better fit for a specific type of task. Figure 1 will give the reader a brief overview of what different types of neural networks exist (certainly not all), what their architecture looks like and what type of network structure (discussed in the paragraph above) they belong to.

2.2.2 General functionality of a an artificial neuron

An ANN and its artificial neurons are structurally and operationally very similar to the human biological neural network and its biological neurons. Biological neurons are interconnected with and transfer information via a synopsis. On the other hand, when it comes to artificial neurons, this connection is replaced by a weight designation. A positive weight means that the connection between neurons is excitatory (enambling an action) and a negative weight represents an inhibitory connection (preventing an action). [11]

The structure of an artificial neuron is illustrated in Figure 2 and might serve the reader as a good visual representation of a step-by-step description of a neuron‘s function.

A neuron receives information from another neurons of the previous layer, calculates its input and gives an output to the neuron of the following layer. The calculations that take place between the neuron receiving the input and transmitting the output to the next neurons can be reduced to 3 consecutive steps:

1. the first is to multiply the inputs with the synaptic weights to calculate the products;
2. the second is to add the products to calculate the sum of the weights;
3. the third and last step is to apply the chosen activation function to the summed weights, which in its place generates the neuron‘s output.

   It is very important to take a closer look at the very tool that produces the output of a neuron: the activation function. It is a critical part of the design of a neural network. As such, the choice of an activation function is extremely important for the performance of a neural network. Different types of layers of the ANN require different activation functions, as they all have different purposes. For example, the hidden layer and the sublayers within it require different activation functions not only from the output layer, but sometimes from each other as well. [11]

2.2.3 Learning

In the context of ANNs, learning may be regarded as the problem of updating the network architecture and connection weights so that the network can effectively perform a particular task.

Typically, the network must learn connection weights from existing training models. Performance improves over time by iteratively updating the weights in the network. Instead of following a set of rules given by a human, ANNs seem to learn basic rules (such as input-output relationships) from a given set of representative examples. To understand or design the learning process, we first need to have a model of the environment in which the neural network operates, that is, to know what information is available to the network. We call this model the learning paradigm. Second, you must understand how the weights of the network are updated, that is, what learning rules govern the updating process. A learning algorithm refers to a procedure in which learning rules are used to adjust the weights. The main learning paradigms are supervised, unsupervised, hybdrig, and reinforced.

- In supervised learning (or learning with a 'teacher'), the network obtains the correct response (output) for each input pattern. It is a sequence of desired outputs given to the model. The weights are determined to allow the network to produce answers that are as close as possible to the known correct answers. The goal of the machine is to learn to produce the correct output given a new input. This output could be a class label (in classification) or a real number (in regression) [12]. Figure 3 shows supervised learning as a distinction along a defined criteria in data, that being predefined classification and the results falling into exact categories.
- Reinforcement learning is in itself a variant of supervised learning in which the network is only given a critique of the correctness of the network's output, in the form of scalar feedback (positive or negative, also called “reward and punishment”), not the correct answers themselves. The goal is to learn to act in a way that maximizes future positive feedback (or minimizes negative feedback) over the learning period. There is also an advanced form or reinforcement learning, which revolves around “game theory” and generalizes the concept. It works with the same principles of learning as before, but with one major difference – the environment is no longer static, but dynamic, with other learning algorithms taking part in it as well, constantly picking up on information and “spitting out” the results. The goal of the machine is to act in a way that maximizes positive feedback in relation to the current and future actions of other machines. Thus, the learning process of all models in this paradigm is more flexible and generalizable [12]. Figure 3 shows this type of learning as a more complex set of agents that interchange feedback among each other regarding the produced output.
- Unsupervised learning (teacherless learning) does not require a correct response associated with each input pattern in the training dataset. In this case, the machine receives input, but

not a label, classification, or even potential feedback (reward or punishment) from the environment. It examines the underlying structure of the data, or correlations between patterns in the data, and combines patterns into categories based on these correlations. In a sense, unsupervised learning can be thought of as looking for patterns in the data, beyond what can be thought of as pure unstructured noise [12]. Figure 3 shows unsupervised learning on an example of a clustering algorithm, where data is arranged into non predefined groups, but groups are arranged during learning by data patterns found by the algorithm.

- Hybrid learning (as the name suggests) combines supervised and unsupervised learning. Some of the weights are usually determined by supervised learning and the rest by unsupervised learning.

Learning theory must address three fundamental and practical issues related to learning from samples: capacity, sample complexity and computational complexity.

- **Capacity** refers to how many samples can be stored, as well as the features and decision bounds that the network can form. It can be measured by the number of training examples that the learning body can always fit, no matter how the values are changed [13].
- The number of training patterns required to train the network to ensure reliable generalization is determined by **sampling complexity**. Overfitting can occur when there are insufficient samples (where the network performs well on the training dataset, but poorly on independent test samples, that are part of the same dataset as the training samples).
- **Computational complexity** of a learning algorithm is the time it takes to evaluate a solution based on the training patterns. The computational complexity of many existing learning algorithms is high [14]. The development of efficient algorithms for training neural networks is a topic of intense scientific interest.

2.2.4 Convolutional neural network (CNN)

Convolutional neural networks (later referred to as “CNN”) are used broadly in the fields of image processing, computer vision, speech recognition, machine translation and so on. In the case of this work, image recognition and classification bein the task fields, CNNs are dominant in solving the problems in that field. A convolutional neural network has had major success in studies over the last years and has achieved revolutionary results. The most useful aspect of CNN is the reduction of number the of parameters in the network. This option allowed researchers and developers to turn to larger models to solve complex problems, for which ordinary ANNs lacked the functionality to solve. The most important assumption is that problems solved with CNNs should not have features that are spatially dependent. For example, in a face detection application, there is no need to pay attention to where faces are in images, and the only task remaining is to detect them regardless of their position in the image data.

A CNN usually consists of 3 different layer types:

- Convolutional layer - most of the computational activity of a CNN takes place in the convolutional layers. “A convolution is an integral that expresses the amount of overlap of one function g as it is shifted over another function f. It therefore "blends" one function with another“ [15]. The layer is called convolutional since it performs a dot product (or it convolves) between two matrices. The first matrix is a set of parameters able to be taught and changed (also known as a kernel), and the second one is a portion of the receptive field. The kernel moves (or strides) along the height and width of the given image, generating the image data in the receptive field. This results in a two-dimensional image representation known as an activation map, which returns data for every spatial position of the image [15].
- Pooling layer – the pooling layer analyzes the output data of the convolutional layer and replaces the output at certain locations with calculated statistics of its neighboring output. A big advantage of this layer is that it reduces the number of weights and calculations drastically. By stacking several convolutional layers and a pooling layer, we can extract the high-level characteristics of inputs [16].
- Fully connected layer – this layer serves as an assistance in mapping of the representation between input and output. Neurons of this layer are fully connected with all the neurons in the previous and following layer. A classifier of a convolutional neural network consists of one or more fully connected layers. These layers do not preserve spatial data [16].

2.2.5 Problematics of Machine Learning

With the theoretical basis of machine learning and CNNs left behind, it is time to look at the main problems mainly confronted when training and testing CNNs. To conclude this chapter, it is important to talk about the problems and challenges that stand in the way of successful machine learning, as a kind of preface, but also as an important thematic transition to the topic of data augmentation.

Convolutional neural networks are very effective and efficient at performing computer vision tasks. However, the main and one of the most difficult challenges is increasing the generalizability of a model and preventing overfitting during training. For better understanding, the important definitions will be elaborated upon separately.

2.2.5.1 Generalizability

When training CNNs, one very important attribute has to be optimally increased, and that attribute is generalizability. It is the ability of the model to adapt correctly to different data sets rather than the training data set. It indicates the difference between the output of the model with training data (i.e., data that is already "known" to the model) and testing data (data that is unknown to the model). **[17]**

It is not always possible to provide a data set of the right size and content to train a neural network successfully and achieve a high degree of generalization. The data may not be sufficient, have class imbalances, or not be homogeneous. In the last case, the amount of data may not matter or may even be detrimental, because, if not addressed properly, the algorithm will be trained to recognize similar patterns in the dataset and, if they are homogeneous, will be trained to identify or work only with certain data presented in the dataset. The potential problems that may arise are called underfitting and overfitting.

2.2.5.2 Underfitting and Overfitting

- The term **underfitting** refers to a model that cannot model the data used for training. It can also not generalise itself when given new data (testing data). Underfitting will become noticeable early on, because at the training stage its representation will not be satisfactory [18].

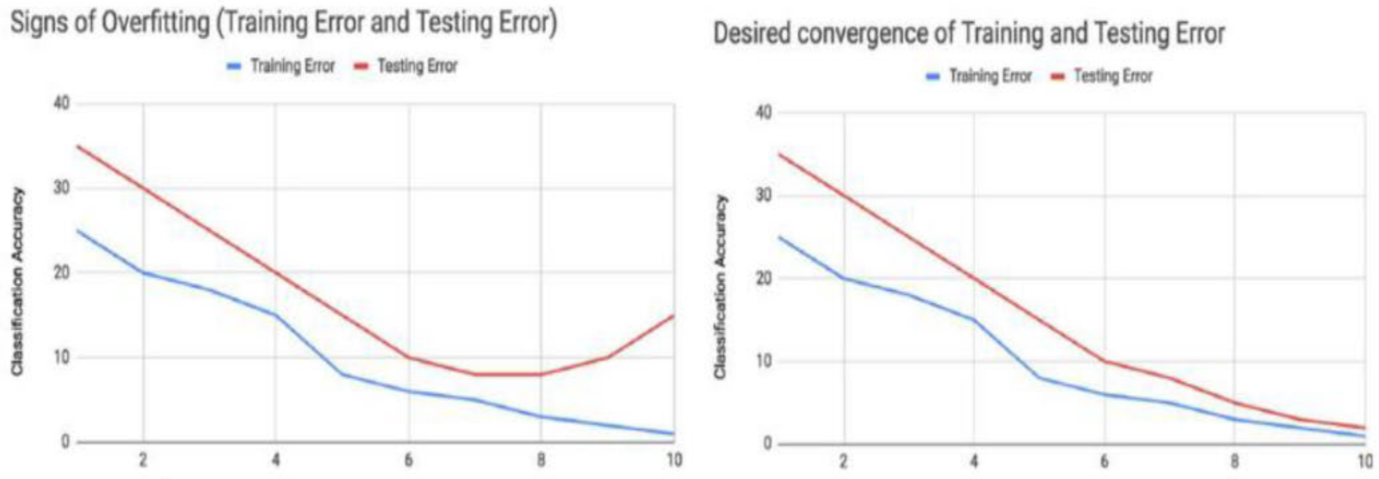


Figure 2: Overfitting (left) and a good fit (right) depicted with error rates on both training (blue) and testing (orange) dataset.

- The term „**overfitting**“ is more complex. It is used to refer to a state, that occurs when a network models the set of training data perfectly, but fails to generalize its learning to predict the class of unseen data correctly [17]. The results when the network has trained too well on the data presented for training that the slightest fluctuation (noise) or an unknown class in the testing or real time data and this is picked up by the model and learned to be considered as a general pattern. This state occurs when a classification algorithm learns to classify the training data better than the population of cases at hand, meaning the algorithm does not generalize well to the population of cases from which the training data was acquired [19].

  The pattern of overfitting can be seen in Figure 2 (above). The left depicts an inflection point at which the validation error begins to increase as the training rate decreases. Because of the increased training, the model has overfit to the training data and performs poorly on the testing set in comparison to the training set. On the right, on the other hand, a model is producing the desired relationship between training and testing error [22].

  Overfitting is a more complex problem than underfitting. Its higher complexity lies in the fact that overfitting related inoperability of the model is not noticed during the training period. This becomes noticeable during the testing period and marks the fact that the training dataset is not suitable for successful learning as well as optimal model generalization. CNNs are very prone to overfitting as a result of the high number of parameters involved in their training.

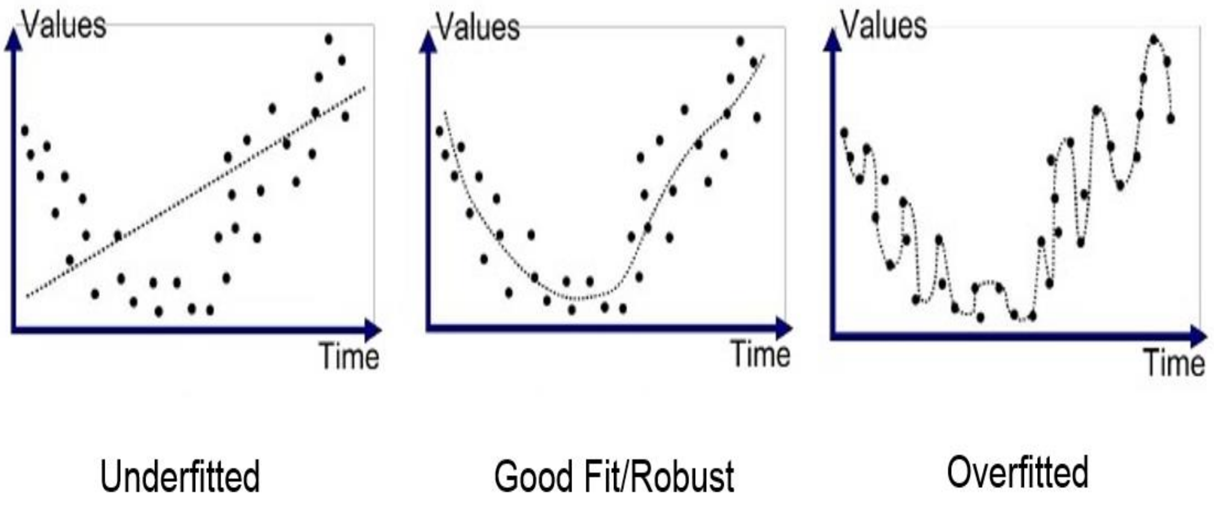


Figure 3: Examples of machine learning algorithm output scatter in comparison to test data in case of underfitting, good fit and overfitting.

3. Data Augmentation

3.1 General concept of Data Augmentation

L. Taylor and G. Nitschke gave a plausible definition of the concept of DA in their paper on „Improving Deep Learning using Generic Data Augmentation“, which will explore and ellaborate the general concept of data augmentation (later referred to as DA) and „label preservation“ :

The core of data augmentation methods is an artificial inflation of the original training set. An important feature of those methods is that the transformations they enable are label preserving. The core transformation can be represented as this mapping: $\phi : \mathcal{S} \mapsto \mathcal{T}$

where S reprsents the original training data set and T represents the augmented data set of S. The augmented and artificall inflated set can be represented with a following relation:

$\mathcal{S}' = \mathcal{S} \cup \mathcal{T}$ where S' contains the original training set S and the augmented transformation set T. The transformations taking place in T are defined by φ. These transformations are label preserving, as the following rule must persist with a good data augmentation method: "if image x is an element of class y then φ(x) is also an element of class y".[20]

It is important to note that although DA methods are widely used to inflate training datasets, it is also a common practice to augment a specific percentage of the dataset without increasing the number of samples. There is a wide range of data augmentation methods for different kinds of machine learning tasks. Since the focus of this paper is on image processing and image-text classification, only the DA techniques that are suitable for this task will be discussed.

3.2 Image Data Augmentation Techniques

DA (as can be derived from a name) combats overfitting by trying to fix the problem in the phase before initial training, mainly in the dataset. To avoid overfitting and train the model more effectively a broad line of DA techniques has been developed. The techniques vastly differ from each other in the sole structure of the algorithm and especially in its complexity. Research papers on DA often divide its methods into two broad categories: data warping and synthetic oversampling.

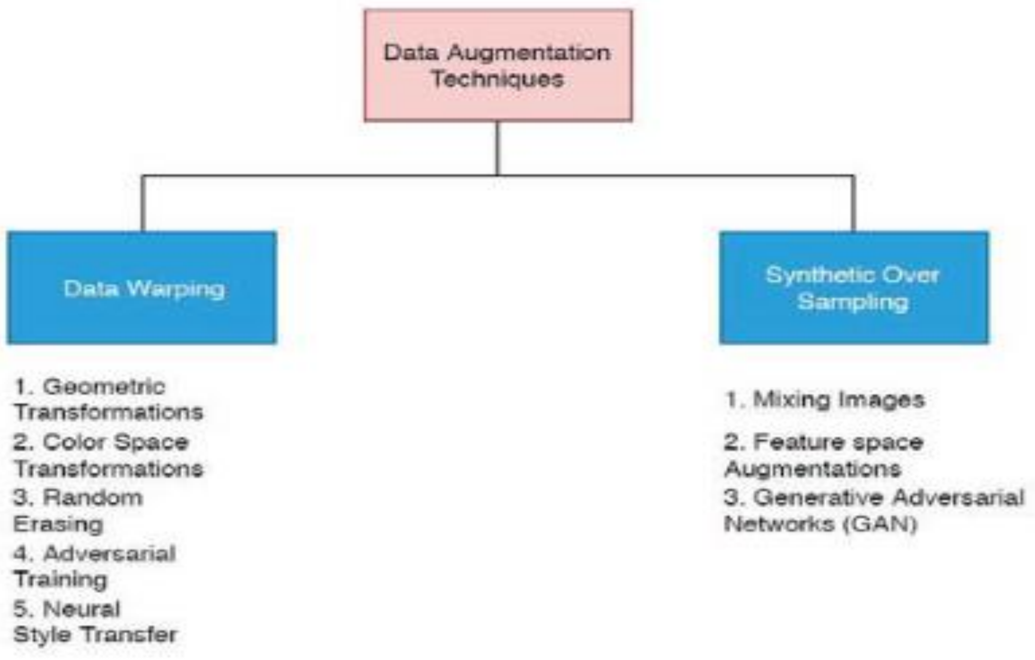

Data Augmentation Techniques
Data Warping
Synthetic Over Sampling
1. Geometric Transformations
2. Color Space Transformations
3. Random Erasing
4. Adversarial Training
5. Neural Style Transfer
1. Mixing Images
2. Feature space Augmentations
3. Generative Adversarial Networks (GAN)

Figure 4: Classification of (some) Data Augmentation Techniques

The basic idea of data warping revolves around altering already given data (without introducing new samples) and thus accelerating the learning process of the model. It does not mean it cannot be used to create new data and add new samples alongside existing ones. Nevertheless, the idea is that these methods influence the image on an already persisted level. The concept of Synthetic Oversampling revolves around the idea, that its methods will result in creating new images (for example, a mixed image of 2 existing images) and adding them to the dataset, thus inflating the dataset and creating new samples [21].

3.2.1 Geometric Transformations

First on the list are the DA methods that rely on basic image manipulations. This class of augmentations is in implementation and in concept, but nevertheless poses challenges when it comes to safety. Safery relates to the ability to preserve data labels after the transformation of samples. A basic example would be applying horizontal flipping to a dataset of images of cats and dogs, which is label preserving and does not pose a conflict or an error and applying the same method to datasets with digits, particularly with 6 and 9 [22].

Non label-preserving DA methods might strengthen the ability of a model to hesitate on certain predictions and show a lack of confidence in some predictions. However such a result would require to commit label refinement [22] after the augmentation took place and that is an extremely challenging computing task.

3.2.1.1 Horizontal and vertical Flipping

One of the least complex techniques to implement is being widely used as an addition to other techniques. A straight forward technique, horizontal or vertical flip, reverses the pixels (in the context of images) rows or columns. Flipping the horizontal axis is more common than vertical axis flipping. This technique has proven very helpful in cases of classifying images to text. However, in the context of text recognition, numerous experiences have shown this transformation to fail to preserve labels. The most obvious example is performing horizontal flipping in the already mentioned above case of digit recognition, specifically with digits 6 and 9. [22]

3.2.1.2 Cropping

This method revolves around cropping a part of the image sample and reducing the size of the input data. Image cropping can be used as a practical step in processing image data with mixed dimensions in height and width by cropping the central portion of each image. Depending on the reduction threshold chosen for pruning, this may not be a label preserving transformation [22].

3.2.1.3 Rotation

Rotation is performed on the image sample by rotating the image along the axis ranging from 1° to 359°. The safety of this augmentation method hangs from the rotation degree.

3.2.1.4 Translation (Shifting)

Offsetting images left, right, up, or down can be a very useful transformation to avoid positional bias in the data. For example, if all images in a dataset are centered, as is often the case in face recognition datasets, this will require testing the model on perfectly centered images as well. When the original image is translated in a particular direction, the remaining space can either be filled with a constant value, such as 0 s or 255 s, or with random or Gaussian noise. Such filling preserves the spatial dimensions of the image after augmentation [22].

3.2.1.5 Noise Injection

This technique refers to adding „noise“ artifically to the data that is then served as an input to a CNN during the training process. What is referred to as „noise“ is a matrix of random values injected into the dataset with the purpose of slightly altering it in a specific area. There are different methods of implementing noise injection. One of them is called „jitter“ – adding a noise vector to each training sample in between training iterations. This causes the training data to “jitter” (tremble, quiver, shake) in the feature space during training, making it difficult for the model to find a solution that fits precisely to the original training dataset, and thereby reducing overfitting. Due to those constant injection, the ANN works with slightly different data every training iteration [19].

An important parameter is the variance of the noise kernel, which controls the effect of noise injection on model training. When variance is too small, it fails to make the training cases‘ „jitter“-effect significant enough when confronted with them in the feature space, which overall results in a minimal result on the training of an ANN. A variance too large, however, can cause the „jitter“-effect to be so significant, that the two classes of the training cases will become too hard for the model to distinguish from one another, therefore making the training process ineffective. The perfect match is when the value of variance of the noise kernel is selected so that the ANN training case feature vectors „jitter“ in the featre space to an extent that an overall „jitter“ of all given training cases imitates the underlying distribution of each class of the training cases in the feature space [19].

3.2.1.6 Color space

Digital images are usually encoded as 3 dimensional matrices, encapsulating a height vector, a width vector, and an RGB channel vector, basically encoding which pixel at which position has what kind of RGB value or color. A practical strategy is to perform augmentations in the RGB channels space. It is a simple case, isolating a single color channel. RGB values can be tweaked to set brightness higher or lower, or, by isolating its matrix and adding two zero matrices from other color channel, the image will have change its pixels RGB values to value of one specific channel [22].

3.2.1.7 Advantages and disadvantages of geometric transofrmations

Geometric transformations provide the case with a very strong solution to positional biases appearing in the data used for training. For instance, face recognition datasets use images that are faces perfectly centered in the picture, which spawns a bias that hinders the model from effectively learning to recognise different placement of a face on the photo. In this case, geometrical transoformations serve as a perfect solution. Geometric transformations require little implementation due to their easy concepts and a large number of libraries that provide these methods for instant use.

However, the downside of these transformations includes the resource cost of calculating the transformation, very high additional learning time and additional memory usage. Further disadvantage of these methods is a limited application scope, since in some areas of image recognition (for example medical image analysis), geometrical transformations that differentiate the training set images from testing set images on a structural level are actually hurting the learning process and therefore cannot be used as an optimal solution. [22]

Also, it is a challenge to give a simple evaluation on the efficiency of one method over the other. There are a lot of geometrical data augmentation techniques (not all techniques or variations have been discussed prior) and some of them are pretty simple in their implementation.

3.2.2 Color space augmentations

As mentioned previosly, image data is encoded as three stacked matrices of height x width size. These matrices represent individual RGB color values as pixel values. Lighting biases are among the most common challenges to image recognition problems. As a result, the effectiveness of color space transformations, also known as photometric transformations, is fairly easy to grasp. A quick fix for overly bright or dark images is to loop through them and change the pixel values by a constant amount. Another transformation is to limit pixel values to a specific minimum or maximum value. A variety of augmentation strategies can be derived from working with pixel color manipulations. Changing the color distribution of images can be an excellent solution to the lighting issues encountered by testing data. Image datasets can be represented more simply by converting the RGB matrices into a single grayscale image. This results in smaller images (height x width x 1) and faster computation. Color space transformations, like geometric transformations, also pose a disadvantage of increased memory requirements, transformation costs, and training time [22]. Color transformations may also discard important color information, making them not always a label-preserving transformation. For example, when the pixel values of an image are reduced to simulate a darker environment, the objects in the image may become impossible to see. Image Sentiment Analysis [23] provides an indirect example of color transformations having a non-label preserving effect. CNNs attempt to predict the sentiment score of an image in this application, such as highly negative, negative, neutral, positive, or highly positive. And the presence of blood is an important indicator of a negative/extremely negative sentiment image. The dark red color of blood destinguishes it from water, paint, or any other liquid. If color space transformation is used multiple times, the model will perform poorly in image sentiment analysis because it will be unable to distinguish red blood from green paint. Color space transformations, in effect, eliminate color biases in the dataset in favor of spatial characteristics [22].

3.2.3 Kernel filters

Kernel filters are a very popular and often used method in image processing. It helps sharpen blurry images in the dataset. The filters work by moving an n x n matrix over the image with one of the following filters: Gaussian blur filter (which makes the image blurrier) or a high contrast vertical or horizontal filter, which will make the image sharper on the edges [22]. In theory, blurring images to augment data can result in increased resistance to motion blur during testing. Furthermore, sharpening images for data augmentation can result in the encapsulation of more information about the objects of interest.

Kernel filters are commonly used to enhance and blur images. The concept has been experimented with to create a novel filter operation that swaps pixel values in a sliding x-matrix. This image data augmentation technique got a name Patchshuffle Regularization. It has shown an improved error rate of 5.66% on CIFAR-10 compared to the previous result of 6.33% [24].

3.2.4 Mixing images

Mixing images is not a straightforward and obvious data augmentation method, which revolves around combining images by averaging their pixel values, essentially blending images together. At first glance, it makes little sense for the human eye to see a mixture between an image of a dog and a cat. However, it has been proven to be a well working data augmentation technique. By precropping two random images from 256 x 256 to 224 x 224 and flipping them horizontally, after which the mixing of images takes place via averaging the pixel values for each of the RGB channels. The mixed image is generated that way and it is given a label of the first of the two randomly chosen images. By testing this method on the CIFAR-10 dataset, it managed to improve the error rate of a learning model to 6.93% from the previous 8.22% without mixing images. Also, this technique was tested out to see how it would work on a very small dataset. For that purpose, CIFAR-10 has been reduced to

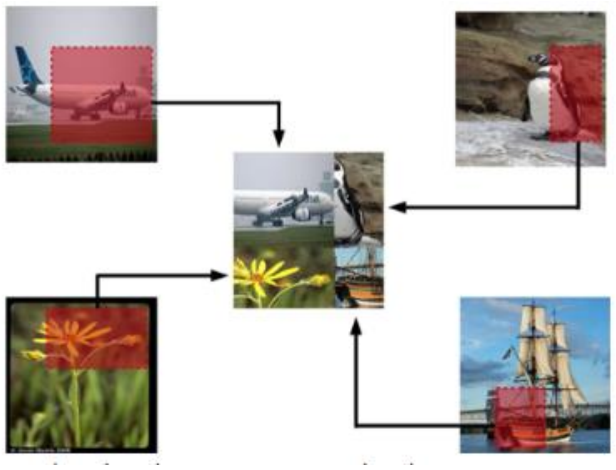


Figure 5: Mixing images through random image cropping and patching

just 1000 samples (100 per class) and image mixing has been performed on the dataset. Of course, the error rate due to insufficient datasets remains high, but the use of image mixing has had a significant decrease in error rate from 43.1% to 31% [29]. This showed the usefulness of image mixing in the case of insufficient data. Another important note from the study showed that combining images from the entire training set has higher results, as opposed to combining only images of the same class. But this method of image blending is a linear one. A non-linear approach to image mixing has been explored out in multiple scientific works and has shown effective results when used on both CIFAR-10 (error rate reduced from 5.4% to 3.8%) and CIFAR-100 (error rate reduced from 23.6 % to 19.7 %) [25]. Another effective variation of image mixing is randomly cropping parts of random images and blending them together, generating an entirely new sample.

The curious case of image mixing is that it is highly effective in training models. It is highly increasing its performance across a variety of tasks, but it makes no sense for human observation. Furthermore, no one has yet proven the reason for its high effectiveness, and there have only been theories and hypotheses, like the ones from Zhang [26] and Tokozume [27].

3.2.5 Random erasing

Zhong et al. developed another non-trivial data augmentation technique called random erasing [28]. When used on CIFAR-10, it managed to achieve the highest accuracy. The technique is inspired by Dropout regularization and has a lot of similarities to it. The primary distinction between the two is that one is used during the data input phase and the other is not directly embedded in the network architecture.

This method was created solely to address issues in the image recognition process caused by a phenomenon known as “occlusion”. Occlusion refers to an image state in which some parts are less clear than others, which can lead to a model learning some specific features better than others and thus overfitting. Random erasing forces the model to learn more descriptive features about an image, avoiding overfitting to a specific visual feature. As a result, the model is trained to study and process the entire image equally, rather than using more learning resources to process the parts of an image that do not have occlusion. Random erasing works by randomly selecting a nx m patch of an image and masking it with 0s, 255s, mean pixel values, or random values. This reduced the error rate on the CIFAR-10 dataset from 5.17 to 4.31 percent. Random erasing is a data augmentation technique that modifies the input space to prevent overfitting directly [22].

This augmentation technique has also shown benefits when combined with other augmentation techniques like random zoom, rotation, and shifting.

The disadvantage of random erasing is that it does not always result in a label-preserving transformation. In handwritten digit recognition, if the top part of a '8' is randomly cropped out, it is no different from a '6' [22]. As a result, some manual intervention may be required depending on the dataset and task.

3.3 Deep Learning based Data Augmentation Methods

Methods, that have been described prior are powerful and useful when it comes to assisting machine learning, but they are, nevertheless, very limited and constrained by the fact that they have to be explicitly selected and used by a human expert in the pre-training phase, more specifically in the input space of the network. These methods help the network to generalize but are themselves not very generalizable in that they are not fixed as a "silver bullet" for any particular problem. More often than not, researchers have to combine different techniques with different parameters to achieve a result, let alone show significant progress. It is left to the researcher to decide, which method should be used and if it is the optimal combination. And these standard methods produce only "limited plausible alternative data" [31]. But what if there was a way to automate the choice and application of data augmentation methods?

Some researchers went as far as to apply deep learning algorithms to enhance and optimize data augmentation. And when it comes to mapping high-dimensional inputs into lower-dimensional representations, neural networks are unrivaled. In flattened layers, these networks can map images to binary classes or nx 1 vectors. The sequential processing of neural networks can be manipulated in order to separate the intermediate representations from the network as a whole. Lower-dimensional image data representations in fully-connected layers can be extracted and isolated. For example, it has been found that manipulating the modularity of neural networks to isolate and refine individual layers after training improves performance on CIFAR-100 from 66 to 73 percent accuracy [30].

Some of the known techniques will be covered in the following chapters since they pose a very high research interest. These methods will not be a part of practical implementation since it requires higher computational power to run such methods and implementations of them are much more complex than the standard data augmentation methods.

3.3.1 Adversarial Network

A relevant and exciting field of study is expanding the space of possible augmentations and classifying the situations, to which an individual or combination of methods is plausible. And the concept of adversarial training provides the necessary ground for that. It is a framework for the application of two or more networks, which all have competing objectives encoded into their loss function. One of the sufficient parts of this concept is called “adversarial attack” [22]. Two or more networks a performing task, that are rival to one another. One network learns image data augmentations (with different standard methods) and the other one learns to classify visual input data. The goal of the first network is to perform an augmentation in such a way to make its rival misclassify. Many interesting results have been derived from experiments using this concept. For example, Su et. al [35] revealed that 70,97% of images can lead to a misclassification by changing just one pixel in them. Zajac et al. [36] showed that an adversarial network can cause misclassifications with adversarial attacks limited to the border of images and that these attacks have higher success rates with higher image resolution. Such attacks may be targeted or untargeted, depending on the deliberation they are trying to cause in the classification network [22]. This concept actually helps identify and analyze weaknesses in network classification better than standard metrics used for classification.

Adversarial training provides evaluation metrics for classification algorithms and defense against adversarial attacks, but it can also be an effective way of searching for potential augmentations. By learning to apply diverse data augmentation methods, it eventually produces an augmentation that will cause misclassification. Such augmentations will make the classification model more robust since it will target and identify weak spots in it. The major difference to the standard methods is that the augmentations are not taking place in the training set, but “but they can improve weak spots in the learned decision boundary” [22].

The efficiency of adversarial training in the form of noise or augmentation search is still a relatively novel concept that has not been well examined. It has been demonstrated that adding noise to adversarial cases improves performance, but it is uncertain whether this is effective for the goal of reducing overfitting.

3.3.2 GAN-based Data Enhancement

GAN-based Data Enhancement utilizes the concept of generative modeling – essentially generating artificial instances in the original training set out of samples of a dataset with similar properties. The concept of adversarial training led Goodfellow et al. [33] to come up with the Generative Adversarial Network (GAN). Providing the highest computationally speed and highest output quality, this technique is the go-to in the fields of image generation, deepfakes, and so on. Its solid performance in these fields led to researchers consider its use for solving data

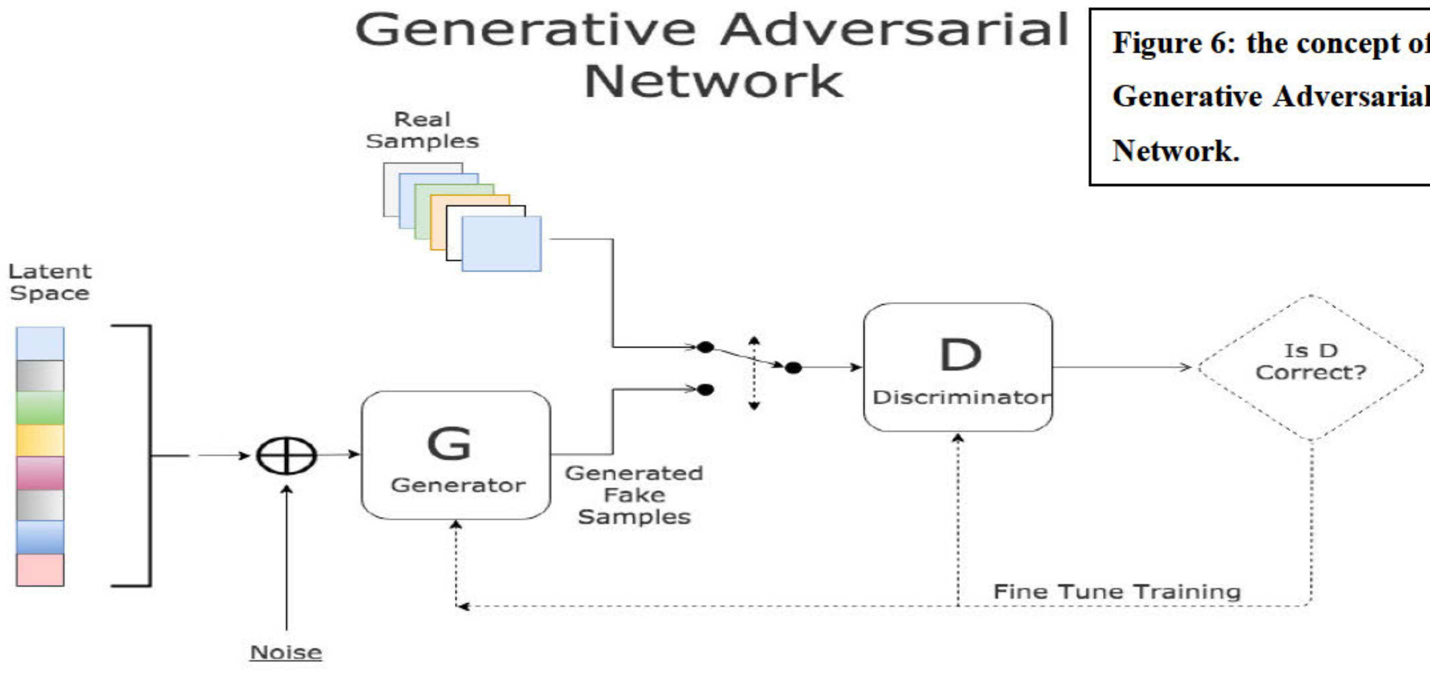


Figure 6: the concept of Generative Adversarial Network.

augmentation problems. GANs are very efficient at generating new training data, resulting in higher classification accuracy during training. Ian Goodfellow is considered to be the "godfather" behind the concept of GANs. The concept is very intriguing. In essence, two networks are working with (or against) each other. One of the networks is called the "generator network" (further referred to as "G"), the other is called the "discriminator network" (further referred to as "D") [33]. Figure 6 will give a good representation of the concept's basic idea and structure.

As illustrated in the figure above, G is a generator network, and it accepts input (random vector, text, picture, and so on.). G learns to create an augmented version of this input and passes it on to D, a discriminator network. D becomes data from two potential sources training set or from G but does not know which one is it. D is to supposed to take the input, process it and decide, if the input it got is a real data sample from the training set or is it generated or augmented in any way. The goal of G is to perform high quality augmentations and make D misclassify. G becomes feedback about D's decisions after every iteration and thus can learn and adjust its operations. D also becomes feedback if its prediction was correct or not. This means G and D both work against each other and learn accordingly in the case of one's output success or failure [33]. Deepfakes are being created with this remarkable technique. The generator network's ability to circumvent the discriminator makes it a potent tool for generative modelling.

In the years following the publication of GAN, many papers proposed improvements to the design, architecture, loss functions, and so on. For example, the initial GAN uses multilayer perceptrons in both generative and discriminative networks [33], which poses some limitations when it comes to generating images with higher resolution and quality. Alternatively, a concept of Deep Convolutional GAN (DCGAN) has been introduced, which implements deep convolutional neural networks in both the generative and discriminative components of GAN [38] and is supposed to increase the internal complexity of the generator and discriminator networks [22] [38].

3.4 Test-time augmentation

Data augmentation on training data is the most common method, but surprisingly, a non-trivial approach has also shown effectiveness. This approach is data augmentation on test-time data. Augmentation of a test image in the same way as augmenting a training image can lead to a more accurate prediction. This does, however, spike up the computational cost and slow down the model learning process and may cause major bottlenecks in models that require real-time prediction [22].

Nevertheless, test-time augmentation has shown great promise for applications such as medical image diagnosis [41]. Multiple studies have been conducted in order to see the effectiveness of different standard data augmentation methods in the context of test-time data augmentation.

Wang et al. [42] have researched a mathematical framework for expressing test-time augmentation. The results have shown that the test-time augmentation scheme outperformed the single-prediction baseline and dropout-based multiple predictions on medical image segmentation. Also, it has been shown that data augmentations of test-time data improve uncertainty estimation, reducing predictions that are highly confident, yet incorrect. In Wang's et al. [42] research, they worked with the Monte Carlo simulation in order to generate parameters for various augmentations such as flipping, scaling, rotation, translation, and noise injections.

Perez et al. [42] have conducted a study that was dedicated to testing various standard data augmentation techniques in context of test-time augmentation. Among them were color augmentation, rotation, shearing, scaling, flipping, random cropping, random erasing, elastic, mixing, and combinations of all the mentioned techniques were tested as well.

The effect of test-time augmentation on classification accuracy is another mechanism for determining a classifier's robustness. As a result, a robust classifier has low prediction variance across augmentations. Predictions do not differ significantly when an image is rotated by 20 degrees.

If an image is rotated 20 degrees, for example, the prediction should not differ significantly. Minh et al. [44] compare accuracy on un-augmented data to accuracy on augmented data in their experiments, searching for augmentations with Reinforcement Learning. When evaluated on augmented test images, the model's performance drops from 74.61 to 66.87 percent.

Some classification models put heavier emphasis on the importance of speed. This suggests that methods for gradually increasing prediction confidence hold promise. This could be accomplished by first producing a prediction with little or no test-time augmentation and then gradually increasing the confidence of the prediction.

4. Practical experiment

4.1 Concept of the practical experiment

In the last chapter, a rather extensive range of data augmentation methods for image processing has been discussed. This part of the paper will look at their technical implementation and application.

The experiment is a PyCharm Project (but is also provided in a Jupiter-Notebook format) of working with a “GTSRB - German Traffic Sign Recognition Benchmark” dataset. The purpose of this part is to look at what effect some data augmentations have the learning process and the result and what additional training measures lead to optimal model training.

The experiment will be conducted in multiple iterations, using slightly altered version of the project: with or without data augmentation, with data augmentation in feature space or in phase of data processing as well as with different number of epochs used for training.

Note: machine learning with relatively large datasets uses a lot of computer resources and time in order to bring the learning process to completion. In the case of data augmentation, the dataset sometimes grows several times larger, which slows down the training algorithm. For this reason, the files containing the practical part and provided by the author to the reader are used for the first training without data augmentation (for future comparison). The author transferred and ran some of the iterations of the experiment that used data augmentation to Google Collab Notebook for a more optimal processing speed, as Google Collab allows working with Google cloud resources, which allows for a much faster training process.

4.2 Experiment: Traffic Sign Classification with Data Augmentation

When working through research papers that cover data augmentation, CIFAR-10 is very often used as popular dataset for is a popular dataset for research on machine learning in context of image processing and image-text classification. It is an interesting visual dataset that helps to diversify the dataset and learning process. But for this experiment, it has been decided to take a dataset that is remotely closer to a real world dataset and that would be used for a specific practical purpose.

The goal of this experiment is to show, how a process would look when working on a „real world“ dataset, where and how data augmentation is being used, and what effect they have on the learning process. This experiment does not focus solely on the data augmentation effect on machine learning. The focus is to see what effect data augmentation has on a model in combination with normalization techniques and working with a dataset that is not built for research or presentation purposes. The attributes that will be looked at will be the ETA (estimated time of arrival or the time it takes for a model to go through a learning epoch), training accuracy, training loss, validation accuracy, validation loss, test accuracy, test loss.

4.2.1 Data

For the second experiment, the German Traffic Sign Recognition Benchmark (GTSRB) dataset has been selected. This dataset contains 60 000 samples (50 000 for training, 10 000 for testing) and 43 different classes as well as meta data in the form of non real world images of traffic signs (see below).

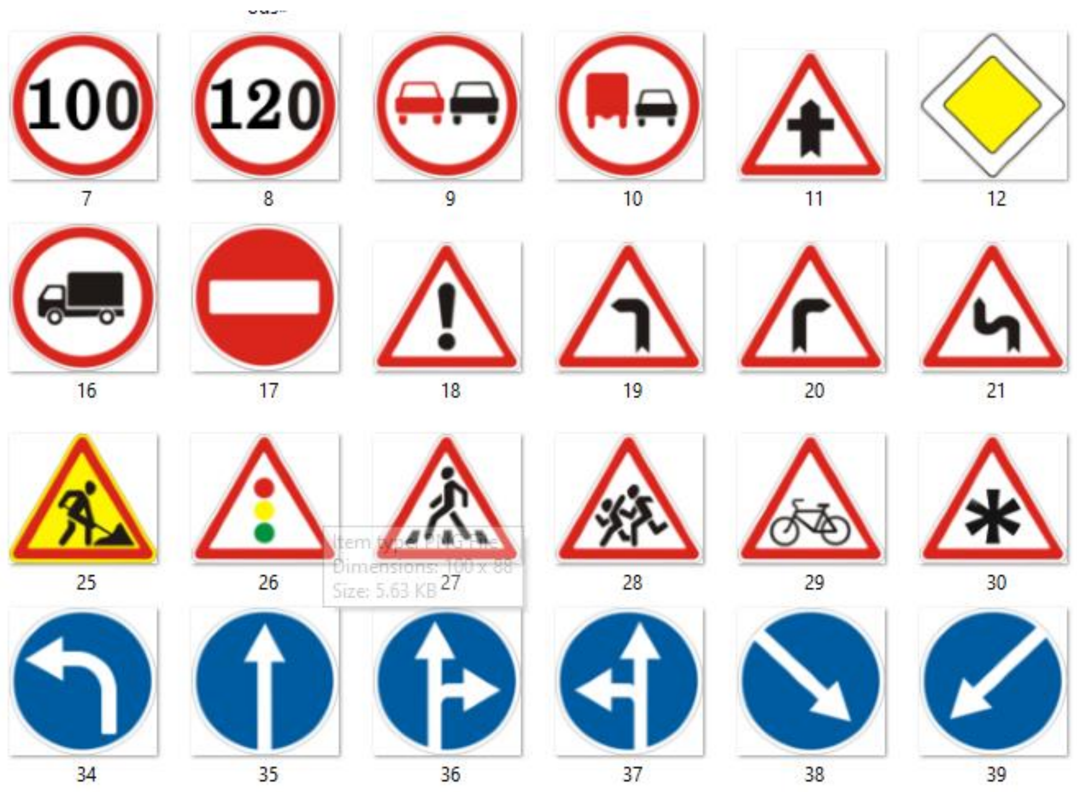


Figure 7: Snippet of meta data from GTSRB dataset

The GTSRB is a suitable candidate for such an experiment since it is a very broad and diverse dataset, that contains samples from a homogenous thematic field but brings variety. It poses some challenges to the experiment, which have to be dealt with. One of them is that the traffic sign images have low resolution and poor contrast. The other is class imbalance. For example the class depicting the traffic sign „Speed limit (50km/h)“ has 2010 samples in the dataset. „Speed limit (30km/h)“ has 1980 samples. However, „Speed limit (20km/h)“ only has only 180 samples. It is a representation of the real world since the 50km/h and 30km/h are encountered way more often than 20km/h. Nevertheless, it is still a challenge for the model, if the goal is to have a high accuracy and an optimal generalizability. Class imbalance is represented more precisely in the Figure below.

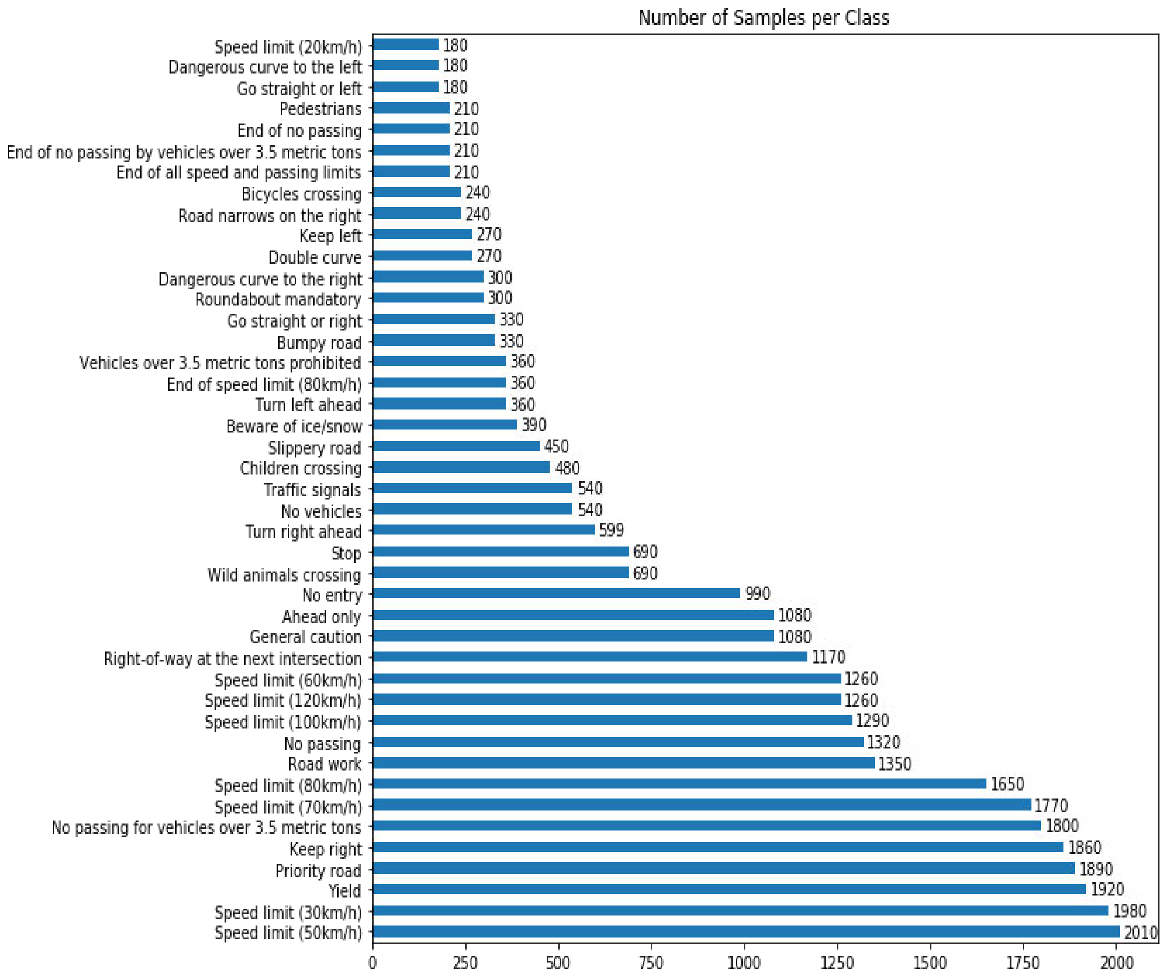


Figure 8: representation of class distribution in GTSRB dataset

4.2.2 Implementation:

The project is implemented using PyCharm and has specific arguments that need to be passed to the execution path in order for it to run correctly. This is all stored in a README file for better clarification.

4.2.2.1 The model

The model is implemented in the class TraficSignCNN.py. In this file, the initialization of the CNN is taking place per static call of the build-method. It is a sequential class model, and its structure is realized as follows:

1. The input layer consists of a convolutional layer => “relu” activation function => batch normalization => pooling layer.
2. First set of layers is a set of (convolutional layer => “relu” activation function => batch normalization)*2 => pooling layer.
3. Second set of layers is a set of (convolutional layer => “relu” activation function => batch normalization)*2 => pooling layer.

   Note: the sets make the learning of the model and reduce volume dimensionality by stacking two sets of convolutional layers with activation functions and normalization layer before applying a max pooling layer.
4. First set of fully connected layers is (flatten layer=> dense layer=> „relu“ activation => batch normalization=> dropout normalization)
5. Second set of fully connected layers is (flatten layer=> dense layer=> „relu“ activation function => batch normalization=> dropout normalization)
6. Finally a classifier layer (output layer) consists of dense layer => „softmax“ activation function

4.2.2.2 Data preprocessing

The image dataset is stored locally within the project.

- In this step, the images are being retrieved from the datasets training and testing directories separetely.
- The data is being randomly shuffled to avoid samples of a particular class following in a sequential order.
- The data is then splitted into labels and images and images go through a preprocessing phase.

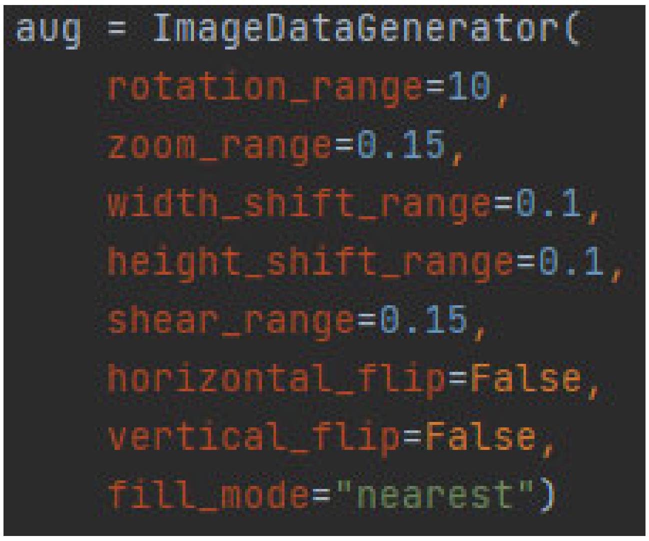
- As mentioned before, the images in the dataset have low contrast, making it challenging for the model to destinguish between the input data. Because of this the class ImageProcessor.py is applying Contrast Limited Adaptive Histogramm Equalization (CLAHE) to all the images in order to improve their contrast [50]. While our images may look "unnatural", the increased contrast will help our model distinguish our traffic signs automatically.
- The images are also being resized to be 32x32 pixels, ignoring aspect ratio. It is done so because the image data in the dataset is not of the same height and width and this way it is normalized.
- Images and labels are then stored in arrays and converted to NumPy arrays.
- After that label names and their respective Ids are retrieved from „signnames.csv“.
- The numpy arrays are then being normalized. Image data is divided by 255.0 to be scaled down to the range from 0 to 1. Label data is categorized using one-hot encoding [51].
- The next step is the calculation of class weight. It is an important step, since we have to account for the class imbalance in the dataset. That way, each class gets a weight assign to it.
- The data is preprocessed and ready to be served as an input to our model.

4.2.2.3 Training

4.2.2.3.1 Data Augmentation

This is where data augmentation finally comes into play. The ImageDataGenerator class is being used as a data augmentation tool. It will perform data augmentation on the test set at the feature level (or will be left out if an iteration without data augmentation is running).

Initial implementation uses methods like random rotation, zoom, shift, shear, and settings flipping for our training data. Horizontal and vertical flipping methods have been excluded, since there is no practical need for their use. It is not expected for the network to ever encounter a real-life flipped traffic sign. ImageDataGenerator has been used to both augment data in preprocessing as well as in feature space.



```
aug = ImageDataGenerator(
    rotation_range=10,
    zoom_range=0.15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.15,
    horizontal_flip=False,
    vertical_flip=False,
    fill_mode="nearest")
```

Figure 9: Implementation of class that performs geometric transformation augmentations

Gaussian injection has been implemented with its own method and applied to the dataset in its iteration.

4.2.2.3.2 Training initialization

The model is initialized by a static call to the TrafficSignCNN class. It is then compiled and the learning process starts either with ImageGenerator performing data augmentations on the dataset on the fly or with the unchanged training dataset (in case augmentation is not activated in the iteration). After that, the model is evaluated on a test dataset and stored locally for future predictions in the „output“ directory along with the plot of training and testing accuracy and loss. A classification report is also written to the console.

Predicting: In order to see the results of the experiment and not only statistics, predict.py has been implemented. It has no use of going into detail and dedicating a whole chapter to it, since all that is being done is once again preprocessing of image data and then delivering it as input to the model. The model will classify each given image, write its output on that image, and store it in the „examples“ directory.

5. Evaluation and results

Methods such as geometric transformations and noise injection were used in the experiment. In different iterations, these methods were applied at different stages of the learning process. The experiment went through four different iterations, each time having a brand new model work with the GTSRB dataset: no data augmentation, noise injection in preprocessing, Geometric transformations in preprocessing, geometric transformations in feature. Each iteration was conducted with 20 epochs.

Note: there were more iterations made in Google Collab (they are located in directory “da_method_implementation”), but only these 4 resulted in plausible results that are worth considering. All of these iterations have been conducted with local computational resources, so the time it takes for each iteration to run from start to finish is significantly higher, than on Google Cloud. The performance difference is immense, since it took only around 10 seconds to preprocess all the training and testing data and start the learning process in Google Collab, but it took every iteration over 20 minutes to just preprocess the data with local computational resources, before even starting the learning process. Nevertheless, we are going to look at these 4 iterations and compare them without having the benefit of the doubt that some of them ran faster due to the utilization of Google Cloud GPU.

1. No data augmentations

In one of the first iterations, the version of the project ran without any data augmentations conducted at all. The performance of the model was poor. Signs of overfitting were clear to see. The model reached an accuracy of 80% on training data (which is still not good enough) and only 62% on testing data. The entire process was finished in 33 minutes. The results for every epoch are depicted in Table 1 (see below).

Accuracy on training: 0.8048

Accuracy on testing: 0.624000012874603

Loss on Testing: 1.48353326320648

Average ETA of an epoch: 48.15 seconds

Note: this being the first iteration, some metrics were not included in the epoch report, such as validation loss and validation accuracy.

Table 1: Learning results with no data augmentation

Epoch	Training Accuracy	Training Loss	ETA
1	0.3265	2.1954	45
2	0.4721	1.4737	50
3	0.5477	1.289	45
4	0.5928	1.1605	59
5	0.626	1.0713	48
6	0.6541	0.9917	43
7	0.6702	0.9435	44
8	0.6945	0.8841	46
9	0.712	0.8355	47
10	0.7235	0.7974	48
11	0.7368	0.7599	51
12	0.7501	0.7254	55
13	0.7585	0.6951	43
14	0.7705	0.6661	45
15	0.7736	0.6523	47
16	0.7787	0.6317	49

17	0.7876	0.604	50
18	0.7944	0.5931	51
19	0.8008	0.5744	50
20	0.8048	0.5593	47

2. Noise Injection in preprocessing

Not the optimal result, but certainly the most curious one: after applying Gaussian noise injection to the dataset, the most unexpected result came out. As expected, the dataset has been artificially inflated with the help of data augmentation and has become double the size of the original (40 000 to 80 000). The process time for each epoch grew significantly to an average of 163.9 seconds, which is 3.4 times higher. The preprocessing, though expected to be longer, lasted 25 minutes longer than in the first iterations. One must forget that every sample has been augmented, inserted into the data array, and then normalized. But what was surprising is that the model reached only 65% accuracy on training data and 91.4% on validation and testing data. This can be attributed to the fact that testing data has not been altered and the augmented training data input consisted of 50% of data augmented with noise injection. This led to the model failing to classify the augmented data samples, but still training it to recognize the original images. A review of training results can be seen in Table 4 (see below).

Accuracy on training: 0.6575

Accuracy on testing: 0.9144

Loss on Testing: 0.2702

Average ETA of an epoch: 163.9 seconds

Note: Noise injection in feature space has also been implemented and used for probe, but the results that came out after a 50-minute training period did not seem plausible and were discarded due to potential wrong implementation of the method. Training accuracy was below 10% in every epoch (sometimes even close to 1-2%) and the same situation was with testing accuracy. The loss was naturally extremely high. The correct predictions seemed to be more random than systematic. For this reason, this iteration was discarded.

Table 2: Learning result with Noise Injection in preprocessing

Epoch	Training accuracy	Training loss	Validation accuracy	Validation loss	ETA (seconds)
1	0.0481	9.4514	0.0705	3.4986	156
2	0.2165	7.0056	0.1384	4.58	174
3	0.3311	6.0092	0.5456	1.5315	205
4	0.3847	5.5721	0.644	1.1903	160
5	0.4126	5.3362	0.8344	0.5525	162
6	0.4349	5.1542	0.6325	1.2904	157
7	0.4498	5.021	0.8748	0.4214	158
8	0.4592	4.9235	0.8406	0.5002	156
9	0.4684	4.8289	0.8898	0.3596	161
10	0.4777	4.7548	0.9093	0.3009	165
11	0.4842	4.6537	0.8265	0.5428	152
12	0.4961	4.4394	0.641	1.1856	167
13	0.5436	3.731	0.8856	0.3869	159
14	0.5824	3.2976	0.8989	0.3512	165
15	0.6031	3.0669	0.8923	0.3607	168
16	0.6182	2.896	0.9044	0.3133	167
17	0.6278	2.7799	0.8366	0.5445	166
18	0.6405	2.6702	0.9154	0.2846	162
19	0.6499	2.5792	0.9093	0.3013	167
20	0.6575	2.4979	0.9144	0.2702	151

3. Geometric Transformations in Preprocessing

Similar as with noise injection, augmentation methods from the class ImageDataGenerator have been applied to the training dataset in the processing phase. The chosen set up for data augmentation performed a combination of five geometrical image data augmentation methods on each data sample, inflating the dataset to the size of 80 000. Significantly the longest of all iterations (successful and unsuccessful ones), this iteration took 78.5 minutes to complete, but had an average ETA of 164.75 seconds. The results can be seen in Table 3 (see below). It is possible to see uneven fluctuations throughout the validation accuracy and loss. For example, while training accuracy grows steadily, validation accuracy spikes in epoch 6 from 41% to 74%, grows steadily during the next 3 epochs and then drastically falls to 4% in epoch 9. It grows steadily after this, and spikes again to 49% in epoch 13. A logical explanation would be that the application of data augmentation methods in preprocessing increases the amount of data samples (2x), but the implementation has a major flaw in that the original dataset is initially randomly shuffled so that same classes do not follow each other in a long sequence, but then they are being processed in a loop and during this stage augmentations take place and basically create an augmented version of the same sample and then insert it into the array (which is later normalized and passed to the network) right after its original image. This results in a very unbalanced array, where original samples and its augmented counterparts come in pairs. So, when the classes with less representation in the dataset come as validation data, the network suddenly drops its performance drastically since it cannot distinguish the underrepresented classes that well. The same happens when an overrepresented class appears in the validation set, but in reverse – accuracy spikes up unnaturally.

Accuracy on training: 0.7125

Accuracy on testing: 0.7344

Loss on Testing: 1.199

Average ETA of an epoch: 164.75 seconds

Table 3: Learning result in geometric transformations in preprocessing phase

Epoch	Training accuracy	Training loss	Validation accuracy	Validation loss	ETA (seconds)
1	0.0472	9.5404	0.1414	3.6547	158
2	0.1935	7.224	0.2384	3.886	171
3	0.3069	6.2271	0.3156	2.996	158
4	0.3728	5.707	0.3944	2.87	157
5	0.4109	5.3756	0.4144	2.4331	162
6	0.4324	5.1708	0.7425	0.8576	162
7	0.4475	5.0099	0.7948	0.7214	164
8	0.4608	0.8781	0.8006	0.5002	160
9	0.4711	4.7609	0.0498	5.2699	159
10	0.4907	4.4448	0.0593	0.5167	160
11	0.5474	3.6877	0.0765	3.657	165
12	0.5841	3.288	0.141	4.1145	166
13	0.61	3.0199	0.4956	1.974	168
14	0.6365	2.7256	0.7389	0.7512	185
15	0.6579	2.526	0.8023	0.3793	181
16	0.6726	2.3959	0.7444	0.8633	158
17	0.6852	2.2915	0.6266	1.3095	164
18	0.6926	2.213	0.9054	0.3302	168
19	0.7016	2.1349	0.2393	3.3455	166
20	0.7125	2.0499	0.7344	0.7702	163

4. Geometric Transformations in Feature Space

The final iteration sees geometric data transformations take place in feature space. This iteration took 41 minutes to complete and had the best result yet. Table 4 shows that the network's training accuracy starts at over 30% and strides upward until it reaches the mid and upper 90% and just keeps fluctuating in that area. Validation accuracy also makes a leap from epoch 1 to 2 (63%), and then steadily grows in the upper direction. This is the closest the network got to performing optimally and not overfitting. The results of this training sequence can be viewed in Figure 10. It is clearly visible, that both training and testing accuracy develop steadily and converge slowly towards 100%. Training loss also converges towards 0.

Accuracy on training: 0.9794

Accuracy on testing: 0.9529

Loss on Testing: 0.1753

Average ETA of an epoch: 91.35 seconds

Table 4: Learning with geometric augmentations in feature space

Epoch	Training accuracy	Training loss	Validation accuracy	Validation loss	ETA (seconds)
1	0.3031	6.2238	0.1708	3.1653	82
2	0.6602	2.399	0.801	0.6317	98
3	0.7983	1.3471	0.8826	0.4057	93
4	0.8579	0.9124	0.8988	0.3383	86
5	0.8941	0.6504	0.8677	0.457	89
6	0.9148	0.5352	0.9276	0.2479	86
7	0.9292	0.4413	0.9162	0.2813	92
8	0.9409	0.3695	0.9359	0.215	78
9	0.9478	0.3175	0.9441	0.1895	97
10	0.9531	0.282	0.9429	0.2016	98
11	0.9592	0.2384	0.9401	0.2025	92
12	0.962	0.2272	0.9256	0.2522	83
13	0.9647	0.2158	0.9423	0.1989	101
14	0.9684	0.1889	0.9345	0.2068	88
15	0.9735	0.1557	0.946	0.229	83
16	0.9743	0.1495	0.946	0.2056	101
17	0.9765	0.1379	0.9143	0.2	89
18	0.9768	0.1334	0.9143	0.317	95
19	0.9784	0.1274	0.9461	0.2049	100
20	0.9794	0.1178	0.9529	0.1753	96

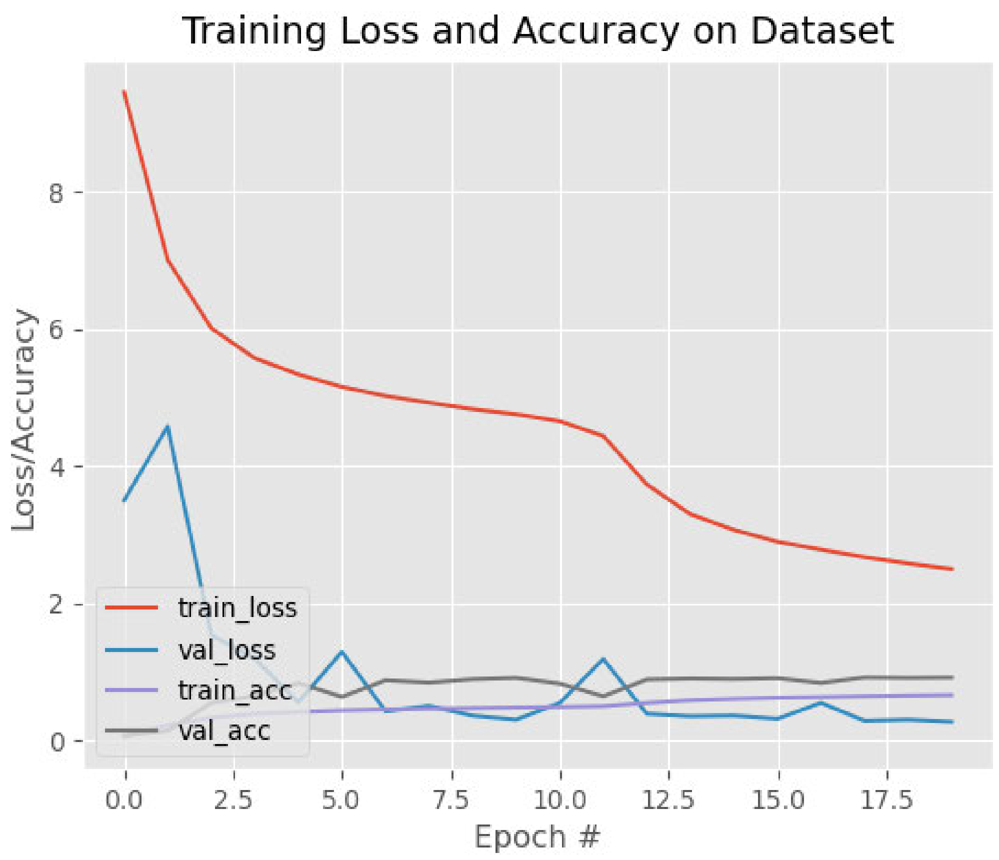


Figure 10: Training accuracy, Training loss, Validation accuracy, Validation loss plotted after learning with geometric data augmentations in feature space

Iteration 4 clearly achieved the set goal of performing with maximal accuracy, both in testing and training, and at optimal time. The Predict script can allow the reader to see for himself, how the network classifies the input data correctly. A snippet of the prediction examples can be seen in Figure 11 (see below).

5.1 Experiment conclusion

This experiment has shown clearly that geometric data augmentations are way more effective when applied in combination and in feature space. It resulted in the most optimal result of all and in a decent amount of computational time. It did not require that many computational resources, and it got the job done, whereas using them directly on the dataset in preprocessing phase has

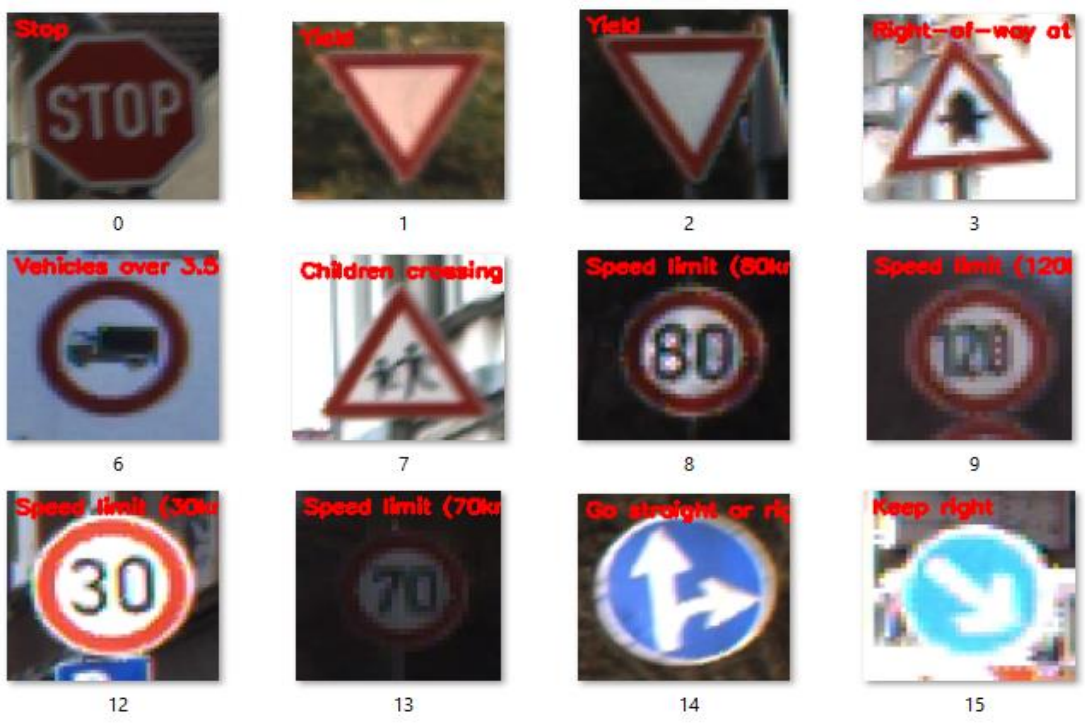


Figure 11: Snippet of example image data samples in "examples" directory with predicted labels written on respective sample

proven to do more harm than good in case of the GTSRB dataset. It has also been shown that this approach functions well when combined with popular normalization methods (Batch normalization and Dropout normalization were both part of the neural network).

6. Conclusion

The intriguing methods for enhancing image data fall into two broad categories: data warping and oversampling. The methods and techniques of data augmentation have ranged from the most primitive to the most difficult to implement and even the least explored. Data augmentation is an extremely useful tool in combatting overfitting and generalizing a model. Yet this field is still genuinely unexplored and has a lot of research potential. The big question of data augmentation, which helps with generalization is whether it itself is generalizable? Different techniques are appropriate for different tasks, but there is no common understanding of which techniques to use for which tasks. A lot of scientific work in the field of data augmentation is based on trying and discovering the smallest usefulness of different techniques for different tasks. Data augmentation techniques are difficult to qualify, especially when they can be combined in different ways. For certain tasks, geometric data transformations are known to be suitable, for example. But even in this situation, there is no clear answer as to which combination is best to solve the problem or to improve the performance of the model in a way that other techniques or combinations of techniques cannot. Many of the augmentation methods explain how to improve the image classifier, while others do not. GANs have a great potential to generalize standard data augmentation methods and generalize the concept (at least partially) and maybe even help automate it? A vast part of data augmentation still revolves around human choice. The development of software tools is an important area of future work for the practical integration of data augmentation into deep learning workflows. If data augmentation libraries will automate preprocessing functions in the same way that the Tensorflow system automates the back-end processes of gradient-descent learning, then the human factor might play an insignificant role in choosing and applying data augmentation to combat overfitting.

Another fascinating practical question is determining the size of a dataset post augmentation. Clearly, inflating the dataset 2 times or even more is not always an optimal solution. There is no agreement on which original-to-final dataset size ratio produces the best performing model. If we take solely color augmentations and an initial dataset of with 2 classes, 50 samples each. If each image is augmented with 100 color filters to yield 5000 samples of both classes, then the resulting dataset will be heavily biased toward the spatial characteristics of the original 50 samples of each class. Due to the abundance of color-augmented data, a deep model will outperform the original [22].
Furthermore, there is no unanimity of opinion on the best strategy for combining the two types: data warping and oversampling. The inherent bias in the initial, limited dataset is an important factor. There are currently no augmentation techniques that can correct a dataset with very low diversity in comparison to the testing data. All of these augmentation algorithms perform best when the training and testing data are drawn from the same distribution. If this is not the case, these methods are unlikely to be useful.[22]

Yet again, there is a lot of ground to be explored when it comes to the potential of GANs and also test-time augmentation. The latter has not been explored much yet and has a potential to bring significant benefits to the field of computer vision and help the field of machine learning improve drastically on a grand scale.

Bibliography

1. Libbrecht, M. W., & Noble, W. S. (2015). Machine learning applications in genetics and genomics. Nature Reviews Genetics, 321–332
2. Alpaydın, E. (2014). Introduction to machine learning. Cambridge, MA: MIT Press.
3. Berry, M. W., Azlinah, M., Yap, B.W., 2020, Supervised and Unsupervised Learning for Data Science, Springer Nature Switzerland AG, https://doi.org/10.1007/978-3-030-22475-2, Retrieved on 20.04.2022
4. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436-444.
5. Gupta, P., Sharma, A., & Jindal, R. (2016). Scalable machine-learning algorithms for big data analytics: a comprehensive review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *6*(6), 194-214.
6. Traeger, M., Eberhart, A., Geldner, G., Morin, A. M., Putzke, C., Wulf, H., & Eberhart, L. H. J. (2003). Künstliche neuronale Netze. *Der Anaesthesist*, *52*(11), 1055-1061
7. Zell, A. (1994). *Simulation neuronaler netze* (Vol. 1, No. 5.3). Bonn: Addison-Wesley.
8. Xu, J., & Bao, Z. (2002). Neural networks and graph theory. *Science in China Series F: Information Sciences*, *45*(1), 1-24.
9. Cilimkovic, M. (2015). Neural networks and back propagation algorithm. *Institute of Technology Blanchardstown, Blanchardstown Road North Dublin*, *15*(1).
10. Medsker, L. R., & Jain, L. C. (2001). Recurrent neural networks. *Design and Applications*, *5*, 64-67.
11. Zhang, L. (2017, July). Implementation of fixed-point neuron models with threshold, ramp and sigmoid activation functions. In *IOP Conference Series: Materials Science and Engineering* (Vol. 224, No. 1, p. 012054). IOP Publishing.
12. Ghahramani, Z. (2004). Unsupervised Learning. In: Bousquet, O., von Luxburg, U., Rätsch, G. (eds) Advanced Lectures on Machine Learning. ML 2003. Lecture Notes in Computer Science(), vol 3176. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-28650-9_5
13. Wang, H., Lei, Z., Zhang, X., Zhou, B., & Peng, J. (2016). Machine learning basics. *Deep learning*, 98-164.
14. Kearns, M. J. (1990). *The computational complexity of machine learning*. MIT press.
15. Weisstein, E. W. (2003). Convolution. *https://mathworld. wolfram. com/*. Retrieved on 03.05.2022

16. Guo, T., Dong, J., Li, H., & Gao, Y. (2017, March). Simple convolutional neural network on image classification. In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)* (pp. 721-724). IEEE.
17. D. M. Hawkins, The Problem of Overfitting, 2004, *Journal of Chemical Information and Computer Sciences, 44* (1), 1-12 DOI: 10.1021/ci0342472
18. Koehrsen, W. (2018). Overfitting vs. underfitting: A complete example. *Towards Data Science*.
19. Zur, R. M., Jiang, Y., Pesce, L. L., & Drukker, K. (2009). Noise injection for training artificial neural networks: A comparison with weight decay and early stopping. *Medical physics*, *36*(10), 4810-4818.
20. L. Taylor and G. Nitschke, "Improving Deep Learning with Generic Data Augmentation," *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2018, pp. 1542-1547, doi: 10.1109/SSCI.2018.8628742.
21. Wong, S. C., Gatt, A., Stamatescu, V., & McDonnell, M. D. (2016, November). Understanding data augmentation for classification: when to warp?. In *2016 international conference on digital image computing: techniques and applications (DICTA)* (pp. 1-6). IEEE.
22. Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, *6*(1), 1-48.
23. Quanzeng Y, Jiebo L, Hailin J, Jianchao Y. Robust image sentiment analysis using progressively trained and domain transferred deep networks. In: AAAI. 2015, p. 381–8
24. Kang, G., Dong, X., Zheng, L., & Yang, Y. (2017). Patchshuffle regularization. *arXiv preprint arXiv:1707.07103*.
25. Summers, C., & Dinneen, M. J. (2019, January). Improved mixed-example data augmentation. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 1262-1270). IEEE
26. Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.
27. Tokozume, Y., Ushiku, Y., & Harada, T. (2018). Between-class learning for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5486-5494).
28. Zhong, Z., Zheng, L., Kang, G., Li, S., & Yang, Y. (2020, April). Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 34, No. 07, pp. 13001-13008).

29. Inoue, H. (2018). Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*.
30. Konno, T., & Iwazume, M. (2018). Icing on the cake: An easy and quick post-learnig method you can try after deep learning. *arXiv preprint arXiv:1807.06540*.
31. Antoniou, A., Storkey, A., & Edwards, H. (2017). Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*.
32. Jain, A. K., Mao, J., & Mohiuddin, K. M. (1996). Artificial neural networks: A tutorial. *Computer*, *29*(3), 31-44.
33. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*.
34. Seyed-Mohsen MD, Alhussein F, Pascal F. DeepFool: a simple and accurate method to fool deep neural networks. arXiv preprint. 2016.
35. Su, J., Vargas, D. V., & Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, *23*(5), 828-841.
36. Zajac, M., Zołna, K., Rostamzadeh, N., & Pinheiro, P. O. (2019, July). Adversarial framing for image and video classification. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 01, pp. 10077-10078).
37. Xie, L., Wang, J., Wei, Z., Wang, M., & Tian, Q. (2016). Disturblabel: Regularizing cnn on the loss layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4753-4762).
38. Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.
39. Fang, W., Zhang, F., Sheng, V. S., & Ding, Y. (2018). A method for improving CNN-based image recognition using DCGAN. *Computers, Materials and Continua*, *57*(1), 167-178.
40. Shanmugam, D., Blalock, D., Balakrishnan, G., & Guttag, J. (2020). When and why test-time augmentation works. *arXiv e-prints,* arXiv-2011.
41. Wang, G., Li, W., Ourselin, S., & Vercauteren, T. (2018, September). Automatic brain tumor segmentation using convolutional neural networks with test-time augmentation. In *International MICCAI Brainlesion Workshop* (pp. 61-72). Springer, Cham.
42. Wang, G., Li, W., Aertsen, M., Deprest, J., Ourselin, S., & Vercauteren, T. (2018). Test-time augmentation with uncertainty estimation for deep learning-based medical image segmentation.

43. Perez, F., Vasconcelos, C., Avila, S., & Valle, E. (2018). Data augmentation for skin lesion analysis. In *OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis* (pp. 303-311). Springer, Cham.

44. Minh, T. N., Sinn, M., Lam, H. T., & Wistuba, M. (2018). Automated image data preprocessing with deep reinforcement learning. *arXiv preprint arXiv:1806.05886*.

45. C. Khosla and B. S. Saini, "Enhancing Performance of Deep Learning Models with different Data Augmentation Techniques: A Survey," *2020 International Conference on Intelligent Engineering and Management (ICIEM)*, 2020, pp. 79-85, doi: 10.1109/ICIEM48762.2020.9160048.

46. Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, *4*(11), e00938.

47. Sazli, M. H. (2006). A brief review of feed-forward neural networks. *Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering*, *50*(01).

48. Takahashi, R., Matsubara, T., & Uehara, K. (2019). Data augmentation using random image cropping and patching for deep CNNs. *IEEE Transactions on Circuits and Systems for Video Technology*, *30*(9), 2917-2931.

49. Santurkar, S., Tsipras, D., Ilyas, A., & Madry, A. (2018). How does batch normalization help optimization?. *Advances in neural information processing systems*, *31*.

50. Setiawan, A. W., Mengko, T. R., Santoso, O. S., & Suksmono, A. B. (2013, June). Color retinal image enhancement using CLAHE. In *International Conference on ICT for Smart Society* (pp. 1-3). IEEE.

51. Choong, A. C. H., & Lee, N. K. (2017, November). Evaluation of convolutionary neural networks modeling of DNA sequences using ordinal versus one-hot encoding method. In *2017 International Conference on Computer and Drone Applications (IConDA)* (pp. 60-65). IEEE.

52. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929-1958.

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen.“

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Dieses Blatt, mit der folgenden Erklärung, ist nach Fertigstellung der Abschlussarbeit durch den Studierenden auszufüllen und jeweils mit Originalunterschrift als letztes Blatt in das Prüfungsexemplar der Abschlussarbeit einzubinden.
Eine unrichtig abgegebene Erklärung kann -auch nachträglich- zur Ungültigkeit des Studienabschlusses führen.

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: Nikita

Vorname: Ostrovskiy

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Comparison of Data Augmentation Techniques for efficient Training of Image-Text Classification Algorithms

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der -bitte auswählen- ist erfolgt durch:

Hamburg	08.06.2022	[redacted]
Ort	Datum	Unterschrift im Original