CARL VON OSSIETZKY universität OLDENBURG

Department für Informatik
Abteilung für Computational Intelligence

Adversarials^{-1}: Detecting Adversarial Inputs with Internal Attacks

Von der Fakultät für Informatik der Carl von Ossietzky Universität Oldenburg zur Erlangung des Grades und Titels eines

Doktor der Naturwissenschaften (Dr. rer. nat.)

angenommene Dissertation von

Herrn Nils Steffen Worzyk

geboren am 26.04.1992 in Hannover

Erstgutachter: Prof. Dr. Oliver Kramer
Externer Zweitgutachter: Prof. Dr. Mike Preuss

Tag der Disputation: 13.05.2020

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Außerdem versichere ich, dass ich die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichung, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe.

Nils Steffen Worzyk
Matrikelnummer 1427487
Oldenburg, den 24. März 2020

Zusammenfassung

Tiefe neuronale Netze (engl. deep neural networks (DNNs)) sind sehr erfolgreich bei verschiedenen anspruchsvollen Aufgaben, z.B. bei der Bild- und Sprachklassifikation. In den letzten Jahren gab es jedoch eine rasante Entwicklung bei den sogenannten gegnerischen Angriffen. Die Grundidee der gegnerischen Angriffe besteht darin, einen vorhandenen Input zu manipulieren oder einen neuen Input zu erzeugen, der von einem DNN als eine bestimmte Klasse klassifiziert wird, während der Input von einem Menschen völlig anders klassifiziert wird. Ein Beispiel, auf das im Zusammenhang mit gegnerischen Eingaben oft Bezug genommen wird, weil es höchst sicherheitskritisch ist, ist das autonome Fahren. In diesem Zusammenhang ist ein übliches Szenario die Manipulation eines Stoppschildes, damit es von einem DNN als Vorfahrtschild erkannt wird, während ein Mensch noch immer ein Stoppschild erkennen würde.

Nachdem die Grundlagen des maschinellen Lernens, insbesondere neuronale Netze, und gegnerische Eingaben im Allgemeinen vorgestellt wurden, wurde ein neues gegnerisches Angriffsszenario eingeführt und untersucht. Um auf das Beispiel der Manipulation von Straßenschildern zurückzukommen, wurde in der bisherigen Literatur üblicherweise davon ausgegangen, dass ein Angreifer das Straßenschild physisch manipuliert. Zum Beispiel durch das Abdecken des ursprünglichen Straßenschildes mit einer manipulierten und gedruckten Version oder durch das Anbringen von graffitiähnlichen Aufklebern oder Streifen auf dem ursprünglichen Straßenschild. Bei der vorgeschlagenen Methode wurde die Notwendigkeit der physischen Manipulation dadurch ersetzt, dass die Störungen mit einem Videoprojektor auf das Straßenschild projiziert wurden. Dadurch sind Angriffe wahrscheinlich schwieriger zu erkennen und im Falle eines Unfalls ist es schwieriger, das Versagen des autonomen Fahrsystems zu beweisen. Durch die Durchführung dieser Arbeit wurde verstärkt, wie anfällig DNNs gegenüber manipulierten Eingaben sind, und die Notwendigkeit zuverlässiger Abwehrmechanismen erhöht.

Anschließend wurde eine Abwehrtechnik zur Erkennung und Korrektur von gegnerischen Eingaben im Bereich der Bildklassifikation eingeführt. Die Motivation für diesen Ansatz war, dass gegnerische Eingaben unvermeidlich sind, und es wurde vorgeschlagen, diese durch interne gegnerische Angriffe als Mittel der Verteidigung zu erkennen. Nach einem erneuten Angriff werden gegnerische Inputs als adversarial^{-1} Inputs bezeichnet. Allgemeiner werden Eingaben nach dem internen Angriff als internes Gegenstück bezeichnet. In einer Vorstudie wurde der pixelweise Unterschied zwischen den ursprünglichen und gegnerischen Eingaben, und ihrem entsprechenden internen Gegenstück untersucht. Insbesondere in Bezug auf die pixelweise L_2- und L_∞-Norm war für die Änderung der Klassifizierung eines ursprünglichen Inputs mehr Störung erforderlich als für die Änderung der Klasse eines bereits manipulierten gegnerischen Inputs. Außerdem kehrten bis zu 89,94% der ursprünglich gegnerischen Eingaben nach dem internen Angriff in ihre ur-

sprüngliche Klasse zurück. Auf der Grundlage dieser Beobachtungen wurden interne Klassifizierer trainiert, um gegnerische Eingaben zu erkennen und sie in die ursprünglich richtige Klasse zu transformieren.

Um zu zeigen, dass das Gesamtkonzept verallgemeinerbar ist, wurde der Prozess auf die Sprachklassifizierung übertragen. Da bei der Sprachklassifikation die Angriffe teilweise andere Eigenschaften haben als in der Bildklassifikation, musste der Mechanismus leicht angepasst werden. Es war möglich, eine dem Stand der Technik entsprechende kontradiktorische Erkennungsgenauigkeit zu erreichen, und außerdem konnte die ursprüngliche wahre Klasse bis zu einem gewissen Grad wiederhergestellt werden.

Wenn dem Angriff jedoch mehr Wissen über die Verteidigung zur Verfügung gestellt wird, war es möglich, Eingaben zu konstruieren, die von einem unverteidigten Bildklassifikator korrekt klassifiziert wurden, aber die vorgeschlagene Verteidigung erkennt sie als gegnerische Eingaben. Daher wurde untersucht, ob es möglich ist, die Richtung zu bestimmen, in der eine unbekannte Eingabe die Entscheidungsgrenze des Bildklassifikators durch den internen Angriff überschreitet. Dazu wurden die Ausgaben der Zwischenschichten des Bildklassifikators untersucht. Basierend auf diesen Ausgaben wurde die Mahalanobis-Distanz zu bereits gefundenen Clustern berechnet, insbesondere wurden die Unterschiede zwischen den Abständen vor und nach dem internen Angriff bestimmt. Die Annahme, dass die Richtung aufgrund der Unterschiede identifiziert werden kann, konnte nicht bestätigt werden. Aber sie könnten dazu verwendet werden, um gegnerische Eingaben zu erkennen, oder noch allgemeiner out-of-distribution data (OOD) (dt. Daten, die außerhalb einer bekannten Verteilung liegen). Schließlich wurden die Fragestellungen gegnerische und OOD Eingaben zu erkennen kombiniert und auf der Grundlage der Unterschiede in der Mahalanobis-Distanz wurden vielversprechende Ergebnisse gefunden.

Abstract

Deep neural networks (DNNs) are very successful in various challenging tasks, e.g., image and speech classification. However, in recent years rapid developments in so-called adversarial attacks were made. The basic idea of adversarial attacks is to manipulate an existing or create a new input, to be classified as a specific class by the DNN, while the input is classified completely different by a human. An example which is often referred to in the context of adversarial inputs, because it is highly safety-critical, is autonomous driving. In that context, one usual scenario is to manipulate a stop sign to be recognised as a priority road sign by the DNN, while a human would still recognise a stop sign.

After presenting the basics on machine learning, in particular neural networks, and adversarial inputs in general, a new adversarial attack scenario was introduced and investigated. Returning to the example of manipulating street signs, previous literature usually considered that an attacker manipulates the street sign physically. For example, by covering the original street sign with a manipulated and printed version, or to attach graffiti-like stickers or stripes to the original street sign. In the proposed method, the necessity of physical manipulation was substituted by projecting the perturbations onto the street sign with a video projector. Thereby, attacks are probably more difficult to detect and in case of an accident, it is more difficult to prove the failure of the autonomous driving system. By conducting this work, it was reinforced how prone DNNs are to manipulated inputs, and therefore enhanced the necessity for reliable defence mechanisms.

Afterwards, a defence technique to detect as well as reform adversarial inputs was introduced in the domain of image classification. The motivation for the approach was that adversarial inputs are inevitable and it was proposed to detect them by applying internal adversarial attacks as a means to defend. Adversarial inputs, after being attacked again, are called adversarial^{-1} inputs. More generally, inputs after the internal attack are referred to as internal counterpart. In a preliminary study, the pixel-wise difference between initially original and adversarial inputs, and their corresponding internal counterpart were investigated. In particular regarding the pixel-wise L_2- and L_∞-norm, more perturbation was required to change the classification of an original input, than for changing the class of an already manipulated adversarial input. Also, up to 89.94% of initially adversarial inputs returned to their original class, after the internal attack. Based on these observations, internal classifiers were trained to detect adversarial inputs and transform them into the original correct class.

To show that the overall concept is generalisable, the process was transferred to speech classification. Since in speech classification the attacks have partially different properties than in image classification, it was necessary to adapt the mechanism slightly. It was possible to achieve state of the art adversarial detection accuracy, and besides, the original true class was able to be restored to a certain extend.

However, giving the attack more knowledge about the defence, it was possible to construct inputs which were classified correctly by an undefended image classifier, but the proposed defence detects them as adversarial. Hence it was studied if it is possible to determine the direction in which an unknown samples crosses the decision boundary of the image classifier when attacked internally. To do so, the outputs of the intermediate layers of the image classifier were examined. Based on these outputs the Mahalanobis distance towards previously found clusters was calculated, in particular, the differences between the distances before and after the internal attack were determined. The assumption that the direction can be identified based on the differences, could not be confirmed. But they could be used to detect adversarial inputs, or even more generally out-of-distribution data (OOD). Finally, the problem of adversarial input and OOD detection were combined and based on the Mahalanobis distance differences promising results were found.

Contents

List of Figures

List of Tables

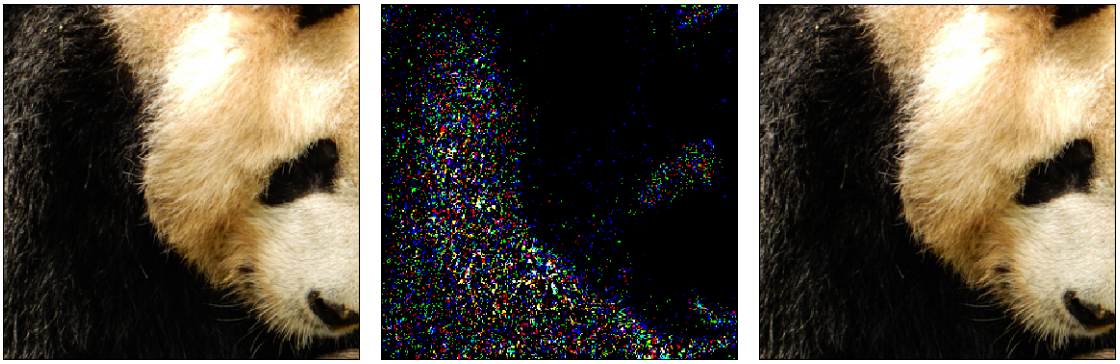
1. Introduction

In recent years, machine learning in the form of so-called deep neural networks (DNNs), has become more and more ubiquitous in everyday devices such as mobile phones or smart home systems. One common application for DNNs is image processing, which is implemented in various scenarios. Some are arguably less sensitive, such as the automatic tagging of images uploaded to cloud storages like *Google Photo* [Goo]. Other applications can be at least security-sensitive, like the usage of face recognition in border control, as used by the United States Customs and Border Protection (CBP) [CP]. Whereas other applications can be even live critical, like image processing in health care [Fin+19] or autonomous driving where image processing is used to detect obstacles, pedestrians, and street signs.

Another common application of DNNs is speech recognition. Smart home devices like Amazon's *Echo Dot* or Google's *Google Home* solely rely on speech commands to assist at home. In some scenarios, the application is again arguably less sensitive, like searching for a specific song or artist to play their music. If the smart home device is connected to a credit card or other smart home devices such as smart locks, voice commands could become at least security-critical. In some cases, voice commands can even become life-critical. For example in semi-automatic cars like the ones produced by Tesla, which can partly be controlled by voice commands.

Pointing out those safety-critical applications is necessary since recent history has shown that DNNs are prone to so-called adversarial inputs. This term was coined by a publication of Szegedy et al.[Sze+14], who in 2013 demonstrated that small perturbations to an input lead to misclassification by the employed DNN. An example of such a manipulation is given in Figure 1.1. The original input in Figure 1.1a is correctly classified as 'giant panda' by a given DNN, while the adversarially manipulated version (see Figure 1.1c) is classified as 'chow-chow'. However, to a human, both images look the same and the perturbation is shown in Figure 1.1b had to be enhanced to be visible.

Revising the safety-critical applications in image processing, it has been exposed that face recognition systems could be fooled to match a given input image to a different identity (e.g., [DZJ19]). This could potentially be used for identity theft during the, e.g., registration process for border crossing. In the health care domain, Finlayson et al. [Fin+19] converted an image of a benign mole to be detected as malignant, which could lead to unnecessary or even harmful treatment. The other way around, manipulating a malignant input to be detected as benign can lead to possible harm because necessary treatment is not provided. In the example scenario of autonomous driving, the probably most prominent way of fooling the image classification is to manipulate a stop sign to be classified as a priority road sign, e.g., as demonstrated by Eykholt et al. [Eyk+18]. Another scenario is pointed out in the work of Ranjan et al. [Ran+19], in which the



(a) Original image, correct classified as *giant panda*.

(b) Perturbation added to the original image, to create the adversarial image.

(c) Adversarial image, classified as *chow-chow*.

Figure 1.1.: Example for an adversarial input, imperceptible for a human being.

authors used a printed patch which hinders the detection of movements. Thereby, for example, pedestrians present in the detected image are not recognised.

More recently, in the speech domain, similar attacks have been applied. For example, Schönherr et al. [Sch+19][1] transformed a spoken sequence, which has initially been correctly recognised and transcribed by a neural network as:

> "Specifically the union said it was proposing to purchase all of the assets of the of the United Airlines including planes gates facilities and landing rights."

After transformation, the spoken sequence is indistinguishable for a human listener but recognised and transcribed by the system as:

> "Deactivate security camera and unlock front door."

The given example demonstrates that if certain smart home devices are present, adversaries can, for example, unlock a home for the intrusion. The necessary audio clips can be hidden in, e.g., YouTube videos, or played via radio as adds. Thereby, such attacks could also be used to attack semi-autonomous cars. Currently, the list of accepted commands might be small, but already directing the mirrors or the steering wheel to adjust rapidly to different positions, could distract the driver and lead to accidents.

1.1. Structure of the Thesis

In this thesis, two aspects of adversarial inputs are investigated, namely adversarial attacks and adversarial defences. To provide a basis, in Chapter 2, the general field of machine learning is introduced. In particular, classification is introduced in Section 2.1 as the general objective in this thesis and supervised learning (Section 2.2) as the considered

[1] `https://adversarial-attacks.net`

training process. In Section 2.3 the models used for the considered classification tasks, so-called artificial neural networks (ANNs), are introduced. Starting from their basic components, backpropagation as a training process deserves special attention, as the process is important for the adversarial attacks.

From Chapter 3 on, the focus is set to adversarial inputs, by first reviewing important concepts like distance measures (Section 3.2) to quantify the quality of adversarial inputs, the general taxonomy of adversarial attacks (Section 3.3), and the taxonomy of adversarial defences (Section 3.4). As the transition to the first contribution, in Section 3.5 the fast gradient sign method (FGSM) is outlined, which is one of the earliest adversarial attacks and implements concepts, which are also used by more recent attacks. Those used in later chapters are introduced in the Section 3.6. Based on the initial observation that adversarial inputs exist in the digital world, they have been transferred to the real world more recently. One of the earlier works considering the transfer to the real world is reviewed in Section 3.7.

Talking about transferring adversarial manipulations into the real world, one common use case are autonomous cars, and in particular, the scenario of manipulating a stop sign to be classified as a priority road sign is a frequent example. Most preliminary work is based on manipulating the street sign physically, by sticking a digitally manipulated print out over the real street sign, or otherwise physically manipulate the environment.

In Chapter 4 a new threat model is proposed, in which the adversarial manipulations are projected onto existing street signs with a home projector, or a limited projector like a laser pointer. After introducing necessary adaptations for the proposed scenario in Section 4.1 and the experimental setup in Section 4.2, the corresponding results are presented in Section 4.3.

The Chapters 5 to 7 focus on a new adversarial defence. The basic idea of the proposed defence is to adversarially attack an unknown input in an internal process. If an adversarial input is attacked internally, the resulting sample is called adversarial^{-1}. In general, the outcome of the internal attack is referred to as internal counterpart of an initial input. Based on the different behaviour of original and adversarial inputs when attacked internally, the aim is to 1) detect and 2) revert adversarial inputs to their original true class. After motivating and outlining the idea in more detail, the overall workflow of the proposed defence is introduced in Section 5.1, followed by the experimental setup in Section 5.2, and the results in Section 5.3. The results in Chapter 5 are based on image classification.

To demonstrate that the approach is generalisable, in Chapter 6 the proposed adversarial defence is transferred to the domain of speech recognition. After an introduction to speech processing in Section 6.1, and the employed adversarial attacks in Section 6.2, the approach and its corresponding adaptations necessary is outlined in Section 6.3. The results are displayed in Section 6.4.

Based on the observations in Chapter 5, in Chapter 7 the existing method is extended to consider the intermediate output of a given image classifier during the decision making process. After explaining the motivations in more detail, the approach with its necessary adaptations is described in Section 7.1, followed by the experimental setup in Section 7.2. In Section 7.3, the corresponding results are reported and discussed.

Finally, Chapter 8 concludes this thesis, and outline possible directions for future research.

1.2. Preliminary Publications

Some results and contributions presented in this thesis have previously been published in the following conference articles:

- Results and concepts of Chapter 5 are partly partly published in:
  - Nils Worzyk and Oliver Kramer. "Properties of adv-1 - Adversarials of Adversarials". In: *26th European Symposium on Artificial Neural Networks, ESANN 2018, Bruges, Belgium, April 25-27, 2018*. 2018.
  - Nils Worzyk and Oliver Kramer. "Adversarials $^{-1}$: Defending by Attacking". In: *2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8-13, 2018*. 2018, pp. 1–8. DOI: 10.1109/IJCNN.2018.8489630.
- Results and concepts of Chapter 4 are partly partly published in:
  - Nils Worzyk, Hendrik Kahlen, and Oliver Kramer. "Physical Adversarial Attacks by Projecting Perturbations". In: *Artificial Neural Networks and Machine Learning - ICANN 2019: Image Processing - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings, Part III*. 2019, pp. 649–659. DOI: 10.1007/978-3-030-30508-6_51.
- Results and concepts of Chapter 6 are partly partly published in:
  - Nils Worzyk, Stefan Niewerth, and Oliver Kramer. "Adversarials $^{-1}$ in Speech Recognition: Detection and Defence". In: *28th European Symposium on Artificial Neural Networks, ESANN 2020, Bruges, Belgium, 2020 (in print)*. 2020.

2. Machine Learning and Neural Networks

In contrast to traditional programming where the developer defines the behaviour of a program explicitly, machine learning aims at providing a program, usually called model, which adapts to a specific behaviour based on given data. In general, the model can be seen as a function $f(\mathbf{x},\theta)$, which maps an I-dimensional input or feature vector $\mathbf{x}$ to an J-dimensional output or prediction vector $\mathbf{y}$, depending on the internal parameter vector θ of the model. Formally this can be written as:

$$f(\mathbf{x},\theta) : \mathbf{x} \in \mathbb{R}^I \rightarrow \mathbf{y} \in \mathbb{R}^J,\ I,J \in \mathbb{N}\ . \tag{2.1}$$

Within the broad field of machine learning, there are different dimensions used to specify different approaches. One dimension is the task the model should solve. In this thesis, the focus is on classification, which is introduced and defined in Section 2.1. Besides classification, *Regression* and *Reinforcement Learning* are two other general types of problems tackled with machine learning. For more information on those tasks, I refer to, e.g., [GBC16] or [RN16].

Another dimension is the accessibility of data during the learning process. In this thesis, supervised learning is assumed, which is explained in Section 2.2. Aside from supervised learning, there exists the scenario of unsupervised learning, where less knowledge about the training data is considered. For more information on unsupervised learning I refer to, e.g., [GBC16] or [RN16].

After introducing these basic concepts, in Section 2.3 ANNs are introduced, as they are the models used in this thesis to classify image or audio inputs. At the hand of an example the elements of ANNs, namely neurons (Section 2.3.1), common activation (Section 2.3.2) and loss functions (Section 2.3.3), and backpropagation (Section 2.3.4) are explained. In Section 2.4 a special form of ANNs is introduced, so called convolutional neural networks (CNNs) which are very successful in, e.g., image classification.

If not otherwise referenced, the depiction in the following Sections is oriented to the introductory books "Deep Learning" by Goodfellow, Bengio, and Courville [GBC16] and the widely known "Artificial Intelligence: A Modern Approach" by Russell and Norvig [RN16].

2.1. Classification

In this thesis classification is considered as main task, in particular image classification. For a given model f, the general goal is to assign the correct of C classes to an unknown input $\mathbf{x}$. In terms of image classification, the input is usually an image of size $h \times w \times z$, where h is the height of the image, w the width, and z the number of colour channels.

Common values for z are 1, if the image is in grey-scale, or $z = 3$ if the image consists of the three colour channels red, green, and blue. In addition, in image classification the pixel values are commonly scaled to the interval [0,1]. Therefore, the general machine learning Equation 2.1 can be reformulated to be

$$f(\mathbf{x},\theta) : \mathbf{x} \in [0,1]^{h \times w \times c} \rightarrow \{1,\ldots,C\} \tag{2.2}$$

A more comprised formulation is $\mathbf{y} = f(\mathbf{x},\theta)$, where $\mathbf{y}$ could be a single number representing the chosen class $\{1,\ldots,C\}$. However, in modern classification models, $\mathbf{y}$ is usually encoded as a vector $\mathbf{y} \in \mathbb{R}^C$ consisting of calculated values for each class. The class with the highest value is assumed to be the chosen class c. Depending on the format of the output, e.g., if the output values are in the interval [0,1], the scores are referred to as probabilities or confidence scores for each class.

As a running example, email spam classification as depicted in Figure 2.1a is considered. For the sake of simplicity, there are two types of words—good words which occur in regular mails, and bad words which occur mainly in spam mails. Furthermore, it is assumed that if the number of good words exceeds the number of bad words, the mail should be classified as regular (blue area above the bold line). Otherwise, the mail should be classified as spam (orange area below the bold line). Based on these assumptions, the ideal decision boundary of the truth f^* is given by the bold line.

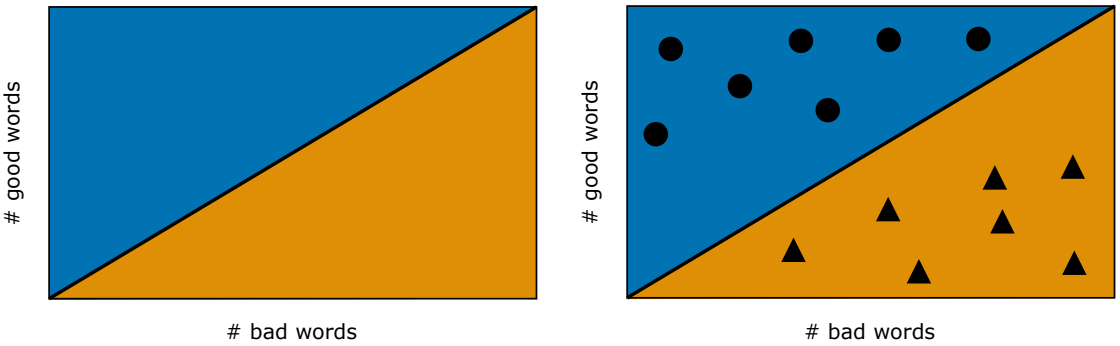
However, in real-world problems like image classification, f^* is not known and often the true decision boundary is too complex to be derived directly. Therefore, somehow the system has to *learn*, i.e., approximate f^* by adapting to given samples of the true function.

2.2. Supervised Learning

In supervised learning, f^* is considered as an oracle, which implements the ideal behaviour desired to approximate. However, the internal structure of an oracle is not accessible, but the oracle can be queried with different inputs and returns the corresponding true output $\mathbf{y}^*$. For example, in image classification, the human decision is usually assumed to be the ground truth. By querying the oracle, a dataset $\mathcal{X}$ of samples is created, each consisting of an input feature vector $\mathbf{x}$ and a corresponding correct output $\mathbf{y}^*$. The samples derived by querying the oracle are called ground truth since they are assumed to be correct in its prediction.

In the spam classification example, the feature vector consists of the number of good and bad words, and the corresponding output is either regular or spam. In Figure 2.1b samples are depicted, where circles represent regular inputs, and triangles depict spam inputs, based on the decision of the true model f^*.

In supervised learning these samples are used to train a model f, to predict the correct output for a given input as precisely as possible. However, if the model is only trained and evaluated on the whole dataset $\mathcal{X}$, in the worst case it simply memorises all inputs and their corresponding outputs, and is not able to abstract the classification to unknown inputs—the model over-fitted to the given data. But the capability to



(a) All emails within the blue area above the decision boundary (bold line) are classified as regular, while the orange area below is classified as spam.

(b) Samples of regular mails (circles) and spam mails (triangles), as classified by the underlying true model f^*.

Figure 2.1.: Schematic representation of the email classification problem. The bold line represents the underlying true decision boundary.

abstract or generalise the classification behaviour to previously unknown inputs is a very important feature of a good classification model. This is particularly important in real-world scenarios when dealing with dynamic environments. To achieve generalisation, and to counter memorisation, $\mathcal{X}$ is usually split into a training set $\mathcal{X}_{train}$ and a test set $\mathcal{X}_{test}$, where $\mathcal{X}_{train} \cup \mathcal{X}_{test} = \mathcal{X}$ and $\mathcal{X}_{train} \cap \mathcal{X}_{test} = \emptyset$. During training, the model only has access to the data of the training set to adapt to the given behaviour. After the model has been trained, the generalisation capability is evaluated on the test set by predicting previously unknown inputs. However, this could either only be done once, after completing the training, or it could be done during the training process to reasoning if the model already over-fitted to the training data or not. In the latter case, the test data is seen during the training process, which should be avoided. Therefore, the training data is further split into a smaller training and a disjunct validation set $\mathcal{X}_{val}$. The new training set is used to train the model, while the validation set is used to check the generalisation capability during the training process. After having the model trained to achieve a good classification accuracy and a good generalisation capability, the final model is tested on the completely unknown test set. The necessity to split the complete dataset $\mathcal{X}$ also shows that usually, it is necessary to acquire lots of samples to achieve a good performance as well as generalisation capability. For modern image classification tasks, the necessary amount of samples to achieve good generalisation is in the millions.

Aside from dataset splitting, there are several other techniques to further improve the generalisation capability of machine learning models, e.g., regularisation or cross-validation. The interested reader is therefore referred to the more complete books of Goodfellow et al. [GBC16] or Russell and Norvig [RN16].

2.3. Artificial Neural Networks

It has been shown that ANNs are very successful when it comes to complex classification tasks[1]. In this thesis mainly feed-forward models are used, for which an example is displayed in Figure 2.2. Formally, a feed-forward ANN is a directed acyclic graph consisting of several neurons (circles). The original structure of neurons was introduced in [MP87], where they were named Perceptron. To structure the graph, several neurons are summarised to layers.

The raw input $\mathbf{x}$, e.g., the pixel values of an image are fed to the input layer, which has as many neurons as the dimensionality of the input. The last layer of an ANN is called output layer since it returns the output or prediction. Therefore, the output layer has as many neurons as the output dimensionality. All layers between the in- and output layers are called hidden layers since they are hidden from direct access by a user.

The example in Figure 2.2 is furthermore called a fully-connected ANN because each neuron of each layer is connected with each neuron of the following layer. The connections between the different neurons represent the flow of the information through the model, and while passing through the model, the information gets transformed by weight matrices $\mathbf{W}_i$ and functions f_i which is explained in more detail in Section 2.3.1. Training an ANN means to adjust the weight matrices $\mathbf{W}_i$ to solve a specific task. The most common routine to adjust the weights—referred to as backpropagation—is explained in Section 2.3.4.

In addition to feed-forward fully-connected ANNs, there are other types of ANNs tailored to solve specific tasks. For time series classification, so-called recurrent neural networks (RNNs) are successful. The speciality of RNNs are that neurons of hidden layers not necessarily feed their output to neurons of the following layer, but instead recurrent connections of neurons to earlier layers or even the same layer are allowed. Thereby, the output of a neuron in one time-step can influence the calculation of an earlier neuron in a later time-step. In Section 6.1.2 an RNN is introduced in more detail in the context of speech classification.

Another specialised type of ANNs are CNNs. They perform better when some sort of locality is important, like in image classification. In Section 2.4 CNNs are explained in more detail.

2.3.1. Perceptron

The basic elements of earlier ANNs have been perceptrons. Although, over time they have been developed from the original version presented in [MP87]. In modern ANNs the perceptrons are usually referred to as neurons. In Figure 2.3 the structure of a single neuron is depicted and each neuron implements the general function

$$\hat{f}\left(\hat{\mathbf{x}},\hat{\theta}\right) : \hat{\mathbf{x}} \in \mathbb{R}^{\hat{I}} \rightarrow \hat{y} \in \mathbb{R}^1 \ . \tag{2.3}$$

[1] For a detailed historic overview of ANNs in general and successful applications, I refer to [Sch15]

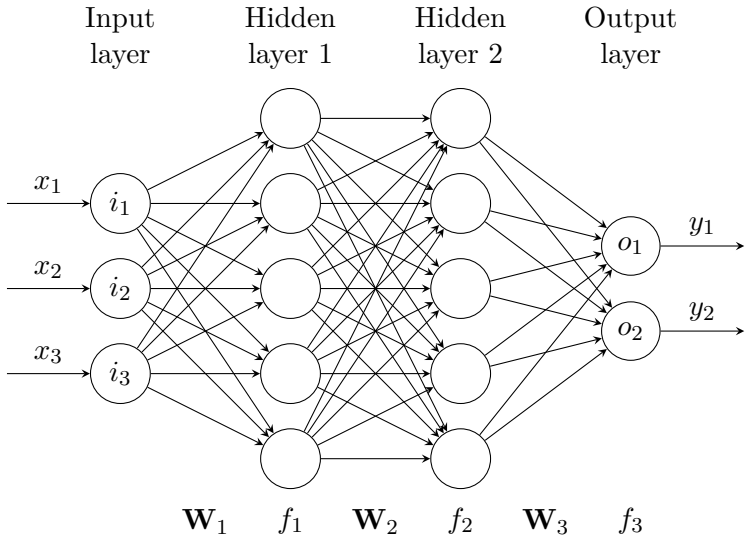


Figure 2.2.: Scheme of a fully-connected feed-forward ANN. Each layer is comprised of several neurons (circles). The complete model is comprised of one input layer, two hidden layers implementing the functions f_1 and f_2, and one output layer implementing function f_3. The weight matrices $\mathbf{W}_1$, $\mathbf{W}_2$, and $\mathbf{W}_3$ implement the relations between the different layers, described by weights and biases.

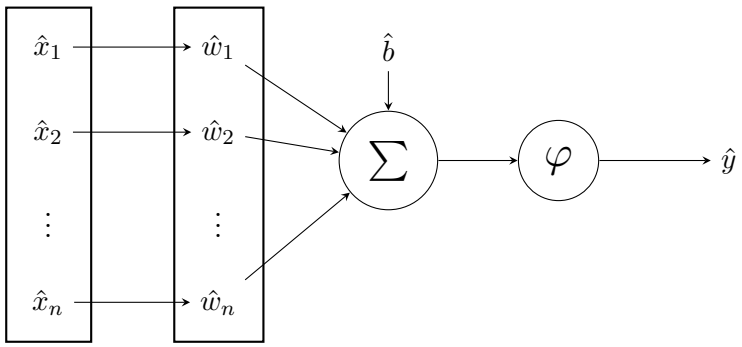


Figure 2.3.: Schematic figure of a neuron with the input features $\{\hat{x}_1, \ldots, \hat{x}_n\}$, their corresponding weights $\{\hat{w}_1, \ldots, \hat{x}_n\}$, the bias $\hat{b}$, the sum $\sum$, the activation function φ, and the output $\hat{y}$.

Most left is the input vector $\hat{\mathbf{x}}$, containing the outputs of the connected neurons from the previous layer. Each input $\hat{x}_i$ is multiplied by a weight factor $\hat{w}_i$, and the weighted inputs are summed up. Mathematically speaking, this is a vector multiplication to whose result the so-called bias $\hat{b}$ is added. The sum of the weighted inputs and the bias is fed into an activation function φ, which calculates the output $\hat{y}$ of the observed neuron. Summarised, the function each neuron implements can be formalised as

$$\hat{y} = \varphi\left(\sum_{i \in \hat{I}} (\hat{x}_i \cdot \hat{w}_i) + \hat{b}\right) \ . \tag{2.4}$$

Generalising to an ANN, the weights between two consecutive layers are summarised as weight matrix $\mathbf{W}$ (compare Figure 2.2), and the biases for each neuron of a given layer are summarised as the bias vector $\mathbf{b}$. Thereby, the output $\hat{\mathbf{y}}$ of a layer i in an ANN is calculated by

$$\hat{\mathbf{y}}_i = \hat{\mathbf{x}}_{i-1} \cdot \mathbf{W}_i + \mathbf{b}_i$$

where $\hat{\mathbf{x}}_{i-1}$ is the output of the previous layer as a row vector, and $\mathbf{b}_i$ are the biases of the observed layer as a column vector. Further more, all weight matrices and bias vectors combined are referred to as the parameter vector θ of the model f.

2.3.2. Activation Function

The sum of Equation 2.4 can only describe linear relationships, but real world problems are usually highly non-linear. Therefore, one desired property of the activation function φ is to be non-linear. One prominent non-linear function is the sigmoid function, defined as

$$\text{sigmoid}\,(\hat{x}) = \frac{1}{1 + e^{-\hat{x}}} = \frac{e^{\hat{x}}}{e^{x} + 1} \ . \tag{2.5}$$

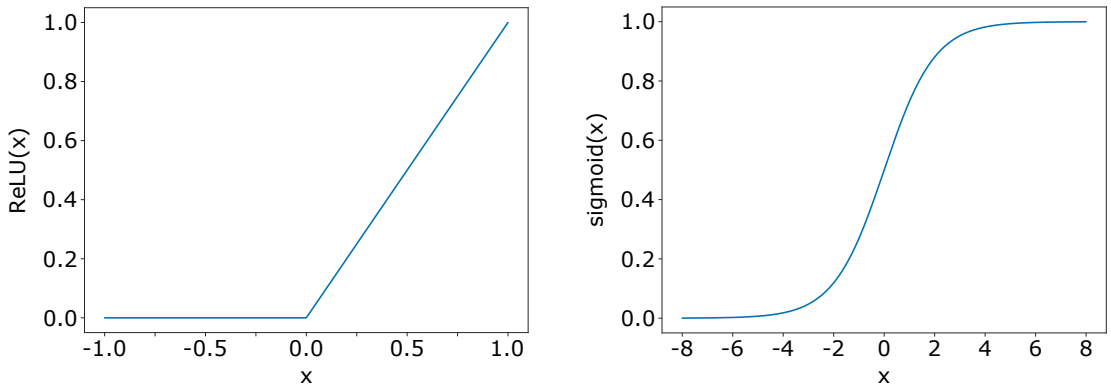
The output of sigmoid is a value between 0 and 1, as can be seen in Figure 2.4b, which could directly be interpreted as probability or confidence score. Hence, the sigmoid function is sometimes used as activation function in the output layer, to return the probability that the input belongs to a certain class. Another common activation function is rectified linear unit (ReLU) [GBB11] (see Figure 2.4a), which is defined as

$$\text{ReLU}\,(\hat{x}) = \max\,(0,\hat{x}) \ . \tag{2.6}$$

In particular, in current state of the art models, ReLU is preferred to other activation functions like sigmoid due to its fast computation, simplicity, and better results [GBB11]. One drawback of ReLU in comparison to sigmoid is that the output values are not bound to the interval [0,1]. Therefore, they can not directly be interpreted as probabilities. To solve this issue, if ReLU is used in the output layer the results are usually transformed at some point by the softmax function, defined as

$$\text{softmax}\,(\hat{\mathbf{x}})_i = \frac{e^{\hat{x}_i}}{\sum_{j=1}^{C} e^{\hat{x}_j}} = \hat{y}_i \ , \tag{2.7}$$

where $\hat{\mathbf{x}}$ is the output vector of the last layer, i is the class for which the softmax value should be calculated, and C is the total number of different classes. After application of softmax, the new values for $\hat{y}_i$ are in the interval [0,1] and $\sum_{i=1}^{C} \hat{y}_i = 1$, and therefore can be interpreted as probabilities.



(a) ReLU activation function.

(b) Sigmoid activation function.

Figure 2.4.: Plot of two common activation functions.

2.3.3. Performance Measure

To evaluate the performance of a model f, a quantitative measure has to be defined. Another name for the performance measure, especially during training is loss function because it calculates the difference—the loss—of the output of f to the ground truth.

In image classification, the desired output $\mathbf{y} \in [0,1]^C$ of a model is a value for each class, which can be interpreted as probability of the input belonging to the class, i.e., a probability distribution over all classes. To compare the predicted probability distribution, the ground truth $\mathbf{y}^*$ is also encoded as a probability distribution, where $y_i^* = 0 \; \forall i \neq c$ and $y_c^* = 1$, where c is the index assigned to the ground truth classification. This type of encoding is referred to as one-hot encoding. To compare the probability distribution calculated by f, and the one-hot encoded ground truth, a common loss function is the cross entropy loss, defined as

$$\mathcal{L}\left(\mathbf{y},\mathbf{y}^*\right) = -\sum_{i=1}^{C} \left[y_i^* \cdot \log\left(y_i\right) \right] . \tag{2.8}$$

Since $y_i^* = 0 \; \forall i \neq c$, and $y_c^* = 1$, where c is the true class, Equation 2.8 is shortened to be

$$\mathcal{L}(\mathbf{y},c) = -\log\left(y_c\right) \ , \tag{2.9}$$

which is referred to as negative log likelihood loss. Since usually softmax is applied before calculating the loss, all values in $\mathbf{y}$ are in the interval [0,1]. Thereby, if the predicted value for the true class is 0, i.e., the lowest value possible, the loss $\mathcal{L}$ is calculated to

be ∞. If the prediction matches the ground truth perfectly, i.e., prediction and ground truth are both 1 for the true class, $\mathcal{L}$ is evaluated to be 0. During the training process it is therefore the goal to minimize the loss function by adapting the parameters of the model accordingly (see Section 2.3.4).

However, the values of the loss function are difficult to interpret. Therefore, the quality of a fully trained model is usually reported by the accuracy or error rate. The accuracy (acc) calculates the proportion of correctly classified samples to the number of all samples N

$$\text{acc} = \frac{\sum_{i=1}^{N} \mathbb{1}\left[\arg\max_j \left(\mathbf{y}_{i,j}\right) = \arg\max_j \left(\mathbf{y^*}_{i,j}\right)\right]}{N} ,$$

where $j \in \{1, \ldots, C\}$, and $\arg\max_j$ is the class with the highest calculated value. The indicator function $\mathbb{1}$ evaluates to 1, if the given condition is true. Contrary, the error rate (err) is defined as the proportion of incorrect classified samples, i.e.,

$$err = 1 - acc = \frac{\sum_{i=1}^{N} \mathbb{1}\left[\arg\max_j \left(\mathbf{y}_{i,j}\right) \neq \arg\max_j \left(\mathbf{y^*}_{i,j}\right)\right]}{N} .$$

2.3.4. Backpropagation

The goal in supervised classification is to adjust the parameter vector θ of a model f to adapt to a specific classification behaviour, given by observed samples. Probably the most common routine to achieve this is backpropagation [RHW86]. To demonstrate the algorithm, the simple feed-forward fully-connected ANN depicted in Figure 2.5 is assumed to solve the spam classification problem introduced in Section 2.1. As a recap, the goal is to calculate the probability of an unknown mail being regular (y_1) or spam (y_2), based on the number of good words (x_1) and bad words (x_2). The neurons in the network have ReLU (Equation 2.6) as activation function, softmax (Equation 2.7) is applied to the outputs of the model, and the negative log-likelihood (Equation 2.9) is used as loss function to quantify the performance of the model during training.

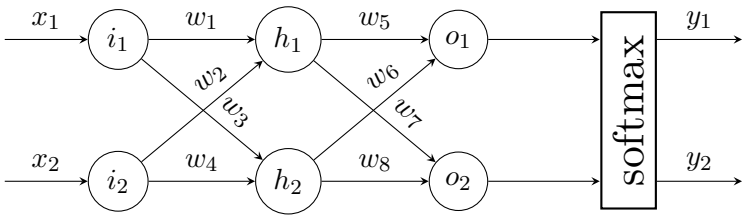


Figure 2.5.: Simple ANN with two inputs, two hidden neurons, and two output neurons to solve the spam classification problem. The result of the output neurons are fed to a softmax function, which returns the final two prediction values y_1 and y_2.

At the start of the process, the weights of the model, i.e., the parameter vector θ is initialised at random. For simplicity, the bias for each neuron is omitted in the example.

Different strategies to initialise θ can be used, where two common strategies are the 'Xavier' [GB10], or the 'He' [He+15] initialisation. Both strategies propose to use a normal distribution with a mean of 0, but different strategies to calculate the standard deviation. For the considered example, the weights are initialised by a normal distribution centred around 0 and a standard deviation of 1, given in Table 2.1.

Table 2.1.: Randomly initialised weights by a normal distribution with mean of 0, and a standard deviation of 1.

$w_1 = -0.4487$	$w_5 = \ \ 0.4378$
$w_2 = \ \ 0.8904$	$w_6 = \ \ 0.5844$
$w_3 = -1.8170$	$w_7 = -0.3223$
$w_4 = -1.2820$	$w_8 = \ \ 0.4010$

After initialisation, the first step of the backpropagation algorithm itself is to calculate the output for a given sample. This step is called forward pass. Assuming the number of good words is $x_1 = 80$ and the number of bad words to be $x_2 = 50$. The true prediction therefore should be 'regular mail', i.e., $y_1^* = 1$ and $y_2^* = 0$. For the given input and weights, the output of h_1 is

$$\begin{aligned} h_1 &= \mathrm{ReLU}(w_1 \cdot i_1 + w_2 \cdot i_2) \\ &= \mathrm{ReLU}((-0.4487 \cdot 80) + (0.8904 \cdot 50)) \\ &= \mathrm{ReLU}(-35.896 + 44.52) \\ &= \mathrm{ReLU}(8.624) \\ &= 8.624 \end{aligned}$$

Doing the same calculation for the other neurons, the outputs given in Table 2.2 are assumed.

Table 2.2.: Exemplary output vales of the internal neurons considering the weights of Table 2.1.

$h_1 = 8.624$	$o_1 \approx 3.7756$
$h_2 = 0$	$o_2 = 0$

Afterwards, the softmax function (Equation 2.7) is applied to the outputs:

$$\begin{aligned} y_1 &= \mathrm{softmax}(o_1) & y_2 &= \mathrm{softmax}(o_2) \\ &= \mathrm{softmax}(3.7756) & &= \mathrm{softmax}(0) \\ &= \frac{e^{3.7756}}{e^{3.7756} + e^0} & &= \frac{e^0}{e^{3.7756} + e^0} \\ &\approx 0.9776 & &\approx 0.0224 \end{aligned} \tag{2.10}$$

Based on these outputs, and considering that the true class should be $y_1 = 1$, the loss for the given sample is calculated by

$$\begin{aligned}\mathcal{L}(\mathbf{y},1) &= -\log(y_1) \\ &= -\log(0.9776) \\ &\approx 0.0227\end{aligned}$$

After the loss has been determined for a given sample, the second step of backpropagation is to calculate the partial influence of each weight on the overall loss. To find the partial influence of a weight to the overall loss, the partial derivation of the loss has to be calculated, concerning the single weights, formally

$$\frac{\partial \mathcal{L}(\mathbf{y},c)}{\partial w_i} \text{ for } i \in \{1,\ldots,8\} ,$$

where c is the true class. This also applies for the following equations.

However, $\mathcal{L}(\mathbf{y},c)$ is not directly dependent of the weights itself, but rather a chained calculation of outputs of earlier neurons. Thus, going backwards through the model and calculate the partial influence of the intermediate outputs is necessary, to calculate the influence of the corresponding weights. Therefore, this second step is called backwards pass. As for the example, the output $\mathbf{y}$ is dependent on the softmax calculation of o_1 and o_2. Formally, the chain rule is applied to get

$$\frac{\partial \mathcal{L}(\mathbf{y},c)}{\partial o_i} = \frac{\partial \mathcal{L}(\mathbf{y},c)}{\partial y_c} \cdot \frac{\partial y_c}{\partial o_i} , \tag{2.11}$$

where i ranges over the number of inputs for the observed derivation from the previous layer. Deriving the first term yields

$$\begin{aligned}\frac{\partial \mathcal{L}(\mathbf{y},c)}{\partial y_c} &= \frac{\partial(-log(y_c))}{\partial y_c} \\ &= -\frac{1}{y_c}\end{aligned} \tag{2.12}$$

Unfolding the second term on the right side of Equation 2.11 yield

$$\frac{\partial y_c}{\partial o_i} = \frac{\partial}{\partial o_i} \cdot \left(\frac{e^{o_i}}{\sum_{j=1}^{C} e^{o_j}} \right) ,$$

and applying the quotient rule grants

$$\begin{aligned}\frac{\partial y_c}{\partial o_i} = \frac{\partial}{\partial o_i} \cdot \left(\frac{e^{o_i}}{\sum_{j=1}^{C} e^{o_j}} \right) &= \frac{\left(\frac{\partial}{\partial o_i} e^{o_i}\right) \cdot \left(\sum_{j=1}^{C} e^{o_j}\right) - (e^{o_i}) \cdot \left(\frac{\partial}{\partial o_c} \sum_{j=1}^{C} e^{o_j}\right)}{\left(\sum_{j=1}^{C} e^{o_j}\right)^2} \\ &= \frac{(e^{o_i}) \cdot \left(\sum_{j=1}^{C} e^{o_j}\right) - (e^{o_i}) \cdot (e^{o_i})}{\left(\sum_{j=1}^{C} e^{o_j}\right)^2} = \frac{e^{o_c}}{\sum_{j=1}^{C} e^{o_j}} \cdot \frac{\left(\sum_{j=1}^{C} e^{o_j}\right) - e^{o_c}}{\sum_{j=1}^{C} e^{o_j}} \\ &= y_c \cdot (1 - y_c)\end{aligned} \tag{2.13}$$

Inserting Equations 2.12 and 2.13 into Equation 2.11 leaves us with

$$\begin{aligned}\frac{\partial \mathcal{L}\left(\mathbf{y},c\right)}{\partial o_i} &= \frac{\partial \mathcal{L}\left(\mathbf{y},c\right)}{\partial y_c} \cdot \frac{\partial y_c}{\partial o_c} \\ &= -\frac{1}{y_c} \cdot \left(y_c \cdot \left(1 - y_c\right)\right) = -\frac{y_c - y_c^2}{y_c} \\ &= y_c - 1\end{aligned}$$

Inserting the example output values given in Table 2.2 into Equation 2.10 with respect to the desired classification, yields

$$\frac{\partial \mathcal{L}(\mathbf{y},1)}{\partial o_1} = y_1 - 1 = 0.9776 - 1 = -0.0224 \tag{2.14}$$

In general, the derivative of a function indicates the slope of the function, which in the given example means that an increase of o_1 leads to a decrease of $\mathcal{L}\left(\mathbf{y},1\right)$.

Following that strategy and proceeding further back through the model, to calculate the derivatives of the hidden layer outputs, it is necessary to know that the derivative of ReLU is

$$\frac{\partial}{\partial \hat{x}} \mathrm{ReLU}\left(\hat{x}\right) = \begin{cases} 0 \text{ if } \hat{x} < 0 \\ 1 \text{ if } \hat{x} > 0 \end{cases} , \tag{2.15}$$

where the derivative at position $\hat{x} = 0$ is undefined but solved by randomly choosing 0 or 1 as solution. Thereby, the influence of w_5 on the output of o_1 is calculated by

$$\begin{aligned}\frac{\partial o_1}{\partial w_5} &= \frac{\partial(\mathrm{ReLU}\left(o_1\right))}{\partial o_1} \cdot \frac{\partial o_1}{\partial w_5} = \frac{\partial(\mathrm{ReLU}\left(h_1 \cdot w_5 + h_2 \cdot w_6\right))}{\partial\left(h_1 \cdot w_5 + h_2 \cdot w_6\right)} \cdot \frac{\partial\left(h_1 \cdot w_5 + h_2 \cdot w_6\right)}{\partial w_5} \\ &= \frac{\partial(\mathrm{ReLU}\left(h_1 \cdot w_5 + h_2 \cdot w_6\right)}{\partial\left(h_1 \cdot w_5 + h_2 \cdot w_6\right)} \cdot h_1 \Big|_{h_1=8.624, w_5=0.4378, h_2=0, w_6=0.5844} \\ &= 1 \cdot 8.625 \\ &= 8.625\end{aligned}$$

Hence, knowing that increasing the value of w_5 increases the value of o_1 as well, and recalling Equation 2.14, increasing o_1 leads to a decrease in the overall loss. The same process can be done for all parameters in θ of the model. The last step of one backpropagation cycle is to adapt all parameters in θ of the model to minimise the loss function. This process is generally called gradient descent and can be formalised as

$$\theta_i = \theta_{i-1} - \eta \nabla \mathcal{L}\left(\mathbf{x},\mathbf{y}^*,\theta_{i-1}\right) ,$$

where η is called learning rate and controls how much the parameters are changed, and $\nabla \mathcal{L}\left(\mathbf{x},\mathbf{y}^*,\theta_{i-1}\right)$ is the gradient, i.e., the tangential vector pointing in the direction of the steepest ascent, when evaluating $\mathcal{L}$ regarding a certain in- and output combination, with the current parameter vector θ_{i-1}.

After all training samples have been processed once, the model has been trained for one epoch. As in the example, the dataset can be processed sample by sample, and after each

input, the parameters are updated. This approach is called stochastic gradient descent (SGD), however, in real-world scenarios when hundreds, thousands, or even more training samples exist, processing one input at a time has several drawbacks. One epoch can take very long, and the parameter updates can be too noisy which reduces the stability and generalisation capability of the learning process. In contrast, processing the whole training dataset at once, i.e., calculate the gradient-based on all samples and update the parameters only once, is called gradient descent. The problem of this approach is that all samples together can be rather large and therefore difficult to process due to memory limitations. Another disadvantage is that θ is only updated once, and hence the training set has to be processed for many epochs to find a satisfying solution. The most common solution is to use batch-wise gradient descent, i.e., divide the whole training set into random disjunct subsets called batches of a given size, which are used to calculate the parameter update. After each epoch, the batches are drawn again at random.

By adjusting the parameter vector θ iteratively, the model learns to adapt to the given samples. In Figure 2.6 the spam classification example is extended accordingly. The circles are the given samples of regular mails, the triangles depict samples of spam emails, and the bold line is the decision boundary of the underlying truth f^*. The dotted line symbolises the decision boundary for a model f, which was trained on the given samples to separate them with the least possible loss. Ideally, the dotted line fits the solid line, but this is hardly possible depending on the complexity of the problem to solve and the given samples.

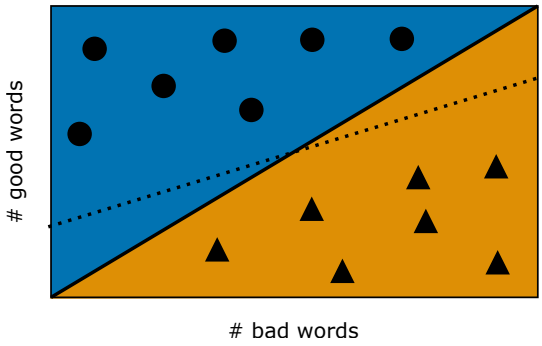


Figure 2.6.: Example of a decision boundary of a model f (dotted line), trained to separate the regular mails (circles) from the spam mails (triangles) with the least possible loss. In comparison the underlying true decision boundary (bold line) by the ground truth f^*.

2.4. Convolutional Neural Networks

Usually, the first groundbreaking success in image classification is attributed to Krizhevsky, Sutskever, and Hinton [KSH12], when they reported a top-5 test error rate[2] of 15.3% in the

[2]The top-5 error rate calculates the percentage of samples, for which the true class is not among the five classes with the highest confidence scores.

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 competition [Rus+15]. One novelty of the proposed architecture was that it used convolutions which led to the name CNN of architectures which leverage such convolutions. Where fully-connected ANNs consider all outputs of the previous layer to calculate the output of one neuron, CNNs make use of so-called filters or kernels to consider only parts of the previous output to calculate their output on. This behaviour is depicted in Figure 2.7.

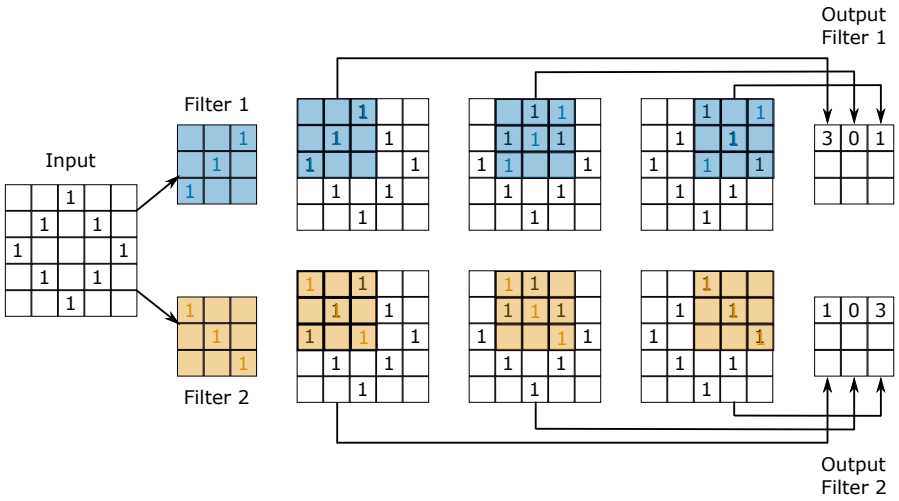


Figure 2.7.: Example of a convolutional layer with two filters. Each filter is shifted over the full input (depicted for three steps) but calculates only one output per receptive field. In this example, the output is calculated as sum over the elementwise product of the filter element and the corresponding input element. Empty elements in the input and filter are considered to have the value 0.

On the most left, the input image is displaying a diamond of 1's. All other values of the input are 0's and therefore omitted. One important task in image processing is edge detection to find the shape of certain objects. Therefore, two filters are considered, where Filter 1 is detecting upward right diagonals, while Filter 2 is detecting downward right diagonals. Both filters are shifted over the whole input (depicted for three steps) with a certain step size called stride while observing the underlying area of the input, called receptive field. The operation $*$ to calculate the output is called convolution and is in the two-dimensional discrete case defined as

$$\text{out}\,(m,n) = \omega * \text{in}\,(x,y) = \sum_{s=0}^{w_k-1} \sum_{t=0}^{h_k-1} \omega(s,t) \cdot i\,(x+s, y+t)$$

where $\text{out}\,(m,n)$ is the output for the position (m,n), ω is the filter matrix, $\text{in}\,(x,y)$ is the input at position (x,y), and w_k, h_k are the width, resp. the height of the filter matrix. In the given example that would be 3 for both, width and height. Other common filter sizes are 1×1 or 5×5, while in general the filter size depends on the problem to solve.

By convolving, the spatial resolution of the input is reduced. In the example, the input is 5×5 pixel large, while the output for each filter is 3×3 pixels. If a reduction in spacial resolution is not wanted, the input can be padded. There are different strategies for padding (see, e.g., Tang et al. [TOB19]), but one of the most common techniques is to add 0's around the input to extend the borders and thereby prohibit the spacial reduction of the input[3].

In Figure 2.7, convolving Filter 1 and the upper left area of the input leads to an output of 3. However, when shifted to the upper right corner of the input, the output value sums up to 1, because Filter 1 and the input values do not overlap well. In contrast, convolving Filter 2 with areas of the input showing downward right edges, i.e., the upper right and lower left area, the results are relatively large. By shifting both filters over the whole input, the output resembles the matching of the filters for the observed receptive fields.

In general, the values within the filters are considered as weights during the training process, and therefore adapt to the specific task. In image processing, it has been shown [Yos+15] that the filters in the first convolutional layer learn edge like structures, as well as colour gradients. The number of filters per convolutional layer is depending on the problem to solve, and architecture of the model. Commonly the number of filters increases from the first to the last layer of a CNN, while the spatial resolution of the output is reduced.

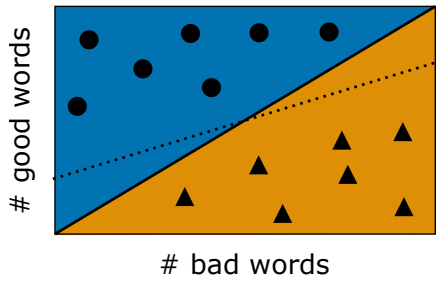
[3]For a more sophisticated review on convolution arithmetic, I like to refer the reader to the paper „A guide to convolution arithmetic for deep learning“ by Dumoulin and Visin [DV16].

Part I.

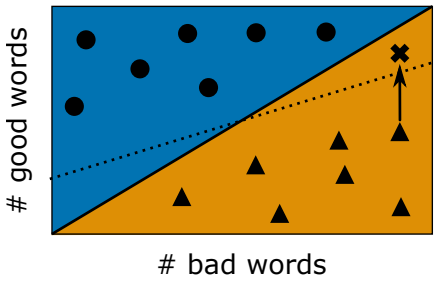
Adversarial Inputs

3. Foundations of Adversarial Inputs

In Chapter 2 it was explained how ANNs learn to solve specific tasks, taking into account the example of spam classification for which the final result of the training process is recapped in Figure 3.1a. The circles and triangles represent the samples used for training. Based on these samples a model f is trained, which implements the decision boundary given by the dotted line. In comparison, the bold line represents the true decision boundary given by the truth f^*. For both decision boundaries, samples above the line are classified as regular emails, while samples below the line are classified as spam.



(a) Example of a trained model f to classify regular (circle) and spam (triangle) mails.



(b) By increasing the number of good words (arrow), a spam mail is converted into a regular one, when classified by f. Classified by f^*, the input would still be identified as spam. The newly created input (cross) is called adversarial input.

Figure 3.1.: Adversarial inputs as a consequence of the different decision boundaries of the true and a trained model. The blue area above the true decision boundary of f^* (bold line) indicates the decision space for regular mails, where the circles are observed samples. The orange area below the bold line contains spam mails, with samples depicted as triangles. Based on the samples, a model f is trained whose decision boundary (dotted line) does not exactly fit the true decision boundary.

As shown in Figure 3.1a, the decision boundary of the trained model (dotted line) and the true model (bold line) do not align. Thereof, areas in the feature space appear which are classified differently by f and the underlying truth f^*. Those areas become dangerous if adversaries manipulate existing samples to specifically fool a trained model, while the same input would still be classified correctly by the underlying truth f^*. This is exemplarily shown in Figure 3.1b, where the number of good words for a given spam sample is artificially increased (arrow). Due to the manipulation, the new input (cross)

is classified as regular mail by f, as it lies above the dotted line indicating the decision boundary. However, according to the underlying true decision boundary, the input should still be classified as spam. This process can also be applied to regular emails, to falsely be classified as spam. In general, inputs which are manipulated to be classified differently by the trained model f in comparison to the underlying truth, are called adversarial inputs. This term was used first by David et al. [Dal+04] to describe manipulated inputs in the context of spam classification.

While the spam classification problem seems trivial, with an increasing number of dimensions the needed perturbation for each of the dimensions to fool a model becomes less and less. This is due to a general problem in machine learning, referred to as the curse of dimensionality. First mentioned by Richard E. Bellman [Bel15] the problem is that with the number of observed dimensions, the volume of the space described by these dimensions increases exponentially. Consequently, the necessary number of samples increases exponentially, when a high dimensional space should be sampled with the same density as a low dimensional space. Exemplary consider the distance one meter, which can be sampled by 100 evenly distributed centimetres. If the dimensionality is increased by one, and hence the feature space is increased to be one square-meter, already $100 \times 100 = 10{,}000$ samples are needed, to have the same equal narrow distribution of the samples as for the one-dimensional meter. Considering this observation, the median distance d for a number of samples N on a p-dimensional unit ball towards the centre is calculated as [HTF13]

$$d\,(p,N) = \left(1 - \frac{1}{2}^{\frac{1}{N}}\right)^{\frac{1}{p}} \quad . \tag{3.1}$$

In image classification a common dataset is MNIST, which has $N = 6000$ samples per class. Each image is 28×28 pixels large, and each pixel has a value in the set $\{0,1,\ldots,255\}$. This leads to an overall dimensionality of $28 \cdot 28 \cdot 256 = 200{,}704$, and inserted into Equation 3.1 a median distance of ≈ 0.99995 is calculated, i.e., the samples are very near the edge of the unit ball. If the boundary of the unit ball is considered to be the decision boundary of a trained model, the samples have only to be slightly perturbed to cross the decision boundary. The assumptions made here are simplified to demonstrate the problem, however, Shafahi et al. [Sha+19] outline a more sophisticated proof that in particular for high dimensional inputs like modern images, adversarial inputs are inevitable.

Considering the dimensionality of modern images, the difference between an original and adversarial input is virtually invisible to humans, while these slight changes change the classification of the targeted model. An example of such an adversarial input was shown in Figure 1.1 in the introduction. The original image would be classified as ‘panda’ from both a human and the assumed image classifier. However, the adversarial input would still be classified as ‘panda’, whereas the model classifies the input as ‘chow-chow’, a certain dog breed.

3.1. Definition of Adversarial Inputs

In general, the problem statement of adversarial inputs is to find a manipulation $\boldsymbol{\delta}$, which is minimal regarding a certain distance function $\mathcal{D}$, such that the perturbed input $\mathbf{x} + \boldsymbol{\delta} = \mathbf{x}'$ is classified different to the true class c. In addition, the manipulated input should still be in the accepted input dimension I and value range of the model f to attack. Considering image classification, the inputs are usually scaled to the interval [0,1]. Formally written

$$\begin{aligned} \text{minimise} \quad & \mathcal{D}(\mathbf{x}, \mathbf{x} + \boldsymbol{\delta}) \\ \text{such that} \quad & f(\mathbf{x} + \boldsymbol{\delta}) \neq c \\ & \mathbf{x} + \boldsymbol{\delta} \in [0,1]^I \quad . \end{aligned} \tag{3.2}$$

In Section 3.2, common measures to quantify the distance between two inputs are introduced. Afterwards, an overview of the taxonomy of adversarial attacks is given in Section 3.3, followed by the taxonomy of adversarial defences in Section 3.4. A more thorough overview on adversarial inputs, attacks, and defences, is provided in different survey papers, e.g., of Serban et al. [SP18], Papernot et al. [Pap+16], Huang et al. [Hua+11], Barreno et al. [Bar+06], Chakraborty et al. [Cha+18], or Yuan et al. [Yua+19]. In Section 3.5 one of the first proposed, and very famous adversarial attacks concerning neural networks, the FGSM, is introduced. The ideas implemented in FGSM are also used in various, and more recently developed adversarial attacks. In Section 3.6 four additional attacks to FGSM are explained, which are used in the later experiments. The last Section 3.7 introduces a framework, which has been proposed to transfer adversarial attacks from the digital world to the physical, real world.

3.2. Distance Measures

As mentioned in Equation 3.2, the perturbation $\boldsymbol{\delta}$ should be minimised according to a certain distance metric $\mathcal{D}$. A common way to quantify the size of an arbitrary vector $\mathbf{x}$, in this case the size of the perturbation, is to calculate a L_p-norm, formalised as

$$\|\mathbf{x}\|_p = \left(\sum_i |x_i|^p \right)^{\frac{1}{p}} \quad , \tag{3.3}$$

where i is the index over all elements of the input vector, and $0 < p \leq \infty$. In the domain of image classification, mainly three different norms are used to calculate the perturbation $\boldsymbol{\delta}$, i.e., the distance between the original input $\mathbf{x}$ and the manipulated input $\mathbf{x}'$.

L_0 is mathematically not a norm and can not be calculated by Equation 3.3. If an element $x_i = 0$, 0^0 is undefined, as well as $\frac{1}{0}$ in the exponent. Therefore, the L_0-"norm" is defined as $\sum_i \mathbb{1}(\delta_i \neq 0)$, where δ_i is an element of the perturbation vector $\boldsymbol{\delta}$. It counts the number of altered inputs. However, this "norm" does not take into account the degree of change of the inputs. In this thesis, when writing L_0-norm, it is referred to the L_0-"norm".

L_2 also called Euclidean norm or distance, is defined as $\|\mathbf{x}\|_2 = \sqrt{\sum_i |x_i|^2}$. Since small changes, especially in the interval $(-1,1)$, contribute less to the overall distance than larger changes, the L_2-norm may be small, even if many input features in $\mathbf{x}$ are altered. With the increase of input dimensionality this becomes more and more problematic.

L_∞ also called maximum norm is defined as $\|\mathbf{x}\|_\infty = \max_i (|x_i|)$. Based on the calculation, only the largest manipulation subject to a single feature contributes to the overall distance. Therefore, if all other input features are distorted a bit less than that maximum, L_∞, in comparison for example to the L_2-norm, would not increase.

Another distance used in Chapter 7 is the Mahalanobis distance $\mathcal{D}_M$ [Mah36] defined as

$$\mathcal{D}_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})} ,$$

where $\boldsymbol{\mu}$ is the mean vector and Σ^{-1} the inverted covariance matrix of a given distribution. Thereby, the distance of one single sample $\mathbf{x}$ to a observed distribution is quantified.

3.3. Taxonomy of Adversarial Attacks

The taxonomy in the field of adversarial attacks is not uniform [Pap+16; SP18; Hua+11; Bar+06; Cha+18; Yua+19]. In this thesis three categories are used to differentiate between adversarial attacks:

1. The objective of the attack (Section 3.3.1)
2. The capability of an attacker during an attack (Section 3.3.2)
3. The time at which an attack takes place (Section 3.3.3)

3.3.1. Objective of an Attack

While the general goal of an adversarial attack is to fool a given target model, the objective in detail can be different and can change the decision which attack is used. On a higher level, Serban et al. [SP18] differ between recognisable and unrecognisable inputs. Recognisable adversarial inputs display something, which is recognised by humans as a specific object, e.g., the 'panda' in the introduction (see Figure 1.1c). Contrary, unrecognisable inputs display something seen as random noise or pattern by humans, but which is classified as an object by a given model. An attack creating such unrecognisable inputs was proposed, e.g., by Nguyen et al. [NYC15]. Another dimension to distinguish adversarial inputs is whether they are generated from scratch, usually starting based on random noise, or if they are created by modifying already existing inputs. Based on the creation method, the adversarial inputs can meet different objectives. Following Serban et al. [SP18], three different objectives are defined.

1. Confidence reduction: An attacker aims to induce class ambiguity by reducing the confidence score of the original correct class. As an example, a classifier is considered which only accepts an input if the confidence score of the highest class is above 90%. Hence, the classifier could be interfered, if an input is manipulated such that the highest confidence is below 90%. This could sometimes already be achieved by applying random noise to the input and not necessarily requires sophisticated attacks.

2. Misclassification or untargeted attack: In this scenario, the objective for an attacker is to create an input, which is classified by a given model, however, the specific classification is not relevant. Usually, untargeted attacks consider a source input, which is already classified by the model and alter the original input to be classified as something else. As an example consider a stop sign and the adversary would like to fool autonomous cars to not stop. To achieve this goal, it is not necessary to alter the original input to be specifically classified, e.g., as a priority road sign, as long as the car does not stop.

3. Targeted misclassification: Here the objective is to produce an input, which is classified as a specific class. Usually, it is assumed, that the attacker manipulates an existing correctly classified input, to be classified as a specific target class t by the target model. For example, an existing stop sign is assumed which is classified correctly as such. An adversary aims to manipulate the existing stop sign to be classified specifically as a priority road sign.

3.3.2. Capability of the Attacker

Another dimension to classify adversarial attacks is the knowledge an attacker has about the model to attack. In this thesis, the name convention of Serban et al. [SP18] is used.

White-box attacks consider full knowledge about the system to attack, i.e., the data used for training, the full network architecture including applied defences, the parameters θ of the network, and in particular the gradients of the model. Because of the detailed knowledge, white-box attacks are considered to be the strongest, and also fastest attacks.

Grey-box attacks summarise several different scenarios, in which the attacker only has partial knowledge about the system to attack. For example Papernot et al. [Pap+16] differentiate between

- knowledge about the architecture of the system to attack, but no knowledge about the training data, and
- knowledge about the training data, but no knowledge about the architecture.

To attack these type of grey-box scenarios, Goodfellow et al. [GSS15] reported that adversarial inputs are transferable between different models trained on the same data. This observation can be leveraged if the attacker does not know the model, but knows

the training dataset. In that case, the attacker can train its own model on the given data, create adversarial inputs by using white-box attacks and transfer the found adversarial input to the model to attack. Some recent approaches following this strategy are from Liu et al. [Liu+17], Wu et al. [Wu+18], Xie et al. [Xie+19b], and Dong et al. [Don+19].

In the later experiments, mainly a grey-box scenario is considered, where the attack has full access to the image or audio classifier, i.e., can apply white-box attacks to the classification model. However, no knowledge about the applied defences is available.

Black-box attacks assume that the attacker has neither knowledge about the internal structure nor knowledge about the training data. However, an attacker can usually query the unknown model with certain inputs and gets an output returned which complies the internal structure.

Strengthening the scenario to be black-box, the attacker could query the model to attack, to create a synthetic dataset resembling its input-output behaviour. Based on the synthetic dataset, the attacker trains a substitute model simulating the target's model behaviour. Having full knowledge about the substitute model, the attacker could apply white-box attacks to create adversarial inputs, which finally could be transferred to the target model. This general strategy is investigated by Papernot et al. [Pap+17] or more recently by Pengcheng et al. [LYZ18]. Depending on the allowed maximal perturbation, the success-rate, i.e., misclassification of transferred adversarial inputs, ranges from around 5% for very low perturbation levels, to around 90% for higher perturbation levels, on the datasets they investigated.

Another approach to tackle black-box scenarios is to employ derivative-free optimisation techniques like natural evolutionary strategy [Ily+18] or other evolution-based methods. One measure to quantify the effectiveness of black-box attacks is the number of queries towards the target model, necessary to create an input fooling the classification. In one recent publication by Guo et al. [Guo+19], the authors, for example, attack the Google Cloud Vision API, which returns a list of labels with corresponding confidence scores for arbitrary inputs. Neither the architecture, nor the data used for training are known, but Guo et al. report that 70% of their attacks are successful within a limit of 5,000 queries.

The frequency, attacks are applied towards the system to attack, is another way to describe the capability of an attacker, as Yuan et al. [Yua+19] point out.

Single-shot attacks manipulate the input only one time. The advantage is that the attack is fast, i.e., the attacker takes an original input, manipulates it by a predefined amount of perturbation, and feeds the manipulated input into the model to attack. However, if an attacker is only capable of perturbing an input once, the perturbation might be too small to fool the neural network and the attack would not be successful. Or, the perturbation could be too large and therefore do not fulfil the minimal perturbation criterion formalised in Equation 3.2. For the latter case, Serban et al. [SP18] note that the necessity of the minimal perturbation criterion is debatable. They argue that in a

real-world scenario, e.g., in autonomous driving, no longer humans are involved in the process of image classification, and therefore the perturbation not necessarily have to be unnoticeable for humans.

Aside from creating specific perturbations for each input, it is possible to create so-called universal adversarial perturbations [Moo+17; Liu+19a]. Those have the property to be transferable between inputs, i.e., the same perturbation pattern applied to different inputs fool the system to attack into misclassification.

Iterative attacks manipulate the input iteratively, starting with small perturbations which are increased, until the manipulated input is classified as desired. In contrast to single-shot attacks, iterative attacks take longer, but as an advantage, they can achieve the minimal or at least more optimal perturbations. This requirement becomes more important when the perturbation has to be invisible to humans, e.g., when a human is involved in the classification/verification process, like in border control. Besides, the optimal perturbation to change the classification as desired, in particular for targeted attacks, is dependent on the specific input. Some transitions require less perturbation as others, and thereby using a predefined amount of manipulation can enhance the problems of single-shot attacks.

3.3.3. Time of the Attack

The whole process of employing a machine learning classifier can be divided into two phases. At first, the model is trained, and after the training phase, it is deployed and can only process new given data. The second phase is called the inference phase. During both phases, attacks can take place.

Poisoning attacks take place during the training phase, by poisoning the training data to achieve a certain objective. For example, the attacker would like to achieve that street signs are classified wrong when a sticker is attached to them. Therefore, he could manipulate the training data in a way that unmodified street signs are still classified correctly, but street signs with a sticker attached are classified wrong. This is proposed, e.g., by Gu et al. [Gu+19]. Other attacks considering this scenario are investigated in, e.g., [Che+17; WC18]. However, poisoning attacks are not considered in later experiments.

Evasion attacks take place during inference, i.e., after the model was fully trained. Here, it is the objective to create inputs which are wrongly classified, based on a given model. Consider an autonomous car whose systems are already trained. The objective is to create certain inputs to fool the car, e.g., to classify a stop sign as a priority sign, based on the already trained systems.

3.4. Taxonomy of Adversarial Defences

Adversarial defences are usually divided into proactive and reactive approaches. Serban et al. [SP18] mention a third category—obfuscation defences—but mention themselves that those techniques heavily overlap with other proactive approaches. Therefore, only the terms proactive and reactive are used in this thesis.

Proactive defences aim to make the model itself more robust against adversarial inputs. This is done during the training phase, such that in inference no additional computations have to be executed.

Reactive defences aim to identify adversarial inputs during inference and process them accordingly afterwards. This can either be done by rejecting adversarial inputs at all or trying to restore the original true class.

A clear differentiation between several proposed techniques is, however, not possible because they often implement proactive and reactive approaches within one technique. In the following sections, some examples of recent defence techniques are explained.

3.4.1. Binary Classification

In addition to an initial model f_1 trained as a usual image classifier, Gong et al. [GWK17] propose to train an additional binary model f_2, based on original and adversarial samples, to distinguish between them based on the raw pixel values. On first-round attacks, i.e., inputs which are adversarial regarding f_1, around 100% of the adversarial inputs are detected as such by f_2. However, the authors find that in case of a second-round attack, i.e., inputs which are created to be adversarial to f_2 as well, the binary classifier has a very high false-negative rate—basically, all inputs are detected as adversarial. In real-world problems the implications of such behaviour are debatable. The authors state that they don't see a problem because the goal is to block adversarial inputs. On the other hand, if more or less all inputs are identified as adversarial and have to be processed manually, there is no need for a machine learning classifier in the first place. Also, they face another drawback, in that the binary classifier is sensitive to the amount of perturbation of the adversarial images. If the binary classifier is trained on adversarial images with a fixed amount of perturbation, but an attacker creates adversarial images with less perturbation, the accuracy of the binary classifier reduces.

3.4.2. Model Regularisation

A common technique to make models generally more robust is to apply random noise to the training samples, i.e., the features during the training process [SD91]. Traditionally, noise is only added to the input features, but more recently, e.g., He et al. [HRF19] propose to inject noise to the parameters of a model as well as the outputs of the hidden layers, to increase the robustness against adversarial attacks. Besides, they propose to

control the magnitude of the applied noise by a parameter, which itself is adapted during training.

3.4.3. Adversarial Training

Instead of including randomly perturbed inputs into the training process, Szegedy et al. [Sze+14] propose to include adversarial inputs into the training process as a mean of regularisation. This process is referred to as adversarial training or retraining. In addition to stronger robustness against adversarial inputs, Szegedy et al. [Sze+14] also report an increase in the overall generalisation capability of their model. However, due to the limited memory BFGS (L-BFGS) [LN89] attack they use, the overall process is computational very expensive and therefore not practical.

In [GSS15], Goodfellow et al. introduce a faster attack method (Section 3.5), and based on their FGSM, they extend the used training loss function to incorporate an adversarial objective function. Testing their model, they report a better classification accuracy on clean data, while additionally reducing the adversarial error.

However, one major drawback of adversarial training is that the perturbation level of the adversarial inputs during training has to be predefined. Therefore, this technique is not robust against iterative adversarial attacks, as, e.g., Kurakin et al. [KGB17b] show.

In a more recent approach, Madry et al. [Mad+18] propose to harden a model against adversarial inputs by "replacing the input points by their corresponding adversarial perturbations and normally training the network on the perturbed input" [Mad+18]. Thereby, they report promising results, but at the same time show that the approach is still prone to iterative attacks.

3.4.4. Data Preprocessing

Aside regularising the model, data preprocessing is another approach to defend against adversarial inputs. Buckman et al. [Buc+18], for example, motivate to use input transformations, to "break the linear extrapolation behaviour of machine learning models by preprocessing the input with an extremely non-linear function. This function must still permit the machine learning model to function successfully on naturally occurring inputs" [Buc+18]. Therefore, they propose to round the real-valued inputs of the interval [0,1] into bins, e.g., values in the interval [0,0. 1) to be in bin 0, values in [0. 1,0. 2) are in bin 1, and so forth. These bins are encoded as binary vectors, either in one-hot or in thermometer encoding. In Table 3.1 some examples for these two types of encoding are given.

Shaham et al. [Sha+18] also experiment with input transformations, namely "low-pass filtering, PCA, JPEG compression, low resolution wavelet approximation, and soft-thresholding" [Sha+18]. They show that JPEG compression and soft-thresholding achieve better accuracies, compared to an unprotected model when attacked. However, depending on the attack, the defensive effect is small and only occurs with stronger perturbation levels. For lower perturbation levels, all input transformations even reduce the classification accuracy of the baseline model. More recently, Liu et al. [Liu+19b]

Table 3.1.: Example mapping of real-valued inputs to the corresponding one-hot-encoding, resp. thermometer-encoding, as introduced by Buckman et al. [Buc+18].

Real-valued	One-hot	Thermometer
0.13	`[0100000000]`	`[0111111111]`
0.66	`[0000001000]`	`[0000001111]`
0.92	`[0000000001]`	`[0000000001]`

propose a revised version of JPEG compression to a) maximise the filtering of malicious features in adversarial inputs, and b) minimise the false-classification of benign inputs. Based on their enhanced compression technique, they report a small degradation of $\leq 1\%$ in accuracy on benign inputs, while improving the classification accuracy of adversarial inputs from $\sim 20\%$ to $\sim 90\%$.

3.4.5. Building Robust Model Architectures

Another approach to defending against adversarial inputs is by modifying the architecture of the model. Meng et al. [MC17] for example propose to use a specific form of ANNs, so-called auto encoders (AEs) [HZ93]. The structure of an AE can be divided into two separate ANNs. The first part, called encoder, takes as input the original sample and usually returns an output of smaller dimension than the original input, called latent space. Based on the reduced representation of the raw input, the second part, the so-called decoder, tries to restore the original input. During training, the whole model learns the most efficient low dimensional representation of the original input, to restore the original input as accurately as possible.

Based on the error between the in- and output, Meng et al. identify adversarial inputs. In addition, they use a specific form of AEs, so-called denoising auto encoders (DAEs) [Vin+08], to restore the original class of adversarial inputs. During the training of DAEs noise is added to the original inputs, but the restored input, i.e., the output of the model is compared to the clean original input. Thereby, the model learns to remove noise. However, Meng et al. report that the restoration is only possible for small amounts of adversarial perturbation. Lamb et al. [Lam+18] further propose to include several DAEs between the hidden layers of a given model to remove adversarial noise successively. More recently, Jia et al. [Jia+19] also propose to use a variant of a DAE, which compresses the original 24-bit maps describing the colour information of an input to 12-bit maps. In the latent space Gaussian noise is added "to improve the reconstructed quality, and further enhance the defense ability" [Jia+19]. Afterwards, the original input is reconstructed based on the noisy reduced version of the input.

Xie et al. [Xie+19a] propose a different approach to clean (adversarial) noise by so-called denoising blocks, which can be added to existing model architectures. Those blocks provide denoising operations like non-local means, bilateral, mean or median filters.

3.4.6. Ensemble Techniques

All prior defences are applied to a single model. However, using an ensemble of models is another common way to achieve better and more robust performance. One usual way to employ ensembles is to train different single models independently, to achieve the best possible classification accuracy on its own. Afterwards, the predictions of the different models are evaluated by a simple majority vote[1] to determine the final prediction. The intuition is that each model has its own decision boundary, which are similar but not identical among the ensemble members. This is due to the randomness in the training process, e.g., by the random initialisation. Thereby, some inputs are classified wrong in one model but are classified correctly in another. Even though adversarial inputs are transferable between models (Section 3.3.2), the success rate of adversarial attacks decreases when transferring manipulated inputs. Tramèr et al. [Tra+18] propose to combine an ensemble of models and adversarial training, to counter the transfer property of adversarial inputs. For each model, adversarial inputs are created and included in the training set. More recently, Pang et al. [Pan+19] propose a new training regime for ensembles, to increase the diversity between the ensemble members. Thereby, they can further increase the adversarial robustness.

Abbasi et al. [AG17] propose a different approach levering ensemble techniques. They observed that for each possible class, an untargeted FGSM attack had a high tendency to change the classification only to a limited number of false classes. Based on the calculated confusion matrix, they trained an ensemble of specialist models. Each specialist is only trained on a subset of classes, and thereby specialised to distinguish between those classes. In addition to the specialists, they trained a general model to differentiate between all classes. During inference, they found that the prediction confidence score for adversarial inputs decreases, in comparison to benign inputs. This observation was used in a second step to identify and reject adversarial inputs based on a threshold for the confidence.

3.5. Fast Gradient Sign Attack

The first adversarial attack optimised for ANNs was proposed by Goodfellow et al. [GSS15] and is called FGSM. The basic idea is to apply backpropagation to optimise the inputs regarding a certain output behaviour. As explained in Section 2.3.4, during training the inputs and corresponding labels are considered as fixed, while the parameters of the model are variable and change over the training process. To create adversarial inputs, the parameters θ of a model are assumed to be fixed, and the input, i.e., the pixel values are treated as variables. Backpropagation is then applied, i.e., the given loss function $\mathcal{L}$ is derived regarding the input pixel values $\mathbf{x}$ to minimise the loss for a given target classification t. To apply this concept, it is necessary to have full access to the target

[1] Assuming there are 5 different models and 4 of them predict class 'A' for a given input, while one model predicts class 'B' to be the true label. In a simple majority vote, the final prediction would be A. Aside simple majority votes, there are other voting strategies, cf. [SL13; AS14; Bab+15].

model, especially the parameters θ and the loss function $\mathcal{L}$, i.e., to operate in a white-box scenario. If these conditions are met, the following equation can be solved

$$\boldsymbol{\delta} = \varepsilon \cdot \operatorname{sign}\left(\nabla_{\mathbf{x}} \mathcal{L}\left(\mathbf{x}, t, \theta\right)\right) \ , \tag{3.4}$$

where ε is a hyper-parameter, controlling the maximum allowed amount of perturbation, and $\operatorname{sign}\left(\nabla_{\mathbf{x}} \mathcal{L}\left(\mathbf{x}, t, \theta\right)\right)$ denotes the sign of the gradient of the loss function $\mathcal{L}$. The value for ε is predefined and then fixed for all inputs to manipulate. However, a fixed value for ε has the disadvantage that if it is chosen too small, inputs could still be classified correctly regarding the ground truth, i.e., the attack is not successful. If the perturbation is too large, the minimality assumption in Equation 3.2 is not fulfilled.

It is therefore important to mention that the implementation of FGSM used in the later experiments is already an optimised version. Starting from an initial small value, ε is iteratively increased to find a minimal value large enough to fool the target model. But still, for each ε the attack calculates the gradient based on the original, and not on intermediate adversarial inputs. When the term FGSM is used in the following, it refers to the optimised version.

3.6. Examples of Adversarial Attacks

Following FGSM, several other adversarial attacks have been proposed, optimised for different objectives. In the following sections, different attacks are introduced, which are used in the later experiments.

3.6.1. Basic iterative method and projected gradient descent

To counter the drawback of the original FGSM's static ε, Kurakin et al. [KGB17a] propose the basic iterative method (BIM) which increases the applied perturbation iteratively until the desired misclassification is achieved, based on intermediate solutions. This can be formalised as

$$\begin{aligned} \mathbf{x'}_{n=0} &= \mathbf{x} \\ \mathbf{x'}_{n+1} &= \operatorname{clip}_{\mathbf{x},\varepsilon}\{\mathbf{x'}_n + \alpha \cdot \operatorname{sign}\left(\nabla_{\mathbf{x'}_n} \mathcal{L}\left(\mathbf{x'}_n, t, \theta\right)\right)\} \ , \end{aligned} \tag{3.5}$$

where $\mathbf{x}$ is the original input, $\mathbf{x'}_n$ is the (intermediate) adversarial input, and $\mathcal{L}\left(\mathbf{x'}_n, t, \theta\right)$ is the loss function, used to train the model in the first place, depending on the parameters θ of the model, the intermediate input $\mathbf{x'}_n$, and the target class t of the attack. The parameter α defines the step size of the attack, and is usually smaller than ε in FGSM. The intermediate result $\mathbf{x'}_n$ is then clipped to be within a given L_∞ ε-neighborhood of the source image $\mathbf{x}$. The clip function is process pixelwise and given by

$$\operatorname{clip}_{\mathbf{x},\varepsilon}\left(\mathbf{x'}_n\left(\mathtt{p}, z\right)\right) = \min\Big\{1, \mathbf{x'}_n\left(\mathtt{p}, z\right) + \varepsilon, \max\left\{0, \mathbf{x'}_n\left(\mathtt{p}, z\right) - \varepsilon, \mathbf{x'}_n\left(\mathtt{p}, z\right)\right\}\Big\},$$

where z indicates the colour channel of the given image, and $\mathtt{p}$ indicates the specific pixel to address. Another name for this attack is projected gradient descent (PGD), as proposed by Madry et al. [Mad+18].

3.6.2. Jacobian-based Saliency Map

Papernot et al. [Pap+16] introduce an attack called Jacobian-based saliency map attack (JSMA). This attack implements three steps, which are iterated until the desired misclassification is achieved or a given limit for introduced distortion is reached.

The first step is to calculate the Jacobian matrix of the function which the neural network implements. The resulting matrix contains for each possible class $c \in \{1, \ldots, C\}$ the derivative regarding all input features $\mathtt{p}$, i.e., the pixel of the input image. The calculation can be formalised as

$$\nabla f(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \left[\frac{\partial f_c(\mathbf{x})}{\partial \mathbf{x}_\mathtt{p}} \right] ,$$

where f is the model to attack, and $\mathbf{x}$ is the given input.

Based on the derivatives, the second step is to calculate a saliency map to determine which input feature has the largest influence on the decision towards the desired target class. With the same variables as previous, the salience map is calculated by

$$S(\mathbf{x},t)_\mathtt{p} = \begin{cases} 0 & \text{if } \frac{\partial f_t(\mathbf{x})}{\partial \mathbf{x}_\mathtt{p}} < 0 \text{ or } \sum_{j \neq t} \frac{\partial f_j(\mathbf{x})}{\partial \mathbf{x}_\mathtt{p}} > 0 \\ \left| \frac{\partial f_t(\mathbf{x})}{\partial \mathbf{x}_\mathtt{p}} \right| \left(\sum_{c \neq t} \frac{\partial f_c(\mathbf{x})}{\partial \mathbf{x}_\mathtt{p}} \right) & \text{otherwise} \end{cases}$$

where t is the desired target class. It is also possible to define the Saliency map to indicate inputs, which should be decreased in order to achieve the desired classification. In the implementation provided by *foolbox* [RBB17], for each pixel the impact on the target classification is taken into account, regardless whether the pixel value has to be in- or decreased.

The third and last step of each iteration is to perturb the pixel with the overall largest impact on the classification, in the calculated direction, i.e., either decrease or increase the pixel for a given amount.

3.6.3. Carlini and Wagner

Carlini and Wagner [CW17] introduce an attack, usually referred to as CW. The attack used in the later experiments is the L_2 based attack, which, given an input $\mathbf{x}$, searches for the perturbation $\boldsymbol{\delta}$, by a change of variables defined as

$$\boldsymbol{\delta}_\mathtt{p} = \frac{1}{2} \left(\tanh\left(\mathbf{w}_\mathtt{p} \right) + 1 \right) - \mathbf{x}_\mathtt{p}$$

that solves

$$\underset{\mathbf{w}}{\text{minimize}} \left|\left| \frac{1}{2} \left(\tanh\left(\mathbf{w} \right) + 1 \right) - \mathbf{x} \right|\right|_2^2 + c \cdot g\left(\frac{1}{2} \left(\tanh\left(\mathbf{w} \right) + 1 \right) \right),$$

with g defined as

$$g(\mathbf{x}') = \max\left[\max\left(Z(\mathbf{x}')_c : c \neq t \right) - Z(\mathbf{x}')_t , -\kappa \right] ,$$

where $Z(\mathbf{x}')$ are the logits of the model for the adversarial candidate $\mathbf{x}'$, t is the target class of the attack, and κ controls the confidence with which the misclassification should occur.

3.6.4. Deepfool

DeepFool (DF) is an untargeted adversarial attack proposed by Moosavi-Dezfooli et al. [MFF16]. The algorithm has three steps, which are iterated until the classification of the adversarial input $\mathbf{x}'$ is different to the classification of the original input $\mathbf{x}$, i.e., $f(\mathbf{x}) \neq f(\mathbf{x}')$, where f is the model to attack. Moosavi-Dezfooli et al. proposed their attack initially based on the L_2-norm, but as well generalised it to any L_p-norm. In the later experiments, when speaking of DF, the L_∞ implementation is meant.

After initialising $\mathbf{x'}_{n=0} = \mathbf{x}$, the first step of the algorithm is to calculate the differences of the gradients $\mathbf{w}'_c$ and predictions p'_c for each class c unequal the initially predicted class c^*. Formally, this can be written as

$$\begin{aligned} \mathbf{w}'_c &= \nabla f_c(\mathbf{x}_N) - \nabla f_{c^*}(\mathbf{x}_N) \\ p'_c &= f_c(\mathbf{x}_N) - f_{c^*}(\mathbf{x}_N) \ . \end{aligned}$$

The second step is to calculate the closest hyperplane h, separating the original class from one of the other classes. Formally

$$h = \arg\min_{c \neq c^*} \frac{|p'_c|}{||\mathbf{w}'_c||_1} \ ,$$

and further on they calculate the minimal perturbation $\boldsymbol{\delta}_n$ needed to project the current adversarial $\mathbf{x'}_n$ onto that hyperplane. Formally

$$\boldsymbol{\delta}_n = \frac{|p'_h|}{||\mathbf{w}'_h||_1} \operatorname{sign}(\mathbf{w}'_h) \ .$$

The last step of each iteration is to update the intermediate adversarial input with the calculated perturbation, i.e., $\mathbf{x'}_{n+1} = \mathbf{x'}_n + \boldsymbol{\delta}_n$, and start the next iteration by checking if $\mathbf{x'}_{n+1}$ is already misclassified.

3.7. Transfer to the Real World

At first adversarial attacks have been applied to digital inputs to show the potential weaknesses of neural networks. Based on the digital application, Lu et al. [Lu+17] claim that there is "NO Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles". They state that in the real world the distance, angle, or rotation of a camera can vary, and thereby adversarial inputs can not robustly fool a given model. To prove their claim, they printed out a manipulated version of a stop sign and attached it to a post. Taking pictures at different distances and angles showed that the stop sign is still classified correctly in the vast majority of situations. However, shortly after Athalye et al. [Ath+18] have proposed a framework to create robust adversarial inputs in the real world, called Expectation over Transformation (EoT).

Their basic idea is to predefine a distribution Tr over expected transformations occurring when transferring adversarial inputs into the real world, like rotation, scaling, tilting, and others. For each transformation $\mathrm{tr}_1, \ldots, \mathrm{tr}_E$ a value range is defined, e.g., rotation in

$[-10°, 10°]$. To describe this problem formally, they redefine the basic formulation for adversarial inputs (Equation 3.2) to be

$$\begin{aligned} \arg\max_{\mathbf{x}'} \quad & \mathbb{E}_{\mathrm{tr}\sim\mathrm{Tr}}\left[\log P\left(t \mid \mathrm{tr}^*\left(\mathbf{x}'\right)\right)\right] \\ \text{subject to} \quad & \mathbb{E}_{\mathrm{tr}\sim\mathrm{Tr}}\left[\mathcal{D}\left(\mathrm{tr}^*\left(\mathbf{x}'\right), \mathrm{tr}^*\left(\mathbf{x}\right)\right)\right] < \varepsilon \\ & \mathbf{x}' \in [0,1]^I \end{aligned} \tag{3.6}$$

where they aim to find the adversarial input $\mathbf{x}'$, which maximises the expected value $\mathbb{E}$ of the logarithmised probability $\log P$ that the adversarial input $\mathbf{x}'$, transformed by the true but unknown transformation tr^*, is classified as the desired target class t. In addition, the distance $\mathcal{D}$ between the original and adversarial input, transformed by the true transformation should not exceed a certain ε, and the adversarial input have to be in the bounds of the original input space.

Internally, Equation 3.6 can be solved by the following iterative equation

$$\begin{aligned} {\mathbf{x}'}_{i=0} &= \mathbf{x} & {\mathbf{x}'}_{i+1}^{j=0} &= {\mathbf{x}'}_i \\ {\mathbf{x}'}_{i+1}^{1\leq j\leq E} &= \mathrm{tr}_j\left({\mathbf{x}'}_{i+1}^{j-1}\right) & {\mathbf{x}'}_{i+1} &= \mathcal{A}\left({\mathbf{x}'}_{i+1}^{j=E}\right) \ , \end{aligned}$$

where initially the original input $\mathbf{x}$ is assigned to be the first intermediate adversarial input $\mathbf{x}'$. Afterwards, all transformations tr are applied successively with a random value sampled from the corresponding value ranges. Finally, an arbitrary adversarial attack $\mathcal{A}$ is executed and the overall iteration proceeds, until a certain number of iteration steps is exceeded.

The authors emphasize that the possible transformations are not limited to rotation or scaling, but also to the texture of the observed object or the shape, which might change when taking pictures of 3-D physical objects. Thereby, the authors can print out 2-D, or even 3-D objects which are robustly, i.e., recorded in different angles, distances, and other transformations, classified as the desired class. Furthermore, the framework is independent of the attack used to create the adversarial images. This implies that an adversary can use white-box attacks if there is access to the model, or black-box attacks when no further knowledge about the model to attack exists.

4. Physical Adversarial Attacks by Projecting Perturbations

CNNs yield remarkable results in the domain of image classification. A specific use-case is the classification of street signs in the domain of autonomous driving systems. In this scenario, the German Traffic Sign Recognition Benchmark (GTSRB) [Sta+11] dataset is widely used, with reported accuracies of more than 99% on the test set, e.g., in [Mao+16; Hoa+18]. However, as described in Chapter 3, it is possible to fool state of the art classifiers by so-called adversarial inputs. Those specifically created inputs aim to fool the classification model to misclassify a given input, in comparison to the correct classification given by a human. One prominent example in street sign classification is to manipulate a stop sign to be classified as a priority sign by a given ANN. Some examples of successful attacks are shown in Figure 4.1.

Besides, Kurakin et al. [KGB17a] pointed out the possibility of transferring adversarial inputs into the real world. Later, Athalye et al. [Ath+18] proposed a framework (Section 3.7) to create more robust adversarial inputs. For that purpose, they printed out manipulated versions of a stop sign, attached them somewhere in the real world, and recorded them again with a camera. This led to robust misclassification of the printed stop sign as a priority road sign. An example of such a print out is displayed in Figure 4.1a.

Aside from the work of Athalye et al., Eykholt et al. [Eyk+18] suggested attaching stickers to the street signs which look like graffiti. An example is shown in Figure 4.1b. They argue that graffiti is widely known by humans and considered as normal. Therefore, even though the stickers can clearly be seen by humans, they are not recognised as a threat. Another, quite different approach by Sitawarin et al. [Sit+18] is to manipulate corporate logos or advertisements to be recognised as the desired street sign, as shown in Figure 4.1c. A human would usually recognise an advertisement, while the car would stop because it detects a stop sign.

The attacks mentioned have in common that the attacker has to manipulate the street sign or the surroundings physically. To fool the classification systems, this requires the attachments to not be torn off or worn-out by weather conditions or pedestrians. Furthermore, at least in the case of an incident, the physical manipulation might be detected by the police or insurances. Probably the manipulation is even noticed earlier and removed or altered by a human, such that the adversarial attack did no harm at all.

In this chapter, a new attack scenario is presented, in which the necessary perturbations are projected onto the street sign with a regular office projector. This means there is no need to physically manipulate the street sign itself, but the attack can still be carried out in the physical world. Also, the usage of a laser pointer is simulated, which might

(a) Subtle Poster [Eyk+18] (b) Camouflage Art [Eyk+18] (c) Corporate Logo [Sit+18]

Figure 4.1.: Examples of current real world attacks on street signs, especially to alter a stop sign to be recognised as a priority road sign.

be handier in a real-life scenario. Therefore, the attack is restricted to only manipulate one colour channel, in particular the green one. The usage of projections over physical manipulations has the advantage for an attacker that non-physical manipulations are harder to proof for insurances or the police. Another advantage is that the projector could be controlled via the internet and only be activated under certain conditions, e.g., when a passing car is detected. Thereby, the detectability of the manipulation would be impeded. In addition to only allow one colour channel to simulate the usage of a laser pointer, employing a projector implies additional, more general restrictions like non-decreasing pixel values when projecting the perturbation. This is because by projecting an image onto an existing surface, the recaptured value of the surface can only be brightened, i.e., increased. Projecting a "shadow", i.e., decreasing the recorded value in comparison to no projection, is hardly noticeable.

The remainder of this chapter is organised as follows. In Section 4.1 the adaptations made to the adversarial attack are explained. Afterwards, the experimental setup for the conducted experiments is outlined in Section 4.2, followed by the corresponding results in Section 4.3. Finally, Section 4.4 concludes this chapter. Parts of the results have been published in [WKK19] and have been achieved under the supervision of the master thesis "Using projectors to deceive traffic sign classifiers with projections of physical adversarial perturbations on a single color channel basis" of Hendrik Kahlen in 2018.

4.1. Adaptation of the Attack

As mentioned in the introduction of this chapter, certain adaptations to the attack are necessary to comply with the described scenario. At first, because the perturbations are projected onto the street sign, the attack is restricted to only increase the pixel values. A decrease in pixel values would hardly be visible when projected. This is implemented by calculating the perturbation $\boldsymbol{\delta}$ as the difference between the (intermediate) adversarial

image $\mathbf{x}'$ and the original image $\mathbf{x}$ after each iteration of the attack, and truncate $\boldsymbol{\delta}$ to the range [0,1] by

$$\boldsymbol{\delta} = \min\left(1, \max\left(0, \mathbf{x}' - \mathbf{x}\right)\right) \ . \tag{4.1}$$

The second adaptation is based on the assumption that the attacker might use a laser pointer with a specific attachment to project the perturbation. Since laser pointers normally only have one colour, the following equation is applied to restrict the attack to manipulate one colour channel

$$RGB_{\mathbf{x},U}\{\mathbf{x}'\}\left(\mathtt{p},z\right) = \begin{cases} \mathbf{x}_{\mathtt{p},z} & z \in U \\ \mathbf{x}'_{\mathtt{p},z} & \text{else} \end{cases} \ , \tag{4.2}$$

where $\mathbf{x}$, $\mathbf{x}'$ denotes the original, resp. adversarial image, $\mathtt{p}$ is a specific pixel, and z is the colour channel of the image. The variable U defines the set of unwanted colour channels over the set $\{r,g,b\}$, where r is the red colour channel, g the green, and b the blue one. Thereby, only the wanted colour channel is perturbed, while the unwanted are kept on their original value. Thus, the initial formulation for BIM (see Equation 3.5) can then be extended to

$$\begin{aligned} &\mathbf{x}'_{n=0} = \mathbf{x} \\ &\mathbf{x}'_{n+1} = RGB_{\mathbf{x},U}\Big\{\mathrm{clip}_{\mathbf{x},\varepsilon}\Big\{\mathbf{x}'_n + \alpha \cdot \mathrm{sign}(\nabla_{\mathbf{x}'_n}\mathcal{L}\left(\mathbf{x}'_n, t, \theta\right))\Big\}\Big\} \end{aligned} \tag{4.3}$$

where in each iteration the perturbation is clipped according to Equation 4.1.

4.2. Experimental Setup

The GTSRB [Sta+11] is used, which consists of 39,209 training and 12,630 test images of more than 40 different classes. Based on this dataset, two CNNs are trained, namely Inception-v3 [Sze+16] and VGG-16 (version D) [SZ15], with 95.2% and 94.02% accuracy, respectively. The adapted attack (Section 4.1) is used to generate the adversarial examples. The aim of the attack is always to transfer a stop sign into a priority sign. Furthermore, an attack is only considered to be successful if the target class is top-1, i.e., the input is classified as the target class with the highest confidence among all possible classes. This explicit assumption differs from other works like [KGB17a], where the authors observe the top-5 success, i.e., the target class is among the classes with the highest five confidences. However, I question the general applicability of the top-5 accuracy in the used case of autonomous driving. Even though the correct class might be among the five classes with the highest confidences, the question concerning the correct reaction is still unsolved[1]. In

[1] The top-5 accuracy can be useful in contexts where different individual classes can be grouped to a higher level class. For example, knives and forks are both cutlery. If an image of a knife is classified as a fork, with the second-highest confidence score belonging to the class 'knife', the detailed classification might be wrong, but both classes belong to the superclass cutlery, and thereby some semantic meaning of the top-5 classes can be extracted. In terms of street sign classification, there is only one stop sign and no superclass other than the general class 'street sign'.

this chapter, it is assumed that a real-life model is only able to make a decision based on the class with the highest confidence.

As framework to transfer the adapted attack into the physical world EoT (see Section 3.7) is used. The different transformations assumed to occur, and their value ranges are given in Table 4.1. The transformations scale, rotation, brightness, and Gaussian noise are taken from the paper of Athalye et al. [Ath+18]. Contrast, gamma, saturation, and 'salt & pepper' are added to the set of transformations. For further information on the different image transformations, see for example [BB16].

Table 4.1.: Value ranges for the assumed transformations.

scale	$\in [0.9, 1.1]$	contrast	$\in [0.9, 1.1]$
rotation	$\in [-10.0°, 10.0°]$	gamma	$\in [0.875, 1.125]$
brightness	$\in [-0.05, 0.05]$	saturation	$\in [0.93, 1.07]$
Gaussian noise	$\in \mathcal{N}(0, 0.1)$	salt & pepper	70%

The first experiments are conducted virtually, to verify the theoretical applicability of the adapted attack and assumed transformations in an ideal environment. The corresponding results are given in Section 4.3.1.

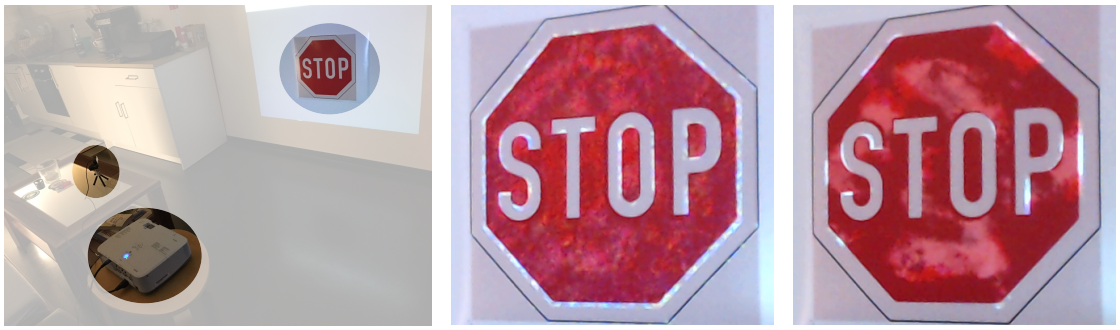
Regarding the physical applicability of the attack, two scenarios are investigated. The first one is to project the created adversarial image onto a white wall and track the adversariality, i.e., the percentage of adversarial images classified as the target class, as well as the top-1 confidence of the captured image classification. The results for the corresponding experiments are given in Section 4.3.2. In the second scenario, only the adversarial perturbation is projected onto a printed stop sign, attached to a white wall. The second scenario is further referred to as physical adversarial perturbation (PAP) and the respective results are outlined in Section 4.3.3. The setup for the physical experiments is depicted in Figure 4.2a. A 1080p home projector, a 1080p webcam, and a professional printout of a stop sign are used. The captured images of the stop sign are then semi-automatically cropped to the necessary area.

Two examples of manipulated stop signs captured by the webcam located at an angle of 45° to the wall are shown in Figure 4.2. The manipulations to the stop sign in Figure 4.2b are calculated based on the Inception-v3 model, the manipulations in Figure 4.2c on the VGG-16 model.

For all experiments, the attack has been restricted in different ways. When referring to 'plain', all colour channels are allowed to be increased and decreased, i.e., no restrictions are applied. Whereas, 'rgb_1' expresses that only the green colour channel is manipulated, and the postfix 'inc' indicates that the attack only increases the pixel values.

4.3. Results

In this section, the results of the experiments described in Section 4.2 are presented, starting with the preliminary virtual study to verify the general approach under optimal conditions in Section 4.3.1. Afterwards, in Sections 4.3.2 and 4.3.3, the observations are



(a) Picture of the setup, consisting of a webcam, a projector, and a printout of a stop sign.

(b) Example of an adversarial input based on Inception-v3.

(c) Example of an adversarial input based on VGG-16.

Figure 4.2.: Setup for the physical tests and two example of projected adversarial perturbations.

given, when the complete adversarial images are projected onto a wall, resp. only the perturbations are projected onto the printed stop sign. In both sections, a 45° camera angle to the projection is assumed, to resemble a more realistic recording of the street sign by an autonomous car. However, in Section 4.3.4 the scenario that the camera is located directly in front of the projection is further investigated.

4.3.1. Preliminary Virtual Study

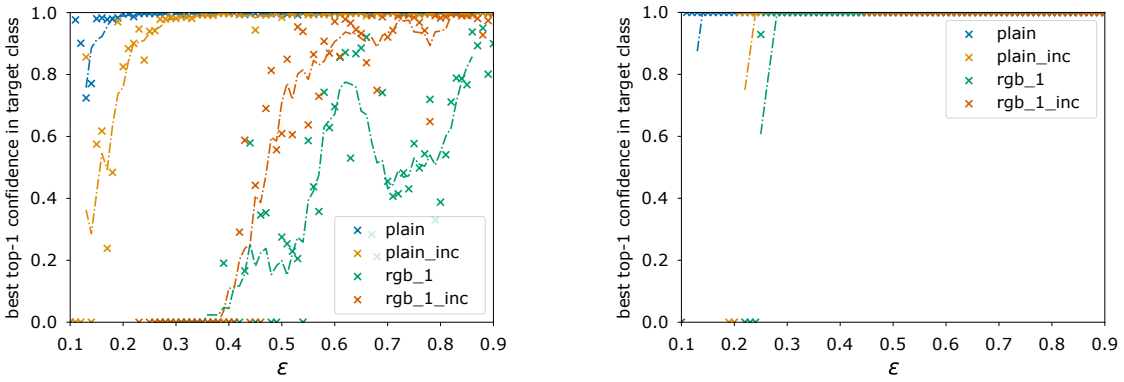
The first goal of the preliminary virtual study is to verify the general setup and to reproduce results published in the literature. Therefore, the attack is allowed to manipulate all colour channels and to both increase and decrease the pixel value. In this basic scenario, an adversariality, i.e., the number of successful adversarial attacks, of 99.89% on Inception-v3, respectively 98.36% on VGG-16 is achieved. These results are slightly better than the reported adversariality of 96.4% on Inception-v3 reported by Athalye et al. [Ath+18].

Based on the verification of the general setup, the adaptations to the attack are applied to establish its applicability in an optimal surrounding. Considering the attack restrictions 'rgb_1_inc', an adversariality of 72.8% on Inception is achieved, respectively 52.04% on VGG. Based on these results, the attack is transferred to the physical world.

4.3.2. Projection of Adversarial Images

In this section, the scenario that an attacker projects the complete manipulated stop sign onto a white wall is considered. Figure 4.3a depicts the results for successful attacks, i.e., the target class is top-1, when attacking the Inception-v3 model. The corresponding results for VGG-16 are given in Figure 4.3b. There the ε-values, which indicate the amount of perturbation allowed for the attack (see Equation 4.3), and the top-1 confidence

of the projected adversarial images are plotted. Each cross illustrates the mean confidence of two adversarial inputs for the given ε.



(a) Top-1 confidences for Inception-v3.

(b) Top-1 confidences for VGG-16.

Figure 4.3.: Top-1 confidence of projected adversarial images for different perturbation levels.

When the attack restriction 'plain' is considered, confidence levels near 100% are achieved in the adversarial class (priority sign) at low perturbation levels for both attacked models. If the attacker is restricted to only increase the pixel values or manipulate a single colour channel, the two models behave differently.

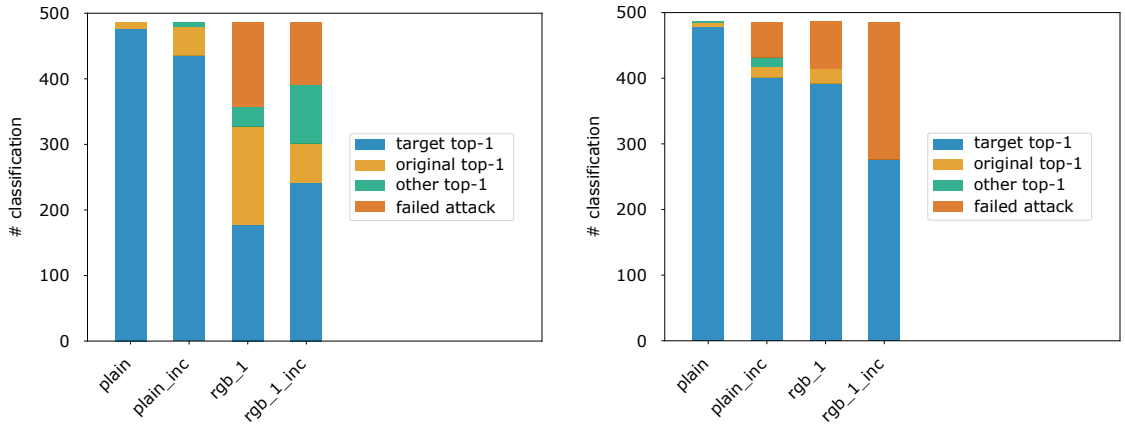
Restricting the attack to scenario 'plain_inc', the confidence levels in the target class raise from 0% to around 100% for perturbation level below 0.3, when attacking Inception-v3. For higher perturbation levels than 0.3, the confidence is very near 100% in this scenario. In comparison, when attacking VGG-16 with the same attack restrictions, the confidence level is more or less binary. Either the confidence is around 0% or 100%, but no noticeable confidences between are measured. Concerning the amount of perturbation, VGG-16 starts being fooled for perturbation levels around 0.2.

The same binary behaviour for VGG-16 also appears if the attacker is restricted to attack scenario 'rgb_1', or even further to scenario 'rgb_1_inc'. The necessary amount of perturbation to fool the model increases as the attacker is further restricted, to be around 0.25 for 'rgb_1' and 0.45 for 'rgb_1_inc'. In comparison, for Inception-v3 continuous confidence scores between 0% and 100% are monitored when further restricting the attacker. More interestingly, scenario 'rgb_1_inc' achieves robustly higher confidence scores at lower perturbation levels, compared to the 'rgb_1' scenario.

Taking a closer look at the adversariality, scenario 'plain' can fool the Inception-v3 model in 97.94% of the cases, resp. VGG-16 in 98.35% of the cases overall observed perturbation levels of ε. This is slightly less compared to the virtual experiment where an adversariality of 99.89% on Inception, resp. 98.36% on VGG is achieved. When restricting the attack to scenario 'plain_inc', the success rate of the attacks reduces to 89.71% for Inception-v3, resp. 82.51% on VGG-16. If the attack is controlled to manipulate the green colour channel, Inception-v3 is only fooled in 36.42% of the cases, while the adversariality for VGG-16 only slightly decreases to 80.66%. More interestingly, the success rate of

the attacks for Inception-v3 increases to 49.59%, when scenario 'rgb_1_inc' is assumed. However, this complies with the observation for the confidence scores. For VGG-16, the adversariality decreases further to 56.79% under condition 'rgb_1_inc'.

Concerning the adversariality, it is noteworthy that the unsuccessful adversarial attacks combine images for which the manipulation already failed in the generation process (failed attack), the manipulated image is still recognised as stop sign (original), or the manipulated image is recognised as a completely different street sign (other top-1), as shown in Figure 4.4. For VGG-16 an attack is basically either successful (target top-1) or the attack failed to find an adversarial image at all (failed attack), as shown in Figure 4.4b. In contrast, for Inception-v3 under restriction 'rgb_1_inc' a proportion of 18.31% of the attacks did not bring the target class into top-1, but another class unequal to the original label.



(a) Distribution of successful and unsuccessful inputs for Inception-v3.

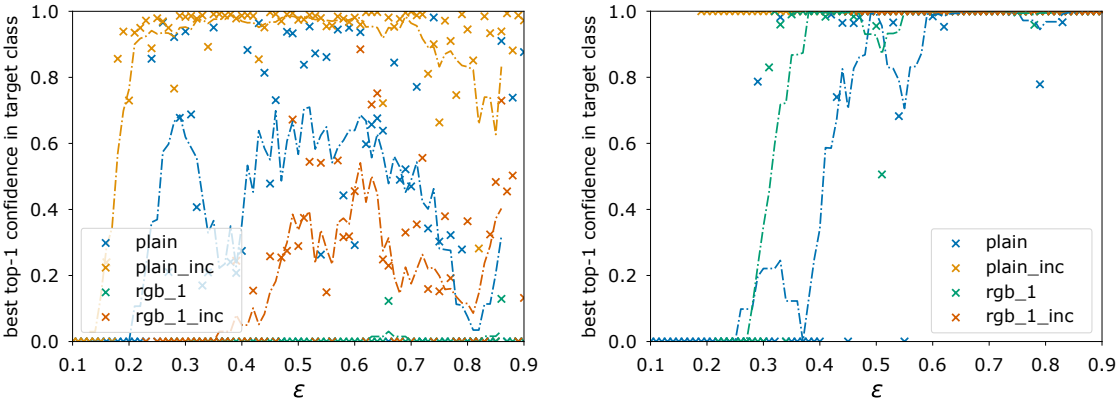
(b) Distribution of successful and unsuccessful inputs for VGG-16.

Figure 4.4.: Distributions of successful (target top-1) and different type of unsuccessful adversarial attacks, while projecting adversarial images.

In the experiments, an attack is not considered as successful, if the resulting image is neither classified as the target class, nor as to the original true class. However, in the context of autonomous cars, those inputs could still yield the desired chaos, where wrongly classified stop signs can have fatal effects on the traffic.

4.3.3. Projection of Adversarial Perturbation

The third experiment investigates the projection of the perturbations onto a printed stop sign, and the top-1 confidence results are shown in Figure 4.5. In comparison to the projected complete adversarial images, the confidence levels are worse in almost all cases. But it is noticeable that for VGG-16, the confidence levels are still near 100% if an adversarial attack is successful. It is also noticeable that the 'inc' variations, i.e., those attacks which are optimised to only increase the pixel values, achieve better results in general than the comparable attacks without 'inc'.



(a) Top-1 confidences for Inception-v3.

(b) Top-1 confidences for VGG-16.

Figure 4.5.: Top-1 confidence of projected adversarial perturbations for different perturbation levels.

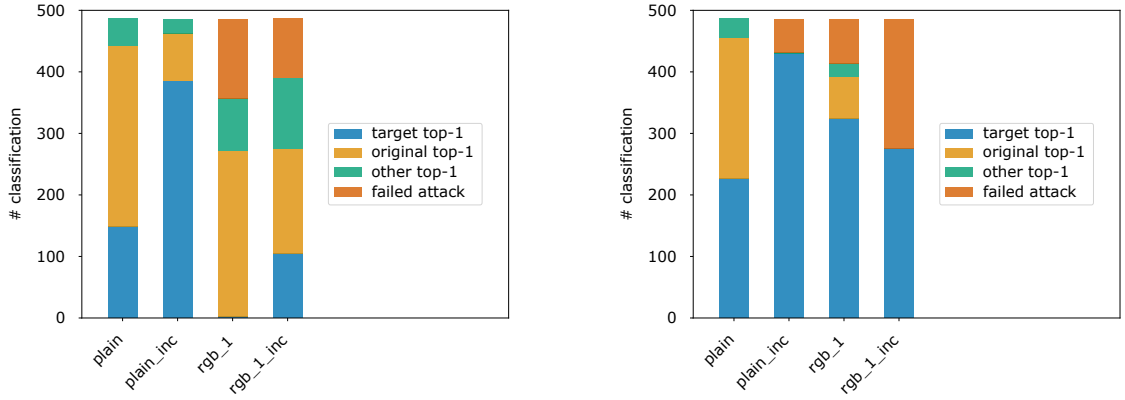
In terms of the adversariality, the non-'inc' variations for Inception-v3 drop from 97.94% to 30.45% under restriction 'plain', and from 36.42% to 0.41% under restriction 'rgb_1', when compared to the fully projected adversarial images. For VGG-16 the adversariality reduces from 98.35% to 46.50% under restriction 'plain', and from 80.66% to 66.67% for restriction 'rgb_1'.

Considering the restriction 'plain_inc' and Inception-v3, the adversariality when projecting the full adversarial image of 89.71% reduces only to 79.22%. In contrast, for VGG-16 the adversariality improves from 82.51% to 88.48%. Observing 'rgb_1_inc', again for Inception-v3, the adversariality diminishes from 49.59% to 21.61%, but for VGG-16 the adversariality stays the same at 56.79%.

A detailed overview of the distribution over the different outcomes of the attacks is given in Figure 4.6. Similar to the results in Section 4.3.2, it is interesting that for VGG-16, an attack is either successful or fails at all because either no adversarial image could be created or the adversarial input is still classified as original. Attacking Inception-v3, however, there are between 4.94% and 22.97% cases, in which the manipulated stop sign is neither classified as the original, nor as to the target label.

4.3.4. Consideration of no Angle towards the Projection

For the previous results, an angle of 45° between the projection on the wall, and the camera recapturing the stop sign is considered. In this section, the camera is assumed to be directly in front of the projection, i.e., recapturing the stop sign with the projected manipulation at an angle of 0°. For the inception-v3 model, the classification results of the manipulated stop sign are shown in Figure 4.7a. The attacks completely fail to put the target label at top-1 confidence, however 35.18% and 41.77% of the inputs are classified as something else that is neither the original nor the target class. In contrast, when attacking the VGG-16 model (Figure 4.7b) the successrate of the attacks



(a) Distribution of successful and unsuccessful inputs for Inception-v3.

(b) Distribution of successful and unsuccessful inputs for VGG-16.

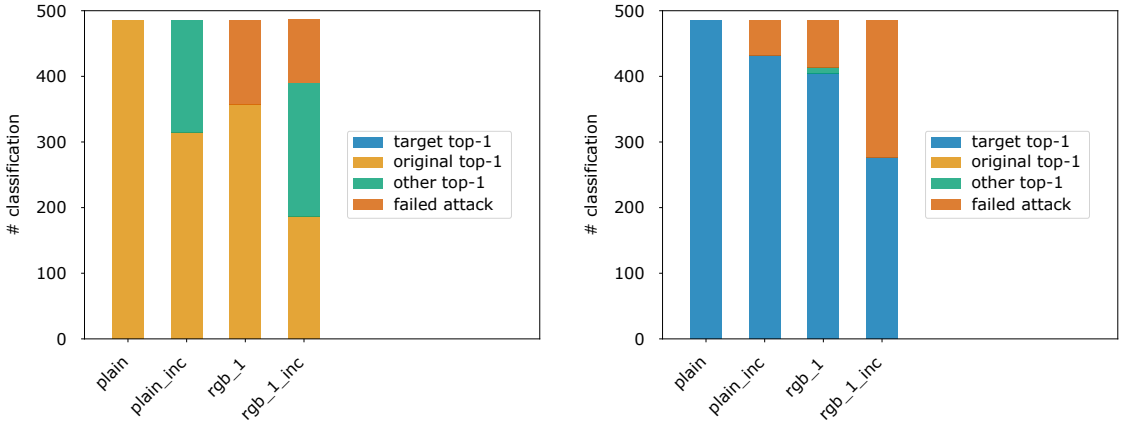
Figure 4.6.: Distributions of successful (target top-1) and different type of unsuccessful adversarial attacks, while projecting adversarial perturbations.

increases. Especially with no restrictions to the attack ('plain'), considering an angle of 45° between projection and camera, 47.12% of the manipulated inputs are classified correctly. Positioning the camera directly in front of the projection leads to a 100% success rate of the adversarial attack.

This behaviour could be explained by differences in the model architectures and reflections, which occur when having zero angle between the projection and the camera. The model architecture of VGG-16 is formed like a cone, i.e., the earlier layers of the classification pipeline have a larger spatial resolution, and the layers are connected in a straight feed-forward manner [SZ15]. When backpropagating the adversarial manipulation, a larger area of the original input is affected, due to the spatial compression of the input during the inference. A sample for the created perturbation for VGG-16 is shown in Figure 4.2c. The pattern already resembles reflections, which are further enhanced by the reflections of the printed stop sign. In comparison, the Inception-v3 architecture contains parallel convolution layers with different filter sizes, to preserve spatial information of the input. Thereby, backpropagating the adversarial manipulations can lead to specific pixels, rather than larger areas, which have to be altered to change the classification. The adversarial patterns look similar to Figure 4.2b, and compared to Figure 4.2c the manipulations are much finer and therefore disturbed by the reflections of the printed stop sign.

4.4. Conclusion

Previous attacks on traffic signs assume direct physical access to the sign or its environment. Eykholt et al. [Eyk+18] for example proposes to attach stickers or graffiti, while others (e.g., Athalye et al. [Ath+18]) propose to print out a manipulated version and cover the original street sign or attach the manipulation somewhere else. Because of the



(a) Distribution of successful and unsuccessful inputs for Inception-v3.

(b) Distribution of successful and unsuccessful inputs for VGG-16.

Figure 4.7.: Distributions of successful (target top-1) and different type of unsuccessful adversarial attacks, while projecting adversarial perturbations with an angle of 0° between camera and projection.

physical manipulation, those attacks might be easy to recognise by, e.g., the police or insurance companies.

In this chapter, a new threat scenario has been proposed, in which a regular business or home projector is used to project the manipulations onto a printout of a stop sign. Those projections alone deceive the target models to classify the initial stop sign with the adversaries target class, i.e., as a priority road sign. At first, the adversarial attack is allowed to manipulate all colour channels but is later restricted to only manipulate one colour channel, in particular the green one. This is to simulate the usage of a limited projector like a laser pointer, to successfully fool the target models. In the proposed scenario, physical manipulation of the road sign itself is no longer necessary, which could make it more difficult to prove manipulation in the event of an accident. Furthermore, an attacker could synchronise the attack with passing cars such that no perturbation is projected when no traffic passes. This imposes new dangers and makes the understanding of neural networks and how they behave under attack even more important in the domain of safety-critical systems.

To verify the applicability of this threat scenario, necessary restrictions to the basic attack are imposed, namely that 1) only one colour channel is manipulated to simulate the usage of a laser pointer and 2) the attack is only allowed to increase the pixel values. The latter adjustment is necessary because projections of darker values are hardly noticeable in real-life scenarios. The presented results show that it is indeed possible to fool neural networks by only projecting adversarial images to a white wall. Restricting the adversarial attack to exclusively increase values of the green colour channel, an adversariality of 49.59% for the investigated Inception-v3 model, resp. 56.79% for the VGG-16 model is reported. Even more concerning is that only projecting the perturbations to a printed sign leads to a high number of successful attacks. For that scenario and applying the

strongest restrictions to the attack, Inception-v3 can be fooled in 21.61% of the cases, whereas VGG-16 can be fooled in 56.79% of the cases.

As the two models have different adversarialities, an option for future work could be to investigate the effect of the model architecture on the appearance of adversarial inputs further. In Section 4.3.4, it is shown that the adversarial manipulations based on the VGG-16 model resemble reflections, which might be the reason why the adversariality even increased when the camera is directly located in front of the projection. For Inception-v3 the adversarial manipulations are subtler, and therefore might be disturbed by the reflections, which in turn leads to unsuccessful adversarial attacks.

Another direction of future research could be to extend the proposed threat scenario by using black-box attacks (Section 3.3.2) and the classification output of the visual system of a real car to test adversarial attacks in a more realistic scenario. Recently, Ranjan et al. [Ran+19] demonstrated that it is possible to fool a real autonomous car, however, they used crafted printouts as adversarial attacks. Using projections would be easier and faster than printing out the manipulated version of a street sign. In addition, using projections enables to perform a sequence of adversarial attacks which preserves the common sequence of observed traffic signs when, for example, changing from the freeway into the city.

Part II.

Adversarial^{-1}

5. Adversarial^{-1}: Defending by Attacking

As demonstrated in Chapters 3 and 4, modern ANNs are prone to adversarial inputs. Those are slightly perturbed inputs, which fool a given classifier into misclassification, even though the manipulated inputs would still be classified correctly by humans.

In this chapter, a new defence against adversarial inputs is presented, which was inspired by a paper of Tabacof and Valle [TV16]. They investigate the research question, if adversarial inputs only "exist as isolated points in the pixel space, reachable only by a guided procedure" [TV16] or if "they inhibit large and contiguous regions in the space" [TV16]. To answer that question, they used a process, schematically shown in Figure 5.1, where the different coloured areas indicate the class, which is assigned to inputs within this area. The original input $\mathbf{x}$ (circle) is manipulated by an adversarial attack (bold line) to be classified wrongly $\mathbf{x}'$ (cross). Afterwards, they perturbed the adversarial input with different amounts of random noise (pentagons). The randomly manipulated inputs $\mathbf{x}''$ are created by $\mathbf{x}'' = \text{clamp}\,(\mathbf{x}' + \varepsilon)$, where $\varepsilon \sim \mathcal{N}\,(\mu,\, \lambda\sigma^2)$, and clamp limits the altered input to be in the acceptable input space [0,1]. The scaling factor λ within the normal distribution $\mathcal{N}$ ranges from 2^{-5} to 2^5, while μ, σ^2 are the mean and variance of the pixel-wise difference between the original input $\mathbf{x}$ and the corresponding adversarial counterpart $\mathbf{x}'$. Afterwards, they classified the randomly perturbed inputs and counted how many of them remained adversarial, and how many were classified as the original, as well as some other arbitrary class after the random perturbation.

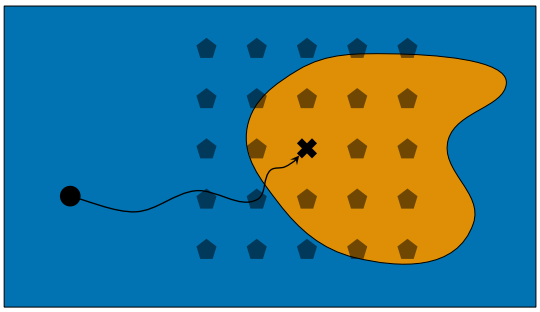


Figure 5.1.: Scheme of the process used by Tabacof and Valle [TV16] to determine the size of adversarial islands. The colours indicate areas of a different classification. The original input (circle) is attacked (bold line) to be an adversarial input (cross). Afterwards, random noise is applied to the adversarial input to create new, randomly perturbed inputs (pentagons). Those pentagons indicate that adversarial inputs lay within "adversarial islands", likely surrounded by the original true class.

Their results suggest that adversarial inputs span larger areas. Even at medium perturbation levels, i.e., $\lambda = 2^0$, depending on the observed dataset between $\approx 30\%$ and $\approx 75\%$ of the randomly perturbed adversarial inputs are still classified as the initial adversarial input. Besides, their results indicate that adversarial inputs which are moved out of the adversarial areas are often classified correctly afterwards.

Based on their observations, in this chapter, the random perturbation is substituted by an internal adversarial attack. This concept is depicted in Figure 5.2. As before, the colour shades indicate different classification areas of inputs. An original input (circle) is attacked (bold line), and thereby brought into an adversarial area (cross). Afterwards, instead of applying random noise, another adversarial attack (dashed lines) to the adversarial input is applied. To differ between inputs which are attacked one, resp. two times, those inputs which have been attacked two times are called adversarial^{-1} inputs, indicated in Figure 5.2 as question marks.

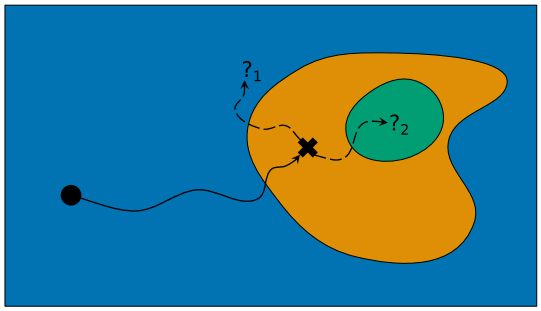


Figure 5.2.: Scheme of the proposed research questions, if original and adversarial inputs behave differently when attacked. The colours indicate areas of classification. The original input (circle) is attacked (bold line) to be an adversarial input (cross). After another adversarial attack (dashed lines), it is to be investigated, if the adversarial^{-1} input (question marks) return to the original classification area (question mark 1) or change its classification to another, different class (question mark 2).

Based on this concept, the initial goal was to answer two research questions:

- Is it possible to distinguish between original and adversarial inputs, based on the perturbation necessary to change the classification of the unknown input?
- To what extend do adversarial^{-1} inputs return to their original correct class ('$?_1$' in Figure 5.2) when an adversarial attack is applied?

In order to investigate these questions, in Section 5.1 our approach to conduct the experiments is outlined, followed by the experimental setup, namely the used dataset and image classifier, in Section 5.2. The corresponding results are presented in Section 5.3. Finally, Section 5.4 concludes this chapter.

5.1. Approach

As introduced by the two initial research questions (see Page 52), the experiments were divided into two phases—the detection and the classification phase. The workflow of the detection phase is depicted in Figure 5.3. In preparation to evaluate the properties of adversarial^{-1} inputs, all attacks introduced in Section 3.6, as well as FGSM (see Section 3.5) were used to create adversarial inputs. The adversarial attack on original inputs before they are fed to the proposed defence is referred to as external attack. Following the literature, the external attacks were targeted whenever possible, while the target is randomly chosen for each sample.

Within the defending process, another attack is applied to the given initial input of unknown origin, either original or adversarial. The internal attacks are executed untargeted because the attack was assumed to find the closest decision boundary itself, independent of a given class. Further on, these attacks are named internal attacks, and the resulting manipulated inputs are defined as the internal counterpart of the initial input. In particular, the L_0, L_2, and L_∞-norm distances (see Section 3.2) between the initial input, and its internal counterpart were calculated, based on their respective pixel values. Here, the distance was calculated based on the pixel values, because the available attacks optimise the perturbation with regards to those. Based on the resulting L_p-norm differences a dataset was build, with the origin of an input (original or adversarial) as labels.

In the later experiments, different classifiers were trained on different subsets of the dataset of calculated L_p-norm distances, to differ between initially original and adversarial inputs. Those classifiers are referred to as internal classifiers.

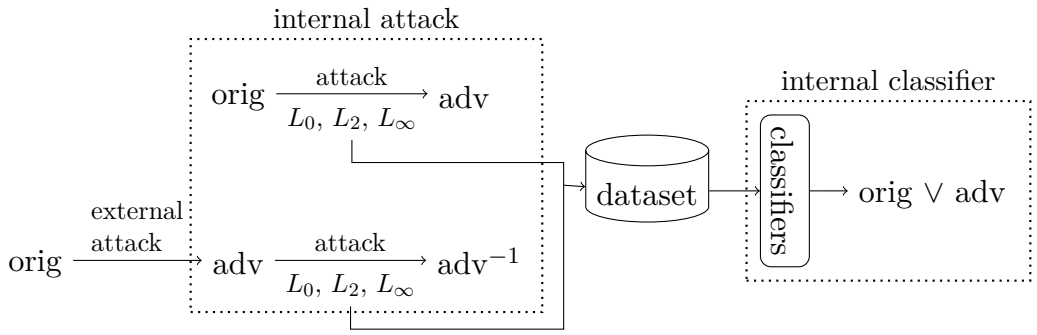


Figure 5.3.: General workflow of the detection stage during training. During training original inputs are transformed by a targeted external attack. Afterwards, original, as well as adversarial inputs are transformed by an untargeted internal attack, and the differences between the input and internal counterpart are calculated. Those differences are used to train a classifier to differentiate between original and adversarial inputs.

Regarding the second initial research question, the adversarial^{-1} inputs were classified by the image classifier, and the classification accuracy concerning the initially correct class was recorded.

5.2. Experimental Setup

In this Chapter, the experiments were conducted on the Cifar-10 [KH09] dataset. It consists of 60,000 images of size 32×32 with 3 colour channels, representing 10 different classes. The classes are 'airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', and 'truck', and some samples as well as the documentation can be found at [Tor]. Each class is equally represented with 6,000 images. The complete dataset was split into 50,000 training and 10,000 test images. For training the model, the training set was further split into 40,000 training and 10,000 validation samples.

During the initial analysis of the properties of adversarial^{-1} and for training the classifiers, adversarial inputs based on the validation set of Cifar-10 were used. The validation set was chosen over the training set because during training the decision boundaries are mainly fitted towards the training set. In a real-world scenario, however, it has to be assumed that an attacker manipulates inputs which are not previously known to the system. When it comes to detecting adversarial inputs, the process was tested on inputs based on the test dataset of Cifar-10.

To process the images a ResNet architecture [He+16] was used. The overall architecture is enlisted in Table 5.1, where the total number of layers is 34. Hence, this specific architecture is referred to as ResNet-34. The structure of the Basic Block (BB) is depicted in Figure 5.4. The input vector $\hat{\mathbf{x}}$ for the block is transformed by a convolutional (conv) layer, followed by batch normalisation (bn) [IS15], ReLU, and another convolutional and batch normalisation layer. A novelty introduced by the ResNet architecture are skip-connections. The input is not only processed by the internal layers but also skips them and is concatenated (plus symbol) to the internal processed information. This allows to preserve certain features of the initial input, which might otherwise get lost during the internal transformations[1]. Finally, another ReLU activation function is applied.

The model was trained with SGD, an initial learning rate of 0.1, a momentum of 0.9, and a weight decay of 0.0005. Besides, the learning rate was reduced by a power of ten after 150 and 250 epochs. Overall, the model was trained for 350 epochs. The loss function for training was the cross-entropy loss. Based on those parameters, a classification accuracy of 95.46% on the test set of Cifar-10 was achieved.

5.3. Results

The first results presented and discussed in Section 5.3.1 are the success rates of the different attacks. Afterwards, in Section 5.3.2 the inspection of the different L_p-distances measured between the input and the internal counterpart is described. Based on these

[1]For further information, it is referred to the original paper by He et al. [He+16].

Table 5.1.: ResNet-34 architecture used for classifying Cifar-10.

layer type	kernel size, channel	padding	stride	output size
Input	-	-	-	$32 \times 32 \times 3$
Convolution	3×3, 64	1×1	1×1	$32 \times 32 \times 64$
BatchNormalisation	-	-	-	$32 \times 32 \times 64$
Basic Block	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	-	-	$32 \times 32 \times 64$
Basic Block	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	-	-	$16 \times 16 \times 128$
Basic Block	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	-	-	$8 \times 8 \times 256$
Basic Block	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	-	-	$4 \times 4 \times 512$
Linear	-	-	-	# classes

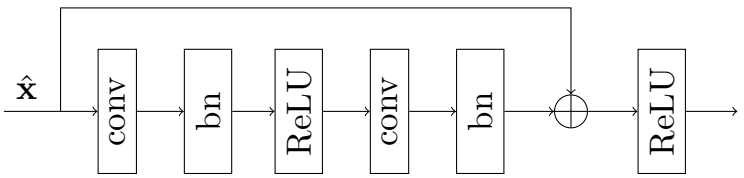


Figure 5.4.: General structure of a Basic Block within the ResNet-34 architecture.

observations, the detection accuracy of adversarial inputs is presented in Section 5.3.3. Also, the internal classifiers were not only trained to distinguish between adversarial and original inputs but also to predict the used external attack. The results and discussion are given in Section 5.3.4. Finally, in Section 5.3.5 the results for the question about how many of the adversarial inputs return to their original true class are outlined and reviewed.

5.3.1. Success Rate of Different Attacks

Adversarial attacks, in general, do not need to be successful in 100% of the cases. In Table 5.2, the success rates of the external adversarial attacks are shown. If for example FGSM was applied, a lower success rate compared to the other attacks was observed. This is important because, in the process of creating the L_p-norm differences dataset, only successful external attacks were considered, and consequently the number of adversarial inputs to apply the internal attack on varies between the different attacks. When training the internal classifiers, this was countered by balancing the observed subset based on the minimum available number of samples over the observed conditions.

Table 5.2.: Success rate of the external attacks in %.

CW	DF	FGSM	JSMA	PGD
95.24	94.82	21.31	95.18	95.24

However, if applying the attacks internally, all attacks have a near or even 100% success rate. The results are shown in Table 5.3. This was a necessity because if the aim is to identify the origin based on the distance between an unknown input and its internal counterpart, a high success rate of the internal attack is essential.

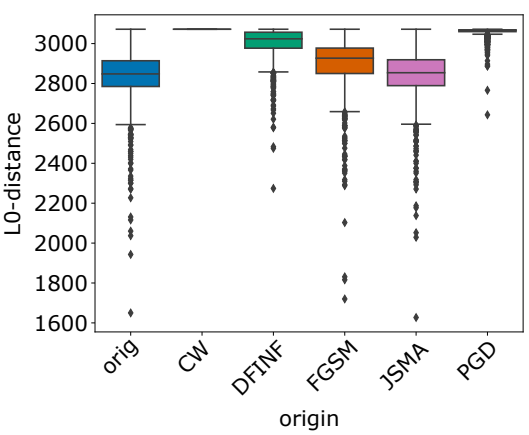
Table 5.3.: Success rate of the internal attacks, depending on the external attack in %. The best mean value is indicated in bold.

		external					
		CW	DF	FGSM	JSMA	PGD	mean
internal	CW	99.94	99.94	100.00	100.00	100.00	99.98
	DF	98.94	99.06	99.72	99.12	98.56	99.08
	FGSM	100.00	100.00	100.00	100.00	100.00	**100.00**
	JSMA	99.94	100.00	100.00	100.00	100.00	99.99
	PGD	99.94	100.00	100.00	100.00	100.00	99.99

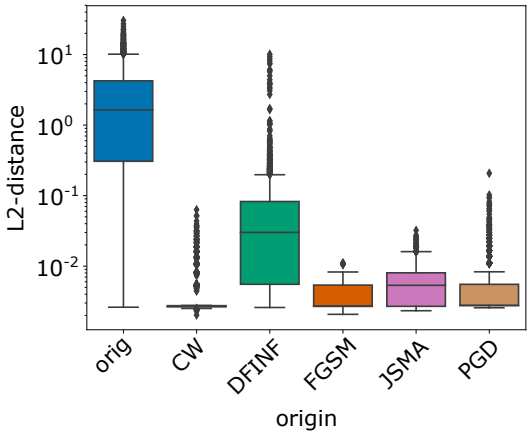
5.3.2. Distribution of Different Distances

In a first examination regarding the properties of adversarial^{-1} inputs, the distribution of the L_0, L_2, and L_∞ distances between the input and its internal counterpart were visualised. In Figure 5.5 the different distributions are shown, where the origin indicates

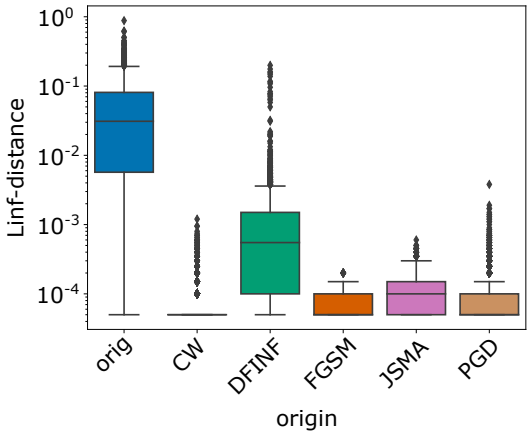
whether the input is an original input, or has previously been created by the given external attack. The internal attack in the given example is FGSM. It is to note that for L_2 and L_∞ the distance distributions are displayed in logarithmic scale.



(a) Distributions for L_0.



(b) Distributions for L_2 in log scale.



(c) Distributions for L_∞ in log scale.

Figure 5.5.: Distributions of different L_p-norms between an input of the given origin, either original (orig) or created by the given adversarial attack, and its internal counterpart created by FGSM.

Especially for L_2 and L_∞ the calculated distances between initially original and adversarial inputs, and their respective internal counterparts differ. Inputs with 'orig' as source have a higher median distance, as well as higher values indicating the second, third, and fourth quartile. In particular, the distributions for inputs of an adversarial origin are located in the lower part of the first quartile of initially original inputs. Intuitively, this made sense because an adversarial attack aims to manipulate the input to be classified differently, with the least possible perturbation. Thereby, an original input is moved into the direction and only slightly across a decision boundary. However, adversarial inputs

are already very close to a decision boundary and need less perturbation to cross the nearest decision boundary.

5.3.3. Detection of Adversarial Inputs

The observations in Section 5.3.2 were based on the validation set of Cifar-10. To evaluate whether these observations could be transferred to previously unknown samples, different internal classifiers were trained based on the L_p-norm difference samples created on the validation set. The performance of those classifiers was then evaluated on samples based on the test set of Cifar-10. Prior to training the classifiers, the features were standardised by $\frac{\mathbf{x}-\mu}{\sigma}$, where $\mathbf{x}$ is the sample to transform, and μ,σ are the mean, and standard deviation for each dimension over all samples. As internal classifiers, a logistic regression classifier (LR), an extra tree classifier (ET), and a k-nearest neighbour classifier (kNN) were chosen. All classifiers are provided by the python library *scikit-learn* [Ped+11]. The training was done with 5-fold cross-validation including a grid search over the parameters given in Table 5.4. It is to mention that the classifiers were only trained on the samples, where the external and internal attack are the same. This assumption was made, because in a real-world scenario the external attack applied by an attacker is unknown, and the defender only has knowledge about the internal attack.

Table 5.4.: Grid search parameters for the different internal classifiers.

LR	solver: L-BFGS
ET	number of estimators: {1,5,10,20,50,100}
kNN	number of neighbors: {1,5,10,20,50,100}, algorithm: auto

In Table 5.5, the detection accuracies for the different internal classifiers are reported, depending on the external and internal attacks. It is to remember that the classifiers were trained on data where the internal and external attack are the same. The corresponding detection accuracies, where the attacks used for training, and the unknown external attack are the same, are indicated in grey. The best detection results for each unknown external attack, as well as the best mean values, are given in bold.

The results in Table 5.5 show that it is possible to predict the origin of an unknown input, based on the L_p-norm difference between the unknown input and its internal counterpart with high accuracy. Furthermore, the detection of unknown inputs generalised over different external attacks. Sometimes, the detection accuracy for the unknown input even improved, compared to the case that the classifier was trained and tested on the same attack. For the parameters which could be controlled, i.e., the internal attack and the internal classifier, LR overall achieved the best mean detection accuracy of 94.08%, when JSMA was used as internal attack.

To put the results into context, in Table 5.6 the best results of the proposed detection method are compared with the reported results of MagNet [MC17], SafetyNet [LIF17], Lee et al. [Lee+18], as well as NIC [Ma+19] as a very recent approach.

Table 5.5.: Detection accuracy in % for different classifiers, depending on the external and internal attack. The grey cells indicate that the classifier has been trained and tested on the same external and internal attack. For each classifier, the best detection results for each external attack, as well as the best mean results, are displayed in bold.

			external					
			CW	DF	FGSM	JSMA	PGD	mean
LR	internal	CW	92.75	84.20	92.88	92.71	92.72	91.05
		DF	86.23	**84.63**	86.55	86.25	86.30	85.99
		FGSM	88.00	85.26	88.55	88.00	88.00	87.56
		JSMA	**97.16**	81.31	**98.04**	**96.90**	**96.97**	**94.08**
		PGD	80.11	77.68	68.30	78.08	79.56	76.75
		mean	**88.85**	82.62	86.86	88.39	88.71	
ET	internal	CW	**98.28**	77.01	98.32	**98.47**	**98.13**	**94.04**
		DF	84.21	81.84	79.97	83.63	84.21	82.77
		FGSM	98.29	73.48	**99.30**	98.45	97.87	93.48
		JSMA	97.97	77.89	97.91	97.42	97.64	93.76
		PGD	93.34	**82.89**	87.85	92.09	93.28	89.89
		mean	**94.42**	78.62	92.67	94.01	94.23	
kNN	internal	CW	**98.19**	**77.23**	**98.04**	**98.34**	**98.06**	**93.97**
		DF	77.10	75.89	68.77	75.90	77.05	74.94
		FGSM	84.74	80.23	82.82	84.19	84.71	83.34
		JSMA	97.68	77.86	98.04	97.22	97.48	93.66
		PGD	76.86	72.32	62.71	74.48	76.31	72.54
		mean	**86.91**	76.70	82.08	86.03	86.72	

Table 5.6.: Reported adversarial detection accuracies in of different defences %, depending on the applied attack. The defences are ordered ascending according to their time of publication.

defence	CW	DF	FGSM	JSMA	PGD
SafetyNet (2017)	-	-	-	-	95.65
MagNet (2017)	93.70	**93.40**	99.90	-	96.00
L_p-Norm (2018)	98.28	84.63	99.30	98.47	98.13
Lee et al.[2] (known) (2018)	97.25	89.13	96.93	97.39	97.53
Lee et al.[2] (unknown) (2018)	97.06	88.94	96.93	96.97	97.26
NIC (2019)	**100.00**	91.00	**100.00**	**100.00**	**100.00**

Taken from Table 5.6, the best results reported in Table 5.5 are comparable or better than previous results, and are still competitive to results of more recent approaches.

5.3.4. Detection of the Different Attacks

In addition to predicting the origin of an unknown input, the classifiers were also trained to predict the external attack used to create an adversarial input in the first place. The motivation here was, that if the prediction of the initial external attack is reliable, specific defences tailored to the external attacks could be employed. Initially, the internal classifiers were trained and tested on disjunct subsets of samples based on the validation set. In Table 5.7 the corresponding confusion matrix of the predictions is reported, when the internal classifier is ET and the attack is FGSM, which led to the overall best accuracy of 77.09%.

Table 5.7.: Confusion matrix for ET, when trained to predict the external adversarial attack. The internal attack is FGSM, and both training and testing have been performed on the validation dataset.

		prediction					
		orig	CW	DF	FGSM	JSMA	PGD
origin	orig	117	0	15	0	0	0
	CW	0	113	0	0	0	0
	DF	9	0	83	4	0	27
	FGSM	0	0	2	86	42	2
	JSMA	0	0	6	50	87	2
	PGD	0	2	13	3	1	113

Even though the prediction was not too accurate, the accuracy is mainly reduced by samples of the origin FGSM and JSMA, which seem to be similar when classified by the L_p-norm differences. However, identifying groups of adversarial attacks may even be sufficient, because in a real-world scenario the external attack might be unknown, but could still be associated with a certain subgroup of attacks. Based on these thoughts, the classifiers were trained on samples of the validation set and evaluated on samples of the test set. The resulting confusion matrix, again for ET as internal classifier and FGSM as attack is displayed in Table 5.8. The overall classification accuracy reduced to 40.92%.

Taking from Table 5.8, more external attacks seem to be similar, when classified based on the L_p-norm differences. For all initially adversarial inputs, a large quantity was predicted to come from PGD as external attack. Besides, CW and DF were confused very often, which could be separated very well when tested on the validation dataset. This observation reinforced the overall assumption that adversarial inputs, based on the validation/training set, and the test set of Cifar-10 might have different properties.

[2]Lee et al. [Lee+18] do not report adversarial detection accuracies themselves. The results reported here are replicated in another study, see Section 7.3.2.

Table 5.8.: Confusion matrix for ET, when trained to predict the external adversarial attack. The internal attack is FGSM, the training samples are based on the validation dataset, and testing is done on samples of the testing dataset.

		prediction					
		orig	CW	DF	FGSM	JSMA	PGD
origin	orig	223	9	124	0	0	2
	CW	0	52	112	0	0	194
	DF	11	50	167	1	1	128
	FGSM	0	0	30	105	51	172
	JSMA	0	16	73	20	16	233
	PGD	0	12	25	3	2	316

However, in some publications, e.g., one by Lee et al. [Lee+18] the training of detection mechanisms is tuned on parts of the test set, which can skew the results.

5.3.5. Classification of Adversarial Inputs

Introduced in Section 3.4, prior and recent defence techniques either focus on detecting adversarial inputs or harden the system to classify adversarial inputs correctly. However, as stated in the second research question (see Page 52), it was to investigate how many of the adversarial inputs return to their original true class, when attacked again internally. In Table 5.9 the results are reported, depending on the external and internal attack.

Table 5.9.: Classification accuracy in % of adversarial inputs after the internal attack, when classified by the image classifier. The best result for each external attack and the best means are displayed in bold.

		external					
		CW	DF	FGSM	JSMA	PGD	mean
internal	CW	86.62	88.50	**51.62**	68.48	79.21	74.89
	DF	87.22	88.44	51.01	68.78	79.79	75.05
	FGSM	89.44	89.11	51.47	**69.75**	80.65	76.08
	JSMA	**89.94**	89.11	51.47	69.68	**81.32**	**76.30**
	PGD	88.03	**89.42**	51.47	69.55	80.18	75.73
	mean	88.25	**88.92**	51.41	69.25	80.23	

Table 5.9 shows that depending on the external and internal attack, a large portion of the adversarial inputs return to their original true class. The best mean results were achieved when JSMA was used as internal attack with a classification accuracy of 76.30% over all external attacks. DF as external attack seemed to be the easiest attack to revert, with a classification accuracy of 88.92% over all internal attacks.

To put these results into context, in Table 5.10 the best results in comparison to two very recent methods, namely Ensemble Diversity [Pan+19] and ComDefend [Jia+19], are presented.

Table 5.10.: Reported classification accuracies in % of different defences, depending on the applied external attack.

			external		
defence	CW	DF	FGSM	JSMA	PGD
Ensemble Diversity (2019)	80.60	-	61.70	43.5	48.40
ComDefend (2019)	89.00	88.00	**86.00**	-	78.00
adversarial^{-1}	**89.94**	**89.42**	51.62	**69.75**	**81.32**

The comparison in Table 5.10 outlines that applying an internal attack reverses adversarial inputs to the original true class to a higher percentage, than other state of the art approaches, excluding FGSM as external attack. Also, the two approaches given as comparison are architectural changes, which could be combined with the approach proposed in this chapter.

5.4. Conclusion

Earlier, as well as recent approaches to defending against adversarial inputs either try to detect those inputs, or harden the model to classify adversarial inputs correctly. In this chapter, a new two-stage approach was introduced to at first detect adversarial inputs, and in a second step restore the original true class.

Because adversarial inputs seem to be inevitable [Sha+19], the basic idea of the detection stage was to attack a given input, either original or adversarial, with an internal attack. Based on the L_0, L_2, and L_∞-norm distances between the unknown input and its internal counterpart, classifiers were trained to predict the origin (original or adversarial) of the given input. Based on the external and internal attack, and the employed classifier, it was possible to differentiate between original and adversarial inputs with an accuracy of up to 99.30%, for the investigated Cifar-10 dataset. Compared to other approaches, the proposed method achieved better or competitive results, even with more recent publications like [Ma+19]. It should also be emphasised that the approach of this Chapter generalised over different external attacks, sometimes even achieving the best results when trained on a different internal attack, than the external attack applied by an adversary.

Besides, in theory, it was possible to detect the external attack or at least certain groups of attacks used to create an adversarial input in the first place. However, when applied on samples, which have been unknown throughout the whole training process, the accuracies reduced and a clear distinction between different external attacks, or at least group of attacks, was not possible anymore.

In the second step, the question about how many of the adversarial inputs return to their original true class when attacked internally was investigated. There it was found

that a portion of up to 89.94% of the adversarial inputs returned to their original class when attacked internally. Comparing these numbers with very recent approaches, the resulting image classification accuracies were higher for four of five external attacks. It is worth to notice that other approaches aiming for a high image classification accuracy are mainly proactive (see Section 3.4). In contrast, the proposed method here is reactive and independent of the architecture to defend. Therefore, it should be possible to combine the internal attacks with different other approaches to further enhance the defence capability of the overall system.

6. Adversarials^{-1} in Speech Recognition: Detection and Defence

Over the last few years, voice-controlled systems such as the Google Home-Systems have become increasingly widespread. Many of those systems incorporate neural networks to process speech data and understand the given commands. One example of such a system is the DeepSpeech project by the Mozilla Foundation, which is based on a research paper by Hannun et al. [Han+14]. However, more recently it was shown, that adversarial attack (see Section 3) are also applicable to automatic speech recognition (ASR) systems (e.g., Carlini & Wagner [CW18], or Alzantot et al. [ABS18]). Thereby, a possible scenario could be that an attacker uploads a manipulated song, e.g., to Youtube. For humans the file sounds like the given song, but it has been manipulated in such a way that it instructs existing smart home devices to transfer money.

In this Chapter, two different attacks are assumed. One is a white-box attack (see Section 3.3.2) proposed by Carlini & Wagner [CW18]. The other one is a black-box attack (see Section 3.3.2) proposed by Alzantot et al. [ABS18].

To defend against those attacks, prior works proposed to use sophisticated preprocessing, e.g., quantization, local smoothing, or down-sampling (e.g., Yang et al. [Yan+18]). Also, Rajaratnam et al. [RSK18] propose to use compression and filtering techniques, as well as using ensembles to identify adversarial images. In their work, they report a maximum precision of 97.3% when using compression/filter techniques, resp. 96.1% when using ensembles. In another recent work, Zeng et al. [Zen+19] proposed to use ensembles of different ASR systems, and based on the similarities of the outputs differentiate between benign and adversarial inputs, resulting in a detection accuracy of 99.78%.

In this Chapter, the defence technique introduced in Section 5 is transferred from the image classification domain to speech recognition. The basic idea is the same, i.e., to attack an unknown input and measure the difference between the input, and the internally manipulated counterpart.

In the remainder of this Chapter, first some necessary preliminaries are introduced, namely speech processing in general (see Section 6.1), the applied adversarial attacks (see Section 6.2), and the investigated dataset (see Section 6.2.3). Afterwards, in Section 6.3 the overall approach is outlined. Focusing on the detection stage in Section 6.3.1 the necessary adaptations for the audio domain are explained, as well as the results are presented and discussed. In Section 6.3.2 the same is done for the classification stage. Finally Section 6.5 concludes this section.

Parts of the results have been published in [WNK20] and have been achieved under the supervision of the master thesis “Adversarials-1 in der Spracherkennung: Erkennung

und Abwehr" (engl. "Adversarials-1 in Speech Recognition: Detection and Defence") of Stefan Niewerth in 2019.

6.1. Speech Recognition

Based on the work of Hannun et al. [Han+14], the Mozilla Foundation developed the DeepSpeech project [Moz], which is used in this section as speech recognition system. The workflow of the system can be separated into three stages—Preprocessing (see Section 6.1.1, the speech recognition itself (see Section 6.1.2, and a final post-processing (see Section 6.1.3).

6.1.1. Preprocessing

When recording a signal, after converting it from analogue to digital, the signal is usually represented in the time domain, storing the specific recorded amplitude for each time step. However, to analyse the signal, for example, to reduce environmental noise or to enhance the spoken words, the signal is transferred to the frequency spectrum. To that aim, the input signal $\mathbf{x}$ is divided into short sequences $\mathbf{x}^{\text{tf}}$ called time frames of usually 20-30 ms length [SS12]. Each sequence $\mathbf{x}^{\text{tf}}$ consists of Tp elements $x_0^{\text{tf}}, x_1^{\text{tf}}, \ldots, x_{\text{Tp}-1}^{\text{tf}}$, which are used to calculate the corresponding frequency spectrum by the discrete fourier transformation (DFT). In general, the DFT returns the same number of frequency coefficients as elements in the input sequence. As an example, a sampling rate of the input signal of 16 kHz is considered, i.e., 16,000 samples per second. Slicing the original input into 25 ms long time frames, each frame consists of $\text{Tp} = 400$ time points. This would lead to 400 frequency coefficients. However, usually 256 or 512 frequency coefficients, depending on the input sampling frequency and the time frame size, are desired. Therefore, zero values are appended to the input sequence until $\text{Tp} = 512$, which is also called zero-padding. The frequency coefficients are then calculated by

$$\hat{x}_k^{\text{tf}} = \sum_{\text{tp}=0}^{\text{Tp}-1} x_{\text{tp}}^{\text{tf}} \cdot \texttt{H}\,(\text{tp}) \cdot e^{-2\pi i \cdot \frac{k\text{tp}}{\text{Tp}}} \;,\; 0 \leq k \leq \text{Tp} - 1 \;, \tag{6.1}$$

where $\texttt{H}$ is a Tp samples long analysis window. In speech recognition, an often applied filtering window is the Hamming window [20003], which is visualised in Figure 6.1 and formally defined as

$$\texttt{H}\,(\text{tp}) = \frac{25}{46} - \frac{21}{46} \cdot \cos\left(\frac{2\pi\text{tp}}{\text{Tp}}\right) ,\; 0 \leq \text{tp} \leq \text{Tp} - 1 \;.$$

Based on the obtained frequency spectrum $\hat{x}_0^{\text{tf}}, \hat{x}_1^{\text{tf}}, \ldots, \hat{x}_{\text{Tp}-1}^{\text{tf}}$, the power spectrum $\mathbf{s}^{\text{tf}} = \mathbf{s}_0^{\text{tf}}, \mathbf{s}_1^{\text{tf}}, \ldots, \mathbf{s}_{\text{Tp}-1}^{\text{tf}}$ is calculated. The power spectrum can be used to quantify the intensity of certain frequencies. For example the ratio between noise and the actual signal can be measured and appropriate filters can be used to reduce the noise. The power spectrum is calculated by

$$\mathbf{s}_j^{\text{tf}} = \frac{1}{\text{Tp}} \left|\hat{x}_j^{\text{tf}}\right|^2 \;,\; 0 \leq j \leq \text{Tp} - 1 \;. \tag{6.2}$$

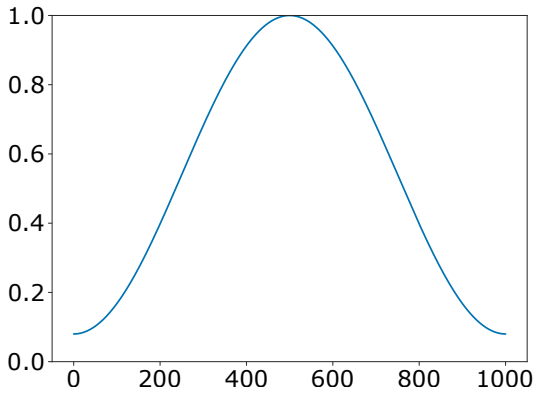


Figure 6.1.: Visualisation of the hamming window function over 1000 steps.

Because humans can not very well distinguish signals of similar frequencies, especially in the higher spectrum, a Mel filter bank (see Figure 6.2) is applied to sum up the power of different frequencies found in the input time frame sequence $\mathbf{x}^{\mathrm{tf}}$. To calculate the Mel filter bank, a lower ν_l and upper frequency ν_u are defined. In the considered example the input is sampled at 16 kHz. Due to the Nyquist–Shannon sampling theorem [Sha49], at a given sampling frequency $\hat{\nu}$ it is possible to restore the frequencies in a given signal up to a maximum of $\hat{\nu}/2$. Here, the lower frequency is set to 0 Hz, and the upper one to 8 kHz, i.e., half of the sampling frequency of the input signal. Both are converted to the Mel scale by

$$Z(\nu) = 1125 \cdot \ln\left(1 + \frac{\nu}{700}\right) \ .$$

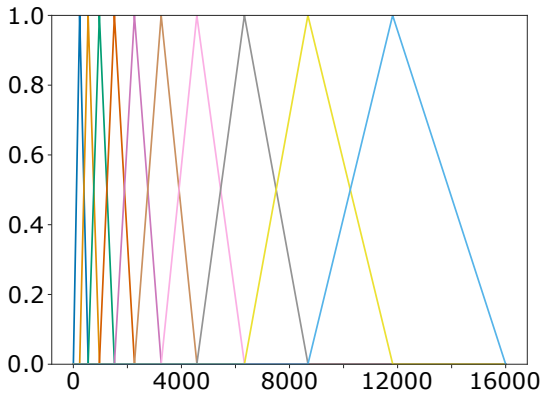


Figure 6.2.: Visualisation of the Mel filter bank with 10 filters over the frequency range 0 Hz to 16 kHz.

Afterwards, `M` evenly spaced points between the lower and upper Mel scaled frequency are determined, where `M` is the number of employed Mel filters. In Figure 6.2 `M` = 10 filters are displayed for better visibility. In DeepSpeech the standard number of filters is

set to 40. Including the previously defined lower and upper frequency, this leads to $\mathtt{M}+2$ Mel scaled frequencies ν_Z, which are reverted to the frequency spectrum by

$$Z^{-1}(\nu_Z) = 700 \cdot \left(e^{\frac{\nu_Z}{1125}} - 1\right) \ ,$$

and rounded to the nearest frequency $\hat{x}_i^{\text{tf}}$ found for the input sequence (see Equation 6.1). The new frequencies ν_r are then used to calculate the filters, formally

$$\text{Mel filter}_r(i) = \begin{cases} 0 & i < \nu_r \\ \frac{i-\nu_r}{\nu_{r+1}-\nu_r} & \nu_r \leq i \leq \nu_{r+1} \\ \frac{\nu_{r+2}-i}{\nu_{r+2}-\nu_{r+1}} & \nu_{r+1} \leq i \leq \nu_{r+2} \\ 0 & i > \nu_{r+2} \end{cases} ,$$

where $i \in [\nu_l,\nu_u]$ are the frequencies by the DFT between the lower and upper frequency limit, in the example 0 and 8 kHz. After having the Mel filter bank applied to the power spectrum (see Equation 6.2), the resulting values are logarithmised.

The last step is to process the logarithmised and Mel transformed sequence $\tilde{\mathbf{x}}^{\text{tf}} = \tilde{x}_0^{\text{tf}},\tilde{x}_1^{\text{tf}},\ldots,\tilde{x}_{\text{Tp}-1}^{\text{tf}}$ with the discrete cosine transform defined as

$$x_j^{\text{tf}} = \sum_{\text{tp}=0}^{\text{Tp}-1} \tilde{x}_{\text{tp}}^{\text{tf}} \cos\left[\frac{\pi}{\text{Tp}}\left(\text{tp}+\frac{1}{2}\right)j\right] \ , \ 0 \leq j \leq \text{Tp}-1 \ .$$

The resulting coefficients are called Mel Frequency Cepstral Coefficients (MFCCs). To suppress environmental noise in the higher frequency spectrum and because the human speech is usually in the lower frequency spectrum of up to 4 kHz, only the first 13 MFCCs are used as input $\mathbf{x}$ for the neural network to process.

6.1.2. Recognition with DeepSpeech

When processing time-dependent inputs, like speech, RNNs have shown to be very effective. In contrast to feed-forward networks (see Section 2.3), RNNs have connections between nodes in both the same and earlier layers. The input for such networks is usually a time frame, in particular, the calculated MFCC outlined in Section 6.1.1, instead of the whole time series. The recurrent connections between the internal nodes of the model allow inputs of earlier time frames at time $\text{tf}-1$ to influence the calculations of later time frames at time tf. This leads to some kind of memory, as earlier observations influence later ones. DeepSpeech integrates a bidirectional neural network, meaning that not only earlier time frames influence the calculation of time frame tf, but also later time frames at time $\text{tf}+1$. This is achieved by feeding the original input $\mathbf{x}$ in reversed order into the model. Therefore, access to the whole sequence is assumed during training and inference. Overall, the DeepSpeech model is comprised of five hidden layers, where the fourth layer is the bidirectional one. At this point it should be noted that in the publication, Hannun

et al. [Han+14] do not use long-short term memory cells (LSTMs)[1] in the recurrent layer because of computational efficiency, whereas in the github implementation [Moz] LSTMs are employed.

For each time frame tf, the output $\hat{\mathbf{y}}^{\text{tf}}$ of the model is a probability distribution $\mathbb{P}\left(c^{\text{tf}}|\mathbf{x}\right)$, which is dependent on the full input sequence $\mathbf{x}$, and where $c^{\text{tf}} \in \{a,b,c,\ldots,z,\text{␣},',\Lambda\}$ is the alphabet of accepted symbols. The complete output of the model is a sequence of those probability distributions, where the symbols with the highest probability are considered as label for the given time frame. The symbols '␣' and ' in the dictionary represent the meaning 'space', and 'apostrophe', while 'Λ', referred to as 'blank' symbolises a non-existing symbol, e.g., silence in a recorded audio file.

When dealing with speech, one problem arising during the labelling of training data is that different people pronounce the same word differently. Therefore, each audio file would have to be analysed separately, and the correct symbol would have to be aligned to each of the time frames. This is, however, very time consuming and thus not practical for large datasets. A loss function which is independent of the correct alignment between the audio samples and the transcribed sentences is the connectionist temporal classification (CTC) [Gra+06]. As mentioned, the output of the neural network are probabilities of characters for each time frame. Considering the exemplary sequence $\varsigma = a\ a\ b\ \Lambda\ \Lambda\ b$ of symbols with the highest probability for the corresponding time frame. At first, CTC removes all sequentially duplicated symbols. In the example $a\ a$ is shortened to a. Afterwards, all Λ symbols are removed, leading to the phrase $\rho = abb$. For a more detailed view on CTC see [Han17], and for further information on the training process of DeepSpeech see [Han+14].

6.1.3. Postprocessing

Based on the calculated character probabilities, words and sentences are composed, which are compared to a language model. The idea of a language model is to learn a) the correct spelling of single words and b) a probability distribution over sequences of words in a sentence. The advantage of language models is that they can be trained fast and unsupervised, i.e., no human labelling previously to the training process is required, on a very large corpus of words in a reasonable time. Based on the comparison between the predicted words and word sequences, small errors during the character classification can be detected and fixed.

[1] LSTMs are a special structure of RNNs, which have shown to be more efficient than plain RNNs. For further information, I refer to the original paper by Hochreiter and Schmidhuber [HS97].

6.2. Adversarial Attacks on Speech Recognition

The general formulation of adversarial inputs in speech recognition is similar to that in image processing, i.e.,

$$\begin{aligned} \text{minimise} \quad & \mathcal{D}(\mathbf{x},\mathbf{x}+\boldsymbol{\delta}) \\ \text{such that} \quad & f(\mathbf{x}+\boldsymbol{\delta}) = \mathbf{t} \\ & \mathbf{x}+\boldsymbol{\delta} \in \left[-2^{15}, 2^{15}\right]^I \ . \end{aligned}$$

However, the target of an attack here is a sequence of target symbols $\mathbf{t}$, instead of one single target class, and the range of the inputs is $\pm\, 2^{15}$ with an arbitrary input dimension I. But the main difference are the distance metrics $\mathcal{D}$ used. Carlini & Wagner [CW18] for example use Decibels (dB) to measure the relative loudness of the distortion $\boldsymbol{\delta}$, compared to the input signal $\mathbf{x}$, formally

$$\begin{aligned} dB(\mathbf{x}) &= \max_i 20 \cdot \log_{10}(x_i) \\ dB_{\mathbf{x}}(\boldsymbol{\delta}) &= dB(\boldsymbol{\delta}) - dB(\mathbf{x}) \ . \end{aligned} \tag{6.3}$$

In this Chapter, two different attacks are employed, in particular a white-box attack explained in Section 6.2.1 and a black-box attack outlined in Section 6.2.2.

6.2.1. White-box attack

As a white-box attack, Carlini & Wagner [CW18] propose to solve the following formulation of the problem

$$\begin{aligned} \text{minimise} \quad & |\boldsymbol{\delta}|_2^2 + b \cdot \ell(\mathbf{x}+\boldsymbol{\delta},\mathbf{t}) \\ \text{such that} \quad & \mathrm{dB}_{\mathbf{x}}(\boldsymbol{\delta}) \leq \varepsilon \end{aligned} \ , \tag{6.4}$$

where in an initial formulation $\ell(\mathbf{x}',\mathbf{t}) = \mathrm{CTC}(\mathbf{x}',\mathbf{t})$ is considered. The parameter ε indicates the maximal amount of perturbation allowed, and is decreased iteratively. Also the parameter b is used to balance between the importance to be adversarial and the importance to be close to the original input. For this to be a white-box attack, it is important to note that Carlini & Wagner were able to differentiate through the whole classifier, i.e., starting from the input audio sample, through the MFCC process, and the neural network, to the final loss.

However, the problem of using plain CTC is that the perturbation is not guaranteed to be minimal. This is because CTC always considers the whole transcription of the input sequence. But if the sequence 'left' which is already transcribed as 'lefd' is considered, minimising CTC would lead to also making the 'l' even more 'l'-like, while only the 'd' has to be transcribed as 't'. To solve this problem, Carlini & Wagner formulate a loss function which does not decrease further if the output character with the highest probability already matches the target character t, formally

$$\ell(\mathbf{y},t) = \max\left(y_t - \max_{t' \neq t} y_{t'}, 0\right) \ . \tag{6.5}$$

In the context of audio signals, they consider an alignment ζ of probabilities $\mathbf{y}$ for a given phrase ρ. Hence, the extended loss function over all time frames tf is

$$\mathcal{L}(\mathbf{x},\zeta) = \sum_{\text{tf}} \ell\left(f\left(\mathbf{x}^{\text{tf}}\right),\zeta^{\text{tf}}\right) \ , \tag{6.6}$$

with ℓ being the loss defined in Equation 6.5. Based on the observation that certain characters are more difficult to transcribe than other characters, they further reformulate Equation 6.4 to consider a different b for each time frame

$$\begin{aligned} &\text{minimise} \quad |\boldsymbol{\delta}|_2^2 + \sum_{\text{tf}} \left[b^{\text{tf}} \cdot \mathcal{L}^{\text{tf}}\left(\mathbf{x}+\boldsymbol{\delta},\zeta^{\text{tf}}\right)\right] \\ &\text{such that} \quad \text{dB}_{\mathbf{x}}(\boldsymbol{\delta}) < \varepsilon \end{aligned} \tag{6.7}$$

where $\mathcal{L}^{\text{tf}}\left(\mathbf{x},\zeta^{\text{tf}}\right)$ is the loss defined in Equation 6.6, but evaluated for one specific time frame tf, i.e., one specific character transcription. The necessary alignment ζ for this calculation is derived by solving the initial Equation 6.4. Based on this approach, Carlini & Wagner report a success rate of 100% for their attack.

6.2.2. Black-box attack

As the second attack in this section, the black-box attack proposed by Alzantot et al. [ABS18] is used, in which they basically employed a $(\mu+\lambda)$-evolutionary strategy. In this context, μ is referred to as 'parents' or 'current population'. In general, evolutionary strategies work iteratively as shown in Figure 6.3.

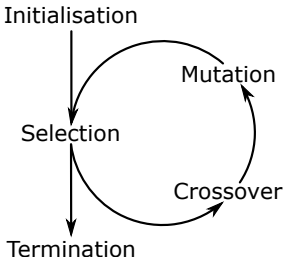


Figure 6.3.: General process of an evolutionary algorithm, starting with the initialisation, followed iteratively by selection, crossover, and mutation, until a termination criterion is reached.

In the initialisation phase, they instantiate μ (default: 20) audio samples referred to as the current population, by randomly perturbing the second byte of each input time point within a certain range (default: $\pm 2^8 = \pm 256$) with a small probability (default: 0.0005). Alzantot et al. [ABS18] only perturb the second byte, representing the lower frequencies of the input signal, because they argue that perturbations to those frequencies are less perceptible by humans. For each of the μ instances, they calculate the respective fitness, i.e., the probability of the individuals being classified as the desired target sequence. The instances with the highest fitness values (`e`, default: 2) are selected and stored as elitists

for the next iteration. If one of the individuals is already classified as the desired sequence, the algorithm stops. Otherwise, based on the current population $\lambda = \mu - \mathsf{e}$ new instances are created by crossover, referred to as children. Alzantot et al. [ABS18] implement crossover by randomly selecting two individuals, where the selection is weighted based on their fitness. Afterwards, for each time point with a probability of 0.5 they either select the value of the first or the second individual to be used as the value for the new child.

Following the circle of evolutionary algorithms, the individuals created by crossover are again mutated by randomly perturbing the second byte of each time point with a small probability. The λ offspring after crossover and mutation, plus the e elitists selected previously, build the new population for the next iteration of the evolutionary algorithm.

This process is repeated until one of the instances is classified as the desired target sequence, or a maximum number of iterations is exceeded. Alzantot et al. [ABS18] report a success rate of 87%.

6.2.3. Dataset

In this Chapter, a pre-trained DeepSpeech model is used, which can be downloaded on the corresponding github repository [Moz]. In accordance to the literature, e.g., [ABS18], an old version [Kag] of the Speech Command Dataset [War18] is employed, consisting of 64,727 one-second long utterances of 30 short words, by thousands of different people. Based on the documentation of the SpeechCommand dataset [War18] and the literature [ABS18], the following ten words are assumed as labels for the classification: 'down', 'go', 'left', 'no', 'off', 'on', 'right', 'stop', 'up', and 'yes'. Tested on words of these categories, the DeepSpeech model achieves an accuracy of 86.51%. Aside from those ten words in the dataset, two additional classes exist, 'silence' and 'unknown', which are omitted in the experiments.

6.3. Approach

The general approach in this chapter is similar to the process introduced in Section 5 for image classification. In the first phase, adversarial inputs are to be detected by applying internal attacks and tracking the differences between the input and its internal counterpart. The adaptations made to the detection phase are outlined in Section 6.3.1. The second phase aims to restore the original true class of adversarial inputs. Here, larger adaptations are necessary, which are explained in Section 6.3.2.

6.3.1. Detection

The general workflow for the detection phase is depicted in Figure 6.4. At first, an initial adversarial dataset is created by randomly sampling sequences from the 10 words present in the SpeechCommand dataset (see Section 6.2.3) and manipulating them to be classified as each of the other nine classes. Afterwards, original and the previously created adversarial inputs are attacked internally, and the differences between the input and its internal counterpart are calculated. In contrast to image classification, no targeted

attacks can be executed here. Therefore, we manipulate each unknown input to be classified as each of the other nine possible classes to build the dataset the internal classifiers are trained on. Another adaptation made is the calculated distances. The raw pixels in image classification would correspond to the raw audio file in speech recognition. Therefore, we calculate the L_0, L_2, and L_∞-norm distances between the raw audio file and the internally manipulated audio file. However, DeepSpeech itself is fed with the MFCC after preprocessing the raw audio file (see Section 6.1) to classify the input sequence. Hence, additionally the L_1, L_2, and L_∞-norm distances between the original and manipulated MFCC are considered. For the MFCC the L_1-norm performs better than the L_0 norm, and is used instead. It is assumed, that this is due to the binning process during calculating the MFCC where the probability that at least one value changes within each bin is high. The L_1-norm is calculated according to Equation 3.3 on Page 23. Thereby, six different values quantifying the induced perturbation are recorded, which are stored to a dataset.

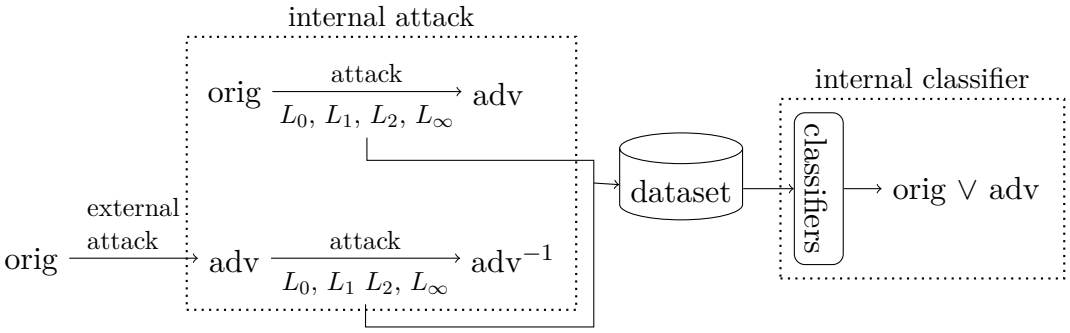


Figure 6.4.: General workflow of the detection stage during training. During training original inputs are transformed by a targeted external attack. Afterwards, original as well as adversarial inputs are transformed by an untargeted internal attack, and the differences between the input and internal counterpart are calculated. Those differences are used to train a classifier to differentiate between original and adversarial inputs.

The internal classifiers used in this chapter are an ANN, kNN, and a decision tree classifier (DT). The ANN is comprised of three layers with 4, 8, and 16 nodes each, and ReLU as activation function. For kNN k is set to 10. All classifier implementations are based on the python machine learning library scikit-learn [Ped+11]. The corresponding results for detecting adversarial inputs are presented in Section 6.4.2.

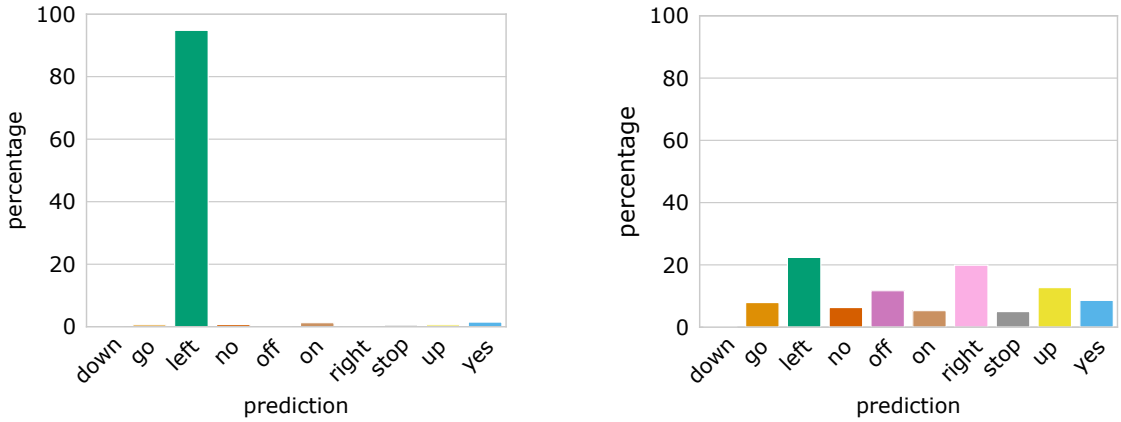
6.3.2. Classification

Regarding the classification stage, it is not directly possible to apply the approach proposed in Section 5.3.5. In the image domain, an untargeted attack is used as the internal attack, and the subsequently predicted class is assumed to be the originally correct one. However, this is not possible in the speech domain. When transferring single

characters untargeted to be classified as random other characters, the resulting sequence would not resemble a known word. As a result, the internally transformed sequence would either be rejected or corrected by the postprocessing step. Hence, it is always necessary to define a certain target sequence which should be detected, i.e., apply a targeted attack.

In a first test, adversarial inputs are transferred internally towards all other nine possible classes, and for each manipulation, the distances are calculated. The assumption is, that the target class which requires the least perturbation applied might be the original true sequence. However, following this approach the prediction accuracy is around 11.1%.

As an alternative, the classifiers of the detection stage were extended to predict the initial correct sequence for adversarial inputs. If the target sequence of the internal attack matches the original true sequence, the prediction accuracy for the original class is very high. This is shown in Figure 6.5 for the instances of the original sequence 'left', which were transferred to all other nine possible classes. When the target of the internal attack is 'left', kNN predicts for 94.8% of these instances, that the original sequence is 'left' (see Figure 6.5a). In comparison, if the target of the internal attack is the sequence 'down', the predictions for the original sequence are distributed among all possible classes (see Figure 6.5b). Overall, for kNN the highest divergence between the prediction distributions of the true source sequence and all other sequences was observed.



(a) 'left' as target

(b) 'down' as target

Figure 6.5.: Distribution of the predictions in % over all possible labels, depending on the target of the internal attack. The original label is 'left'.

The conclusions derived from Figure 6.5 are possible, because the whole dataset was used to create the displayed distributions. In a real-world scenario, however, only one single input is available. To still apply this statistical evaluation, an adversarial input was transformed ten times to each of the nine other words. Here it is important that the transformations had to be different. Therefore Alzantot's attack was applied internally which includes certain randomness. For each of the different transformations, the original source sequence was predicted. Thereby, it was possible to calculate similar distributions as shown in Figure 6.5. Finally, the number of predictions for each of the possible classes are summed up, and the sequence that was most often predicted as the original source

sequence is assumed as the original true classification. The classification results achieved by this method are outlined in Section 6.4.2.

6.4. Results

Based on the experiments described in Section 6.3, in Section 6.4.1 the corresponding results for the detection phase are given, whereas in Section 6.4.2 the results for classifying adversarial inputs are outlined.

6.4.1. Detection of Adversarial Inputs

The results for the detection stage are presented in Table 6.1. The column 'external' indicates the external adversarial attack, which the defender can not control, while column 'internal' indicates the internal adversarial attack. The column 'sensitivity' indicates the percentage of correctly classified benign inputs, while 'specificity' indicates the percentage of correctly classified adversarial inputs. The column 'prediction' gives the overall percentage of correctly classified inputs over all benign and adversarial inputs.

Table 6.1.: Detection accuracy of adversarial (specificity) and benign (sensitivity) inputs.

external	internal	classifier	sensitivity in %	specificity in %	prediction in %
Alzantot	Alzantot	ANN	17.4	97.7	62.8
		kNN	89.7	99.7	95.3
		DT	99.8	99.9	99.8
	Carlini	ANN	67.1	89.6	79.3
		kNN	97.5	98.3	98.0
		DT	**99.9**	97.7	98.7
Carlini	Alzantot	ANN	18.9	98.1	63.5
		kNN	88.4	**100.0**	94.9
		DT	99.8	99.9	**99.9**
	Carlini	ANN	62.2	94.0	79.4
		kNN	92.8	99.4	96.4
		DT	94.2	**100.0**	97.3

Table 6.1 shows that the ANN performs worse, especially regarding the sensitivity, compared to the other two internal classifiers. Presumably, it is possible to increase the performance of the ANN by further investigations of the model architecture or by modifying the training process, e.g., by increasing the amount of training data. However, since the overall good performance, in particular of the DT, there was no urgent necessity to further investigate how to improve the performance of the ANN. This could, however, be investigated in further research.

Regarding the specificity, two special cases can be observed in which a 100% correct identification of adversarial inputs is reported. However, the corresponding sensitivity is

comparatively low. Therefore, it is assumed that the classifier learned an over-specificity, i.e., the classifier classifies more inputs as adversarial as it is necessary. Some people argue, that depending on the context this could be considered as a good behaviour [GWK17], but in general, a good specificity, as well as a good sensitivity is desirable.

Regarding the overall prediction accuracy, DT performs best regardless of the external and internal attack. Besides, 1% to 2% better results are reported, when applying Alzantot's attack internally, compared to the results when using Carlini's attack. Overall, using the proposed technique on audio files, a prediction accuracy of 99.8%, resp. 99.9% is achieved, depending on the attack applied by an adversary.

To put these number into context, in Table 6.2 they are compared to the reported detection results of other defence techniques. Rajaratnam et al. [RSK18] proposed to use two different approaches, where the first one was to use different preprocessing methods and compare the prediction of the raw input with the prediction of the preprocessed input. If the predictions did not align, the input was declared as adversarial. For Band-Pass Filtering, they reported the best precision, i.e., detection of adversarial inputs, of 97.3%. However, the corresponding recall was only 40.6%. Considering F_1 as a quality measure, they reported the highest value of 0.91 for Speex Compression. Based on those results they proposed to employ different compression methods as ensembles, which led to a highest reported precision of 96.1% with a recall of 88.1%, while the best F_1 is reported to be 0.924. Similarly, Zeng et al. [Zen+19] proposed to use different ASR-systems and compare their outputs. Based on the distance between the predicted sequences they classified the input as adversarial or benign. As adversarial attacks they used a black-box attack proposed by Taori et al. [Tao+19] and the attack proposed by Carlini & Wagner [CW18]. Trained and tested on benign and adversarial inputs they reported their best detection accuracy of 99.88%.

Table 6.2.: Reported success measures for different defence strategies in %. Rajaratnam et al. reported the precision, i.e., the percentage of correctly detected adversarial samples over all samples classified as adversarial. These values are used in the comparison since the aim is to detect adversarial inputs. The reported results for Zeng et al. [Zen+19] and the defence proposed in this chapter are the detections accuracies over adversarial and benign samples.

defence		white-box	black-box	
		Carlini	Taori	Alzantot
Rajaratnam	compression	-	-	97.30
	ensemble	-	-	96.10
Zeng et al. [Zen+19]	ensemble	**99.88**	**99.88**	-
L_p-norm		99.85	-	**99.85**

Considering the most similar results, the adversarial detection based on the L_p-norm is competitive when Carlini & Wagner's attack is used externally. When applying Alzantot et al.'s attack, the detection accuracy based on the L_p-norm is higher than the other reported values. However, the other proposed methods only detect adversarial inputs

and discard them. The approach proposed in this chapter is also used to restore the correct label of an adversarial input.

6.4.2. Classification of Adversarial Inputs

Similar to the detection stage, an ANN, kNN, and a DT were used to predict the original sequence. In contrast to the detection stage, for this task kNN achieved the best results, as shown in Table 6.3. Over a testset of 259 adversarial words, the correct label was restored in 67.6% of the cases, by the process introduced in Section 6.3.2.

Table 6.3.: Correct classification of adversarial inputs after the internal attack.

	NN	kNN	DT
accuracy	9.7%	**67.6%**	33.6%

6.5. Conclusion

In this chapter the concept proposed in Chapter 5 was transferred from image classification to speech recognition. To adapt the detection stage to the speech domain, not only the L_0-, L_2-, and L_∞-norm distances of the raw audio input and its internal counterpart are calculated, but also the L_1-, L_2-, and L_∞-norm differences of the MFCC. Based on these six values an ANN, kNN, and a DT were trained to distinguish between benign and adversarial inputs. The results show that a DT is best suited for this task, as well as using the attack of Alzantot et al. [ABS18] as an internal attack. On an independent test set an overall detection accuracy of 99.8%, resp. 99.9% was achieved, depending on the attack used by the adversary. These detection accuracies itself are competitive or better than other reported values in the literature (e.g., [RSK18; Zen+19]).

However, in addition to identifying adversarial inputs, in a second step the process proposed to restore the original class of adversarial inputs (see Section 5.3.5) was adapted to the speech domain as well. Similar to the detection stage, different classifiers were trained to predict the source sequence of the adversarial inputs. If the target sequence of the internal attack was the same as the original source sequence, the prediction accuracy of the internal classifiers are higher than if the internal target sequence is different from the original one. Based on this observation, the reformation process was adapted to transform an adversarial input ten times to each of the other nine words, and for each transformation, the original source sequence was predicted. Evaluating this process shows that it is possible to restore the source sequence successfully in 67.6% of the cases on an independent test set of adversarial inputs. To the best of my knowledge, this is the first method to restore the original sequence for an adversarial input in the audio domain, in such a manner.

However, a general drawback of the proposed process is that it requires internal adversarial attacks, which can take a non-negligible amount of time. Especially in the classification stage, where it is necessary to apply $C \cdot (C-1) = C^2 - C$ attacks to an

identified adversarial input, where C is the number of accepted words for the observed dataset. As the corpus of accepted sequences grows, like in multi-purpose ASR-systems, e.g., in mobile phones, the number of internal attacks required increases quadratically with the size of the corpus.

To counter this drawback, it might be an option to randomly sample target labels from the possible acceptable sequences for the internal attack. This is done until a meaningful distribution like in Figure 6.5a emerges.

7. Adversarials^{-1} for Hidden Layers

In Chapter 5 and 6 the L_p-norm difference between an unknown input and its internal counterpart was used to detect adversarial inputs. However, during the development of the method, the question arose whether it is possible to create inputs which are classified correctly by a system without any adversarial detection mechanism, but are detected as adversarial, and thereby classified wrong by the proposed defence. The intuition was, that the internal classifier creates a margin around the decision boundary where samples inside this margin are classified as adversarial, while samples outside the margin are classified as original. This is depicted in Figure 7.1. Based on an original input, a new input (question mark) is created, which is very near to the decision boundary of the undefended system but still classified correctly. If the internal attack is applied, the internal counterpart (cross) of the input is within the adversarial area. However, the distance between the input and its internal counterpart is similar, as if the adversarial input (cross) would be internally attacked and its internal counterpart would be the input depicted as a question mark. Consequently, the input (question mark) is wrongly classified as an adversarial input, and thereof the wrong class prediction is given.

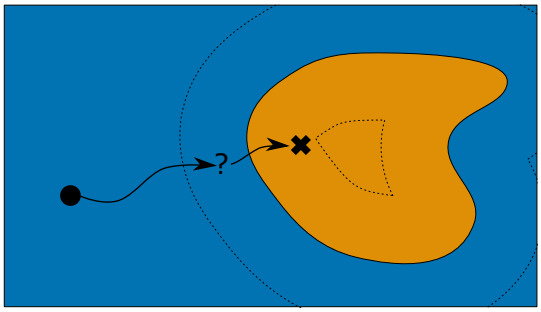


Figure 7.1.: The internal classifier creates a margin (dashed lines) around the decision boundary of the image classification model. The unknown sample (question mark) would be classified correctly by the image classifier. However, the defence would detect it as adversarial input, because it behaves similar to genuine adversarial inputs, and by the internal attack change the image classification to be wrong.

In a preliminary test, a recent black-box attack proposed by Guo et al. [Guo+19] was applied to investigate this question further. The simplest version of the attack iterates randomly over all pixels of the input. For each pixel a certain amount of perturbation α is added, and if the predicted probability of the original class already decreases, the algorithm proceeds with the next pixel. Otherwise, the original value of the observed pixel

is decreased by α. This process is repeated until either the input is classified differently to the original classification, or a limit of iterations is exceeded. Employing the attack, inputs were detected as adversarial a few iteration steps earlier, before the classification of the input changes, i.e., the inputs are legitimate adversarial inputs. This observation led us to the question if it is possible to identify the direction in which an input crosses the decision boundary when attacked internally. So whether an input moves from the original area into an adversarial island or vice versa.

In Section 7.1 the overall approach for the experiments in this chapter is outlined, followed by the experimental setup in Section 7.2. The results for the different experiments are given in Section 7.3 and Section 7.4 concludes this chapter.

7.1. Approach

As described in the introduction, the research question was extended to identify whether a given sample crosses the decision boundary from the original area towards an adversarial island or vice versa, when attacked internally. Therefore, it was necessary to further specify the position of a given sample or in particular the change in position by the internal attack. For this purpose, the outputs of the hidden layers were added to the considerations. In this chapter we use the same ResNet-34 architecture introduced in Section 5.2, and modified it to return the output values after each of the four internal BBs. The adapted overall workflow is depicted in Figure 7.2.

An unknown input $\mathbf{x}$ is processed by the ResNet-34 model and the calculated outputs after the four BBs are recorded. Since the dimensionality of the raw outputs is rather large, e.g., the output dimension of the first Basic Block is $32 \cdot 32 \cdot 64 = 65{,}536$, they are compressed. Initially, the dimensionality was tried to be reduced with different forms of AEs, but the training process is tedious and according to the reconstruction error, the results were not convincing. Lee et al. [Lee+18] proposed to calculate the channel-wise mean of the outputs to reduce the dimensionality. Thereby, the observed output dimensionality for the four BBs was reduced to be 64, 128, 256, and 512 respectively.

Based on the compressed intermediate outputs, the Mahalanobis distance [Mah36] (see Section 3.2) is evaluated. The mean values and the inverted covariance matrices necessary to implement the Mahalanobis distance were calculate based on the intermediate outputs of images used for training the image classifier, for a given class. This is schematically shown in Figure 7.3.

Returning to Figure 7.2, the same process described for the unknown input is applied to its internal counterpart $\mathbf{x}'$, i.e., the intermediate outputs are evaluated and compressed. However, the Mahalanobis distance is calculated towards the class associated with the mean and covariance values which led to the closest distance when processing the initial input. The four differences between the calculated Mahalanobis distances for the initial input and its internal counterpart are stored to a dataset which is used to train classifiers to determine if the initial input was original or adversarial.

The overall intuition to use the Mahalanobis distance was that benign samples are close to other clean samples which have been seen by the model during training. Therefore,

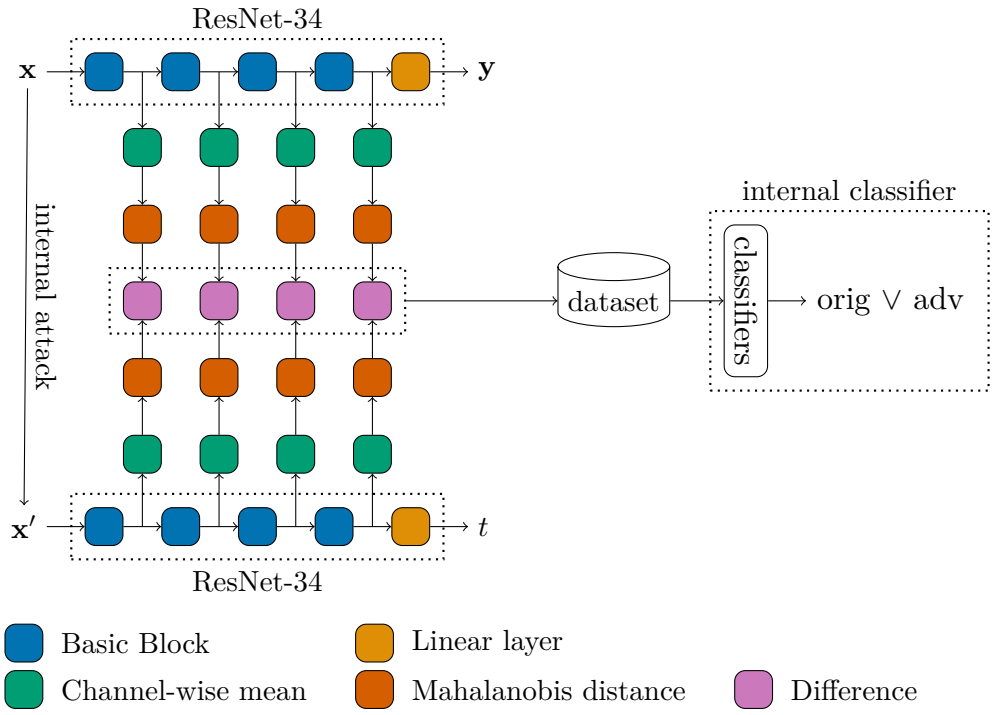


Figure 7.2.: Schematic figure of the overall workflow in this section. For an unknown input $\mathbf{x}$, as well as its internal counterpart $\mathbf{x}'$, the outputs of the Basic Blocks are calculated and compressed. Afterwards, the Mahalanobis distances are calculated, and the difference between them is taken. The resulting four distance differences are stored to a dataset to train and test the internal classifiers.

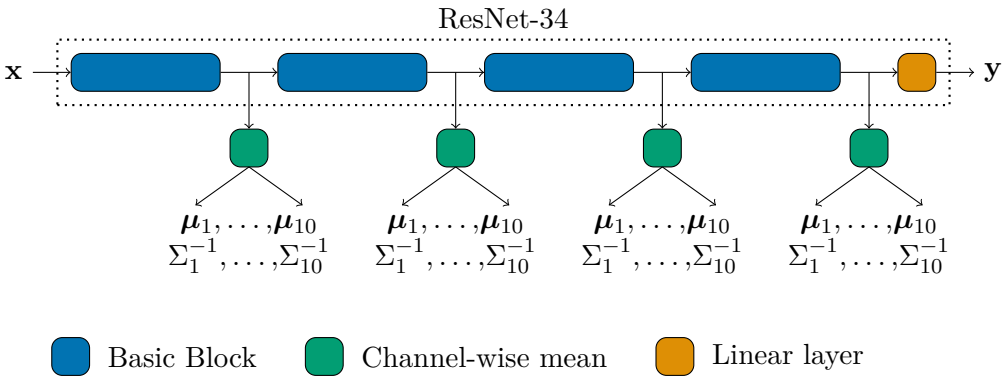


Figure 7.3.: Schematic figure of the outputs the extended ResNet-34 architecture generates. Here, $\mathbf{x}$ are samples from the training set which was used to train the image classifier. The intermediate outputs after each Basic Block are compressed by calculating the channel-wise mean. Based on the compressed values, the ground truth label dependent mean μ and inverted covariance matrices Σ^{-1} are determined. Those are needed for the later Mahalanobis distance calculation.

when processing an initially original input, the corresponding Mahalanobis distance is assumed to be small. By applying the internal attack, the sample was assumed to move "away", i.e., the Mahalanobis distance increases concerning the initially found cluster. Consequently, the difference between the Mahalanobis distance previous and after the internal attack should be negative. Vice versa, observing an initially adversarial input, the initial Mahalanobis distance was assumed to be large. However, after the internal attack, the distance was assumed to decrease towards the initially found cluster, because the initially adversarial input moves out of the adversarial island and towards the centre of the surrounding benign area. The difference between the two distances should, therefore, be positive.

The first experiments were conducted to evaluate this assumption by applying the black-box attack proposed by Guo et al. [Guo+19] to an initially clean sample. After each iteration of the attack, the Mahalanobis distance was calculated and the difference between the distances of consecutive steps built. It was assumed, that the difference will only become positive in the last iteration step, i.e., when the classification of the initially original input changes. The corresponding results are shown in Section 7.3.1. Based on the obtained observations, the distance differences are used to detect adversarial inputs in general, with the results outlined in Section 7.3.2.

In addition to detecting adversarial inputs, Lee et al. [Lee+18] proposed to use Mahalanobis distance to detect out-of-distribution data (OOD) in general. The idea of OOD detection is, that samples, e.g., adversarial inputs, differ from the observed distribution of the clean training samples. The intuition is that samples which have no direct connection to the dataset the image classifier was trained on, are located somewhere between original inputs, which are close to the clusters, and adversarial inputs, which

are close to the assumed decision boundary. The results of this investigation are outlined in Section 7.3.4.

Following the observations made in adversarial and OOD detection, in the last set of experiments the internal classifiers were trained to differentiate between original, adversarial, and OOD data. The gained insights could be used to further describe the decision space of image classifiers or in general the decision-making process of neural networks. To the best of my knowledge, this problem has not been investigated before, and the corresponding results are summarised in Section 7.3.5. d

7.2. Experimental setup

As indicated in Section 7.1, the experiments in this chapter were conducted using the ResNet-34 architecture (see Section 5.2). The image classifier was trained on Cifar-10 (see Section 5.2), and the adversarial attacks used as external, as well as internal attacks, are CW, DF, FGSM, JSMA, and PGD (see Sections 3.6 and 3.5).

For the OOD detection experiments, the street view house number dataset (SVHN) [Net+11] was used, as it has the same spacial dimension as Cifar-10, i.e., 32×32 pixels with 3 colour channels. The dataset consists of 73,257 samples for training and 26,032 samples for testing[1], where only samples of the test set were used for the experiments. Each sample displays a number between 0 and 9 taken from house number plates.

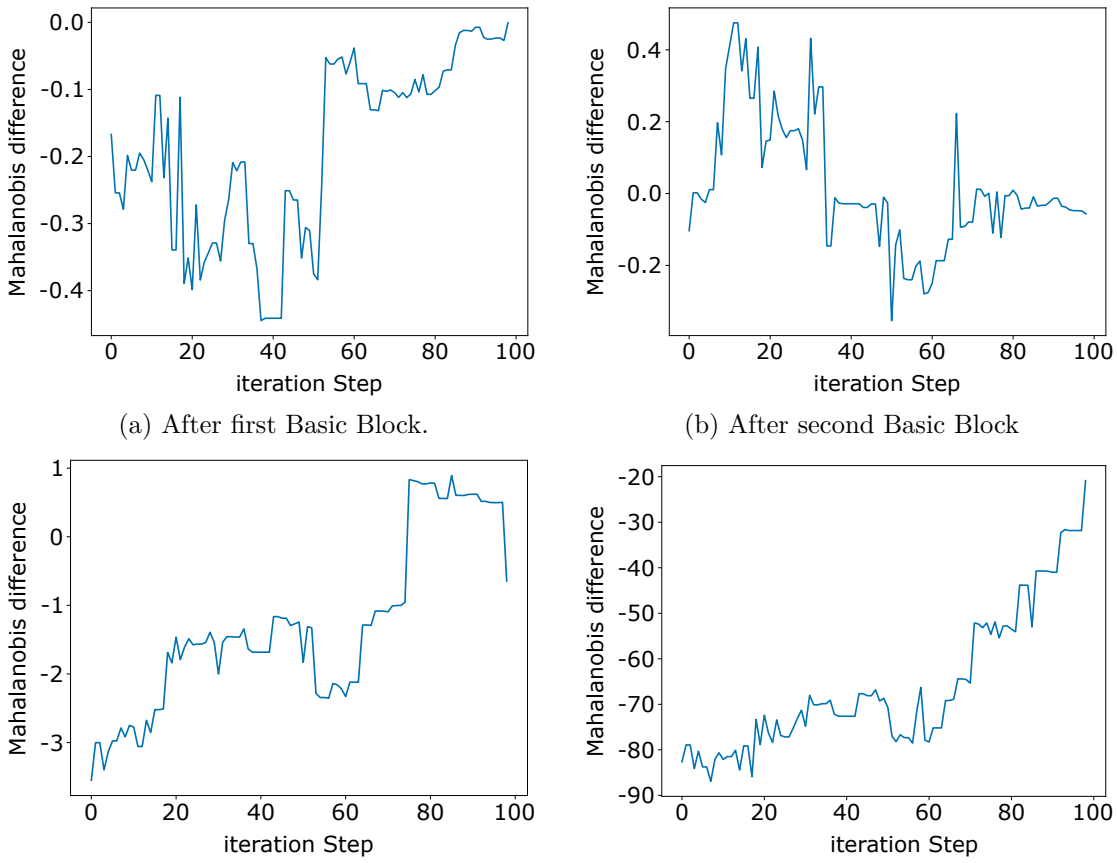
7.3. Results

In Section 7.3.1 the results are given, when the image classifier is attacked with the black-box attack proposed by Guo et al. [Guo+19]. In particular, the development of the Mahalanobis distance differences throughout the attack is outlined. Based on the observations there, in Section 7.3.2 the results are shown when using the distance differences to detect adversarial inputs. Following, in Section 7.3.3, the results are given when the Mahalanobis distances are used to classify the images, in particular the internal counterparts of a detected adversarial input. The outcomes for the OOD detection experiments are outlined in Section 7.3.4. Finally, in Section 7.3.5, the observations for the combined detection of original, adversarial, and OOD samples are presented.

7.3.1. Black-box Attack

The motivation to observe the Mahalanobis distance differences was to investigate if it is possible to determine the direction towards which a sample crossed the decision boundary when attacked internally. Applying the black-box attack by Guo et al. [Guo+19] to original samples, the Mahalanobis distances were tracked previously to and after each iteration step of the attack. The changes in the distances between consecutive steps after each of the four BBs are shown in Figure 7.4 for one observed sample.

[1]There are 531,131 more samples available, which are not used in this section.



(a) After first Basic Block.

(b) After second Basic Block

(c) After third Basic Block

(d) After fourth Basic Block

Figure 7.4.: Progression of the Mahalanobis distance difference between input and internal counterpart during the black-box attack after the four Basic Blocks.

Figure 7.4 shows, that it is not directly possible to identify the iteration at which the sample crosses the decision boundary. However, in particular, the calculated Mahalanobis distance differences after the fourth BBs evolved as assumed. At the beginning of the black-box attack, large negative distance differences were observed, which approached towards 0 throughout the attack, i.e., the initial input got closer to the assumed decision boundary. The distance difference progression of two examples, which strongly behaved as assumed, are shown in Figure 7.5.

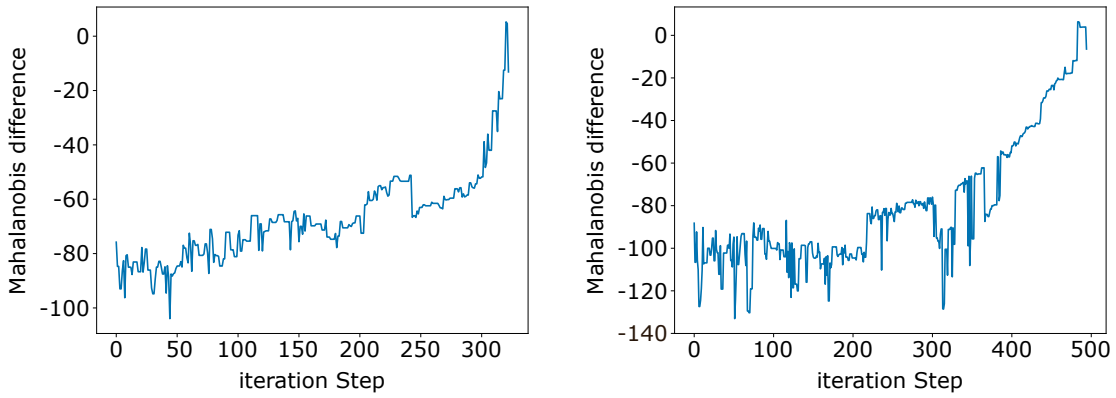
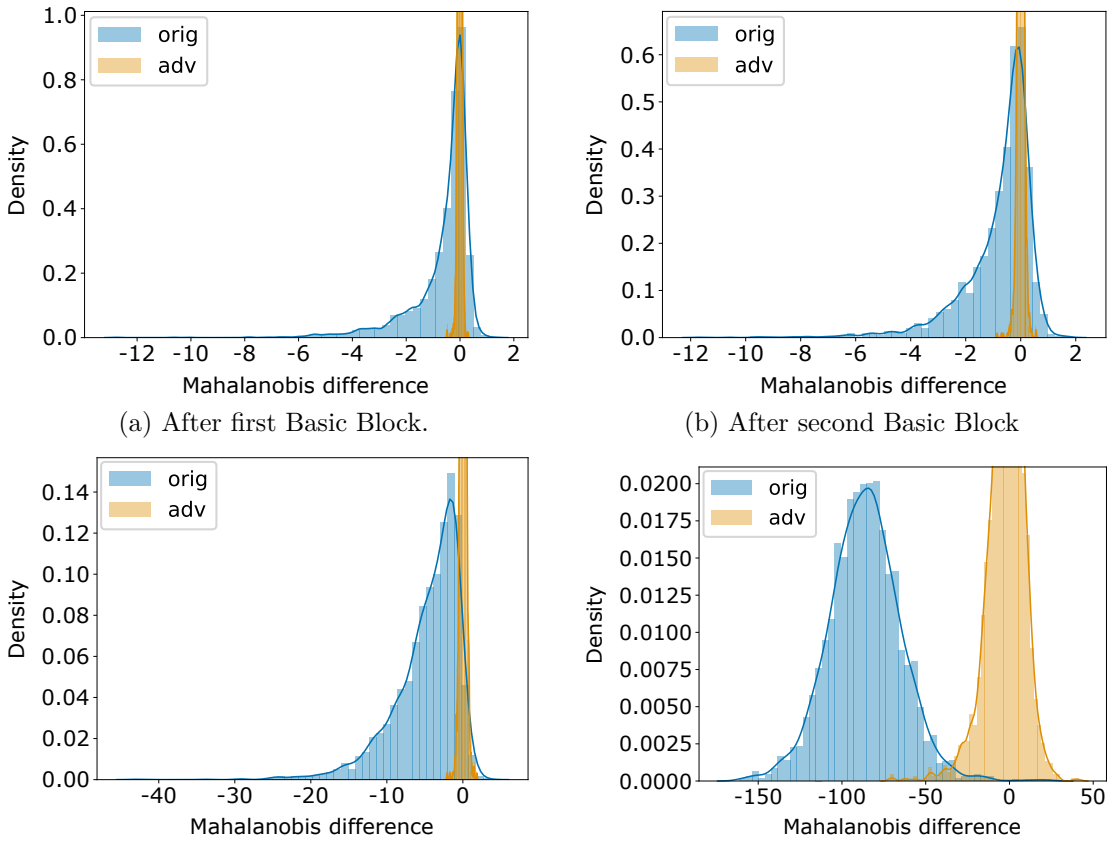


Figure 7.5.: Progression of the Mahalanobis differences for two samples after the fourth Basic Block. In both cases no clear point can be defined when the decision boundary is crossed.

Even though it was not directly possible to detect the exact iteration at which a sample crossed the decision boundary, a strong decrease in the distance differences from the initial clean sample towards the final adversarial input was observable. To evaluate this behaviour on a larger scale, the black-box attack necessary to investigate the difference progressions was exchanged for the different white-box attacks (see Section 7.2) which only calculates a final adversarial input. For each adversarial input, as well as the original inputs, the Mahalanobis distance was calculated before and after the internal attack, and afterwards, the difference between these distances was determined. In Figure 7.6 the resulting distribution densities of the differences for initially original and adversarial inputs is shown when the internal attack is JSMA. For a better visibility, the densities are scaled to the maximal expansion of the original inputs. It is recognisable that the distributions differ between original and adversarial inputs, especially for the outputs after the fourth BB. In general, the distance differences for adversarial inputs were observed to be centred around 0, while the distance differences for original inputs are located further away from the assumed decision boundary.

7.3.2. Detection of Adversarial Inputs

As outlined in Section 7.1, a dataset of Mahalanobis distance differences was build, based on original, as well as different adversarially manipulated inputs, to train and test the



(a) After first Basic Block.

(b) After second Basic Block

(c) After third Basic Block

(d) After fourth Basic Block

Figure 7.6.: Density distribution of the Mahalanobis difference between input and internal counterpart after the four Basic Blocks for original and adversarial inputs. The internal attack is JSMA.

internal classifiers on. Prior to training the classifiers, the features were standardised by $\frac{\mathbf{x}-\boldsymbol{\mu}}{\boldsymbol{\sigma}}$, where $\mathbf{x}$ is the sample to transform, $\boldsymbol{\mu}$ are the mean values for each dimension over all samples, and $\boldsymbol{\sigma}$ are the standard deviation values for each dimension over all samples. As internal classifiers a LR, an ET, and kNN were used. All classifiers are provided by the python library *scikit-learn* [Ped+11]. The training was done with 5-fold cross-validation, and a grid search over the parameters given in Table 7.1. It is also to mention that the classifiers were only trained on samples, where the external and internal attack were the same. This assumption was made, because in a real-world scenario the external attack is not known, and only the internal attack is available. To test the classifiers, images of the test set of Cifar-10 were processed the same way as described in Section 7.1, i.e., the intermediate outputs before and after the internal attack were compressed and the Mahalanobis distance difference was calculated. However, for the testing set, all attacks presented were used to create the initially adversarial inputs.

Table 7.1.: Grid search parameters for the different internal classifiers.

LR	solver: L-BFGS
ET	number of estimators: {1,5,10,20,50,100}
kNN	number of neighbors: {1,5,10,20,50,100}, algorithm: auto

As the usage of Mahalanobis distances was investigated by Lee et al. [Lee+18], the first aim was to replicate their results. Therefore, only the Mahalanobis distances of the initial inputs were used to train the internal classifiers. For the training process, it is important to mention, that Lee et al. [Lee+18] used parts of the testing data to tune their LR. Besides, they considered that the external attack was either known, i.e., the adversarial inputs were created by the same adversarial attack during training and testing, or FGSM was considered as the known attack to evaluate the generalisation of their method. In the replication study, no parts of the testing data were used to train the internal classifiers. In Table 7.2 the detection accuracies are shown when the different combinations of internal and external attacks are employed. Here, the internal attack indicates the attack applied to create the adversarial inputs for training the classifiers.

After building the baseline, the features for the training and testing process were changed to be the differences between the Mahalanobis distance before and after the internal attacks. The results are given in Table 7.3.

Motivated by the promising results reported in Section 5.3.3 the Mahalanobis distance differences were combined with the L_p-Norm differences, to evaluate if the addition in features leads to better detection accuracy. The results were worse in almost all cases, and are therefore not reported.

To put the results of this section into context, in Table 7.4 the detection accuracies are compared with the reported results of MagNet [MC17], SafetyNet [LIF17], the replicated results based on the approach of Lee et al. [Lee+18], and NIC [Ma+19], as well as the results reported in Section 5.3.3. One conclusion from the comparison is that observing the Mahalanobis distance differences yields overall slightly better results than the baseline, i.e., observing the Mahalanobis distance of the initial input. This

Table 7.2.: Detection accuracy in % when using the Mahalanobis distance before the internal attack to train and test the classifiers. The grey cells indicate, that the classifier was trained and tested on the same external and internal attack. The best detection results for each external attack, depending on the classifier, as well as the best mean results for each classifier, are displayed in bold.

			external					
			CW	DF	FGSM	JSMA	PGD	mean
LR	internal	CW	97.25	87.78	**97.49**	97.34	97.34	95.44
		DF	91.63	**89.13**	93.00	91.64	91.66	91.41
		FGSM	97.06	88.94	96.93	96.97	97.26	95.43
		JSMA	97.26	87.09	**97.49**	**97.39**	97.42	95.33
		PGD	**97.34**	88.58	97.35	97.37	**97.53**	**95.63**
		mean	96.11	88.30	**96.45**	96.14	96.24	
ET	internal	CW	96.78	87.85	96.79	96.90	96.94	95.05
		DF	90.49	88.02	90.90	90.47	90.52	90.08
		FGSM	96.81	**88.03**	97.07	96.90	97.03	95.17
		JSMA	**97.26**	86.77	97.21	**97.39**	97.35	95.19
		PGD	97.22	87.60	**97.63**	97.37	**97.38**	**95.44**
		mean	95.71	87.65	**95.92**	95.81	95.84	
kNN	internal	CW	97.19	88.03	97.35	97.25	97.25	95.41
		DF	92.04	**89.48**	93.42	92.11	92.17	91.84
		FGSM	97.06	88.52	96.93	97.13	97.16	95.36
		JSMA	97.06	87.93	97.49	97.16	97.16	95.36
		PGD	**97.25**	87.98	**97.77**	**97.34**	**97.34**	**95.54**
		mean	96.12	88.39	**96.59**	96.20	96.22	

Table 7.3.: Detection accuracy in % when using the differences between the Mahalanobis distance before and after the internal attack to train and test the classifiers. The grey cells indicate, that the classifier was trained and tested on the same external and internal attack. The best detection results for each external attack, depending on the classifier, as well as the best mean results for each classifier, are displayed in bold.

			external					
			CW	DF	FGSM	JSMA	PGD	mean
LR	internal	CW	97.56	90.05	97.77	95.77	98.34	95.90
		DF	85.30	87.39	75.71	85.09	80.70	82.84
		FGSM	**97.91**	**90.48**	98.88	**98.04**	98.60	**96.78**
		JSMA	97.87	90.06	98.32	97.68	98.62	96.51
		PGD	97.88	90.08	**98.94**	97.87	**98.72**	96.69
		mean	**95.30**	89.61	93.92	94.89	95.00	
ET	internal	CW	97.75	92.89	97.81	95.22	98.16	96.37
		DF	85.68	88.34	75.61	82.47	81.20	82.66
		FGSM	98.04	**93.84**	98.74	**97.91**	98.74	**97.45**
		JSMA	98.06	92.84	98.29	97.64	98.31	97.03
		PGD	**98.09**	92.87	**98.90**	97.87	**98.84**	97.32
		mean	**95.53**	92.16	93.87	94.22	95.05	
kNN	internal	CW	97.53	93.08	98.10	94.71	98.37	96.36
		DF	86.24	88.37	76.81	84.09	80.32	83.17
		FGSM	97.63	**93.84**	98.88	97.77	98.74	**97.37**
		JSMA	97.87	93.06	98.48	98.03	98.66	97.22
		PGD	**97.91**	93.09	**98.94**	**98.16**	**98.78**	**97.37**
		mean	**95.44**	92.29	94.24	94.55	94.97	

increase is most noticeable for the strongest attack—DF—where an increase from 89.48% detection accuracy in the baseline to 93.84% when considering the distance differences was monitored. This was also the best detection accuracy for DF overall compared defences. However, the overall results when observing the L_p-norm differences are slightly better than monitoring the Mahalanobis distance differences, in three of the five investigated attacks.

Table 7.4.: Reported detection accuracies in % of different defences, depending on the applied attack. The defences are ordered ascending according to their time of publication.

defence	CW	DF	FGSM	JSMA	PGD
SafetyNet (2017)	-	-	-	-	95.65
MagNet (2017)	93.70	93.40	99.90	-	96.00
L_p-Norm (our)	98.28	84.63	99.30	98.47	98.13
Mahalanobis (known) (our)	97.25	89.13	96.93	97.39	97.53
Mahalanobis (unknown) (our)	97.06	88.94	96.93	96.97	97.26
Mahalanobis (overall best) (our)	97.34	89.48	97.77	97.39	97.53
NIC (2019)	**100.00**	91.00	**100.00**	**100.00**	**100.00**
Mahalanobis differences (our)	98.09	**93.84**	98.94	98.16	98.84

7.3.3. Reversion to the True Class

In addition to detecting adversarial inputs, Lee et al. [Lee+18] also report that the Mahalanobis distance can be used as a way to classify original inputs. They propose to calculate the class of the nearest cluster after each of the four BBs. Based on the classes associated with the found clusters, they execute a majority vote to identify the final classification. This process led to a small increase in classification accuracy when compared to the standard softmax classification implemented by the ResNet-34 model.

Motivated by the observations reported in Section 5.3.5 that up to 89.94% of adversarial inputs return to their original true class after the internal attack, it was investigated, whether a classification based on the nearest clusters can achieve similar or even better results. In Table 7.5 the classification accuracies are reported for adversarial inputs after the internal attack, based on the approach to calculate the nearest cluster after each BBs and the following majority vote. For comparison, the classification results of the standard softmax approach are shown.

Comparing the results in Table 7.5 show, that taking the Mahalanobis approach to classifying adversarial^{-1} inputs is not well suited for the investigated scenario.

7.3.4. Out of Distribution Detection

Based on the observations of Lee et al. [Lee+18], it was investigated whether the distributions of the Mahalanobis distance differences can also be used to differentiate between original and OOD data. In Figure 7.7 the calculated density distributions are

Table 7.5.: Classification accuracy in % of the internal counterpart of adversarial inputs after the internal attack, when taking the majority vote over the four nearest clusters. In comparison, the classification results taking the softmax prediction of the image classifier. The best result for each external attack and the best means are displayed in bold.

			external					
			CW	DF	FGSM	JSMA	PGD	mean
majority vote	internal	CW	54.90	32.61	16.54	49.03	46.88	39.99
		DF	52.53	36.29	15.81	48.67	46.03	39.87
		FGSM	53.35	33.05	15.92	48.93	46.71	39.59
		JSMA	**61.70**	**36.47**	**17.77**	**51.98**	**50.77**	**43.74**
		PGD	52.48	32.29	15.46	48.29	46.04	38.91
		mean	**54.99**	34.14	16.30	49.38	47.29	
softmax	internal	CW	86.62	88.50	**51.62**	68.48	79.21	74.89
		DF	87.22	88.44	51.01	68.78	79.79	75.05
		FGSM	89.44	89.11	51.47	**69.75**	80.65	76.08
		JSMA	**89.94**	89.11	51.47	69.68	**81.32**	**76.30**
		PGD	88.03	**89.42**	51.47	69.55	80.18	75.73
		mean	88.25	**88.92**	51.41	69.25	80.23	

shown, when the internal attack is JSMA. Similar to adversarial inputs, the distributions differ and can potentially be used to train internal classifiers to differentiate between original and OOD data. To quantify the detection capability of the Mahalanobis distance differences in this scenario, at first, the results of Lee et al. [Lee+18] were replicated as a baseline to compare to. The results of the replication study are listed in Table 7.6.

Table 7.6.: Detection accuracy for different classifiers, when considering the initial Mahalanobis distance to differentiate between original and SVHN data. The best results are displayed in bold.

LR	ET	kNN
94.47	94.06	**94.47**

In Table 7.7 the results are listed when the Mahalanobis distance differences are considered to train and test the classifiers, denoted as mahal_diff. Besides, the results are listed when adding the L_p-norm differences to the training and testing data (mahal_diff + L_p), as well as singularly considering the L_p-norm differences (L_p) between the input and the internal counterpart to differentiate between original and the SVHN data.

Table 7.7 display that including the L_p-norm data into the detection process leads to a small increase in accuracy. However, overall considering the Mahalanobis distance differences was slightly worse than considering the initial Mahalanobis distance of the unknown input, without an internal attack. To put the numbers into context, in Table 7.8

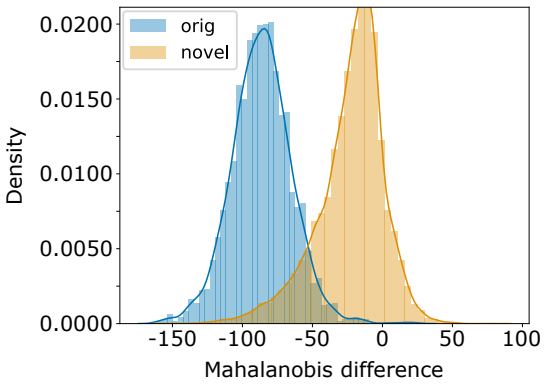


Figure 7.7.: Density distribution of the Mahalanobis distance differences for original and SVHN data.

Table 7.7.: Detection accuracy in % when using different combinations of data to train and test the internal classifiers to differentiate between original, and SVHN data. The best results for each classifier are displayed in bold.

		internal attack				
	classifier	CW	DF	FGSM	JSMA	PGD
mahal_diff	LR	91.81	87.40	90.03	**93.58**	91.38
	ET	92.41	88.33	90.45	**93.12**	91.03
	kNN	91.66	88.62	90.97	**92.19**	90.59
mahal_diff + L_p	LR	91.56	90.53	89.26	**93.83**	91.31
	ET	93.59	93.42	92.13	**93.90**	93.34
	kNN	91.44	83.20	89.16	**91.96**	89.81
L_p	LR	**90.44**	91.22	62.97	86.22	89.97
	ET	92.22	92.32	87.61	85.57	**92.47**
	kNN	**93.13**	87.30	84.06	86.12	73.63

the results of this section are set in comparison to other techniques, namely a baseline Hendrycks et al. [HG17] proposed and ODIN [LLS18]. For the approach considering the initial Mahalanobis distance, the values of Lee et al. (Lee [Lee+18]) are reported, as well as the accuracies found in the replication study (mahal).

Table 7.8.: Detection accuracies in % for different detection algorithms, when considering SVHN as OOD data, while Cifar-10 is the original dataset.

Baseline	ODIN	Lee [Lee+18]	mahal	mahal_diff + L_p
85.1	91.1	95.8	94.47	93.90

Overall, the results reported in this section verified that original and OOD data behave differently when attacked internally. Even though for this purpose the Mahalanobis distance of the initial unknown input was better suited to identify OOD data, observing the distance difference still outperformed earlier methods.

7.3.5. Original vs. Adversarial vs. Out of Distribution detection

Comparing the density distributions of the Mahalanobis distance differences for original, adversarial and SVHN data in Figure 7.8 illustrates, that they are different to each other and could potentially be used to differentiate between these three sources of unknown input samples.

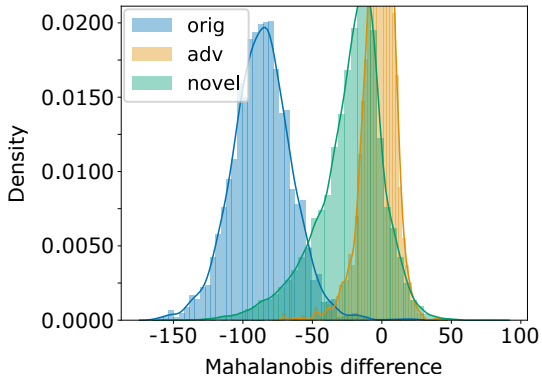


Figure 7.8.: Density distribution of the Mahalanobis distance differences for original, adversarial, and SVHN data.

In the experiments, the detection accuracies were determined, when the internal classifiers were trained and tested on the Mahalanobis distance of the initial inputs (mahal), the Mahalanobis distance differences between the input and its internal counterpart (mahal_diff), and the mahal_diff values enriched by the L_p-norm differences (mahal_diff + L_p) proposed in Section 5. Considering Mahal_diff + L_p achieved the overall best detection accuracies on the task to differentiate between original, adversarial, and SVHN inputs. The comparison is given in Table 7.10, while a detailed overview of the results

achieved when using mahal_diff + L_p to train and test the internal classifiers is outlined in Table 7.9.

Table 7.9.: Detection accuracy in % for different internal classifiers when using the Mahalanobis distance differences combined with the L_p-norm differences to differ between original, adversarial, and SVHN inputs, based on the internal and external attack. The best detection results for each external attack, depending on the classifier, as well as the best mean results for each classifier, are displayed in bold.

			external					
			CW	DF	FGSM	JSMA	PGD	mean
LR	internal	CW	79.15	64.07	79.80	80.86	81.21	77.02
		DF	48.47	59.15	49.95	50.09	48.64	51.26
		FGSM	79.12	64.99	80.63	80.02	80.90	77.13
		JSMA	**90.57**	**76.87**	**88.83**	**90.17**	**89.91**	**87.27**
		PGD	81.24	64.72	81.01	79.88	82.23	77.82
		mean	75.71	65.96	76.04	76.21	**76.58**	
ET	internal	CW	88.18	70.25	79.80	88.99	88.54	83.15
		DF	66.92	65.42	60.13	66.54	68.08	65.42
		FGSM	82.19	65.57	88.08	79.87	82.97	79.74
		JSMA	**89.00**	**74.69**	**90.60**	**91.18**	**90.10**	**87.11**
		PGD	79.11	66.54	70.58	77.24	80.15	74.72
		mean	81.08	68.49	77.84	80.76	**81.97**	
kNN	internal	CW	79.30	63.61	79.80	80.15	80.63	76.70
		DF	69.28	60.69	62.46	67.28	70.03	65.95
		FGSM	52.81	58.45	75.14	69.25	55.91	62.51
		JSMA	**82.74**	**72.02**	**88.36**	**87.17**	**86.42**	**83.34**
		PGD	71.61	61.23	61.67	69.56	71.15	67.04
		mean	71.35	63.20	73.49	**74.68**	72.83	

7.4. Conclusion

When applying a black-box attack to an original input, it was classified as adversarial by the process proposed in Chapter 5 a few iteration steps before the classification changes and the input is a legitimate adversarial input. Motivated by this observation, it was the aim to investigate if it is possible, to determine the direction in which an unknown sample crosses a decision boundary. To specify the position of an unknown sample in the decision space, in particular during the decision making process of the image classifier, the intermediate outputs of the four BBs were monitored. Since the raw outputs itself are high dimensional, they were compressed by calculating the channel-wise

Table 7.10.: Overall best detection accuracy scores for different external attacks when considering the Mahalanobis distances of the input (mahal), the Mahalanobis distance differences between input and internal counterpart (mahal_diff), and mahal_diff plus the L_p-norm differences between input and internal counterpart (mahal_diff + L_p). The best results are displayed in bold.

	external				
	CW	DF	FGSM	JSMA	PGD
mahal	87.45	**82.26**	86.22	88.01	85.87
mahal_diff	89.11	78.31	89.48	90.02	88.72
mahal_diff + L_p	**90.57**	76.87	**90.60**	**91.18**	**90.10**

mean values. This might lead to information loss, in particular to spatial information loss of specific features, and can be the topic of further research to find a probably better way of compressing the features.

Based on the compressed intermediate outputs of unknown inputs, the Mahalanobis distance towards previously defined clusters was determined. The necessary mean and covariance values describing the clusters for each possible class were calculated based on training samples of Cifar-10. After the internal attack, the Mahalanobis distance of the internal counterpart was calculated towards the same clusters as for the initial input, and finally, the difference between these distances was monitored. The difference was assumed to be the distance of the initial input towards the decision boundary separating the input from another classification area. The assumption here was that the difference is negative, i.e., the input moves further away from the cluster's centre, when processing initially original inputs. Contrary, when processing initially adversarial inputs, the difference was assumed to increase, because the input moves towards the centre of the cluster surrounding the adversarial island. To evaluate these assumptions, the progression of the distance differences throughout a black-box attack was monitored. The results showed, that the distance towards the assumed decision boundary indeed shrunk during the adversarial attack, however, the difference either did not become positive, i.e., the sample did not cross the assumed decision boundary, or the distance difference became already positive before the input's classification changed. Therefore, no precise point could be identified at which the sample crossed the decision boundary.

Even though at this point it is not possible to identify the direction towards which a sample crosses the decision boundary, the distributions of the distance differences were different between original and adversarial inputs. Following the idea of Lee et al. [Lee+18], at first the classifiers were trained on the Mahalanobis distances of the initial input without any internal attack to replicate the results of Lee et al. [Lee+18] and to establish a baseline. By using the distance differences instead, the adversarial detection results improved slightly to be between 93.84% and 98.94%. However, those results were still slightly worse to the detection results reported in Section 5.3.3, when observing the L_p-norm differences between input and internal counterpart. But looking

at the most difficult attack—DF—the detection accuracy improved to 93.84% when using the Mahalanobis distance differences and surpass state of the art.

Even though the adversarial detection results reported in this section are similar to the ones reported in Section 5.3.3, a similar problem is imaginable which led to the investigation of the Mahalanobis distances in the first place. Namely, that the internal classifier creates margins around the decision boundaries, which could lead to clean inputs being detected as adversarial inputs, and also contrary, adversarial inputs outside that margin to be classified as original inputs. However, the approach proposed in this section might be more challenging for an attacker, because several steps during the decision-making process are considered to identify adversarial inputs. An attacker would, therefore, have to optimise for several problems, which could increase the hardness of the model. This problem should be investigated in future work.

Lee et al. [Lee+18] report, that the Mahalanobis distance can also be used as an alternative to classifying unknown inputs. Therefore, they consider the class associated with the nearest cluster after each of the four BBs and make a majority vote over the found classes to determine the final classification. Motivated by the results reported in Section 5.3.5, that up to 89.94% of adversarial inputs return to their original true class after the internal attack when classified by the standard softmax image classifiers, the classification accuracies were also calculated based on the Mahalanobis distance approach. However, the results showed that taking the Mahalanobis approach is worse than the standard softmax classifier, and thereby is no option for this kind of classification.

In addition to detecting specifically adversarial inputs, Lee et al. [Lee+18] propose to use their method to detect OOD data in general. Considering SVHN as OOD, detecting them based on the initial Mahalanobis distance yielded the best detection accuracy of 94.47%. In comparison, making the decision based on the distance differences, even combined with the L_p-norm differences, achieved a detection accuracy of at best 93.90%.

Finally, the problems of detecting adversarial and SVHN as OOD samples were combined. Even tough adversarial and SVHN data both can be considered as OOD, an investigation to differentiate between original, adversarial, and OOD samples was interesting. Original samples were assumed to have the strongest relationship to the clusters, i.e., are located near the centres of the clusters and further away from the assumed decision boundaries. In contrast, adversarial inputs were assumed to be located near the decision boundary. While OOD samples were assumed to be located somewhere between the decision boundary and the centre of the clusters. In Figure 7.8 it is shown that this assumption is valid. The distance differences indicate that original samples are the furthest away from the assumed decision boundary, while adversarial inputs are the closest. The distance distributions based on SVHN samples are located somewhere in the middle between the clusters and the decision boundary. The detection results further confirmed, that it is possible to differentiate between original, adversarial, and SVHN samples with an accuracy between 90.10% and 91.18%, depending on the external attack.

In the investigated scenario, access to the OOD data during the training process is given. But the detection results indicate that it should be possible to extend the scenario to also detect novel inputs, i.e. inputs about which no knowledge is available during the training process. Traditionally, novelty detection is considered as one-class

problem [Pim+14], because during training only the clean original data are usable, for example, Cifar-10. However, by training the image classifier as usual and applying internal attacks to unknown inputs, the position of the unknown input in the decision space could be specified, and thereby it could be determined if the input is an original, adversarial, or novel sample. Even though the Mahalanobis distance of the initial unknown input towards the clusters could also be used as a feature to decide whether an input is original or novel, a threshold would have to be defined from which distance on an input is considered as novel. Monitoring the distance difference between an input and its internal counterpart, the decision is anchored around the decision boundary at a distance difference of 0, and the mean of the cluster is at distance d_m. The closer an unknown input is to the decision boundary, the more likely it is to be a novel input. When also including adversarial inputs into the overall consideration, it could be possible to estimate a distance difference towards the decision boundary, for which inputs are considered to be adversarial instead of novel. In general, a differentiation between original, adversarial, and novel inputs is useful in, e.g., autonomous driving. For adversarial inputs, the true classification, in theory, should be known, while for novel inputs there is no chance to classify them correctly. Therefore, only separating between original and adversarial inputs could lead to a false prediction of an even in theory unpredictable input, while only differentiating between original and novel inputs could lead to not correctly predicting an in theory correctly predictable sample. These considerations should be the subject of further studies.

Part III.

Conclusion and Future Work

8. Conclusion

In modern society, machine learning in the form of DNNs becomes more and more ubiquitous. Whether autonomous cars are considered, which observe their environment with cameras, or smart home assistances which are commanded by speech. The underlying system usually at some point implements DNNs to classify a given input, i.e., give an input a certain semantic meaning. In the example of an autonomous car, this could be to classify a found street sign as a stop sign and as a consequence the car brakes. Despite their good performance, DNNs are prone to adversarial inputs. The modern usage of the term describes the phenomenon, that inputs can be manipulated, essentially undetectable for humans, but trick the classifier into misclassification. In this thesis, two aspects of adversarial inputs were investigated, namely adversarial attacks, i.e., how to create adversarial inputs, and adversarial defences to detect manipulated samples.

8.1. Physical Adversarial Attacks

One common application scenario of adversarial attacks is in the context of autonomous driving. A stop sign is manipulated in such a way, that the car's visual system classifies a priority road sign, even though humans would still classify the stop sign correctly. Previous approaches tackling this scenario are based on physical manipulation of the environment. Athalye et al. [Ath+18] for example propose to print out a manipulated version of the stop sign, and either attach it somewhere or paste it over the original stop sign. A different approach by Eykholt et al. [Eyk+18] is to attach stickers to an existing stop sign to fool models into misclassification. Even though the stickers are noticeable to humans, the authors argue that the stickers look like graffiti and therefore seem innocuous to humans. Both approaches share the necessity to manipulate the original street sign or environment physically. Thereby, the attachments have to be resistant to different weather conditions and may be easy to prove in an investigation.

In Section 4 a new threat model was introduced, where a projector was used to project an adversarial image onto a wall or to project the adversarial perturbations onto an existing stop sign. Thereby, no physical manipulations were needed, and the attack might be harder to prove in a real scenario. When projecting the complete adversarial image onto the wall, the adversarial attack did not need to be restricted in any way, i.e., all colour channels were allowed to be de- and increased to create an adversarial image of a stop sign. Without any restrictions to the attack, the investigated Inception-v3 model was fooled in 97.94% of the cases. Considering the VGG-16 model, an adversariality of 98.35% was achieved. When projecting only the perturbations to an existing stop sign, different restrictions to the attack were applied. The first one was to increase the pixel values only. This was, because a decrease in pixel values, i.e., making the recaptured value at a specific

location darker, is hardly noticeable. The results confirmed this assumption. Considering no restrictions to the attack, but only projecting the perturbation, the adversariality for Inception-v3 was 30.45%. However, restricting the attack to only increase the pixel values, the success rate of the attack was 79.22% on the same Inception-v3 model. Considering VGG-16 as the target model, the adversariality was 88.48% for the same scenario of only increasing the pixel values. Aside from restricting the attack to increase the pixel values only, it was further restricted to only manipulate the green colour channel. This restriction aimed to simulate the usage of a laser pointer, which might be easier to carry around, and can thereby further decrease the detectability of an attack in a real-life scenario. Restricting the attack to exclusively increase the pixel values of the green colour channel, Inception-v3 was still fooled in 21.61% of the cases, while VGG-16 seems to be more prone to the adapted attack with an adversariality of 56.79%. Overall, it was possible to fool classification models in the context of street sign classification by only projecting the necessary perturbations onto an existing street sign. Even when considering a projector with one colour channel, like a laser pointer, the investigated models were fooled substantially.

8.2. Adversarials^{-1}: Detecting Adversarial Inputs with Internal Attacks

In the Chapters 5 to 7 a new adversarial defence was proposed, based around the idea of internal attacks. Taking an unknown input, it was manipulated by an adversarial attack internally, and afterwards, the L_0, L_2, and L_∞-norm distances between the input and its internal counterpart were calculated. In a preliminary study, the observed distances for an initially original and an initially adversarial input were found to be different—the distances for initially adversarial inputs were smaller than for initially benign inputs. Based on that observation, a LR, kNN, and an ET were trained, to distinguish between original and adversarial inputs, based on the calculated distances after the internal attack. Hence, an accuracy to differentiate the two possible origins of an unknown input between 84.63% and 99.30% was reported. Comparing these results to the reported values of other researchers [MC17; LIF17; Lee+18; Ma+19], the proposed method to monitor the L_p-norm differences had a higher detection accuracy than previous publications. Compared to more recent approaches, the process implemented for the experiments achieved in part still better results or is at least comparable in its performance. Our observations are reinforced by a recent publication of Hu et al. [Hu+19], who also proposed to use internal attacks to differ between original and adversarial inputs. However, they did not consider the L_p-norm distances as a measure to quantify the difference between the initial input and its internal counterpart, but the number of necessary iteration steps to change the classification. In the outlook in Section 8.3.2, the difference is discussed in more detail with regard to possible future work.

In addition to detecting adversarial inputs, it was also investigated how many of the initially adversarial inputs return to their original true class after the internal attack. The results showed that up to 89.94% of the initially adversarial inputs reverted to their

original true class. In comparison to other publications, the classification accuracy of adversarial inputs after the internal attack surpassed even very recent publications in four of the five considered external attacks.

Based on the promising results in image classification, the general approach was transferred to speech classification. Concerning the detection of adversarial inputs, the approach from the image domain could be transferred without larger adaptations. For the observed dataset, and the two adversarial attacks[1] an accuracy to differentiate between original and adversarial inputs of 99.9% was reported for Carlini & Wagners attack [CW18], resp. an accuracy of 99.8% for Alzantot et al.'s [ABS18] attack. Compared with recent literature, the reported detection accuracies were competitive or higher than other reported values [RSK18; Zen+19].

As with image classification, it was also examined if the internal attacks could be used to restore the original class of adversarial inputs. Here, major adjustments were necessary, because in speech classification no untargeted attacks, like in image classification, are possible. This is because even when each subsequence of the audio file could be attacked untargeted, the final word has to be coherent. To counter this property, the internal classifiers were trained to predict the original true class for an initially adversarial input. When the original true class is the same as the target of the internal attack, the prediction accuracy of the initial correct word is rather high. Using this observation, 67.6% of the adversarial inputs were reverted to their original true classification. To the best of my knowledge, this was the first time the original true class of adversarial inputs was restored in such a manner.

Even though the detection accuracies of adversarial inputs in a grey-box scenario were convincing, a problem occurred when giving the attack knowledge about the defence. Having that knowledge, it was possible to create inputs which were still benign but detected as adversarial, and in the consequence classified wrong. This observation led to the research question if it is possible to determine the direction in which a sample crosses the decision boundary when attacked internally. Switching back to image classification, the intermediate outputs of the employed ResNet-34 image classifier were included in the overall considerations. Based on the outputs of the hidden layers, the Mahalanobis distance towards previously defined clusters was calculated to specify the position of unknown inputs in the decision space. The mean and covariance values defining the clusters were determined based on the training samples of Cifar-10, on which ResNet-34 was trained as well. Besides, the Mahalanobis distance was also calculated for the internal counterpart after the internal attack. Afterwards, the differences between the distances before and after the internal attack were monitored. It was assumed, that initially original inputs move away from the centre of the clusters, i.e., crosses the decision boundary from an original area towards an adversarial island. Vice versa, a decrease in the distances indicates that the input was initially adversarial and moved towards the original true classification area. However, applying a black-box attack to investigate the progression of the distance differences towards a misclassification, there was no evidence that it is

[1] Adversarial attacks in speech classification are more recent than in image classification. Therefore, in the current literature mainly two adversarial attacks are considered.

possible to determine the exact point, at which an input crosses the decision boundary, and in which direction.

Still, the Mahalanobis distance differences could be used to identify adversarial inputs, similar to the L_p-norms. The results on detecting adversarial inputs were comparable to the accuracies reported for the L_p-norms, i.e., between 98.09% and 98.94% for four of the five attacks. For the fifth attack, DF, which is the hardest to detect for all compared defences, the detection accuracy of 84.63% when considering the L_p-norm differences, increased to 93.84% when taking the Mahalanobis distance differences for the detection process. Compared to the literature, the results reported in this thesis surpasses the state of the art.

In addition to detecting adversarial inputs, the Mahalanobis distance differences could be used to identify OOD samples in general. In comparison on detecting a different dataset, a slightly lower detection accuracy of 93.90% was reported, compared to the 95.8% published by Lee et al. [Lee+18] considering the Mahalanobis distance of the initial input to differentiate between original and OOD samples. However, it was also investigated if it is possible to differentiate between original, adversarial and SVHN data, all at once. To the best of my knowledge, this is the first time considering that type of problem. Taking the Mahalanobis distance of the initial input towards the nearest cluster, as Lee et al. [Lee+18] propose, as a feature to differ between the different origins of an unknown sample, accuracies between 85.87% to 88.01% were achieved, for four of the five external attacks. In the case of DF, an accuracy of 82.26% was reported. When taking the Mahalanobis distance differences combined with the L_p-norm differences, those accuracies improved to be between 90.10% and 91.18%. Only for DF as the external attack, the accuracy reduces to 76.87%.

8.3. Future work

Based on the results, and in general, during the development of this thesis ideas for future research emerged.

8.3.1. Physical adversarial attacks

Motivated by a variety of possible attacks to autonomous cars, further investigations are crucial to make the underlying systems more robust, and also more trustworthy by giving some guarantees for the safety of such systems. Regarding the exploration of adversarial attacks, it seems to be necessary to have better insight into the systems used in real autonomous cars. In the experiments conducted in Chapter 4, the attacks were tested on openly accessible image classification models, similar to other comparable publications. Currently, there are very few publications using the classification feedback of real autonomous cars. However, when such feedback of the whole system is available, the proposed method of projecting adversarial inputs can speed up the process of, e.g., black-box attacks to find weaknesses of the overall system. By reversing the found weaknesses, the overall system could become more robust.

8.3.2. Adversarials^{-1}: Detecting adversarial inputs with internal attacks

The observation that original and adversarial inputs behave differently when attacked internally, is reinforced by the very recent work of Hu et al. [Hu+19]. In contrast to this thesis, they considered the number of iteration steps necessary to change the classification of an input as the distance measure, instead of L_p-norm distance. This decision has also been questioned by Tramèr et al. [Tra+20] in a very recent publication. In addition to the distance towards the decision boundary, Hu et al. [Hu+19] also included the robustness of an input against Gaussian noise to differentiate between original and adversarial inputs. They argue that original inputs are more robust against Gaussian noise, in particular when random noise is used within the training process of the image classifier. However, adversarial inputs more easily change their classification when subjected to Gaussian noise. If the adversarial input is therefore moved deep into an adversarial island to be more robust against Gaussian noise, it would need more iteration steps of an adversarial attack to change its classification, in comparison to an original input. Against Hu et al.'s claim, that they can achieve a worst-case detection accuracy of 49% against white-box attacks, Tramèr et al. [Tra+20] were still able to produce inputs which are wrongly classified and to reduce the detection accuracy to 0%.

Another aspect of internal attacks is to quantify the distance towards the nearest decision boundary. This knowledge could be used during the training process, by maximising the distance of training samples towards the nearest decision boundary. Thereby, it could be quantified how distant an unknown sample has to be, to be classified differently. This would enhance the explainability and general understanding of DNNs.

Other than that, it was shown that the proposed defence can be transferred from image to speech classification. Future work could be to transfer the defence to even other domains like malware or spam classification, as it has been shown that adversarial attacks are also possible in those domains. As a necessity, there have to exist some sort of distance measure to create adversarial inputs. And in consequence, it should be possible to attack adversarial inputs again, and based on the internal distances detect adversarial inputs.

Bibliography

[20003] European Telecommunications Standards Institute 2003. "Speech processing, transmission and quality aspects (STQ); Distributed speech recognition; Advanced front-end feature extraction algorithm; Compression algorithms". In: *ETSI ES* 201.108 (2003), p. v1.1.3 (cit. on p. 66).

[ABS18] Moustafa Alzantot, Bharathan Balaji, and Mani B. Srivastava. "Did you hear that? Adversarial Examples Against Automatic Speech Recognition". In: *CoRR* abs/1801.00554 (2018). arXiv: 1801.00554 (cit. on pp. 65, 71, 72, 77, 103).

[AG17] Mahdieh Abbasi and Christian Gagné. "Robustness to Adversarial Examples through an Ensemble of Specialists". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. 2017 (cit. on p. 31).

[AS14] Shruti Asmita and KK Shukla. "Review on the architecture, algorithm and fusion strategies in ensemble learning". In: *International Journal of Computer Applications* 108.8 (2014) (cit. on p. 31).

[Ath+18] Anish Athalye et al. "Synthesizing Robust Adversarial Examples". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. 2018, pp. 284–293 (cit. on pp. 34, 37, 40, 41, 45, 101, 119).

[Bab+15] Nura Muhammad Baba et al. "Current issues in ensemble methods and its applications". In: *Journal of Theoretical and Applied Information Technology* 81.2 (2015), p. 266 (cit. on p. 31).

[Bar+06] Marco Barreno et al. "Can machine learning be secure?" In: *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2006, Taipei, Taiwan, March 21-24, 2006*. 2006, pp. 16–25. DOI: 10.1145/1128817.1128824 (cit. on pp. 23, 24).

[BB16] Wilhelm Burger and Mark J. Burge. *Digital Image Processing - An Algorithmic Introduction Using Java, Second Edition*. Texts in Computer Science. Springer, 2016. ISBN: 978-1-4471-6683-2. DOI: 10.1007/978-1-4471-6684-9 (cit. on p. 40).

[Bel15] Richard Bellman. *Adaptive Control Processes - A Guided Tour (Reprint from 1961)*. Vol. 2045. Princeton Legacy Library. Princeton University Press, 2015. ISBN: 978-1-4008-7466-8. DOI: 10.1515/9781400874668 (cit. on p. 22).

[Buc+18] Jacob Buckman et al. "Thermometer Encoding: One Hot Way To Resist Adversarial Examples". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. 2018 (cit. on pp. 29, 30).

[Cha+18] Anirban Chakraborty et al. "Adversarial Attacks and Defences: A Survey". In: *CoRR* abs/1810.00069 (2018). arXiv: 1810.00069 (cit. on pp. 23, 24).

[Che+17] Xinyun Chen et al. "Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning". In: *CoRR* abs/1712.05526 (2017). arXiv: 1712.05526 (cit. on p. 27).

[CP] U.S. Customs and Border Protection. Accessed: March 24, 2020. URL: https://www.cbp.gov/travel/biometrics (cit. on p. 1).

[CW17] Nicholas Carlini and David A. Wagner. "Towards Evaluating the Robustness of Neural Networks". In: *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*. 2017, pp. 39–57. DOI: 10.1109/SP.2017.49 (cit. on pp. 33, 119).

[CW18] Nicholas Carlini and David A. Wagner. "Audio Adversarial Examples: Targeted Attacks on Speech-to-Text". In: *2018 IEEE Security and Privacy Workshops, SP Workshops 2018, San Francisco, CA, USA, May 24, 2018*. 2018, pp. 1–7. DOI: 10.1109/SPW.2018.00009 (cit. on pp. 65, 70, 76, 103).

[Dal+04] Nilesh N. Dalvi et al. "Adversarial classification". In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*. 2004, pp. 99–108. DOI: 10.1145/1014052.1014066 (cit. on p. 22).

[Don+19] Yinpeng Dong et al. "Evading Defenses to Transferable Adversarial Examples by Translation-Invariant Attacks". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. 2019, pp. 4312–4321. DOI: 10.1109/CVPR.2019.00444 (cit. on p. 26).

[DV16] Vincent Dumoulin and Francesco Visin. "A guide to convolution arithmetic for deep learning". In: *CoRR* abs/1603.07285 (2016). arXiv: 1603.07285 (cit. on p. 18).

[DZJ19] Debayan Deb, Jianbang Zhang, and Anil K. Jain. "AdvFaces: Adversarial Face Synthesis". In: *CoRR* abs/1908.05008 (2019). arXiv: 1908.05008 (cit. on p. 1).

[Eyk+18] Kevin Eykholt et al. "Robust Physical-World Attacks on Deep Learning Visual Classification". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, pp. 1625–1634. DOI: 10.1109/CVPR.2018.00175 (cit. on pp. 1, 37, 38, 45, 101).

[Fin+19] Samuel G Finlayson et al. "Adversarial attacks on medical machine learning". In: *Science* 363.6433 (2019), pp. 1287–1289 (cit. on p. 1).

[GB10] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*. 2010, pp. 249–256 (cit. on p. 13).

[GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*. 2011, pp. 315–323 (cit. on p. 10).

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016 (cit. on pp. 5, 7).

[Goo] Google. Accessed: March 24, 2020. URL: `https://www.google.com/photos/about/` (cit. on p. 1).

[Gra+06] Alex Graves et al. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks". In: *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*. 2006, pp. 369–376. DOI: `10.1145/1143844.1143891` (cit. on pp. 69, 119).

[GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015 (cit. on pp. 25, 29, 31, 120).

[Gu+19] Tianyu Gu et al. "BadNets: Evaluating Backdooring Attacks on Deep Neural Networks". In: *IEEE Access* 7 (2019), pp. 47230–47244. DOI: `10.1109/ACCESS.2019.2909068` (cit. on p. 27).

[Guo+19] Chuan Guo et al. "Simple Black-box Adversarial Attacks". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. 2019, pp. 2484–2493 (cit. on pp. 26, 79, 82, 83).

[GWK17] Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. "Adversarial and Clean Data Are Not Twins". In: *CoRR* abs/1704.04960 (2017). arXiv: `1704.04960` (cit. on pp. 28, 76).

[Han+14] Awni Y. Hannun et al. "Deep Speech: Scaling up end-to-end speech recognition". In: *CoRR* abs/1412.5567 (2014). arXiv: `1412.5567` (cit. on pp. 65, 66, 69).

[Han17] Awni Hannun. "Sequence Modeling with CTC". In: *Distill* (2017). https://distill.pub/2017/ctc. DOI: `10.23915/distill.00008` (cit. on p. 69).

[He+15] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. 2015, pp. 1026–1034. DOI: `10.1109/ICCV.2015.123` (cit. on p. 13).

[He+16] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90 (cit. on p. 54).

[HG17] Dan Hendrycks and Kevin Gimpel. "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017 (cit. on p. 93).

[Hoa+18] Van-Dung Hoang et al. "Improving Traffic Signs Recognition Based Region Proposal and Deep Neural Networks". In: *Intelligent Information and Database Systems - 10th Asian Conference, ACIIDS 2018, Dong Hoi City, Vietnam, March 19-21, 2018, Proceedings, Part II*. 2018, pp. 604–613. DOI: 10.1007/978-3-319-75420-8_57 (cit. on p. 37).

[HRF19] Zhezhi He, Adnan Siraj Rakin, and Deliang Fan. "Parametric Noise Injection: Trainable Randomness to Improve Deep Neural Network Robustness Against Adversarial Attack". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. 2019, pp. 588–597. DOI: 10.1109/CVPR.2019.00068 (cit. on p. 28).

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735 (cit. on pp. 69, 120).

[HTF13] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2013 (cit. on p. 22).

[Hu+19] Shengyuan Hu et al. "A New Defense Against Adversarial Images: Turning a Weakness into a Strength". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*. 2019, pp. 1633–1644 (cit. on pp. 102, 105).

[Hua+11] Ling Huang et al. "Adversarial machine learning". In: *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, AISec 2011, Chicago, IL, USA, October 21, 2011*. 2011, pp. 43–58. DOI: 10.1145/2046684.2046692 (cit. on pp. 23, 24).

[HZ93] Geoffrey E. Hinton and Richard S. Zemel. "Autoencoders, Minimum Description Length and Helmholtz Free Energy". In: *Advances in Neural Information Processing Systems 6, [7th NIPS Conference, Denver, Colorado, USA, 1993]*. 1993, pp. 3–10 (cit. on p. 30).

[Ily+18] Andrew Ilyas et al. "Black-box Adversarial Attacks with Limited Queries and Information". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. 2018, pp. 2142–2151 (cit. on p. 26).

[IS15] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, July 6-11, 2015*. 2015, pp. 448–456 (cit. on p. 54).

[Jia+19] Xiaojun Jia et al. "ComDefend: An Efficient Image Compression Model to Defend Adversarial Examples". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. 2019, pp. 6084–6092. DOI: 10.1109/CVPR.2019.00624 (cit. on pp. 30, 62).

[Kag] Kaggle. Accessed: March 25, 2020. URL: https://www.kaggle.com/c/tensorflow-speech-recognition-challenge/data (cit. on p. 72).

[KGB17a] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. "Adversarial examples in the physical world". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. 2017 (cit. on pp. 32, 37, 39, 119).

[KGB17b] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. "Adversarial Machine Learning at Scale". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017 (cit. on p. 29).

[KH09] Alex Krizhevsky and Geoffrey Hinton. "Learning multiple layers of features from tiny images". In: (2009) (cit. on pp. 54, 119).

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*. 2012, pp. 1106–1114 (cit. on p. 16).

[Lam+18] Alex Lamb et al. "Fortified Networks: Improving the Robustness of Deep Networks by Modeling the Manifold of Hidden Representations". In: *CoRR* abs/1804.02485 (2018). arXiv: 1804.02485 (cit. on p. 30).

[LeC98] Yann LeCun. "The MNIST database of handwritten digits". In: *http://yann.lecun. com/exdb/mnist/* (1998) (cit. on p. 120).

[Lee+18] Kimin Lee et al. "A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*. 2018, pp. 7167–7177 (cit. on pp. 58, 60, 61, 80, 82, 87, 90, 91, 93, 95, 96, 102, 104).

[LIF17] Jiajun Lu, Theerasit Issaranon, and David A. Forsyth. "SafetyNet: Detecting and Rejecting Adversarial Examples Robustly". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. 2017, pp. 446–454. DOI: 10.1109/ICCV.2017.56 (cit. on pp. 58, 87, 102).

[Liu+17] Yanpei Liu et al. "Delving into Transferable Adversarial Examples and Black-box Attacks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017 (cit. on p. 26).

[Liu+19a] Hong Liu et al. "Universal Adversarial Perturbation via Prior Driven Uncertainty Approximation". In: *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. 2019, pp. 2941–2949. DOI: 10.1109/ICCV.2019.00303 (cit. on p. 27).

[Liu+19b] Zihao Liu et al. "Feature Distillation: DNN-Oriented JPEG Compression Against Adversarial Examples". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. 2019, pp. 860–868. DOI: 10.1109/CVPR.2019.00095 (cit. on p. 29).

[LLS18] Shiyu Liang, Yixuan Li, and R. Srikant. "Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. 2018 (cit. on p. 93).

[LN89] Dong C. Liu and Jorge Nocedal. "On the limited memory BFGS method for large scale optimization". In: *Math. Program.* 45.1-3 (1989), pp. 503–528. DOI: 10.1007/BF01589116 (cit. on pp. 29, 120).

[Lu+17] Jiajun Lu et al. "NO Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017 (cit. on p. 34).

[LYZ18] Pengcheng Li, Jinfeng Yi, and Lijun Zhang. "Query-Efficient Black-Box Attack by Active Learning". In: *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*. 2018, pp. 1200–1205. DOI: 10.1109/ICDM.2018.00159 (cit. on p. 26).

[Ma+19] Shiqing Ma et al. "NIC: Detecting Adversarial Samples with Neural Network Invariant Checking". In: *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. 2019 (cit. on pp. 58, 62, 87, 102).

[Mad+18] Aleksander Madry et al. “Towards Deep Learning Models Resistant to Adversarial Attacks”. In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. 2018 (cit. on pp. 29, 32, 120).

[Mah36] Prasanta Chandra Mahalanobis. “On the generalised distance in statistics”. In: *Proceedings of the National Institute of Sciences of India*. Vol. 2. 1. 1936, pp. 49–55 (cit. on pp. 24, 80).

[Mao+16] Xuehong Mao et al. “Hierarchical CNN for traffic sign recognition”. In: *2016 IEEE Intelligent Vehicles Symposium, IV 2016, Gotenburg, Sweden, June 19-22, 2016*. 2016, pp. 130–135. DOI: 10.1109/IVS.2016.7535376 (cit. on p. 37).

[MC17] Dongyu Meng and Hao Chen. “MagNet: A Two-Pronged Defense against Adversarial Examples”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*. 2017, pp. 135–147. DOI: 10.1145/3133956.3134057 (cit. on pp. 30, 58, 87, 102).

[MFF16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. “DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016, pp. 2574–2582. DOI: 10.1109/CVPR.2016.282 (cit. on pp. 34, 119).

[Moo+17] Seyed-Mohsen Moosavi-Dezfooli et al. “Universal Adversarial Perturbations”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 86–94. DOI: 10.1109/CVPR.2017.17 (cit. on p. 27).

[Moz] Mozilla. Accessed: March 25, 2020. URL: https://github.com/mozilla/DeepSpeech (cit. on pp. 66, 69, 72).

[MP87] Marvin Minsky and Seymour Papert. *Perceptrons - an introduction to computational geometry*. MIT Press, 1987. ISBN: 978-0-262-63111-2 (cit. on p. 8).

[Net+11] Yuval Netzer et al. “Reading digits in natural images with unsupervised feature learning”. In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning, Granada, Spain, December 16, 2011*. 2011 (cit. on pp. 83, 121).

[NYC15] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. “Deep neural networks are easily fooled: High confidence predictions for unrecognizable images”. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. 2015, pp. 427–436. DOI: 10.1109/CVPR.2015.7298640 (cit. on p. 24).

[Pan+19] Tianyu Pang et al. "Improving Adversarial Robustness via Promoting Ensemble Diversity". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. 2019, pp. 4970–4979 (cit. on pp. 31, 62).

[Pap+16] Nicolas Papernot et al. "The Limitations of Deep Learning in Adversarial Settings". In: *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*. 2016, pp. 372–387. DOI: 10.1109/EuroSP.2016.36 (cit. on pp. 23–25, 33, 120).

[Pap+17] Nicolas Papernot et al. "Practical Black-Box Attacks against Machine Learning". In: *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*. 2017, pp. 506–519. DOI: 10.1145/3052973.3053009 (cit. on p. 26).

[Ped+11] Fabian Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *J. Mach. Learn. Res.* 12 (2011), pp. 2825–2830 (cit. on pp. 58, 73, 87).

[Pim+14] Marco A. F. Pimentel et al. "A review of novelty detection". In: *Signal Process.* 99 (2014), pp. 215–249. DOI: 10.1016/j.sigpro.2013.12.026 (cit. on p. 97).

[Ran+19] Anurag Ranjan et al. "Attacking Optical Flow". In: *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. 2019, pp. 2404–2413. DOI: 10.1109/ICCV.2019.00249 (cit. on pp. 1, 47).

[RBB17] Jonas Rauber, Wieland Brendel, and Matthias Bethge. "Foolbox: A Python toolbox to benchmark the robustness of machine learning models". In: *arXiv preprint arXiv:1707.04131* (2017). arXiv: 1707.04131 (cit. on p. 33).

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536 (cit. on p. 12).

[RN16] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016 (cit. on pp. 5, 7).

[RSK18] Krishan Rajaratnam, Kunal Shah, and Jugal Kalita. "Isolated and Ensemble Audio Preprocessing Methods for Detecting Adversarial Examples against Automatic Speech Recognition". In: *Proceedings of the 30th Conference on Computational Linguistics and Speech Processing, ROCLING 2018, Hsinchu, Taiwan, October 4-5, 2018*. 2018, pp. 16–30 (cit. on pp. 65, 76, 77, 103).

[Rus+15] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *Int. J. Comput. Vis.* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y (cit. on p. 17).

[Sch+19] Lea Schönherr et al. “Adversarial Attacks Against Automatic Speech Recognition Systems via Psychoacoustic Hiding”. In: *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. 2019 (cit. on p. 2).

[Sch15] Jürgen Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural Networks* 61 (2015), pp. 85–117. DOI: 10.1016/j.neunet.2014.09.003 (cit. on p. 8).

[SD91] Jocelyn Sietsma and Robert J. F. Dow. “Creating artificial neural networks that generalize”. In: *Neural Networks* 4.1 (1991), pp. 67–79. DOI: 10.1016/0893-6080(91)90033-2 (cit. on p. 28).

[Sha+18] Uri Shaham et al. “Defending against Adversarial Images using Basis Functions Transformations”. In: *CoRR* abs/1803.10840 (2018). arXiv: 1803.10840 (cit. on p. 29).

[Sha+19] Ali Shafahi et al. “Are adversarial examples inevitable?” In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. 2019 (cit. on pp. 22, 62).

[Sha49] Claude Elwood Shannon. “Communication in the presence of noise”. In: *Proceedings of the IRE* 37.1 (1949), pp. 10–21 (cit. on p. 67).

[Sit+18] Chawin Sitawarin et al. “Rogue Signs: Deceiving Traffic Sign Recognition with Malicious Ads and Logos”. In: *Proceedings of the 1st Deep Learning and Security Workshop, DLS 2018, Hyatt Regency, San Francisco, CA, USA, May 24, 2018*. 2018 (cit. on pp. 37, 38).

[SL13] Raja Khurram Shahzad and Niklas Lavesson. “Comparative Analysis of Voting Schemes for Ensemble-based Malware Detection”. In: *JoWUA* 4.1 (2013), pp. 98–117. DOI: 10.22667/JOWUA.2013.03.31.098 (cit. on p. 31).

[SP18] Alexandru Constantin Serban and Erik Poll. “Adversarial Examples - A Complete Characterisation of the Phenomenon”. In: *CoRR* abs/1810.01185 (2018). arXiv: 1810.01185 (cit. on pp. 23–26, 28).

[SS12] Md. Sahidullah and Goutam Saha. “Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition”. In: *Speech Commun.* 54.4 (2012), pp. 543–565. DOI: 10.1016/j.specom.2011.11.004 (cit. on p. 66).

[Sta+11] Johannes Stallkamp et al. “The German Traffic Sign Recognition Benchmark: A multi-class classification competition”. In: *The 2011 International Joint Conference on Neural Networks, IJCNN 2011, San Jose, California, USA, July 31 - August 5, 2011*. 2011, pp. 1453–1460. DOI: 10.1109/IJCNN.2011.6033395 (cit. on pp. 37, 39, 120).

[SZ15] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015 (cit. on pp. 39, 45).

[Sze+14] Christian Szegedy et al. "Intriguing properties of neural networks". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. 2014 (cit. on pp. 1, 29).

[Sze+16] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016, pp. 2818–2826. DOI: 10.1109/CVPR.2016.308 (cit. on p. 39).

[Tao+19] Rohan Taori et al. "Targeted Adversarial Examples for Black Box Audio Systems". In: *2019 IEEE Security and Privacy Workshops, SP Workshops 2019, San Francisco, CA, USA, May 19-23, 2019*. 2019, pp. 15–20. DOI: 10.1109/SPW.2019.00016 (cit. on p. 76).

[TOB19] Hongxiang Tang, Alessandro Ortis, and Sebastiano Battiato. "The Impact of Padding on Image Classification by Using Pre-trained Convolutional Neural Networks". In: *Image Analysis and Processing - ICIAP 2019 - 20th International Conference, Trento, Italy, September 9-13, 2019, Proceedings, Part II*. 2019, pp. 337–344. DOI: 10.1007/978-3-030-30645-8_31 (cit. on p. 18).

[Tor] University of Toronto. Accessed: March 27, 2020. URL: https://www.cs.toronto.edu/~kriz/cifar.html (cit. on p. 54).

[Tra+18] Florian Tramèr et al. "Ensemble Adversarial Training: Attacks and Defenses". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. 2018 (cit. on p. 31).

[Tra+20] Florian Tramèr et al. "On adaptive attacks to adversarial example defenses". In: *arXiv preprint arXiv:2002.08347* (2020) (cit. on p. 105).

[TV16] Pedro Tabacof and Eduardo Valle. "Exploring the space of adversarial images". In: *2016 International Joint Conference on Neural Networks, IJCNN 2016, Vancouver, BC, Canada, July 24-29, 2016*. 2016, pp. 426–433. DOI: 10.1109/IJCNN.2016.7727230 (cit. on p. 51).

[Vin+08] Pascal Vincent et al. "Extracting and composing robust features with denoising autoencoders". In: *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*. 2008, pp. 1096–1103. DOI: 10.1145/1390156.1390294 (cit. on p. 30).

[War18] Pete Warden. "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition". In: *CoRR* abs/1804.03209 (2018). arXiv: 1804.03209 (cit. on p. 72).

[WC18] Yizhen Wang and Kamalika Chaudhuri. "Data Poisoning Attacks against Online Learning". In: *CoRR* abs/1808.08994 (2018). arXiv: 1808.08994 (cit. on p. 27).

[WK18a] Nils Worzyk and Oliver Kramer. "Adversarials $^{-1}$: Defending by Attacking". In: *2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8-13, 2018*. 2018, pp. 1–8. DOI: 10.1109/IJCNN.2018.8489630 (cit. on p. 4).

[WK18b] Nils Worzyk and Oliver Kramer. "Properties of adv-1 - Adversarials of Adversarials". In: *26th European Symposium on Artificial Neural Networks, ESANN 2018, Bruges, Belgium, April 25-27, 2018*. 2018 (cit. on p. 4).

[WKK19] Nils Worzyk, Hendrik Kahlen, and Oliver Kramer. "Physical Adversarial Attacks by Projecting Perturbations". In: *Artificial Neural Networks and Machine Learning - ICANN 2019: Image Processing - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings, Part III*. 2019, pp. 649–659. DOI: 10.1007/978-3-030-30508-6_51 (cit. on pp. 4, 38).

[WNK20] Nils Worzyk, Stefan Niewerth, and Oliver Kramer. "Adversarials $^{-1}$ in Speech Recognition: Detection and Defence". In: *28th European Symposium on Artificial Neural Networks, ESANN 2020, Bruges, Belgium, 2020 (in print)*. 2020 (cit. on pp. 4, 65).

[Wu+18] Lei Wu et al. "Understanding and Enhancing the Transferability of Adversarial Examples". In: *CoRR* abs/1802.09707 (2018). arXiv: 1802.09707 (cit. on p. 26).

[Xie+19a] Cihang Xie et al. "Feature Denoising for Improving Adversarial Robustness". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. 2019, pp. 501–509. DOI: 10.1109/CVPR.2019.00059 (cit. on p. 30).

[Xie+19b] Cihang Xie et al. "Improving Transferability of Adversarial Examples With Input Diversity". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. 2019, pp. 2730–2739. DOI: 10.1109/CVPR.2019.00284 (cit. on p. 26).

[Yan+18] Zhuolin Yang et al. "Towards mitigating audio adversarial perturbations". In: (2018) (cit. on p. 65).

[Yos+15] Jason Yosinski et al. "Understanding Neural Networks Through Deep Visualization". In: *Deep Learning Workshop @ 32nd International Conference on Machine Learning, ICML 2015, Lille, France, July 6-11 2015*. 2015 (cit. on p. 18).

[Yua+19] Xiaoyong Yuan et al. "Adversarial Examples: Attacks and Defenses for Deep Learning". In: *IEEE Trans. Neural Networks Learn. Syst.* 30.9 (2019), pp. 2805–2824. DOI: 10.1109/TNNLS.2018.2886017 (cit. on pp. 23, 24, 26).

[Zen+19] Qiang Zeng et al. “A Multiversion Programming Inspired Approach to Detecting Audio Adversarial Examples”. In: *49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2019, Portland, OR, USA, June 24-27, 2019*. 2019, pp. 39–51. DOI: 10.1109/DSN.2019.00019 (cit. on pp. 65, 76, 77, 103).

Acronyms

AE auto encoder

ANN artificial neural network

ASR Automatic speech recognition.

BB A component of the ResNet-34 architecture, comprised of a convolutional layer, followed by batch normalisation, ReLU, another convolutional layer, and batch normalisation. Afterwards, the original input of the basic block is concatenated with the processed input. Finally, ReLU is applied to calculate the output for the basic block.

BIM Iterative version of FGSM, the basic iterative method, as proposed by Kurakin et al. [KGB17a].

Cifar-10 Cifar-10 dataset [KH09], consisting of 60,000 images (50,000 for training and 10,000 for testing) of size 32×32 with 3 colour channels, representing natural objects like 'airplane' or 'bird', available at: `https://www.cs.toronto.edu/~kriz/cifar.html`

CNN convolutional neural network

CTC Connectionist temporal classification as proposed by Graves et al. [Gra+06] to train ASR systems.

CW Adversarial attack proposed by Carlini and Wagner [CW17], simply referred to as CW.

DAE denoising auto encoder

DF DeepFool is an untargeted attack, proposed by Moosavi-Dezfooli et al. [MFF16]

DFT Discrete Fourier Transformation to transfer a signal from time to frequency domain.

DNN deep neural network

DT Decision Tree classifier.

EoT Expectation over Transformation framework to transfer adversarial inputs into the real world, as proposed by Athalye et al. [Ath+18]

ET Extra Tree classifier.

FGSM Fast gradient sign method, as proposed by Goodfellow et al. [GSS15].

GTSRB German Traffic Sign Recognition Benchmark [Sta+11], available at: `http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset`

JSMA Jacobian-based saliency map attack, as proposed by Papernot et al. [Pap+16].

kNN k-nearest Neighbour classifier.

L-BFGS Limited memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm, as proposed by Liu et al. [LN89].

logits Logits are the raw, non-normalised predictions of a neural network.

LR Logistic regression classifier.

LSTM Long-short term memory cells, as proposed by Hochreiter and Schmidhuber [HS97]

MFCC Mel Frequency Cepstral Coefficients, used to simulate the human hearing.

MNIST MNIST dataset [LeC98], consisting of 70,000 images of size 28×28 in grey scale, representing handwritten digits, available at: `http://yann.lecun.com/exdb/mnist/`

OOD By name, out-of-distribution (OOD) data is data, which differs from the observed distribution given by clean training data. Examples of OOD are adversarial samples, outliers, or data from a different dataset than the initial classifier was trained on.

oracle An oracle is a system, which is assumed to represent the truth, and a user can query to obtain certain input-output combinations, but no knowledge of the internal structure of that system is available.

PAP Projection of adversarial perturbations onto a street sign

PGD Projected gradient descent as proposed by Madry et al. [Mad+18] is an alternative name for the BIM.

ReLU rectified linear unit

RNN recurrent neural network

SGD stochastic gradient descent

SVHN Street view house number dataset [Net+11], consisting of 630,420 images (73,257 for training, 26,032 for testing, and 531,131 for optional additional training) of size 32×32 with 3 colour channels, representing house numbers between '0' and '9', available at: `http://ufldl.stanford.edu/housenumbers/`

Symbols

c The predicted class or label of a given sample.

c^* The ground truth class or label for a given sample.

C Number of classes or labels for a classification problem.

$\mathcal{D}$ Distance between two inputs arbitrary inputs $\mathbf{x}_1$,$\mathbf{x}_2$

f Arbitrary machine learning model trained for a specific task.

f^* Unknown model implementing the ground truth.

I Arbitrary input dimension $\in \mathbb{N}$.

J Arbitrary output dimension $\in \mathbb{N}$.

$\mathcal{L}$ Loss function to train a machine learning model.

L_0 L_0 distance between two inputs, calculated as $\sum_i \mathbb{1}\left(x_i \neq 0\right)$.

L_1 L_1 distance between two inputs, calculated as $\|\boldsymbol{x}\|_1 = \sum_i |x_i|$.

L_2 L_2 distance between two inputs, calculated as $|\boldsymbol{x}\|_2 = \sqrt{\sum_i |x_i|^2}$.

L_∞ L_∞ distance between two inputs, calculated as $\|\boldsymbol{x}\|_\infty = \max_i \left(|x_i|\right)$.

L_p General L_p distance between two inputs, calculated as $\|\boldsymbol{x}\|_p = \left(\sum_i |x_i|^p\right)^{\frac{1}{p}}$.

N Overall number of samples in a given dataset.

p Arbitrary dimension.

$\mathtt{p}$ A specific pixel, at position x,y, where x is the row, and y is the column of a given image.

$\mathbb{P}$ Probability distribution.

t Target class or label for an adversarial attack.

x Single input value.

$\mathbf{x}$ Arbitrary input vector.

$\mathcal{X}$ Dataset of samples, each consisting of a feature vector $\mathbf{x}$ and a corresponding output $\mathbf{y}$

$\mathcal{X}_{test}$ Subset of a dataset $\mathcal{X}$ to test a model on.

$\mathcal{X}_{train}$ Subset of a dataset $\mathcal{X}$ to train a model on.

$\mathcal{X}_{val}$ Subset of a dataset $\mathcal{X}$ to validate a model on.

$\mathbf{x}'$ Adversarial input vector.

$\hat{\mathbf{x}}$ Intermediate input vector to the hidden layers of a model.

y Single output value.

$\mathbf{y}$ Arbitrary output or prediction vector.

$\mathbf{y}^*$ The ground truth output or prediction vector for a given sample.

z Colour channel of an image.

α An iterative amount of perturbation added to an (intermediate) adversarial input.

θ Parameters of a model, including all weights and biases.

$\boldsymbol{\delta}$ Perturbation, which is applied to an original image to create an (intermediate) adversarial input.

ε Maximum allowed perturbation.

$\boldsymbol{\mu}$ Vector of mean values over all feature dimensions of an observed dataset.

ω Filter or kernel of a CNN

φ Activation function applied to the output of a neuron.

Σ Covariance matrix based on the features of an observed dataset.

$\boldsymbol{\sigma}$ Vector of standard deviation values over all feature dimensions of an observed dataset.