## **DLA: Dense-Layer-Analysis for Adversarial Example Detection**

Philip Sperl, Ching-Yu Kao, Peng Chen, Konstantin Böttinger Fraunhofer Institute for Applied and Integrated Security {philip.sperl, ching-yu.kao, peng.chen, konstantin.boettinger}@aisec.fraunhofer.de

#### Abstract

In recent years Deep Neural Networks (DNNs) have achieved remarkable results and even showed super-human capabilities in a broad range of domains. This led people to trust in DNNs' classifications and resulting actions even in security-sensitive environments like autonomous driving.

Despite their impressive achievements, DNNs are known to be vulnerable to adversarial examples. Such inputs contain small perturbations to intentionally fool the attacked model.

In this paper, we present a novel end-to-end framework to detect such attacks during classification without influencing the target model's performance. Inspired by recent research in neuron-coverage guided testing we show that dense layers of DNNs carry security-sensitive information. With a secondary DNN we analyze the activation patterns of the dense layers during classification runtime, which enables effective and real-time detection of adversarial examples.

Our prototype implementation successfully detects adversarial examples in image, natural language, and audio processing. Thereby, we cover a variety of target DNNs, including Long Short Term Memory (LSTM) architectures. In addition, to effectively defend against state-of-the-art attacks, our approach generalizes between different sets of adversarial examples. Thus, our method most likely enables us to detect even future, yet unknown attacks. Finally, during white-box adaptive attacks, we show our method cannot be easily bypassed.

#### 1 Introduction

Machine learning (ML) and especially deep learning (DL) applications transform modern technologies at an impressive pace. Research progress and the availability of high performance hardware enable the training of increasingly complex models. Such DL models have achieved even super-human results in a broad range of domains: From classical image classification tasks [70], to outplaying humans in Go [67], or even autonomously driving cars [10].

In numerous scenarios the security and safety are of crucial importance. Errors in the ML processing pipeline can affect our daily routine, lead to severe incidents in the users' health, or threaten future critical infrastructures. Such errors not only stem from inaccuracies in the training phase, but also from intentionally performed attacks. Kurakin et al. [36] for example showed the vulnerability of self driving cars and demonstrated a successful attack. Hence, the security of systems incorporating DL concepts is a major task for engineers, data scientists, and the research community.

Malicious actions aiming at DL models come in two flavours according to their attack timing. Poisoning attacks target the training phase, while *evasion* attacks are performed in the test phase. For *poisoning* attacks the attacker induces changes to the training dataset and especially to the labels to provoke misclassifications, see [52, 54]. As the training dataset is typically not available for attackers, the majority of recent work focuses on evasion attacks. Here the attacker manipulates the behavior of the DL model itself such that intended misclassifications occur. In 2014, Szegedy et al. [71] first demonstrated that small perturbations on images fed to a deep neural network (DNN) can provoke such a misclassification. Since then, new attacks and countermeasures have been introduced at a fast pace without the discovery of a fundamental and general defense strategy, yet. Perturbed inputs which successfully fool the target network are known as adversarial examples. In this paper, we propose an effective defense mechanism that detects such adversarial example attacks with high accuracy. Our approach generalizes between a broad range of state-of-the-art attacks and therefore does not only cover contemporary attacks, but will most likely also defend against future attacks. Further, our method defends against attacks in image classification, natural language, and even audio processing scenarios.

Currently, adversarial attacks seem to subdue corresponding defence methods. Research in this field is yet to provide a generally applicable solution to this problem, which motivates the work in this paper. Our main idea is based on observing neural activity during classification runtime. We were inspired by recent findings in the field of neural network testing and its interesting prospects. Pei et al. [59] introduced the idea of neuron coverage, which serves as a metric to guide testing of neural networks. Since then, further coverage metrics have been proposed and various testing techniques have made use of them [44, 69]. Odena and Goodfellow [55] reported promising results when applying concepts of coverage-guided fuzzing to neural network testing using neuron coverage.

These recent findings indicate that the neuron coverage of DL models carry security-sensitive information. This hypothesis at hand led us to the main insight of this paper: we show that neuron coverage exhibits a characteristic behavior when processing adversarial examples. In particular, adversarial examples provoke a unique pattern in the coverage such that respective inputs become detectable. Interestingly, this characteristic is independent of the attack method, as our results strongly indicate. With this observation we optimistically assume that our approach will also defend against future unknown attacks.

In summary we make the following contributions:

- We propose a general end-to-end method to detect adversarial examples generated using different state-of-the-art methods.
- We successfully detect adversarial examples in image classification, natural language processing (NLP) and DL-based audio processing.
- We implement and evaluate our approach to successfully detect prior unseen adversarial examples of various attack methods.
- Finally, we evaluate our method during adaptive attacks and achieve superior results compared to related methods.

The rest of this paper is organized as follows. In Section 2, we review related work and summarize latest findings on defense strategies against adversarial attacks. We present our main contribution, a novel concept of detecting adversarial attacks on neural networks, in Section 3. Sections 4 and 5 present a thorough proof-of-concept including experiments and evaluation of the according results. For future work on this topic, we plan to publish our code and used datasets. In Section 6, we discuss the cost of our method as well as the add-on analysis regarding the real-world application, transferability, and generalisation to future attacks. Finally, we conclude the paper with Section 7.

#### 2 Related Work and Background

In this section, we discuss related work and the theoretical background that forms the basis of our approach. We start with adversarial attacks and discuss corresponding state-ofthe-art defense methods.

## 2.1 Adversarial Attacks

We focus on test-time attacks exclusively and therefore define the following categories as introduced in [16]. The different evasion attack types differ in the amount and nature of information available for the attacker. In *white-box* attacks the attacker has full control over the target which includes knowledge about the architecture and parameters of the trained model. Hence, the adversary is able to deliberately craft adversarial examples exploiting the knowledge of the model. Contrary to that, in *black-box* attacks, the attacker has neither knowledge of the target model architecture nor access to the parameters after training. In the following, state-of-the-art attacks belong to the class of *white-box* methods. The aim of an evasion attack is to generate an adversarial example that is misclassified by the targeted DL model. More formally:

**Definition:** Adversarial Examples. Let  $f(\cdot)$  be a trained neural network used for classification tasks. Let  $H(\cdot)$  be a human oracle with the same classification capabilities. Assume that for a given legitimate input x the following equation holds:

$$f(x) = H(x)$$

Let x' be a mutated version of x that is close to x, i.e.,  $||x' - x|| \leq \varepsilon$  for some small  $\varepsilon \in \mathbb{R}^+$ . Then x' is an adversarial example, if the following holds:

$$H(x) = H(x') \land f(x') \neq H(x').$$

Informally, adversarial examples are slightly mutated versions of their original counterparts that lead the targeted network to misclassification.

Szegedy et al. [71] first demonstrated the vulnerability of neural networks to slightly mutated inputs. The authors formulated the problem of finding those mutations with a minimization problem. To solve this problem the authors used a box-constrained *L-BFGS* [21].

In 2014, Goodfellow et al. [24] picked up the previous findings and proposed their resulting "Fast Gradient Sign Method" (*FGSM*).

Kurakin et al. [35] proposed the "Basic Iterative Method" (*BIM*). In this attack, the inputs are mutated based on single steps which aim to increase the loss function. After each step the direction is adjusted.

Madry et al. [47] further refined the approach. The authors showed the BIM attack being equivalent to "Projected Gradient Descent" (*PGD*). By making use of the  $L_{\infty}$  version of this standard convex optimization method the authors further improved the previously shown BIM.

Moosavi-Dezfooli et al. [51] proposed *DeepFool*, which generates adversarial perturbations by iteratively pushing the inputs towards the decision boundary of the attacked network. In order to model the decision boundary in a simplified manner, it is linearized and represented using a polyhydron.

The majority of current attacks are restricted by the  $L_{\infty}$  or  $L_2$  norm between benign and adversarial examples. In contrast to that, Papernot et al. [56] proposed in their "Jacobian-based Saliency Map Attack" (*JSMA*) to restrict the perturbations with respect to the  $L_0$  norm. Hence, the attack tries to minimize the amount of input points being changed rather than restricting the global change to the input.

This idea was picked up by Su et al. [68]. In this publication the authors successfully fooled DNNs using their *One Pixel Attack*.

Currently the most powerful attack was proposed by Carlini and Wagner (C&W) in [11]. This method is capable of crafting adversarial examples even for targets protected by state-of-the-art defense methods. The basic idea of the attack is, instead of optimizing the loss function directly, to introduce a cost function  $f_y$  as substitute.

Moosavi-Dezfooli et al. [50] presented *Universal Adversarial Perturbations*. Rather than calculating individual adversarial examples, the authors calculated a universal perturbation such that when added to an arbitrary input, the target network is fooled.

If the attacker does not have access to the target model and its parameters, black-box attacks still pose an alternative to manipulate the classifications. In this paper, we focus on *Transfer Attacks* exclusively, when confronted with a blackbox situation. Here, the attacker uses a neural network over which she has full control and creates adversarial examples for it. The attacker then transfers the resulting examples to the actual target to provoke a misclassification.

#### 2.2 Defenses against Adversarial Attacks

Akhtar and Mian [8] categorize adversarial defenses using three classes. The first class introduces a modified training procedure or various prepossessing methods to the input data, respectively. In the second and third class, modifications to the targeted model itself or an additional model are introduced to increase overall robustness. For both classes, some techniques aim to increase the robustness by detecting adversarial examples. As we propose a new technique to achieve the same goal, we sum up related methods into a fourth class. In the following we make use of this categorization and present previous findings for each class and explain them briefly. We pay special attention to the methods trying to detect adversarial examples. For further analysis of state-of-the-art detection techniques we refer to the survey by Carlini and Wagner [14].

#### 2.2.1 Changes to the Training Process or Input Data

Adversarial training: The most intuitive and widely performed defense technique is to include adversarial examples in the training phase of the model to protect. This is achieved by simply extending the training set with adversarial examples [36]. Adversarial training is often introduced by authors of attacks as the first strategy to prevent a successful attack [24, 51, 71].

The authors of [24] proposed training based on a modified objective function. The idea is to force the prediction of adversarial and benign images of one class to the same direction. Additional regularization avoids over-fitting, which again increases the robustness of the network against unseen adversarial examples [24, 64].

In 2017 Madry et al. [47] interpreted adversarial defense as a robust optimization problem. The authors claimed the PGD attack method to be a universal attack as it supposedly makes use of the local first order information about the target network in a superior way compared to other attack techniques. Hence, the authors used examples created with PGD during the adversarial training. The resulting networks are robust against other adversaries, which is also shown in [13].

As adversarial training is easy to implement it may act as a first line of defense against known attacks. Nevertheless, it should not be used as the single approach to protect against adversaries. Moosavi-Dezfooli et al. [50] showed that adversarially trained models are still vulnerable using other known attack methods. Moreover, Tramèr et al. [34] presented a twostep attack method which also circumvents security provided by adversarial training. The final drawback of adversarial training is the fact, that it is prone to black-box attacks [53,58].

**Data compression and feature squeezing:** Dziugaite et al. [19] first showed that adversarial images created with the FGSM method can be classified correctly if JPG compression is applied. Based on this finding, further experiments using JPG and JPEG compression resulted in successful defense methods [17, 26]. However, Shin and Sing [66] showed that a considerable amount of adversarial images are not vulnerable to a JPEG compression, especially when crafted with the C&W method.

Similar strategies have been proposed in [74] and [40]. Here "Feature Squeezing" is used to reduce the complexity of the inputs, by reducing the color depth or applying smoothing filters. Additionally, adversarial example detection can be conducted which we will discuss in Section 2.2.4.

The disadvantage of using the above mentioned techniques prior to the classification is a decreasing classification accuracy. Since no prior knowledge about the images is given, each has to be compressed before being classified, resulting in a information loss for benign images.

**Data randomization reprocessing:** Luo et al. [42] proposed to apply the targeted neural network only to a certain region of the currently classified image. This technique is shown to be a valuable countermeasures against adversarial images created by L-BFGS and FGSM based algorithms. Xie et al. [73] analyzed the effects of random resizing and padding. The authors reported positive effects on the classification accuracy of adversarial images. Similarly, Wang et al. [72] made use of a separately executed data-transformation module, which partially removes adversarial perturbations.

2.2.2 Modifying the Network

In recent years, a substantial increase of DL robustness was achieved by making changes directly to the network. The inputs remain unchanged which reduces preprocessing time.

Gradient Hiding limits the accessibility of the gradients and successfully circumvents associated attacks. Nonetheless, this technique does not provide protection against black-box attacks, as shown in [58].

Related to Gradient Hiding, Ross and Doshi-Velez [61] introduced Gradient Regularization. The authors proposed to penalize the degree of variation of the output, based on changes in the input. This concept led to further techniques like [43] and [65].

In 2015, Papernot et al. [57] presented Defensive Distillation. The originally introduced distillation technique shown by Hinton et al. [28] aims to simulate a neural network using a smaller one. In contrast to that, the authors try to generate a smoother, less sensitive version of the original model. This is achieved by reusing the probability vectors of the training data during the training of the model. In 2017, Papernot and Mc-Daniel further improved the concepts conveyed in the initial publication. Nonetheless, Carlini and Wagner [11] claim their C&W attack to be successful against Defensive Distillation.

#### 2.2.3 External Network Add-Ons

Akhtar et al. [7] proposed the idea of Perturbation Rectifying Networks (PRN). These sub-networks are added in front of the original network and are trained separately after the actual training phase. The PRN rectifies the perturbations on the adversarial images, which are subsequently identified by an additional detector.

Since the 2014 released paper by Goodfellow et al. [23], Generative Adversarial Networks (GANs) are widely used and referred to in numerous publications. Some promising publications using GANs to protect DNNs against adversarial attacks are [31, 38, 63].

#### 2.2.4 Detecting Adversarial Examples

Our concept can be added to this class of defense strategies, hence, we provide a detailed overview of the latest related findings. As mentioned before, detection techniques can be based on both, changes to the input data or to the model itself. Additionally, observations of the model behaviour or the model's input provide insights on whether processed inputs are of adversarial nature or not. In Section 2.2.1 we showed various preprocessing and compression methods which can be applied in order to reduce the effects of adversarial perturbations. Moreover, these methods can additionally be used to detect attacks.

Baluja and Fischer [9] showed this by using the so-called feature squeezing technique: the authors created different versions of the input, based on different squeezing methods and let the target network classify them. If the returned labels differ, the authors assume this input to be adversarial. In a follow-up work, Xu et al. [74] used this technique to protect networks against the C&W attack.

Similarly, Hendrycks and Gimpel [27] performed a principal component analysis (PCA) on the inputs of neural networks. The authors found that for adversarial examples, a higher weight is placed on larger principal components in comparison to benign examples. With this knowledge, a binary classifier can detect attacks.

Liang et al. [40] interpreted adversarial perturbations as noise and tried to detect them by using scalar quantization and smoothing filters.

A more straightforward approach was evaluated by Gong et al. [22]. By applying a binary classifier on the input examples directly, the authors were able to detect adversarial input among benign examples. Positive results were achieved using the MNIST dataset exclusively, during a later analysis in [14] the approach failed to reach similar detection rates for different datasets.

Meng et al. [48] proposed their framework MagNet, which evaluates the original dataset and analyses the manifold of the benign examples. If a new examples is passed to the network to be classified, it is compared to the findings about the manifold. This method is shown to be vulnerable against attacks incorporating larger perturbations shown in [15].

A comparable pre-classification was introduced by Grosse et al. [25]. The authors used the maximum mean discrepancy test, based on sets of benign and adversarial examples. This test provides evidence on whether the two sub-datasets are drawn from the same distribution or not.

Hosseini et al. [30] added a new class to the used dataset and try to unify adversarial examples in it. During training, the network is set to assign adversarial images to this so-called NULL class.

Metzen et al. [49] showed a method by adding a subnetwork to the original neural network. This sub-network is adversarially trained and acts as a binary classifier during the classification of the inputs. In [41], the authors showed that this method can again be bypassed by an attack.

Lu et al. [41] hypothesized that adversarial examples produce a pattern of Relu activation values in the late stages of a target network which differ from those based on benign examples. In their framework called SafetyNet, the authors used a radial basis function support vector machine (SVM) to distinguish between original and perturbed examples.

Trying to increase the security of convolutional neural networks (CNNs), Li et al. [39] extracted the intermediate values after convolutional layers. The authors performed a PCA of the extracted features and a cascaded classifier to detect attacks.

In 2017, Feinman et al. [20] tried to detect adversarial attacks using two features which they extracted from dropout neural networks. With these features a simple logistic regression is performed as the basis for a binary classifier. The first feature the authors introduced is the density estimate, based on which the distance between a given example and the submanifold of a class is quantified. For this purpose the authors used the feature space of the last hidden layer of the target network. With their second feature, the Bayesian uncertainty estimate, the authors introduced an alternative feature to detect adversarial examples missed by the first feature. Here, points shall be detected which lie in low-confidence regions of the original input space, indicating an attack.

Similar to our method Ma et al. [45] detect attacks by observing the NN's hidden activations. The authors identify two exploitation channels which form the basis of their detection approach. By extracting provenance and value invariants attacks are detected using a one-class SVM. Compared to this method, we propose a fully automated end-to-end system without further feature engineering steps

## 3 Methodology

In this section we introduce our main concept of detecting adversarial examples during classification time. The core idea originates from our hypothesis as initially indicated in Section 1: Adversarial examples provoke the dense layer neuron coverage of neural networks to behave in such a distinctive manner that attacks become detectable by observing their activity patterns. We provide a detailed description on how to expand and build upon this idea in the following.

Fig. 1 shows an overview of the overall process. In the following, we discuss each step in detail. Joining these steps provides an end-to-end pipeline for fully automated detection of adversarial examples.

Our method is designed to help developers and maintainers of neural networks to secure their models against attacks. Hence, we assume access to the fully trained model as well as (read-only) access to the benign training dataset  $D_{benign}$ . We refer to the model that we want to secure as the *target model*  $N_{target}$ . Our overall aim is to generate a secure model  $N_{target}^{secure}$  that throws an alarm signal whenever an adversarial example is being processed. To achieve this, we first generate adversarial examples, second extract the dense layer neuron coverage, and third train an alarm model that enables secure operation of our initial target model.

#### 3.1 Generating Adversarial Examples

In the first step of our concept, we generate adversarial examples  $D_{adv}$  for our target model  $N_{target}$ . We craft these examples in a white-box manner by exploiting all available information.

It is important to note that we create adversarial examples for each class of the dataset. Hence, we try to push the generated adversarial examples to be misclassified with an equal distribution among all remaining (i.e. false) classes. This is a crucial step during the generation phase in order to cover all possible cases which might occur during the application of our method in the field.

Once the adversarial examples are generated, we summarize them in a dataset  $D_{adv}$ . The outputs of the adversarial example generator, i.e., the elements of  $D_{adv}$ , are labeled as *adversarial*, while the original unmutated samples  $D_{benign}$  are labeled as *benign*.

For the adversarial example generation, we use a wide range of adversarial crafting methods, including state-of-theart techniques. As we discussed in Section 2, the attacks do not only differ in success rates but also in their detectability. Hence, by covering the currently strongest attacks we try to circumvent this issue. Moreover, to cover the case of black-box attacks, we recommend to use transferred adversarial examples as well. It is important to note that only these examples should be considered that actually provoke a misclassification of the target network.

#### **3.2 Extracting Dense Layer Neuron Coverage**

This step of our concept can be viewed as an additional preprocessing phase prior to the application of the target model in its intended environment. We refer to this step as feature extraction. Here, the datasets  $D_{benign}$  and  $D_{adv}$  are fed to the well-trained target model which performs classifications using the individual samples. Since the feature extraction is not part of the actual function and objective of  $N_{target}$ , we ignore its classification outputs. Instead, we extract the activation values of all available dense layers and concatenate them to one sequence. The resulting datasets, which hold the sequences for all samples, are called  $I_{benign}$  and  $I_{adv}$ , respectively. For further usage of the datasets, we adopt the labels to distinguish between adversarial and benign samples. In summary, the dataset  $I_{<attackname>}$  holds the activation value sequences for all benign and adversarial examples regarding the target model for one specific attack method. We preserve this separation of the activation value sequences, because we assume the different attack methods to have characteristic impacts on the behavior of the target and the resulting features. This not only enables us to detect the individual attacks, but also to assess the impact of the individual crafting methods.

#### 3.3 Training an Alarm Model

The dense-layer neuron coverage we extracted in the previous step builds the basis for our core idea to detect adversarial examples. Assuming that this coverage contains information about the model, its behaviour, and the input sent to it, we need a supplementary analysis of the extracted information.

Previous work [20], as discussed in Section 2.2.4, followed a related idea. The authors try to extract information from neural layers and further manually process them to detect adversarial images. However, taking information directly from all dense layers of the trained model is more efficient for



Figure 1: Overview of our concept showing the required neural networks, datasets, and calculations.

providing an end-to-end solution without further processing steps.

Accordingly, we propose the following method which generalizes well over different scenarios and model architectures: We interpret the analysis of the dense layer features as a binary classification. Instead of including hands-on measures and distinguishing between different scenarios, we train an additional neural network to perform the required actions. We refer to this network as the *alarm model* N<sub>alarm</sub> in the following.

To train the alarm model, we use the features stored in  $I_{<attackname>}$ . Therefore, the network is trained to distinguish between activation values gathered during the classification of benign and adversarial features, respectively. In the actual secure operation phase,  $N_{alarm}$  performs a binary classification of newly extracted features provoked by the current samples to classify. This forms the final adversarial example detection running alongside  $N_{target}$ .

The architecture of the alarm model heavily influences the success of our approach. Furthermore, different architectures need to be tested against each other to provide a viable solution. In Section 4, we give a recommendation for a specific architecture. Still, future work needs to further evaluate this part of the concept.

Note that we recommend to create one alarm model for each introduced attack method. The attack methods differ in their approach and complexity and thus influence the neuron activation patterns distinctively. Hence, using a set of different alarm models allows us to detect a broader range of attacks. Furthermore, we are able to evaluate the capability of each alarm model version of detecting different attack methods. This provides information on the applicability of our concept when detecting future attack methods.

#### 3.4 Concept Overview

After we introduced the building blocks of our concept, we can now link them and present an overview of our approach with Algorithm 1. The application of our method in a real-world scenario can be divided into two steps. A prior initialization phase prepares our framework to enable a secure operation of the target model  $N_{target}^{secure}$ .

In the initialisation phase, we create adversarial examples and perform the according feature extraction steps. We have shown the importance of using different attack methods to create the adversarial examples. This ultimately leads to an assemblage of alarm models, each capable of detecting adversarial examples created by one specific attack method.

During the secure operation of the target model, we continuously extract the features during classification of new, unseen samples. The resulting activation sequences are fed to all available alarm models to perform binary classifications. If the classifications indicate an attack, our framework throws an alarm signal and a human expert is consulted to evaluate the current input. Here, the maintainer chooses if one assumes an attack based on one or more alarm signals, majority votes, or all alarm models synchronously indicating such an event.

#### 4 Implementation and Experimental Setup

In the following, we present details regarding our proof-ofconcept implementation and our experimental setup. We discuss our choice of datasets and accompanying target model architectures on which we tested our approach. The description of feature extraction and the following alarm model training form the core of this section. We introduce one exemplary alarm model implementation to finally detect adversarial examples. Subsequently we present the test scenario for which **Input:** D<sub>benign</sub>, D<sub>test</sub>, N<sub>target</sub>, N<sub>alarm</sub> Result: Nsecure target for Initialization do  $D_{adv} \leftarrow \text{CreateAdvExamples}(D_{benign}, N_{target});$  $I_{benign} \leftarrow \text{ExtractInformation}(D_{adv}, N_{target});$  $I_{adv} \leftarrow \text{ExtractInformation}(D_{benign}, N_{target});$  $N_{alarm} \leftarrow \text{Train}(N_{Alarm}, I_{benign}, I_{adv});$ end while Secure Operation do while 1 do  $x \leftarrow \text{Sample}(D_{test});$  $y_{target} \leftarrow \text{Classify}(N_{target}, x);$  $i_{y_{target}} \leftarrow \text{ExtractInformation}(x, N_{target});$  $y_{alarm} \leftarrow \text{Classify}(N_{alarm}, i_{y_{target}});$ if  $y_{alarm} == 1$  then Alarm ; ConsultHumanExpert(); end end end

**Algorithm 1:** Main algorithm, divided in an initialization and a secure operation phase.

we show the evaluation results in the next section. This includes details regarding our test environment. Finally, we introduce a series of extension test scenarios which are not part of the actual proof-of-concept. Instead, we try to provide the reader with a better intuition for our approach and rule out possible restrictions. We motivate generality of our main hypothesis as discussed in Section 1 and the resulting idea of this paper. For this purpose, we included tests for noisy images, two special target model architectures, and two additional dataset types, namely a natural language and an audio dataset. Moreover we evaluate adaptive attacks, in which the attacker has full knowledge and control over the target and alarm model.

In this section of the paper, we solely covey information on the executed experiments. Hence, we refer to the next chapter for the respective results. We divided the proof-of-concept into two sections in order to preserve the paper's readability.

#### 4.1 Datasets

For proof-of-concept, we considered the MNIST [18] and CIFAR10 [33] image datasets. We decided to do so based on the following two observations: First, to allow a comparison of our method and state-of-the art defense techniques. Second, the usage of image datasets enables us to better visualize the adversarial examples and evaluate the performance of different attack methods.

The MNIST dataset consists of 70000 handwritten digits ranging from 0 to 9 of which 60000 build the training set and 10000 the test set. Each digit is represented by  $28 \times 28$ 

grey-scale pixels.

CIFAR10 consists of 60 000 colored images of which again 10 000 images build the test set. Each image is stored using  $32 \times 32 \times 3$  pixels, which makes this dataset more difficult to classify. The CIFAR10 dataset therefore enables us to perform tests close to a real-world scenario.

To prove detectability of adversarial examples in the natural language processing (NLP) context, we used the IMDB dataset of movie reviews [46]. Both, the train and test set, contain 25000 samples each. For both subsets, positive and negative reviews are distributed evenly.

The audio examples we considered during our tests are drawn from the Mozilla Common Voice dataset [3], which contains 803 hours of recorded human sentences. Contrary to the above mentioned datasets, the instances in the Common Voice dataset are used for speech-to-text conversions rather than being classified according to known classes.

#### 4.2 Target Models

Throughout the proof-of-concept, we payed attention to use state-of-the art target models in order to keep a close relation to real-word scenarios. Table 1 sums up the used models and shows their training and test accuracy as well as a short summary on the individual architectures. For MNIST, we chose LeNet [37] and a simple Multi-Layer-Perceptron (MLP) [2] that we refer to as kerasExM. In contrast, for CIFAR10 we decided to utilize ResNet [6] and a deep CNN [1], we refer to as kerasExC.

Furthermore, in order to evaluate if our method can generally be applied to a wide range of DL architectures, we conducted experiments using the two following examples. On the one hand, we included a Long Short Term Memory (LSTM) based target model. This architecture achieves a remarkable performance, especially in image classification tasks. On the other hand, we included a so-called Capsule Network (Capsule NN), which is supposed to be more robust than conventional models.

For our tests regarding the NLP dataset, we again used an LSTM target model. We chose a pre-trained DeepSpeech implementation (0.4.1) [4] during our tests on audio examples. Hence, we did not add its training and test accuracy to Table 1. This neural network converts speech in the form of audio files to according text.

#### 4.3 Adversarial Attack Methods

We evaluate the detectability of the following attack methods: FGSM, C&W, DeepFool, PGD, and BIM. The motivation to do so originates from the nature and popularity of these methods. We include diverse attacks, such that remarkable differences in the basic idea can be seen. Moreover, we payed attention to add attacks which differ in strength and complexity. The C&W attack, for instance, is currently considered to Table 1: Target models used during experiments. Showing details on the architectures and performance. Capsule and ResNet are trained using the *adam* optimizer [32]. The remaining models are trained with *stochastic gradient descent*. The DeepSpeech model is pre-trained.

Dataset	Model Name	Model Details	Training Accuracy	Test Accuracy
MNIST	LeNet [37]	<ul> <li>2 convolutional layers with filter size 5</li> <li>– each convolutional layer is followed by a max-pooling layer with size 2</li> <li>– 2 dense layers after each max-pooling layer</li> </ul>	0.976	0.987
	kerasExM [2]	– one hidden layer with 512 neurons	0.972	0.985
	CapsuleNN [62]	- 10 capsules each of size 6	0.992	0.991
	LSTM [29]	<ul> <li>1 LSTM layer followed by two dense layers with 64 and 32 neurons</li> </ul>	0.975	0.978
CIFAR10	kerasExC [1]	<ul> <li>4 convolutional layers with filter of size 3</li> <li>each pair of convolutional layers followed by a max- pooling layer of size 2</li> <li>last hidden layer of dimen- sion 512 is fully connected.</li> </ul>	0.888	0.889
	ResNet [6]	- 3 blocks followed by an average pooling size of 8	1.000	0.840
IMDB	LSTM (for NLP)	<ul><li>1 embedding layer</li><li>1 64-neuron dense layer</li></ul>	0.996	0.81
Mozilla Common Voice	DeepSpeech [4]	<ul> <li>– containing 2 parts</li> <li>– convolutional and recurrent network</li> </ul>	_	_

be the most powerful attack. Hence, this and future adversarial detection schemes need to be tested against this method. Fig. 2 shows a series of adversarial images for both datasets crafted with the above mentioned techniques.

Alongside the five stated attack methods, we include one *black-box* attack as well. We create adversarial images in a *white-box* setup on model *A* and try to fool model *B* with the resulting examples. We call this *transfer* attack in the following.

Since the actual crafting and implementation of the attacks is not part of our concept, we applied the *foolbox* framework [60] to generate adversarial examples for MNIST and CIFAR10. To create adversarial examples based on the Common Voice dataset, we refer to [12]. Finally, for the IMDB dataset we created an algorithm to create adversarial examples, which we briefly describe in Appendix D. In future work, we will further explore and refine this attack method.

#### 4.4 Alarm Model Architecture and Training

During our proof-of-concept, we exclusively used one specific alarm model architecture to detect adversarial examples. We chose to do so in order to show the generality of our concept and keep the space of tunable parameters as small as possible. Hence neither the used dataset nor the applied target model, which needs to be protected, affect our alarm model architecture. For future work or in real-world applications, a deliberate choice of the alarm model will likely further improve the strength of our concept. In this paper, we use a DNN with six dense layers which we train for ten epochs and a batch-size of 100 in each scenario. We use the *adam* optimizer [32] during training. Note that in some cases this alarm model suffers from underfitting. We included a more detailed description of the architecture in Appendix A.

#### 4.5 Main Test Scenario

We have discussed all building blocks of the experimental setup and the according preliminaries, in this section, we focus



Figure 2: Adversarial images created with (from left to right): FGSM, C&W, DeepFool, PGD, BIM. The top images are based on the MNIST dataset and are crafted on the target model LeNet. The bottom images are based on the CIFAR10 dataset and are crafted on the target model kerasExC.

on the actual test scenario and performed actions. We assume ourselves in the position of the trained model's maintainer and try to increase its trustworthiness. Each step we present is performed for all introduced attack methods. For the sake of simplicity we neglect this fact in the following and show each step once.

In the first step, we craft adversarial images using the above stated methods. Since we have full control over Ntarget and its parameters, we perform white-box attacks. During this process, we pay attention to the way the datasets have been split beforehand. Consequently, we create two separate adversarial datasets, based on the train and test subsets, respectively. By doing so, we simulate the real-world case in which the maintainer only possesses the training data. The samples in the test set simulate inputs fed to the target during its usage in the field. Additionally, we can later check detectability of adversarial examples which are based on unseen benign inputs to rule out a detection bias. We refer to the datasets as  $D_{adv}^{train}$  and  $D_{adv}^{test}$ , respectively. For both, MNIST and CIFAR10, we create adversarial counterparts of  $D_{benign}^{train}$  and  $D_{benign}^{test}$  containing a similar amount of examples: (60000, 10000) for MNIST and (50000, 10000) for CIFAR10. We let Ntarget classify all samples in the four datasets and store the activation sequences in  $I_{benign}^{train}$ ,  $I_{benign}^{test}$ ,  $I_{adv}^{train}$ , and  $I_{adv}^{test}$ . Each individual set contains

features extracted during the classification of benign and adversarial samples while we preserve the division between test and training samples. Note that we neglected this split of the datasets in Algorithm 1 for the sake of simplicity.

In the second step, we use  $I_{adv}^{train}$  and  $I_{benign}^{train}$  to train the alarm model. Hence, for each target model and attack method, we create one specific alarm model which we call  $N_{alarm}$ . To test our capability of detecting adversarial examples, we let  $N_{alarm}$  classify all samples in  $I_{benign}^{test}$  and  $I_{adv}^{test}$ .

To further show the generality of our concept, we conduct cross-testing cases. This means that we test a trained alarm model against the features of a different attack. With this, we simulate the scenario in which we encounter a new and yet unknown attack.

Furthermore, we create a combined alarm model  $N_{alarm}^{combined}$  which is trained using all features gained during the extraction of a specific target model. Here, we verify if considering more information, based on a wider range of attacks, improves the alarm model's performance and provides a stronger setup in the context of our concept.

## 4.6 Additional Experiments and Adaptive Attacks

At the beginning of this section, we briefly introduced the supplementary tests we conducted to further establish confidence in our approach. These tests can be divided into four parts.

In the first part, we used the previously created adversarial images for the MNIST and CIFAR10 datasets and conducted transfer attacks targeting an LSTM neural network and a Capsule network, respectively. This experiment gives insights on whether our approach can be applied in the context of different DL architectures or not.

In the second part, we run two experiments on the regular target models classifying MNIST and CIFAR10 images. With the first test we analyze if our concept is robust to noisy input images. Hence, we exclude the possible effect in which our framework is solely able to distinguish between clean, benign images and adversarial images containing noise. We achieve this by creating noisy benign images with the same amount of distortion as their adversarial counterparts. For this purpose we calculate the distances between the original and adversarial images with respect to the used distance metric of the attack. The resulting datasets now contain original, adversarial, and benign noisy images.

Subsequently, we provide evidence for our initial hypothesis of the paper presented in Section 3. We analyze the dense layer activation values of the target models during misclassification of original inputs. Thus, we extract the according features and treat them as activation values gained during classification of adversarial examples. The features are then used to train an alarm model which tries to detect misclassifications during testing of the benign dataset. In the third part, we test our concept in the context of two additional types of datasets. We investigate if we are able to detect adversarial examples in NLP and audio datasets. This test gives first evidence on the applicability in numerous DLbased environments. State-of-the-art defense methods mostly focus on image processing target models. Therefore, proving the applicability of our concept in additional types of datasets poses a significant step towards more robust defense methods. We introduce the test environment and results in a stand-alone paragraph in Section 5.4.

Finally we evaluate adaptive, white-box attacks in which the attacker has perfect knowledge of the target model and our detection method. As shown by Carlini and Wagner [14], the majority of proposed detection methods can easily be bypassed by an adaptive attack. To perform such an attack, we require a loss function based on the new target N<sub>secured</sub>, which consists of the original target model  $N_{target}$  and our alarm model  $N_{alarm}$ . We use the loss function defined by Carlini and Wagner [14], which the authors use to attack similar detection-based defense methods. Here, we profit from the code the authors published to reproduce their results [5]. Thus, we generate adversarial examples for N<sub>secured</sub>, using the C&W method exploiting perfect knowledge of the target and alarm model. We perform this attack on the secured version of LeNet for MNIST and on the secured version of kerasExC for CIFAR10.

### 5 Evaluation

In this section, we sum up the evaluation results. We split this into three parts. First, we discuss our analysis of the extracted features and show their distribution for one specific example. Second, we present the main results accumulated during our experiments. This includes the performance of the different alarm models while detecting adversarial examples. Third, we present the results we gained during our supplementary experiments including adversarial example detection in NLP and audio datasets and adaptive attacks.

#### 5.1 Feature Analysis

As the extracted features are the core of our hypothesis and concept, we illustrate some findings during the analysis. In Fig. 3 we show neuron activation sequences for the LeNet target model for all attack methods after we reduced the dimensionality of the data using PCA and t-distributed stochastic neighbor embedding (t-SNE). The figures show the neuron coverage of the dense layers during the classification of benign and adversarial images. The red dots and gray crosses represent the benign and adversarial instances.

We can clearly see a difference in the dense-layer activation values. Interestingly, we can see artifacts of the ten classes of the MNIST dataset in the t-SNE figures. This finding gives first evidence on the verity of our initial hypothesis. Furthermore, we can show a first estimate for the complexity and detectability of the attack methods. The PCA data points of the C&W-based activation sequences overlap to a higher extent than for the remaining methods. This indicates a more challenging detection of the C&W attack.

Nonetheless, since we want to provide an end-to-end framework to detect adversarial examples, we directly use the raw extracted data.

#### 5.2 Detection Performance

In this section, we present the performance of our concept using the differently trained alarm models. We assess the success of our detection method with the f1-score. Furthermore we present the the mean *false positive* and *false negative* rates.

During the proof-of-concept, we conducted numerous experiments. For the sake of simplicity and readability, we exclusively include results significant for the proof of our main idea and hypothesis. We provide the remaining results, tables, confusion matrices and figures in the appendix of this paper.

In Table 2, we list the f1-scores of the individual alarm models when tested against their dedicated attack method. We can see a strong detection capability for all attack methods and target models regarding the MNIST dataset. The respective f1-scores range above 0.9. For the CIFAR10 dataset our framework detects the majority of examples and poses a viable solution for real-world applications.

To evaluate if a combination of features while considering different attack methods at once improves the performance of our method, we created a combined alarm model. Hence, for each target model, its combined alarm model is trained with features extracted during the evaluation of all attack methods. Table 3 provides an overview of this experiment. The f1-scores show that the combined alarm models are able to detect all tested adversarial attack methods. Comparing the results to Table 2, we can report an superior performance of this combined method.

Above, we indicated the evaluation of cross-tests. This corresponds to analyzing if the concept is capable of detecting new, unseen adversarial attacks. In summary, for both datasets and for each target model we tested seven alarm model versions against six attack methods. Five of the seven alarm models are based on the attack methods FGSM, C&W, Deep-Fool, PGD, and BIM. Two additional alarm models are the combined one, and the alarm model trained using features extracted during transfer attacks. The result-space of the crosstests exceeds the frame of this paper. Hence, with the following two tables, we express the performance of our approach during cross-tests with mean values. We added the individual result values to Appendix B. Table 4 shows the mean performance of the individual alarm models, when tested against all attack methods. In Table 5, we show the mean detectability of the individual attack methods, when utilizing all alarm models



Figure 3: Visualization of the extracted features during the classification of MNIST-based adversarial and benign images for the LeNet target model. For better visualisation, the dimensionality of the features were reduced using PCA and t-SNE, respectively. Each column shows the plots for one attack method. The red dots and gray crosses represent the benign and adversarial samples, respectively.

Table 2: F1-Scores of the individual alarm models when tested against their underlying attack method. Main results of the evaluation.

Datacat	Target Model	f1- Score	f1- Score of the Alarm Models with the according attacks:										
Dataset	iuiget mouel	FGSM	C&W	DeepFool	PGD	BIM							
MNIST	LeNet	0.988	0.977	0.981	0.991	0.990							
MINIS I	kerasExM	0.992	0.975	0.982	0.993	0.991							
CIEAD10	kerasExC	0.847	0.733	0.855	0.843	0.852							
CIFARIO	ResNet	0.815	0.727	0.833	0.833	0.832							

(except the model based on transfer attacks). With the results we show that we are able to detect adversarial examples, for which the underlying method has not been known beforehand. Thus, our method most likely enables us to detect even future, yet unknown attacks.

In the following we present the mean error rates during our evaluation. The individual values can be found in Appendix C. For MNIST, the mean *false positive* and *false negative* ratios are 0.02 and 0.01, respectively. Similarly, for CIFAR10 we report the mean ratios of 0.22 and 0.16, respectively. For both examples we observe a higher *false positive* ratio. Hence, our method does not miss a disproportionate amount of adversarial examples. This is an important finding with regard to the applicability in a real-world setup.

#### **5.3 Results of further Experiments**

During our tests with a Capsule and an LSTM target network, we were able to detect adversarial images based on the MNIST dataset with f1-scores of 1.000 and 0.968, respectively. The positive results emphasize the applicability of our concept on a wide range of neural networks using dense layers.

In Table 6, we show the results of the tests containing noisy images. The performance of the individual alarm models are

decreased by 10% in the worst case. Hence, our method is robust against noisy inputs.

As the core hypothesis of our paper is strongly correlated to adversarial inputs, we introduced the tests regarding misclassified images. We argued that the adversarial inputs provoke distinctive neuron coverage patterns, enabling detecting. This excludes ordinary misclassifications of benign images due to inaccuracies in the model. Tests in which we treated a misclassification the same way we treated attacks, provide evidence for our hypothesis. In the tests we were not able to detect regular misclassifications with our framework.

#### 5.4 Experiments on Text and Audio Datasets

With the following test, we evaluate the generalizability of our method to different application domains. During this analysis, we craft adversarial examples based on an NLP and audio dataset respectively. As previously introduced, we use the IMDB and Mozilla Common Voice datasets. Subsequently, we assess whether our framework is able to detect adversarial examples when classified by the according target models.

The process of generating adversarial examples for the two datasets is not part of the contribution of this paper. Nevertheless, some basic notes are worth mentioning. Regarding

Table 3: F1-Scores of the combined alarm models when tested against each attack separately. The right most column holds the f1-scores of the combined alarm models when tested with a combined dataset containing all used attack methods.

Datasat	Target Model	f1- Scor	f1- Score of the combined Alarm Model with:										
Dataset	Turget mouer	FGSM	C&W	Deep Fool	PGD	BIM	Transfer	Combined					
MNIST	LeNet	0.981	0.973	0.974	0.981	0.981	0.931	0.993					
	kerasExM	0.984	0.972	0.977	0.984	0.984	0.947	0.993					
CIEAD10	kerasExC	0.771	0.741	0.771	0.772	0.772	0.754	0.932					
CIFARIO	ResNet	0.812	0.687	0.818	0.820	0.819	0.791	0.864					

Table 4: Mean f1-scores of the individual alarm models when detecting all attack methods.

Dataset	Target Model	Mean f1	Mean f1- Score of the individual Alarm Models with all attacks, model name:										
Dataset	laiget mouel	FGSM	C&W	DeepFool	PGD	BIM	Transfer	Combined					
MNIST	LeNet	0.918	0.945	0.940	0.878	0.885	0.950	0.970					
MINIS I	kerasExM	0.892	0.962	0.943	0.875	0.888	0.959	0.975					
CIFAR10	kerasExC	0.737	0.626	0.728	0.727	0.728	0.729	0.763					
	ResNet	0.733	0.723	0.747	0.735	0.737	0.584	0.791					

the IMDB dataset we use Algorithm 2 found in Appendix D to generate misclassified movie reviews. Instead of adding or deleting words, we chose to replace words in the individual instances. With this approach, we preserve the lengths of the classified sentences and reduce the distance between benign and adversarial examples. We are able to report a detection f1-score of 0.960 for this dataset.

For the audio dataset, we used Carlini and Wagner's [12] approach to create adversarial examples. Extracting the features of the audio files leads to neuron activation sequences of different lengths. This is the result of the various sampling rates during the recording of the original samples in the dataset. To be able to perform binary classifications using all feature instances, we used a different alarm model architecture here. The alarm model contains one LSTM layer followed by one output layer with two neurons to enable the detection of attacks. For this dataset we are able to detect adversarial examples with an f1-score of 0.820.

We clearly see successful detection of the majority of adversarial examples for the here used NLP and audio processing target models.

#### 5.5 Evaluation of Adaptive Attacks

To evaluate the robustness of our method against adaptive attacks, we consider the mean  $L_2$ -distance between adversarial and benign images to either fool  $N_{target}$  or  $N_{secured}$ , respectively. Hence, we generate adversarial images for  $N_{target}$  and  $N_{secured}$  using the same attack parameters to preserve comparability. We list the chosen parameters in Appendix E. For MNIST and the LeNet target model, the mean  $L_2$ -distance is 61.82% higher when attacking  $N_{secured}$ , compared to attacking the unsecured target model. Regarding related work, this poses a significant improvement. Carlini and Wagner [14] show that related defense techniques only reach a 10% higher mean  $L_2$ -distance. The authors also show that related defense techniques can be bypassed with a success rate of 100%. During our evaluation we reduce this rate to 99%. For CIFAR10 and the kerasExC target model we achieve even better results. When attacking  $N_{target}$  the mean  $L_2$ -distance is 0.14. In contrast to that, when attacking  $N_{secured}$ , the adversarial images show a mean  $L_2$ -distance of 5.06 with respect to their benign counterparts. Moreover, only 84.1% of adversarial examples of the adaptive attack are successful. Hence, we can report a significant security improvement of our target DNNs, even during white-box adaptive attacks.

#### 6 Discussion

With the in-depth experiments in this paper we show the importance of including an analysis of the dense layer neuron coverage in future defense strategies to increase the trustworthiness of neural networks. In Section 5, we sum up the most important result values to demonstrate this fact. Nonetheless, some aspects regarding detection performance, transferability, and real-world applications require further discussion.

We have shown the mean *false positive* and *false negative* rates in 5.2 and added the individual values to Appendix C. With the given values we can report a lower *false negative* error throughout our proof-of-concept. This corresponds to rather classifying benign images as adversarial than missing an attack on the target model. Hence, we can recommend using our approach in security-sensitive setups.

Another aspect of our research worth supplemental assessment are the results during the cross-testing of various alarm model versions. We were able to detect adversarial examples with alarm models trained using other attack methods. With this, we give evidence for the special behaviour of neural net-

Table 5: Mean detectability of the individual attack methods, when tested against all alarm models expressed with the accroding f1-scores.

Datasat	Target Model	Mean f1-S	Mean f1-Score of the individual attacks when tested with all alarm models										
Dataset	Target Wiouer	FGSM	C&W	DeepFool	PGD	BIM	Transfer Attack						
MNIST	LeNet	0.978	0.770	0.940	0.982	0.982	0.827						
MINIS I	kerasExM	0.987	0.768	0.917	0.987	0.980	0.832						
CIEAD10	kerasExC	0.813	0.500	0.803	0.807	0.805	0.528						
CIFARIO	ResNet	0.802	0.504	0.806	0.807	0.806	0.685						

Table 6: Performance of our method when detecting adversarial examples among clean and noisy benign images. Our capability of detecting attacks is not reduced significantly.

Dataset	Target Model	Attack	f1-score
MNIST	LaNat	FGSM	0.896
IVIINIS I	Lenet	C&W	0.913
CIFAR10	ResNet	FGSM	0.791

works when confronted with adversarial examples. Hence, we can report two additional contributions and application scenarios. First, future, yet unknown attacks seem detectable using our concept. We will investigate this in future work including potentially new attacks and those which are already published. Second, with the results gained during our cross-testing we are able to provide a ranking of the attack methods. This ranking is based on two findings. First, the difficulty in detecting the attack, expressed by the f1-score of the respective alarm model, provides a score for rating the attack. A second attack ranking is expressed by the performance of the alarm model created for the attack itself when detecting examples crafted with other attack methods. As an example, let's focus on the C&W attack on the ResNet target model which classifies CI-FAR10 images. We can clearly see that this attack method can only be detected by the alarm model especially created for this purpose. Hence, we deduce the C&W attack being the most powerful method. This interpretation of our results correlates with current findings in the field of adversarial attacks and defense strategies: as discussed in Section 2, the C&W attack is currently considered to be one of the most powerful attacks.

Regarding real-world application of our concept and the transferability of the approach, some aspects are worth mentioning. In a real-world scenario DL models are often confronted with noisy inputs. We have shown that such perturbations do not dramatically decrease the performance of the individual alarm models. Related to this is the question about the transferability. To give first evidence on this assumption, we tested detectability of adversarial examples in the NLP and audio processing environment with positive results. In future work we will provide a more analysis of the performed tests in these application domains.

The second possible restriction, that our approach may suf-

fer from, is the architecture of the model to protect. One could argue that we are not able to detect attacks if the target model does not incorporate dense layers. This can be ruled out considering the following. The maintainer of the target model creates a so-called *substitution model* which performs the same task as the target model itself and achieves a similar accuracy. If this *substitution model* uses dense layers, we are again able to apply our concept. The positive results during our experiments regarding transfer attacks indicate the practicality of the *substitution model*.

Finally, we want to emphasize the simplicity of our approach. During our research on related work, we noticed some defense strategies to be rather unintuitive including several sources of errors when not applied correctly. Our method, in contrast, is easy to use and seems intuitively reasonable. In addition, our method does not decrease the accuracy of the model to protect when tested against benign images, which is the case for some state-of-the-art defense strategies.

## 7 Conclusion

In this paper, we introduce a general end-to-end framework to detect adversarial examples during classification time. Our approach consists of two main parts.

In the first phase we initialize our alarm model to detect benign and adversarial inputs during unsecured target model runtime. In this phase, we extract neuron coverage of the target model in order to train a secondary alarm neural network. This approach is motivated by the initial hypothesis that the dense layers of the target model carry security sensitive information. Thus, the alarm network is trained to detect malicious activity patterns in the neurons of the target network during classification tasks.

In the second phase, the target model runs in secure operation mode, which is enabled by enhancing it with our trained secondary alarm network. When the target model classifies new, unseen inputs, the alarm network runs in parallel and throws an alarm if an adversarial example is being processed by the target. This approach leaves all parameters - especially the accuracy - of the target model untouched, yet improving overall application robustness significantly. In our proof-ofconcept implementation, we show the extensive capability of our approach to detect adversarial examples in image, NLP, and audio datasets. The evaluation results strongly indicate that we can not only defend with high accuracy against stateof-the-art adversarial examples, but most likely also against future, yet unknown attacks. Finally, with adaptive attacks we show that an attacker needs to perform significantly more adversarial perturbations to successfully attack our detectionenhanced target model, compared to attacking the unsecured target model.

## References

- Keras Convolutional Neural Network for CIFAR10. https://keras.io/examples/cifar10\_cnn/. Accessed: 2019-05-20.
- [2] Keras Convolutional Neural Network for MNIST. https://keras.io/examples/mnist\_cnn/. Accessed: 2019-05-20.
- [3] Mozilla Common Voice dataset. https://voice. mozilla.org/datasets. Accessed: 2019-05-20.
- [4] Mozilla Common Voice dataset. https://github. com/mozilla/DeepSpeech. Accessed: 2019-05-20.
- [5] Nicholas Carlini: Breaking Neural Network Detection Schemes. https://nicholas.carlini.com/code/ nn\_breaking\_detection. Accessed: 2019-08-21.
- [6] ResNet for CIFAR10. https://keras.io/examples/ cifar10\_resnet/. Accessed: 2019-05-20.
- [7] N. Akhtar, J. Liu, and A. Mian. Defense against universal adversarial perturbations. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3389–3398, June 2018.
- [8] N. Akhtar and A. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.
- [9] Shumeet Baluja and Ian Fischer. Adversarial transformation networks: Learning to generate adversarial examples. arXiv preprint arXiv:1703.09387, 2017.
- [10] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316, 2016.
- [11] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In 2017 IEEE Symposium on Security and Privacy (SP), pages 39–57, May 2017.

- [12] N. Carlini and D. Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In 2018 IEEE Security and Privacy Workshops (SPW), pages 1–7, May 2018.
- [13] Nicholas Carlini, Guy Katz, Clark Barrett, and David L. Dill. Ground-truth adversarial examples, 2018.
- [14] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, pages 3–14, New York, NY, USA, 2017. ACM.
- [15] Nicholas Carlini and David Wagner. Magnet and" efficient defenses against adversarial attacks" are not robust to adversarial examples. arXiv preprint arXiv:1711.08478, 2017.
- [16] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.
- [17] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Li Chen, Michael E Kounavis, and Duen Horng Chau. Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression. arXiv preprint arXiv:1705.02900, 2017.
- [18] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE Conference on Computer Vision and Pattern Recognition, pages 248–255, June 2009.
- [19] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. A study of the effect of jpg compression on adversarial images. *arXiv preprint arXiv:1608.00853*, 2016.
- [20] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. arXiv preprint arXiv:1703.00410, 2017.
- [21] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.
- [22] Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960*, 2017.
- [23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672– 2680. Curran Associates, Inc., 2014.

- [24] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.
- [25] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.
- [26] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.
- [27] Dan Hendrycks and Kevin Gimpel. Visible progress on adversarial images and a new saliency map. *arXiv preprint arXiv:1608.00530*, 2016.
- [28] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.
- [29] Sepp Hochreiter and Jürgen Schmidhuber. Long shortterm memory. *Neural computation*, 9(8):1735–1780, 1997.
- [30] Hossein Hosseini, Yize Chen, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. Blocking transferability of adversarial examples in black-box learning systems. *arXiv preprint arXiv:1703.04318*, 2017.
- [31] G. Jin, S. Shen, D. Zhang, F. Dai, and Y. Zhang. Apegan: Adversarial perturbation elimination with gan. In ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3842–3846, May 2019.
- [32] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [33] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.
- [34] Alex Kurakin, Dan Boneh, Florian Tramèr, Ian Goodfellow, Nicolas Papernot, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. 2018.
- [35] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.
- [36] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

- [37] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551, December 1989.
- [38] Hyeungill Lee, Sungyeob Han, and Jungwoo Lee. Generative adversarial trainer: Defense to adversarial perturbations with gan. *arXiv preprint arXiv:1705.03387*, 2017.
- [39] X. Li and F. Li. Adversarial examples detection in deep networks with convolutional filter statistics. In 2017 IEEE International Conference on Computer Vision (ICCV), pages 5775–5783, Oct 2017.
- [40] B. Liang, H. Li, M. Su, X. Li, W. Shi, and X. Wang. Detecting adversarial image examples in deep neural networks with adaptive noise reduction. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2018.
- [41] J. Lu, T. Issaranon, and D. Forsyth. Safetynet: Detecting and rejecting adversarial examples robustly. In 2017 IEEE International Conference on Computer Vision (ICCV), pages 446–454, Oct 2017.
- [42] Yan Luo, Xavier Boix, Gemma Roig, Tomaso Poggio, and Qi Zhao. Foveation-based mechanisms alleviate adversarial examples. *arXiv preprint arXiv:1511.06292*, 2015.
- [43] C. Lyu, K. Huang, and H. Liang. A unified gradient regularization family for adversarial examples. In 2015 *IEEE International Conference on Data Mining*, pages 301–309, Nov 2015.
- [44] Lei Ma, Felix Juefei-Xu, Jiyuan Sun, Chunyang Chen, Ting Su, Fuyuan Zhang, Minhui Xue, Bo Li, Li Li, Yang Liu, et al. Deepgauge: Comprehensive and multi-granularity testing criteria for gauging the robustness of deep learning systems. arXiv preprint arXiv:1803.07519, 7, 2018.
- [45] Shiqing Ma, Yingqi Liu, Guanhong Tao, Wen-Chuan Lee, and Xiangyu Zhang. 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.
- [46] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of* the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

- [47] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.
- [48] 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, pages 135–147. ACM, 2017.
- [49] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. In *Proceedings of 5th International Conference* on Learning Representations (ICLR), 2017.
- [50] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 86–94, July 2017.
- [51] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2574–2582, June 2016.
- [52] M. Mozaffari-Kermani, S. Sur-Kolay, A. Raghunathan, and N. K. Jha. Systematic poisoning attacks on and defenses for machine learning in healthcare. *IEEE Journal* of Biomedical and Health Informatics, 19(6):1893–1905, Nov 2015.
- [53] Nina Narodytska and Shiva Prasad Kasiviswanathan. Simple black-box adversarial perturbations for deep networks. arXiv preprint arXiv:1612.06299, 2016.
- [54] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D. Joseph, Benjamin I. P. Rubinstein, Udam Saini, Charles Sutton, J. D. Tygar, and Kai Xia. Exploiting machine learning to subvert your spam filter. In Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats, LEET'08, pages 7:1– 7:9, Berkeley, CA, USA, 2008. USENIX Association.
- [55] Augustus Odena and Ian Goodfellow. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. *arXiv preprint arXiv:1807.10875*, 2018.
- [56] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In 2016 IEEE European Symposium on Security and Privacy (EuroS P), pages 372–387, March 2016.
- [57] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In 2016 IEEE Symposium on Security and Privacy (SP), pages 582–597, May 2016.

- [58] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, ASIA CCS '17, pages 506–519, New York, NY, USA, 2017. ACM.
- [59] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In proceedings of the 26th Symposium on Operating Systems Principles, pages 1–18. ACM, 2017.
- [60] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017.
- [61] Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018.
- [62] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3856–3866. Curran Associates, Inc., 2017.
- [63] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018.
- [64] Swami Sankaranarayanan, Arpit Jain, Rama Chellappa, and Ser Nam Lim. Regularizing deep networks using efficient layerwise adversarial training. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [65] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing*, 307:195 – 204, 2018.
- [66] Richard Shin and Dawn Song. Jpeg-resistant adversarial images. In *NIPS 2017 Workshop on Machine Learning and Computer Security*, 2017.
- [67] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.

- [68] J. Su, D. V. Vargas, and K. Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions* on Evolutionary Computation, pages 1–1, 2019.
- [69] Youcheng Sun, Xiaowei Huang, and Daniel Kroening. Testing deep neural networks. *arXiv preprint arXiv:1803.04792*, 2018.
- [70] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2818–2826, June 2016.
- [71] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.
- [72] Qinglong Wang, Wenbo Guo, Kaixuan Zhang, II Ororbia, G Alexander, Xinyu Xing, Xue Liu, and C Lee Giles. Learning adversary-resistant deep neural networks. arXiv preprint arXiv:1612.01401, 2016.
- [73] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille. Adversarial examples for semantic segmentation and object detection. In 2017 IEEE International Conference on Computer Vision (ICCV), pages 1378–1387, Oct 2017.
- [74] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing mitigates and detects carlini/wagner adversarial examples. *arXiv preprint arXiv:1705.10686*, 2017.

## Appendix

## A Alarm Model Architecture

In the following we provide more information on the architecture of the target model we used throughout this paper. As we said before, the alarm model is a seven-layer neural network. The input flatten-layer accepts the concatenated extracted features, while the output layer contains two softmax-neurons in order to perform a binary classification. As hidden layers we exclusively chose dense layers with the following amount of Relu-neurons for each layer: 112, 100, 300, 200, 77. We trained this model for ten epochs and a batch size of 100.

## **B** All Result Values

In the following four Tables 7, 8, 9, and 10 we present all result values gained during the proof-of-concept. The gray cells in each table show the accuracy and f1-score of one specific alarm model when tested against its dedicated attack method. We emphasized the best test result in each table.

## C Confusion Matrix Values

In Table 11 we show the confusion matrix values of each performed test.

## D Adversarial Example Generation for the NLP scenario

With Algorithm 2 we generated adversarial examples for our target model classifying IMBD reviews. The target model performs a binary classification and tries to distinguish between positive and negative reviews respectively.

**Data:** IMDB reviews **Result:** adversarial IMDB reviews train a Word2Vec Model with all reviews: randomly pick one word to start; while not at the end of this document do find N most similar words of current word with Word2Vec; **for** *substitute*  $\leftarrow$  *next most similar word* **do** replace the current word with the *substitute*; predict and calculate the margin; if margin decrease then break end end if margin < MarginThreshold then break else recover the current word to the original word; move to next word; end end Algorithm 2: Generation of adversarial examples in the

**Algorithm 2:** Generation of adversarial examples in the IMDB dataset containing movie reviews.

		Accuracy and f1-Scores of the Alarm Models when tested agains												st: (acc; f1-score)				
Alarm Models trained	FGSM		C&W		Deen	Fool	PG	מי	RI	м	all att	tacks	transf	erred				
with:	I USIM				Deeproot		IUD		DIM		combined		examples					
FGSM	0.988	0.988	0.802	0.758	0.953	0.945	0.987	0.987	0.987	0.987	0.917	0.947	0.924	0.839				
C&W	0.943	0.942	0.977	0.977	0.943	0.933	0.957	0.956	0.958	0.957	0.944	0.965	0.951	0.905				
DeepFool	0.987	0.987	0.862	0.843	0.983	0.981	0.987	0.987	0.987	0.987	0.949	0.968	0.930	0.857				
PGD	0.989	0.989	0.719	0.615	0.935	0.921	0.991	0.991	0.991	0.991	0.881	0.923	0.895	0.760				
BIM	0.986	0.986	0.740	0.655	0.933	0.920	0.990	0.990	0.990	0.990	0.887	0.927	0.899	0.772				
all attacks combined	0.981	0.981	0.973	0.973	0.977	0.974	0.981	0.981	0.981	0.981	0.989	0.993	0.963	0.931				
transferred examples	0.977	0.977	0.925	0.921	0.949	0.941	0.968	0.968	0.966	0.966	0.951	0.970	0.961	0.926				

Table 7: All result values for the MNIST dataset and target model LeNet.

Table 8: All result values for the MNIST dataset and target model *kerasExM*.

		Accuracy and f1-Scores of the Alarm Models when tested against: (acc; f1-score												
Alarm Models trained	FGSM		C&	W	Deen	Fool	PG	מי	RI	м	all at	tacks	transferred	
with:	10	5111	C.a	. •••	Deep	1001	10	υ	DIM		comb	combined		ples
FGSM	0.992	0.992	0.767	0.700	0.917	0.892	0.992	0.992	0.985	0.985	0.890	0.929	0.893	0.792
C&W	0.973	0.973	0.975	0.975	0.968	0.962	0.968	0.968	0.961	0.960	0.969	0.981	0.959	0.933
DeepFool	0.986	0.986	0.868	0.850	0.985	0.982	0.988	0.988	0.980	0.980	0.946	0.966	0.928	0.871
PGD	0.992	0.991	0.734	0.641	0.899	0.865	0.992	0.993	0.986	0.986	0.873	0.917	0.885	0.773
BIM	0.992	0.992	0.752	0.674	0.912	0.886	0.992	0.993	0.991	0.991	0.886	0.926	0.893	0.792
all attacks combined	0.984	0.984	0.972	0.972	0.980	0.977	0.984	0.984	0.984	0.984	0.988	0.993	0.967	0.947
transferred examples	0.983	0.983	0.916	0.910	0.967	0.960	0.982	0.982	0.975	0.975	0.955	0.972	0.966	0.943

Table 9: All r	esult values for	r the CIFAR1(	) dataset and	target model	kerasExC.
14010 7.74111	count values for		J unuser and	target model	Refuserc.

		Accuracy and f1-Scores of the Alarm Models when tested agains												nst: (acc; f1-score)					
Alarm Models trained	FGSM		C&W		Daan	Fool	PG	מי	RI	м	all att	acks	transf	erred					
with:	I USINI				Deeprooi		IUD				combined		examples						
FGSM	0.843	0.847	0.589	0.470	0.839	0.842	0.840	0.844	0.839	0.843	0.774	0.849	0.639	0.578					
C&W	0.699	0.679	0.739	0.733	0.662	0.625	0.673	0.642	0.665	0.631	0.640	0.740	0.549	0.449					
DeepFool	0.851	0.852	0.571	0.414	0.853	0.855	0.853	0.855	0.853	0.855	0.767	0.843	0.624	0.540					
PGD	0.839	0.840	0.584	0.449	0.841	0.842	0.841	0.843	0.841	0.843	0.762	0.840	0.623	0.543					
BIM	0.848	0.850	0.580	0.434	0.849	0.850	0.849	0.851	0.850	0.852	0.767	0.844	0.618	0.530					
all attacks combined	0.708	0.771	0.678	0.741	0.709	0.771	0.709	0.772	0.709	0.772	0.882	0.932	0.688	0.754					
transferred examples	0.749	0.777	0.577	0.557	0.738	0.766	0.743	0.771	0.743	0.770	0.769	0.852	0.704	0.732					

Table 10: All result values for the CIFAR10 dataset and target model ResNet.

		Accu	iracy ar	nd f1-S	cores of	the Ala	arm Mo	dels wl	hen test	ed agai	nst: (ac	c; f1-sc	ore)	
Alarm Models trained	FGSM		C&	W	DeenFool		PG	מ	RI	м	all att	tacks	transferred	
with:					Deepi oor		1.50		DIM		combined		examples	
FGSM	0.810	0.815	0.559	0.446	0.812	0.820	0.812	0.821	0.812	0.821	0.761	0.809	0.658	0.674
C&W	0.701	0.715	0.707	0.727	0.698	0.716	0.697	0.716	0.695	0.713	0.725	0.789	0.703	0.753
DeepFool	0.819	0.827	0.568	0.469	0.822	0.833	0.823	0.835	0.823	0.834	0.778	0.826	0.666	0.686
PGD	0.822	0.826	0.560	0.434	0.824	0.831	0.826	0.833	0.825	0.832	0.767	0.813	0.644	0.651
BIM	0.819	0.825	0.559	0.445	0.821	0.830	0.822	0.832	0.822	0.832	0.771	0.818	0.646	0.658
all attacks combined	0.788	0.812	0.677	0.687	0.792	0.818	0.793	0.820	0.793	0.819	0.812	0.864	0.744	0.791
transferred examples	0.672	0.614	0.570	0.440	0.655	0.593	0.655	0.595	0.653	0.592	0.583	0.602	0.662	0.670

Table 11: Confusion Matrix values for all datasets, target models, and attack methods. Each result belongs to the detection of adversarial attack methods with the according target model.

Detect	Target Medel	Attack	Performance	Performance of the Alarm Models when tested									
Dataset	Target Model	Attack	against the ac	cording attack m	ethod								
			True Positive	True Negative	False Positive	False Negative							
MNIST	LeNet	FGSM	1.00	0.98	0.02	0.00							
		C&W	0.98	0.97	0.03	0.02							
		DeepFool	0.99	0.98	0.02	0.01							
		PGD	0.99	0.99	0.01	0.01							
		BIM	0.99	0.99	0.01	0.01							
	kerasExM	FGSM	1.00	0.99	0.01	0.00							
		C&W	0.98	0.97	0.03	0.02							
		DeepFool	0.99	0.98	0.02	0.01							
		PGD	1.00	0.99	0.01	0.00							
		BIM	0.99	0.99	0.01	0.01							
	LSTM	Transfer	0.97	0.88	0.12	0.03							
	CapsuleNN	Transfer	1.00	1.00	0.00	0.00							
CIFAR10	kerasExC	FGSM	0.87	0.81	0.19	0.13							
		C&W	0.72	0.76	0.24	0.28							
		DeepFool	0.87	0.84	0.16	0.13							
		PGD	0.85	0.83	0.17	0.15							
		BIM	0.86	0.84	0.16	0.14							
	ResNet	FGSM	0.86	0.76	0.24	0.14							
		C&W	0.78	0.63	0.37	0.22							
		DeepFool	0.89	0.75	0.25	0.11							
		PGD	0.87	0.78	0.22	0.13							
		BIM	0.88	0.76	0.24	0.12							

# E C&W Attack Parameters for the Adaptive Attack

Table 12 shows the attack parameters of the C&W attack during the adaptive white-box attacks for the MNIST dataset.

Parameter Name	Parameter Value
max-iteration	3000
batch-size	100
learning-rate	0.005
binary-search-steps	20

Table 12: C&W attack parameters during the adaptive attack for MNIST.

Table 13 shows the attack parameters of the C&W attack during the adaptive white-box attacks for the CIFAR10 dataset.

Parameter Name	Parameter Value
max-iteration	100
batch-size	100
learning-rate	0.01
binary-search-steps	5

Table 13: C&W attack parameters during the adaptive attack for CIFAR10.