DLA: Dense-Layer-Analysis for Adversarial Example Detection

Philip Sperl, Ching-Yu Kao, Peng Chen, Xiao Lei, Konstantin Böttinger
Fraunhofer AISEC, Germany
{*philip.sperl, ching-yu.kao, peng.chen, xiao.lei, 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 DNN classifications 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 without influencing the target model's performance. Inspired by 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 run-time, 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 DNN architectures. In addition to effectively defending against state-of-the-art attacks, our approach generalizes between different sets of adversarial examples. Our experiments indicate that we are able to detect future, yet unknown, attacks. Finally, during white-box adaptive attacks, we show our method cannot be easily bypassed.**

***Index Terms*—Deep Learning, Adversarial Machine Learning, Neural Network Security**

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 [1], to outplaying humans in Go [2], or even autonomously driving cars [3].

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. 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 flavors 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 [4], [5]. As the training dataset is typically not available to 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. [6] first demonstrated that small perturbations on images fed to a deep neural network (DNN) can provoke such misclassifications. Since then, new attacks and countermeasures against so-called *adversarial examples* have been introduced at a fast pace without the discovery of a fundamental and general defense strategy, yet. 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, based on our experiments, also defend against future attacks. Further, our method defends against attacks in image classification, natural language, and audio processing scenarios.

Currently, adversarial attacks seem to subdue corresponding defense 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 run-time. We were inspired by recent findings in the field of neural network (NN) testing and its interesting prospects. Pei et al. [7] introduced the idea of neuron coverage, which serves as a metric to guide testing of NNs. Since then, further coverage metrics have been proposed and various testing techniques have made use of them [8], [9]. Odena and Goodfellow [10] reported promising results when applying concepts of coverage-guided fuzzing to NN 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 yet unknown attacks. For future work on this topic, we plan to publish our code, models, and used datasets.

In summary we make the following contributions:

- We propose a general end-to-end method to de-

A preprint of this paper has been deposited on ArXiv.

tect 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.
- We evaluate our method during adaptive attacks.

The rest of this paper is structured as follows. In Section 2, we review related work and summarize the latest findings on defense strategies against adversarial attacks. In order to fully describe the environment in which we can successfully detect adversarial examples we introduce the threat models we consider throughout this paper in Section 3. We present our main contribution, a novel concept of detecting evasion attacks on NNs, in Section 4. Sections 5 and 6 present a thorough proof-of-concept including experiments and evaluation of the results. To further gain trust in our concept, we perform and evaluate adaptive white-box attacks in Section 7. In Section 8, we discuss the restrictions of our method as well as the real-world applicability, transferability, and generalization to future attacks. Finally, we conclude the paper with Section 9.

2. Related Work and Background

2.1. Adversarial Attack Methods

In this paper, we introduce a framework to detect test-time evasion attacks. 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., $\|x' - x\| \leqslant \epsilon$ for some small $\epsilon \in \mathbb{R}^+$. Then x' is an adversarial example, if the following holds:

$$H(x) = H(x') \quad \wedge \quad f(x') \neq H(x').$$

Informally, adversarial examples are slightly mutated versions of their original counterparts that lead the targeted network to misclassification. We categorize test-time evasion attacks as introduced in [11]. The different attack types differ in the amount and nature of information available to 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 neither has knowledge of the target model architecture nor access to the parameters after training. In the following, we introduce state-of-the-art *white-box* attack methods. Throughout our experiments in Section 5 we will revisit the here introduced attack methods and motivate our choice of considered attacks in Section 5.3.

Szegedy et al. [6] first demonstrated the vulnerability of NNs to slightly mutated inputs. The authors formulated the problem of finding such mutations with a minimization problem. To solve this problem the authors used a box-constrained *L-BFGS* [12].

In 2014, Goodfellow et al. [13] refined the previous findings and proposed their resulting "Fast Gradient Sign Method" (*FGSM*).

Kurakin et al. [14] 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. [15] further refined the approach. The authors showed the BIM attack being equivalent to "Projected Gradient Descent" (*PGD*). By making use of the l_∞ version of this standard convex optimization method the authors further improved the previously shown BIM.

Moosavi-Dezfooli et al. [16] 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 by a polyhydron.

The majority of current attacks are restricted by the l_∞ or l_2 norm between benign and adversarial examples. In contrast to that, Papernot et al. [17] 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 refined by Su et al. [18]. In this publication the authors successfully fooled DNNs using their *One Pixel Attack*.

Currently the most powerful white-box attack was proposed by Carlini and Wagner (*C&W*) in [19]. 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 rather introduce a cost function f_y as substitute.

Moosavi-Dezfooli et al. [20] 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 black-box situation. Here, the attacker uses a NN 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. This property of adversarial examples has been shown and analyzed by numerous publications, e.g., [6], [13], [21].

2.2. Defenses against Adversarial Examples

Akhtar and Mian [22] categorize adversarial defenses using three classes. The first class introduces a modified training procedure or various prepossessing methods to the input data. In the second, modifications to the targeted model itself are summarized, while the third class contains concepts using an additional model to increase overall robustness. For the latter two 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 state-of-the-art detection methods and refer to the survey by Carlini and Wagner [23] for further information.

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 [24]. Adversarial training is often introduced by authors of attacks as the first strategy to prevent a successful attack [6], [13], [16].

The authors of [13] 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 [13], [25].

In 2017 Madry et al. [15] interpreted adversarial training 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 authors then show the resulting networks to be robust against a wide range of adversaries.

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. [20] showed that adversarially trained models are still vulnerable using other known attack methods. Moreover, Tramèr et al. [26] presented a two-step 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 [27], [28].

Data Compression and Feature Squeezing: Dziugaite et al. [29] 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 compression resulted in successful defense methods [30], [31]. However, Shin and Sing [32] showed that a considerable amount of adversarial images are not affected by to a JPG compression, especially when crafted with the C&W method.

Similar strategies have been proposed in [33] and [34]. Here "Feature Squeezing" is used to reduce the complexity of the inputs, by reducing the color depth or applying smoothing filters.

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

Data Randomization Preprocessing: Luo et al. [35] proposed to apply the targeted neural network only to a certain regions of the classified images. This technique is shown to be a valuable countermeasure against adversarial images created by L-BFGS and FGSM. Xie et al. [36] analyzed the effects of random resizing and padding. Similarly, Wang et al. [37] made use of a separately executed data-transformation module, which partially removes adversarial perturbations.

2.2.2. Modifying the Network. 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 [28].

Related to Gradient Hiding, Ross and Doshi-Velez [38] 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 [39] and [40].

In 2015, Papernot et al. [41] presented Defensive Distillation. The originally introduced distillation technique shown by Hinton et al. [42] 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 McDaniel further improved the concepts conveyed in the initial publication. Nonetheless, Carlini and Wagner [19] claim their C&W attack to be successful against Defensive Distillation.

2.2.3. External Network Add-Ons. Akhtar et al. [43] 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. [44], Generative Adversarial Networks (GANs) are widely used and referred to in numerous publications. Some promising publications using GANs to protect DNNs against attacks are [45]–[47].

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 behavior 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. These

methods can additionally be used to detect attacks. Baluja and Fischer [48] 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. [33] used this technique to protect networks against the C&W attack.

Similarly, Hendrycks and Gimpel [49] 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. [34] interpreted adversarial perturbations as noise and detected them by using scalar quantization and smoothing filters.

A more straightforward approach was evaluated by Gong et al. [50]. 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 [23] the approach failed to reach similar detection rates for different datasets.

Meng et al. [51] proposed their framework MagNet, which evaluates the original dataset and analyzes the manifold of the benign examples. If a new example 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 [52].

A comparable pre-classification was introduced by Grosse et al. [53]. 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. [54] 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. [55] added a sub-network to the original neural network. This sub-network is adversarially trained and acts as a binary classifier during the classification of the inputs. In [56], the authors showed that this method can again be bypassed by an attack.

Lu et al. [56] 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. [57] 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. [58] tried to detect adversarial examples 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 sub-manifold 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. [59] 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.

As our concept is closely related to [58] and [59] we briefly stress the main differences and potential advantages provided by our approach. Both stated frameworks detect adversarial examples by examining the inner processing of the protected NN. In contrast to our approach, the authors further process the extracted information to craft features enabling a detection. Instead, we propose a method which directly works on the hidden activation values of the dense layers. Opposed to [58] and [59], we use a secondary NN for the final detection. This poses a more intuitive and easy to implement solution. Our detection scheme works out-of-the-box without additional preprocessing or optimization steps. Furthermore, as our system solely comprises NNs, the detection scheme can be easily integrated in existing DL pipelines. An advantage of the concept by Ma et al. [59] is that it does not require adversarial examples during training.

3. Considered Threat Models

When presenting a new defense method the definition of the considered threat model builds an essential but often neglected part. The threat model describes the conditions under which the defense method is designed and tested. Hence, by defining the threat model, we convey our assumptions under which our method is capable of guaranteeing a certain level of security. In the following we present the two threat models we considered throughout this paper. For this purpose we followed the guideline on the evaluation of adversarial defense methods by Carlini et al. [60].

A threat describes the following three attacker characteristics:

- Goals of the adversary
- Capabilities of the adversary
- Knowledge of the adversary

The goals of the adversary describe whether an attacker performs nontargeted or targeted attacks. In nontargeted attacks the adversary simply tries to alter the classification output to any class except the true class. Contrary, during targeted attacks the goal is to alter the classification output of the attacked NN to a specifically chosen class.

The capabilities of an adversary can range from making changes to the input to altering the trained network. In evasion attacks, solely changes to the input are assumed. Such changes are typically constrained by an l_p-norm describing the distance between the original inputs and the corresponding adversarial examples. The norm can be chosen fitting to the environment in which the attacked

network is being used. In image classification tasks the l_0, l_2, or l_∞ distance are usually considered

For the evaluation of new defense methods it is crucial to specifically define the adversary's knowledge. This knowledge ranges from a black-box setting in which the attacker only sees the output of the NN, to a white-box setting. For white-box settings, two distinct scenarios are possible. In the first scenario, the attacker is not aware of the applied defense method. On the other hand, in the second white-box scenario, the attacker knows the applied defense method. Hence, the attacker is capable of directly attacking the combined system consisting of defense measure and NN in an adaptive manner. This scenario allows an evaluation of the resulting security level of the protected network without relying upon the defense method being secret.

Main Threat Model. In our main proof-of-concept experiments we consider the following threat model: The attacker performs evasion attacks and tries to alter the classification output of our NN in a targeted manner. For this purpose, the attacker uses various state-of-the-art attack algorithms. The added adversarial perturbations are desired to be small enough, so they are imperceptible for a human expert, which is consistent with the common definition of adversarial examples. Finally, we consider a white-box scenario in which the attacker performs simple attacks to the NN only. Hence, the attacker is not aware of our proposed defense strategy. With this measure, we provide an unaltered evaluation of the performance of our detection scheme allowing comparison to related methods.

Threat Model for Adaptive Attacks. For the adaptive attacks which we describe in more detail in Section 7, we need to adjust the attacker's knowledge. Here, the attacker is aware of our proposed defense method and mounts and adaptive attack leveraging this knowledge.

4. Methodology

In this section we introduce our main concept to detect adversarial examples during classification time. The core idea originates from our hypothesis as initially shown in Section 1: Adversarial examples provoke a distinctive behavior of dense layer neuron activations such that attacks become detectable. We provide a detailed description on how to expand and build upon this idea in the following.

Fig. 1 shows an overview of our concept and underlying data flow. Joining the individual steps provides an end-to-end pipeline for fully automated adversarial example detection.

Our method is designed to help developers and maintainers of NNs 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 we want to secure as the *target model* N_{target}. Our aim is to generate a secure model N_{target}^{secure} that throws an alarm signal whenever an adversarial example is being processed. We achieve this as follows: We generate adversarial examples and extract the dense layer neuron coverage of the target model, triggered by benign and adversarial inputs. Using the extracted coverage, we train an alarm model that enables secure operation of our target model.

4.1. 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. We summarize the produced adversarial examples in the 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-the-art techniques. As we discussed in Section 2, the attacks do not only differ in success rates but also in their detectability. By covering the currently strongest attacks we try to circumvent this issue. Moreover, to cover the case of black-box attacks, we recommend using transferred adversarial examples as well. It is important to note that only mutated examples should be considered which lead to misclassifications in N_{target}.

4.2. Extracting Dense Layer Neuron Coverage

In this step, we observe the target model's behavior while processing benign and adversarial inputs. We refer to this step as *feature extraction*. Here, the datasets D_{benign} and D_{adv} are fed to the 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 omit its classification outputs. Instead, we extract the activation values of all available dense layers and concatenate them to one sequence for each input. The resulting datasets, which hold the sequences for all samples, are called I_{benign} and I_{adv}, respectively. For further usage, we adopt the labels to distinguish between adversarial and benign samples. The dataset $I_{<attackname>}$ holds the target model's activation value sequences for all benign and adversarial examples for one specific attack method. We preserve this separation of the activation value sequences, since 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.

4.3. Training an Alarm Model

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

Previous work [58], as discussed in Section 2.2.4, follows a similar idea. The authors try to extract information

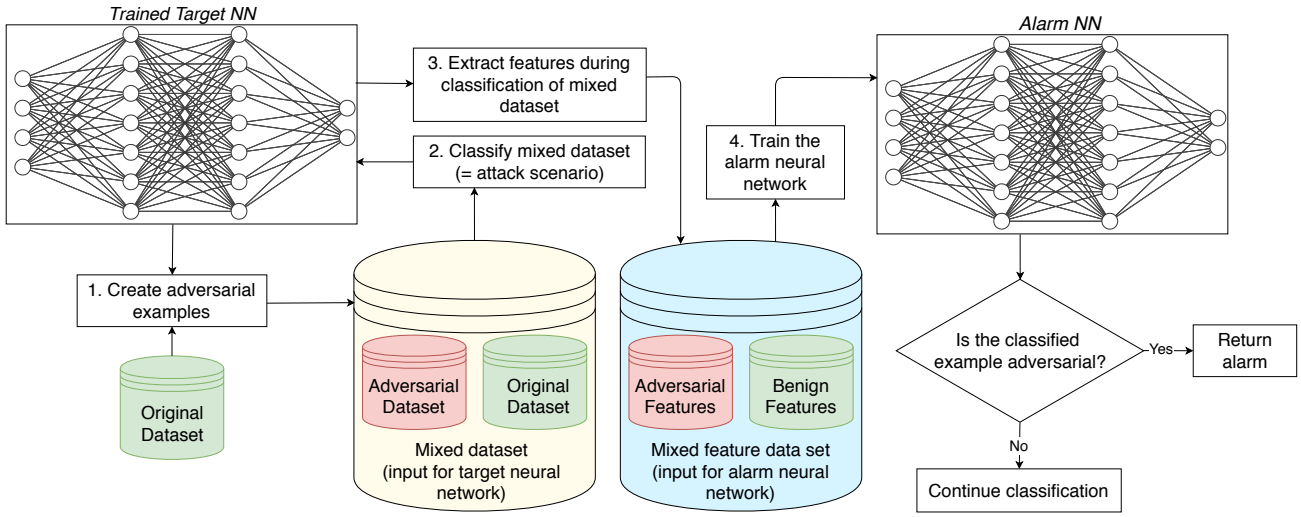

Figure 1: Overview of our concept showing the required neural networks, datasets, and calculations.

from neural layers and further deliberately process them to detect adversarial images. However, taking information directly from all dense layers of the trained model is more efficient providing an end-to-end solution without further processing steps.

Accordingly, we propose to interpret the analysis of the dense layer features as a binary classification, which generalizes well over different scenarios and model architectures: Instead of including hands-on measures and distinguishing between different scenarios, we train an additional NN to perform the required actions which we call *alarm model*, N_{alarm}.

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

The architecture of the alarm model heavily influences the success of our approach. Different architectures need to be tested against each other to provide a viable well generalizing solution. In Section 5, we recommend a specific architecture. Still, future work needs to further investigate 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.

4.4. Concept Overview

We 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: the initialization and secure operation.

In the initialization 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 a group of alarm models, each capable of detecting adversarial examples created by one specific attack method.

In the second phase, during the secure operation of the target model, we continuously extract the features during classification of new, unseen samples, D_{test}. The resulting activation sequences are fed to all available alarm models performing binary classifications. If the alarm models' outputs indicate attacks, 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. This use-case-depended choice provides different levels of security.

5. Implementation and Experimental Setup

In the following, we present details regarding our proof-of-concept implementation and our experimental setup. We evaluate the results in the next section.

5.1. Considered Datasets

For our main experiments, we considered the MNIST [61] and CIFAR10 [62] image datasets. This allows a comparison of our method to state-of-the art defense techniques. Furthermore, 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 70 000 handwritten digits ranging from 0 to 9 of which 60 000 build the training set and 10 000 the test set. Each digit is represented by 28×28 gray-scale pixels.

CIFAR10 consists of 60 000 colored images of which again 10 000 images build the test set. Each image is

Input: D_{benign}, D_{test}, N_{target}, N_{alarm}
Result: N_{target}^{secure}
for *Initialization* **do**
  $D_{adv} \leftarrow$ CreateAdvExamples(D_{benign}, N_{target});
  $I_{benign} \leftarrow$ ExtractInformation(D_{adv}, N_{target});
  $I_{adv} \leftarrow$ ExtractInformation(D_{benign}, N_{target});
  $N_{alarm} \leftarrow$ Train(N_{Alarm}, I_{benign}, I_{adv});
end
while *Secure Operation* **do**
  while *1* **do**
    $x \leftarrow$ Sample(D_{test});
    $y_{target} \leftarrow$ Classify(N_{target}, x);
    $i_{y_{target}} \leftarrow$ ExtractInformation(x, N_{target});
    $y_{alarm} \leftarrow$ 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.

stored using $32 \times 32 \times 3$ pixels, which makes this dataset more difficult to classify.

To prove detectability of adversarial examples in the natural language processing (NLP) context, we used the IMDb dataset of movie reviews [63]. Both the train and test set contain 25 000 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 [64], 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 into known classes.

5.2. Target Models

Throughout the proof of concept, we used state-of-the art target models in order to keep a close relation to real-word scenarios. In Table 1 we sum up the used models and show their training and test accuracy as well as a short description of the individual architectures. For MNIST, we chose LeNet [65] and a simple Multi-Layer-Perceptron (MLP) [66], we refer to as kerasExM. For CIFAR10 we considered ResNet [67] and a deep CNN [68], we refer to as kerasExC.

In order to evaluate if our method can generally be applied to a wide range of DL architectures, we additionally conducted experiments using the following two examples: On the one hand, we included a Long Short Term Memory (LSTM) based target model. Here we chose an architecture which achieves remarkable results on the MNIST dataset. On the other hand, we considered a capsule network (CapsuleNN). This type of NN is shown to be more robust to white-box attacks compared to conventional CNNs [69].

For our NLP-based tests, we used an LSTM target model with one embedding layer. Finally, for the audio

Figure 2: Adversarial images created with: FGSM, C&W, DeepFool, BIM, PGD. 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.

experiments we chose DeepSpeech (version 0.4.1) [70] which converts speech to text. Since it is pretrained we did not add its training and test accuracy to Table 1.

CapsuleNN and ResNet are trained using the *adam* optimizer [71] while the remaining models are trained with *stochastic gradient descent*.

5.3. Considered Attack Methods

We evaluated the detectability of the following attack methods: FGSM, C&W, DeepFool, PGD, and BIM. The motivation to choose them originates from the nature and popularity of these methods. We included diverse attacks, such that 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 be the most powerful white-box 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 methods, we additionally considered *black-box transfer* attacks. Here we created adversarial images in a *white-box* setup on model *A* and transferred the resulting examples to attack model *B*.

Since the actual crafting and implementation of the attacks is not part of our concept, we used the *foolbox* framework [74] to generate adversarial examples for MNIST and CIFAR10. To create adversarial examples based on the Common Voice dataset, we refer to [75]. Finally, for the IMDb dataset we created an algorithm to produce adversarial examples, which we briefly describe in Appendix D. In future work, we will further explore and refine this attack method.

TABLE 1: ARCHITECTURES AND PERFORMANCE OF THE TARGET MODELS USED DURING THE EXPERIMENTS

Dataset	Model Name	Model Details	Training Accuracy	Test Accuracy
MNIST	LeNet [65]	– 2 convolutional layers with filter size 5 – each convolutional layer is followed by a max-pooling layer with size 2 – 2 dense layers after each max-pooling layer	0.976	0.987
	kerasExM [66]	– one hidden layer with 512 neurons	0.972	0.985
	CapsuleNN [72]	– 10 capsules each of size 6	0.992	0.991
	LSTM [73]	– 1 LSTM layer followed by two dense layers with 64 and 32 neurons	0.975	0.978
CIFAR10	kerasExC [68]	– 4 convolutional layers with filter of size 3 – each pair of convolutional layers is followed by a max-pooling layer of size 2 – last hidden layer of dimension 512 is fully connected	0.852	0.790
	ResNet [67]	– 3 blocks followed by an average pooling size of 8	0.961	0.790
IMDb	LSTM (for NLP)	– 1 embedding layer – 1 64-neuron dense layer	0.996	0.81
Mozilla Common Voice	DeepSpeech [70]	– containing 2 parts – convolutional and recurrent network	–	–

5.4. Alarm Model Architecture and Training

During our proof of concept, we exclusively used one well generalizing alarm model architecture to detect adversarial examples for MNIST and CIFAR10. We restricted the space of tunable parameters to show the generality and simplicity of our concept. Neither the used dataset nor the applied target model, which needs to be protected, affected our alarm model architecture. For future work or in real-world applications, a deliberate choice of the alarm model will further improve the strength of our concept. In this paper, we used a DNN with six dense layers which we trained for ten epochs and a batch-size of 100 in each scenario. We used the *adam* optimizer [71] with a learning rate of 0.001. In some cases this alarm model suffered from underfitting. A more detailed description of the architecture can be found in Appendix A. As the alarm model performs a binary classification the single output logit decides about the nature of the analyzed input. Throughout this paper we did not optimize the decision threshold of this logit which influences the trade-off between precision and recall during detection.

5.5. Main Test Scenario

We assume ourselves in the position of the trained model's maintainer and try to increase its trustworthiness under the consideration of our main threat model. Each step we present was performed for all introduced attack methods. For the sake of simplicity we show each step only once.

First, we crafted adversarial images using the above stated methods. During this process, we payed attention to the way the datasets have been split beforehand. Consequently, we created two separate adversarial datasets, based on the train and test subsets. The samples in the test set simulate inputs fed to the target during an attack while being used in the field. Additionally, we can check detectability of adversarial examples which are based on unseen benign inputs to rule out a detection bias. We call the datasets D_{adv}^{train} and D_{adv}^{test}. To form D_{adv}^{train} and D_{adv}^{test}, we created (60 000, 10 000) for MNIST and (50 000, 10 000) adversarial examples for CIFAR10. We let N_{target} classify all samples in the four datasets and stored 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 preserved the division between test and training samples. This allowed a sound evaluation during the proof of concept. Note, that we neglected this split of the datasets in Algorithm 1 for the sake of simplicity.

In the second step, we used I_{adv}^{train} and I_{benign}^{train} to train the alarm model. Hence, for each target model and attack method, we created 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 performed cross-testing and evaluated the robustness of our approach towards new attack methods. Thus, we tested a trained alarm model against the features of a different attack. Put simply, we trained the alarm model with activation values triggered by one specific attack and detected adversarial examples created by another attack method. With this, we simulated the scenario in which we encounter a new and yet unknown attack.

Furthermore, we created a combined alarm model $N_{alarm}^{combined}$ which is trained using features triggered by various attacks. Here, we verified if considering diverse information, based on a wider range of attacks, improves the alarm model's performance and provides a stronger detection capability.

5.6. Additional Experiments

We divided our supplementary tests into four parts to further establish confidence in our approach.

In the first part, we used the previously created adversarial images for the MNIST dataset and conducted transfer attacks targeting an LSTM neural network and a capsule network. This experiment investigated whether our approach can be applied in the context of different DL architectures or not. Both targets contain dense layers from which we extracted the activations values in order to train our alarm model.

In the second part, we ran two experiments with the regular target models classifying MNIST and CIFAR10 images. With the first test we analyzed if our concept is robust to noisy input images. Hence, we excluded the possible effect in which our framework is solely able to distinguish between clean benign images and adversarial images containing perturbations. For this purpose, we created noisy benign images with the same amount of distortion as their adversarial counterparts. We calculated the distances between the original and adversarial images with respect to the used distance metric of the attack. The resulting datasets contain original, adversarial, and benign noisy images. To provide comparability, we preserved the distribution of benign and adversarial examples in this supplementary test set.

Subsequently, in the third part we provided evidence for our initial hypothesis of the paper presented in Section 4. We assume that adversarial examples provoke a unique activation pattern in the dense layers which can be exploited to detect attacks. For this purpose, we analyzed the dense layer activation values of the target models during misclassification of original, benign inputs. Then, we extracted the according features and trained an alarm model to detect such incorrectly classified inputs. If our main hypothesis holds true, misclassified original inputs will not be easily detectable. The behavior of the target model should be similar during correct and incorrect classifications of benign inputs. Solely adversarial examples are assumed to provide a distinct and detectable behavior.

In the fourth part, we tested our concept in the context of two additional types of datasets. We investigated if we are able to detect adversarial examples in NLP and audio datasets. This test gives first evidence on the applicability in numerous DL-based 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 6.3.

Finally we perform and evaluate adaptive attacks which we describe in more detail in Section 7.

5.7. Experiment Overview

With the following list we summarize our performed experiments and provide an orientation for our subsequent evaluation. In summary, we performed the following experiments:

- Main proof of concept:
  - Detecting one specific attack method
  - Detecting new attack methods
  - Detecting multiple attack methods
- Additional experiments:
  - LSTM and capsule targets
  - Noisy inputs
  - Misclassified inputs
  - Detecting NLP and audio attacks
- Adaptive attacks

6. Evaluation

In this section, we show our experimental results and evaluate the performance of our detection method. We split this into three parts. First, we discuss our analysis of the extracted features and show their distribution using a representative 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 of our supplementary experiments.

6.1. Feature Analysis

As the extracted features are the core of our hypothesis and concept, we illustrate the major findings during our analysis. In Fig. 3 we show neuron activation sequences for the LeNet target model and all attack methods. For better visualization 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. Gray dots represent benign instances and red crosses indicate adversarial ones. We can clearly see a difference in the dense-layer activation patterns. 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 suggests a more challenging detection of the C&W attack. Note, that since we want to provide an end-to-end framework to detect adversarial examples, we directly use the raw extracted data.

6.2. Results of the Main Proof-of-Concept

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 mean *false positive* and *false negative* rates.

Detecting One Specific Attack Method. 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,

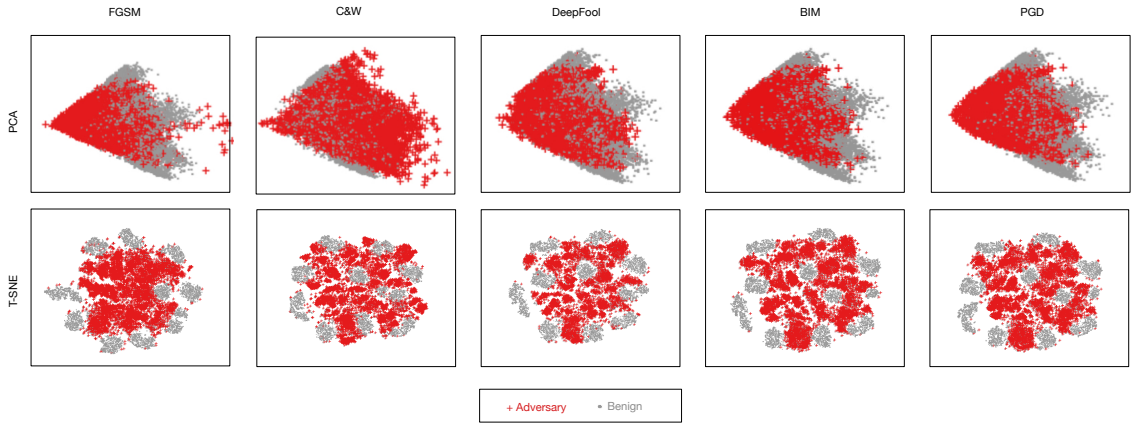

Figure 3: Visualization of the extracted features during the classification of MNIST-based adversarial and benign images for the LeNet target model. The dimensionality of the features was reduced using PCA and t-SNE. Each column shows the plots for one attack method. The gray dots and red crosses represent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

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 with the MNIST dataset. The respective f1-scores range above 0.9. For the CIFAR10 dataset, our framework detects the majority of attacks, posing as a viable solution for real-world applications. In Table 13 (Appendix B) we show the achieved precision and recall values of this experiment. With further optimizations of the alarm model's output threshold, the performance of the overall system can be adapted to the analyzed data and use-case.

Detecting New Attack Methods. With the following cross-testing experiments we analyze if our concept is capable of detecting new, unseen attacks. We train the alarm model with features provoked by one attack method and test it with features based on the other attacks. In summary, for both datasets and for each target model we tested seven alarm models against six attack methods. Five of the seven alarm models are based on the attack methods FGSM, C&W, DeepFool, PGD, and BIM. The two additional alarm models are the combined one which we will introduce below, and the alarm model trained using features extracted during transfer attacks. The result-space of the cross-tests exceeds the frame of this paper. Hence, in Table 3 we summarize the mean performance of our approach during this experiment while showing the individual result values in Appendix B. With the results we report successful detection of adversarial examples, for which the underlying method has not been known beforehand. Thus, our method likely detects future, yet unknown attacks and can therefore be employed in real-world scenarios.

Detecting Multiple Attack Methods. To evaluate if training on a combination of features based on multiple attack methods improves the performance of our method, we created a combined alarm model for each dataset. For each target model, its combined alarm model is trained with features extracted during the evaluation of all attack methods. Table 4 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 similar results with the combined models. Regarding the trade-off between robustness against multiple attacks and the resulting detection capability we can recommend using a combined model during the application in the field.

Error Rates. During detection of adversarial examples it is important to evaluate the error rates which heavily influence the real-world applicability of the system. We performed detection runs on multiple nonoverlapping batches of our test dataset and show the resulting mean rates with the according standard deviations in Appendix C. For MNIST, the mean *false positive* and *false negative* rates are 0.009 and 0.019, respectively. Similarly, for CIFAR10 we report mean error rates of 0.164 and 0.209, respectively. For both settings 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. Still further use-case and dataset-depended optimizations are required.

6.3. Results of Additional Experiments

LSTM and Capsule Targets. During our tests with an LSTM and capsule target network, we were able to detect adversarial images based on the MNIST dataset with f1-scores of 0.929 and 0.936, respectively. The positive results emphasize the applicability of our concept for a wide range of neural network architectures using dense layers. Furthermore, with this experiment we make first steps towards applying our method to dense-layer-free target models. Note, that the used LSTM target model incorporates two dense layers after the LSTM unit. Even though, the main learning power resides in the LSTM unit, our framework can leverage the information from the additional dense layers. This suggests that adding dense layers to dense-layer-free target models again enables the application of our concept.

Noisy Inputs. In Table 5, we show the results of the tests containing noisy images. The performance of the individ-

TABLE 2: F1-SCORES OF THE INDIVIDUAL ALARM MODELS WHEN TRAINED AND TESTED WITH THE FEATURES EXTRACTED USING THE CORRESPONDING ATTACK METHOD. MAIN RESULTS OF THE PROOF OF CONCEPT

Dataset	Target Model	F1-Score of the Alarm Models with the according attacks:				
		FGSM	*C&W*	*DeepFool*	*PGD*	*BIM*
MNIST	LeNet	0.988	0.977	0.981	0.991	0.990
	kerasExM	0.992	0.975	0.982	0.992	0.991
CIFAR10	kerasExC	0.847	0.733	0.855	0.843	0.852
	ResNet	0.815	0.727	0.833	0.833	0.832

TABLE 3: F1-SCORES OF THE INDIVIDUAL ALARM MODELS WHEN DETECTING ALL ATTACK METHODS. EACH ALARM MODEL IS TRAINED WITH FEATURES PROVOKED BY ONE ATTACK AND TESTED WITH A COMBINED FEATURE SET INCLUDING UNSEEN ATTACK METHODS

Dataset	Target Model	F1- Score of the individual Alarm Models with all attacks, model name:						
		FGSM	*C&W*	*DeepFool*	*PGD*	*BIM*	*Transfer*	*Combined*
MNIST	LeNet	0.943	0.956	0.961	0.920	0.924	0.957	0.980
	kerasExM	0.927	0.970	0.959	0.915	0.923	0.965	0.982
CIFAR10	kerasExC	0.783	0.663	0.786	0.780	0.785	0.735	0.767
	ResNet	0.761	0.722	0.778	0.769	0.771	0.568	0.796

TABLE 4: F1-SCORES OF THE COMBINED ALARM MODELS WHEN TESTED AGAINST EACH ATTACK SEPARATELY AND ALL ATTACKS AT ONCE

Dataset	Target Model	F1-Score of the combined Alarm Model with:						
		FGSM	*C&W*	*DeepFool*	*PGD*	*BIM*	*Transfer*	*Combined*
MNIST	LeNet	0.981	0.973	0.974	0.981	0.981	0.930	0.980
	kerasExM	0.984	0.972	0.977	0.984	0.984	0.946	0.982
CIFAR10	kerasExC	0.771	0.740	0.771	0.772	0.772	0.741	0.767
	ResNet	0.812	0.686	0.818	0.820	0.819	0.750	0.796

ual alarm models is decreased by 10% in the worst case. Even though, our method can still detect a fair amount of attacks, noise noticeably reduces the performance of the system.

TABLE 5: PERFORMANCE WHEN DETECTING ADVERSARIAL EXAMPLES AMONG CLEAN AND NOISY BENIGN IMAGES

Dataset	Target Model	Attack	F1-score
MNIST	LeNet	FGSM	0.896
		C&W	0.913
CIFAR10	ResNet	FGSM	0.791

Misclassified Inputs. In this experiment we evaluated if our approach allows detection of original but incorrectly classified samples. We argued that adversarial examples provoke a distinct pattern in the activation values enabling detection. Hence, training an alarm model which allows detection of misclassified inputs would contradict our main hypothesis. During our experiments with the MNIST and CIFAR10 datasets, we were not able to train such an alarm model. This indicates that our intuition holds true. Contrary to adversarial examples, misclassified original inputs provoke a behavior of the target model identical to correctly classified inputs and are thus correctly identified as benign by our system.

Detecting NLP and Audio Attacks. With the following experiments, we evaluated the generalizability of our method to different application domains. During this analysis, we crafted adversarial examples based on an NLP and audio dataset. As previously introduced, we used the IMDb and Mozilla Common Voice datasets. We assessed whether our framework is able to detect adversarial examples of this domain when processed by 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. With the IMDb dataset, we used Algorithm 2 (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 preserved the lengths of the classified sentences and reduced the distance between benign and adversarial examples. Here, we are able to report a detection f1-score of 0.966.

For the audio dataset, we used Carlini and Wagner's approach to create adversarial examples [75]. Extracting the features of the audio files leads to neuron activation sequences of different lengths. This is the result of 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 0.860.

7. Robustness Against Adaptive Attacks

In this section we evaluate adaptive white-box attacks in which the attacker has perfect knowledge of the target model and our detection method. For this purpose we assume the second threat model presented in Section 3. As shown by Carlini and Wagner [23], the majority of proposed detection methods can easily be bypassed by an adaptive attack.

7.1. Experimental Setup

To perform such an attack, we have to combine our target and alarm models. We use the following function $G(x)$ to represent this combination of classifier and detector building $N_{secured}$:

$$G(x)_i = \begin{cases} Z_F(x)_i, & \text{if } i \leq N \\ A_D(x) \cdot \max\limits_j Z_F(x)_j, & \text{if } i = N+1 \end{cases}$$

where $A_D(x) = (2(\max_j Z_F(x)_j > 0) - 1) \cdot Z_D(x) + 1$ decides if our secured model will output an alarm signal or not. $N_{secured}$ contains eleven output logits. The first ten logits represent the classification output, while the last logit shows the output of the detector. Note, that this formula allows the first ten logits to be negative.

With $G(x)$ we generated adversarial examples for $N_{secured}$, using the C&W method. Here, we used the code published together with paper [76]. We performed this attack on our four main target models: LeNet and kerasExM for MNIST as well as kerasExC and ResNet for CIFAR10.

7.2. Evaluation

To evaluate the robustness of our method against adaptive attacks, we use the mean l_2-distance between adversarial and benign images to either fool N_{target} or $N_{secured}$ as well as the attack success rates as metrics. We generated adversarial images for N_{target} and $N_{secured}$ using the same attack parameters to preserve comparability. This allows an estimation of the security improvement due to our detection scheme. The attack parameters listed in Appendix E are chosen to reach a 100% success rate when attacking N_{target} building the baseline security level.

In Table 6 we summarize our results. The four secured target models require a significantly higher l_2-distortion compared to their unsecured counterparts.

For the MNIST-based LeNet target model, our defense method more than doubles the required mean l_2-distance to successfully attack $N_{secured}$, compared to the unsecured version. Furthermore, we reduced the attack success rate to 44.6%.

Similarly, for kerasExM we increase the mean l_2-distance to fool our secured model by 80.74%. We report an attack success rate of 97.6%.

For CIFAR10 and the kerasExC target model we achieve similar results. When attacking N_{target}, the mean l_2-distance is 0.430. In contrast to that, when attacking $N_{secured}$, the adversarial images show a mean l_2-distance of 0.853 with respect to their benign counterparts. Moreover, only 53.3% of adversarial examples of the adaptive attack are successful.

Concerning our ResNet target model we are able to report a minor improvement of its security level. This is due to its complexity and size of the resulting feature space. As mentioned above, we used the same alarm model architecture for all test scenarios to show the simplicity of our approach. In this case, a bigger alarm model is required to cope with the extracted features more efficiently.

With our numerical study we show the robustness of our concept against adaptive attacks. To support our conclusion, we show the adversarial examples which are able to fool our secured kerasExM and kerasExC target models in Fig. 4. For both datasets we show the original images in the first and the adversarial counterparts in the second row. We report a significant difference between benign and adversarial images. The perturbations required to fool our secured systems are clearly visible which enables a human expert to identify the adversarial examples. This becomes even more clear when comparing the images to the previously successful adversarial examples in Fig. 2. With respect to our introduced threat models, we emphasize that the resulting adversarial attacks may not be considered successful. In our threat model we restricted the capabilities of the attacker to mutate the inputs of our target models, such that the changes are not easily visible to a human expert. This is not fulfilled here. Hence, we can report a significant security improvement of our target DNNs, even during white-box adaptive attacks.

TABLE 6: RESULTS OF ADAPTIVE ATTACKS

Dataset	Target Model	Attack Success Rate		Mean l_2 Distortion	
		unsecure	*secure*	*unsecure*	*secure*
MNIST	LeNet	100%	44.6%	2.171	4.422
	KerasExM	100%	97.6%	1.506	2.722
CIFAR10	KerasExC	100%	53.3%	0.430	0.853
	ResNet	100%	99.7%	0.133	0.378

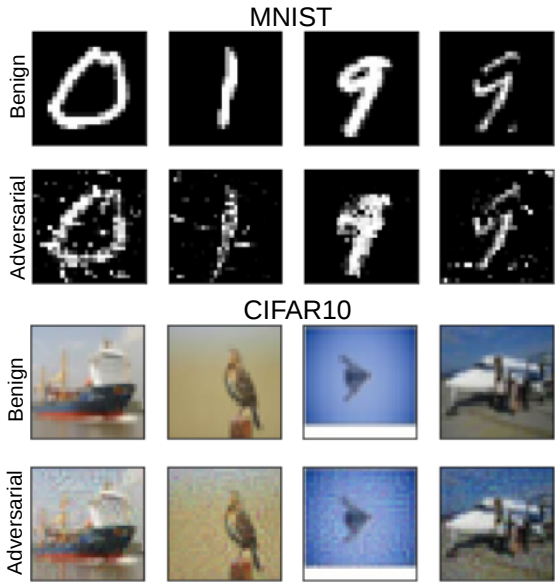

Figure 4: C&W-based adversarial images during adaptive attacks. The first two rows show MNIST-based examples for the kerasExM target model. The lower two rows are based on CIFAR10 and kerasExC.

8. Discussion

With the in-depth experiments in this paper we show the importance of the dense layers analysis in future NN defense strategies. In Section 6, we sum up the most important results to underline our conclusion. Nonetheless, some aspects regarding detection performance, transfer-

ability, and real-world applications, as well as a comparison to related work, require further discussion.

First, we want to discuss the trade-off between *false positive* and *false negative* errors during detection. Without optimizing the detection threshold of the alarm models we report the *false negative* rate to be higher than the *false positive* rate. Hence, we recommend further use-case specific adaptations before deploying our method in security-sensitive setups. Here, the base-rate fallacy is worth mentioning. The number of false alarms is often a crucial performance characteristic of attack detection systems [77] and should not be neglected when aiming to reduce the *false negative* rate.

Throughout our cross-testing experiments, we evaluated the generality of our method. We give evidence for the distinct behavior of NNs when confronted with adversarial examples, independently of the used attack method. This allows two main conclusions: Future, yet unknown attacks seem detectable using our concept. Furthermore, it allows a ranking of attack methods. This ranking is based on two findings. First, the difficulty in detecting each attack, expressed by the f1-score of the respective alarm model. Secondly, by the performance of the alarm model created for the attack itself when detecting examples crafted with other attack methods. As an example, if we focus on the C&W attack on the ResNet target model. We can clearly see that this attack method can only be detected by the alarm model specifically created for this purpose. Hence, we deduce the C&W attack being the most powerful method used here. This 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 the most powerful white-box attack.

A possible restriction our approach may suffer 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 two concepts. The maintainer of the target model creates a so-called *substitute model* which performs the same task as the target model itself, achieving a similar accuracy. If this *substitute 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 this idea. Alternatively, additional dense layers may be added to the original target model. Our experiments with LSTM target models indicate the practicability of this idea as well. The target models contain dense layers after the LSTM units in which the main learning power resides. A thorough evaluation of both solutions is out of scope in the context of this paper.

Finally, we want to emphasize the simplicity of our approach. During our research on related work, we noticed the defense strategies to be rather unintuitive and having 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. One important aspect when comparing our method to related techniques is worth mentioning. We did not optimize our framework to its full detection capability. Note, that we solely used one alarm model architecture during our main experiments on MNIST and CIFAR10. Furthermore, we did not tune the training process of the alarm models in relation to each target model. We regard this as out of scope for this study. Moreover, it shows the simplicity and generality of our approach. Even with a generic setting, promising results can be achieved. If the user further tunes the settings in a use-case-specific manner, superior results are achievable. The drawback of this approach is the limited possibility of comparing our method directly to related work. As related detection schemes are often optimally adapted to the underlying use-case and dataset, a direct comparison would not lead to meaningful conclusions. Therefore, we solely compared our method to state-of-the-art techniques on a conceptual level.

9. Conclusion

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

First, in the training phase we observe the dense-layer activation patterns of the model to protect. For this purpose, we extract the neuron coverage of the target model to directly train a secondary NN we call alarm model. This alarm model distinguishes between activations sequences extracted during the classification of benign and adversarial inputs. This approach is motivated by our main hypothesis that the dense layers of the target model carry security sensitive information. Thus, the alarm model is trained to detect malicious activity patterns triggered by adversarial examples during classification time.

In the second phase, the target model runs in secure operation mode, which is enabled by enhancing it with our trained alarm model. When the target model classifies new, unseen inputs, the alarm model runs in parallel and produces 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, while improving overall application robustness significantly. In our proof-of-concept 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 state-of-the-art adversarial examples, but also against future, yet unknown attacks. Finally, with adaptive attacks we show that an attacker needs to perform significantly more adversarial perturbations to attack our detection-enhanced system, compared to attacking the unsecured target model.

Finally, we see three points of interest for future work: (1) Experiments regarding dense-layer-free target models. This includes a comparison between substitute models and adding additional dense layers to the original target models. (2) Further evaluation of the performance overhead when detecting new and unknown attack methods, for example training the alarm model with multiple attack methods and detecting a single new attack. (3) A thorough evaluation of the performance overhead compared to related work and potential solutions to automatically handle detected adversarial examples.

References

[1] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, Jun. 2016.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016.

[3] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[4] 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*, vol. 19, no. 6, pp. 1893–1905, Nov 2015.

[5] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia, "Exploiting machine learning to subvert your spam filter," *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, pp. 7:1–7:9, 2008.

[6] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *International Conference on Learning Representations*, 2014.

[7] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 1–18, 2017.

[8] L. Ma, Y. Liu, J. Zhao, Y. Wang, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, C. Chen, and et al., "Deepgauge: multi-granularity testing criteria for deep learning systems," *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018.

[9] Y. Sun, X. Huang, and D. Kroening, "Testing deep neural networks," *arXiv preprint arXiv:1803.04792*, 2018.

[10] A. Odena and I. Goodfellow, "Tensorfuzz: Debugging neural networks with coverage-guided fuzzing," *arXiv preprint arXiv:1807.10875*, 2018.

[11] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," *arXiv preprint arXiv:1810.00069*, 2018.

[12] R. Fletcher, *Practical methods of optimization*. John Wiley & Sons, 2013.

[13] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *International Conference on Learning Representations*, 2015.

[14] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[15] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[16] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2574–2582, Jun. 2016.

[17] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," *IEEE European Symposium on Security and Privacy*, pp. 372–387, Mar. 2016.

[18] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2019.

[19] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," *IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, May 2017.

[20] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 86–94, Jul. 2017.

[21] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *Proceedings of 5th International Conference on Learning Representations*, 2017.

[22] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.

[23] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 3–14, 2017.

[24] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[25] S. Sankaranarayanan, A. Jain, R. Chellappa, and S. N. Lim, "Regularizing deep networks using efficient layerwise adversarial training," *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[26] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[27] N. Narodytska and S. P. Kasiviswanathan, "Simple black-box adversarial perturbations for deep networks," *arXiv preprint arXiv:1612.06299*, 2016.

[28] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," *Proceedings of the Asia Conference on Computer and Communications Security*, pp. 506–519, 2017.

[29] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, "A study of the effect of jpg compression on adversarial images," *arXiv preprint arXiv:1608.00853*, 2016.

[30] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," *arXiv preprint arXiv:1711.00117*, 2017.

[31] N. Das, M. Shanbhogue, S.-T. Chen, F. Hohman, L. Chen, M. E. Kounavis, and D. H. Chau, "Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression," *arXiv preprint arXiv:1705.02900*, 2017.

[32] R. Shin and D. Song, "Jpeg-resistant adversarial images," *NIPS Workshop on Machine Learning and Computer Security*, 2017.

[33] W. Xu, D. Evans, and Y. Qi, "Feature squeezing mitigates and detects carlini/wagner adversarial examples," *arXiv preprint arXiv:1705.10686*, 2017.

[34] 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*, pp. 1–1, 2018.

[35] Y. Luo, X. Boix, G. Roig, T. Poggio, and Q. Zhao, "Foveation-based mechanisms alleviate adversarial examples," *arXiv preprint arXiv:1511.06292*, 2015.

[36] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, "Adversarial examples for semantic segmentation and object detection," *IEEE International Conference on Computer Vision (ICCV)*, pp. 1378–1387, Oct. 2017.

[37] Q. Wang, W. Guo, K. Zhang, I. Ororbia, G. Alexander, X. Xing, X. Liu, and C. L. Giles, "Learning adversary-resistant deep neural networks," *arXiv preprint arXiv:1612.01401*, 2016.

[38] A. S. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients," *Thirty-second AAAI conference on artificial intelligence*, 2018.

[39] C. Lyu, K. Huang, and H. Liang, "A unified gradient regularization family for adversarial examples," *IEEE International Conference on Data Mining*, pp. 301–309, Nov. 2015.

[40] U. Shaham, Y. Yamada, and S. Negahban, "Understanding adversarial training: Increasing local stability of supervised models through robust optimization," *Neurocomputing*, vol. 307, pp. 195 – 204, 2018.

[41] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," *IEEE Symposium on Security and Privacy (SP)*, pp. 582–597, May 2016.

[42] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[43] N. Akhtar, J. Liu, and A. Mian, "Defense against universal adversarial perturbations," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3389–3398, Jun. 2018.

[44] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in Neural Information Processing Systems 27*, pp. 2672–2680, 2014.

[45] H. Lee, S. Han, and J. Lee, "Generative adversarial trainer: Defense to adversarial perturbations with gan," *arXiv preprint arXiv:1705.03387*, 2017.

[46] G. Jin, S. Shen, D. Zhang, F. Dai, and Y. Zhang, "Ape-gan: Adversarial perturbation elimination with gan," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3842–3846, May 2019.

[47] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-GAN: Protecting classifiers against adversarial attacks using generative models," *International Conference on Learning Representations*, 2018.

[48] S. Baluja and I. Fischer, "Adversarial transformation networks: Learning to generate adversarial examples," *arXiv preprint arXiv:1703.09387*, 2017.

[49] D. Hendrycks and K. Gimpel, "Visible progress on adversarial images and a new saliency map," *arXiv preprint arXiv:1608.00530*, 2016.

[50] Z. Gong, W. Wang, and W.-S. Ku, "Adversarial and clean data are not twins," *arXiv preprint arXiv:1704.04960*, 2017.

[51] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pp. 135–147, 2017.

[52] N. Carlini and D. Wagner, "Magnet and "efficient defenses against adversarial attacks" are not robust to adversarial examples," *arXiv preprint arXiv:1711.08478*, 2017.

[53] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (statistical) detection of adversarial examples," *arXiv preprint arXiv:1702.06280*, 2017.

[54] H. Hosseini, Y. Chen, S. Kannan, B. Zhang, and R. Poovendran, "Blocking transferability of adversarial examples in black-box learning systems," *arXiv preprint arXiv:1703.04318*, 2017.

[55] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," *Proceedings of 5th International Conference on Learning Representations (ICLR)*, 2017.

[56] J. Lu, T. Issaranon, and D. Forsyth, "Safetynet: Detecting and rejecting adversarial examples robustly," *IEEE International Conference on Computer Vision (ICCV)*, pp. 446–454, Oct. 2017.

[57] X. Li and F. Li, "Adversarial examples detection in deep networks with convolutional filter statistics," *IEEE International Conference on Computer Vision (ICCV)*, pp. 5775–5783, Oct. 2017.

[58] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *arXiv preprint arXiv:1703.00410*, 2017.

[59] S. Ma, Y. Liu, G. Tao, W. Lee, and X. Zhang, "NIC: detecting adversarial samples with neural network invariant checking," *26th Annual Network and Distributed System Security Symposium, NDSS, San Diego, California, USA*, Feb. 2019.

[60] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin, "On evaluating adversarial robustness," *arXiv preprint arXiv:1902.06705*, 2019.

[61] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, Jun. 2009.

[62] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[63] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Jun. 2011.

[64] "Mozilla Common Voice dataset," accessed: 2019-05-20. [Online]. Available: https://voice.mozilla.org/datasets

[65] 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.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[66] "Keras Convolutional Neural Network for MNIST," accessed: 2019-05-20. [Online]. Available: https://keras.io/examples/mnist_cnn/

[67] "ResNet for CIFAR10," accessed: 2019-05-20. [Online]. Available: https://keras.io/examples/cifar10_resnet/

[68] "Keras Convolutional Neural Network for CIFAR10," accessed: 2019-05-20. [Online]. Available: https://keras.io/examples/cifar10_cnn/

[69] N. Frosst, S. Sabour, and G. E. Hinton, "DARCCC: detecting adversaries by reconstruction from class conditional capsules," *arXiv preprint arXiv:1811.06969*, 2018.

[70] "Mozilla Project DeepSpeech," accessed: 2019-05-20. [Online]. Available: https://github.com/mozilla/DeepSpeech

[71] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[72] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," *Advances in Neural Information Processing Systems 30*, pp. 3856–3866, 2017.

[73] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[74] J. Rauber, W. Brendel, and M. Bethge, "Foolbox: A python toolbox to benchmark the robustness of machine learning models," *arXiv preprint arXiv:1707.04131*, 2017.

[75] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," *IEEE Security and Privacy Workshops (SPW)*, pp. 1–7, May 2018.

[76] "Nicholas Carlini: Breaking Neural Network Detection Schemes," accessed: 2019-08-21. [Online]. Available: https://nicholas.carlini.com/code/nn_breaking_detection

[77] S. Axelsson, "The base-rate fallacy and the difficulty of intrusion detection," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 3, p. 186–205, Aug. 2000.

Appendix A.
Alarm Model Architecture

In the following, we provide more information on the architecture of the alarm 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.

Appendix B.
All Result Values

In the Tables 9, 10, 11, and 12 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 emphasize the best test result in each table in bold. In Table 13 we summarize the precision and recall for our main experiments shown in Table 2.

Appendix C.
Confusion Matrix Values

In Table 14 we show the confusion matrix values of each performed test. Note, that we tested the alarm models against multiple nonoverlapping batches of test data to show the standard deviation of the according results.

Appendix D.
Adversarial Example Generation for the NLP Scenario

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

Appendix E.
C&W Attack Parameters for the Adaptive Attack

Table 7 shows the attack parameters of the C&W attack during the adaptive white-box attacks for the MNIST dataset. Table 8 shows the attack parameters of the C&W attack during the adaptive white-box attacks for the CIFAR10 dataset.

TABLE 7: C&W ATTACK PARAMETERS DURING THE ADAPTIVE ATTACK FOR MNIST

Parameter Name	Parameter Value
max-iteration	2000
batch-size	100
learning-rate	0.1
binary-search-steps	5

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 IMDb dataset containing movie reviews.

TABLE 8: C&W ATTACK PARAMETERS DURING THE ADAPTIVE ATTACK FOR CIFAR10

Parameter Name	Parameter Value
max-iteration	200
batch-size	100
learning-rate	0.01
binary-search-steps	3

TABLE 9: ALL RESULT VALUES FOR THE MNIST DATASET AND TARGET MODEL *LeNet*

Alarm Models trained with:	**Accuracy and f1-Scores of the Alarm Models when tested against: (acc; f1-score)**													
	FGSM		***C&W***		***DeepFool***		***PGD***		***BIM***		***all attacks combined***		***transferred examples***	
FGSM	0.988	0.988	0.802	0.758	0.953	0.945	0.987	0.987	0.987	0.987	0.945	0.943	0.924	0.840
C&W	0.943	0.942	0.977	0.977	0.943	0.933	0.957	0.956	0.958	0.957	0.957	0.956	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	0.962	0.961	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**	0.925	0.920	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	0.929	0.924	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	0.980	0.980	0.962	0.930
transferred examples	0.977	0.977	0.925	0.921	0.949	0.941	0.968	0.968	0.966	0.966	0.958	0.957	0.960	0.926

TABLE 10: ALL RESULT VALUES FOR THE MNIST DATASET AND TARGET MODEL *kerasExM*

Alarm Models trained with:	**Accuracy and f1-Scores of the Alarm Models when tested against: (acc; f1-score)**													
	FGSM		***C&W***		***DeepFool***		***PGD***		***BIM***		***all attacks combined***		***transferred examples***	
FGSM	0.992	0.992	0.767	0.700	0.917	0.892	0.992	0.992	0.985	0.985	0.931	0.927	0.893	0.792
C&W	0.973	0.973	0.975	0.975	0.967	0.962	0.968	0.968	0.961	0.960	0.970	0.970	0.959	0.933
DeepFool	0.986	0.986	0.867	0.850	0.985	0.982	0.988	0.988	0.980	0.980	0.960	0.959	0.928	0.871
PGD	0.992	0.991	0.734	0.641	0.899	0.865	0.992	**0.992**	0.986	0.986	0.920	0.915	0.885	0.773
BIM	0.992	0.992	0.752	0.674	0.912	0.886	0.992	**0.992**	0.991	0.991	0.928	0.923	0.893	0.791
all attacks combined	0.984	0.984	0.972	0.972	0.980	0.977	0.984	0.984	0.984	0.984	0.981	0.982	0.967	0.946
transferred examples	0.983	0.983	0.916	0.910	0.967	0.960	0.982	0.982	0.975	0.975	0.965	0.965	0.966	0.943

TABLE 11: ALL RESULT VALUES FOR THE CIFAR10 DATASET AND TARGET MODEL *kerasExC*

Alarm Models trained with:	**Accuracy and f1-Scores of the Alarm Models when tested against: (acc; f1-score)**													
	FGSM		***C&W***		***DeepFool***		***PGD***		***BIM***		***all attacks combined***		***transferred examples***	
FGSM	0.843	0.847	0.589	0.470	0.839	0.842	0.840	0.844	0.840	0.843	0.789	0.783	0.646	0.574
C&W	0.699	0.679	0.739	0.733	0.662	0.625	0.673	0.642	0.665	0.631	0.687	0.663	0.556	0.444
DeepFool	0.851	0.852	0.571	0.414	0.853	**0.855**	0.853	**0.855**	0.853	**0.855**	0.795	0.786	0.631	0.533
PGD	0.839	0.840	0.584	0.449	0.841	0.842	0.841	0.843	0.841	0.843	0.789	0.780	0.628	0.535
BIM	0.848	0.850	0.580	0.434	0.849	0.850	0.849	0.851	0.850	0.852	0.794	0.758	0.627	0.527
all attacks combined	0.708	0.771	0.678	0.740	0.709	0.771	0.709	0.772	0.709	0.772	0.704	0.767	0.678	0.740
transferred examples	0.749	0.777	0.577	0.557	0.738	0.766	0.743	0.771	0.743	0.770	0.712	0.735	0.700	0.721

TABLE 12: ALL RESULT VALUES FOR THE CIFAR10 DATASET AND TARGET MODEL *ResNet*

Alarm Models trained with:	**Accuracy and f1-Scores of the Alarm Models when tested against: (acc; f1-score)**													
	FGSM		***C&W***		***DeepFool***		***PGD***		***BIM***		***all attacks combined***		***transferred examples***	
FGSM	0.810	0.815	0.559	0.446	0.812	0.820	0.812	0.821	0.812	0.821	0.761	0.761	0.680	0.651
C&W	0.701	0.715	0.707	0.727	0.698	0.716	0.697	0.716	0.695	0.713	0.703	0.722	0.692	0.710
DeepFool	0.819	0.827	0.568	0.469	0.822	0.833	0.823	**0.835**	0.823	0.834	0.774	0.778	0.684	0.660
PGD	0.822	0.826	0.560	0.434	0.824	0.831	0.826	0.833	0.825	0.832	0.772	0.769	0.670	0.628
BIM	0.819	0.825	0.559	0.445	0.821	0.830	0.822	0.832	0.822	0.832	0.770	0.771	0.664	0.626
all attacks combined	0.788	0.812	0.677	0.686	0.792	0.818	0.793	0.820	0.793	0.819	0.771	0.796	0.729	0.750
transferred examples	0.672	0.613	0.570	0.440	0.655	0.593	0.655	0.595	0.653	0.592	0.638	0.568	0.687	0.646

TABLE 13: PRECISION AND RECALL OF THE INDIVIDUAL ALARM MODELS. EACH ALARM MODEL IS TRAINED AND TESTED WITH THE FEATURES EXTRACTED FOR ONE ATTACK METHOD. SUPPORTING THE MAIN RESULTS OF THE PROOF-OF-CONCEPT EVALUATION

Dataset	**Target Model**	**Precision and recall of the Alarm Models with the according attacks: (prec.; rec.)**				
		FGSM	***C&W***	***DeepFool***	***PGD***	***BIM***
MNIST	LeNet	0.995; 0.981	0.981; 0.974	0.990; 0.972	0.995; 0.987	0.994; 0.986
	kerasExM	0.995; 0.989	0.981; 0.968	0.989; 0.975	0.995; 0.990	0.994; 0.989
CIFAR10	kerasExC	0.871; 0.824	0.719; 0.749	0.868; 0.843	0.853; 0.833	0.864; 0.850
	ResNet	0.861; 0.774	0.781; 0.680	0.891; 0.782	0.870; 0.799	0.881; 0.788

TABLE 14: CONFUSION MATRIX VALUES FOR ALL DATASETS, TARGET MODELS, AND ATTACK METHODS. EACH RESULT CORRESPONDS TO THE DETECTION OF ADVERSARIAL ATTACK METHODS WITH THE SPECIFIED TARGET MODEL

Dataset	**Target Model**	**Attack**	**Performance of the Alarm Models when tested against the according attack method**			
			True Positive	***True Negative***	***False Positive***	***False Negative***
MNIST	LeNet	FGSM	0.981±0.004	0.995±0.002	0.005±0.002	0.019±0.004
		C&W	0.974±0.005	0.980±0.003	0.020±0.003	0.026±0.005
		DeepFool	0.972±0.006	0.992±0.002	0.008±0.002	0.028±0.006
		PGD	0.987±0.003	0.995±0.002	0.005±0.002	0.013±0.003
		BIM	0.986±0.004	0.994±0.002	0.006±0.002	0.014±0.004
	kerasExM	FGSM	0.989±0.003	0.995±0.002	0.005±0.002	0.011±0.003
		C&W	0.968±0.005	0.981±0.004	0.019±0.004	0.032±0.005
		DeepFool	0.975±0.006	0.992±0.003	0.008±0.003	0.025±0.006
		PGD	0.990±0.002	0.995±0.003	0.005±0.003	0.010±0.002
		BIM	0.989±0.003	0.994±0.002	0.006±0.002	0.011±0.003
	LSTM	Transfer	0.890±0.012	0.968±0.005	0.032±0.005	0.110±0.012
	CapsuleNN	Transfer	0.946±0.022	0.929±0.036	0.071±0.036	0.054±0.022
CIFAR10	kerasExC	FGSM	0.824±0.010	0.864±0.015	0.136±0.015	0.176±0.010
		C&W	0.749±0.018	0.729±0.011	0.271±0.011	0.251±0.018
		DeepFool	0.843±0.010	0.864±0.011	0.136±0.011	0.157±0.010
		PGD	0.833±0.013	0.849±0.012	0.151±0.012	0.167±0.013
		BIM	0.841±0.010	0.860±0.015	0.140±0.015	0.159±0.010
	ResNet	FGSM	0.774±0.015	0.852±0.012	0.148±0.012	0.226±0.015
		C&W	0.680±0.015	0.743±0.010	0.257±0.010	0.320±0.015
		DeepFool	0.782±0.015	0.875±0.011	0.125±0.011	0.218±0.015
		PGD	0.799±0.010	0.857±0.008	0.143±0.008	0.201±0.009
		BIM	0.788±0.009	0.865±0.013	0.135±0.013	0.212±0.009
NLP and Audio	LSTM (NLP)	Custom Attack	0.973±0.032	0.962±0.032	0.037±0.031	0.027±0.032
	DeepSpeech	Carlini's Attack	1.000±0.000	0.674±0.069	0.326±0.069	0.000±0.000