AN ENSEMBLE APPROACH FOR EXPLANATION-BASED ADVERSARIAL DETECTION

A Thesis

by

RAJ VARDHAN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Guofei Gu
Committee Members,	Xia Hu
	Jeyavijayan Rajendran
Head of Department,	Duncan M. (Hank) Walker

May 2021

Major Subject: Computer Science

ABSTRACT

Recent research has shown Deep Neural Networks (DNNs) to be vulnerable to adversarial examples that induce desired misclassifications in the models. Such risks impede the application of machine learning in security-sensitive domains. Several defense methods have been proposed against adversarial attacks to detect adversarial examples at test time or to make machine learning models more robust. However, while existing methods are quite effective under blackbox threat model, where the attacker is not aware of the defense, they are relatively ineffective under whitebox threat model, where the attacker has full knowledge of the defense.

In this thesis, we propose ExAD, a framework to detect adversarial examples using an ensemble of explanation techniques. Each explanation technique in ExAD produces an explanation map identifying the relevance of input variables for the model's classification. For every class in a dataset, the system includes a detector network, corresponding to each explanation technique, which is trained to distinguish between normal and abnormal explanation maps. At test time, if the explanation map of an input is detected as abnormal by any detector model of the classified class, then we consider the input to be an adversarial example. We evaluate our approach using six state-of-the-art adversarial attacks on three image datasets. Our extensive evaluation shows that our mechanism can effectively detect these attacks under blackbox threat model with limited false-positives. Furthermore, we find that our approach achieves promising results in limiting the success rate of whitebox attacks.

DEDICATION

To my father and mother, who have supported and inspired me for all my endeavours in life.

ACKNOWLEDGMENTS

I would like to express my deepest appreciation and gratitude to my advisor Prof. Gu for his guidance and mentorship in carrying out the research work detailed in this thesis. I am also thankful to the committee members, Dr. Hu and Dr. Rajendran, for their insightful reviews that helped improve this work. I would also like to acknowledge the contributions of Ninghao Liu, Weijie Fu, Zhenyu Hu, and Phakpoom Chinprutthiwong who helped me in revising the writing and in certain aspects of experimentation and design of the proposed system.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

The thesis committee for this work include Dr. Guofei Gu (Chair) from the Department of Computer Science, Dr. Xia Hu of the Department of Computer Science and Dr. Jeyavijayan Rajendran of the Department of Electrical Engineering at Texas A& M University.

In an early stage of this work, the design of an explanation-based adversarial detection mechanism was reviewed by Ninghao Liu, Prof. Hu, and Prof. Gu.

The analysis of a state-of-the-art system called MagNet was done in collaboration with Zhenyu Hu.

Issues encountered in conducting the whitebox attack were resolved in collaboration with Weijie Fu.

Funding Sources

This material is based upon work supported in part by the National Science Foundation (NSF) under Grant no. 1816497. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

NOMENCLATURE

DNN	Deep Neural Network
CNN	Convolutional Neural Network
JSMA	Jacobian-based Saliency Map Attack
BIM	Basic Iterative Method
MIM	Momentum Iterative Method
CW	Carlini and Wagner Attack
IG	Integrated Gradients
GBP	Guided Backpropagation
LRP	Layer-wise Relevance propagation
PA	Pattern Attribution

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

In recent years, Deep Neural Networks (DNNs) are being increasingly adopted in a wide range of tasks such as face-recognition [4], natural language processing [5], and malware classification [6]. This trend can be attributed to the superior performance achieved by DNNs in solving computational tasks that rely on high-dimensional data. However, increasing adoption of DNNs to security-critical applications, such as self-driving cars and malware classification, is hindered by the vulnerability of DNNs to adversarial attacks [7, 8, 9, 10]. Specifically, minor yet carefully computed perturbations to natural inputs can cause DNNs to misclassify.

Several methods have been proposed for defending against adversarial examples. One direction of research is to improve the robustness of neural networks, such as through adversarial training [9] or gradient masking [11]. However, subsequent works have shown that neural network architectures modified with such techniques can still be attacked [12]. Another research direction is adversarial detection, where the goal is to detect if an input is an adversarial example or a normal example. Early works in this area either used a second neural network [13, 14, 15], or statistical tests [16, 17, 18] to classify between normal and adversarial examples. However, Carlini et al. [19] showed that while most of these mechanisms are successful against blackbox attacks, they lack robustness to whitebox attacks, where the adversary has knowledge of the defense. Although many recent methods have enhanced the detection of blackbox adversarial attacks [1, 20, 2, 3], improving the robustness to whitebox attacks remains an open problem.

One way of uncovering the reasons for the resulting misclassification of an adversarial example can be understanding why the model predicts what it predicts through explanation techniques [21, 22, 23, 24, 25, 26, 27, 28]. For an image input, the result from an explanation technique encodes the relevance of each pixel for the prediction result and is commonly referred to as an *explanation map*. Our hypothesis is that the explanation map of an adversarial example being misclassified as the target class may not be consistent with explanation maps generated for correctly classified normal examples of that class. We term the former type as *abnormal explanations* and the latter as *normal*

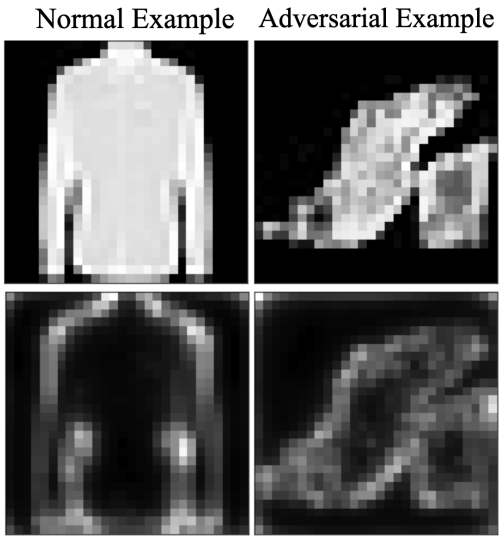

Figure 1.1: Intuition behind the proposed ExAD framework.

explanations throughout this paper. Figure 1.1 shows an intuitive example where we can observe that the explanation map of an adversarial example classified into the shirt class (bottom-right) is quite distinguishable from that of a normal example of the targeted class (bottom-left). Overall, the distinguishability between normal and abnormal explanation maps guides us in exploring the effectiveness of using explainability as a tool for detecting adversarial examples.

However, a defense method that relies on a single explanation technique may still not be robust under whitebox setting. An adaptive adversary can leverage recent findings which show that explanations can be unreliable [29] and can be manipulated to produce a target explanation map [30, 31]. Such an adaptive adversary can generate adversarial examples that not only fool the target model into producing desired misclassifications, but also fool the targeted explanation technique into producing normal explanation maps. Towards building a mitigation strategy, we take motivation from previous work on N-variant systems [32]. To provide higher resistance against attacks on software vulnerabilities, these systems combine multiple variants with disjoint exploitation sets into a single system. In context of our work, we propose to use an ensemble of multiple kinds of explanation techniques. The benefit of this approach is that it requires an adaptive adversary to construct an adversarial example that fools the target model and simultaneously fools all explana-

tion techniques. By incorporating diverse explanation techniques, we can reduce the probability that an attacker will achieve this goal.

In this thesis, using the above insights, we propose ExAD, an Ensemble approach for Explanation-based Adversarial Detection. ExAD uses an ensemble of explanation techniques wherein each technique provides an explanation map for every classification decision by a target model. To introduce explanation diversity, we include both gradient-based [21, 22, 33, 24, 23] and propagation-based [26, 27, 23] explanation techniques in ExAD. Furthermore, for any class in a dataset, the system includes a detector model associated with each explanation technique. The detector model determines if an explanation map produced by the respective explanation technique is normal or not for that class. The key idea here is to use the distinguishability between normal and abnormal explanations for any class. Finally, for a test input classified into a particular class, if the explanation map produced by any technique is detected as abnormal by the corresponding detector model, then we classify the input as an adversarial example.

To assess the effectiveness of ExAD, we evaluate it using six state-of-the-art adversarial attacks on three image datasets, namely MNIST [34], Fashion-MNIST (FMNIST) [35] and CIFAR-10 [36]. We first perform the evaluation under the blackbox threat model. Our experimental results show that we can effectively detect all attacks, achieving a detection rate above 98% (many having 100% detection rate) across the three datasets with a low false-positive rate of under 1.1%. We also compare ExAD with three state-of-the-arts Magnet [1], Feature Squeezing [2], and LID [3]. Our results show that ExAD can consistently achieve high detection rates with low false-positive rates, while state-of-the-art systems perform very well only on a subset of attacks or datasets.

More importantly, we further evaluate ExAD under whitebox threat model. We build on previous research [31, 30], and create a strong adaptive adversary to generate adversarial examples that fool the target model as well as a target explanation technique. Through experimental results, we make an interesting finding on the transferability of adaptive attacks on explanation-based detector models. We observe that on targeting a propagation-based technique, the resulting adversarial examples are more successful in fooling detector models of other propagation-based techniques (into

misclassifying an explanation map as normal) as compared to fooling detector models of gradient-based techniques. Likewise, we find that targeting a gradient-based technique transfers better to the detector model of the other gradient-based technique compared to those of propagation-based techniques. Using an ensemble of detector models corresponding to diverse techniques, ExAD achieves a mean detection rate of over 88% for this whitebox attack across the three datasets. The results indicate that our proposed defense can significantly limit the success rate of such whitebox attacks. Additionally, we find that our ensemble approach makes it considerably harder for attackers to perform more advanced whitebox attacks, such as simultaneously targeting all explanation techniques.

2. BACKGROUND AND RELATED WORK

2.1 Neural networks

A DNN is a computational graph of elementary computing units, called neurons, organized into layers that represent the extraction of successive representations from the input. We use notations consistent with previous work [20, 19] to denote an m-class DNN as a function $f : \mathbb{R}^d \rightarrow \mathbb{R}^m$. Each layer of the network takes as input the result from the previous layer. Neurons in adjacent layers are connected with links that have associated weights and biases. Some layers involve an activation function such as the non-linear ReLU [37]. Thus, the i-th layer of the network computes

$$f^i(x) = ReLU(W^i f^{i-1}(x) + b^i)$$

where W^i is a weight matrix, b^i is a vector of bias values, and ReLU is a non-linear activation function. Let $Z(x)$ denote a vector of m elements representing the output of the last layer (before softmax), known as *logits*, i.e., $Z(x) = f^n(x)$. A softmax function is used to obtain the normalized output of the network given by $y = f(x) = \text{softmax}(Z(x))$ where $x \in \mathbb{R}^d$ and $y \in \mathbb{R}^m$ with y_i representing the probability of the input being recognized as class i. Then, we represent the classification of $f(\cdot)$ on x by

$$C(x) = argmax_i(f(x)_i)$$

At test-time, a trained model is provided with test inputs X_t, and for each input $x_t \in X_t$, the model assigns its classification to be $C(x_t) = argmax_i(f(x_t)_i)$. The result is considered correct if the classification $C(x_t)$ is same as the true label $C^*(x_t)$.

2.2 Adversarial examples

Adversarial examples are crafted by imperceptibly perturbing normal inputs to cause DNNs into misclassifying them. Formally, an input to the classifier $f(\cdot)$ is termed as normal if it occurs naturally [1] or was benignly created [19]. Then, given a normal input $x \in \mathbb{R}^d$ with correctly

classified class $C(x) = c$, we call x' an (*untargeted*) adversarial example if it is close to x, i.e., $\Delta(x, x') < \epsilon$ and $C(x') \neq c$, where $\Delta(.)$ denotes a measure of similarity between two inputs and ϵ is a threshold that limits the permissible perturbations in the adversarial example. In a more restrictive case, an attacker could also target a desired class $t \neq c$ and find a x' close to x such that $C(x) = c$ and $C(x') = t$. We call x' a *targeted* adversarial example.

In the case of images, the closeness function $\Delta(.)$ and threshold ϵ should be chosen such that the adversarial example and its *seed image* (normal counterpart) are indistinguishable to a human eye. To define $\Delta(.)$, a popular distance metric is the L_p norm, defined as

$$\|d\|_p = \Big(\sum_{i=0}^{n} |v_i|^p \Big)^{\frac{1}{p}}$$

Common choices for L_p include: L_0, a measure of the number of pixels which have different values in corresponding positions in two images; L_2, which measures the standard Euclidean distance; or L_∞, a measure of the maximum change among all pixels at corresponding places in two images.

2.3 Existing attacks

Researchers have developed a number of methods for constructing adversarial examples. Broadly, these methods can be categorized into *gradient-based* attacks [12, 9, 7], which leverage gradient-based optimizations, and *content-based* attacks [38, 39], where perturbations are made in accordance with the semantics of the input content to simulate real-world scenarios. In this paper, we focus on six state-of-the-art gradient-based attacks for neural network classifiers, namely Jacobian-based Saliency Map Attack (JSMA) [8], Basic Iterative Method (BIM) [40], Momentum Iterative Method (MIM) [41], and Carlini and Wagner Attacks (CW) [12] tailored to L_0, L_2, and L_∞ norms.

2.3.1 Jacobian-based Saliency Map Attack (JSMA)

Papernot et al. proposed the Jacobian-based Saliency Map Attack (JSMA) [8], which is a greedy algorithm that optimizes the L0 distance. The attack is carried out in an iterative process. In each iteration, the attack first computes the $Jacobian$, i.e., the forward derivative of a DNN. Next, it generates an *adversarial saliency map* which is used to measure the degree to which each

output class label will be impacted when individual pixels are modified by using the Jacobian matrix. Then, the most relevant pixel is perturbed to obtain the highest benefit towards getting the model to misclassify the adversarial example. Finally, the iterative process concludes when an adversarial example is found or if the adversarial threshold is exceeded. While JSMA only modifies a small number of pixels, it has a very high computational cost as it involves computing the Jacobian matrix, making it impractical for high dimensional inputs.

2.3.2 Fast Gradient Sign Method (FGSM)

Goodfellow et al. proposed the fast gradient sign method [9] to generate adversarial examples. For a normal example x, FGSM changes all the pixels simultaneously with the same magnitude along the gradient direction and finds an adversarial example x' in the L_∞ neighborhood of x. The gradient sign is computed using backpropagation and is quite fast. This makes FGSM an efficient attack algorithm. Formally, FGSM attack generates an adversarial example using the following equation:

$$x' = x + \epsilon \cdot sign(\nabla_x J(f(x)))$$

where ∇ represents the gradient, $J(\cdot)$ is the loss function used to train the model $F(\cdot)$, and ϵ is a parameter chosen to be sufficiently small for the adversarial example to be hard to detect.

2.3.3 Basic Iterative Method (BIM):

Kurakin et al. extended the FGSM attack using an iterative optimization method [40]. For a normal example x, in each iteration, the attack applies the fast gradient sign method with a small step size, and clips pixel values of intermediate results after each step to ensure that they are in an L_∞ neighbourhood of the original example x. Formally, BIM performs the following update in the Nth iteration:

$$x_N^{adv} = Clip_{x,\epsilon} = \left\{ x_{N-1}^{adv} + \alpha \cdot sign(\nabla_x J(x_{N-1}^{adv}, y_{true})) \right\}$$

where x_0^{adv} is the original correctly classified input and α is the step-size.

2.3.4 Carlini and Wagner Attack (CW):

Carlini and Wagner proposed three attack algorithms tailored to different L_p norms, namely, L_2, L_0, and L_∞. Following notations in a previous work [20], we refer to these attacks as CW_2, CW_0, and CW_∞ respectively. With the distance metric instantiated with a particular L_p norm, the overall attack approach can be formalized as the following optimization problem. Given original example x, find perturbation δ that solves

$$\text{minimize} \quad \|\delta\|_p + \alpha \cdot f(x+\delta)$$

$$\text{such that} \quad x + \delta \in [0,1]^n$$

where $f(.)$ is an objective function such that $C(x+\delta) \neq C^*(x))$, i.e., the classifier makes a misclassification, if and only if $f(x+\delta) \leq 0$, and $\alpha > 0$ is a hyperparameter.

The attack algorithm ensures the adversarial modification results in a valid example, i.e., $0 \leq x_i + \delta_i \leq 1$ for all i. For this goal, it introduces a new variable w which it optimizes over instead of δ

$$\delta_i = \frac{1}{2}(\tanh(w_i) + 1) - x_i$$

Since $-1 \leq \tanh(w_i) \leq 1$, it can be seen that $0 \leq x_i + \delta_i \leq 1$, and therefore the resulting adversarial example will be valid.

2.3.5 Momentum Iterative Method (MIM)

Dong et al. built upon the Basic Iterative Method (BIM) with an added momentum term and proposed the Momentum Iterative Method (MIM). The technique accelerates gradient descent based attack algorithms by accumulating a velocity vector in the direction of the gradient of the loss function across iterations. For targeted attacks, the Momentum Iterative Method is formally represented as:

$$g^{t+1} = \mu \cdot g^t + \frac{\nabla_x J(f(x^t_{adv}), y_{target})}{std(\nabla_x J(f(x^t_{adv}), y_{target})}$$

$$x_{adv}^{t+1} = Clip_{[0,1]}\big(x_{adv}^{t} - \alpha \cdot Clip_{[-2,2]}(round(g^{t+1}))\big)$$

where $g^0 = 0$, $x_{adv}^0 = x$, $\alpha = \frac{\epsilon}{T}$ with T being the number of iterations, std(•) is the standard deviation and round(•) is rounding to nearest integer.

2.4 Existing work on adversarial detection

Adversarial detection is a defense approach with the goal of building a classifier g with a binary output $y \in \{0, 1\}$, where labels 0 and 1 denote that the input instance is normal or adversarial, respectively. We briefly review state-of-the-art works in detecting adversarial examples, and divide them into three categories as below.

Training a Detector. First, we can use adversarial examples to train detectors. The input into detectors can be chosen as data instances in raw feature space or the intermediate representation space of the target model. Using the former strategy, Gong *et al.* show that a simple binary classifier can learn to separate normal and adversarial instances [13]. In a related work, Grosse et al. add a new class, solely for adversarial examples, in the output layer of the model [14]. But, modifying the model architecture impacts the accuracy on normal examples. Based on the latter approach, Metzen *et al.* use representations generated by inner deep neural network layers as inputs into detectors which are augmented to the classification network [15]. By freezing the weights of the classification network before training the detectors, this method does not affect the classification accuracy on normal examples. However, in subsequent work, Carlini et al. showed that these detectors don't generalize well and lack robustness to whitebox attacks [19]. In our work, we mitigate the generalization challenge by including an attack-independent defense setting (discussed in Section 3.4.2).

Statistical Metrics. Second, we can use statistical metrics to design detectors. Grosse *et al.* [14] study two statistical distance measures, Maximum-Mean-Discrepancy and Energy Distance, where a sample is regarded as adversarial if it is rejected by statistical testing. Ma et al. estimate an LID value which assesses the space-filling capability of the region around an example by measuring the distance distribution with respect to its neighbors [3]. The authors demonstrate

that estimated LID of adversarial examples tends to be much higher than that of normal examples. However, a challenge faced by these approaches is in developing more effective and transferable metrics to separate clean instances from adversarial examples generated by different attacks.

Prediction Abnormality. Third, we can also resort to detecting the abnormality of input instances. Meng and Chen proposed Magnet [1] which learns to approximate the manifold of normal examples using autoencoders. Another method called Feature Squeezing [2] proposes reducing the degree of freedom of an adversary, such as by smoothing images or minimizing their color depth. Another recent work called Neural-network Invariant Checking (NIC) proposed leveraging the provenance channel and the activation value distribution channel in DNNs by showing that adversarial examples tend to violate either provenance invariant or value invariant [20]. However, while these methods have improved the detection rates on blackbox attacks, they have shown very limited success against whitebox attacks. Zhang *et al.* proposed a detection method based on perturbation of saliency maps [42]. The authors find that on adding adversarial perturbations, the saliency of the adversarial example is also perturbed compared to that of the seed image. However, this difference between saliency may not be effective because, at test-time, we do not know the class from which an adversarial example originated. Therefore, we do not know what its normal saliency would look like had the example not been perturbed. Besides, explanations have also been used by Liu et al. for a different goal of crafting adversarial examples [43]. In this paper, we further explore the premise of using explanations, but to detect adversarial examples and based on a fundamentally different approach. Our work was done concurrently with a similar approach presented by Wang et al. [44]. In contrast, we have the following differentiating aspects. First, our work offers a more detailed evaluation on whitebox attacks. Second, we provide a discussion on the fragility of an explanations-based defense. Finally, we compare the proposed method with a number of state-of-the-art detection systems.

2.5 Explainable machine learning

Our work utilizes recent advances in explainable machine learning. Specifically, we focus on *local explainability* methods [22, 45] which explain the output of DNN models for a given input.

For computer vision models, these techniques identify which regions in an input image are most responsible for the prediction result. The explanation result is often termed as a *saliency map* [21], or more generally, an *explanation map* [31]. Naturally, our defense is compatible with models that are inherently explainable (e.g., linear models) and models that produce an explanation result along with the prediction [46, 47]. However, we focus on local explainability methods as they build on top of existing models. This allows us to add our adversarial detection capability to any existing blackbox model without sacrificing its prediction power for explainability, or putting the burden of producing explanations during classification.

Among local explanation techniques, backpropagation-based methods have gained considerable attention. These can be further categorized into the following. The first is *gradient-based* techniques which rely on the gradient of the neural network function to generate explanations [21, 22, 33, 24, 23]. The second category is *propagation-based* techniques [26, 27, 23]. These techniques view the neural network as a computational graph, and generate explanations by starting with the prediction score at the output layer and progressively redistributing it backwards by means of propagation rules until the input layer is reached. To achieve diversity in explanation methods, we use both gradient-based and propagation-based explanation techniques, which we discuss further in Section 3.3.

3. DESIGN

3.1 Threat model

In designing our defense, we assume that the attacker has complete knowledge of the target classifier $f(\cdot)$ including its architecture and parameters. This is a conservative and practical assumption, consistent with prior works [20, 2, 1]. Also, depending upon whether the attacker has knowledge of the defense, we consider two types of threat models. First, we consider *blackbox* attack, where the attacker does not have any knowledge of the defense mechanism. Second, we consider *whitebox* attack, where the attacker has complete knowledge of the defense mechanism including its structure and parameters. For ExAD, this implies that the attacker has full knowledge of the explanation techniques and detector models.

3.2 Overview of ExAD

ExAD is a framework that uses an ensemble of explanation techniques to detect adversarial examples. The role of explanation techniques is to allow ExAD to examine the reasons for the misclassification of adversarial examples. Our hypothesis is that the explanations of adversarial examples being misclassified as the target class (*abnormal explanations*) may not be consistent with explanations generated for correct classifications of normal examples of that class (*normal explanations*). Our design relies upon the consistency of normal explanations, and their distinguishability from abnormal explanations. We provide an intuitive example for the distinguishability aspect in Figure 1.1. Further examples showing the consistency aspect can be found in Figure 3.1. The first row shows five normal examples from the Coat class of FMNIST dataset. The third row shows normal examples from the Airplane class of CIFAR-10 dataset. The fifth row shows normal examples of class Three from MNIST dataset. The second, fourth, and sixth rows show corresponding explanations for the preceeding row using the IG, LRP, and GBP techniques, respectively.

While we provide such motivating examples, it is worth noting that an explanation itself may

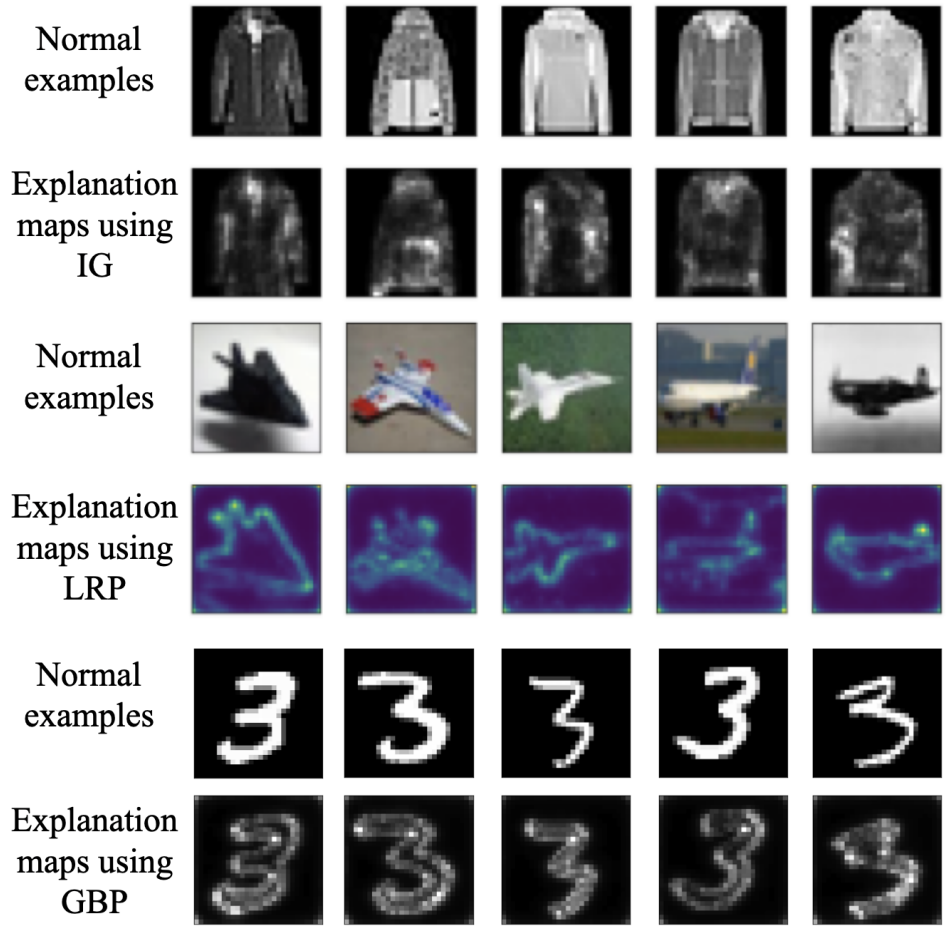

Figure 3.1: Similarity in normal explanations.

be incomprehensible to humans as recent work has shown neural networks to use non-robust features (that may not align with human perception of a class) to make predictions [48]. Therefore, even the distinguishability may not necessarily be apparent to humans. Nevertheless, we empirically show that we can train detector models to learn to distinguish between normal and abnormal explanations.

An overview of our approach is as follows. First, we train the target model as usual on the clean training set. Second, we use a set of diverse explanation techniques to generate explanation maps for normal examples of each class. Third, for every class, we train a detector model corresponding to each technique. The detector model identifies if the explanation map of an example is normal for the classified class. We study two approaches to build the detector model: a binary classifier

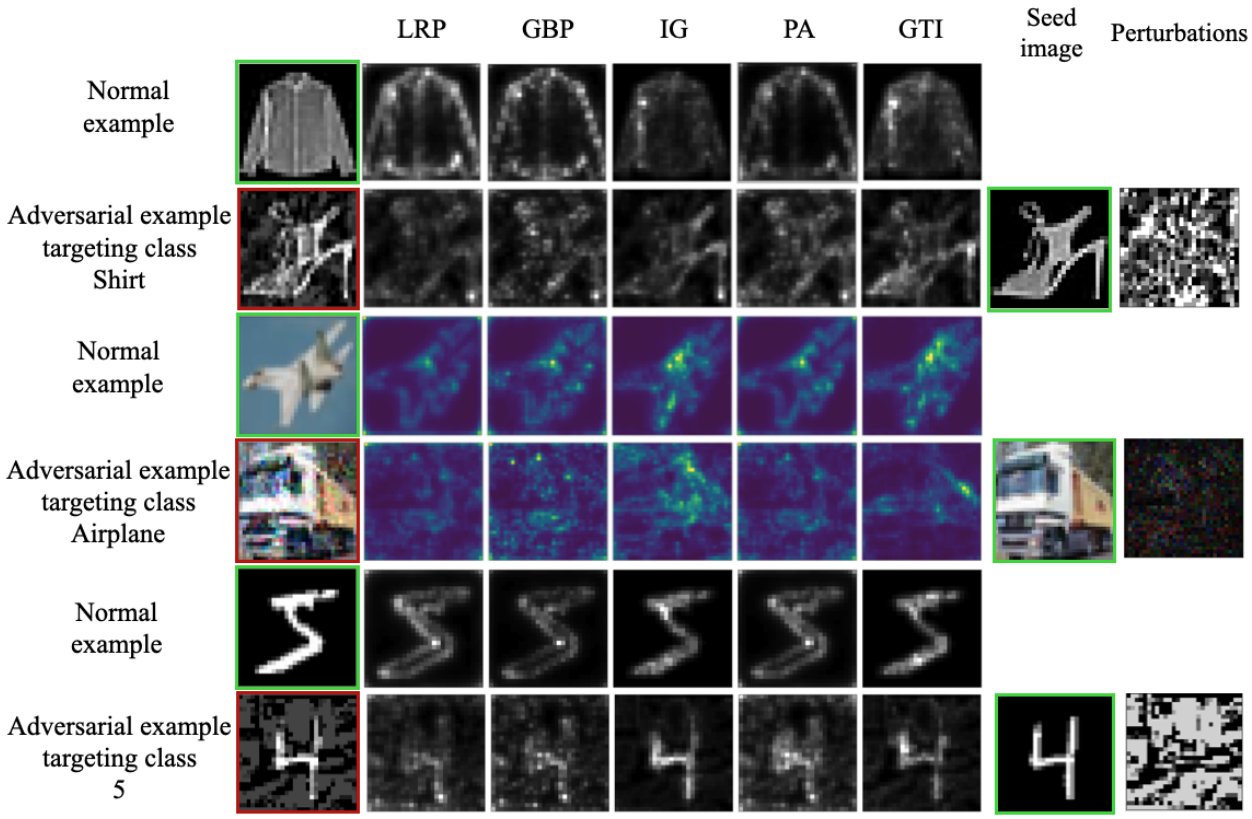

Figure 3.2: Distinguishability between normal and abnormal explanations.

approach (where we use both normal and abnormal explanations) and an anomaly detection approach (where we only use normal explanations). Whereas the former setting makes the training and validation process simple (once we have the required data), the motivation for the latter setting is to make our defense attack-independent so that it is more likely to generalize to unknown attacks. At test time, if the explanation map of an input is classified as abnormal by any detector model of the classified class, then we consider it to be an adversarial example.

3.3 Generation of explanations

Given a neural network classifier $f(\cdot)$ and an input x, the explanation of the classification of x is represented as an explanation map denoted by $h : \mathbb{R}^d \rightarrow \mathbb{R}^d$. The explanation map $h(x)$ encodes the relevance score of every pixel in x for the neural network's prediction. We consider the following explanation generation techniques towards building an ensemble of methods.

- **Gradient**: The gradient of the output $f(x)$ with respect to the input x is indicative of how

infinitesimal changes in each pixel can influence the output [21, 22]. The explanation map using the gradient method is given by

$$h(x) = \frac{\partial f}{\partial x}(x)$$

- **Gradient $*$ Input (GTI)**: This method computes an element-wise product between the gradient-based explanation map of Simonyan et al. [21] and the input to quantify the influence of each pixel on the prediction score [23]. Formally, the explanation map produced by gradient $*$ input is given by

$$h(x) = x \odot \frac{\partial f}{\partial x}(x)$$

- **Integrated Gradients (IG)**: In contrast to GTI, which performs a single computation of the gradient at the input x, integrated gradients computes the gradients at all points along a linear path from a baseline $\bar{x}$ to x, and averages them [24]. The baseline $\bar{x}$ can be defined by the user and is generally chosen as a black image. Formally,

$$h(x) = (x - \bar{x}) \odot \int_{\alpha=0}^{1} \frac{\partial f(\bar{x} + \alpha(x - \bar{x}))}{\partial x} \mathrm{d}\alpha$$

- **Guided Backpropagation (GBP)**: This method is an extension of gradient-based explanation with the key difference that it prevents backward flow of negative gradients through non-linearities, such as ReLUs [25].

- **Layer-wise Relevance propagation (LRP)**: To explain the prediction of class c, LRP [26, 27] starts with the output neuron of class c and goes backwards through the network by following the z^+ rule for all layers except the first.

$$R_i^l = \sum_j \frac{x_i^l (W^l)_{ji}^+}{\sum_i x_i^l (W^l)_{ji}^+} R_j^{l+1}$$

Here, i and j are two neurons of consecutive layers, R_i^l denotes the relevance of i-th neuron

in the l-th layer, x_i^l represents the activation vector, and $(W^l)_{ji}^+$ denotes the positive weight between the two neurons. Then, to account for the bounded range of an input, we use the $z^{\mathcal{B}}$ rule in the first layer

$$R_i^0 = \sum_j \frac{x_j^0 W_{ji}^0 - l_j (W^0)_{ji}^+ - h_j (W^0)_{ji}^-}{\sum_i (x_j^0 W_{ji}^0 - l_j (W^0)_{ji}^+ - h_j (W^0)_{ji}^-)} R_j^1$$

where l and h are the lowest and highest allowed pixel values, respectively.

- **Pattern Attribution (PA)**: Kindermans et al. [28] proposed patter attribution as an improvement over the LRP framework. The method is analogous to the backpropagation operation with the weights in the backward pass replaced by element-wise multiplication of weights W^l and learned patterns A^l.

While we considered all six of the above-mentioned explanation techniques to include in ExAD, we found the performance of the gradient method ($h(x) = \frac{\partial f}{\partial x}(x)$) to be unacceptable based on evaluations on the validation sets, whereas remaining techniques performed significantly better. Therefore, in this work, ExAD uses an ensemble of $k = 5$ techniques- LRP, GBP, IG, PA, and GTI.

Figure 3.2 shows examples of normal and abnormal explanations produced by different techniques used in ExAD. When a test input x_t is classified by the target model $f(\cdot)$ as class c, each of the k techniques produce an explanation map for this classification. In column 1, the first, third, and fifth rows show a normal example from FMNIST, CIFAR-10, and MNIST datasets, respectively. In the same column, the second, fourth, and sixth rows show an adversarial example which is misclassified as the class represented by the normal example in the preceding row. Columns 2-6 show the corresponding explanation maps produced by the five explanation techniques. The distinguishability between explanation maps of normal and adversarial examples allows the detector models to determine if an explanation map is normal or not. In the following section, we discuss how a set of k detector models, one corresponding to every explanation technique, evaluate the explanation maps towards determining if x_t is an adversarial example.

3.4 Detector models

The detector model determines if the explanation map of an input is normal or abnormal for the classified class. We study the following two methods for building detector models.

3.4.1 Detection using a CNN-based binary classifier

First, we consider an approach of building a CNN-based binary classifier. We term this as the *CNN-based detector model* and denote it as $g(\cdot)$. Under this setting, we refer to the defense as ExAD-CNN. Below, we describe the training procedure for these detector models.

For every class c, we build a separate detector model $g_{c,j}(\cdot)$ for each explanation technique h_j. At test-time, for an input classified as class c, the detector model $g_{c,j}(\cdot)$ takes the explanation map of the input, produced by the corresponding explanation technique h_j, and classifies it as normal or abnormal. We build this new model $g_{c,j}(\cdot)$ as follows. We take every normal example x_{normal} from class c, which is correctly classified by the target model $f(\cdot)$, and generate the explanation map $h_j(x_{\text{normal}})$ for its classification into that class. These explanation maps are considered as normal explanations, and are labeled as negative *class 0*. Then, we generate a number of adversarial examples using different adversarial attacks where the targeted class is c. Next, for each successful adversarial example x_{adv}, we generate the explanation map $h_j(x_{\text{adv}})$ for its classification as target class c. These explanation maps are considered as abnormal explanations, and labeled as positive *class 1*. Then, we train $g_{c,j}(\cdot)$ on this labeled training set using a CNN-based architecture.

3.4.2 Detection based on reconstruction error

In this approach, we avoid the requirement of adversarial examples to train a detector model, and thereby make the defense more likely to generalize on unknown attacks. Here, we propose using reconstruction error by an autoencoder to determine if the explanation map of a test example is normal or not. We term this as the *autoencoder-based detector model*. Under this setting, we refer to the defense as ExAD-AE. Similar to ExAD-CNN setting, we consider each class and build k autoencoder-based detector models, one corresponding to every explanation technique.

An autoencoder $ae = \psi \circ \phi$ contains two components, an encoder and a decoder, which can be

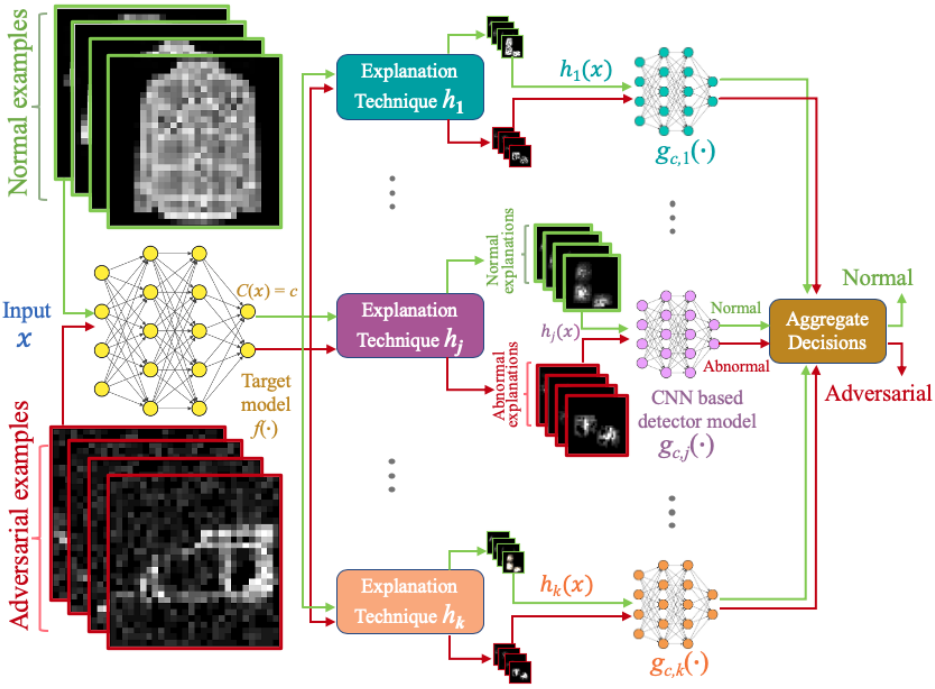

Figure 3.3: Illustration of the proposed ExAD framework.

defined as transitions $\phi : \mathbb{R}^d \to \mathbb{R}^z$ and $\psi : \mathbb{R}^z \to \mathbb{R}^d$, respectively, where $\mathbb{R}^d$ is the input space and $\mathbb{R}^z$ is the latent space. For class c and the j-th explanation technique, the input space for the autoencoder $ae_{c,j}$ in our system is formed by the set of explanations produced by h_j for correctly classified normal examples of class c. We train the autoencoder to minimize a loss function over this set of explanations, where the loss function is taken as the mean squared error (MSE):

$$L(\mathbb{E}_{\text{train}}) = \frac{1}{\mathbb{E}_{\text{train}}} \sum_{h_j(x) \in \mathbb{E}_{\text{train}}} \|h_j(x) - (\psi \circ \phi) h_j(x)\|_2$$

For a test image, the explanation map $h_j(x)$ produced by the j-th technique is given as input to the autoencoder $ae_{c,j}$ which generates a reconstructed image. Then, we compute a reconstruction error:

$$R(h_j(x)) = \|h_j(x) - (\psi \circ \phi) h_j(x)\|_p \tag{3.1}$$

where $\|\cdot\|_p$ is a suitable p-norm. If the reconstruction error is above a threshold t_{re}, we consider the explanation map $h_j(x)$ to be abnormal. The threshold value is a hyperparameter for each detector

model. It should be low enough to detect abnormal explanations, but sufficiently high to not falsely flag normal explanations. We decide t_{re} values using a validation set of normal explanations, which are in turn derived from a validation set of normal examples. For any detector, we select the highest t_{re} such that its false-positive rate on the validation set is below a threshold t_{fp}. The threshold t_{fp} can be chosen depending upon system requirements.

3.5 Test-time detection of adversarial examples

Figure 3.3 illustrates our overall approach, with the ExAD-CNN setting. At test-time, if an unknown input x is being classified by the target classifier $f(\cdot)$ as class c, our goal is to identify if x is a normal example of class c or an adversarial example. To this end, we take the following steps. First, we generate k explanation maps for the classifier's decision to classify x as class c using k explanation techniques. Second, for each explanation map $h_j(x)$, we use the corresponding detector model to determine if the explanation map is normal or abnormal. For ExAD-CNN, each detector model directly provides a classification of normal ($g_{c,j}(h_j(x))$=0) or abnormal ($g_{c,j}(h_j(x))$=1). On the other hand, for ExAD-AE, we obtain the reconstruction error for each explanation map $h_j(x)$ using the corresponding autoencoder $ae_{c,j}$. If the reconstruction error computed by a detector model is above its threshold, then it considers the explanation map to be abnormal. Finally, in both settings, we aggregate the results from all the detector models. In our current implementation, we use a simple (and strict) strategy of inferring that an input x is an adversarial example if any of the k detector models classifies the respective explanation map as abnormal. However, this setting can be configured based on the needs of the application, such as the false-positive and false-negative rates that can be tolerated. In future work, we will explore more approaches to aggregate these results, such as using majority vote or another machine learning model to produce the final outcome more effectively.

4. EXPERIMENTS

In this section, we evaluate the effectiveness of ExAD. This section is organized as follows. First, we provide details on the experiment settings in Section 4.1. In Section 4.2, we report the performance of the system on normal examples. Then, in Section 4.3, we show the performance of ExAD on blackbox attacks. Next, we investigate the generalizability of ExAD-CNN in Section 4.4. We compare the performance of our approach with three state-of-the-art detection methods in Section 4.5. Finally, in Section 4.6, we present our evaluation on whitebox attacks.

4.1 Experimental settings

4.1.1 Environment

We implement the proposed framework using the Python libraries Keras and TensorFlow. We conducted our experiments on a Linux server with one GPU (GeForce RTX 2080 Ti) and CPU (Intel Xeon Silver 4116 processor).

4.1.2 Image Datasets

We evaluated the performance of our detection mechanism on three image datasets: MNIST [34], Fashion-MNIST (FMNIST) [35] and CIFAR-10 [36]. MNIST is a well-known gray-scale image dataset of handwritten digits from 0 to 9. FMNIST is a relatively more challenging dataset of article images where each example is a 28x28 grayscale image associated with a label from 10 classes (shirts, sandals, etc.). Both datasets consist of 60000 examples in the training set and 10000 examples in the testing set. CIFAR-10 is a colored image dataset of tiny 32x32x3 images used for object recognition. It comprises of 50000 training images and 10000 testing images. We chose MNIST and CIFAR-10 datasets as they are most widely used for evaluating defenses against adversarial attacks [20, 1, 2, 19], and additionally used FMNIST as it provides more challenges for a gray-scale dataset.

Table 4.1: Evaluation of blackbox attacks.

	Attack		Parameter	Cost (s)	Success Rate	Prediction Confidence	L_2 Distortion
MNIST	L_∞	CW_∞	-	90.57	100%	39.11%	3.47
		BIM	eps:0.3	0.003	99%	99.94%	4.10
		MIM	eps:0.3	0.003	100%	99.99%	5.98
	L_2	CW_2	confidence:0	0.001	100%	97.11%	4.33
	L_0	CW_0	-	8.98	100%	38.18%	5.47
		JSMA	gamma:0.2	0.84	94%	76.86%	7.00
FMNIST	L_∞	CW_∞	-	90.07	100%	38.51%	0.729
		BIM	eps:0.3	0.004	98%	100%	4.06
		MIM	eps:0.3	0.004	100%	100%	5.87
	L_2	CW_2	confidence:0	0.006	100%	96.09%	2.20
	L_0	CW_0	-	8.95	100%	37.80%	2.88
		JSMA	gamma:0.2	0.96	85%	83.20%	4.53
CIFAR-10	L_∞	CW_∞	-	68.90	100%	26.91%	1.19
		BIM	eps:0.3	0.008	100%	100%	6.1
		MIM	eps:0.3	0.001	100%	100%	7.9
	L_2	CW_2	confidence:0	0.005	100%	94.52%	3.86
	L_0	CW_0	-	13.35	100%	27.09%	2.98
		JSMA	gamma:0.2	6.42	100%	43.35%	1.78

4.1.3 Training the Target Models

On MNIST and FMNIST datasets, we trained a CNN based target model with 54000 examples in the training set and 6000 examples in the validation set. For CIFAR-10, we trained the CNN based target model with 44000 examples in the training set and 6000 examples in the validation set. Table 4.2 shows the CNN architectures, and Table 4.3 shows the hyperparameters for training the three target models.

4.1.4 Generating Adversarial Examples

As described in Section 2.3, we generate adversarial examples using six state-of-the-art attacks- JSMA [8], BIM [40], MIM [41], and CW_0, CW_2, and CW_∞ variants of the CW attack [12]. For JSMA, BIM, MIM, and CW_2 attacks, we created adversarial samples using their implementations

Table 4.2: Architecture of the image classifiers to be defended.

MNIST		FMNIST		CIFAR-10	
Conv.ReLU	8x8x64	Conv.ReLU	3x3x32	Conv.ReLU	3x3x64
Conv.ReLU	6x6x128	Conv.ReLU	3x3x32	Conv.ReLU	3x3x128
Conv.ReLU	5x5x128	MaxPooling	2x2	AvgPooling	2x2
Softmax	10	Conv.ReLU	3x3x64	Conv.ReLU	3x3x128
		Conv.ReLU	3x3x64	Conv.ReLU	3x3x256
		MaxPooling	2x2	AvgPooling	2x2
		Dense.ReLU	200	Conv.ReLU	3x3x256
		Dense.ReLU	200	Conv.ReLU	3x3x512
		Softmax	10	AvgPooling	2x2
				Conv.ReLU	3x3x10
				Softmax	10

Table 4.3: Training and architecture hyperparameters of the target classifiers.

Hyperparameter	MNIST	FMNIST	CIFAR-10
Learning Rate	0.001	0.01	0.001
Optimization Method	Adam	SGD	Adam
Batch Size	128	128	256
Epochs	50	50	50
Padding (Conv layers)	Valid	Valid	Same

in the Cleverhans library [49]. For CW_0 and CW_∞ attacks, we use the implementation from the authors [12, 50]. For our evaluation, we adopt the *target-next* attack setting in which the targeted label is the class next to the ground truth class modulo the number of classes (e.g., misclassify an input of class 4 to class 5). Table 4.1 shows a summary of our evaluation of the six blackbox attacks.

We generate adversarial examples for two purposes. First, as discussed in section 3.4, we need adversarial examples to derive abnormal explanations for training and validating ExAD-CNN. To this end, we consider each class in a dataset and generate as many adversarial examples as the number of normal examples of that class in the training and validation sets. These adversarial examples are unevenly distributed by attack methods, due to relatively higher cost involved in con-

ducting certain attacks. Column 5 in Table 4.1 shows the average cost (in seconds) to generate one adversarial example for different attacks. We observe that the CW_∞, CW_0, and JSMA attacks incur much more overhead than remaining three attacks. Therefore, we generate 80% of the examples using BIM, MIM, and CW_2 attacks, and the remaining using CW_∞, CW_0, and JSMA attacks. We empirically found this distribution to be sufficient, based on performance on the validation sets. Furthermore, seed images for these adversarial examples are randomly selected from normal examples of the source class. Note that we only utilize examples from the training (resp., validation) set towards training (resp., validating) ExAD; any example in the test set is considered non-accessible for this purpose, as is standard practice.

The second purpose is to evaluate the detection rate of ExAD. To this end, for every dataset, we generate 100 adversarial examples using each attack. This number is consistent with previous works [20, 2] and is limited since many attacks are too expensive to execute. In this process, we create the same number of adversarial examples for every target class to ensure a balanced evaluation. Furthermore, seed images for adversarial examples are taken from correctly classified examples in the test set so that the normal counterparts are unseen by the target model, and the resulting abnormal explanations are unseen by ExAD-CNN.

In Table 4.1, column 6 shows the success rate achieved by the six blackbox attacks when ExAD is not included as a defense. We consider an attack to be successful if the target model predicts the targeted class. The resulting examples from such attacks are termed as *successful adversarial examples*. We observe that most attacks are very effective against three target models. The BIM, MIM, and CW_2 attacks are particularly effective in generating high-confidence adversarial examples as shown in column 7.

4.1.5 Training ExAD-CNN and ExAD-AE

Table 4.4 and Table 4.5 show the CNN architectures and hyperparameters used for training the CNN-based and autoencoder-based detector models, respectively. For each setting, we use the same architecture and hyperparameters for all three datasets as the performance on the respective validation sets was found acceptable. As discussed in Section 3.4, for each target class, we train

Table 4.4: Architecture and hyperparameters of ExAD-CNN.

Architecture		Hyperparameters	
Conv.ReLU	3x3x32	Learning Rate	0.01
Conv.ReLU	3x3x64	Optimization Method	Adam
MaxPooling	2x2	Batch Size	32
Conv.ReLU	3x3x128	Epochs	50
Conv.ReLU	3x3x128	Padding (Conv layers)	Same
MaxPooling	2x2		
Dense.ReLU	512		
Dense.ReLU	64		
Softmax	2		

Table 4.5: Architecture and hyperparameters of ExAD-AE.

Architecture		Hyperparameters	
Dense.ReLU	HxW	Learning Rate	10^{-5}
Dense.ReLU	400	Optimization Method	Adam
Dense.ReLU	20	Batch Size	32
Dense.ReLU	400	Epochs	100
Dense.ReLU	HxW		
Softmax	2		

a detector model for every explanation technique. For training and validation, while ExAD-CNN uses both normal and abnormal explanations, ExAD-AE only uses normal explanations. To obtain normal explanations for a class, we take all its normal examples in our training and validation sets and generate corresponding explanations. For abnormal explanations (to be used by ExAD-CNN), we generate explanations of the adversarial examples being classified as the target class using each explanation technique. As discussed earlier, for this purpose, we had generated as many adversarial examples as the number of normal examples of each class in the training and validation sets. This provides us with balanced training and validation sets of normal and abnormal explanations. For ExAD-CNN, we label the normal and abnormal explanations as negative and positive class, respectively. We train the detector models on the training set, and use the validation set for tuning

the hyperparameters. For ExAD-AE, we exclude the abnormal explanations in both training and validation sets (so that they online consist of normal explanations). Then, we train the detector models on the training set, and use the validation set for setting the threshold t_{re} values. Also, for computing the reconstruction error (equation 3.1), we empirically find it sufficient to use the L_2 norm. Furthermore, we selected the threshold t_{re} such that the false-positive rate for any detector model is at most 0.2% on its validation set.

4.1.6 Comparison

We compare ExAD with three state-of-the-arts- MagNet [1], Feature Squeezing (FS) [2], and LID [3]. For their implementation, we use the respective GitHub repositories. We follow instructions in the repositories and papers to identify optimal configurations. Feature Squeezing, in particular, allows many configurations for its squeezers. Consistent with the author's work, we utilize the optimal join-detection setting with multiple squeezers. For MNIST and FMNIST, we use the combination of a 1-bit depth squeezer with 2x2 median smoothing. For CIFAR-10, we use a 5-bit depth squeezer with 2x2 median smoothing and 13-3-2 non-local means filter. To ensure fair comparison, for all detection methods, we set thresholds such that the false-positive rate on the validation set is at most 0.2% (same as ExAD). Additionally, our comparison could not include NIC [20] as it is yet to be made open-source, and we were not successful in reproducing the system.

4.2 Performance on normal examples

Table 4.7 shows the classification accuracy of the three target models on their test set (of normal examples). Without ExAD, we achieve an accuracy of 99.15%, 90.68%, and 84.54% for MNIST, FMNIST, and CIFAR-10 datasets, respectively. When ExAD is included as a defense, it is possible that a correctly classified normal example (by the target model) is misclassified as an adversarial example, termed as a *false-positive* (FP). With ExAD-CNN, we obtained a false-positive rate of 0.90% on MNIST dataset, as 89 of 9915 correctly classified normal examples are classified as adversarial. Thus, with ExAD-CNN, the accuracy of the target system is reduced only slightly to

Table 4.6: Detection rate of ExAD on blackbox attacks.

Dataset	Attack		Parameter	No Defense	**ExAD-CNN**	**ExAD-AE**	MagNet [1]	FS [2]	LID [3]
MNIST	L_∞	CW_∞	-	0%	100%	100%	96%	100%	92%
		BIM	eps:0.3	1%	100%	100%	100%	97.98%	97.98%
		MIM	eps:0.3	0%	100%	100%	100%	98%	99%
	L_2	CW_2	confidence:0	0%	100%	100%	86%	100%	91%
	L_0	CW_0	-	0%	100%	100%	86%	91%	91%
		JSMA	gamma:0.2	6%	100%	100%	84.04%	100%	93.62%
FMNIST	L_∞	CW_∞	-	0%	100%	100%	97%	100%	94%
		BIM	eps:0.3	2%	100%	100%	100%	97.96%	93.88%
		MIM	eps:0.3	0%	100%	100%	99%	99%	95%
	L_2	CW_2	confidence:0	0%	100%	100%	85%	100%	92%
	L_0	CW_0	-	0%	100%	100%	85%	90%	91%
		JSMA	gamma:0.2	15%	100%	100%	87.06%	100%	92.94%
CIFAR-10	L_∞	CW_∞	-	0%	99%	100%	84%	98%	90%
		BIM	eps:0.3	0%	100%	100%	100%	52%	97%
		MIM	eps:0.3	0%	100%	100%	100%	51%	97%
	L_2	CW_2	confidence:0	0%	100%	100%	92%	100%	89%
	L_0	CW_0	-	0%	98%	100%	76%	98%	91%
		JSMA	gamma:0.2	0%	100%	99%	95%	83%	92%

98.26%. Similarly, on FMNIST and CIFAR-10 datasets, the accuracy has a minor drop to 89.70% and 83.74%, respectively. The low false-positive rates are indicative of explanation maps of normal examples rarely being mistaken as abnormal by the detector models. This allows us to maintain a strict policy of classifying a test input as adversarial if even a single detector model considers its explanation map as abnormal.

With ExAD-AE, we obtain false-positive rates of 0.62%, 0.85%, and 0.82% on the test sets of MNIST, FMNIST, and CIFAR-10 datasets, respectively. The accuracy of the target systems under this setting is 98.54% for MNIST, 89.91% for FMNIST, and 83.85% for CIFAR-10 dataset. These results are close to the performance of ExAD-CNN on normal examples. In the following section, we show that both settings of ExAD can effectively detect adversarial attacks while maintaining these low false-positive rates.

Table 4.7: Classification accuracy on normal examples with and without defense.

Dataset	Accuracy without ExAD	Top-1 Mean Confidence	Accuracy with ExAD (CNN)	FP rate with ExAD (CNN)	Accuracy with ExAD (AE)	FP rate with ExAD (AE)
MNIST	99.15%	99.86%	98.26%	0.90%	98.54%	0.62%
FMNIST	90.68%	97.86%	89.70%	1.08%	89.91%	0.85%
CIFAR-10	84.54%	76.64%	83.74%	0.95%	83.85%	0.82%

4.3 Evaluation on blackbox attacks

Table 4.6 shows the detection rates of our approach on the six blackbox attacks. Columns 1-4 show datasets and details of the attacks. Column 5 shows the detection rate of adversarial examples when ExAD is not included as a defense (which corresponds to the success rate achieved by blackbox attacks on the target models). Columns 6 and 7 show the detection rates of ExAD-CNN and ExAD-AE, respectively. Note that, except when noted explicitly, “detection rate" of a detection method refers to its detection rate on successful adversarial examples, consistent with previous work [2]. The remaining columns report our comparison with three state-of-the-art detectors, which we will discuss in Section 4.5.

We first consider the ExAD-CNN setting. For this setting, our approach achieves consistently high detection rates for all attacks across the three datasets. As shown in Table 4.6, for MNIST and FMNIST datasets, we get 100% detection rates for all six attacks. For CIFAR-10 dataset, we obtain 98% detection rate for CW_2 and CW_0 attacks, and 100% detection rate for remaining attacks. For the ExAD-AE setting, the detection rate of adversarial examples is again consistently high. We achieve 100% detection rate for all six attacks on MNIST and FMNIST datasets. On CIFAR-10 dataset, we obtain a 99% detection rate for JSMA attack. For all other attacks on CIFAR-10 dataset, the detection rate is 100%.

While we find both settings of ExAD are effective against adversarial attacks, each has its own relative advantages and disadvantages. A benefit of ExAD-AE is that it does not rely on adversarial examples for training or validation. This reduces the training overhead as many adversarial attacks

Table 4.8: Detection rate of ExAD-CNN with limited attacks used in training.

Attack	Train on BIM, MIM, CW_2, JSMA			Train only on CW_2		
	MNIST	FMNIST	CIFAR-10	MNIST	FMNIST	CIFAR-10
CW_∞	100%	100%	92%	99%	99%	96%
BIM	100%	100%	100%	97.98 %	96.94%	100%
MIM	100%	100%	100%	89%	86%	100%
CW_2	100%	100%	100%	100%	100%	100%
CW_0	100%	100%	92%	100%	97%	91%
JSMA	100%	100%	100%	91.49%	94.12%	87%

Table 4.9: False-positive rates obtained for MagNet [1], FS [2], and LID [3].

Dataset	MagNet [1]	FS [2]	LID [3]
MNIST	0.50%	3.85%	4.24%
FMNIST	0.81%	3.76%	3.89%
CIFAR-10	4.25%	4.81%	5.36%

incur significant cost (Table 4.1). More importantly, being attack-independent, the performance of ExAD-AE indicates that our approach generalizes well to unknown attacks. But, compared to the effort required in setting the threshold t_{re} values for ExAD-AE, it is relatively simpler to tune the hyperparameters for ExAD-CNN. However, training the detector models in ExAD-CNN requires adversarial examples (to derive abnormal explanations). In the following section, we investigate the extent to which this requirement impacts the generalizability of ExAD-CNN in detecting unknown attacks.

4.4 Generalizability of ExAD-CNN

In the generation of abnormal explanations, we notice that adversarial examples created using attacks of the same category (L_∞ or L_0) produce similar explanations. We show an example of this similarity in Figure 4.1. Rows 1 and 2 show explanation maps for adversarial examples which were created using CW_∞ and BIM attacks, respectively. Both attacks come under the L_∞ category. The adversarial examples are targeting the Pullover class and their seed images are taken from the Trouser class of FMNIST dataset. Comparing explanation maps in row 1 and row 2 shows that

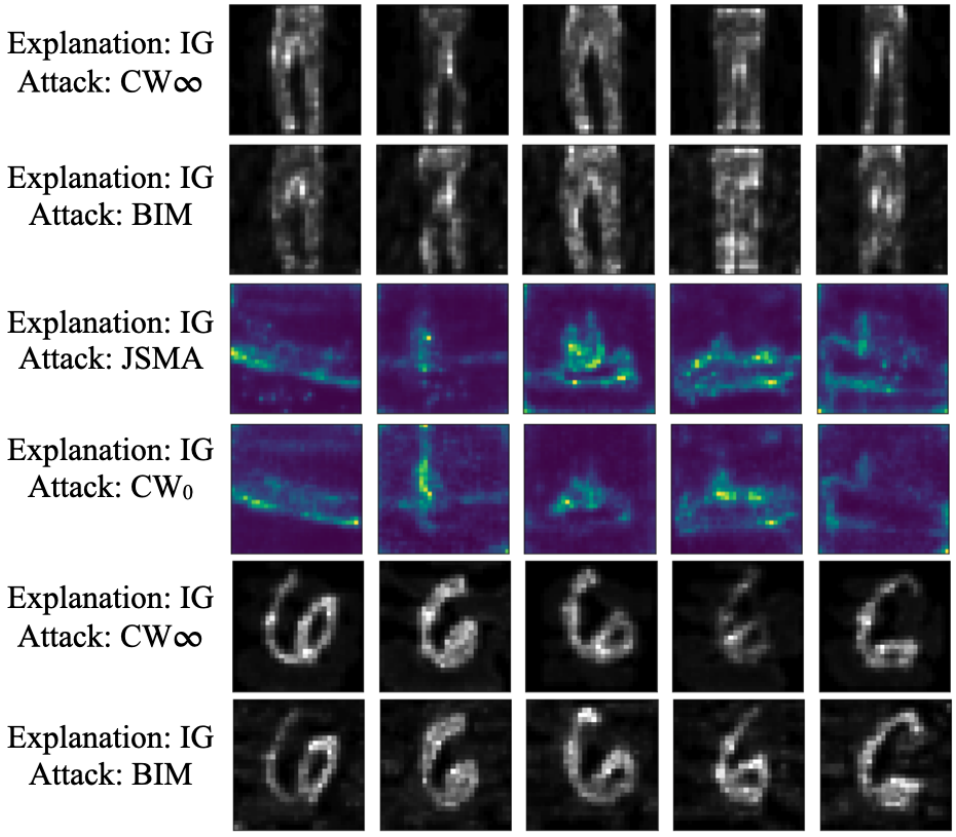

Figure 4.1: Similarity in explanation maps across adversarial attacks.

adversarial examples created using different attacks, under the same category (L_∞ in this case) can result in similar explanation maps. This can also be observed for CIFAR-10 dataset (rows 3-4), where we use two L_0 attacks, and MNIST dataset (rows 5-6), where we again use two L_∞ attacks.

Using this observation, we leave out the CW_∞ and CW_0 attacks, and only use adversarial examples from other four attacks for training ExAD-CNN. We keep other training and attack parameters same as before. In Table 4.8, columns 2-4 show the new performance of ExAD-CNN. On MNIST and FMNIST datasets, we find that ExAD-CNN effectively detects the unknown attacks. On CIFAR-10, ExAD-CNN still achieves a good detection rate of 92% for both unknown attacks, CW_∞ and CW_0. We also observe that the detection of other attacks is not affected on any dataset. These results appear to indicate that abnormal explanations are influenced more by the original class of the seed images and the category (norm) of the attack, rather than the attack variant in

that category used to craft the adversarial examples. This allows ExAD-CNN to generalize well to unknown attacks when we train on representative attacks from different categories. Furthermore, to study the generalizability in a more restrictive case, we only train ExAD-CNN on CW_2 attack. Columns 5-7 in Table 4.8 show the performance under this scenario. For all datasets, we find the results are good on the three CW attacks, with the lowest detection rate of 91% obtained for CW_0 attack on CIFAR-10 dataset. For BIM attack, the performance remains consistently high across datasets. However, the detection is less effective for MIM attack on MNIST and FMNIST datasets, for which we have 89% and 86% detection rates, respectively, and for JSMA attack on CIFAR-10, for which we obtain 87% detection rate. Overall, we see that ExAD-CNN can detect many unknown attacks even in this restrictive case, but there is still room for improvement. For boosting the detection in such scenarios, we can consider performing join-detection using both ExAD-CNN and ExAD-AE (which obtained high detection rates while being attack-independent). Building such joint-detectors to improve generalizability will be an interesting topic for future work.

4.5 Comparison

Table 4.6 shows the comparison of ExAD with three state-of-the-art adversarial detection methods- MagNet [1], Feature Squeezing [2], and LID [3]. The false-positive rates for these methods on the test sets are reported in Table 4.9.

MagNet. We find that MagNet's false-positive rates on the two grayscale datasets are marginally lower than those of ExAD-AE. However, it has much higher false-positive rate of 4.25% on CIFAR-10 dataset. We also find its detection performance to vary depending upon the dataset and attack-norm. MagNet uses trained autoencoders to detect adversarial examples, and to reform them based on the differences between the manifolds of normal and adversarial examples. MagNet's denoising strategy is quite effective against L_∞ attacks, for which adversarial examples tend to have a large number of modified pixels, with a limit on the change per pixel. MagNet achieves high detection rates (most being 100%) for L_∞ attacks on both grayscale datasets. On CIFAR-10, a colored dataset, it achieves similar performance on BIM and MIM attacks, but relatively low detection rate of 84% on CW_∞ attack. Furthermore, we observe that MagNet's denoising mech-

anism is not as effective on L_0 attacks. These attacks make changes of high magnitude to very few pixels, thereby making denoising difficult. This is consistent with findings by Ma et al. [20]. Similarly, we find MagNet's performance on CW_2 attack is not as good on MNIST and FMNIST datasets. Moreover, MagNet requires training a single detector network, which is computationally expensive. A benefit of our approach is that we use small detector models for every class, which are much easier to train. This makes our approach more practical.

LID. We find the detection performance of LID to be consistent across attacks and datasets. This method computes an LID value that captures the intrinsic dimensional properties of adversarial regions[3]. LID achieves its highest detection for L_∞ attacks on the three datasets. For most of the other attacks, its detection was consistently above 90%. However, as shown in Table 4.9, a downside of using LID values is the relatively higher false-positive rates on normal examples, which impacts the reliability of its classifications.

Feature Squeezing. For Feature Squeezing, we observe that joint-detection provides fairly consistent detection rates (Table 4.6), but introduces high false-positive rates between 3.76% to 4.81% (Table 4.9). This is natural, given the use of a single threshold across all squeezers, consistent with original work [2]. Nevertheless, as reported by the authors, this can be improved by combining multiple squeezers with different thresholds in future work. Feature Squeezing obtains very high detection rates on all attacks on MNIST and FMNIST datasets. On CW_2 attack, it achieves 100% detection rate on all three datasets. However, for L_∞ attacks, while it obtains detection rates of above 98% on CW_∞ attack, we find it less effective against BIM and MIM attacks on CIFAR-10 dataset with a detection rate of nearly 52%. This reflects upon the generalizability challenges in building squeezers. In contrast, our approach is more general and achieves consistent detection and false-positive rates.

4.6 Evaluation with adaptive adversaries

In this section, we evaluate our defense, with the ExAD-CNN setting, under whitebox threat model for an adaptive adversary. Here, we build upon recent research that shows that explanations can be unreliable [29] and can be manipulated to produce a target explanation map [30, 31]. Below,

Table 4.10: Evaluation of whitebox attacks.

	Target Explanation Method	Mean Success Rate without defense	Cost (s)	L_2 Distortion
MNIST	LRP	99.00%	137.07	2.49
	GBP	95.17%	50.15	2.64
	IG	82.00%	506.75	2.81
	PA	96.17%	57.74	2.55
	GTI	91.50%	48.96	2.50
FMNIST	LRP	99.17%	169.36	2.28
	GBP	95.33%	51.03	2.34
	IG	89.00%	510.47	2.33
	PA	93.67%	57.77	2.76
	GTI	91.50%	49.12	2.31
CIFAR-10	LRP	99.00%	138.83	2.26
	GBP	97.33%	51.35	2.24
	IG	94.00%	484.02	3.87
	PA	96.00%	66.53	3.20
	GTI	95.33%	66.80	3.45

we present our approach to conduct a whitebox attack for generating an adversarial example.

4.6.1 Whitebox Attack Approach

Given a normal or seed image $x \in \mathbb{R}^d$ with correctly classified class $C(x) = c$ by target model $f(\cdot)$, we follow a two-step process towards conducting a whitebox attack. First, we use a blackbox attack to generate an adversarial example x' which is misclassified as $C(x') = t$, where t is the targeted class. While x' is likely to fool $f(\cdot)$, it is likely to be correctly classified as an adversarial example by the defense, which has not yet been accounted for by the attack. As a next step, we consider a target explanation technique h_j, which produces an explanation map $h_j(x')$ that is correctly classified as abnormal by the corresponding detector model $g_{t,j}(\cdot)$. Our goal in this step is to manipulate x' to create a final adversarial example $x'' = x' + \delta x'$, such that

- The target model's classification remains approximately constant, i.e. $f(x'') \approx f(x')$
- The explanation map $h_j(x'')$ is close to a target explanation map h_j^t that is classified as normal by $g_{t,j}(\cdot)$

- The norm of the perturbation $\delta x'$ added is small so that it remains imperceptible.

To obtain the target explanation map h_j^t, we randomly select a normal example x_r from class t and check if its explanation map $h_j(x_r)$ is classified as normal by $g_{t,j}(\cdot)$. If so, we set the target explanation map as $h_j^t = h_j(x_r)$. Else, we repeat the process until we find such an example. This search is fairly quick because explanation maps of normal examples of a class are very likely to be correctly classified as normal by the detector models. Finally, we generate x'' by optimizing the following loss function

$$\mathcal{L} = ||h_j(x'') - h_j^t||^2 + \gamma||f(x'') - f(x')||^2 \tag{4.1}$$

with respect to x'' using gradient descent, such that $\|\delta x'\|_2 < \epsilon$. The first term in the loss function ensures that the explanation map for x'' is close to the target map, while the second term ensures the prediction by the target model is still the misclassified class t. The weighting of these two terms is controlled by hyperparameter $\gamma \in \mathbb{R}_+$. To compute the gradient with respect to the input $\nabla h_j(x'')$, we follow the strategy by Dombrowski et al. of replacing relu with the softplus function to circumvent the problem of vanishing second-derivative [31]. A similar strategy of approximating relu was used by Zhang et al. [30]. After the optimization completes, we test whether the manipulated image x'' fools the original relu-based target model $f(\cdot)$ as well as our defense.

4.6.2 Illustration of whitebox attack

Figure 4.2 shows an illustration of our whitebox attack approach, discussed in Section 4.6. The leftmost image in the first row shows a seed image x, which is a normal example from class Airplane of CIFAR-10 dataset. In the first step, we use CW_∞ attack to add perturbations δx to x, which results in the adversarial example $x' = x + \delta x$. The rightmost image of row 1 shows x'. The example x' is misclassified as class Automobile by target model $f(\cdot)$. The second row shows the explanation maps produced for x' by the five techniques. We find that all detector models classify these explanation maps as abnormal. We observe in row 2 that the explanation maps

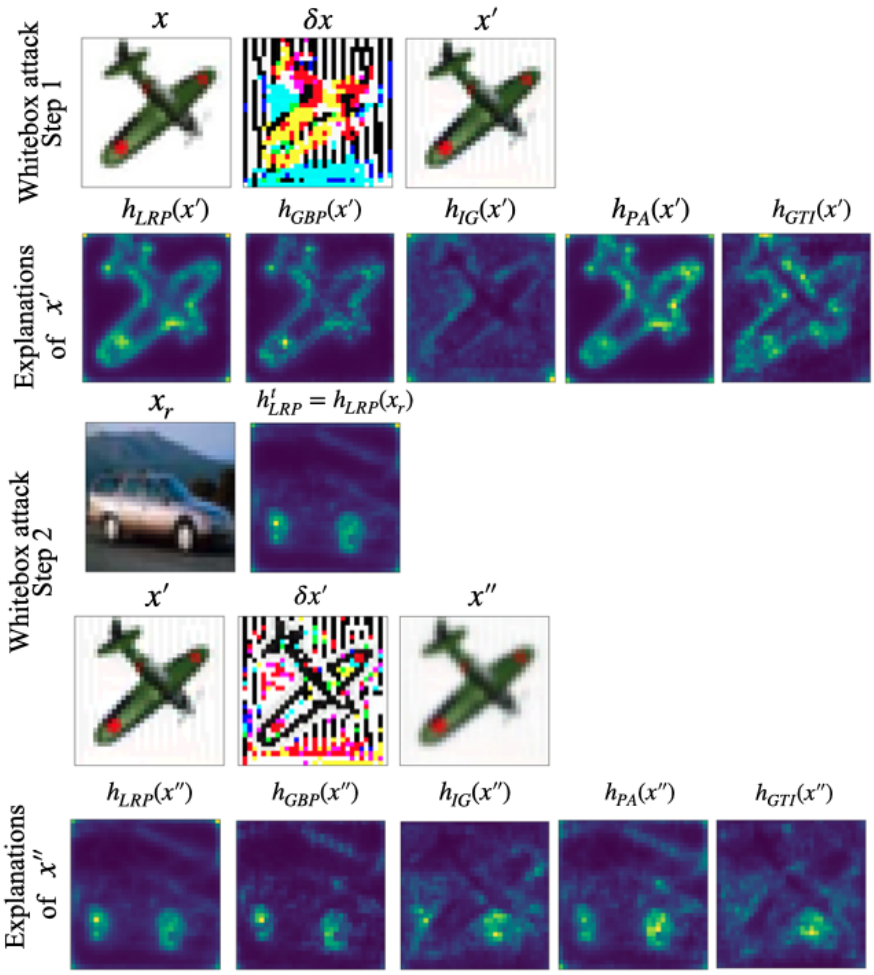

Figure 4.2: Illustration of whitebox attack.

are not consistent with the expected normal explanations of class Automobile. As an adaptive adversary, we now intend to target an explanation technique. For this illustration, we choose to target LRP. To this end, we randomly select an example x_r from class Automobile, for which the explanation map produced by LRP is classified as normal by the corresponding detector model. We show x_r as the first image in row 3. Its explanation map by LRP, shown as the second image in row 2, is set as the target map h^t_{LRP}. Then, we perform the optimization for the loss function 4.1 (shown in Section 4.6) using x' and h^t_{LRP}. After the optimization completes, we obtain the final adversarial example $x'' = x' + \delta x'$, which is shown as the rightmost image in row 4. The bottom row shows the explanation maps produced by the five techniques for x''. We observe that targeting

LRP results in the explanation map produced by LRP (first image in row 5) to be very close to the target map. Also, the explanation maps for GBP (second image in row 5) and PA (fourth image in row 5) are also fairly close to the target map. We find that all three of these explanation maps are classified as normal by their respective detector models. However, we observe that the explanation maps produced by the gradient-based techniques, i.e., IG (third image in row 5) and GTI (rightmost image in row 5), are not as close to the target map. Both these explanation maps are classified as abnormal by their respective detector models. Therefore, using an ensemble of detector models corresponding to diverse explanation techniques, our defense is able to mitigate this whitebox attack by correctly identifying x'' as an adversarial example.

4.6.3 Evaluation of Whitebox Attack

To perform the first step of the above approach, we re-use the successful adversarial examples X' created using blackbox attacks. However, while targeting integrated gradients (IG), we only used adversarial examples from CW_2 attack as we find targeting IG to incur very high cost. For targeting remaining techniques, we use adversarial examples from all six attacks. Then, for each adversarial example $x' \in X'$, we perform the second step using the optimization process described above to obtain final adversarial examples X''.

Table 4.10 shows a summary of the whitebox attack. Column 3 shows the mean success rate of the final adversarial examples in retaining the desired misclassification in the target model (when ExAD is not included). The mean is computed over success rates obtained for different attacks (except for IG where we only use CW_2 attack). We do not show individual success rates for each attack as they were very similar for any target technique. In Table 4.10, we observe that targeting LRP results in the highest success rate, with least L_2 distortion. However, as shown in Column 4, the average time required per example to target LRP is over 130 seconds which is quite high compared to targeting GBP, PA, or GTI techniques.

Figure 4.3 shows the results of the whitebox attack. Each row represents the targeted explanation technique. Columns 1-5 show the detection rate obtained by individual detector models corresponding to the five explanation techniques in correctly classifying explanation maps as ab-

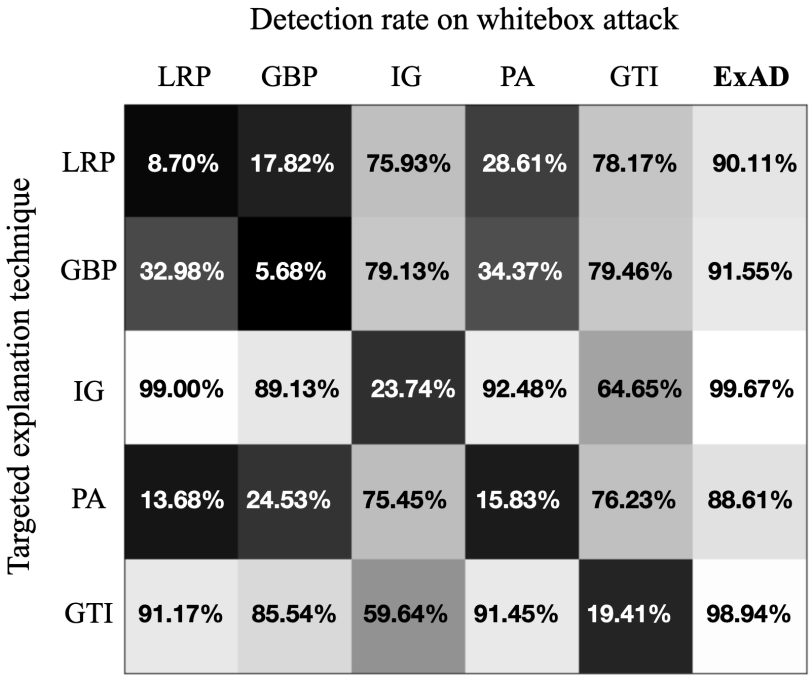

Figure 4.3: Transferability of whitebox attack.

normal. Column 6 shows the overall performance by ExAD-CNN in correctly classifying the examples as adversarial. From Figure 4.3, we make several interesting observations. First, we notice the values along the diagonal (from top-left to bottom-right) are all very low. This is natural as the detector model corresponding to the targeted technique is expected to correctly classify very few explanations as abnormal. For instance, targeting GTI causes its detector model to have a detection rate of only 19.41%. Second, we observe that targeting gradient-based techniques do not severely impact detector models of propagation-based techniques, and vice-versa. For instance, on targeting IG or GTI, the detector models corresponding to LRP, GBP, and PA still achieve detection rates above 85%. This is consistent with the transferability findings by Zhang et al. [30]. On a different set of diverse explanation techniques, the authors showed that manipulated images created by targeting one technique rarely produce desirable explanations (which are close to the target map) on other techniques. Finally, we find that targeting propagation-based (resp., gradient-based) techniques transfer well to detector models corresponding to the other propagation-based (resp., gradient-based) techniques. For instance, targeting LRP results in the GBP-based detec-

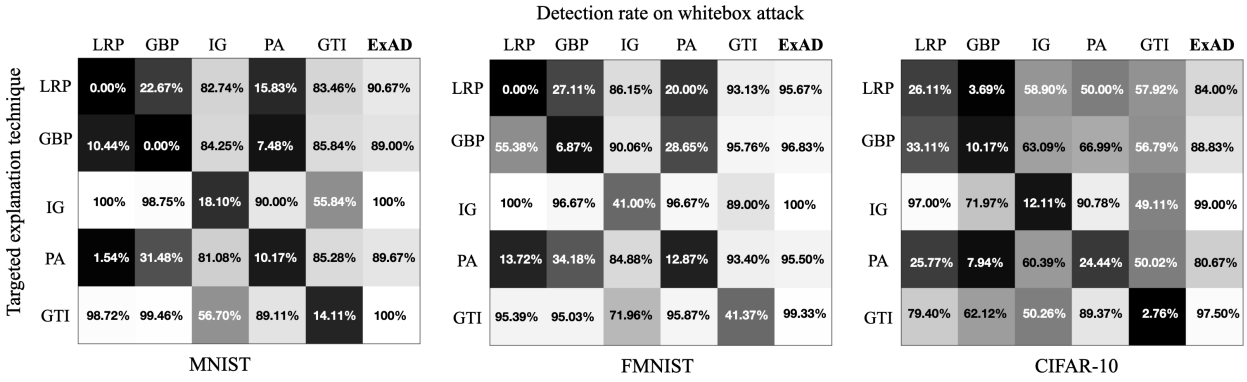

Figure 4.4: Transferability of whitebox attack on individual datasets.

tor model to have a detection rate of only 17.82%. The same phenomenon can be observed for gradient-based techniques (IG and GTI). This empirically supports the need to have diversity in the explanation methods to build robustness against adaptive attacks. As shown in Column 6 of Figure 4.3, using an ensemble of gradient-based and propagation-based techniques, ExAD is able to significantly limit the success rate of whitebox attacks. We find ExAD to be relatively more robust when gradient-based techniques are targeted. We achieve 99.67% and 98.94% detection rate for IG and GTI as the target, respectively. For propagation-based techniques, the highest impact is caused by targeting PA, which results in a detection rate of 88.61%. For the case of targeting LRP and GBP, we obtain detection rates of 90.11% and 91.55%, respectively.

Furthermore, in Figure 4.4, each value shows the mean detection rate for adversarial examples created using different attacks (except while targeting IG technique which only uses CW_2 attack) and considering all target classes. On all three datasets, we find that targeting a gradient-based technique has relatively less impact on detector models corresponding to propagation-based techniques. For instance, targeting GTI on MNIST causes the IG-based detector model to have a detection rate of only 56.70% whereas detector models corresponding to LRP, GBP, and PA have detection rates above 89%. Interestingly, on CIFAR-10 dataset, we observe that the detector models corresponding to IG and GTI are impacted by most techniques, although the impact is relatively higher when we target either of IG or GTI. For instance, on CIFAR-10 dataset, targeting PA has a noticeable impact on IG-based detector model as it achieves a detection rate of 60.39%, whereas

targeting GTI results in the IG-based detector model to have a relatively lower detection rate of 50.26%.

In terms of impact on ExAD, on all three datasets, targeting gradient-based techniques (IG and GTI) is relatively less effective. For instance, on MNIST, we obtain 100% detection rate by our approach while targeting IG and GTI techniques. In contrast, we observe that targeting propagation-based techniques is more effective across datasets. On MNIST dataset, the detection rate of the defense on propagation-based techniques is nearly 89%. On CIFAR-10 dataset, targeting PA results in the lowest detection rate of 80.67% by ExAD. On FMNIST dataset, the overall performance is fairly consistent while targeting propagation-based techniques. The highest impact is produced by targeting PA and LRP, for which ExAD obtains detection rates of 95.50% and 95.67%, respectively.

5. DISCUSSION

5.1 Fragility of Explanations

In Section 4.6, we discussed the reliability of explanations in context of an adaptive adversary. Recently, there has also been research that shows the fragility of explanations in an adversarial context. In this section, we discuss two related scenarios and any potential impact to our defense strategy.

5.1.1 Hiding the attack from explanations

Recently, *adversarial patches* [38, 51] were introduced to make adversarial examples more practical in the physical world. This attack restricts the spatial dimensions of the perturbation, but removes the imperceptibility constraint. However, Subramanya *et al.* [52] demonstrated that we can generate adversarial patches that not only fool the prediction by the target classifier, but also change the explanation of the modified example such that the adversarial patch is no longer considered important. Nevertheless, here the attacker only manages to make the explanation technique focus in a region outside the adversarial patch; she does not try to make the explanation itself appear more normal for the target class. Therefore, even though such adversarial examples may evade the explanation mechanism, they are still likely to be correctly classified as adversarial by our defense.

5.1.2 Changing the explanation but not the classification

Another possibility is that an attacker may add adversarial perturbations which produce examples that are classified into the same class, but have very different explanations [53]. Figure 5.1 shows an example of this attack for the case of random perturbations. The leftmost image in first row is a normal example from the Sneaker class of FMNIST dataset. The next image in first row shows a manipulated image created by adding random perturbations to the normal example such that the prediction remains unchanged, and the perturbations do not exceed a threshold value of 1, in terms of L_∞ distance. The next three images are created in a similar manner but with increased

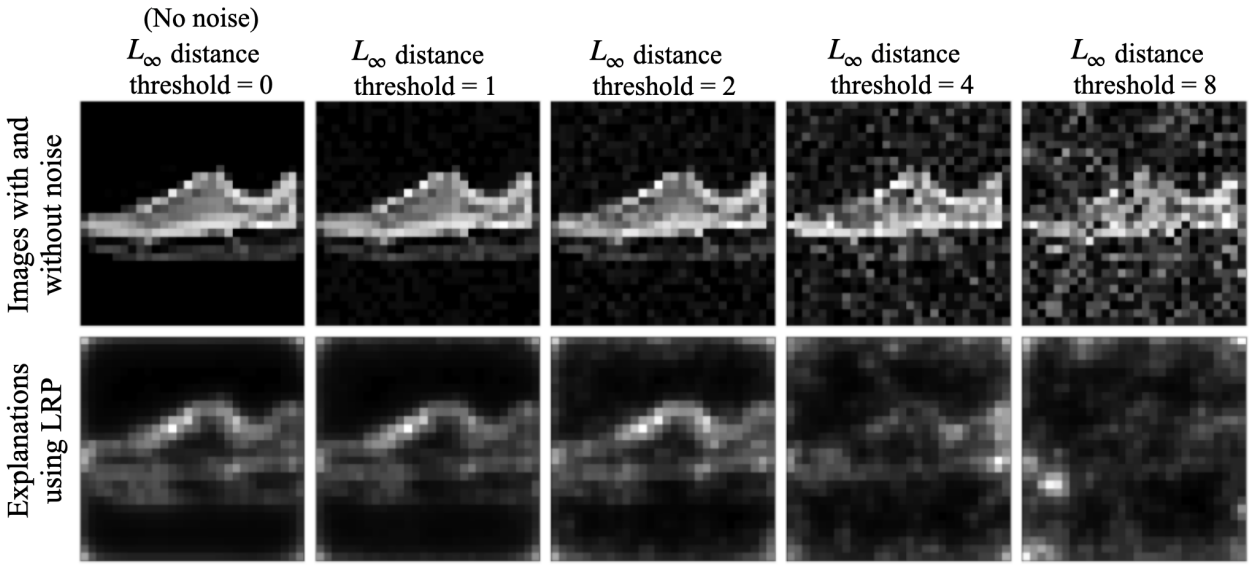

Figure 5.1: Effect of adding random perturbations on explanations.

threshold values of 2, 4, and 8, respectively. The second row shows the corresponding explanations produced by LRP technique. We observe that with increased noise threshold, the explanations also become noisy. With our defense, if such explanations are classified as abnormal, then the corresponding inputs would be considered adversarial (false-positive). Figure 5.2 shows that adding random perturbations to normal examples results in a decline of the target classifier's classification accuracy with increasing noise threshold.

Nevertheless, we believe the impact of this attack may depend upon the nature of the application where the defense is being used. If the perturbed examples resulting from such attacks are not considered normal for the system, then the abnormality produced in the explanations can be beneficial because the examples will likely be classified as adversarial. However, if an application considers such perturbed examples as normal, such as if the norm of perturbation is within an allowed threshold, then our defense could result in false-positives. For such applications, we would require explanation techniques to be robust enough to allowed perturbations. We leave further research towards such methods as future work.

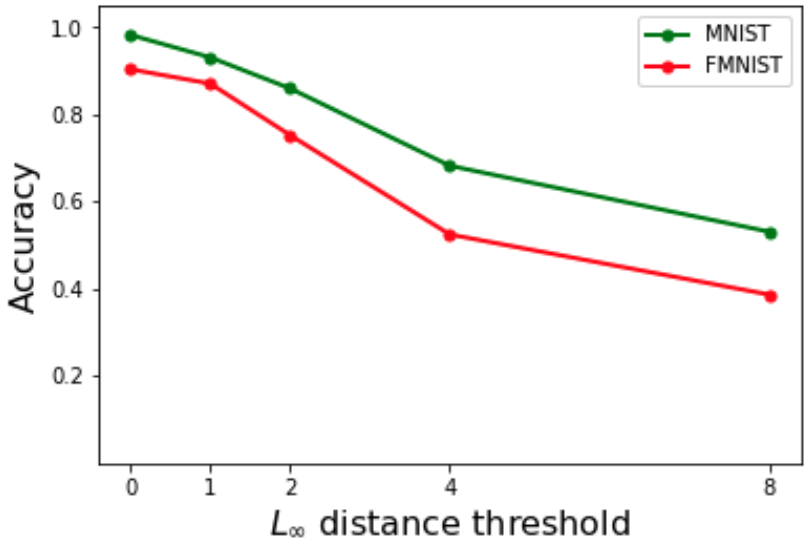

Figure 5.2: Impact of fragility of explanations.

5.2 Limitations

Our detection mechanism and scope of evaluation has certain limitations. First, our current evaluation only considers the targeted attack setting for generating adversarial examples. In future work, we will extend our evaluation to cover untargeted adversarial attacks as well. Second, currently we do not have a unified optimization-based approach that simultaneously (and successfully) targets multiple explanation techniques. We attempt doing so in two ways. One approach is to modify the loss function 4.1 as follows

$$\mathcal{L} = \Big(\sum_{j=1}^{5} ||h_j(x'') - h^t||^2 \Big) + \gamma ||f(x'') - f(x')||^2 \tag{5.1}$$

Here, we create the target map h^t using any one of the explanation techniques. However, in this case, when the target map is created using a gradient-based technique, we did not notice any significant change in the performance of the detector models corresponding to propagation-based techniques, and vice-versa. The results remained consistent with our findings in Figure 4.3. One reason behind this could be the diversity in explanation techniques due to which the target maps required by gradient-based techniques differ substantially from those required by propagation-

based techniques. Then, we further consider modifying the loss function 5.1 as follows

$$\mathcal{L} = \Big(\sum_{j=1}^{5} ||h_j(x'') - h_j^t||^2 \Big) + \gamma ||f(x'') - f(x')||^2 \tag{5.2}$$

Here, target maps used are created using the corresponding explanation techniques. However, in this case, we found the explanation-loss component did not reduce much during the optimization process. Moreover, it often led to memory errors on the GPU. One reason for this is that in the current whitebox attack framework[31], each explanation technique requires a different set of hyperparameters (e.g., learning rate, β growth, and iterations), as shown in Table 5.1. Therefore, we find that simultaneously attacking multiple explanation techniques is considerably difficult for an adaptive adversary. We leave further exploration on improving the efficiency and effectiveness of such attack to future work.

Table 5.1: Hyperparameters used in whitebox attack.

Method	Iterations	Learning Rate	Factors
LRP	1500	10^{-3}	2x10^{-4}, 10^{6}
GBP	1500	10^{-3}	10^{11}, 10^{6}
IG	500	$5x10^{-3}$	10^{11}, 10^{6}
PA	1500	$2x10^{-3}$	10^{11}, 10^{6}
GradxInput	1500	10^{-3}	10^{11}, 10^{6}

6. CONCLUSION

We proposed ExAD, a framework to detect adversarial examples using an ensemble of explanation techniques. The use of explanations is motivated by the distinguishability between normal and abnormal explanations for any target class. Furthermore, motivated by previous work on N-variant systems, we used an ensemble of gradient-based and propagation-based explanation techniques to introduce diversity in our defense. Experiments showed that our approach is effective against blackbox attacks, and outperforms three state-of-the-art detectors. We also find that ExAD significantly limits the success rate of whitebox attacks. In this process, we made interesting findings on the transferability of adaptive attacks. We acknowledge the possibility of more sophisticated whitebox attacks in future, and hope our work will inspire further research in this direction. We believe our proposed defense is complementary to state-of-the-art detection methods and can be used in conjunction with them to boost the detection of adversarial attacks.

REFERENCES

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

[2] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *arXiv preprint arXiv:1704.01155*, 2017.

[3] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. Wijewickrema, G. Schoenebeck, D. Song, M. E. Houle, and J. Bailey, "Characterizing adversarial subspaces using local intrinsic dimensionality," *arXiv preprint arXiv:1801.02613*, 2018.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[5] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury, *et al.*, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal processing magazine*, vol. 29, 2012.

[6] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu, "Large-scale malware classification using random projections and neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3422–3426, IEEE, 2013.

[7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[8] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016.

[9] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[10] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *arXiv preprint arXiv:1611.02770*, 2016.

[11] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," *arXiv preprint arXiv:1412.5068*, 2014.

[12] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE S&P*, pp. 39–57, IEEE, 2017.

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

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

[15] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," *ICLR*, 2017.

[16] D. Hendrycks and K. Gimpel, "Early methods for detecting adversarial images," *arXiv preprint arXiv:1608.00530*, 2016.

[17] X. Li and F. Li, "Adversarial examples detection in deep networks with convolutional filter statistics," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5764–5772, 2017.

[18] A. N. Bhagoji, D. Cullina, and P. Mittal, "Dimensionality reduction as a defense against evasion attacks on machine learning classifiers," *arXiv preprint arXiv:1704.02654*, 2017.

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

[20] S. Ma, Y. Liu, G. Tao, W.-C. Lee, and X. Zhang, "Nic: Detecting adversarial samples with neural network invariant checking.," in *NDSS*, 2019.

[21] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[22] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. Mueller, "How to explain individual classification decisions," *arXiv preprint arXiv:0912.1128*, 2009.

[23] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje, "Not just a black box: Learning important features through propagating activation differences," *arXiv preprint arXiv:1605.01713*, 2016.

[24] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *ICML*, 2017.

[25] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[26] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, 2015.

[27] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, "Explaining nonlinear classification decisions with deep taylor decomposition," *Pattern Recognition*, vol. 65, pp. 211–222, 2017.

[28] P.-J. Kindermans, K. T. Schütt, M. Alber, K.-R. Müller, D. Erhan, B. Kim, and S. Dähne, "Learning how to explain neural networks: Patternnet and patternattribution," *arXiv preprint arXiv:1705.05598*, 2017.

[29] P.-J. Kindermans, S. Hooker, J. Adebayo, M. Alber, K. T. Schütt, S. Dähne, D. Erhan, and B. Kim, "The (un) reliability of saliency methods," *arXiv preprint arXiv:1711.00867*, 2017.

[30] X. Zhang, N. Wang, H. Shen, S. Ji, X. Luo, and T. Wang, "Interpretable deep learning under fire," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020.

[31] A.-K. Dombrowski, M. Alber, C. Anders, M. Ackermann, K.-R. Müller, and P. Kessel, "Explanations can be manipulated and geometry is to blame," in *Advances in Neural Information Processing Systems*, pp. 13589–13600, 2019.

[32] B. Cox, D. Evans, A. Filipi, J. Rowanhill, W. Hu, J. Davidson, J. Knight, A. Nguyen-Tuong, and J. Hiser, "N-variant systems: A secretless framework for security through diversity.," in *USENIX Security Symposium*, pp. 105–120, 2006.

[33] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "Smoothgrad: removing noise by adding noise," *arXiv preprint arXiv:1706.03825*, 2017.

[34] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[35] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017.

[36] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[37] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[38] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," *arXiv preprint arXiv:1712.09665*, 2017.

[39] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning models," *arXiv preprint arXiv:1707.08945*, 2017.

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

[41] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *CVPR*, 2018.

[42] C. Zhang, Z. Ye, Y. Wang, and Z. Yang, "Detecting adversarial perturbations with saliency," in *2018 IEEE 3rd International Conference on Signal and Image Processing (ICSIP)*, pp. 271–275, IEEE, 2018.

[43] N. Liu, H. Yang, and X. Hu, "Adversarial detection with model interpretation," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1803–1811, 2018.

[44] J. Wang, Y. Wu, M. Li, X. Lin, J. Wu, and C. Li, "Interpretability is a kind of safety: An interpreter-based ensemble for adversary defense," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 15–24, 2020.

[45] Z. C. Lipton, "The mythos of model interpretability," *arXiv preprint arXiv:1606.03490*, 2016.

[46] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, 2019.

[47] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[48] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," in *Advances in Neural Information Processing Systems*, pp. 125–136, 2019.

[49] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, and A. K. et al., "Technical report on the cleverhans v2.1.0 adversarial examples library," *arXiv preprint arXiv:1610.00768*, 2018.

[50] N. Carlini, "Robust evasion attacks against neural network to find adversarial examples.." https://github.com/carlini/nn_robust_attacks.

[51] D. Karmon, D. Zoran, and Y. Goldberg, "Lavan: Localized and visible adversarial noise," *arXiv preprint arXiv:1801.02608*, 2018.

[52] A. Subramanya, V. Pillai, and H. Pirsiavash, "Towards hiding adversarial examples from network interpretation," *arXiv preprint arXiv:1812.02843*, 2018.

[53] A. Ghorbani, A. Abid, and J. Zou, "Interpretation of neural networks is fragile," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 3681–3688, 2019.