Interpretability is a Kind of Safety: An Interpreter-based Ensemble for Adversary Defense

Jingyuan Wang[1,3,4], Yufan Wu[1], Mingxuan Li[1], Xin Lin[1], Junjie Wu[2,3,*], Chao Li[1,4]
1. MOE Engineering Research Center of ACAT, School of Computer Science Engineering, Beihang University
2. Beijing Key Laboratory of ESSTCO, School of Economics and Management, Beihang University, Beijing, China
3. State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China
4. Beijing Advanced Innovation Center for BDBC, Beihang University, Beijing, China * Corresponding author.

ABSTRACT

While having achieved great success in rich real-life applications, deep neural network (DNN) models have long been criticized for their vulnerability to adversarial attacks. Tremendous research efforts have been dedicated to mitigating the threats of adversarial attacks, but the essential trait of adversarial examples is not yet clear, and most existing methods are yet vulnerable to hybrid attacks and suffer from counterattacks. In light of this, in this paper, we first reveal a gradient-based correlation between sensitivity analysis-based DNN interpreters and the generation process of adversarial examples, which indicates the Achilles's heel of adversarial attacks and sheds light on linking together the two long-standing challenges of DNN: fragility and unexplainability. We then propose an interpreter-based ensemble framework called X-Ensemble for robust adversary defense. X-Ensemble adopts a novel detection-rectification process and features in building multiple sub-detectors and a rectifier upon various types of interpretation information toward target classifiers. Moreover, X-Ensemble employs the Random Forests (RF) model to combine sub-detectors into an ensemble detector for adversarial hybrid attacks defense. The non-differentiable property of RF further makes it a precious choice against the counterattack of adversaries. Extensive experiments under various types of state-of-the-art attacks and diverse attack scenarios demonstrate the advantages of X-Ensemble to competitive baseline methods.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**.

KEYWORDS

Adversarial Example Defense, DNN Interpretation, Ensemble

ACM Reference Format:
Jingyuan Wang[1,3,4], Yufan Wu[1], Mingxuan Li[1], Xin Lin[1], Junjie Wu[2,3,*], Chao Li[1,4]. 2020. Interpretability is a Kind of Safety: An Interpreter-based Ensemble for Adversary Defense. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394486.3403044

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403044

1 INTRODUCTION

Recent years have witnessed the unprecedented success of neural network-based deep learning (DL) models being widely deployed in many critical applications, such as financial investment and trading [33], and route recommendation [12, 32]. But with power comes a fatal weakness: the fragility of DL models when they face adversarial attacks. As early proposed by Szegedy *et al.* [28], it is very easy to mislead the outputs of a DL model by slightly perturbing input examples to form adversarial examples. This weakness causes DL models very unreliable in applications that are vulnerable to deliberate attacks, such as dangerous goods transportation [29] and military systems, and eventually could lead to substantial economic losses and even the costs of lives.

In the literature, tremendous research efforts have been devoted to designing adversary defense (AD) methods to mitigate the threats of adversarial examples. Studies in this stream can be roughly divided into two categories: one is to strengthen target models [18, 22] and the other is to detect and drop adversarial examples [14, 16, 24]. While these methods have made great success in adversary defense, most of them yet suffer from three fundamental and long-standing challenges. The first challenge is to explore the intrinsic mechanism of adversarial attacks so as to enhance the defense ability of DL methods. The second challenge is to defense hybrid adversarial attacks that might include various types of attacks or even unknown types. The third challenge is an interesting game problem: how to protect a defender itself from adversarial attacks? These challenges indeed motivate our study in this paper.

We argue that the adversarial attacks have a close connection to another widely-admitted fatal weakness of DL models: the unexplainable problem [9]. Deep neural networks are typically regarded as "black-boxes" since their stacked model structures are very hard to understand by human intuitions, which as the attack vulnerability also makes them not trustworthy in real-life critical applications. While there has been a lot of research on improving the interpretability of DL models, it is not until quite recently there appear some works that consider both the interpretation and the adversarial attacks from a cross-cutting perspective. It is reported that some types of local interpretations are more sensitive to adversarial attacks [8] and a more robust neural network has better interpretability [6]. Along this line, we may find the Achilles' heel of adversarial examples and design interpretation-based models for high-performance adversary defense.

In this work, we address the above three AD challenges by designing an interpreter-based ensemble framework called X-Ensemble for robust adversary defense. Specifically, we reveal a gradient-based correlation between the sensitivity analysis-based interpreters and

the generation process of adversarial examples, which implies that the Achilles' heel of an adversarial example is the interpretation map. Following this line, we propose the X-Ensemble model, which adopts a detection-rectification pipeline and consists of three main components: *i*) A group of interpreter-based sub-detectors to classify input samples as adversarial/benign ones. The sub-detectors are all DNN-based models that can achieve high performance through fully exploiting different types of interpretation information toward target classifiers. *ii*) An ensemble mechanism uses a Random Forest model to combine sub-detector as a final detector. On one hand, an ensemble can improve the robustness of sub-detectors for hybrid attacks detection. On the other, the non-differentiable property of Random Forest (RF) can help against adversarial attacks toward detectors themselves. *iii*) An interpreter-based rectifier recovers adversarial examples to benign ones. With the guidance of model interpreters, our rectifier can purposefully erase pixels perturbed by attackers in a randomized mechanism.

Our main contribution can be summarized as follows:

• First, we reveal an underlying connection between sensitivity analysis-based interpretation of DL models and adversarial examples generation. We leverage this fundamental connection to design the detector and rectifier of X-Ensemble, and all achieved advantageous performances (*i.e.,* solution to the first AD challenge).

• Second, we propose an RF-based framework to combine multiple types of interpreters as an ensemble detector for adversarial attacks of various types (*i.e.,* solution to the second AD challenge). The non-differentiable property of RF makes our detector much more robust compared with existing differentiable detectors (*i.e.,* solution to the third AD challenge).

• Finally, the performance of X-Ensemble is verified over three popular image data sets under the attacks of five types of state-of-the-art adversarial examples. For diversified attack scenarios, including white-/black-/grey-box threat models, vaccinated/unvaccinated training, as well as targeted/untargeted adversarial specificities, our model all shows superior accuracy and robustness.

2 THE MODEL FRAMEWORK

Threats of adversarial examples exist in many applications, such as image classification [13] and natural language processing [36], and traffic speed prediction [30]. In this work, we select image classification as an example to introduce our model, which can be extended to other application scenarios.

We use a matrix $\boldsymbol{X} \in \mathbb{R}^{I \times J}$ to denote an image with $I \times J$ pixels. Given an image $\boldsymbol{X}$, we have a one-hot coded label, denoted as $\boldsymbol{y}^\circ \in \{0,1\}^L$, to indicate its category. Note that $\|\boldsymbol{y}^\circ\|_1 = 1$, *i.e.,* only one item equals 1 in $\boldsymbol{y}^\circ$, and the l-th item equaling 1 means the image belongs to the class l, $l = 1, \cdots, L$. For adversarial examples in image classification, there exists a target classifier and an attacker.

Definition 1 (**Classifier**). *Given a set of images, we train a* classifier $F : \mathbb{R}^{I \times J} \to [0,1]^L$. *For each image sample* $\boldsymbol{X}$ *with a true label* $\boldsymbol{y}^\circ$, *the predicted label given by* F *is denoted as* $\hat{\boldsymbol{y}} = F(\boldsymbol{X})$. *We assume* F *is well trained and* $\hat{\boldsymbol{y}} = \boldsymbol{y}^\circ$ *for most of samples.*

In this work, we assume the classifier is implemented using deep neural network models.

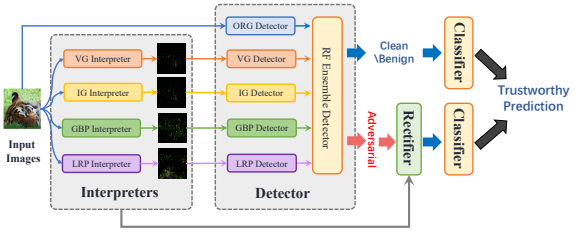

Figure 1: The framework of X-Ensemble.

Definition 2 (**Attacker**). *Given a classifier* F *for a set of images, we train an* attacker $A : \mathbb{R}^{I \times J} \to \mathbb{R}^{I \times J}$. *For each original image* $\boldsymbol{X}^\circ$ *with a true label* $\boldsymbol{y}^\circ$, *the attacker can generate an adversarial counterpart* $\boldsymbol{X}^{(a)} = A(\boldsymbol{X}^\circ)$, *satisfying that* $F(\boldsymbol{X}^{(a)}) \neq \boldsymbol{y}^\circ = F(\boldsymbol{X}^\circ)$ *and* $\text{Dist}(\boldsymbol{X}^\circ, \boldsymbol{X}^{(a)}) < \epsilon$, *where* $\text{Dist}(\cdot,\cdot)$ *is a distance measurement between two images, and* ϵ *is a small threshold to limit the difference between a benign example and its adversarial counterpart.*

The distance function $\text{Dist}(\cdot,\cdot)$ for attackers could be l_0, l_2, l_∞ norms or others [35]. For the attacker A, the classifier F is also called a target classifier for correspondence.

Since the classifier F is a DNN-based "black-box" model, we use sensitivity-analysis-based interpreters [27] to explain F. As will be given in Sec. 3.1, sensitivity-analysis-based interpreters have close connections to adversarial examples, which are defined as follows.

Definition 3 (**Interpreter**). *Given a classifier* F, *we define its interpreter as* $Ex : \mathbb{R}^{I \times J} \to \mathbb{R}^{I \times J \times L}$. *For an input image* $\boldsymbol{X}$, *the interpreter generates a sensitivity tensor* $\mathcal{R} = Ex(\boldsymbol{X})$, *and the* (i,j,l)*-th item of* $\mathcal{R}$, i.e., r_{ijl}, *indicates the importance of* x_{ij} *for* F *if labeling* $\boldsymbol{X}$ *with the class* l.

In this work, we propose a framework consisting of a detector and a rectifier for adversary defense, where the detector is in charge of detecting adversarial examples and the rectifier is in charge of recovering adversarial examples to benign ones. Below are the formal definitions of the two components.

Definition 4 (**Detector**). *A detector is a function in the form of* $Det : \mathbb{R}^{I \times J} \times \mathbb{R}^{I \times J \times L} \to \{0,1\}$. *For an input image* $\boldsymbol{X}$ *and its classifier* F, *a detector gives a binary output*

$$z = Det(\boldsymbol{X}, Ex(\boldsymbol{X})) \in \{0,1\}, \tag{1}$$

indicting that $\boldsymbol{X}$ *is an adversarial example if* $z = 1$ *and a benign example otherwise.*

As reported by some previous works [3] , using only $\boldsymbol{X}$ as feature to detect adversarial examples is not very robust. We therefore incorporate interpretations of $\boldsymbol{X}$ *w.r.t.* the target classifier into the detector, *i.e.,* $Ex(\boldsymbol{X})$, as given in the above definition. Similarly, we also include interpretations to build a rectifier as follows.

Definition 5 (**Rectifier**). *A rectifier is a function in the form of* $Rec : \mathbb{R}^{I \times J} \times \mathbb{R}^{I \times J \times L} \to \mathbb{R}^{I \times J}$. *For an adversarial example* $\boldsymbol{X}^{(a)}$ *and its classifier* F, *a rectifier can convert it to a benign example as*

$$\boldsymbol{X}^{(r)} = Rec(\boldsymbol{X}^{(a)}, Ex(\boldsymbol{X}^{(a)})), \tag{2}$$

such that $F(\boldsymbol{X}^{(r)}) = \boldsymbol{y}^{(r)} = \boldsymbol{y}^\circ$.

Framework of X-Ensemble. Given the above definitions, we now introduce the general framework of our adversary defense model: X-Ensemble, as shown in Fig. 1, by combining the detector and the rectifier as a pipeline. Given an input image $\boldsymbol{X}$, the model first uses the detector *Det* to classify it as a benign or adversarial example. If $\boldsymbol{X}$ is benign, X-Ensemble uses the classifier F to predict a label, otherwise, X-Ensemble uses the rectifier to recover $\boldsymbol{X}$ as $\boldsymbol{X}^{(r)}$ and then applies F on $\boldsymbol{X}^{(r)}$ to make the prediction. The image classification in our model is essentially in the form of

$$\boldsymbol{y} = \begin{cases} F(\boldsymbol{X}) & \text{if } Det(\boldsymbol{X}, \mathcal{R}) = 0 \\ F(Rec(\boldsymbol{X}, \mathcal{R})) & \text{if } Det(\boldsymbol{X}, \mathcal{R}) = 1 \end{cases}, \tag{3}$$

where $\mathcal{R} = Ex(\boldsymbol{X})$. Fig. 1 is the illustration of Eq. (3), and more details about model training will be given in the following sections.

It is noteworthy that X-Ensemble is novel in integrating the detection and rectification of adversarial examples into a unified process for high-quality prediction. X-Ensemble is also novel in exploiting example-level interpretation information for building robust detectors and rectifiers, which requires a local interpreter for the classifier to be built in advance.

3 THE DETECTOR

3.1 The Idea of Detector

As shown in Eq. (1), the detector uses the outputs of an interpreter as inputs. This design is motivated by the connection between adversarial attacks and model interpretations [6, 8], with the details introduced below.

The interpreters adopted by X-Ensemble are gradient-based sensitivity analysis methods. The idea of this kind of interpreters is to describe the influences of inputs to outputs in a classifier. A simple method to describe this influences is using the gradient of the output $y^{(l)} = F_l(\boldsymbol{X})$ to the input pixel x_{ij}, *i.e.,*

$$g_{ijl} = \frac{\partial F_l(\boldsymbol{X})}{\partial x_{ij}}, \tag{4}$$

where F_l is the l-th component of the classifier F's output logits. A large value of $\left| g_{ijl} \right|$ means that the pixel x_{ij} is an important factor for classifying the image $\boldsymbol{X}$ as the class l.

The gradient g_{ijl} also plays an important role in adversarial examples generation. Given a raw image $\boldsymbol{X}^{\circ}$ and a target classifier F, the objective function of generating an adversarial example $\boldsymbol{X}^{(a)}$ that can fool F to give an output $l^{(a)}$ is

$$\begin{aligned} &\underset{\boldsymbol{X}^{(a)}}{\arg\min}\ \mathcal{L}\left(F\left(\boldsymbol{X}^{(a)}\right), l^{(a)}\right), \\ &s.t.\ \mathrm{Dist}\left(\boldsymbol{X}^{(a)}, \boldsymbol{X}^{\circ}\right) < \epsilon, \end{aligned} \tag{5}$$

where $\mathcal{L}$ is a loss function and the cross entropy is the usual choice.

If we use the gradient descent method to optimize the objective function Eq. (5), the pixel x_{ij} can be iteratively perturbed as

$$x_{ij}^{(\tau+1)} := \Gamma_{D_\epsilon(\boldsymbol{X}^{\circ})}\underbrace{\left(x_{ij}^{(\tau)} - \alpha \frac{\partial \mathcal{L}\left(F\left(\boldsymbol{X}^{(\tau)}\right),\ l^{(a)}\right)}{\partial x_{ij}^{(\tau)}}\right)}_{\text{Term I}}, \tag{6}$$

where $\Gamma_{D_\epsilon(\boldsymbol{X}^{\circ})}$ is the projection operator to ensure $\mathrm{Dist}(\boldsymbol{X}^{(a)}, \boldsymbol{X}^{\circ}) < \varepsilon$, τ is an iteration round index, and α is a learning rate. Using the chain rule, we rewrite *Term I* in Eq. (6) as

$$x_{ij}^{(\tau)} - \alpha \frac{\partial \mathcal{L}}{\partial F_{l^{(a)}}\left(x_{ij}^{(\tau)}\right)} \cdot g_{ijl^{(a)}}. \tag{7}$$

As shown in Eq. (7), pixels with larger $\left| g_{ijl^{(a)}} \right|$ tend to be more seriously perturbed by attackers. Although different attackers may adopt different objective functions and optimization algorithms, the endogenous connection between adversarial attacks and g_{ijl}-based interpretations can be described schematically by Eq. (4)-Eq. (7).

Inspired by the above connection between attackers and interpreters, we argue that the g_{ijl} information can be adopted to detect adversarial examples. Specifically,

- *From the attacker perspective*, a pixel with large $|g_{ijl}|$ in an adversarial example tends to be perturbed until its $|g_{ijl}|$ is low enough and significantly different from its benign counterpart. In other words, although the attacker in Eq. (6) constrains the distance between $\boldsymbol{X}^{\circ}$ and $\boldsymbol{X}^{(a)}$ using the Γ operator, the distance between the gradient matrices, *i.e.,* $\boldsymbol{G}_l^{\circ}$ and $\boldsymbol{G}_l^{(a)}$ consisting of g_{ijl}, is not constrained consciously and might provide important clues for adversarial and benign examples differentiation.
- *From the interpreter perspective*, the interpretation matrix $\boldsymbol{G}_l$ indicates the pixels that have decisive influences to image classification of the label l. For a benign image with a correct classification, the distribution of high $|g_{ijl}|$ pixels should be in line with human's visual cognition, *i.e.,* heavily distributed on indication objects of an image, such as a dog in an image. However, this might not be the case for an adversarial image with a misleading classification. In other words, we could expect a great difference for high $|g_{ijl}|$ pixel distributions in a benign example and its adversarial counterpart. Therefore, we can train a model to classify benign and adversarial examples by leveraging this difference.

The above insights indeed motivates the design of our detector for the X-Ensemble model below.

3.2 Interpreters in X-Ensemble

Given the effectiveness of interpretation information for detector training, the detector is yet vulnerable to adversarial attacks if only one kind of interpreters is used. Thus, we adopt an ensemble approach to combine multiple interpretation methods as the detector of X-Ensemble. The interpretation methods adopted by X-Ensemble include four types: Vanilla Gradient (VG), Integrated Gradient (IG) [27], Guided Backpropagation (GBP) [25] and Layer-wise Relevance Propagation (LRP) [20]. The principle of method selection is to allow as much diversity as possible.

Vanilla Gradient. Given a classifier $\hat{\boldsymbol{y}} = F(\boldsymbol{X})$, the influence of pixels in $\boldsymbol{X}$ to the one-hot classification output $\hat{\boldsymbol{y}}$ in the vanilla gradient is calculated as

$$\mathcal{G}^{(VG)} = \frac{\partial F(\boldsymbol{X})}{\partial \boldsymbol{X}}, \tag{8}$$

where $\mathcal{G} \in \mathbb{R}^{I \times J \times L}$ is the gradient matrix *w.r.t.* F. The (i, j, l)-th element of $\mathcal{G}$, *i.e.,* g_{ijl}, expresses the influence of pixel x_{ij} to the class l.

Integrated Gradient. The integrated gradient calculates the influence of pixels x_ij to output y_l as an integration of vanilla gradient [27], *i.e.,*

$$g_{ijl}{}^{(IG)} = (x_{ij} - x'_{ij}) \times \int_{\alpha=0}^{1} \frac{\partial F_l\left(x'_{ij} + \alpha(x_{ij} - x'_{ij})\right)}{\partial x_{ij}} \, \mathrm{d}\alpha, \tag{9}$$

where x'_{ij} is the pixel of a baseline image usually set as a blank picture. The tensor $\mathcal{G}^{(IG)}$, which consists of g_{ijl}, is proposed to avoid the gradient saturation problem of the vanilla gradient for neural network interpretation.

Guided Backpropagation. GBP modifies the local gradient of ReLU activation neuron [25]. For a neural network classifier F, we denote f_m as the output of a ReLU activation neuron at the m-th layer, and r_m, r_{m+1} are the vanilla gradients of $\hat{\boldsymbol{y}}$ to the ReLU neuron's output and input respectively in gradient backward pass chain. GBP modifies the output gradient of a ReLU activation neuron as

$$r_m = \mathbb{1}(f_{m-1} > 0) \cdot \mathbb{1}(r_{m+1} > 0) \cdot r_{m+1}, \tag{10}$$

where $\mathbb{1}(\cdot) \in \{0, 1\}$ is an indicative function. GBP uses Eq. (10) to calculate local gradient of each ReLU neuron, and uses the vanilla gradient defined in Eq. (8) to calculate the end-to-end interpretation tensor as $\mathcal{G}^{(GBP)}$. The GBP method is proposed to improve the interpretation performance of gradient methods for neural networks with ReLU activation functions.

Layer-wise Relevance Propagation. LRP is an approximate method of Taylor decomposition [20]. LRP assigns each pixel x_{ij} a relevance score g_{ijl}, which have a relation with y_l as

$$y_l = \sum_{ij} g_{ijl}. \tag{11}$$

The g_{ijl} is calculated along the backward pass of neural networks layer by layer. Specifically, for the $m+1 \rightarrow m$ layers, LRP calculates a relevance $r_m^{(p)}$ for the p-th neuron in the layer m as

$$r_m^{(p)} = \sum_q r_{m+1}^{(q)} \frac{f_{m+1}^{(p)} w_{pq}}{\sum_h f_{m+1}^{(h)} w_{hq}} \tag{12}$$

where $f_{m+1}^{(p)}$ is the output of the p-th neuron in the layer $m+1$, and w_{pq} is the weight connecting the neurons $f_{m+1}^{(p)}$ and $f_m^{(q)}$. LRP is considered as a type of gradient-based interpretation method. The tensor composed of g_{ijl}, denoted as $\mathcal{G}^{(LRP)}$, is adopted as the input of our detector.

3.3 Ensemble Detector

The ensemble detector of X-Ensemble consists of multiple sub-detectors and an ensemble classifier.

3.3.1 Sub-Detectors. For each interpreter, X-Ensemble trains a Convolutional Neural Network (CNN) to classify image $\boldsymbol{X}$ as an adversarial or benign example. The CNN's input is $\boldsymbol{G}_l$ generated by the interpreter, where l is the index of the class predicted by the original image classifier F and hence $\boldsymbol{G}_l$ is the slice in the tensor $\mathcal{G}$ corresponding to class l. We call these adversarial/benign example classifiers as sub-detectors of X-Ensemble. For the four interpreters

Table 1: Attack success rate with largest g_{ijl} pixels dropped.

Erased Rate	Deepfool	CW	DDN
top 0%	1.000	1.000	1.000
top 5%	0.637	0.665	0.656

in Sec. 3.2, the corresponding sub-detectors are in the form of

$$\begin{aligned} \boldsymbol{z}^{(VG)} &= \mathrm{CNN}_1\left(\boldsymbol{G}_l^{(VG)}\right), \qquad \boldsymbol{z}^{(IG)} = \mathrm{CNN}_2\left(\boldsymbol{G}_l^{(IG)}\right), \\ \boldsymbol{z}^{(GBP)} &= \mathrm{CNN}_3\left(\boldsymbol{G}_l^{(GBP)}\right), \qquad \boldsymbol{z}^{(LRP)} = \mathrm{CNN}_4\left(\boldsymbol{G}_l^{(LRP)}\right). \end{aligned} \tag{13}$$

Besides the four interpretation-based sub-detectors, we also adopt original images as inputs to train a sub-detector as

$$\boldsymbol{z}^{(ORG)} = \mathrm{CNN}_5(\boldsymbol{X}). \tag{14}$$

To train the above sub-detectors, we need to adopt attack algorithms to prepare adversarial examples. In the literature, however, there exists a large number of attackers [35], and it is unpractical to train a set of sub-detectors for each attacker. We therefore group attackers according to their perturbation measurements, which typically include l_0, l_2, and l_∞. To further simplify the problem, we only adopt l_2 and l_∞ measurements in our experiments because they are the most widely used in the state-of-the-art attackers. In fact, the performance of using l_2 and l_∞ measurements to detect l_0 attackers is also satisfactory enough, which we will show in Sec. 5.

Given a clean example set $\mathcal{X} = \{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$, we use different adversarial models to generate two sets of adversarial counterparts: one is generated by l_2 attackers and the other is generated by l_∞ attackers. We mix $\mathcal{X}$ and the two sets of adversarial examples as a training set. For a training example, we set a label $\hat{\boldsymbol{z}} \in \{0, 1\}^3$, where $\hat{\boldsymbol{z}} = (1, 0, 0)$ indicates clean example, $\hat{\boldsymbol{z}} = (0, 1, 0)$ indicates l_2 attack, and $\hat{\boldsymbol{z}} = (0, 0, 1)$ indicates l_∞ attack. Finally, we use the training set to train the sub-detectors defined in Eq. (13) and Eq. (14).

3.3.2 Ensemble Classifier. X-Ensemble adopts a random forest (RF) to combine the sub-detectors as an ensemble adversarial/benign classifier, *i.e.,*

$$z = RF\left(\boldsymbol{z}^{(ORG)}, \boldsymbol{z}^{(VG)}, \boldsymbol{z}^{(IG)}, \boldsymbol{z}^{(GBP)}, \boldsymbol{z}^{(LRP)}\right). \tag{15}$$

The ensemble detector no longer classify adversarial examples by their perturbation measurements. The output z is the predicted benign/adversarial label, *i.e.,* $z = 0$ for benign examples and $z = 1$ for adversarial examples.

Compared with CNN-based models, the random forest model is non-differentiable and therefore not easy to be directly attacked. Moreover, the random forest as an ensemble detector is much more robust than a single sub-detector when facing different types of adversarial attacks even unknown ones.

4 RECTIFIER

Most of traditional studies in adversarial attack detection directly drop suspicious examples to protect target classifiers [14, 16]. Here, we go beyond the detect-and-drop scheme and set a rectifier to recover adversarial examples as benign ones.

The proposed rectifier also has a close connection with interpreters. In a sensitivity-based interpretation map $\boldsymbol{G}_l$, a large value of g_{ijl} indicates the pixel x_{ij} is an important factor for assigning the image $\boldsymbol{X}$ with the class l. For an adversarial example, its predicted

label l is unreliable, so the large g_{ijl} pixels are very likely to be modified by attackers to mislead the classifier. Therefore, it is intuitive to erase the pixels with large g_{ijl} in an adversarial example to weaken the attack effects.

Table 1 is a toy experiment. We use three attackers, *i.e.,* Deepfool [21], CW [4] and DDN [22], to generate adversarial examples for a classifier model, and set the pixels with top 5% largest g_{ijl} in the adversarial examples as zero, where g_{ijl} is the vallina gradient. As can be seen from the table, the success rates of the original adversarial examples are 100% while the modified ones fall to only 63.7%. This confirms our argument that g_{ijl} can be used to pick polluted pixels out from adversarial examples. We design our rectifier based on this observation accompanied by a random erasion scheme.

Given an adversarial example $\boldsymbol{X}^{(a)}$ with its predicted label l, our rectifier selects a pixel as a suspect when its g_{ijl} is larger than a threshold ; that is,

$$g_{ijl} > \alpha \cdot \left(g_{max} - g_{min}\right) + g_{min}, \tag{16}$$

where g_{max} and g_{min} are the maximum and minimum values among all pixel's g_{ijl} in the adversarial example, and α is a preset parameter to control the ratio of suspect pixels.

For a suspect pixel x_{ij}, the rectifier randomly erases it by setting

$$x_{ij}^{(r)} := \begin{cases} x_{ij}^{(a)} & \text{if } u_{ran} = 0, \\ x_{ij}^{(a)} + x_{ran} & \text{if } u_{ran} = 1, \end{cases} \tag{17}$$

where $u_{ran} \in \{0, 1\}$ follows a Bernoulli distribution with $p = 0.5$ and controls whether a pixel needs to be erased. If $u_{ran} = 0$, we set $x_{ij}^{(r)}$ as its raw value, otherwise we add it with a random value x_{ran} that follows a normal distribution $N(0, \sigma)$, with σ being the standard deviation of pixels in $\boldsymbol{X}^{(a)}$.

The random erasure mechanism of the rectifier enables us to generate multiple randomized duplicates for every adversarial example, to fine-tune the original classifier F and improve its ability to handle the rectified examples. Finally, we use the fine-tuned classifier to predict the labels of rectified adversarial examples.

So far, there remains one issue in the implementation of the rectifier: how to calculate g_{ijl} given the four types of interpreters in the ensemble detector. We follow two principles to solve this: *i*) The chosen interpreter-based sub-detector should have the same detection result with the ensemble detector; *ii*) Given an input example $\boldsymbol{X}$, the chosen interpreter-based sub-detector should have the lowest information entropy on its prediction $\hat{z} = (z_1, z_2, z_3),\ \sum_i z_i = 1$, which is calculated as

$$H^{(pred)} = -\sum_{i=1}^{3} z_i \log_2(z_i). \tag{18}$$

A lower $H^{(pred)}$ means the information provided is richer than others, which implies that the g_{ijl} adopted in Eq. (16) can offer more information to adversarial examples rectification.

5 EXPERIMENTS

5.1 Datasets and Attackers

In the experiments, we employ three popular image datasets, namely Fashion-MNIST [34], CIFAR-10 [15] and ImageNet [5].

The experiments are conducted on five extensively used and state-of-the-art attack algorithms. They are l_∞ *attackers*: FGSM [7], PGD [18], and l_2 *attackers*: Deepfool (DFool) [21], CW [4], DDN [22].

These attacks have two types of adversarial specificities, *i.e.,* untargeted attacks and targeted attacks [1]. The targeted attacks mislead the output of a classifier to a given category, while untargeted attacks mislead the output to any false category. In the experiments, we mark the untargeted version of attackers with the postfix "-U", and the targeted with "-T". FGSM and Deepfool only have untargeted versions, and others have both. We finally have eight types of different attackers.

Attackers in our experiments have three threat models:

• *Grey-box.* Attackers have knowledge about the network parameters and structure of the classifier but are not aware that the classifier is under the protection of any defence, so they generate adversarial examples that can only fool the target classifier.

• *Black-box.* Attackers have no access to the model of the target classifier, and are not aware that the classifier is under protections. They have to use adversarial examples generated for a surrogate classifier to attack the target classifier.

• *White-box.* Attackers have full knowledge about the target classifier and the defence scheme, so they can generate adversarial examples to fool the entire X-Ensemble model.

5.2 Performance of Detector

Here, we evaluate the performance of the detector in X-Ensemble. We compare our ensemble detector, namely X-Det, with three types of baselines using different feature engineering methods. They are:

• *PixelDefend (PD)* [24]. It utilizes a PixelCNN network to extract features of examples, and adopts a test statistic to decide whether an input is an adversarial example. This baseline is a representative of the detectors that use neural networks to extract features of adversarial examples.

• *TWS* [14]. It utilizes two seemingly paradoxical criteria of benign images to detect adversarial examples, with the assumption that an adversarial example cannot satisfy the two criteria. It could be considered as a type of hand-crafted feature-based detectors.

• *MDS* [16]. It utilizes Mahalanobis Distance-based Score (MDS) to measure different distributions of adversarial and benign examples in a network's middle-layer outputs. This baseline is considered as a type of distribution feature-based detectors.

Since X-Ensemble is an ensemble model, the sub-detectors are also given as baselines to evaluate the performance of our ensemble mechanism, named with their interpreters' names. We report the binary classification performance of the sub-detectors, *i.e.,* the outputs of both l_2 and l_∞ are treated as adversarial examples. Therefore, we use AUC, *i.e.,* Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC), to measure detectors' performance.

In the experiments, we use different attackers to generate adversarial examples, and mix them with the original benign ones to train and test the detectors. Our experiments employ two kinds of training methods, namely *Vaccinated Training* and *Unvaccinated Training*. In the vaccinated training, adversarial examples in the training and test sets are generated by the same kind of attackers. Oppositely, in the unvaccinated training, adversarial examples are generated by different attackers. For example, if we train a detector

Table 2: Adversarial examples detection performance for vaccinated training.

Grey-Box																		
	Fashion-MNIST									CIFAR10								
Attackers	X-Det	PD	TWS	MDS	VG	IG	GBP	LRP	ORG	X-Det	PD	TWS	MDS	VG	IG	GBP	LRP	ORG
FGSM-U	**1.00**	**1.00**	0.63	0.71	0.97	0.99	**1.00**	0.99	**1.00**	**1.00**	0.98	0.52	0.83	0.88	0.86	0.98	0.99	**1.00**
PGD-U	**1.00**	**1.00**	0.65	0.79	0.98	**1.00**	0.99	0.99	**1.00**	**0.99**	**0.99**	0.52	0.76	**0.99**	0.95	0.96	0.97	0.98
PGD-T	**1.00**	**1.00**	0.83	0.80	0.97	**1.00**	0.99	0.99	**1.00**	0.98	0.96	0.48	0.71	0.93	0.90	0.95	0.98	**1.00**
DFool-U	0.99	0.98	0.99	0.77	0.95	0.99	**1.00**	0.94	0.99	0.98	0.77	0.83	0.93	0.89	0.90	**0.99**	0.92	0.83
CW-U	0.98	0.93	0.95	0.79	0.94	0.98	**1.00**	0.98	0.96	0.98	0.78	0.90	0.93	0.90	0.89	**0.99**	0.92	0.86
CW-T	**1.00**	0.98	0.99	0.83	0.97	**1.00**	**1.00**	**1.00**	0.99	**0.99**	0.84	0.94	0.94	0.93	0.93	**0.99**	0.96	0.95
DDN-U	0.99	0.98	0.80	0.79	0.96	0.99	0.99	**1.00**	0.99	**0.99**	0.70	0.91	0.93	0.91	0.90	0.92	**0.99**	0.90
DDN-T	**1.00**	0.99	**1.00**	0.85	**1.00**	0.90	0.98	**1.00**	**1.00**	**0.99**	0.81	0.96	0.94	**0.99**	0.93	0.95	**0.99**	0.97
Black-Box																		
	Fashion-MNIST									CIFAR10								
Attackers	X-Det	PD	TWS	MDS	VG	IG	GBP	LRP	ORG	X-Det	PD	TWS	MDS	VG	IG	GBP	LRP	ORG
FGSM-U	**1.00**	0.99	0.76	0.54	**1.00**	0.98	0.99	**1.00**	**1.00**	0.98	0.99	0.66	0.93	0.88	0.92	0.99	0.99	**1.00**
PGD-U	**1.00**	0.99	0.77	0.53	**1.00**	0.98	0.99	**1.00**	**1.00**	0.97	0.98	0.57	0.59	0.76	0.80	0.91	0.98	**1.00**
PGD-T	**1.00**	0.99	0.78	0.55	**1.00**	0.97	0.99	**1.00**	**1.00**	0.99	0.99	0.72	0.59	0.78	0.83	0.92	0.96	**1.00**
DFool-U	0.94	0.93	0.81	0.52	0.85	0.94	**0.98**	0.91	0.95	0.79	0.74	0.75	0.54	0.70	**0.80**	**0.80**	**0.80**	0.60
CW-U	0.91	0.87	0.81	0.53	0.83	0.91	**0.99**	0.90	0.86	**0.82**	0.75	0.75	0.53	0.71	**0.82**	0.80	0.81	0.70
CW-T	0.97	0.96	0.80	0.52	0.91	**0.99**	0.98	0.95	0.98	**0.82**	0.77	0.76	0.53	0.80	**0.82**	**0.82**	**0.82**	0.77
DDN-U	0.88	0.86	0.80	0.52	0.82	**0.95**	0.94	0.91	0.93	0.80	0.63	0.76	0.54	0.71	0.80	**0.81**	0.80	0.76
DDN-T	0.98	0.96	0.79	0.54	0.92	0.97	**0.99**	0.96	**0.99**	0.82	0.72	0.76	0.54	0.71	0.80	0.82	0.82	**0.89**

using adversarial examples generated by the FGSM algorithm, for vaccinated training, we also need to use FGSM to attack our model for test (*i.e.,* our model has been "vaccinated" by FGSM); but for unvaccinated training, we may use DDN or other attackers for test (*i.e.,* our model has not been "vaccinated" by unknown attackers).

We first give the experimental results of vaccinated training, where we use all eight attackers to generate adversarial examples to train our detector and then fool our model using the attackers one by one. The detection performance is tested on Fashion-MNIST and CIFAR-10 datasets. The AUC scores of the detectors under grey- and black-box attacks are given in Table 2. As can be seen,

• In general, neural network-based detectors, including PixelDefend, X-Det and sub-detectors, have better performances than distribution and hand-crafted feature-based methods, indicating the neural network is more effective than traditional feature engineering approaches for extracting the features of adversarial examples.

• For the l_∞ attacks, *i.e.,* FGSM and PGD, the performance of X-Det is superior to TWS and MDS and is very close to PixelDefend. For the l_2 attackers, *i.e.,* Deepfool, CW and DDN, the performance of X-Det is significantly better than TWS, MDS and PixelDefend. Even for the cases where X-Det is not as better as PixelDefend, the AUC scores of X-Det are more than 97%, indicating pretty good performance. These verify the effectiveness of X-Det.

• X-Ensemble shows greater advantages on the CIFAR-10 dataset than on the Fashion-MNIST dataset. Since CIFAR-10 (color images) is deemed more complex than Fashion-MNIST (grayscale images), this implies that our model suits complex application scenarios.

• For most of the rows, X-Det's performance is just a little worse than the best sub-detector. In fact, the primary task of the RF ensemble in X-Det is not for improving the detection accuracy but for enhancing robustness. As the table shows, we cannot find a sub-detector that can always outperform the others, while X-Det's performance is very stable across all cases, showing the precious ability to resist hybrid attacks.

• The detection performance under black-box attacks is worse than that under grey-box attacks. In the black-box attack scenario, the detectors are trained for protecting the target classifier F but the attackers are not trained for attacking F. This mismatch causes the detection success rates of black-box underperform that of grey-box. However, since the attack success rates of black-box are also substantially lower than that of grey-box, the performance loss will not cause severe problems. In Table 2, the black-box performance is reported for successful adversarial examples (see SI for details). We can see X-Det still outperforms PixelDefned, TWS and MDS.

Table 3: AUC score of adversarial examples detection for unvaccinated training.

Grey-Box								
	Fashion-MNIST				CIFAR-10			
Attacker	X-Det	PD	l_∞-D	l_2-D	X-Det	PD	l_∞-D	l_2-D
PGD-U	**1.00**	**1.00**	**1.00**	0.90	**1.00**	0.99	**1.00**	0.39
PGD-T	**1.00**	**1.00**	0.99	0.91	**1.00**	0.99	**1.00**	0.50
CW-U	0.95	0.93	0.73	**0.97**	**0.98**	0.78	0.49	0.97
CW-T	0.98	0.98	0.84	**0.99**	**0.99**	0.84	0.49	0.98
DDN-U	0.99	0.98	0.80	**1.00**	**0.99**	0.70	0.49	0.98
DDN-T	**1.00**	**1.00**	0.93	**1.00**	**0.99**	0.81	0.49	0.98
OnePixel	**0.82**	0.61	0.59	0.75	**0.83**	0.81	0.51	0.77
Black-Box								
	Fashion-MNIST				CIFAR-10			
Attacker	X-Det	PD	l_∞-D	l_2-D	X-Det	PD	l_∞-D	l_2-D
PGD-U	**0.99**	**0.99**	0.98	0.91	0.99	0.99	**1.00**	0.70
PGD-T	**0.99**	**0.99**	0.98	0.92	0.99	0.99	**1.00**	0.78
CW-U	**0.87**	0.85	0.51	0.73	**0.80**	0.75	0.48	0.77
CW-T	**0.97**	0.93	0.78	0.88	**0.80**	0.77	0.49	0.76
DDN-U	0.85	**0.88**	0.53	0.83	**0.80**	0.63	0.49	0.75
DDN-T	**0.95**	0.98	0.84	0.90	**0.82**	0.72	0.48	0.77
OnePixel	**0.73**	0.71	0.57	0.69	**0.72**	0.70	0.51	0.69

Table 3 reports the performance of X-Ensemble using *Unvaccinated Training*. Here, the training examples of X-Ensemble are generated by two types of attackers, *i.e.,* FGSM as an l_∞ attacker

Table 4: Robustness of detectors under attacks of transferable adversarial examples.

Fashion-MNIST							
Targeted Detector	Detector Performance						Success Rate
	ORG	VG	GBP	IG	LRP	X-Det	
toORG	0.98	0.60	0.41	0.63	0.65	**0.39**	0.10
toVG	0.59	0.64	0.34	0.58	0.57	**0.33**	0.11
toGBP	0.63	0.55	0.63	0.57	0.71	**0.45**	0.10
toIG	0.66	0.69	0.37	0.98	0.93	**0.37**	0.10
CIFAR10							
Targeted Detector	Detector Performance						Success Rate
	ORG	VG	GBP	IG	LRP	X-Det	
toORG	0.98	0.56	0.53	0.55	0.57	**0.34**	0.11
toVG	0.40	0.69	0.42	0.39	0.49	**0.27**	0.11
toGBP	0.47	0.50	0.63	0.51	0.57	**0.26**	0.11
toIG	0.38	0.60	0.60	0.98	0.63	**0.28**	0.10

and DeepFool as an l_2 attacker. We use targeted and untargeted versions of PGD, CW and DDN attacks to evaluate the performance. The X-Det model is compared with three baselines: *i*) PixelDefend, which has the best performance among the baselines. *ii*) l_∞-D, which is a X-Det model using only unvaccinated training by FGSM. *iii*) l_2-D, which is a X-Det model using only unvaccinated training by DeepFool. From Table 3, we can observe that:

• l_∞-D is effective for l_∞ attackers, *i.e.,* PGD, but is invalid for l_2 attackers, *i.e.,* CW and DDN. This indicates that the difference between l_∞ and l_2 adversarial examples is very significant, while the difference between l_∞ (or l_2) adversarial examples is relatively small. The performance of X-Det is better than that of both l_∞-D and l_2-D, indicating the effectiveness of X-Det that classifies examples into three classes (l_∞, l_2 and clean) in sub-detectors.

• The last row of Table 3 gives the detection performance of l_0 adversarial examples generated by OnePixel attack [26]. As can be seen, X-Det achieves superior performances compared with the PD baseline. Note that in unvaccinated training, our model is not trained by l_0 adversarial examples, so this result further verifies the adaptability of X-Det.

To conclude, the above experiments demonstrate the effectiveness of the RF ensemble detector of X-Ensemble.

5.3 Robustness of Detector

A type of threats to a detector is the white-box attack, *i.e.,* training a detector-targeted attacker to attack the detector directly. In fact, the RF ensemble of X-Det is to defend detector-targeted attacks. To the best of our knowledge, there still exists no algorithms that can directly attack RF models. A feasible attack method is to use transferable adversarial examples [17]. Following this way, we use transferable examples targeted to sub-detectors to attack X-Det.

For each row of Table 4, the transferable adversarial examples are generated and targeted to one type of sub-detectors. For example, the first row is for the ORG sub-detector, denoted as "toORG", and we transfer the "toORG" examples to attack X-Det and the sub-detectors. LRP is not differentiable in the second-order, so we cannot generate transferable examples from the LRP sub-detector. Table 4 gives attack success rates of the transferable adversarial examples. The attack success rates are average of three types of attackers, *i.e.,* PGD-T, CW-T, DDN-T. The results show that:

• When the adversarial example generator and the test sub-detector are based on the same type of interpretation (see the diagonal line of the table), the attack success rates are significantly higher than others. Even so, the success rates of detector-targeted attacks are still significantly lower than that of classifier-targeted attacks. This suggests the interpreter-based sub-detectors are more robust than classifiers, although they are all neural network models.

• For most of cases, the attack success rate is in the range of 40%-60%. For some special cases, such as toGBP→LRP and toIG→LRP on Fashion-MNIST, it reaches 71% and 93%, respectively. This suggests that the transferable attacks indeed work for detector-targeted attacks, although the performance is mediocre.

• The attack success rates of transferable adversarial examples are less than 45% to X-Det. For all rows, X-Det achieves the smallest success rates. This indicates the robustness of X-Det compared with the sub-detectors and the effectiveness of the ensemble mechanism.

• As Table 4 shows, there are still 26%-45% adversarial examples that escape the detection of X-Det. In the last column of the table, we give the final attack success rates of the escaped adversarial examples to the classifier F, which are actually very low to about 10%. This implies that while the transferable adversarial examples can threaten the X-Ensemble's detector, it is very hard to fool both the detector and the classifier of X-Ensemble as a whole.

5.4 End-to-end Performance of X-Ensemble

Here, we evaluate the end-to-end image classification performance of X-Ensemble with the presence of four baselines. The first is PixelDefend, which can be used to detect and recover adversarial examples. The second is Total Variance Minimization (TVM), which is a type of adversarial example rectifying method [11]. The rest two are adversarial training methods based on PGD [18] and DDN [22], denoted as PGD_a and DDN_a, respectively. For the adversarial training baselines, we use adversarial examples to fine-tune the classifier, and then generate new adversarial examples to fool the fine-tuned classifier. Through iterating the training and attack process, the classifier is expected to converge to a robust state. Compared with our detector-rectifier framework, adversarial training is very time and computational resources consuming.

Table 5 gives the comparative results of X-Ensemble and the baselines. The performance is measured by the classification accuracy. The column F corresponds to the classification accuracy of adversarial examples for an unprotected classifier. As can be seen,

• The accuracy is very low for the classifier F under the grey-box attacks, and it even falls to zero under the l_2 attacks. In contrast, all the baselines can improve the classification accuracy and X-Ensemble is the best.

• For clean examples, the accuracy of F is better than others. The reason is that F is directly trained on clean examples while our rectifier and the baselines are not. Nevertheless, the accuracy of X-Ensemble is significantly higher than the baselines and is very close to that of F.

• In the black-box experiments, to ensure fair comparison under a same standard, we balance the proportion of successful and failed adversarial examples in the test set so as to fix the performance of F to 50%. The performance is calculated over the balanced test set. Again X-Ensemble significantly outperforms F and the baselines.

Table 5: Image classification accuracy of X-Ensemble and the baselines.

Grey-Box																		
	Fashion-MNIST						CIFAR-10						ImageNet					
	Our	PD	DDN_a	PGD_a	TVM	F	Our	PD	DDN_a	PGD_a	TVM	F	Our	PD	DDN_a	PGD_a	TVM	F
Clean	0.90	0.90	0.86	0.84	0.67	**0.92**	0.82	0.79	0.75	0.64	0.35	**0.86**	0.89	0.66	0.78	0.72	0.75	**0.95**
FGSM-U	**0.84**	0.75	0.82	0.82	0.49	0.56	**0.55**	0.36	0.48	0.43	0.29	0.24	**0.60**	0.47	0.49	0.47	0.36	0.44
PGD-U	**0.79**	0.64	0.80	0.81	0.57	0.27	**0.41**	0.30	0.37	0.35	0.32	0.08	**0.75**	0.70	0.38	0.47	0.66	0.02
PGD-T	**0.89**	0.86	0.84	0.87	0.53	0.66	**0.62**	0.60	0.33	0.48	0.32	0.05	**0.73**	0.66	0.29	0.51	0.70	0.00
Dfool-U	0.87	**0.88**	0.26	0.76	0.65	0.00	**0.71**	0.68	0.19	0.29	0.34	0.00	**0.75**	0.58	0.37	0.35	0.71	0.01
CW-U	0.86	**0.88**	0.70	0.73	0.66	0.00	**0.74**	0.73	0.70	0.63	0.34	0.00	**0.74**	0.64	0.50	0.53	0.71	0.00
CW-T	**0.86**	0.85	0.72	0.53	0.65	0.00	0.74	**0.75**	0.45	0.46	0.33	0.00	**0.79**	0.61	0.40	0.39	0.75	0.00
DDN-U	**0.90**	0.89	0.74	0.76	0.66	0.00	0.69	**0.74**	0.66	0.52	0.34	0.00	**0.76**	0.60	0.56	0.44	0.75	0.03
DDN-T	**0.90**	0.89	0.59	0.64	0.65	0.00	0.71	**0.75**	0.53	0.43	0.34	0.00	**0.79**	0.60	0.50	0.39	0.74	0.00
Black-Box																		
	Fashion-MNIST						CIFAR-10						ImageNet					
	Our	PD	DDN_a	PGD_a	TVM	F	Our	PD	DDN_a	PGD_a	TVM	F	Our	PD	DDN_a	PGD_a	TVM	F
Clean	0.90	0.90	0.86	0.84	0.67	**0.92**	0.82	0.79	0.75	0.64	0.35	**0.86**	0.89	0.66	0.78	0.72	0.75	**0.95**
FGSM-U	**0.72**	0.70	0.68	0.71	0.46	0.50	**0.43**	0.27	0.41	0.41	0.31	0.50	**0.60**	0.49	0.51	0.48	0.54	0.50
PGD-U	0.78	0.80	0.77	**0.82**	0.48	0.50	0.66	**0.70**	0.68	0.58	0.31	0.50	**0.63**	0.61	0.58	0.50	0.51	0.50
PGD-T	0.79	0.78	0.74	**0.81**	0.43	0.50	0.63	**0.73**	0.70	0.59	0.30	0.50	**0.65**	0.52	0.55	0.49	0.50	0.50
Dfool-U	**0.87**	0.86	0.84	**0.87**	0.48	0.50	**0.78**	0.76	0.71	0.61	0.29	0.50	**0.67**	0.60	0.58	0.51	0.43	0.50
CW-U	**0.88**	0.87	0.84	0.87	0.48	0.50	**0.78**	0.75	0.71	0.61	0.30	0.50	**0.65**	0.58	0.51	0.51	0.46	0.50
CW-T	**0.87**	**0.87**	0.84	0.85	0.53	0.50	**0.77**	0.75	0.71	0.60	0.29	0.50	**0.67**	0.45	0.56	0.51	0.44	0.50
DDN-U	**0.88**	0.87	0.84	0.87	0.50	0.50	**0.77**	0.76	0.72	0.61	0.30	0.50	**0.67**	0.43	0.57	0.50	0.45	0.50
DDN-T	**0.88**	0.87	0.84	0.87	0.49	0.50	**0.77**	0.74	0.71	0.60	0.28	0.50	**0.68**	0.36	0.53	0.46	0.41	0.50

Table 6: Classification accuracy of X-Ensemble under white-box attacks.

X-Ensemble			
	Fashion-MNIST	CIFAR-10	ImageNet
PGD-T	0.87	0.67	0.72
CW-T	0.90	0.69	0.83
DDN-T	0.90	0.71	0.78

Finally, we test the performance of X-Ensemble via white-box attacks. Since there exists no white-box algorithms that can directly attack X-Ensemble, we propose a new attack approach based on Ref. [3]. Specifically, we concatenate all sub-detectors and the classifier F as a new function and generate targeted adversarial examples to mislead the new function. The effect of the white-box attacker is to require all sub-detectors to give a benign prediction and require F to give an incorrect prediction. If the white-box attack succeeds, an adversarial example can fool the RF ensemble by fooling all sub-detectors, and the classifier F can also be misled at the same time. Table 6 gives the classification accuracy of X-Ensemble under the white-box attack. There are no untargeted attackers since the white-box approach can be only implemented using targeted algorithms. As the table shows, our model achieves very high classification accuracy by beating most of the white-box attacks. The reason is that it is very hard to search a perturbation that can fool all sub-detectors and the classifier F simultaneously.

6 RELATED WORKS

Adversary Defense. In the literature, there are two types of approaches to mitigate the threats of adversarial examples. The first is to directly strengthen classifiers against adversaries. The models along this line include three categories: *i*) Adversarial Training [18, 22], which adds adversarial examples to augment training data and ensure a robust classifier. It usually consumes exorbitant time and computational resources by iteratively generating adversarial examples and optimizing classifiers, which makes it difficult to suit real-world large-scale applications. *ii*) Modifying Model Architectures [2], which manipulates model gradients to be nondifferentiable in inference. For instance, Decision-making Tree (DT) and Random Forest (RF) are ideal nondifferentiable models falling in this category, but they also suffer from relatively poor performance in image classification. In our X-Ensemble model, we propose an RF-based ensemble framework to combine sub-detectors, which fully exploited the advantage of nondifferentiable models for adversarial examples detection. *iii*) Rectifying Adversarial Examples, which removes input perturbations to some extent so that the input can be classified correctly, such as TVM [11]. However, traditional example rectifying methods do not consider the influences of pixels to attack effect when rectifying adversarial image pixels. Our X-Ensemble model, in contrast, rectifies image pixels with the guidance of interpretation maps to ensure higher performance.

The second type of approaches for adversary defense focuses on the detection of adversarial examples. Models along this line mainly include using statistical methods to verify hand-crafted features of images and the network parameters [14, 16, 24], and using neural networks to judge whether an input is clean or adversarial [19]. While neural network models are typically deemed more powerful in detection, they might suffer from adversarial attacks themselves. Our X-Ensemble model solves this problem by adopting an RF-based ensemble framework. More importantly, X-Ensemble plays a dual

role as a detector-rectifier by establishing a novel joint framework for adversarial examples detection and rectification.

Neural Networks Interpretation. There is a longstanding voice arguing that deep learning based systems are unexplainable black boxes and therefore cannot be used in crucial application domains such as medicine, investment and military. Many interpretation methods have been proposed to solve the black-box problem of DNNs, including time series analysis [31], intermediate representations [9], sensitivity analysis [27], and among others. As reported by Ref. [6], the sensitivity analysis methods have close connection with adversarial robustness of neural networks. Our X-Ensemble model utilizes this information both in the detection part and rectification part to defend adversarial attacks.

7 CONCLUSION

In this paper, we proposed X-Ensemble, a detection-rectification pipelined ensemble classifier for high-performance adversary defense. X-Ensemble leverages target classifiers' interpretation information of various types and designs interpreter-based multiple sub-detectors for accurate detection of adversarial examples. X-Ensemble further employs the random forests model to form an ensemble detector upon sub-detectors, which not only improves the robustness of detectors against hybrid attacks but also avoids targeted attacks to detectors owing to the nondifferentiable property of a random forest classifier. We conducted extensive experiments to manifest the advantages of X-Ensemble to competitive baseline methods under different attacks and rich scenarios.

ACKNOWLEDGMENTS

This work was supported by the National Key R&D Program of China (Grant No. 2019YFB2102100). Dr. Jingyuan Wang's work was partially supported by the National Natural Science Foundation of China (Grant No.61572059), the Fundamental Research Funds for the Central Universities (Grant No. YWF-20-BJ-J-839) and the CCF-DiDi Gaia Collaborative Research Funds for Young Scholars. Dr. Junjie Wu's work was partially supported by the National Special Program on Innovation Methodologies (Grant No. SQ2019IM4910001), and the National Natural Science Foundation of China (Grant No. 71531001, 71725002, U1636210, 71490723).

REFERENCES

[1] Naveed Akhtar and Ajmal Mian. 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access* 6 (2018), 14410–14430.
[2] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian J. Goodfellow. 2018. Thermometer Encoding: One Hot Way To Resist Adversarial Examples. In *ICLR'18*.
[3] Nicholas Carlini and David Wagner. 2017. Adversarial examples are not easily detected: Bypassing ten detection methods. In *10th ACM Workshop on Artificial Intelligence and Security*. ACM, 3–14.
[4] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *IEEE SP'17*. IEEE, 39–57.
[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR'09*.
[6] Christian Etmann, Sebastian Lunz, Peter Maass, and Carola Schoenlieb. 2019. On the Connection Between Adversarial Robustness and Saliency Map Interpretability. In *ICML*. 1823–1832.
[7] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *ICLR*.
[8] Jindong Gu and Volker Tresp. 2019. Saliency methods for explaining adversarial attacks. *arXiv preprint arXiv:1908.08413* (2019).
[9] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2019. A survey of methods for explaining black box models. *ACM CSUR* 51, 5 (2019), 93.
[10] Chuan Guo, Mayank Rana, Moustapha Cissé, and Laurens van der Maaten. 2017. Countering Adversarial Images using Input Transformations. *CoRR* abs/1711.00117 (2017). arXiv:1711.00117 http://arxiv.org/abs/1711.00117
[11] Chuan Guo, Mayank Rana, Moustapha Cissé, and Laurens van der Maaten. 2018. Countering Adversarial Images using Input Transformations. In *ICLR'18*.
[12] Suiming Guo, Chao Chen, Jingyuan Wang, Yaxiao Liu, Xu Ke, Zhiwen Yu, Daqing Zhang, and Dah-Ming Chiu. 2019. ROD-Revenue: Seeking Strategies Analysis and Revenue Prediction in Ride-on-demand Service Using Multi-source Urban Data. *IEEE Transactions on Mobile Computing* (2019).
[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR'*. 770–778. https://doi.org/10.1109/CVPR.2016.90
[14] Shengyuan Hu, Tao Yu, Chuan Guo, Wei-Lun Chao, and Kilian Q Weinberger. 2019. A New Defense Against Adversarial Images: Turning a Weakness into a Strength. In *NeurIPS'19*. 1633–1644.
[15] Alex Krizhevsky et al. 2009. *Learning multiple layers of features from tiny images*. Technical Report.
[16] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *NeurIPS'18*. 7167–7177.
[17] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2016. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770* (2016).
[18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *ICLR*.
[19] Dongyu Meng and Hao Chen. 2017. MagNet: A Two-Pronged Defense against Adversarial Examples. In *SIGSAC'17*. 135–147.
[20] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. 2019. Layer-wise relevance propagation: an overview. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 193–209.
[21] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*. 2574–2582.
[22] Jérôme Rony, Luiz G Hafemann, Luiz S Oliveira, Ismail Ben Ayed, Robert Sabourin, and Eric Granger. 2019. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In *CVPR*. 4322–4330.
[23] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR'15*.
[24] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. 2017. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*.
[25] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. 2014. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806* (2014).
[26] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One pixel attack for fooling deep neural networks. *IEEE T-EC* 23, 5 (2019), 828–841.
[27] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *ICML*. JMLR. org, 3319–3328.
[28] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. arXiv:1312.6199 [cs.CV]
[29] Jingyuan Wang, Chao Chen, Junjie Wu, and Zhang Xiong. 2017. No longer sleeping with a bomb: a duet system for protecting urban safety from dangerous goods. In *Proceedings of the 23rd ACM SIGKDD*. ACM, 1673–1681.
[30] Jingyuan Wang, Qian Gu, Junjie Wu, Guannan Liu, and Zhang Xiong. 2016. Traffic speed prediction and congestion source exploration: A deep learning method. In *2016 IEEE 16th ICDM*. IEEE, 499–508.
[31] Jingyuan Wang, Ze Wang, Jianfeng Li, and Junjie Wu. 2018. Multilevel wavelet decomposition network for interpretable time series analysis. In *Proceedings of the 24th ACM SIGKDD*. ACM, 2437–2446.
[32] Jingyuan Wang, Ning Wu, Wayne Xin Zhao, Fanzhang Peng, and Xin Lin. 2019. Empowering a* search algorithms with neural networks for personalized route recommendation. In *Proceedings of the 25th ACM SIGKDD*. 539–547.
[33] Jingyuan Wang, Yang Zhang, Ke Tang, Junjie Wu, and Zhang Xiong. 2019. AlphaStock: A Buying-Winners-and-Selling-Losers Investment Strategy using Interpretable Deep Reinforcement Attention Networks. In *SIGKDD*. 1900–1908.
[34] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2018. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. (2018).
[35] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. 2019. Adversarial examples: Attacks and defenses for deep learning. *IEEE T-NNLS* (2019).
[36] Wei Emma Zhang, Quan Z Sheng, AHOUD Alhazmi, and CHENLIANG LI. 2019. Adversarial attacks on deep learning models in natural language processing: A survey. *arXiv preprint arXiv:1901.06796* (2019).

A SUPPLEMENTAL MATERIALS

A.1 Datasets

We use the whole dataset of Fashion-MNIST [34] and CIFAR-10 [15], but select images of 20 object classes from ImageNet [5] as a subset for our experiments. All pixels are projected into [0, 1].

A.2 Model Structures

Here we report the model structures in our experiments. For Fashion-MNIST and CIFAR-10, VGG11[23] structure is applied to the target classifier, detector and rectifier in X-Ensemble, while VGG16[23] is for ImageNet experiments. The black-box classifiers for Fashion-MNIST , CIFAR-10 and ImageNet are contructed by CWnet[4], Wide-ResNet[22] and ResNet152[13] respectively.

Since it is very computational resource consuming to train a VGG16 and an ResNet152 model for ImageNet, we adopt the pre-trained VGG16 as the target classifier and the pre-trainded ResNet152 as the black-box classifier. When using these models for ImageNet, we select the output logits of those 20 chosen classes out of the 1000 categaries and then compute the probabilities for them with $softmax$ function. In this way, the pre-trained VGG16 and ResNet152 both have more than 95% accuracy on the ImageNet subset.

A.3 Setting of Attackers

The codes of all attack methods in our experiments are implemented by AdverTorch [1] v0.2.

The ϵ in FGSM and PGD is to constrain the L_∞ perturbation of adversarial examples; the α in PGD and the learning rate (lr) in CW control the step size in each iteration when searching the adversarial perturbation; the iteration (I) limits how many times that an iterative attack can compute. $\epsilon = 0.031$, $\alpha = 0.0078$ and $lr = 0.01$ for all the three datasets. I is set to 100 for Fashion-MNIST and CIFAR-10, 50 for ImageNet.Other parameters use their default values in AdverTorch.

OnePixel attack is only evaluated in unvaccinated training. We set the pixel numbers to 3, which means the attack can only modify 3 pixel values in an image. And its iteration is set to 30, with 100 candidates in each iteration.

A.4 Interpretation Methods

VG, GBP and IG methods are implemented on our own and use LRP code from [20]. The integrated step in IG is set to 50. Note that LRP from [20] cannot be applied on ResNet152 directly. So we remove the LRP detector only for ImageNet.

A.5 Details of Rectifier

We use Alg. 1 to compute rectified images on adversarial examples and the hyperparameter α is set 0.6, 0.9, 0.5 for Fashion-MNIST, CIFAR-10 and ImageNet respectively.And we find that a rectifier tuned with rectified images of DDN-T mixed with clean images has a better performance.

A.6 Baselines

In this section, we briefly introduce the sources of four baselines used in our work.

• **PD.** The code is from [24]. When purifying images, ϵ is set to 0.125 for Fashion-MNIST and 0.063 for both CIFAR-10 and ImageNet.

Algorithm 1 Rectified Image For Tuning Rectifier

Variables: $\{D_1, ..., D_j\}$ are the sub-detectors that predict an input image x as an adversarial one, $\{R_1, ..., R_j\}$ are the interpreting methods corresponding to $\{D_1, ..., D_j\}$ respectively, $\alpha \in (0, 1)$ is a threshold parameter, $rand()$ returns a random value in $[0, 1]$, and σ is the variance of pixel values in x.

for $k = 1$ to j **do**
  $E_k \leftarrow Entropy(D_k(x))$
end for
$R \leftarrow R_i$ where $i = argmin(E_1, ..., E_j)$
$g \leftarrow R(x)$
$thres \leftarrow \alpha * (\max(g) - \min(g)) + \min(g)$
for ixel (i, j) in x **do**
  if $g_{i,j} > thres$ **and** $rand() > 0.5$ **then**
    $x_{i,j} \leftarrow x_{i,j} + Normal(0, \sigma)$
  end if
end for
return x

The pretrained model of PixelCNN for CIFAR-10 is from $openai$[2], and we train PixelCNN on Fashion-MNIST and ImageNet with code from $openai$ [2].

• **TWS.** The code is from [14]. We set the parameters n_radius = 0.01, targeted_lr = 0.0005, t_radius = 0.5, u_radius = 0.5 and untargeted_lr = 1.

• **MHL.** The code is from [16]. The magnitude of noise starts from 0.05 to 0.3 with an interval of 0.05 to compute its AUC.

• **TVM.** The code is from [10]. We set TVM_WEIGHT = 0.03, PIXEL_DROP_RATE = 0.5, TVM_METHOD = 'tvl2'.

• **Adversarial Training** Adversarial training is computational expensive. To speed up the training process while achieving a good accuracy, we use adversarial data to fine tune the original classifier with 15 epochs.

A.7 White-box Attacker for X-Ensemble

In this section, we show the details of the end-to-end white-box evaluation in our experiments.

Carlini *et al.* [3] proposed a variant of CW attack to combine a classifier and a neural network detector into a new classifier G with L+1 classes, whose $(L+1)^{th}$ class is for adversarial inputs. In this way, an attacker can directly attack G to break F and D at the same time. G is defined as,

$$G(x)_i = \begin{cases} F(x)_i & \text{if } i \le L \\ (D(x)+1) \cdot max_j F(x)_j & \text{if } i = L+1 \end{cases} \tag{19}$$

where if x is adversarial then $D(x) > 0$, so we have $G(x)_{L+1} > max(F(x))$ and $argmax_i G(x)_i = L+1$; if x is clean then $D(x) < 0$, so we have $argmax_i G(x)_i = argmax_i F(x)_i$. An attacker will optimize this joint objective (target $t \neq l$ and $t \neq L+1$) to construct adversarial examples on G.

In our experiment, an attacker needs to fool all the detectors in X-Ensemble. So we modify G into an L+4 classifier as,

$$G(x)_i = \begin{cases} F(x)_i & \text{if } i \le L \\ (D_k(x)+1) \cdot max_j F(x)_j & \text{if } i = L+k \end{cases} \tag{20}$$

where D_k is one part of X-DET and $k = 1, 2, 3, 4$. Notice that here we remove the LRP detector since it is not differentiable in the second-order. So that a targeted attacker can generate examples on the classifier and the detectors to perform a white-box attack.

[1] https://github.com/BorealisAI/advertorch

[2] https://github.com/openai/pixel-cnn