Towards Certifying the Asymmetric Robustness for Neural Networks: Quantification and Applications

Changjiang Li, Shouling Ji, Haiqin Weng, Bo Li, Jie Shi, Raheem Beyah, Shanqing Guo, Zonghui Wang, Ting Wang

Abstract—One intriguing property of deep neural networks (DNNs) is their vulnerability to adversarial examples – those maliciously crafted inputs that deceive target DNNs. While a plethora of defenses have been proposed to mitigate the threats of adversarial examples, they are often penetrated or circumvented by even stronger attacks. To end the constant arms race between attackers and defenders, significant efforts have been devoted to providing certifiable robustness bounds for DNNs, which ensures that for a given input its vicinity does not admit any adversarial instances. Yet, most prior works focus on the case of symmetric vicinities (e.g., a hyperrectangle centered at a given input), while ignoring the inherent heterogeneity of perturbation direction (e.g., the input is more vulnerable along a particular perturbation direction).

To bridge the gap, in this paper, we propose the concept of *asymmetric robustness* to account for the inherent heterogeneity of perturbation directions, and present `Amoeba`, an efficient certification framework for asymmetric robustness. Through extensive empirical evaluation on state-of-the-art DNNs and benchmark datasets, we show that compared with its symmetric counterpart, the asymmetric robustness bound of a given input describes its local geometric properties in a more precise manner, which enables use cases including (i) modeling stronger adversarial threats, (ii) interpreting DNN predictions, and makes it a more practical definition of certifiable robustness for security-sensitive domains.

Index Terms—Robustness Certification, Deep Learning Security, Adversarial Example.

1 Introduction

Despite their widespread use in a variety of applications (e.g., captcha recognition [1], image classification [2], facial recognition [3], and speech recognition [4]), deep neural networks (DNNs) are inherently susceptible to adversarial examples [5], which are maliciously crafted inputs to deceive target DNNs. Such vulnerabilities have raised significant concerns about the use of DNNs in security-critical tasks. For example, the adversary may exploit adversarial examples to manipulate DNN-based autonomous driving systems, threatening passenger safety [6].

To mitigate the threats of adversarial examples, a plethora of defense mechanisms have been proposed, including defensive distillation [7], adversarial training [8], [9], [10], and automated detection [11]. However, due to their lack of theoretical robustness guarantees, most existing defenses are often penetrated or circumvented by even stronger adversarial attacks [12], [13], resulting in a constant arms race between the attackers and defenders.

Motivated by this, intensive research has been devoted to providing certifiable robustness [14], [15], [16], [17], which ensures that the bounded proximity of given inputs does not admit any adversarial examples. By enforcing such bounds during training, DNNs are bestowed with provable robustness against any norm-bounded attacks [18], [19], [20], [21]. Early works in this direction focus on certifying uniform robustness, which consider all features equally vulnerable. To account for the varying vulnerabilities of different features to adversarial perturbation, Liu *et al.* [22] introduced the concept of non-uniform robustness. Yet, both uniform and non-uniform robustness assumes symmetric vicinity definitions (e.g., a hyperrectangle centered at a given input), while ignoring the inherent heterogeneity of perturbation directions (e.g., the input is more vulnerable along a particular perturbation direction), resulting in suboptimal robustness bounds.

To illustrate the drawbacks of symmetric robustness, consider the following simplified example:

$$f(x) = \begin{cases} 1 & (0.5 < x \leq 1) \\ 0 & (0 < x \leq 0.5) \end{cases} \tag{1}$$

where the input x is a scalar bounded by $[0, 1]$ and $f(x)$ is the classification model. Figure 1 shows an example of the certified robustness spaces of data example $x = 0.3$, where no adversarial example could be admitted. In this example, the symmetric robustness space is $[0.1, 0.5]$ since the left and right bounds should be equal (both are 0.2). In contrast, the asymmetric robustness space is $[0, 0.5]$ since the left and right are allowed to be unequal (the left is 0.3 while the right is 0.2)

The above comparison shows evidently that asymmetric robustness bounds tend to provide more precise estimates of the

- *S. Ji and Z. Wang are the corresponding authors.*
- *C. Li is with the College of Computer Science and Technology at Zhejiang University, Hangzhou, Zhejiang, 310027, China, and also with the College of Information Science and Technology, Pennsylvania State University, University Park, PA 16802 USA. Email: changjiang.li@psu.edu.*
- *S. Ji is with the College of Computer Science and Technology, Zhejiang University, and the Binjiang Institute of Zhejiang University, Hangzhou, Zhejiang 310027, China. Email: sji@zju.edu.cn*
- *H. Weng is with the Ant Group, Hangzhou, China. Email: haiqin.wenghaiqin@antgroup.com.*
- *B. Li is with Illinois at Urbana-Champaign, Champaign, USA. Email: lbo@illinois.edu.*
- *J. Shi is with Huawei International, Singapore. Email: shi.jie1@huawei.com.*
- *R. Beyah is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332. Email: raheem.beyah@ece.gatech.edu.*
- *S. Guo is with the School of Cyber Science and Technology at Shandong University, Qingdao, Shandong, 266237, China. Email: guoshanqing@sdu.edu.cn*
- *Z. Wang is with the College of Computer Science and Technology at Zhejiang University, Hangzhou, Zhejiang, 310027, China. Email: zhwang@zju.edu.cn*
- *T. Wang is with the College of Information Science and Technology, Pennsylvania State University, University Park, PA 16802 USA. E-mail: inbox.ting@gmail.com.*

** This work was partially conducted when C. Li was at Zhejiang University.*

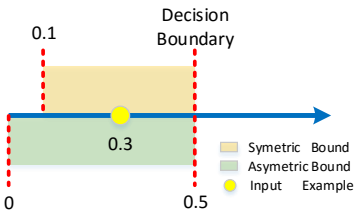


Fig. 1. Symmetric versus asymmetric robustness bounds.

robustness spaces surrounding given inputs. However, adopting asymmetric robustness entails the non-trivial challenges of (i) easy-to-certify formalization of asymmetric robustness and (ii) efficient certification for diverse DNNs. To our best knowledge, most existing certification methods (e.g., non-uniform robustness [22]) are only applicable to fully-connected neural networks (FCNNs), while approximating alternative constructs (e.g., convolutional layers) using fully-connected layers with sparse weight matrices, without accounting for the unique characteristics of DNN architectures (e.g., convolutional neural networks (CNNs)).

Our Work. To tackle the above challenges, in this paper, we present `Amoeba`, the first framework for certifying asymmetric robustness bounds. Specifically, `Amoeba` highlights in three aspects: *formalization*, *relaxation* and *optimization*.

(1) **Formalization**. Accounting for the inherent heterogeneity of perturbation directions, `Amoeba` employs independent variables to represent the bounds along different directions, leading to a novel optimization formulation; further, it unifies varying DNNs architectures (e.g., FCNNs and CNNs) in the certification formalization, leading to optimization constraints that natively capture their unique characteristics.

(2) **Relaxation**. This optimization formulation is computationally expensive. Therefore, to render it computationally feasible, `Amoeba` relaxes the inequality constraints with their equality counterparts. Specifically, through bounding the output of each layer in a layer-wise manner, it computes the overall bounds of the DNN output, which are then used to relax the constrained optimization.

(3) **Optimization**. To solve the relaxed optimization problem, `Amoeba` further adopts a customized augmented Lagrangian method and efficiently computes the asymmetric robustness bounds of given inputs, which provide quantitative robustness measures with respect to both perturbation direction and magnitude.

To demonstrate the significance and utility of asymmetric robustness, we conduct extensive empirical evaluation of `Amoeba` on widely used DNNs and benchmark datasets. We show that compared with its symmetric counterpart, asymmetric robustness is able to describe the local geometric properties of given inputs in a much more precise manner, enabling various security-related use cases, including modeling stronger adversarial threats, explaining DNN predictions, and exploring transferable adversarial examples.

Our main contributions are summarized as follows.

- ***Quantification.*** To our best knowledge, we are the first to propose the concept of asymmetric robustness. In contrast of symmetric robustness in existing works, asymmetric robustness provides more precise quantitative robustness measures with respect to both adversarial perturbation direction and magnitude.
- ***Framework.*** We design and implement `Amoeba`, a first-of-its-kind framework that efficiently certifies asymmetric robustness. With both empirical and analytical evidence, we show that `Amoeba` provides much tighter robustness bounds compared with previous works.
- ***Application.*** We apply `Amoeba` to two security-related use cases including (i) modeling stronger adversarial threats; (ii) explaining DNNs predictions. The experimental results show that asymmetric robustness is more effective and practical than its symmetric counterpart for these use cases. These findings lead to several promising directions for further research. For instance, asymmetric robustness-guided adversarial training gives rise to DNNs with better robustness and utility.

2 Background

2.1 Adversarial Learning

A neural network is a function f_θ that can accept an input $x \in \mathbb{R}^n$ and output a vector $f_\theta(x) \in \mathbb{R}^K$. The neural network will classify the input example x as $C(x) = \arg\max_{0 \le j < K} (f_\theta(x)_j)$. Recently, Szegedy et al. [5] found that neural networks can be easily deceived by adversarial examples. Formally, given an input example x with the ground truth label t, the model can correctly classify x as $C(x) = t$. Then, x' is an adversarial example generated from x with respect to the target model f_θ such that $C(x') \neq t$, and x', x are close according to some distance metric and a smaller distance implies a better utility of the adversarial example.

To measure the utility of adversarial examples, the l_p ($p = 0, 1, 2, \infty$) distance is widely used. Taking x and x' as an example, the l_p distance is defined as $\|x - x'\|_p$, where $\|v\|_p = (\sum_{i=1}^{n}(|v_i|^p))^{\frac{1}{p}}$. Intuitively, l_0 measures the number of features with non-zero perturbations in x', l_1 measures the sum of the absolute value of the feature perturbation in x', l_2 measures the Euclidean distance between x and x', and l_∞ measures the maximum feature perturbation in x'.

In adversarial learning, *transferability* is also an important characteristic of adversarial examples, which characterizes the capability of the adversarial examples generated under one model causing the misclassification of another model.

2.2 Adversarial Attacks

In the context of adversarial learning, adversarial attacks aim to efficiently and effectively generate adversarial examples with high evasion rate as well as high utility. Below, we introduce four representative adversarial attacks, while deferring more discussions in Section 7.

FGSM is a gradient-based adversarial attack [23], which first calculates the gradient of the loss function with respect to the input examples, and then utilizes the gradient information to generate adversarial examples with the goal of increasing the value of the loss function. Specifically, FGSM can be formulated as:

$$x' = x + \epsilon sign(\nabla_x L(f_\theta(x), t)),$$

where $L(,)$ denotes the loss function with respect to the output of $f_\theta(x)$ and the ground truth label t, and $sign$ is a function of extracting the sign of the input. The parameter ϵ controls the perturbation magnitude, and FGSM will disturb all the features with magnitude ϵ according to the sign of the gradient. FGSM is an efficient attack, which can generate a large number of adversarial examples in a short time.

PGD [10] is a variant of FGSM. Unlike FGSM, which disturbs a clean example using only one step, PGD disturbs a clean example

several times in the direction of gradient-sign with a smaller step size. It iteratively computes the formulation:

$$x'_{(i+1)} = Clip_{\epsilon}(x'_{(i)} + \alpha sign(\nabla_x L(f_\theta(x), t))),$$

where $x'_{(i)}$ denotes the perturbed example at the i-th iteration, $Clip_{\epsilon}()$ is a function that controls the perturbation magnitude of each feature of the example not greater than ϵ and α determines the step size. PGD starts with $x'_{(0)} = x$ and runs for n iterations, which can usually generate better adversarial examples than FGSM [10]. The above formulation is derived under the l_∞ norm. Generally speaking, l_p-based PGD takes each gradient step in the direction of the greatest loss and projects the perturbation back into the l_p-norm ball until converge or the maximum iteration n reached.

The C&W attack is an optimization-based attack [13], and includes three versions that generate adversarial examples by limiting the l_0, l_2, and l_∞ distance, respectively. These generated adversarial examples can reach a very high evasion rate with imperceptible perturbations in many scenarios [13]. The l_p-based C&W attack can be formulated as an optimization problem:

$$\min \|x' - x\|_p^2 + c \cdot g(x')$$

with the objective function g defined as:

$$g = \max(\max\{f_\theta(x')_i : i \neq t\} - f_\theta(x')_t, -\kappa),$$

and the confidence of the output is controlled by κ. Then, the adversary can effectively generate adversarial examples by performing gradient descent on g.

The UAP (Universal Adversarial Perturbations) attack is an example-agnostic attack [24], which generates a universal adversarial perturbation that fools the neural network on most natural examples. Specifically, the UAP attack seeks an adversarial perturbation δ such that:

$$f_\theta(x+\delta) \neq f_\theta(x) \text{ for most } x \sim D,$$
$$\text{subject to}$$
$$\|\delta\|_p \leq \epsilon$$
$$\mathbb{P}_{x\sim D}(f_\theta(x+\delta) \neq f_\theta(x)) \geq \xi,$$

where D denotes the distribution of the input examples, ϵ controls the magnitude of the perturbation, and ξ denotes the desired evasion rate for the examples drawn from D. The UAP attack focuses on finding a universal adversarial perturbation across different examples sampled from the data distribution D, and it will run n iterations or exit ahead of schedule after reaching the desired evasion rate ξ.

2.3 Adversarial Training

Adversarial training is a defense method first proposed by Goodfellow et al. [8], which aims to improve the robustness through training the model with the augmented dataset that contains adversarial examples. The objective can be written as a min-max problem:

$$\min_\theta \max_{x':G(x,x')\leq\delta} L(f_\theta(x), t),$$

where G is a distance function and δ is the upper limit of the distance. The inner maximization problem is to find an adversarial example within a given distance δ to maximize the loss function. The outer minimization problem is to optimize the model parameters to minimize the loss of the adversarial example. Recently, many adversarial training methods are proposed, e.g., ensemble adversarial training [9], and PGD-based adversarial training [10].

2.4 Robustness Certification

Robustness certification aims to find the largest certified region around a data point, where there exists no adversarial instances. Formally, given a K-classification task, a DNN model $f_\theta : R^n \to \mathbb{R}^K$ and a benign input example $x \in R^n$, the goal of robustness certification is to find the largest vicinity Ω around x such that $f(x) = f(x'), \forall x' \in \Omega$. There are two main criteria to evaluate certification approaches: soundness and completeness. Soundness measures whether the certification has false negatives or not. The solution of the above problem can provide a soundness guarantee. Completeness measures whether the certification has false positives or not. Existing robustness certification techniques generally can be divided into two categories: *complete methods*, which do not have false positives but are computationally expensive, and *incomplete methods*, which have false positives but are computation friendly and easy to scale on larger dataset or models.

3 Methodology

In this section, we first give the threat model. Then, we introduce `Amoeba` for certifying the asymmetric robustness bounds of DNNs. The asymmetric robustness bounds logically contain both the right and left bounds, representing the model's resistance to adversarial perturbations in positive and negative directions, respectively.

3.1 Threat Model

Since this research is focusing on analyzing the robustness of a neural network model, without explicit specification, our discussion is conducted in the white-box setting (an adversary can obtain the full knowledge of the target model).

3.2 Design of `Amoeba`

3.2.1 Formalization

We define a general DNN consisting of N convolutional layers and M fully-connected layers as $f_\theta : \mathbb{R}^n \to \mathbb{R}^K$. f_θ maps an n-dimensional input to K outputs corresponding to K classes (in the formalization, we ignore the max pooling layer just for notational convenience):

$$\begin{aligned} \mathbf{z}_{(i+1)} &= Conv(\hat{\mathbf{z}}_{(i)}, \mathbf{W}_{c(i)}) + \mathbf{b}_c(i), i = 1, ..., N \\ \mathbf{z}_{(i+1)} &= \mathbf{W}_{(i)}\hat{\mathbf{z}}_{(i)} + \mathbf{b}_{(i)}, i = N+1, ..., N+M-1 \\ \hat{\mathbf{z}}_{(i)} &= \sigma(\mathbf{z}_{(i)}), i = 2, ..., N+M-1 \end{aligned} \tag{2}$$

where $\hat{\mathbf{z}}_{(1)} = \mathbf{x}$ is the input example and $f_\theta(x) = \mathbf{z}_{(N+M)}$ is the output, $\mathbf{W}_{c(i)}$ and $\mathbf{b}_{c(i)}$ denote the parameters of the i-th convolutional layer, $\mathbf{W}_{(i)}$ and $\mathbf{b}_{(i)}$ denote the parameters of the i-th fully-connected layer, $Conv$ represents the convolution operation, and σ is a monotone activation function. For simplicity, we use $\theta = \{\mathbf{W}_{c(i)}, \mathbf{b}_{c(i)}\}_{i=1}^{N} \cup \{\mathbf{W}_{(i)}, \mathbf{b}_{(i)}\}_{i=(N+1)}^{(N+M-1)}$ to denote the parameter set and n_i to represent the number of the output features of the DNN's i-th layer. Note that n_i represents the number of the output features after flattening if the i-th layer is a convolutional layer.

Based on the above general DNN, we further define the problem of certifying the asymmetric robustness bounds for a given neural network and input examples. Following [22], we denote the asymmetric robustness bound of an input example $\mathbf{x}$ as $\mathbf{\Omega}^{(p)}_{\boldsymbol{\epsilon_1},\boldsymbol{\epsilon_2}}(\mathbf{x})$. $\mathbf{\Omega}^{(p)}_{\boldsymbol{\epsilon_1},\boldsymbol{\epsilon_2}}(\mathbf{x})$ can be defined as an equation based on the l_p-norm: $\mathbf{\Omega}^{(p)}_{\boldsymbol{\epsilon_1},\boldsymbol{\epsilon_2}}(x) = \{\mathbf{x}+\boldsymbol{\epsilon_1}\odot\min(\mathbf{t},\mathbf{0})+\boldsymbol{\epsilon_2}\odot\max(\mathbf{t},\mathbf{0}) | \|\mathbf{t}\|_p \leq 1\}$, with $\mathbf{x}, \mathbf{t} \in \mathbb{R}^{n_1}$ and $\boldsymbol{\epsilon_1}, \boldsymbol{\epsilon_2} \in \mathbb{R}_+^{n_1}$. $\boldsymbol{\epsilon_1}$ and $\boldsymbol{\epsilon_2}$ denote the left and

right bounds respectively, min() and max() denote the element-wise minimum and maximum respectively, and $\odot$ is an element-wise product operation. In this paper, we mainly focus on the l_∞-norm for convenient discussion. Hence, we use $\mathbf{\Omega}_{\boldsymbol{\epsilon_1},\boldsymbol{\epsilon_2}}(\mathbf{x})$ to replace $\mathbf{\Omega}^{(\infty)}_{\boldsymbol{\epsilon_1},\boldsymbol{\epsilon_2}}(\mathbf{x})$ without further explanation. Note that our analysis can be easily extended to the setting of other norms.

Now, we seek to find the maximum robustness space represented by the asymmetric bound $\mathbf{\Omega}_{\boldsymbol{\epsilon_1},\boldsymbol{\epsilon_2}}(\mathbf{x})$. Formally, we aim to solve the below problem.

Problem 1. *Given a DNN with parameter set θ, and an input example $\mathbf{x}$ with ground truth label $t \in \{1, 2, ..., K\}$, the problem of obtaining the maximum asymmetric robustness bound $\mathbf{\Omega}_{\boldsymbol{\epsilon_1},\boldsymbol{\epsilon_2}}(\mathbf{x})$ can be defined as*

$$\min_{\boldsymbol{\epsilon_1},\boldsymbol{\epsilon_2}} \left\{ \sum_{i=0}^{n_1-1} -\log(\boldsymbol{\epsilon_1}[i] + \boldsymbol{\epsilon_2}[i]) \right\}, \textit{subject to}$$

$$\begin{aligned}
\hat{\mathbf{z}}_{(1)} &\in \mathbf{\Omega}_{\boldsymbol{\epsilon_1},\boldsymbol{\epsilon_2}}(\mathbf{x}) \\
\mathbf{z}_{(i+1)} &= Conv(\hat{\mathbf{z}}_{(i)}, \mathbf{W}_c(i)) + \mathbf{b}_c(i), i = 1, ..., N \\
\mathbf{z}_{(i+1)} &= \mathbf{W}_{(i)}\hat{\mathbf{z}}_{(i)} + \mathbf{b}_{(i)}, i = N+1, ..., N+M-1 \\
\hat{\mathbf{z}}_{(i)} &= \sigma(\mathbf{z}_{(i)}), i = 2, ..., N+M-1 \\
\mathbf{z}_{(N+M)}[t] &- \mathbf{z}_{(N+M)}[j] \geq \delta, j = 1, .., K \textit{ and } j \neq t,
\end{aligned}$$

where δ is a small positive number to ensure that all the examples in the asymmetric robustness space $\mathbf{\Omega}_{\boldsymbol{\epsilon_1},\boldsymbol{\epsilon_2}}(\mathbf{x})$ are classified as t by the model. From the definition, the problem can be degenerated into certifying the non-uniform robustness bound if $\boldsymbol{\epsilon_1} = \boldsymbol{\epsilon_2}$; it can be further degenerated into certifying the uniform robustness bound problem if $\boldsymbol{\epsilon_1} = \boldsymbol{\epsilon_2} = \alpha\mathbf{1}$, where α is a scalar. We will show the experimental results on a 2D synthetic dataset in Section 4 for a more intuitive understanding about the difference between the asymmetric and symmetric robustness bounds. According to [25], the symmetric robustness certification problem is NP-complete. It follows that our problem here is at least NP-complete, which is computationally difficult to seek the optimum solution.

3.2.2 Relaxation

In order to make Problem 1 computationally feasible, we relax it to an optimization problem with equality constraints. Towards this, we strengthen the inequality constraints in Problem 1 into the equality constraints, which involves bounding the output of the model. Further, the upper and lower bounds of the model can be obtained by bounding the output of each layer. Now, to bound an output at each layer, we can bound the before-activation output and the after-activation output. Towards this, we first give the following theorem, which can provide the upper and lower bounds of the before-activation output of each layer. In the rest of this paper, the vector inequalities are element-wise without explicit specification. For simplicity, we define $[\mathbf{W}]_+ = \max\{\mathbf{W}, 0\}$ and $[\mathbf{W}]_- = \min\{\mathbf{W}, 0\}$.

Theorem 1. *Suppose that a fully-connected layer or a convolutional layer with an input $\mathbf{z}_{in}$ bounded by $\mathbf{l}_{in} \leq \mathbf{z}_{in} \leq \mathbf{u}_{in}$. (1) For the fully-connected layer, we assume its parameters are $\mathbf{W}$ and $\mathbf{b}$. Then, the output of the layer $\mathbf{z}_{out}$ is bounded by $\mathbf{l}_{out} \leq \mathbf{z}_{out} \leq \mathbf{u}_{out}$, where*

$$\begin{aligned}
\mathbf{l}_{out} &= [\mathbf{W}]_+\mathbf{l}_{in} + [\mathbf{W}]_-\mathbf{u}_{in} + \mathbf{b} \\
\mathbf{u}_{out} &= [\mathbf{W}]_+\mathbf{u}_{in} + [\mathbf{W}]_-\mathbf{l}_{in} + \mathbf{b}.
\end{aligned}$$

(2) For the convolutional layer, we assume its parameters are $\mathbf{W}_c$ and $\mathbf{b}_c$. Then, the output of the layer $\mathbf{z}_{out}$ is bounded by $\mathbf{l}_{out} \leq \mathbf{z}_{out} \leq \mathbf{u}_{out}$, where

$$\begin{aligned}
\mathbf{l}_{out} &= Conv(\mathbf{l}_{in}, [\mathbf{W}_c]_+) + Conv(\mathbf{u}_{in}, [\mathbf{W}_c]_-) + \mathbf{b}_c \\
\mathbf{u}_{out} &= Conv(\mathbf{u}_{in}, [\mathbf{W}_c]_+) + Conv(\mathbf{l}_{in}, [\mathbf{W}_c]_-) + \mathbf{b}_c.
\end{aligned}$$

Proof. For simplicity, we only give the proof for the fully-connected layer scenario. The proof for the convolutional layer scenario can be deduced in a similar way. We start from considering a single entry $(\mathbf{W}\mathbf{z}_{in} + \mathbf{b})_i$, given by $(\mathbf{W}\mathbf{z}_{in} + \mathbf{b})_i = \sum_j \mathbf{W}_{ij}\mathbf{z}_{in(j)} + \mathbf{b}_i$, where $\mathbf{W}_{ij}$ denotes the entry in the i-th row and j-th column of $\mathbf{W}$, $\mathbf{z}_{in(j)}$ denotes the j-th entry of $\mathbf{z}_{in}$, and $\mathbf{b}_i$ denotes the i-th entry of $\mathbf{b}$. When $\mathbf{W}_{ij} > 0$, $\mathbf{W}_{ij}\mathbf{z}_{in(j)}$ increases with the increase of $\mathbf{z}_{in(j)}$, and vice versa. Therefore, to maximize it, we choose $\mathbf{z}_{in(j)} = \mathbf{u}_{in(j)}$ if $\mathbf{W}_{ij} > 0$, else $\mathbf{z}_{in(j)} = \mathbf{l}_{in(j)}$. Whereas, to minimize the entry, we choose $\mathbf{z}_{in(j)} = \mathbf{l}_{in(j)}$ if $\mathbf{W}_{ij} > 0$, else $\mathbf{z}_{in(j)} = \mathbf{u}_{in(j)}$. Then, we have

$$\begin{aligned}
(\mathbf{W}\mathbf{z}_{in} + \mathbf{b})_i &\geq \sum_j ([\mathbf{W}_{ij}]_+\mathbf{l}_{in(j)} + [\mathbf{W}_{ij}]_-\mathbf{u}_{in(j)}) + \mathbf{b}_i \\
(\mathbf{W}\mathbf{z}_{in} + \mathbf{b})_i &\leq \sum_j ([\mathbf{W}_{ij}]_+\mathbf{u}_{in(j)} + [\mathbf{W}_{ij}]_-\mathbf{l}_{in(j)}) + \mathbf{b}_i.
\end{aligned}$$

Through the above deduction, we can get the upper and lower bounds of a single entry. Based on the upper and lower bounds of each entry in $(\mathbf{W}\mathbf{z}_{in} + \mathbf{b})$, we can bound $(\mathbf{W}\mathbf{z}_{in} + \mathbf{b})$ by

$$\begin{aligned}
(\mathbf{W}\mathbf{z}_{in} + \mathbf{b}) &\geq [\mathbf{W}]_+\mathbf{l}_{in} + [\mathbf{W}]_-\mathbf{u}_{in} + \mathbf{b} \\
(\mathbf{W}\mathbf{z}_{in} + \mathbf{b}) &\leq [\mathbf{W}]_+\mathbf{u}_{in} + [\mathbf{W}]_-\mathbf{l}_{in} + \mathbf{b}.
\end{aligned}$$

□

According to Theorem 1, we can bound the before-activation output of a layer. Then, we can get the lower and upper bounds of the after-activation output of a layer by activating the corresponding bound. Up to now, we can bound the output of the model by applying the above process layer by layer (for the max pooling layer, we can get the lower and upper bounds of its output by max pooling the corresponding bound of its input).

Specifically, given a DNN defined in Equation 2 with the parameter set θ, and an input example $\mathbf{x}$, $\hat{\mathbf{z}}_{(1)}$ can be bounded by $(\mathbf{x} - \boldsymbol{\epsilon_1}) = \hat{\mathbf{l}}_{(1)} \leq \hat{\mathbf{z}}_{(1)} \leq \hat{\mathbf{u}}_{(1)} = (\mathbf{x} + \boldsymbol{\epsilon_2})$. The bounding process follows two steps: first, we get the lower and upper bounds of the output of the last convolutional layer; second, based on the obtained bounds, we can further get the upper and lower bounds of the output of the model. Formally, based on Theorem 1, we show the two steps below.

STEP 1 : $2 \leq i \leq N+1$:

$$\begin{aligned}
\mathbf{l}_{(i)} &= Conv(\hat{\mathbf{l}}_{(i-1)}, [\mathbf{W}_{\mathbf{c(i-1)}}]_+) + Conv(\hat{\mathbf{u}}_{(i-1)}, \\
&\quad [\mathbf{W}_{\mathbf{c(i-1)}}]_-) + \mathbf{b}_{c(i-1)} \\
\mathbf{u}_{(i)} &= Conv(\hat{\mathbf{u}}_{(i-1)}, [\mathbf{W}_{\mathbf{c(i-1)}}]_+) + Conv(\hat{\mathbf{l}}_{(i-1)}, \\
&\quad [\mathbf{W}_{\mathbf{c(i-1)}}]_-) + \mathbf{b}_{c(i-1)} \\
\mathbf{l}_{(i)} &\leq \mathbf{z}_{(i)} \leq \mathbf{u}_{(i)}, \hat{\mathbf{l}}_{(i)} = \sigma(\mathbf{l}_{(i)}), \hat{\mathbf{u}}_{(i)} = \sigma(\mathbf{u}_{(i)})
\end{aligned} \tag{3}$$

STEP 2 : $N+2 \leq i \leq N+M-1$:

$$\begin{aligned}
\mathbf{l}_{(i)} &= [\mathbf{W}_{\mathbf{c(i-1)}}]_+\hat{\mathbf{l}}_{(i-1)} + [\mathbf{W}_{\mathbf{c(i-1)}}]_-\hat{\mathbf{u}}_{(i-1)} + \mathbf{b}_{c(i-1)} \\
\mathbf{u}_{(i)} &= [\mathbf{W}_{\mathbf{c(i-1)}}]_+\hat{\mathbf{u}}_{(i-1)} + [\mathbf{W}_{\mathbf{c(i-1)}}]_-\hat{\mathbf{l}}_{(i-1)} + \mathbf{b}_{c(i-1)} \\
\mathbf{l}_{(i)} &\leq \mathbf{z}_{(i)} \leq \mathbf{u}_{(i)}, \hat{\mathbf{l}}_{(i)} = \sigma(\mathbf{l}_{(i)}), \hat{\mathbf{u}}_{(i)} = \sigma(\mathbf{u}_{(i)})
\end{aligned} \tag{4}$$

Equation 3 and Equation 4 represent bounding the output of each convolutional layer and fully-connected layer, respectively. Based on the two equations, we can obtain the lower and upper

bounds ($\mathbf{l}_{(i)}$ and $\mathbf{u}_{(i)}$ in the above steps) of the output of each layer, and further we can bound the output of the model.

Recall that Problem 1 is a constrained optimization problem with inequality constraints and thus practically infeasible to be solved via standard approaches. Therefore, we utilize the upper and lower bounds of an output to relax it to an optimization problem with equality constraints. Then, we solve it through the augmented Lagrangian method [26], [27]. Based on the lower and upper bounds of an output, the relaxed problem can be formulated as below.

Problem 2. *Given a DNN with parameter set θ, and an input example $\mathbf{x}$ with ground truth label $t \in \{1, 2, ..., K\}$, the problem of obtaining the maximum asymmetric robustness bound $\mathbf{\Omega}_{\boldsymbol{\epsilon}_1,\boldsymbol{\epsilon}_2}(\mathbf{x})$ can be defined as*

$$\min_{\boldsymbol{\epsilon}_1,\boldsymbol{\epsilon}_2,\mathbf{c}} \left\{ \sum_{i=0}^{n_1-1} -\log(\boldsymbol{\epsilon}_1[i] + \boldsymbol{\epsilon}_2[i]) \right\}, \textit{subject to} \quad \mathbf{l}_{(N+M)}[t]\mathbf{1} - \mathbf{u}_{(N+M)}[j \neq t] - \delta = \mathbf{c}, \tag{5}$$

where $\mathbf{l}_{(N+M)}$ and $\mathbf{u}_{(N+M)}$ denote the lower and upper bounds of the output logits of model f_θ respectively, $\mathbf{u}_{(N+M)}[j \neq t] \in \mathbb{R}^{K-1}$ denotes the output logits except the label class t, $\mathbf{1} \in \mathbb{R}^{K-1}$ is a vector whose elements are 1, and $\mathbf{c} \in \mathbb{R}_+^{K-1}$ is a slack variable.

Evidently, since the constraints of Problem 2 are stronger than that of Problem 1, the solution of Problem 2 provides an upper bound of that of Problem 1. That is to say, it provides a sound but incomplete solution for Problem 1.

Algorithm 1: Optimization of the Asymmetric Bound

Input: Network parameters: $\{\mathbf{W}_{c(i)}, \mathbf{b}_{c(i)}\}_{i=1}^{N} \cup \{\mathbf{W}_{(i)}, \mathbf{b}_{(i)}\}_{i=(N+1)}^{(N+M-1)}$, the asymmetric bounds: $\boldsymbol{\epsilon}_1(0), \boldsymbol{\epsilon}_2(0)$, the maximum iterations: I, augmented coefficient $\{\eta^{(i)}\}_{i=1}^{I}$, decay factor τ.

Output: $\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2$

1 //initialization;
2 $\boldsymbol{\epsilon}_1 = \boldsymbol{\epsilon}_1(0), \boldsymbol{\epsilon}_2 = \boldsymbol{\epsilon}_2(0), \boldsymbol{\lambda} = 0, \eta^{(1)} = 1$;
3 **for** *i=1,...,I* **do**
4 Update $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$ by minimizing the inner problem of Problem 2 of substituting the optimal solution ;
5 $\boldsymbol{\lambda} = \boldsymbol{\lambda} + \eta^{(i)}(\mathbf{v} - \mathbf{c})$;
6 **end**
7 **while** $\mathbf{v} \geq 0$ *is not satisfied* **do**
8 $\boldsymbol{\epsilon}_1 = \tau\boldsymbol{\epsilon}_1$;
9 $\boldsymbol{\epsilon}_2 = \tau\boldsymbol{\epsilon}_2$;
10 **end**

3.2.3 Optimization

Now, we are ready to solve the asymmetric robustness certification problem. Since Problem 2 is an optimization problem with equality constraints, it can be solved with a customized augmented Lagrangian method [26], [27]. Specifically, we transform Problem 2 to an unconstrained one (the Lagrange of constrained problem) with an additional penalty term which is designed to mimic a Lagrange multiplier:

$$\max_{\boldsymbol{\lambda}} \min_{\boldsymbol{\epsilon}_1,\boldsymbol{\epsilon}_2,\mathbf{c}} \left\{ \sum_{i=0}^{n_1-1} -\log(\boldsymbol{\epsilon}_1[i] + \boldsymbol{\epsilon}_2[i]) \right\} + \boldsymbol{\lambda}^{\mathbf{T}}(\mathbf{v} - \mathbf{c}) + \frac{\eta}{2} \|\mathbf{v} - \mathbf{c}\|_2, \tag{6}$$

where $\mathbf{v}$ is a simple representation of $(\mathbf{l}_{(N+M)}[t]\mathbf{1} - \mathbf{u}_{(N+M)}[j \neq t] - \delta)$, $\boldsymbol{\lambda} \in \mathbb{R}^{K-1}$ is the dual variable of $(\mathbf{v} - \mathbf{c})$, and η is a positive coefficient. Since the inner problem is a quaratic form of $\mathbf{c}$, we can get the optimal $\mathbf{c}$: $\mathbf{c} = \max(0, \mathbf{v} + \frac{1}{\eta}\boldsymbol{\lambda})$. Substituting the optimal solution into the unconstrained problem, we can use gradient descent to optimize $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$. The pseudo code of optimizing the asymmetric bound is described in Algorithm 1, where line 4 utilizes the gradient descent to optimize $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$, and lines 7-9 ensure $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$ meet the constraints of Problem 2.

4 Evaluation

In this section, we evaluate the performance of `Amoeba`. We first compare its performance with existing robustness certification methods. Second, we visualize the asymmetric robustness bound to present an intuitive understanding. Finally, we focus on the observations different from previous works via the asymmetric robustness bound based fine-grained analysis.

4.1 Experimental Setup

Datasets. We evaluate `Amoeba` on four qualitatively different datasets: a 2D synthetic dataset [22] and three commonly used benchmark datasets: MNIST [28], FMNIST [29] (we use FMNIST to denote Fashion-MNIST for convenience), and SVHN [30].

The 2D synthetic dataset is a task to classify points in $[-1, 1]^2$ into 10 classes. To construct the 2D synthetic dataset, following the method in [22], we first randomly select 10 points in the space of $[-1, 1]^2$ as seeds and label them with $\{0, 1, ..., 9\}$. Then, we assign another 10,000 randomly selected points in $[-1, 1]^2$ the same label as the seed with the closest l_2 distance, and add them to the 2D synthetic dataset. If a point has the same closest l_2 distance with two or more seeds, from which we randomly select one and assign its label to the point.

MNIST and FMNIST are frequently used for the handwritten digit recognition task and fashion recognition task respectively, and their examples are 28×28 grayscale images. SVHN is used for the street view digit image recognition task, and its examples are 32×32 RGB images. To facilitate the comparison with state-of-the-art work [22], we also normalize the pixels of each image in MNIST, FMNIST and SVHN to $[-1, 1]$.

Baseline Methods and Evaluation Metrics. We evaluate `Amoeba` from its effectiveness and efficiency. (1) For effectiveness, following the previous work [22], we evaluate `Amoeba` in terms of the volume of the certified robustness space. In fact, the adversary-free constraints in the certification guarantee that the certified region is within the decision boundary, i.e, the volume of certified region is bounded by that of the decision boundary. Therefore, a large volume implies a tight approximation of the real robustness space. Specifically, we use the geometric mean[1] (for convenience, we use mean and geometric mean alternatively

1. The geometric mean is defined as the n-th root of the product of n numbers. The certified robustness space is a hyperrectangular space, and the volume of the hyperrectangle is the product of n side lengths. Therefore, the geometric mean of the certified bound can be used to measure the volume of the corresponding bounded space.

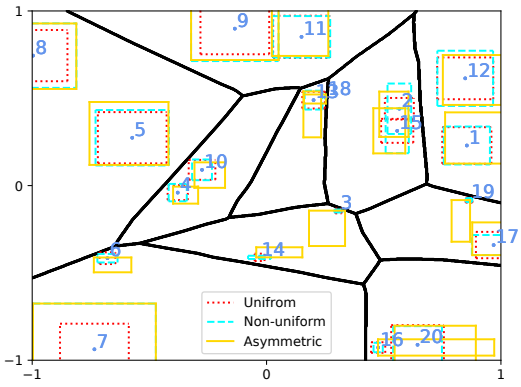


Fig. 2. Visualization on the 2D synthetic dataset. This figure shows the uniform robustness bound (red dotted line), non-uniform robustness bound (cyan dashed line) and asymmetric robustness bound (gold solid line) of 20 randomly selected examples, where the black lines represent the real decision boundary and the number represents the index of the example.

in the following) to calculate the volume of the certified robustness space. Mathematically, the geometric mean of the asymmetric robustness bound is $(\prod_{i=0}^{n_1-1}(\boldsymbol{\epsilon_1}[i]+\boldsymbol{\epsilon_2}[i]))^{\frac{1}{n_1}}$. (2) For efficiency, we compare `Amoeba` with the symmetric counterparts [19], [22] in terms of their running time on the real-world datasets.

4.2 Performance Comparison

4.2.1 2D Synthetic Dataset

We start by reporting the results on the 2D synthetic dataset to get an intuitive understanding of the difference among the three certified bounds: the asymmetric robustness bound (this paper), the non-uniform robustness bound [22] and the uniform robustness bound [18], [19], [20], [31], [32]. We randomly select 90% of the data points in the dataset for training and the rest are used for testing. Then, we train a FCNN as the classification model, which has two hidden layers that contain 16 and 32 neurons respectively. This model can achieve an accuracy of 99.9%.

Figure 2 shows the certified bounds of 20 randomly selected examples. In the certification, we aim to make the certified bound occupy as large blank area as possible without crossing the decision boundary (black line). Taking example 3 as an example, its uniform robustness bound (red box) is very small as it is very close to the decision boundary; the non-uniform robustness bound (cyan box) is slightly larger than the uniform robustness bound. As a comparison, the blank area covered by the asymmetric robustness bound (gold box) is much larger than those of the previous methods. Based on this observation, we can find that the asymmetric robustness bound is more accurate than existing counterparts.

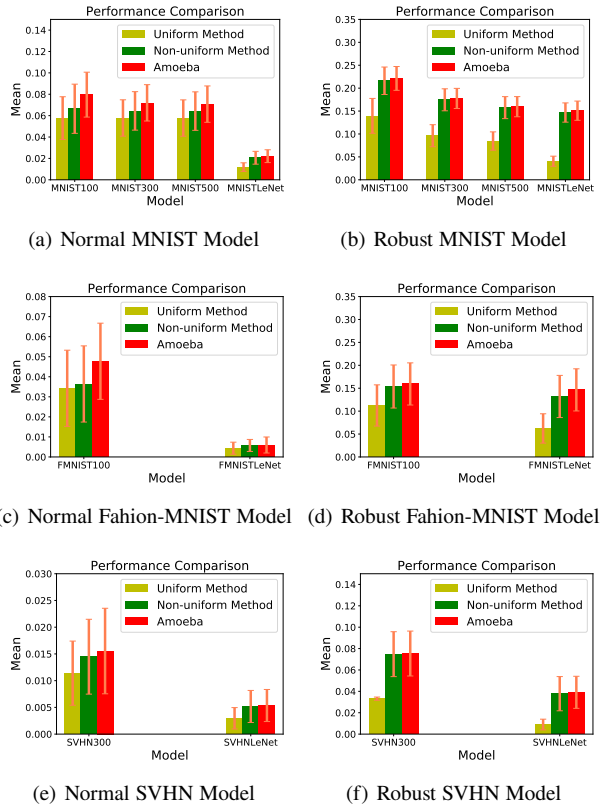
In the above case, the uniform robustness bound forms a square *centered on* the input example, the non-uniform robustness bound forms a rectangle *centered on* the input example, while the asymmetric robustness bound forms a rectangle *containing* the input example. In fact, both the uniform and non-uniform robustness bounds can be considered as special cases of the asymmetric robustness bound, which can also be seen from the formalization of Problem 1.

4.2.2 Real-world Datasets

Now, we present the quantitative comparison results on the real-world datasets, where MNIST and FMNIST are respectively divided into a training set of 60,000 examples and a test set of 10,000 examples, and SVHN is divided into a training set of 73,657 examples and a test set of 26,032 examples. In this evaluation, we test the mean of the asymmetric and symmetric robustness bounds using normal and robust models. The normal models are trained with normal training while the robust models are trained with the PGD-based adversarial training (Section 2.3). We here select the PGD-based adversarial training since it is a widely used one [33] and is shown to be able to improve the robustness of many normal models significantly [34]. Note that, our evaluation can be trivially extended to other adversarial training methods. Our approach does not modify the original model. Therefore, it does not affect the model accuracy on benign examples, as shown in Table 1. The detailed training configurations and the model architectures are shown in Tables 5, 6, and 7 of the appendix, respectively.

TABLE 1
Normal classification accuracy of different models.

Dataset	Model	Accuracy	Dataset	Model	Accuracy
MNIST	MNIST100 (Normal)	97.66%	FMNIST	FMNIST100(Normal)	88.38%
	MNIST100 (Robust)	98.12%		FMNIST100(Robust)	84.19%
	MNIST300 (Normal)	98.13%		FMNISTLeNet(Normal)	90.08%
	MNIST300 (Robust)	98.56%		FMNISTLeNet(Robust)	85.47%
	MNIST500 (Normal)	97.89%	SVHN	SVHN300(Normal)	82.82%
	MNIST500 (Robust)	98.89%		SVHN300(Robust)	75.94%
	MNISTLeNet (Normal)	98.91%		SVHNLeNet(Normal)	88.63%
	MNISTLeNet (Robust)	99.19%		SVHNLeNet(Robust)	76.55%



(a) Normal MNIST Model (b) Robust MNIST Model

(c) Normal Fahion-MNIST Model (d) Robust Fahion-MNIST Model

(e) Normal SVHN Model (f) Robust SVHN Model

Fig. 3. `Amoeba` vs. State-of-the-art robustness certification methods.

Figure 3 shows the mean of the three certified bounds on different datasets and with different models (for each dataset, we randomly select 2,000 correctly classified examples from its test set to conduct the certification evaluation and without of explicit specification, our evaluations in the rest of this paper are conducted on the 2,000 selected examples), where a large mean implies a large robustness space. As shown in Figure 3, in all the cases, `Amoeba` can obtain larger means than that of the uniform and non-uniform robustness bound certification methods, i.e., the result of

TABLE 2
Running time comparison.

Model	Time (s)		
	Uniform [19]	Non-uniform [22]	Amoeba
MNIST100	1.26	1.41	1.51
MNIST300	1.31	1.66	1.88
MNIST500	1.85	2.47	2.76
MNISTLeNet	1.23	1.24	1.24
FMNIST100	1.23	1.43	1.53
FMNISTLeNet	1.23	1.24	1.24
SVHN300	2.63	4.01	4.79
SVHNLeNet	1.29	1.3	1.31

Amoeba is more accurate, which is consistent with our intuition. Especially, in Figure 3(c), for the normal FMNIST100 model, the mean of the asymmetric robustness bound is 1.41 and 1.33 times of that of the uniform and non-uniform robustness bounds, respectively. Since an example in FMNIST has 784 features, the volume of the asymmetric robustness bound is 1.41^{784} and 1.33^{784} times of that of the uniform and non-uniform robustness bounds respectively, which is a great performance gain. This is because the symmetric robustness bound considers that the left and right bounds are equal and will be limited by the smaller one of the two bounds. As a comparison, the asymmetric robustness bound takes into account the inherent heterogeneity of perturbation direction, so the left and right bounds will not be restricted by each other. Evidently, by considering the inherent heterogeneity, Amoeba can provide a quantitative robustness measurement to the perturbation direction.

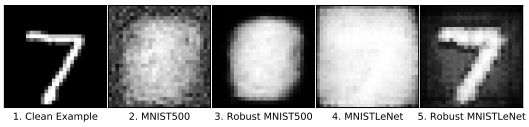
To demonstrate that the significant performance gain is not accompanied by a large efficiency overhead, we compare Amoeba with state-of-the-art methods [22], [19] in terms of running time. Specifically, we examine the average running time of Amoeba and the symmetric counterparts on different datasets and with different models. The results are shown in Table 2. From Table 2, compared with the symmetric counterparts, Amoeba only introduces negligible extra running time. For example, for the MNIST100 model, compared with [22], Amoeba introduces only 0.1s of extra running time. Therefore, considering the significant performance gain of Amoeba, the marginal extra running time is acceptable. In the future, it is interesting to further optimize the efficiency of Amoeba.

In summary, the results on the real-world datasets show that Amoeba outperforms the symmetric counterparts: (1) Amoeba provides a much more accurate quantitative robustness measurement regarding the perturbation magnitude with negligible extra overhead; (2) since the asymmetric robustness bounds contain the left and right bounds, Amoeba can also provide a quantitative robustness measurement to the perturbation direction, which cannot be achieved by previous works.

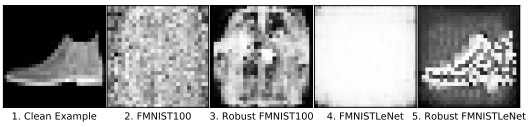
4.3 Visualization

Now, we visualize the asymmetric robustness bound $(\epsilon_1 + \epsilon_2)$ to provide better understanding. In our visualization, we plot the asymmetric robustness bound in terms of $1 - (\epsilon_1 + \epsilon_2)$, and thus a small bounded space will result in a brighter visual perception. Notice that, according to the physical meaning of the asymmetric robustness bound, a small bounded space also indicates the weak resistance against adversarial perturbations. Hence, these pixels with small bounded space are critical for both model decision-making and model security.

Figure 4 visualizes the asymmetric robustness bounds of several test examples. We can see from Figure 4 that, on the normal model, the shape of the input example and its bounded robustness space are less related. Further, the robustness spaces of the pixels are very small, even for those that are not related to decision-making from the human's view. This reveals that small perturbations on those "unimportant" pixels can also fool the normal model, making it not much robust.



(a) MNIST



(b) FMNIST

Fig. 4. Visualization of the asymmetric bounds of test examples on MNIST and FMNIST. Figures 4(a).2, 4(a).4, 4(b).2 and 4(b).4 represent the asymmetric bounds of the normal models, and Figures 4(a).3, 4(a).5, 4(b).3 and 4(b).5 represent the asymmetric bounds of the robust models.

For robust FCNNs, Figures 4(a).3 and 4(b).3 show the asymmetric robustness bounds of robust MNIST500 and robust FMNIST100, respectively. From Figure 4(a).3, we can observe a large white elliptic area which implies that the robustness spaces of the pixels in this area are very small. It follows that small perturbations on these pixels will very likely cause misclassification of the model, which is consistent with our human perception since the example's feature pixels are located in the elliptic area. Figure 4(b).3 shows a white area with the shape of the coat, which is quite different from the shape of the original example with the ground truth label boot. In fact, we find a very interesting phenomenon: for robust FCNNs on MNIST (e.g., the robust MNIST500), the asymmetric robustness spaces of all the examples in MNIST are elliptic areas; while for robust FCNNs on FMNIST (e.g., the robust FMNIST100), the asymmetric robustness spaces of all the examples in FMNIST are coat-like areas (see more examples in Figure 9 of the appendix). We will make more exploration and explanation on this interesting phenomenon in Section 5.2.

For robust CNNs, the visualization results are more interpretable. We can clearly see the outline of 7 from Figure 4(a).5, and of a boot from Figure 4(b).5. These outlines are also critical to our human decision-making.

In a word, through the above visualizations, we can find that the robust models are more interpretable than the normal models, and the CNNs are more interpretable than the FCNNs.

4.4 Robustness Space Analysis

In the state-of-the-art work [22], the authors certified the robustness for FCNNs and found that the robustness space shapes (in terms of ϵ) of different examples under the same model are highly correlated and such correlation is even stronger under the robust models. However, after we evaluate on more general neural networks (e.g., CNNs) and analyze it more finely (e.g., the asymmetry), we find that such claim may not necessarily hold.

[22] used the pair-wise cosine similarity to measure such correlation. Following [22], we first calculate the pair-wise cosine

similarity[2] with respect to $(\epsilon_1 + \epsilon_2)$ of 200 randomly selected examples. Then, we report the average cosine similarity as the result.

The third column of Table 3 shows the average cosine similarity of the robustness space shapes. As shown in Table 3, the cosine similarity of the robustness spaces is very high, which is consistent with [22]'s observation. For example, the average cosine similarity for the robust MNISTLeNet model is 0.9234, indicating a high similarity. However, for the robust MNISTLeNet, the shapes of the asymmetric spaces of different examples are very different, as shown in Figure 9 of the appendix. In other words, they should not have a high similarity from human's perspective. In fact, the cosine similarity is a measure of direction similarity between two non-zero vectors and the direction of a vector is mainly affected by the large elements in it. However, the difference of two robustness spaces mainly lies in the positions with small elements since the key pixels tend to have small robustness space values. Even if the robustness spaces of different examples differ greatly, their cosine similarity is still very high, as supported by the results in Table 3. Therefore, cosine similarity may not be a proper metric for measuring the similarity of robustness space shapes.

Different from cosine similarity, the Wasserstein distance[3] can measure such difference. The Wasserstein distance for the asymmetric robustness bound can be intuitively understood as the minimum cost of changing from one robustness bound to another. A small Wasserstein distance implies a high geometric similarity. Therefore, we use the average pair-wise Wasserstein distance within the 200 examples' certified robustness space to evaluate their shape correlation. For a more intuitive understanding of the size of the Wasserstein distance, we also calculate the average pair-wise Wasserstein distance within the example distributions of different categories of MNIST, and the result is 0.045. As shown in Table 3, we have the following observations. (1) For most models, the large average Wasserstein distance indicates large difference between the robustness spaces. For example, the average Wasserstein distance 0.0573 of the robust MNISTLeNet demonstrates the difference between robustness spaces is even larger than that between the examples of different categories. Therefore, the robustness space shapes of different examples under the same model are not highly correlated. (2) For FCNNs, we find that except for FMNIST100, the average Wasserstein distance under the robust model is smaller than that under the normal model, which is roughly consistent with [22]. However, for CNNs, we find that the average Wasserstein distance under the robust model is larger than that under the normal model. Therefore, under more general network architectures, it is not suitable to think that the robustness space correlation of robust models is stronger than that of normal models.

The above correlation analysis is coarse-grained because it does not consider the asymmetry characteristic of the certified bounds. For a 1D case as an example, we assume that the asymmetric robustness bounds of two different examples are $(\epsilon_1 = 0.1, \epsilon_2 = 0.2)$ and $(\epsilon_1 = 0.2, \epsilon_2 = 0.1)$, respectively. In this case, the Wasserstein distance between the two examples in terms of $(\epsilon_2 + \epsilon_1)$ is 0 even though they are different. Therefore, we use the average Wasserstein distance of $(\epsilon_2 - \epsilon_1)$ for a more fine-grained analysis. In fact, $(\epsilon_2 - \epsilon_1)$ can provide more detailed information about the asymmetry of the robustness space. The average pair-wise Wasserstein distance with respect to $(\epsilon_2 - \epsilon_1)$ of the 200 examples is shown in Table 3. As we can see from Table 3: (1) The asymmetry of the robustness space shapes of different examples under the same model differ greatly. For example, the average Wasserstein distance is 0.067 and 0.0826 under the normal and robust MNISTLeNet, respectively, which indicates high difference in the asymmetry of robustness spaces. (2) The difference is even stronger under the robust models. Specifically, the average Wasserstein distance of all the robust models now is greater than those of the normal models except for FMNISTLeNet. We speculate that such difference is related to the robustness against universal adversarial perturbation [24]. This is also supported by the observation that the robust models are more resistant against universal adversarial perturbation [35]. It might be interesting to explore the relationship between the robustness spaces of different examples and universal adversarial perturbation [24]. We will take this as a future work.

TABLE 3
Cosine similarity and wasserstein distance of asymmetric robustness bounds.

Model	Adversarial Training	Average Cosine	Average Wasserstein	
			$\epsilon_2 + \epsilon_1$	$\epsilon_2 - \epsilon_1$
MNIST100	-	0.9602	0.0581	0.0429
MNIST100	PGD	0.9926	0.0363	0.0672
MNIST300	-	0.9788	0.0416	0.0387
MNIST300	PGD	0.9964	0.0259	0.0594
MNIST500	-	0.9832	0.0451	0.0389
MNIST500	PGD	0.9968	0.0261	0.0758
MNISTLeNet	-	**0.9304**	0.0343	0.0673
MNISTLeNet	PGD	**0.9234**	0.0573	0.0826
FMNIST100	-	0.8989	0.0724	0.0587
FMNIST100	PGD	0.9724	0.1098	0.0657
FMNISTLeNet	-	0.7782	0.0209	0.1765
FMNISTLeNet	PGD	0.8726	0.1178	0.0668
SVHN300	-	0.956	0.0692	0.042
SVHN300	PGD	0.9928	0.0601	0.0686
SVHNLeNet	-	0.7822	0.0412	0.0778
SVHNLeNet	PGD	0.9764	0.0708	0.0897

In a word, through the above analysis on general network architectures, we get a conclusion different from [22]: the robustness spaces between different examples under the same model differ greatly, and the difference is even more significant under the robust model.

5 Application

In this section, for demonstrating the superiority of the asymmetric robustness bounds, we apply `Amoeba` in two security-related downstream tasks.

5.1 Modeling Stronger Adversarial Threats

`Amoeba` can be used as a navigator to provide knowledge for modeling stronger adversarial threats. Usually, we can evaluate an attack using two criteria: (1) the *evasion rate*, i.e., the proportion of the generated adversarial examples that can fool the target model, and (2) the *preserved utility*, i.e., the similarity between an adversarial example and its corresponding clean example. Considering these two criteria, in this section, we employ `Amoeba` as a navigator to enhance the utility of adversarial examples without reducing or even improving the evasion rate (i.e., modeling stronger adversarial threats), and compare its effectiveness with the state-of-the-art symmetric certification method [22]. For convenience, we denote

2. Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.

3. The Wasserstein distance is defined as the cost of the optimal transport plan for moving the mass in one distribution to another and provides a measure of the distance between two distributions. Different robustness spaces can be regarded as different distributions.

TABLE 4
Evaluation results on MNIST.

Model	Attacks	Accuracy	Distance			
			l_∞	l_0	l_2	l_1
MNIST100	FGSM (l_∞), eps = 0.2	20.30%	0.2 (0)	393.7 (33.2)	4.17 (0.17)	88.05 (6.95)
	FGSM^s(l_∞), eps = 0.2	20.30%	0.2 (0)	393.7 (33.25)	4.17 (0.21)	88.05 (6.72)
	FGSM^a(l_∞), eps = 0.2	20.30%	0.2 (0)	375.1 (30.41)	4.08 (0.16)	84.32 (6.34)
	FGSM(l_∞), eps = 0.3	9.15%	0.3 (0)	393.7 (35.3)	6.23 (0.26)	130.99 (10.8)
	FGSM^s(l_∞), eps = 0.3	9.15%	0.3 (0)	393.7 (35.5)	6.23 (0.27)	130.99 (11.4)
	FGSM^a(l_∞), eps = 0.3	9.15%	0.3 (0)	393.7 (34.7)	6.23 (0.21)	130.99 (11.9)
	PGD(l_∞), eps=0.2	6.05%	0.2 (0)	405.1 (37.4)	3.99 (0.15)	83.72 (6.42)
	PGD^s(l_∞), eps=0.2	6.05%	0.2 (0)	405.1 (37.8)	3.99 (0.15)	83.72 (6.45)
	PGD^a(l_∞), eps=0.2	6.05%	0.2 (0)	405.1 (37.9)	3.99 (0.16)	83.72 (6.48)
	PGD(l_2), eps=5.0	6.95%	0.86 (0.16)	452.8 (65.9)	4.97 (0.07)	83.7 (12.8)
	PGD^s(l_2), eps=5.0	**4.80%**	0.92 (0.19)	229.1 (44.87)	4.97 (0.06)	**66.64** (5.39)
	PGD^a(l_2), eps=5.0	**4.25%**	0.92 (0.17)	229.1 (47.2)	4.97 (0.05)	**66.65** (6.13)
	C&W(l_2), $\kappa = 0$	0.00%	0.3 (0.18)	377.6 (32.1)	1.62 (0.77)	25.45 (6.49)
	C&W^s(l_2), $\kappa = 0$	0.00%	0.51 (0.31)	77.39 (9.12)	2.18 (1.01)	18.61 (7.33)
	C&W^a(l_2), $\kappa = 0$	0.00%	0.48 (0.28)	70.32 (9.5)	2.08 (0.91)	18.21 (7.2)
MNISTLeNet	FGSM (l_∞), eps = 0.2	67.95%	0.2 (0)	395.6 (39.7)	4.19 (0.19)	88.59 (7.63)
	FGSM^s(l_∞), eps = 0.2	67.95%	0.2 (0)	395.6 (40.1)	4.19 (0.18)	88.59 (7.52)
	FGSM^a(l_∞), eps = 0.2	67.95%	0.2 (0)	395.6 (37.4)	4.19 (0.18)	88.59 (7.54)
	FGSM(l_∞),eps = 0.3	45.15%	0.3 (0)	395.6 (39.2)	6.26 (0.23)	131.8 (12.6)
	FGSM^s(l_∞), eps = 0.3	45.15%	0.3 (0)	395.6 (41.2)	6.26 (0.24)	131.8 (11.2)
	FGSM^a(l_∞), eps = 0.3	45.15%	0.3 (0)	395.6 (38.4)	6.26 (0.21)	131.8 (10.9)
	PGD(l_∞), eps=0.2	38.75%	0.2 (0)	462.5 (42.3)	3.69 (0.16)	77.03 (6.54)
	PGD^s(l_∞), eps=0.2	38.75%	0.2 (0)	462.5 (40.3)	3.69 (0.15)	77.03 (6.07)
	PGD^a(l_∞), eps=0.2	38.75%	0.2 (0)	462.5 (40.5)	3.69 (0.16)	77.03 (7.09)
	PGD(l_2), eps=5.0	**20.84%**	1.0 (0.34)	565.9 (94.5)	4.96 (0.11)	**86.5** (17.6)
	PGD^s(l_2), eps=5.0	**19.60%**	1.25 (0.41)	126.5 (43.1)	4.91 (0.16)	**47.53** (5.01)
	PGD^a(l_2), eps=5.0	**15.35%**	1.35 (0.38)	134.9 (35.8)	4.95 (0.08)	**45.95** (6.14)
	C&W(l_2), $\kappa = 0$	**0.00%**	0.58 (0.28)	415.3 (38.2)	2.22 (0.92)	**28.8** (11.2)
	C&W^s(l_2), $\kappa = 0$	**0.00%**	0.87 (0.46)	72.64 (8.6)	2.79 (1.38)	**22.71** (11.4)
	C&W^a(l_2), $\kappa = 0$	**0.00%**	1.11 (0.36)	57.48 (8.12)	3.04 (1.45)	**19.46** (9.82)

[1] We use all the normal models from Section 4.2.2 as the target models. Refer to Tables 1, 5, 6, and 7 for more details about the normal models.
[2] The values in brackets denote the corresponding standard deviation, which is small compared to the corresponding distance.

the certification method in [22] as `Nonuniform` in the following discussion.

Intuitively, the pixels with smaller robustness spaces are more vulnerable to adversarial perturbations. Based on this, for generating an adversarial example more effectively and efficiently, we can start from manipulating the pixels with small robustness spaces. Specifically, given the robustness bound certified by `Amoeba` or `Nonuniform` [22], we first select a certain proportion of pixels with the smallest robustness spaces and then generate perturbations on those pixels using existing attacks. To ensure that the perturbations are generated only on those selected pixels, two mask matrices, whose elements corresponding to the selected pixels are valid and otherwise invalid, can be used when generating adversarial perturbations. We show the pseudo code of the enhanced attack in Algorithm 2, which is deferred in A of the appendix due to the space limitations.

To evaluate Algorithm 2, we mainly use three widely used adversarial attacks: FGSM (l_∞-based) [23], PGD (l_2-based and l_∞-based) [10], and the C&W (l_2-based) attack [13]. Note that, Algorithm 2 can also be applied to enhance other adversarial attacks. We use A^a and A^s to denote the enhanced version of the attack A based on `Amoeba` and `Nonuniform`, respectively, e.g, FGSM^a and FGSM^s denote the enhanced version of FGSM based on `Amoeba` and `Nonuniform`, respectively. The setting of hyperparameters for all the attacks is summarized in Table 8 of the appendix. For each dataset, we use the original and enhanced attacks to generate adversarial examples, and then evaluate their evasion rate (in terms of model accuracy) and utility (in terms of l_1-distance).

Due to the space limitations, we only report partial results on MNIST here in Table 4. More experimental results on MNIST, FMNIST and SVHN are deferred to Tables 9, 10, and 11 in the appendix. From Tables 4, 9,10 and 11, we have the following observations. (1) Without reducing the evasion rate, both `Amoeba` and `Nonuniform` can help all the existing l_2-based attacks decrease the total perturbation required to generate adversarial examples. For example, for MNISTLeNet, compared with C&W, C&W^s and C&W^a decrease the total perturbations from 28.8 to 22.71 and 19.46 respectively, while maintaining the evasion rate of 100%. (2) In some cases, both `Amoeba` and `Nonuniform` can even improve the evasion rate. For example, for MNISTLeNet, compared with PGD (l_2), PGD^s (l_2) and PGD^a (l_2) not only decrease the total perturbation from 86.5 to 47.53 and 42.95 respectively, but also reduce the model accuracy from 20.84% to 19.60% and 15.35% respectively. (3) Both `Amoeba` and `Nonuniform` cannot enhance l_∞-based attacks in some cases. We speculate this is due to the limitation of the added maximum perturbation on a single pixel in the l_∞-based attacks and so the robustness space cannot be broken. An auxiliary evidence is that since there is no such limitation in l_2-based attacks, all the enhanced l_2-based attacks in Tables 4 9,10 and 11 not only reduce the total perturbations but also remain or even improve the evasion rate. (4) For enhancing attacks, `Amoeba` is more effective than `Nonuniform`. Specifically, compared with `Nonuniform`-enhanced attacks, `Amoeba`-enhanced attacks can achieve the same success rate with less perturbations. For example, for MNISTLeNet, compared with C&W^s, C&W^a decreases the total perturbations from 22.71 to 19.46. (5) In some rare cases, compared with `Nonuniform`-enhanced attacks, `Amoeba`-enhanced attacks can achieve higher evasion rate with the cost of slight larger total perturbations. For example, for MNIST100, compared with PGD^s (l_2), PGD^a (l_2) decreases the model accuracy from 4.80% to 4.25% while only increases the perturbation from 66.64 to 66.65, which is acceptable.

In summary, `Amoeba` can serve as a navigator to guide attacks to generate adversarial perturbations at the most vulnerable pixel positions, thus enhancing their performance. According to the evaluation results, compared with the symmetric counterparts, `Amoeba` is more effective and efficient in modeling stronger adversarial threats.

5.2 Explaining the Prediction of DNNs

Understanding a DNN's prediction is quite crucial for their use in security-related scenarios. In this section, we leverage `Amoeba` to explain the DNN prediction. Similar to humans, DNNs also classify examples according to the presence or absence of certain features. For example, 7 and 1 can be distinguished according to the presence or absence of the horizontal stroke. For images, the presence of a feature comes from a large pixel value and vice versa. Therefore, increasing a pixel value implies a trend of a feature from absence to presence, and decreasing a pixel value implies a trend of a feature from presence to absence. For example, increasing the pixel values of the horizontal stroke in 1 (the trend of the horizontal stroke from absence to presence) will change 1 to 7, i.e., 7 prefers to have the horizontal stroke. Based on this, we can use `Amoeba` to explain the DNN prediction: if the predicted class prefers to have a feature, the right bound of this feature is larger than its left bound since decreasing the feature value (the trend of this feature from presence to absence) can easily result in misclassification, and vice versa. It should be noted that such explanation cannot be provided by existing symmetric certification methods since they assume the left and right bounds are equal.

Specifically, we design a method called `Explorer`, which uses the relative size of the left and right bounds to explain the DNN prediction. Given the asymmetric robustness bound certified by `Amoeba`, we perform $(\boldsymbol{\epsilon_2} - \boldsymbol{\epsilon_1})$ and a feature with large $(\boldsymbol{\epsilon_2} - \boldsymbol{\epsilon_1})$ means that the predicted class prefers to have this feature, and vice versa. Therefore, by visualizing $(\boldsymbol{\epsilon_2} - \boldsymbol{\epsilon_1})$, we can identify the features that the predicted class prefers to have. Below, we evaluate the explanation provided by `Explorer` from three important and widely used perspectives: *fidelity*, *stability*, and *comprehensibility*

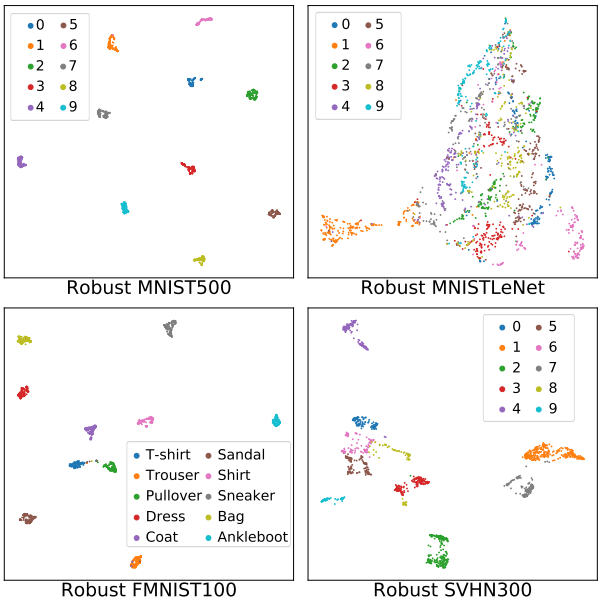


Fig. 5. Visualization of the explanation results after dimension reduction.

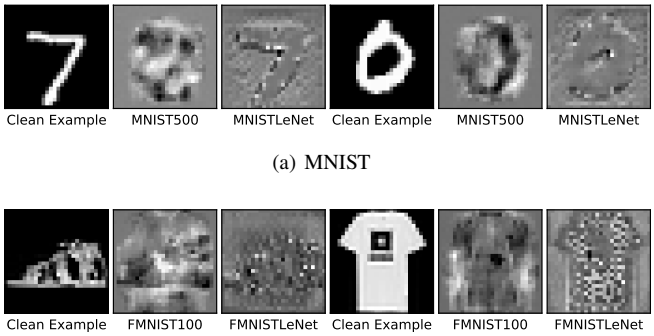
[36]. Since normal models are notorious for their interpretability [37], [38], we consider robust models in the evaluation.

Fidelity: How well does the explanation approximate the DNN prediction? High fidelity is crucial for an explanation since otherwise the explanation is meaningless. `Explorer` is a prediction-based explanation method, which uses the dynamic behavior of the model over perturbations to explore the features that the predicted class tends to have or tends not to have. This process reflects the prediction behavior of the model to be explained. Therefore, the explanation provided by `Explorer` has high fidelity by design.

Stability: How similar are the explanations for similar examples? High stability means that explanations for similar examples should be similar (unless the predicted classes are different). In order to evaluate the stability of `Explorer`, we first select several representative models trained on MNIST, FMNIST and SVHN. Then, for a given model, we use `Explorer` to explain 2,000 randomly selected examples and use UMAP [39] (a dimension reduction technique) to reduce each explanation result to 2 dimensions. After dimension reduction, we visualize the results to examine the stability of `Explorer`.

Figure 5 visualizes the dimension reduction results for the explanations on four representative models. From Figure 5, we have the following observations. (1) For each model, the explanations of similar examples in the same class are geometrically adjacent, i.e. their explanation results are similar, which indicates the high stability of `Explorer`. (2) The explanation results of examples in different classes are very different, which indicates that `Explorer` can illustrate the differences between examples in different classes. Therefore, `Explorer` can provide explanations with high stability.

Comprehensibility: How well do humans understand the explanation? A good explanation should be easily understandable by humans and lead to some insights. To evaluate the comprehensibility of `Explorer`, we randomly select some explanation results for visualization. We visualize $(\epsilon_2 - \epsilon_1)$ in terms of $1-(\epsilon_2 - \epsilon_1)$ to explain the DNN prediction. Therefore, a pixel with large $(\epsilon_2 - \epsilon_1)$ implies a dark pixel in visualization, and vice versa. Note that, based on the definition of stability and the results in Figure 5, the explanations of several examples in a certain class are sufficient to illustrate the comprehensibility of other examples in the corresponding class.



(a) MNIST

(b) FMNIST

Fig. 6. Explaining the prediction of the model. The bigger the bias bound of the pixel, the darker the pixel appears in the image. Since all the models in this evaluation are robust models, the prefix of robust is omitted for convenience.

Figure 6 shows several randomly selected examples explained by `Explorer`. Due to the space limitation, we place more explanation results in Figure 10 of the appendix. From Figures 6 and 10 of the appendix, we can understand the decision-mechanism of a DNN easier. First, we can find that the FCNNs classify an example according to the presence or absence of certain pixel features, which is consistent with our intuition. Taking the second example of Figure 6(b) as an example, we can clearly see a darker T-shirt area in the coat-like background area, while the lower half of the sleeve is brighter. In other words, the second example of Figure 6(b) is classified as a T-shirt since there are no lower half sleeves. Second, `Explorer` can also provide some insights about the interesting phenomenon mentioned in Section 4.3: for the FCNNs, the certified robustness spaces of the examples in MNIST are elliptic areas, while the certified robustness spaces of the examples in FMNIST are coat-like areas. Taking the second example of Figure 6(b) as an example, for the features in the T-shirt area, the right bound is larger than the left bound, while for the features in the lower sleeve area, the left bound is larger than the right bound. Since the sum of the left and right bounds of the features in the two areas is close, the robustness space in terms of $(\epsilon_1 + \epsilon_2)$ will result in a coat-like area, which is different from the shape of T-shirt. Finally, as shown in Figures 6 and 10 of the appendix, for CNNs, we can clearly see that the example contour area is darker in the visualization, and is very close to the shape of the corresponding example, which also indicates the high comprehensibility of the explanation provided by `Explorer`.

In summary, `Explorer` can explain the model prediction based on the bounds derived by `Amoeba`, and the explanation results have high fidelity, stability, and comprehensibility. To the best of our knowledge, `Explorer` is the first one to establish the connection between adversarial robustness and interpretability. We believe that our research of the robustness-based interpretation is a promising direction and deserves more future research.

6 Discussion

Asymmetric Robustness vs Symmetric Robustness. Compared with the symmetric robustness bound, the asymmetric robustness bound additionally takes the heterogeneity of perturbation direction into account, which further enables more in-depth and reliable model security research. Firstly, the asymmetric robustness bound can more accurately estimate the real robustness space and provide

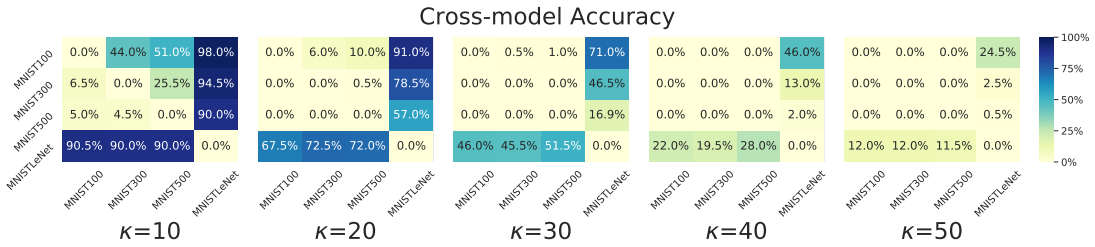


Fig. 7. Cross-model accuracy of adversarial examples generated by the C&W attack on MNIST. The horizontal axis indicates the target model and the vertical axis indicates the corresponding surrogate model.

a quantitative robustness measurement regarding both the perturbation direction and magnitude, whereas the existing symmetric robustness bound cannot provide such measurement regarding the perturbation direction. Secondly, the asymmetric robustness bound can be more valuable and practical in safety-related tasks. For example, the asymmetric robustness can be used to explain model predictions and bridge the gap between adversarial robustness and interpretability. Finally, through the asymmetric robustness analysis, it is possible for us to further improve the robustness of a model. In the previous works [18], [19], their methods are equivalent to train robust models by maximizing $\min(\epsilon_1, \epsilon_2)$. However, this may limit further improvement on the model robustness since the inherent heterogeneity of perturbation direction. Different from existing works, we find a new insight that we can train a robust model by maximizing both ϵ_1 and ϵ_2 in each dimension. Take a simple 1D task as an example, where we need to decide the magnitude of a number. For simplicity, we assume that a number greater than 100 is a large number. For an input number 101, its left and right robustness bounds are 1 and $+\infty$, respectively. In previous works [18], [19], their robust training method is to make the model think that a number in $(100, 102)$ is a large number. When considering the asymmetric robustness bounds, the robust training can make the model learn that a number in $(100, 101+\epsilon_2)$ ($\epsilon_2 \gg 1$) is a large number. Obviously, the latter helps the model learn the essential features of a large number. Therefore, finding model parameters that maximize both ϵ_1 and ϵ_2 is helpful for the model to learn the essential features of data examples. In this way, we are likely to obtain a more robust model with potentially better utility. In a word, compared with the symmetric counterparts, the asymmetric robustness bound has better accuracy, reliability and potential for many security applications.

Limitations and Future Works. As the first attempt to certify the asymmetric robustness bounds for neural networks, we believe our work can be improved from several aspects. First and foremost is extending the proposed method to certify the robustness bounds for larger models and more general networks (such as ResNets). The main challenge here is to tightly bound the output of the model. The second limitation is the efficiency issue of the proposed method, which could be further improved and dedicated research is necessary. Such research is also meaningful for obtaining more useful information and training more robust neural networks, especially for improving the model robustness in each perturbation direction. Finally, there are correlations between different pixels. Specifically, when the value of a pixel changes, it potentially affects the robustness bounds of other pixels. Therefore, how to determine this correlation and further take account of such correlation to obtain a more accurate robustness space with a more complicated relaxation is an interesting future research direction.

7 Related Works

Adversarial Attacks and Defenses. DNNs are found to be vulnerable to carefully perturbed input examples [5]. Recently, many attacks have been proposed to generate adversarial examples.

As discussed in Section 2, FGSM takes one step to disturb the original example. To enhance FGSM, many multi-step iteration methods based on FGSM are proposed, include PGD [10], BIM [8] and MI-FGSM [40]. Moosavi-Dezfooli et al. proposed another iteration method called DeepFool [41] to minimize the adversarial perturbations. Papernot et al. proposed the Jacobian-based Saliency Map Approach (JSMA) [42] to efficiently generate adversarial examples, which utilizes the Jacobian matrix to calculate the saliency map of an input example, and then modifies a small number of features based on the saliency map to deceive the target model. The C&W attack [13] includes three different attack algorithms, which make the perturbations almost imperceptible by limiting the l_∞, l_2 and l_0 distance between the adversarial example and the original example, and can control the confidence of the generated adversarial examples. Tramer et al. explored the space of transferable adversarial attacks and proposed a method which measures the dimensionality of the adversarial subspace [43].

On the other side, to improve the robustness of neural networks against adversarial attacks, researchers have proposed various defense methods. Madry et al. proposed adversarial training [10], which is to add adversarial examples constantly in the process of model training and build a model with better robustness. Adversarial training mainly includes naive adversarial training [8], ensemble adversarial training [9], and PGD-based adversarial training [10]. Papernot et al. proposed a distillation training method named defensive distillation [7]. Meng et al. proposed Magnet [44] that uses auto-encoder to improve the model robustness. PixelDefense [45] uses the generated model PixelCNN to transform the adversarial examples into the normal example space, and then feeds the transformed examples into the original model for prediction. However, these defense methods usually were broken soon by new stronger attacks [12], [13], [34], and the long-term arms race between adversarial attack and defense continues, motivating the emerging robustness certification research, including this work.

Robustness Certification. Existing robustness certification techniques generally can be divided into two categories: *complete methods*, which can certify the exact robustness bound but computationally expensive, and *incomplete methods*, which can certify the approximate robustness bound but easy to scale.

For the complete methods, many use Satisfiability Modulo Theories (SMT) ([14], [15], [16], [17]) to certify the robustness of neural networks against adversarial attacks. In addition, integer programming approaches [46], [47], [48] are utilized to verify the robustness. However, they can only be used to certify networks

with small sizes, i.e., limited number of layers and neurons, due to the computation cost.

There have also been a number of recent works to certify neural networks via incomplete methods. Wong et al. [18], [19] used a convex polytope relaxation to bound the robust error or loss that can be reached under norm-bounded perturbation. Raghunathan et al. [20], [21] leveraged Semi-Definite Programming (SDP) relaxation to approximate the adversarial polytope and employed this to train a robust model. Abstract interpretation is also used to verifying neural networks, which can soundly approximate the behavior of neural networks ([32], [49], [50], [51], [52]). More recently, another line of work considers certifying model robustness via randomized smoothing [53], [54]. They provide probabilistic robustness guarantees for smoothed models whose predictions cannot be evaluated exactly, only approximated to arbitrarily high confidence. The above methods are verifying a uniform bound of a test example. To get a more realistic bound, Liu et al. [22] proposed a framework to get the non-uniform bound of a test example. They showed that the non-uniform bounds have larger volumes than that of the uniform bounds. However, the proposed non-uniform bound ignores the inherent heterogeneity of the perturbation direction and it was only evaluated on FCNNs.

Different from existing research, first, `Amoeba` accounts for the heterogeneity of perturbation direction and estimates the real robustness space in a more precise manner. Second, we evaluate our method on more general networks (such as CNNs) along with interesting findings. Finally, we also investigate the broad applications of the robustness space for downstream security-related tasks.

8 Conclusion

In this paper, we present `Amoeba`, the first-of-its-kind framework that efficiently certifies the asymmetric robustness of DNNs. Compared with the alternative certification methods in prior work, `Amoeba` provides quantitative robustness measures with respect to both perturbation direction and magnitude, and estimates such bounds in a more precise manner. We further show that asymmetric robustness entails many security-related use cases including (i) modeling stronger adversarial threats, (ii) explaining DNN predictions, leading to several promising directions for further research.

Acknowledgment

This work was partly supported by the National Key Research and Development Program of China under No. 2020AAA0140004, the Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under No. LR19F020003, and NSFC under No. 61772466 and U1836202.

References

[1] B. Zhao, H. Weng, S. Ji, J. Chen, T. Wang, Q. He, and R. Beyah, "Towards evaluating the security of real-world deployed image captchas," in *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*, 2018, pp. 85–96.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] Y. Sun, X. Wang, and X. Tang, "Sparsifying neural network connections for face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4856–4864.

[4] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.

[5] 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.

[6] I. Evtimov, K. Eykholt, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *CVPR*, 2018.

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

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

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

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

[11] X. Cao and N. Z. Gong, "Mitigating evasion attacks to deep neural networks via region-based classification," in *Proceedings of the 33rd Annual Computer Security Applications Conference*. ACM, 2017, pp. 278–287.

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

[13] ——, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy*. IEEE, 2017, pp. 39–57.

[14] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *International Conference on Computer Aided Verification*. Springer, 2017, pp. 3–29.

[15] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *International Conference on Computer Aided Verification*. Springer, 2017, pp. 97–117.

[16] R. Ehlers, "Formal verification of piece-wise linear feed-forward neural networks," in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2017, pp. 269–286.

[17] N. Carlini, G. Katz, C. Barrett, and D. L. Dill, "Ground-truth adversarial examples," *arXiv preprint arXiv:1709.10207*, 2017.

[18] E. Wong, F. Schmidt, J. H. Metzen, and J. Z. Kolter, "Scaling provable adversarial defenses," in *Advances in Neural Information Processing Systems*, 2018, pp. 8400–8409.

[19] E. Wong and J. Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," *arXiv preprint arXiv:1711.00851*, 2017.

[20] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," *arXiv preprint arXiv:1801.09344*, 2018.

[21] A. Raghunathan, J. Steinhardt, and P. S. Liang, "Semidefinite relaxations for certifying robustness to adversarial examples," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 877–10 887.

[22] C. Liu, R. Tomioka, and V. Cevher, "On certifying non-uniform bound against adversarial attacks," *ICML*, 2019.

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

[24] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1765–1773.

[25] T.-W. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, L. Daniel, D. Boning, and I. Dhillon, "Towards fast computation of certified robustness for relu networks," in *International Conference on Machine Learning*, 2018, pp. 5273–5282.

[26] M. R. Hestenes, "Multiplier and gradient methods," *Journal of optimization theory and applications*, vol. 4, no. 5, pp. 303–320, 1969.

[27] M. J. Powell, "A fast algorithm for nonlinearly constrained optimization calculations," in *Numerical analysis*. Springer, 1978, pp. 144–157.

[28] 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.

[29] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[30] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[31] X. Zhang, N. Wang, H. Shen, S. Ji, X. Luo, and T. Wang, "Interpretable deep learning under fire," in *USENIX Security*, 2020.

[32] G. Singh, T. Gehr, M. Mirman, M. Püschel, and M. Vechev, "Fast and effective robustness certification," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 802–10 813.

[33] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" *arXiv preprint arXiv:1904.12843*, 2019.

[34] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," *arXiv preprint arXiv:1802.00420*, 2018.

[35] C. K. Mummadi, T. Brox, and J. H. Metzen, "Defending against universal perturbations with shared adversarial training," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 4928–4937.

[36] C. Molnar, *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable.*, 2019.

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

[38] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," *arXiv preprint arXiv:1805.12152*, 2018.

[39] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[40] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9185–9193.

[41] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[42] 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)*. IEEE, 2016, pp. 372–387.

[43] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "The space of transferable adversarial examples," *arXiv preprint arXiv:1704.03453*, 2017.

[44] 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*. ACM, 2017, pp. 135–147.

[45] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," *arXiv preprint arXiv:1710.10766*, 2017.

[46] A. Lomuscio and L. Maganti, "An approach to reachability analysis for feed-forward relu neural networks," *arXiv preprint arXiv:1706.07351*, 2017.

[47] V. Tjeng, K. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," *arXiv preprint arXiv:1711.07356*, 2017.

[48] C.-H. Cheng, G. Nührenberg, and H. Ruess, "Maximum resilience of artificial neural networks," in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2017, pp. 251–268.

[49] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev, "Ai2: Safety and robustness certification of neural networks with abstract interpretation," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 3–18.

[50] M. Mirman, T. Gehr, and M. Vechev, "Differentiable abstract interpretation for provably robust neural networks," in *International Conference on Machine Learning*, 2018, pp. 3575–3583.

[51] G. Singh, T. Gehr, M. Püschel, and M. Vechev, "An abstract domain for certifying neural networks," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–30, 2019.

[52] M. Mirman, G. Singh, and M. Vechev, "A provable defense for deep residual networks," *arXiv preprint arXiv:1903.12519*, 2019.

[53] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1310–1320.

[54] J. Jia, X. Cao, B. Wang, and N. Z. Gong, "Certified robustness for top-k predictions against adversarial perturbations via randomized smoothing," *arXiv preprint arXiv:1912.09899*, 2019.

APPENDIX

.1 Algorithms

In this section, we show the pseudo code of modeling stronger adversarial threats mentioned in 5.1.

Algorithm 2: Modeling Stronger Adversarial Threats

Input: p: proportion of disturbed pixels;
B: the robustness bound;
A: the attack algorithm;
X, y: the input examples and labels;
Output: Adversarial Examples: X_{adv}

Initialize the attack examples $X_{adv} \leftarrow X$;
repeat
  Store attack from previous iteration: $X \leftarrow X_{adv}$;
  Generate mask M_1,M_2 based on p and B;
  Update step: $X_{adv} \leftarrow M_1 \odot [(A(X,y)-X)]_+ + M_2 \odot [(A(X,y)-X)]_- + X$, where $A(X,y)-X$ is the adversarial perturbations based on the given attack algorithm;
until ***successful attack***;

.2 Experimental Setup

In this section, we show the experimental setup information, including the configuration of training, model architecture, and attack hyperparameters.

TABLE 5
Configuration for Normal and Adversarial Training

Hyperparameters	Normal Training	Adversarial Training		
Bathch Size	100	100		
Epochs	100	100		
Optimizer	Adam	Adam		
Learning Rate	1e-4	1e-4		
Attack	-	PGD	α	0.005
			ϵ	0.1
			n	20

TABLE 6
Model Architecture of Fully-connected Neural Networks

Architecture	Model		
	MNIST100 FMNIST100	MNIST300 FMNIST300	MNIST500 FMNIST500
Input Layer	784	784	784
Hidden Layer1	FC(100)+ReLU	FC(300)+ReLU	FC(500)+ReLU
Hidden Layer2	FC(100)+ReLU	FC(300)+ReLU	FC(500)+ReLU
Hidden Layer3	FC(100)+ReLU	FC(300)+ReLU	FC(500)+ReLU
Output Layer	10	10	10

TABLE 7
Model Architecture of Convolutional Neural Networks

Architecture	Model	
	MNISTLeNet FMNISTLeNet	SVHNLeNet
Input Layer	(1,28,28)	(3,32,32)
Hidden Layer1	Conv(1,6,3,1,1) ReLU MaxPool(2,2)	Conv(3,6,3,1,1) ReLu MaxPool(2,2)
Hidden Layer2	Conv(6,16,5,1,0) ReLU MaxPool(2,2)	Conv(6,16,5,1,0) ReLu MaxPool(2,2)
Hidden Layer3	Flatten	Flatten
Hidden Layer4	FC(120)+ReLU	FC(120)+ReLU
Hidden Layer5	FC(84)+ReLU	FC(84)+ReLU
Output Layer	10	10

TABLE 8
Attack Hyperparameters

Dataset	Attacks	Hyperparameters
MNIST, FMNIST	FGSM, FGSM^s, FGSM^a	l_∞:ϵ = 0.2 and 0.3
	PGD, PGD^s, PGD^a	l_∞:ϵ = 0.2, α = 0.015 l_2: ϵ = 5.0, α = 0.5 n=20
	C&W, C&W^s, C&W^a	κ=0
SVHN	FGSM, FGSM^s, FGSM^a	l_∞:ϵ = 0.2 and 0.3
	PGD, PGD^s, PGD^a	l_∞:ϵ = 0.2, α = 0.015 l_2: ϵ = 10.0, α = 0.5 n=20
	C&W, C&W^s, C&W^a	l_2: ϵ = 5.0, α = 0.5

.3 More Experimental Results

In this section, we show more experimental results.

TABLE 9
Evaluation Results on MNIST

Model	Attacks	Accuracy	Distance			
			l_∞	l_0	l_2	l_1
MNIST100	FGSM (l_∞), eps = 0.2	20.30%	0.2	393.7	4.17	88.05
	FGSM^s(l_∞), eps = 0.2	20.30%	0.2	393.7	4.17	88.05
	FGSM^a(l_∞), eps = 0.2	20.30%	0.2	375.1	4.08	84.32
	FGSM(l_∞), eps = 0.3	9.15%	0.3	393.7	6.23	130.99
	FGSM^s(l_∞), eps = 0.3	9.15%	0.3	393.7	6.23	130.99
	FGSM^a(l_∞), eps = 0.3	9.15%	0.3	393.7	6.23	130.99
	PGD(l_∞), eps=0.2	6.05%	0.2	405.1	3.99	83.72
	PGD^s(l_∞), eps=0.2	6.05%	0.2	405.1	3.99	83.72
	PGD^a(l_∞), eps=0.2	6.05%	0.2	405.1	3.99	83.72
	PGD(l_2), eps=5.0	6.95%	0.86	452.8	4.97	83.7
	PGD^s(l_2), eps=5.0	**4.80%**	0.92	229.1	4.97	**66.64**
	PGD^a(l_2), eps=5.0	**4.25%**	0.92	229.1	4.97	**66.65**
	C&W(l_2), κ = 0	0.00%	0.3	377.6	1.62	25.45
	C&W^s(l_2), κ = 0	0.00%	0.51	77.39	2.18	18.61
	C&W^a(l_2), κ = 0	0.00%	0.48	70.32	2.08	18.21
MNIST300	FGSM (l_∞), eps = 0.2	26.45%	0.2	383.6	4.14	86.47
	FGSM^s(l_∞), eps = 0.2	26.45%	0.2	383.6	4.14	86.47
	FGSM^a(l_∞), eps = 0.2	26.45%	0.2	383.6	4.14	86.47
	FGSM(l_∞),eps = 0.3	8.95%	0.3	383.6	6.17	128.63
	FGSM^s(l_∞), eps = 0.3	8.95%	0.3	383.6	6.17	128.63
	FGSM^a(l_∞), eps = 0.3	8.95%	0.3	383.6	6.17	128.63
	PGD(l_∞), eps=0.2	11.25%	0.2	402.3	4.01	84.16
	PGD^s(l_∞), eps=0.2	11.25%	0.2	402.3	4.01	84.16
	PGD^a(l_∞), eps=0.2	11.25%	0.2	402.3	4.01	84.16
	PGD(l_2), eps=5.0	11.90%	0.88	482.8	4.97	85.93
	PGD^s(l_2), eps=5.0	9.85%	1.39	53.78	4.97	33.51
	PGD^a(l_2), eps=5.0	8.70%	1.31	52.9	4.97	35.35
	C&W(l_2), κ = 0	0.00%	0.38	376.3	1.89	28.89
	C&W^s(l_2), κ = 0	0.00%	0.59	77.7	2.37	20.69
	C&W^a(l_2), κ = 0	0.00%	0.54	72.72	2.27	20.48
MNIST500	FGSM (l_∞), eps = 0.2	27.75%	0.2	371.6	4.1	85.1
	FGSM^s(l_∞), eps = 0.2	27.75%	0.2	371.6	4.1	85.1
	FGSM^a(l_∞), eps = 0.2	27.75%	0.2	371.6	4.1	85.1
	FGSM(l_∞), eps = 0.3	6.95%	0.3	371.6	6.13	126.61
	FGSM^s(l_∞), eps = 0.3	6.95%	0.3	371.6	6.13	126.61
	FGSM^a(l_∞), eps = 0.3	6.95%	0.3	371.6	6.13	126.61
	PGD(l_∞), eps=0.2	12.95%	0.2	400.6	4.02	84.29
	PGD^s(l_∞), eps=0.2	12.95%	0.2	400.6	4.02	84.29
	PGD^a(l_∞), eps=0.2	12.95%	0.2	400.6	4.02	84.29
	PGD(l_2), eps=5.0	10.40%	0.94	477.9	4.97	83.37
	PGD^s(l_2), eps=5.0	9.30%	1.42	55.62	4.97	33.78
	PGD^a(l_2), eps=5.0	7.10%	1.35	53.69	4.97	35.6
	C&W(l_2), κ = 0	0.00%	0.41	371.8	1.99	29.93
	C&W^s(l_2), κ = 0	0.00%	0.61	78.7	2.46	21.53
	C&W^a(l_2), κ = 0	0.00%	0.57	71.9	2.34	21.24
MNISTLeNet	FGSM (l_∞), eps = 0.2	67.95%	0.2	395.6	4.19	88.59
	FGSM^s(l_∞), eps = 0.2	67.95%	0.2	395.6	4.19	88.59
	FGSM^a(l_∞), eps = 0.2	67.95%	0.2	395.6	4.19	88.59
	FGSM(l_∞),eps = 0.3	45.15%	0.3	395.6	6.26	131.8
	FGSM^s(l_∞), eps = 0.3	45.15%	0.3	395.6	6.26	131.8
	FGSM^a(l_∞), eps = 0.3	45.15%	0.3	395.6	6.26	131.8
	PGD(l_∞), eps=0.2	38.75%	0.2	462.5	3.69	77.03
	PGD^s(l_∞), eps=0.2	38.75%	0.2	462.5	3.69	77.03
	PGD^a(l_∞), eps=0.2	38.75%	0.2	462.5	3.69	77.03
	PGD(l_2), eps=5.0	**20.84%**	1.0	565.9	4.96	**86.5**
	PGD^s(l_2), eps=5.0	**19.60%**	1.25	126.5	4.91	**47.53**
	PGD^a(l_2), eps=5.0	**15.35%**	1.35	134.9	4.95	**45.95**
	C&W(l_2), κ = 0	**0.00%**	0.58	415.3	2.22	**28.8**
	C&W^s(l_2), κ = 0	**0.00%**	0.87	72.64	2.79	**22.71**
	C&W^a(l_2), κ = 0	**0.00%**	1.11	57.48	3.04	**19.46**

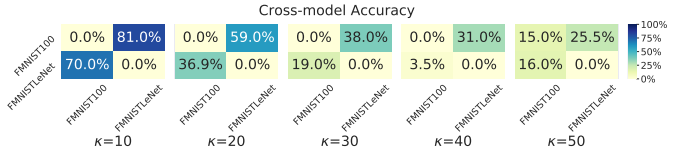
[1] We take all the normal models trained on Section 4.2.2 as the target models. Refer to Tables 1, 5, and 6, 7 for more details about the normal models.
[2] For simplicity, we omit the standard deviation since it's small value.

TABLE 10
Evaluation Results on FMNIST

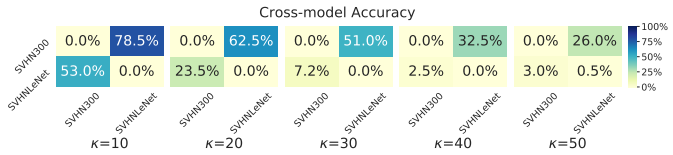
Model	Attacks	Accuracy	Distance			
			l_∞	l_0	l_2	l_1
FMNIST100	FGSM (l_∞), eps = 0.2	7.50%	0.2	394.5	4.77	114.8
	FGSM^s(l_∞), eps = 0.2	7.50%	0.2	394.5	4.77	114.8
	FGSM^a(l_∞), eps = 0.2	7.50%	0.2	376	4.67	110.4
	FGSM(l_∞), eps = 0.3	4.05%	0.3	394.5	7.06	169.44
	FGSM^s(l_∞), eps = 0.3	4.05%	0.3	376.2	6.94	163.5
	FGSM^a(l_∞), eps = 0.3	4.05%	0.3	376	6.92	162.95
	PGD(l_∞), eps=0.2	0.80%	0.2	399.9	4.36	101.2
	PGD^s(l_∞), eps=0.2	0.80%	0.2	373.1	4.31	98.43
	PGD^a(l_∞), eps=0.2	0.80%	0.2	371.9	4.29	98.02
	PGD(l_2), eps=5.0	3.35%	0.77	407.25	4.98	95.23
	PGD^s(l_2), eps=5.0	2.75%	0.84	242.4	4.98	78.41
	PGD^a(l_2), eps=5.0	2.35%	0.84	241.8	4.98	77.72
	C&W(l_2), κ = 0	0.00%	0.16	331.6	0.86	15.64
	C&W^s(l_2), κ = 0	0.00%	0.27	69.23	1.12	10.55
	C&W^a(l_2), κ = 0	0.00%	0.26	67.57	1.11	10.51
FMNISTLeNet	FGSM (l_∞), eps = 0.2	12.05%	0.2	396.8	4.78	115.41
	FGSM^s(l_∞), eps = 0.2	12.05%	0.2	396.8	4.78	115.41
	FGSM^a(l_∞), eps = 0.2	12.05%	0.2	396.8	4.78	115.41
	FGSM(l_∞),eps = 0.3	4.15%	0.3	396.8	7.08	170.3
	FGSM^s(l_∞), eps = 0.3	4.15%	0.3	396.8	7.08	170.3
	FGSM^a(l_∞), eps = 0.3	4.15%	0.3	396.8	7.08	170.3
	PGD(l_∞), eps=0.2	1.40%	0.2	430.2	4.01	91.69
	PGD^s(l_∞), eps=0.2	1.40%	0.2	430.2	4.01	91.69
	PGD^a(l_∞), eps=0.2	1.40%	0.2	430.2	4.01	91.69
	PGD(l_2), eps=5.0	0.05%	0.88	435.9	4.95	90.76
	PGD^s(l_2), eps=5.0	0.05%	0.91	331.7	4.95	83.9
	PGD^a(l_2), eps=5.0	0.00%	0.91	331.7	4.95	83.4
	C&W(l_2), κ = 0	0.00%	0.19	348.6	0.99	17.17
	C&W^s(l_2), κ = 0	0.00%	0.46	71.8	1.58	13.99
	C&W^a(l_2), κ = 0	0.00%	0.52	46.79	1.66	11.68

TABLE 11
Evaluation Results on SVHN

Model	Attacks	Accuracy	Distance			
			l_∞	l_0	l_2	l_1
SVHN300	FGSM (l_∞), eps = 0.2	0.25%	0.2	1528.2	10.96	604.6
	FGSM^s(l_∞), eps = 0.2	0.25%	0.2	1528.2	10.96	604.6
	FGSM^a(l_∞), eps = 0.2	0.25%	0.2	1528.2	10.96	604.6
	FGSM(l_∞), eps = 0.3	0.10%	0.3	1528.2	16.29	895.9
	FGSM^s(l_∞), eps = 0.3	0.10%	0.3	1402.7	15.6	822.5
	FGSM^a(l_∞), eps = 0.3	0.10%	0.3	1337.91	15.23	784.3
	PGD(l_∞), eps=0.2	0.00%	0.2	1506.9	9.88	521.5
	PGD^s(l_∞), eps=0.2	0.00%	0.2	945.5	7.84	332.2
	PGD^a(l_∞), eps=0.2	0.00%	0.2	940.3	7.78	327.2
	PGD(l_2), eps=10.0	19.6%	0.6	1939.2	9.21	418.6
	PGD^s(l_2), eps=10.0	18.10%	0.65	1260.3	9.19	344.7
	PGD^a(l_2), eps=10.0	17.7%	0.71	1285.5	9.17	337.8
	C&W(l_2), κ = 0	0.00%	0.06	1034.9	0.72	26.42
	C&W^s(l_2), κ = 0	0.00%	0.11	243.6	0.95	18.51
	C&W^a(l_2), κ = 0	0.00%	0.11	236.2	0.94	18.36
SVHNLeNet	FGSM (l_∞), eps = 0.2	2.55%	0.2	1511.3	10.91	599.5
	FGSM^s(l_∞), eps = 0.2	2.55%	0.2	1511.3	10.91	599.5
	FGSM^a(l_∞), eps = 0.2	2.55%	0.2	1511.3	10.91	599.5
	FGSM(l_∞),eps = 0.3	2.80%	0.3	1511.3	16.22	888.6
	FGSM^s(l_∞), eps = 0.3	2.80%	0.3	1455.1	15.9	854.4
	FGSM^a(l_∞), eps = 0.3	2.80%	0.3	1451.5	15.8	852.6
	PGD(l_∞), eps=0.2	0.00%	0.2	1458.7	7.71	373.1
	PGD^s(l_∞), eps=0.2	0.00%	0.2	939.5	6.28	246.9
	PGD^a(l_∞), eps=0.2	0.00%	0.2	938.1	6.26	245.7
	PGD(l_2), eps=10.0	0.15%	0.93	1557.1	9.03	327.7
	PGD^s(l_2), eps=10.0	0.10%	1.12	466.4	8.56	184.5
	PGD^a(l_2), eps=10.0	0.05%	1.17	466.7	8.51	178.6
	C&W(l_2), κ = 0	0.00%	0.11	1052.7	0.68	19.07
	C&W^s(l_2), κ = 0	0.00%	0.14	254.8	0.78	13.58
	C&W^a(l_2), κ = 0	0.00%	0.15	247.6	0.71	12.87



(a) FMNIST



(b) SVHN

Fig. 8. Cross-model accuracy of adversarial examples generated by C&W attack on FMNIST and SVHN.

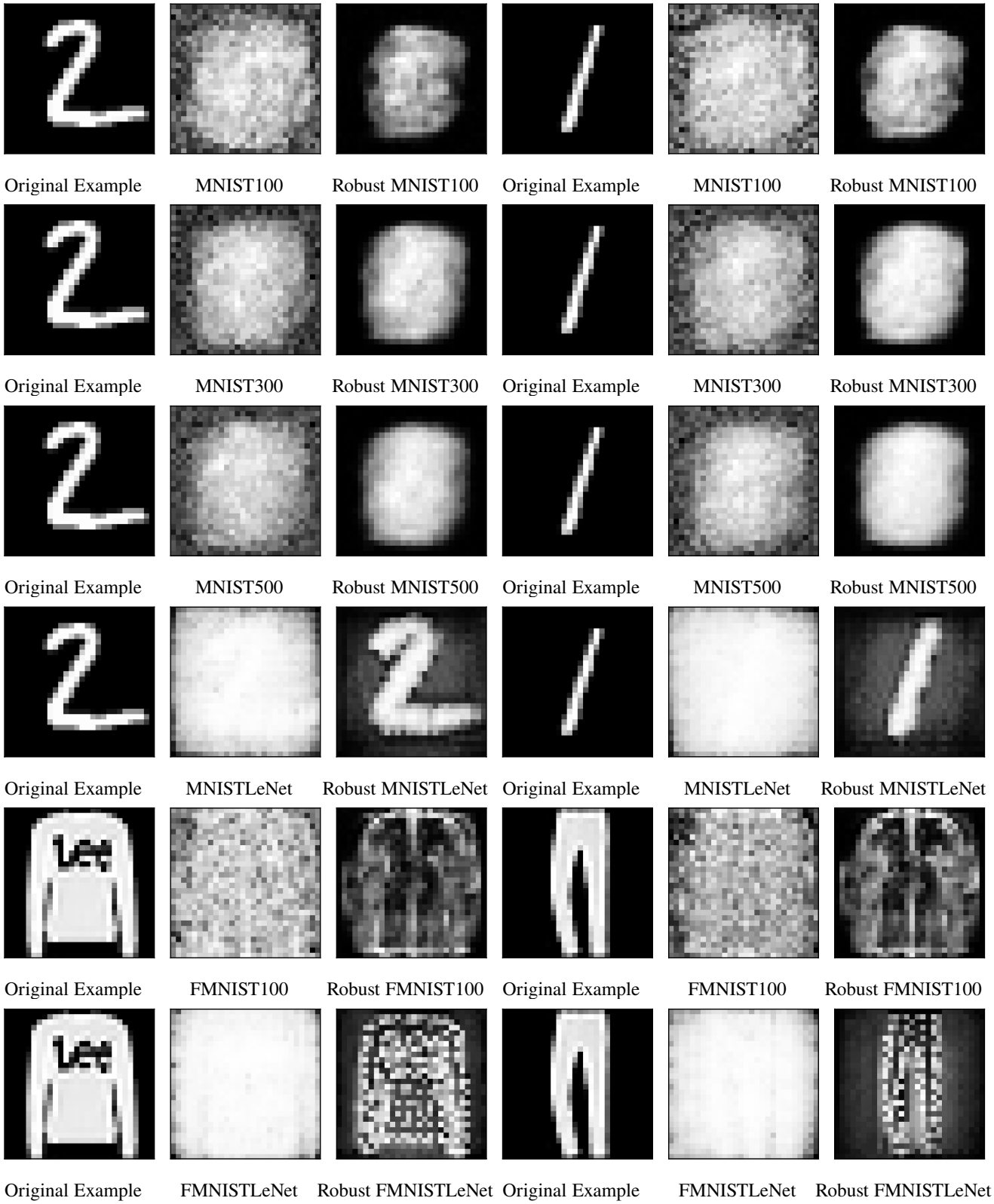


Fig. 9. Visualization of the asymmetric robustness bounds of examples on MNIST and FMNIST.

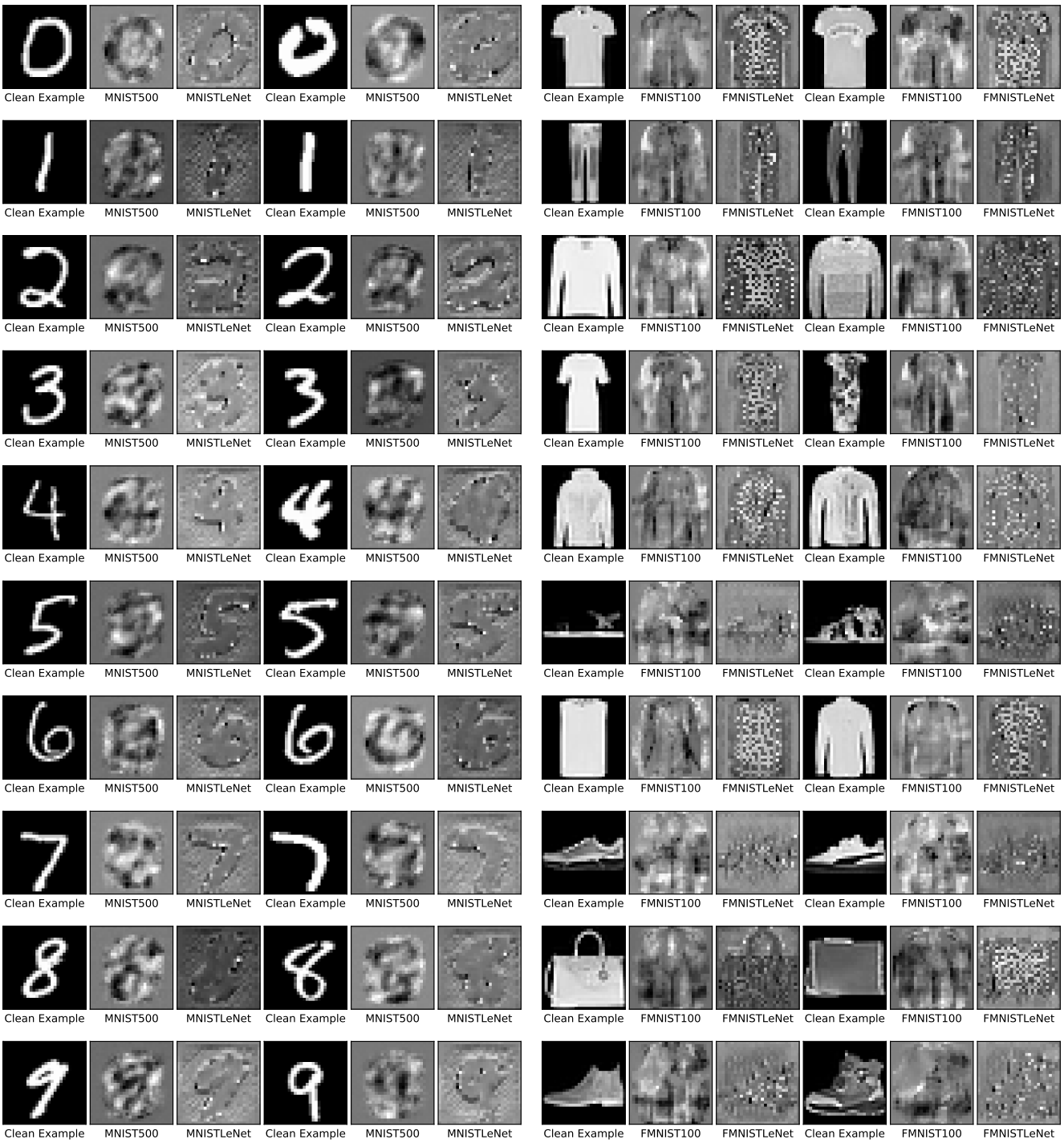
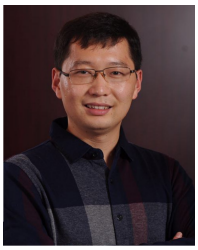


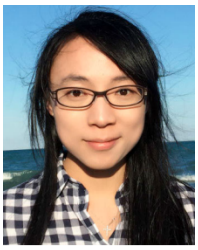
Fig. 10. Explanation results on MNIST and FMNIST.

Changjiang Li is currently a Ph.D. student in the College of Information Science and Technology at Pennsylvania State University. He received the Master degree from the School of Computer Science at Zhejiang University in 2020. His research interest includes Adversarial Machine Learning, AI privacy.



Shouling Ji is a ZJU 100-Young Professor in the College of Computer Science and Technology at Zhejiang University and a Research Faculty in the School of Electrical and Computer Engineering at Georgia Institute of Technology. He received a Ph.D. in Electrical and Computer Engineering from Georgia Institute of Technology, a Ph.D. in Computer Science from Georgia State University. His current research interests include AI Security, Data-driven Security, Privacy and Data Analytics. He is a member of IEEE and ACM and was the Membership Chair of the IEEE Student Branch at Georgia State (2012- 2013).

Haiqin Weng is with the Ant Group, Hangzhou, China. Her research interests include AI security, anomaly detection, and machine learning. She received her Ph.D. degree in College of Computer Science and Technology at Zhejiang University in 2019, and received her BS degree from South China University of Technology in 2014.



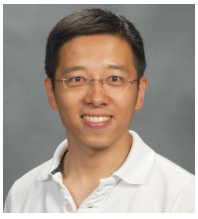
Bo Li is currently an Assistant Professor with the Department of Computer Science, University of Illinois at Urbana–Champaign. She has designed several robust learning algorithms against adversarial behaviors, a scalable framework for achieving robustness for a range of learning methods, and a privacy preserving data publishing systems. Her research interests include theoretical and practical aspects of security, machine learning, privacy, game theory, adversarial deep learning, generative models, and designing scalable robust machine learning models against unrestricted adversarial attacks. Her work has been featured by major publications and media outlets, such as Nature, Wired, Fortune, and the IEEE Spectrum. She was a recipient of the Symantec Research Labs Fellowship and the MIT Technology Review TR-35 Award.

Jie Shi is a Principal Researcher in Huawei Singapore Research Center. His research interests include trustworthy AI, machine learning security, data security and privacy, IoT security and applied cryptography. He has over 10 years' research experience and has published over 30 research papers in refereed journals and international conferences. He received his Ph.D degree from Huazhong University of Science and Technology, China.

Raheem Beyah, a native of Atlanta, GA, serves as Georgia Tech's Vice President for Interdisciplinary Research, Executive Director of the Online Masters of Cybersecurity program (OMS Cybersecurity), and is the Motorola Foundation Professor in School of Electrical and Computer Engineering. He is also Co-Founder of Fortiphyd Logic , Inc. Raheem received his Bachelor of Science in Electrical Engineering from North Carolina A&T State University in 1998. He received his Masters and Ph.D. in Electrical and Computer Engineering from Georgia Tech in 1999 and 2003, respectively. His research interests include Network security and monitoring, Cyber-physical Systems Security, Network traffic characterization and performance, and Critical infrastructure security. He received the National Science Foundation CAREER award in 2009 and was selected for DARPA's Computer Science Study Panel in 2010. He is a member of AAAS, ASEE, a lifetime member of NSBE, a senior member of IEEE, and an ACM Distinguished Scientist.

Shanqing Guo is currently a professor with the School of Cyber Science and Technology at Shandong University. He received his M.S. and Ph.D. degrees in computer science from Ocean University, China, in 2003, and Nanjing University, China, in 2006 respectively. He joined the School of Computer Science and Technology at Shandong University as an assistant professor in 2006. His research interests include AI Security, Software and System Security. He has published in S&P, ICDE, ACSAC, RAID, DSN, CIKM, ICSME, ISSRE and other venues. He also serves as a program committee member or a reviewer for various international conferences and journals, e.g., ISSRE, ICSME and Computer&Security.

Zonghui Wang is a senior engineer in the college of Computer Science and Engineering at Zhejiang University in Hangzhou, China. He received his Ph.D. in the college of Computer Science and Engineering at Zhejiang University in 2007. His current research interests focus on deep learning, AI security, cloud computing and computer architecture.



Ting Wang received the Ph.D. degree from the Georgia Institute of Technology. He is currently an Assistant Professor with the College of Information Science and Technology, Pennsylvania State University. He conducts research at the intersection of machine learning and privacy and security. His ongoing work focuses on making machine learning systems more practically usable through mitigating security vulnerabilities, enhancing privacy awareness, and increasing decision-making transparency.