AuxBlocks: Defense Adversarial Example via Auxiliary Blocks

Yueyao Yu, Pengfei Yu and Wenye Li
The Chinese University of HongKong,Shenzhen
yueyaoyu@link.cuhk.edu.cn, pengfeiyu@link.cuhk.edu.cn, wyli@cuhk.edu.cn

Abstract—Deep learning models are vulnerable to adversarial examples, which poses an indisputable threat to their applications. However, recent studies observe gradient-masking defenses are self-deceiving methods if an attacker can realize this defense. In this paper, we propose a new defense method based on appending information. We introduce the Aux Block model to produce extra outputs as a self-ensemble algorithm and analytically investigate the robustness mechanism of Aux Block. We have empirically studied the efficiency of our method against adversarial examples in two types of white-box attacks, and found that even in the full white-box attack where an adversary can craft malicious examples from defense models, our method has a more robust performance of about 54.6% precision on Cifar10 dataset and 38.7% precision on Mini-Imagenet dataset. Another advantage of our method is that it is able to maintain the prediction accuracy of the classification model on clean images, and thereby exhibits its high potential in practical applications.

Index Terms—adversarial example, defense method, white-box attack

I. INTRODUCTION

Breakthroughs in deep neural networks (DNN) learning have achieved exceptional performance across a wide range of applications, such as in image classification [11] and in speech recognition [9]. However, recent studies showed that neural networks are vulnerable to adversarial examples, and attackers can design maliciously perturbed inputs to mislead a model at the test phase [2], [12], [20]. Furthermore, these inputs can transfer across different models, and one example can attack several models without knowing model structures or parameters [20]. This threatens DNN applications in security-sensitive systems such as in self-driving cars and in identity authentication.

Specifically, the phenomenon of adversarial examples has inspired researchers to defense adversarial examples as a defender; meanwhile others play the part in attackers to design threatening attacks with slight perturbations. For example, a natural image \( x \) is correctly classified by a machine learning model, while an adversarial example \( x' \) is similar to \( x \) but has been predicted as belonging to a different category. To produce such an \( x' \), an algorithm can be designed by minimizing the perturbation and making adversarial samples imperceptible to a human (ref. Fig. 1).

Adversarial examples are constructed to mislead the predicted output. To tackle the problem, a natural idea is to output multiple results instead of one. In this paper, we follow this idea and introduce the **Aux Block** model. In our model, a new block expands the original model; and we propose several Aux Blocks, which is like a self-ensemble model (ref. Fig. 2).

We propose applicable Aux Block structures and show that Aux Block improves the network robustness against white-box attacks. Additionally, we cogitate the full white-box attacks (called as adaptive white-box attack), with which an adversary knows defense details thoroughly. Our algorithm can still work against the irresistible adversarial examples.

Specifically in our work, we have made the following contributions:

1) We propose a novel information-appending defense method for improving the robustness of neural networks to perturbation. The key idea is to affix Aux Blocks in some convolution layers. This algorithm can be regarded as building a self-ensemble model.

2) We propose two Aux Block algorithms: a basic one and a score-based one. We show empirically that Aux Blocks could improve the robustness of model against static white-box attacks compared to the adversarial training model.

3) We investigate the method’s appearances in adaptive white-box attacks. Our method still has a robust performance even when facing with the Adp-FGSM attack to bypass defenses.

The motivation of our method is straightforward and intuitive. In practice, the method can be easily implemented and applied in different scenarios. To evaluate the performance of the proposed models, we have carried out a variety of experiments. Simple as it is, our approach has reported significantly improved results over a number state-of-the-art solutions, and therefore exhibited its high potential as a practical and effective solution to real problems.

The paper is organized as follows. Section 2 introduces the background and related work. Section 3 presents our model. Section 4 reports our experimental results, followed by conclusions and discussions in Section 5.

II. BACKGROUND

A. Neural Networks

For a clean image \( x \), denote by \( f(\cdot) \) a neural network, and \( y \) the class label. The feature vector \( f^{i-1} \) is an input of the \( i \)-th layer for a feed-forward network. Activation functions such

The work is partially supported by Project No. xxxxxx.
as ReLU [16] mix in some layers to make model non-linear. Thus the $i$-th layer computes:

$$f^i(x) = \text{ReLU}(w^i f^{i-1}(x) + b^i),$$

where $w^i$ is a matrix and $b^i$ is a vector. In the final layer, the predicted probability of $y_i$ is $p(y_i|x)$ and the predicted class is given by

$$y = \arg \max_i p(y_i|x).$$

The loss function of a classifier for input $x$ and target $y$ is $J(x,y)$.

**B. Adversarial Examples**

Adversarial examples [20] are inputs crafted by attackers to mislead machine learning models. The malicious input $x' = x + \sigma$ and $\sigma$ is a slightly perturbation with $\|\sigma\| < \epsilon$, where $\epsilon$ is so small that it makes no visual difference between $x$ and $x'$ for human being but deep neural networks will be fooled such that $f(x') \neq y_{true}$.

Most attacks use $L_p$-norm distance matrix to define the magnitude of $\sigma$, i.e.: $L_0$, the number of permuted pixels [17]; $L_2$, the Euclidean distance between $x$ and $x'$ [6], [15], [20]; $L_\infty$, a measure of the maximum absolute change in any pixel [12]. A permuted image will be similar to clean image visually if any of three distance metrics is small.

Szegedy [20] first proposed to find adversarial examples in DNN models, he used L-BFBS algorithm to find them and observed that they catastrophically destroyed DNN models. They also observed that the transferability property of adversarial perturbations, one generated from an arbitrary model can also fool other models to produce incorrect outputs even if they are trained on different datasets with completely different structures. This phenomenon makes black-box attacks feasible.

Next, we discuss several attacking methods in our experiments.

**Fast Gradient Sign Method: FGSM**

Goodfellow et al. [7] proposed the FGSM method for crafting adversarial examples. FGSM is un-targeted and uses the same attack strength at every dimension:

$$x_{FGSM} = x + \epsilon \text{sign}(\nabla_x J(x,y))$$

The above equation increases the loss function $J(\cdot)$ by adding a transformed gradient to input $x$, where $\epsilon$ is small enough to be undetectable.

There are several variants such as R+FGSM [21], BIM [12], and so on.

**Projected Gradient Descent: PGD**

Madry et al. [14] reviewed adversarial robustness in a max-min saddle point problem and suggested that BIM (Basic Iterative Method applies FGSM multiple times with small step size) [12] is a projected gradient descent method essentially. In the following sections, we will replace BIM by PGD. They conjectured that Projected Gradient Descent (PGD) is the strongest attack utilizing the local first order information about the network [14].

$$x_{N+1} = \Pi_{x+S}(x_N + \alpha(\nabla_x J(x,y)))$$

where $\alpha$ controls the magnitude of the perturbation in each iteration.

**Carlini and Wagner’s Attacks**

Carlini and Wagner proposed a high success rate method – optimization-based attack algorithms [6] to craft adversarial with low distortion. There are three versions based on different measures: $L_2$, $L_0$ and $L_\infty$. However, Carlini and Wagner’s attack is computationally expensive compared to FGSM. In our paper, we select Carlini and Wagner’s $L_2$ attack.

**Boundary Attacks**

Wieland et al. [3] introduced a decision-based attack which dose not rely on gradient of the model but the final decision boundary of the model. A random perturbation are crafted at first, reduced iteratively in the adversarial region. We will use this algorithm as a gradient-free attack.
C. Defenses Methods

Adversarial Training is one of the most popular defense strategies, which is to train a model using both clean data and adversarial data [7], [20] to improve robustness. We study the adversarial training algorithm in [14] which solves a min-max problem:

$$\theta^* = \arg \min_{\theta} \mathbb{E}_{x,y \in X} \left[ \max_{\sigma \in [-\epsilon, \epsilon] \mathbb{N}} J(x + \sigma, y) \right].$$

The authors solved the inner problem by crafting adversarial examples with projected gradient descent attacks.

Previous work has observed the phenomenon of gradient masking [1], which refers to a defense model with useless gradients by proposing a non-differentiable model [4] or representing data with less details [24]. Some defenses such as BRELU and MagNet are experimentally defeated with a small distortion [5]. Defensive Distillation [18] is also one of the most famous attack-agnostic techniques, which aims to make NN models more robust against all attacks but this method is not robust under Carlini and Wagner’s attack [6]. Moreover, most defenses obfuscate gradients somewhat, which indicate that the adversarial attack is still an unsolved problem and open for study.

D. Threat Model

In our setting, we assume that an attacker have all knowledge of a model, such as the model parameters and model architectures in white-box attacks. Moreover, we introduce a clever attacker who is assumed to be aware of defenses that may be used.

As in Warren et al. [8], we consider two white-box attacks (ref. Table I):

| Threat Model: Adversary Knows $\theta_{original}$ AND/OR $\theta_{defense}$ |
|-------------------------------------------------|
| Black-box attack | unknown | unknown |
| Static attack   | known   | unknown |
| Adaptive attack | known   | known   |

III. Method

In this section, we illustrate our Aux Block model to improve neural network’s robustness. We will first introduce our algorithm and motivations, and then discuss some observations behind our algorithm.

Ensemble of several different models can improve robustness against malicious inputs. However, the straits of conventional ensemble methods are also apparent: storing and reloading $k$ different models is a huge burden on memory consumption, and tuning parameters in each model is a tricky task. In this paper, we propose an algorithm that is able to generate several auxiliary classifiers in one model with less memory cost and can avoid additional parameter tuning process.

A. Model

Our main idea is to introduce Aux Block and then divide the model into two versions: a public model and a private model. More specifically, the public model is a standard convolutional neural network and the private is several branching auxiliary models as shown in Fig. 2. The public model can be revealed to attackers while the private model is confidential to them, so malicious hackers cannot create valid examples easily. An aux Block can be any structure, and in this paper we propose a tiny neural network.
Algorithm 1 Training and Testing of Aux Block

Require: $m$ Aux Blocks as $f_{A_1}, f_{A_2}, \cdots, f_{A_m}$ and an original CNN model as $f$.

Training Phase:
\begin{itemize}
  \item for $i = 1, 2, \cdots$ do
    \begin{itemize}
      \item Randomly sample $(x_i, y_i)$ in dataset
      \item Compute $loss = \sum_{j=1}^{m} l(f_{A_j}(x_i), y_i) + l(f(x_i), y_i)$
      \item Compute $\Delta w = \nabla_w loss$
      \item Update $w \leftarrow w - \Delta w$
    \end{itemize}
  \end{itemize}

Testing Phase:
\begin{itemize}
  \item A testing image $x$
  \item for $j = 1, 2, \cdots, m + 1$ do
    \begin{itemize}
      \item Calculate probability output of each Aux Block and the original model:
        \[ p_j = f_{A_j}(x), \text{if } j < m + 1 \text{ or } p_j = f(x), \text{if } j = m + 1 \]
      \item Calculate each output $y^j = \arg\max_k p_k$
    \end{itemize}
  \end{itemize}

Predict the class: $\hat{y}$ with $\hat{y} = \arg\max_k Y_k$.

Auxiliary network architectures are well-designed for several different datasets in the context of image classification. Details about the architectures will be discussed concretely in the next section. We denote an integral model as $f$, $x$ is the input image, and the output is an $n$-dimensional vector instead of a single value: $y = [y_1, y_2, y_3, \cdots, y_m+1], y$ in $\mathbb{R}^m$ ($m$ is the number of Aux Blocks $A$), $y_1$ is the output of the public network, while $y_2, y_3, \cdots, y_m+1$ is from Aux Blocks. Since the malevolent examples are crafted without auxiliary information, predicted outputs are less likely to be misled in Aux Blocks. Therefore our model can be deployed in security-sensitive areas.

The new cross-entropy loss function is rewritten as:
\[ Loss = -\sum_{i=1}^{n} \alpha_i(\sum_{j=1}^{T} y_{ij} \cdot \log P_{ij}), \]
where $T$ is the number of class labels, $y_{ij}$ denotes the value of label $j$ in classifier $i$ and $P_{ij}$ is classifying possibility for $y_{ij}$. Since the new loss function is an aggregation function from each classifier, $\alpha_i \in \mathbb{R}$ is a parameter which controls the relative weight of $i$ classifier in the total loss, as the confidence of an Aux Block in the final decision. In this paper, we use $\alpha = 1$ for all Aux Blocks.

Algorithm 2 A score-based Aux Block algorithm

Similar steps as Algorithm 1

Predict the class: $\hat{y}$ with $\hat{y} = \arg\max_s s$

Our algorithm is listed in Algorithm 1. In the algorithm, we select the class $k$ which most Aux Blocks support as the final output. There are also other measures, such as a score-based algorithm with which the only difference is on selecting the final output. In Algorithm 2, we implement the same training phase but in the testing phase. We calculate values of each class and select the maximum value. Compared with
Algorithm 1 Algorithm 2 evaluates probability of each class more discretely and considers that an Aux Block with larger \( \alpha \) has a large weight in testing. We apply Algorithm 1 when \( \alpha_i = 1, (i = 1, \cdots, m) \). Algorithm 1 becomes equivalent to Algorithm 2.

B. Discussion

In the static adversary, adversarial examples are crafted based on the non-defense model. So there can be two totally different approaches to defense these attacks. One is to reduce some necessary information, with which some existing defenses have used \([23]\). However, a defense performs well in a static adversary may be stock in the gradient masking phenomenon. Anish and Nicholas \([1]\) observed that defending models based on gradient obfuscation is a false orientation.

Instead of information-reducing algorithms, our method increases the information by a combination output from all Aux Blocks and the original network. We do not obfuscate gradients in the original model, but an adversarial example crafted from the original model are blind to the extra knowledge in Aux Blocks.

In the adaptive adversary, masking gradient methods are like self-deceived. When an attacker realizes the defense, he or she can bypass the defense effortlessly then the defense has no any effect for adversarial examples. We propose a new attack Adp-FGSM to craft malicious examples from defense models straight. Our method can still improve robustness in this adversary. Though an attack crafts an adversarial image for our approach, the perturbation aims to mislead one Aux Block can be right classified by other Aux Blocks. We will give more explanations about this in Section 4.4.

IV. Experiments

A. Experiments setups

1) Datasets: Our experiments are performed on three datasets:

1) MNIST \([13]\) contains 70,000 samples: 60,000 training samples and 10,000 testing samples. Each sample is a grayscale image with \( 28 \times 28 \) pixels. The number of possible classes is 10.

2) CIFAR10 \([10]\) contains 60,000 images in 10 classes: 50,000 training images and 10,000 testing images with \( 32 \times 32 \) pixels in RGB color channels.

3) We also test on Mini-Imagenet dataset with 10 classes randomly chosen from the 100 MiniImagenet classes \([22]\) (each class with 600 images), on which a VGG16 model has been trained on 5000 images (500 images in each class) with 87.8 % top-1 accuracy.

We consider two convolutional neural networks: LeNet5 \([13]\) and VGG16 \([19]\). The activation function in each network is RELU and loss function is cross-entropy function.

2) Defense: Adversarial training (AT) with adversarial examples crafted by PGD attacks \([14]\). On the MNIST dataset, we pre-train an AT model with the maximum perturbation \( \epsilon = 0.2 \), step size \( \alpha = 0.02 \), while on both CIFAR10 dataset and Mini-Imagenet dataset, \( \epsilon \) is 8/255 and \( \alpha = 2/255 \).

3) Attacking Methods: In the static white-box adversary, we use the following attacks.

1) We implement PGD with 5 iterations: \( \alpha = 0.1, \epsilon = 0.3 \) on MNIST, \( \alpha = 2/255, \epsilon = 8/255 \) on CIFAR10 and \( \alpha = 3/255, \epsilon = 10/255 \) on Mini-Imagenet.

2) We implement Carlini and Wagner’s \( L_2 \) attack as a targeted attack.

3) We implement Boundary attacks as a gradient-free attack.

While in the adaptive adversary, we use the adaptive attacks. We modify FGSM and PGD to Adp-FGSM and Adp-PGD for the adaptive adversary. \( x_{adv} \) are crafted from:

\[
x_{adv} = x_{adv} + \epsilon(\nabla x_{adv} J_{adv}(x_{adv}, y)),
\]

where \( x_{adv} \) and \( J_{adv} \) are re-designed on this particular defense.

Adversarial Training. We re-assess that this algorithm transforms the original inputs to adversarial inputs, therefore we use the new input \( x_{adv} = x + \Delta x \), where \( \Delta x \) is the perturbation for Adversarial Training.

Aux Block. We use the loss function defined in Section 3 to fool all Aux Blocks and the original model together:

\[
J_{adv}(x, y) = \sum_{j=1}^{m} l(f_{A_j}(x), y) + l(f(x), y).
\]

Fig 5 shows the adversarial examples from Adp-PGD.

4) Implementation Details: The training hyper-parameters for all models selected are in Table III. We use a SGD optimizer during the training phase. Especially, we run pre-trained models for 5 epochs to get adversarial training models.

B. Aux Block Details

In this section, We review VGG16 structures as \([64,64,M, 128,128,M, 256,256, 256, M, 512,512, 512, M],[22]\). We craft our Aux Block in Table IV. Feature maps in each convolution layer are our inputs in an Aux Block. Since the feature size shrinks after filtering each maxpooling layer, we use a flexible Aux Block structure: the number of our Aux Block layer decreases if the size of a feature map diminishes.

As mentioned above, the three Aux Blocks have different structures: \( \text{Block1} \) is same as \( \text{Aux Block} \) in Table IV while \( \text{Block2, Block3} \) is in Table V. \( \text{Block3} \) only has a max pooling layer since the feature size is \( 2 \times 2 \).
Fig. 4. The first row shows the accuracy curves of Adp-FGSM and Adp-PGD on MNIST dataset. The second row shows the curves on CIFAR10 dataset.

### TABLE II
COMPARISONS OF ACCURACIES ON MNIST, CIFAR10 AND MINI-IMAGENET DATASETS,

| Dataset          | Method                  | Clean | PGD  | CW   | Boundary Attack |
|------------------|-------------------------|-------|------|------|-----------------|
| MNIST            | NA                      |       | 21.26| 0.0  | 0.2             |
|                  | Adversarial Training    | 98.76 | 89.63| 58.2 | 97.9            |
|                  | Aux Block(all)          | 98.49 | 88.2 | 68.4 | 96.6            |
|                  | Aux Block(private)      | 98.21 | 89.59| 76.8 | 97.7            |
| CIFAR10          | NA                      |       | 12.30| 0.0  | 5.4             |
|                  | Adversarial Training    | 83.89 | 78.82| 76.7 | 83.5            |
|                  | Aux Block(all)          | 91.72 | 74.44| 77.1 | 87.1            |
|                  | Aux Block(private)      | 89.74 | 77.72| 77.5 | 90.0            |
| Mini-Imagenet    | NA                      | 87.8  | 4.6  | 0.0  | -               |
|                  | Adversarial Training    | 72.9  | 68.5 | 75.2 | -               |
|                  | Aux Block(all)          | 86.7  | 67.9 | 84.9 | -               |
|                  | Aux Block(private)      | 85.2  | 71.6 | 86.0 | -               |

### TABLE III
OVERVIEW OF TRAINING PARAMETERS: THE MNIST CONVERGES AFTER ABOUT 20 EPOCHS; THE CIFAR10 AND MINI-IMAGENET USES PARAMETER DECAY AND MULTI-STEP SCHEDULER TO ENSURE MODEL CONVERGENCE.

| Parameter        | MNIST       | CIFAR10     | Mini-Imagenet |
|------------------|-------------|-------------|---------------|
| Learning Rate    | 0.01        | 0.01        | 0.01          |
| Momentum         | 0.5         | 0.9         | 0.9           |
| Decay Delay      | 0           | 0.0005      | 0.0005        |
| Batch Size       | 128         | 128         | 32            |
| Epochs           | 20          | 100         | 100           |
| Multi-step Scheduler | 0.1 (50 epochs) | 0.1 (50 epochs) | |

### TABLE IV
AUX BLOCK

| Layer                        | Parameters |
|------------------------------|------------|
| Convolution + BatchNorm + ReLU | 7 × 7 × 96, strides=2 |
| MaxPool                      | 2 × 2      |
| Convolution + BatchNorm + ReLU | 3 × 3 × 192 |
| MaxPool                      | 2 × 2      |
| Convolution + BatchNorm + ReLU | 3 × 3 × 384 |
| MaxPool                      | 2 × 2      |
| Convolution + BatchNorm + ReLU | 3 × 3 × 512 |
| MaxPool                      | 2 × 2      |
| Fully Connected              | 10         |

We use a two-iteration PGD to craft adversarial examples from the public model and observe each Aux Block’s accuracy to investigate whether the position of an Aux Block will influence its robustness. The result in Table [VI] proposes that Block2 and Block3 have a same performance compared to the original result, while Block1 improves robustness to some extent. Therefore, we propose that deep layers are not a wise election for an Aux Block.
TABLE V

| Layer      | Parameters |
|------------|------------|
| MaxPool    | 2 x 2      |
| Convolution + BatchNorm + ReLU | 3 x 3 x 384 |
| MaxPool    | 2 x 2      |
| Convolution + BatchNorm + ReLU | 3 x 3 x 512 |
| MaxPool    | 2 x 2      |
| Fully Connected | 10         |
| MaxPool    | 2 x 2      |
| Fully Connected | 10         |

The structure of Block2, Block3.

TABLE VI

| Block   | private | public |
|---------|---------|--------|
| Block1  | 60.96   | 53.58  |
| Block2  | 53.92   | 53.99  |
| Block3  | 53.92   | 53.99  |

The accuracy of three different Block in different positions. Block1 has a highest rate.

The second question: What is the number of Aux Blocks?

If we create a large number of Aux Blocks, it is inescapable to select feature maps in deep layers as inputs, which will not increase the robustness of our model but aggravate the weight of incorrect output for a score-based Aux Block model and make the model become too complicated. In our experiments, we choose the first three layers to add Aux Blocks.

The third question: What structure is suitable for an Aux Block?

A rough Aux Block will decrease the model accuracy of clean data, while the model will look overstuffed if an Aux Block is too meticulous. In this paper, we design three Aux Blocks as Aux1, Aux2, Aux3 for three datasets.

For MNIST, we use the model in Table VIII and an Aux Block model in Table IX. For CIFAR10, the Aux Block structure is in Table X. For Mini-Imagenet, the structure is in Table XI.

C. Static Adversary Results

We show our method improves the test accuracy of adversarial examples in the static adversary. We randomly select 1000 samples for the boundary attack in the testing set and do not use this algorithm in Mini-Imagenet due to the shortage of computation.

In Table II, our model outputs two values: one is the result from the whole system and the other is only from Aux Blocks as private results. Compared to adversarial training, Aux Block(private) has a better or equivalent performance, which illustrates that adversarial examples from the original non-defense model can be resisted by the Aux Block model. Particularly, our method’s performance is better than adversarial training against Carlini and Wagner’s attack on all datasets.

D. Adaptive Adversary Results

Compared with the static adversary, an adaptive adversary is a more challenging circumstance to defense. The attacker bypasses the original no-defense model and crafts specific examples from defense models straightforward.

1) Results: Table VII shows the accuracy on the Adp-PGD attacks. In this table, our model outperforms the adversarial training model remarkably 56.55% and 54.6% on Mnist and Cifar10 dataset, while adversarial training models only have about 20% accuracy.

Fig. 4 also shows Aux Block model helps to improve the accuracy of adversarial examples. For small perturbation, even if the attacker know how to craft adversarial examples for our Aux Block model, it is still able to improve robustness compared to adversarial training models. However, as observed in the experiments, if the perturbation is large enough, our Aux Block model will also be collapsed.

Although the experiment concerns with un-targeted attacks, it does not mean targeted attacks are not covered. As we know, targeted attacks are harder to attack and easier to defense.

Fig. 5. Left: In the MNIST dataset, the ratio of the average value $l_{avg}$ to the loss value $l_i$ from the Adp-FGSM perturbation with random initial. A small ratio value represents the average loss value is less than the targeted loss. Right: A figure in the CIFAR10 dataset.

2) Interpretation: An adversarial example should fool most of Aux Blocks to mislead our method. We assume an attack for one Block is not sufficient to fool other Blocks and crafting an example to fool all Blocks together is more difficult than...
fooling one. Therefore, our model is less likely to be collapsed than non-defense one.

To prove an attack for one Block is not sufficient to fool other Blocks, we investigate the loss value in our Aux Block model. We craft the specific perturbation targeted to one Aux Block, and then observe the loss value of this adversarial example in the selected classifier and also the average value in others. For example, the adversarial example $x_{adv} = x + \sigma_i$, where $\sigma_i$ is the perturbation for Aux Block $i$. Therefore, the first loss value is $l_i = l(f_i(x_{adv}), y)$ and the second value is

$$l_{avg} = \frac{1}{m} \sum_{k=1,k\neq i}^{m+1} l(f_k(x_{adv}), y).$$

Other auxiliary classifiers are not sensitive to the perturbation from the targeted classifier in Fig. 5, since the ratio is less than 1 especially when the perturbation is not very large. One severe attack for an Aux Block has only slight influence to other Aux Blocks. We can see that random value do not effect the ratio. Moreover, it is obvious that crafting an example to fool all Blocks together is more difficult than fooling one. Therefore our method is still robust in the presence of adaptive white-box adversaries.

V. Conclusion and Outlook

TABLE VII

| Metric                | MNIST   | CIFAR10 | Mini-Imagenet |
|-----------------------|---------|---------|---------------|
| Adversarial Training  | 22.10   | 22.05   | 25.5          |
| Aux Block             | 56.55   | 54.6    | 38.7          |

In this paper, we propose a new defense technique called the Aux Block model to improve the robustness of neural networks against adversarial perturbations. We show that our algorithm is a self-ensemble model and the experimental results demonstrate that our model is robust in both two adversaries and reports significantly improved empirical results in real applications. Hence, our work lays out a new framework for adversarial machine learning.

We suggest two lines of work for future study of the model. On one hand, more applications are expected other than the image classification applications that has been investigated in this paper. On the other hand, a theoretical analysis of the method is necessary to better understand the application scopes of the proposed model.

ACKNOWLEDGMENT

Suppressed for blind review.

REFERENCES

[1] 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.
[2] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli. Evasion attacks against machine learning at test time. In Joint European conference on machine learning and knowledge discovery in databases, pages 387–402. Springer, 2013.
[3] W. Brendel, J. Rauber, and M. Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. arXiv preprint arXiv:1712.04248, 2017.
[4] J. Buckman, A. Roy, C. Raffel, and I. Goodfellow. Thermometer encoding: One hot way to resist adversarial examples, 2018.
[5] N. Carlini and D. Wagner. Magnet and “efficient defenses against adversarial attacks” are not robust to adversarial examples. arXiv preprint arXiv:1711.08478, 2017.
[6] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In 2017 IEEE Symposium on Security and Privacy (SP), pages 39–57. IEEE, 2017.
[7] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In International Conference on Learning Representations, 2015.
[8] W. He, J. Wei, X. Chen, N. Carlini, and D. Song. Adversarial example defenses: Ensembles of weak defenses are not strong. arXiv preprint arXiv:1706.04701, 2017.
