SPARTA: SPATIALLY ATTENTIVE AND ADVERSARIALLY ROBUST ACTIVATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Adversarial training has been demonstrated to be useful for improving the robustness of deep neural networks (DNNs). However, the impacts of basic network components (*e.g.*, ReLU, the widely used activation function for DNNs) to adversarial training effectiveness received less attention and has not been comprehensively investigated so far. To fill this gap, in this paper, we argue that the *spatially-shared* and *input-independent* activating properties of the ReLU make the DNNs under both standard training and adversarial training less robust to white-box adversarial attacks. To address such challenges, we design a novel activation function, *i.e.*, **SPARTA**: Spatially Attentive and Adversarially Robust Activation, which enables DNNs to achieve higher robustness (*i.e.*, lower error rate on adversarial examples) and accuracy (*i.e.*, lower error rate on clean examples) than the DNNs based on the state-of-the-art activation functions. We further investigate the relationships between our SPARTA and the state-of-the-art search-based activation function, *i.e.*, Swish, and feature denoising method, providing insights about the advantages of our method. Moreover, comprehensive evaluation has demonstrated two important properties of our method: *First, superior transferability across DNNs*. Our adversarially trained SPARTA function for one DNN (*e.g.*, ResNet-18) can be fixed to train another adversarially robust DNN (*e.g.*, ResNet-34), achieving higher robustness than the one using vanilla ReLU as activation. *Second, superior transferability across datasets*. The SPARTA function trained on one dataset (*e.g.*, CIFAR-10) can be employed to train adversarially robust DNNs on another dataset (*e.g.*, SVHN) and helps achieve higher robustness than DNNs with vanilla ReLU as activation. These properties have highlighted the flexibility and versatility of SPARTA. Accompanying code is also submitted in the supplementary material.

1 INTRODUCTION

Ever since the identification of the adversarial examples (Szegedy et al., 2013) posing severe security threats to deep neural networks (DNNs), studies have been pouring in to improve the adversarial robustness of the DNNs (Papernot et al., 2016; Buckman et al., 2018; Xie et al., 2017; Dhillon et al., 2018; Liu et al., 2018; Wang et al., 2018; Bhagoji et al., 2018; Guo et al., 2017; Prakash et al., 2018; Song et al., 2017; Samangouei et al., 2018; Liao et al., 2018). Among these various methods, adversarial training (Goodfellow et al., 2014) is regarded as one of the most effective attempts to improve the adversarial robustness of the neural network. Adversarial training aims at solving a min-max game by training on adversarial examples (on-the-fly) until the model learns to classify them correctly. Given training data-label pairs $(x, y) \in \mathcal{X}$, loss function $\ell(\cdot)$, DNN F_θ with network weights θ, and a pre-specified ϵ-ball range where x can perturb, the adversarial training approach (Madry et al., 2017) solves: $\theta^* = \arg\min_\theta \mathbb{E}_{(x,y)\in\mathcal{X}}[\max_{\delta\in[-\epsilon,\epsilon]^N} \ell(x+\delta; y; F_\theta)]$. As can be seen, adversarial training is composed of two iterative steps: (1) an inner $\max(\cdot)$ step that finds the adversarial examples and (2) an outer $\min(\cdot)$ step that carries out network parameters updates. Under this paradigm and in this work, we set out to investigate the impacts of basic network components, such as the commonly used ReLU activation, to the adversarial training effectiveness. We argue that the spatially-shared and input-independent activating properties of the ReLU make the DNNs under both standard training and adversarial training less robust to white-box adversarial attacks. Such uniformity across input spatial dimensions and among different input data may be less ideal in suppressing adversarial patterns, rendering the adversarial training less effective, as

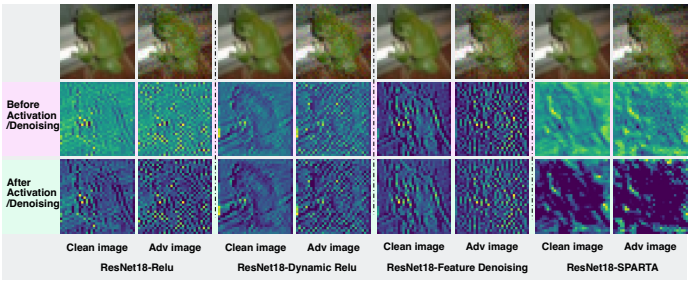

Figure 1: Visualization results of feature maps before and after activation for ReLU, Dynamic ReLU (Chen et al., 2020), and SPARTA or feature denoising for Xie et al. (2019). Note that, all DNNs are based on the ResNet-18 backbone under standard training and the feature maps are from the last layer of the first group.

we will thoroughly explore in the experimental sections. To address such challenges, we design a novel activation function, *i.e.*, SPARTA: spatially attentive and adversarially robust activation by allowing the activation to allocate different amounts of attention across input spatial dimensions, as well as to be dynamically adapted for each individual input. The flexibility in SPARTA, as opposed to the uniformity in ReLU, enables DNNs to achieve higher robustness (*i.e.*, lower error rate on adversarial examples) and accuracy (*i.e.*, lower error rate on clean examples) than the DNNs based on the state-of-the-art (non-spatially attentive and non-dynamic) activation functions.

We further investigate the relationships between our SPARTA and the state-of-the-art search-based activation function, *i.e.*, Swish (Ramachandran et al., 2017), and feature denoising method (Xie et al., 2019), providing insights about the advantages of our method. Moreover, comprehensive evaluation demonstrates two important properties of our method: 1) *superior transferability across DNNs*. The adversarially trained activation function for one DNN (*e.g.*, ResNet-18) can be fixed to train another adversarially robust DNN (*e.g.*, ResNet-34), achieving higher robustness than the one using ReLU; 2) *superior transferability across datasets*. The SPARTA function trained on one dataset (*e.g.*, CIFAR-10) can be employed to train adversarially robust DNNs on another dataset (*e.g.*, SVHN) and helps achieve higher robustness than DNNs with ReLU. These properties demonstrate the advantage of SPARTA in terms of flexibility and versatility.

2 METHODOLOGY

In this section, we introduce our spatially attentive and adversarially robust activations (SPARTA) and investigate its effects to the adversarial robustness under standard training and adversarial training, so that to answer a key question: whether or not the spatial-wise, dynamic, and attentive activation can benefit DNN's adversarial robustness?

2.1 EXISTING ACTIVATION FUNCTIONS AND CHALLENGES

Given an input tensor $\mathbf{X}$, the widely used activation function, *e.g.*, ReLU, can be represented as

$$\mathbf{Y}_p = \max(\mathbf{X}_p, 0), \ \forall p \in \mathcal{P}, \tag{1}$$

where $\mathbf{X}_p$ is the p-th element in $\mathbf{X}$ and $\mathcal{P}$ denotes the set of all element positions of $\mathbf{X}$. The corresponding derivative of this function w.r.t. the input $\mathbf{X}$ is

$$\frac{d\mathbf{Y}_p}{d\mathbf{X}_p} = \begin{cases} 1, & \text{if } \mathbf{X}_p \geq 0, \\ 0, & \text{if } \mathbf{X}_p < 0, \end{cases} \ \forall p \in \mathcal{P}. \tag{2}$$

We argue that *such unified activation across all elements of input tensor during the both forward and backward processes makes the adversarial training less effective.* Intuitively, the white-box adversarial attack can be easily achieved due to: ❶ for the forward process, both clean and corrupted elements in $\mathbf{X}$ are equally activated, making the adversarial noise easily propagate to the deeper layers, thus affecting the prediction results, directly. ❷ during the back-propagation process of the white-box attack, the gradients of all elements in $\mathbf{X}$ pass evenly the activation function for generating the adversarial perturbations, making the white-box attack re-searching optimized solution easily. As shown in Figure 3, during the targeted adversarial attack, the loss is easily minimized when we use the ReLU as the activation function.

To overcome above limitations, we take the following two factors into consideration to design novel activation function: ❶ a spatial-wise and attentive activation should be developed, allowing different elements in $\mathbf{X}$ having different activation scores. For example, the elements corrupted by adversarial noise should be suppressed during the activating while the clean ones should be preserved. ❷ The

activation function should be dynamic, that is, it could be tuned to adapt to different inputs. Actually, the first factor hints that the activation should have the spatial-wise and attentive properties, where the semantic and clean elements should be highlighted while the corrupted ones should be suppressed. The second factor indicates that the activated value of each element should consider the whole input.

There are quite a few works in exploring how to improve the ReLU activation function from the viewpoint of enhancing DNNs' accuracy (Nair & Hinton, 2010; Maas et al., 2013; Goodfellow et al., 2013; He et al., 2015; Clevert et al., 2016; Ramachandran et al., 2017; Hu et al., 2018; Chen et al., 2020; Xie et al., 2020). However, none of them could perfectly fit the above two factors. We summarize their basic information in terms of the spatial-wise, attentive and dynamic properties in Table 1. Among these improved ReLU variants, LeakyReLU, and exponential linear unit (ELU) extend the activation range to negative values while all input elements share the same activation condition, which cannot be tuned according to inputs. PReLU adds extra learnable parameters to the basic ReLU and can be offline trained. However, they are still fixed for different inputs after training. Dynamic ReLU is a spatial-wise and dynamic activation function where each input element has an exclusive activation function represented by several linear functions whose slope and bias parameters are dynamically predicted by a network. Nevertheless, dynamic ReLU is specifically designed for accuracy enhancement, which lacks generality and does not consider the attentive requirement of adversarial robustness, failing to suppress adversarial corrupted elements. As shown in Figure 1, the feature maps of ResNet-18 with dynamic ReLU before and after activation are almost the same and the noise patterns are not removed. We will further discuss the quantitative results in the experimental section. In addition to above activations, MaxOut (Goodfellow et al., 2013) and squeeze-and-excitation networks (SE) (Hu et al., 2018) can be also used to realized activations as introduced in (Chen et al., 2020). MaxOut has learnable parameters for the ReLU and can be offline trained but the parameters cannot change according to different inputs. SE lets the activation rely on the input and realize dynamic activation which however is shared by all elements.

Table 1: Main activation functions for accuracy enhancements and adversarial robustness.

Designing for	Activation	Spatial-wise	Dynamic	Attentive
Accuracy Enhancement	ReLU (Nair & Hinton, 2010)	✗	✗	✗
	LeakyReLU (Maas et al., 2013)	✗	✗	✗
	PReLU (He et al., 2015)	✗	✗	✗
	ELU (Clevert et al., 2016)	✗	✗	✗
	GELU (Hendrycks & Gimpel, 2016)	✗	✗	✗
	Swish (Ramachandran et al., 2017)	✗	✗	✗
	Dynamic ReLU (Chen et al., 2020)	✓	✓	✗
Adversarial Robustness	Smooth ReLU (Xie et al., 2020)	✗	✗	✗
	SPARTA (Ours)	✓	✓	✓

2.2 SPATIALLY ATTENTIVE ACTIVATION FUNCTION

2.2.1 FORMULATION

To address the robustness challenges, we propose the *spatially attentive activation function*. Given an input tensor $\mathbf{X}$, we have

$$\mathbf{Y}_p = \max(\mathbf{X}_p, 0) \cdot \phi_\theta(\mathbf{X})[p], \ \forall p \in \mathcal{P}, \tag{3}$$

where $\phi_\theta(\cdot)$ is a sub-network with θ as the parameters. The sub-network takes all elements of $\mathbf{X}$ as inputs, predicts a new tensor that has the same size with $\mathbf{X}$, and assigns a weight for each element of $\mathbf{X}$. Hence, $\phi_\theta(\mathbf{X})[p]$ denotes the p-th element of $\phi_\theta(\mathbf{X})$ and we have $\{0 \leq \phi_\theta(\mathbf{X})[p] \leq 1 | \forall p, p \in \mathcal{P}\}$. Then, we get the derivative of Eq. 3 w.r.t. the input $\mathbf{X}$

$$\frac{d\mathbf{Y}_p}{d\mathbf{X}_p} = \begin{cases} \phi_\theta(\mathbf{X})[p] + \mathbf{X}_p \frac{\partial \phi_\theta(\mathbf{X})}{\partial \mathbf{X}_p}, & \text{if } \mathbf{X}_p \geq 0, \\ 0, & \text{if } \mathbf{X}_p < 0, \end{cases} \quad \forall p \in \mathcal{P}. \tag{4}$$

Comparing with the formulations in Eq. 1 and 2, we notice that: ❶ for the forward process, each activated element is further processed by a scalar estimated from $\phi_\theta(\mathbf{X})$ that considers the whole input. Intuitively, the offline trained $\phi_\theta(\cdot)$ decides whether the p-th element of $\mathbf{X}$ should be suppressed according to the understanding of the whole input. ❷ In terms of the backward process, in contrast to Eq. 2, the activated elements' gradients are not propagated to the earlier layers directly but determined by $\phi_\theta(\mathbf{X})$ and $\mathbf{X}_p \frac{\partial \phi_\theta(\mathbf{X})}{\partial \mathbf{X}_p}$. When $\phi_\theta(\cdot)$ is a deep neural network with the Sigmoid function as the last layer for activation, the gradient of its input $\frac{\partial \phi_\theta(\mathbf{X})}{\partial \mathbf{X}_p}$ tends to be very small (Glorot & Bengio, 2010). Then, we can see that $\frac{d\mathbf{Y}_p}{d\mathbf{X}_p}$ mainly relies on $\phi_\theta(\mathbf{X})[p]$, which means that the white-box attack based on back-propagation can be affected by the $\phi_\theta(\mathbf{X})[p]$. For example, if $\mathbf{X}_p$ is

the element that should be adversarially corrupted for effective attack and we have $\phi_\theta(\mathbf{X})[p] < 1$, the white-box attack would be harder to be optimized due to the less effective back-propagated gradients. We will validate the two concerns under both standard training and adversarial training in Sec. 2.3. Note that, Eq. 4 does not harm the DNN's accuracy under standard training and can even be helpful to achieve lower error rate. As shown in Table 2, ResNet-18s with our new activation function (*i.e.*, SPARTA-w/o-DPNet and SPARTA that will be introduced in Sec. 2.2.2) achieve lower top-1 error rate than the network using ReLU under both standard training and adversarial training.

2.2.2 ARCHITECTURE OF $\phi_\theta(\cdot)$

A simple architecture for $\phi_\theta(\cdot)$ is a convolutional neural network (CNN) that takes the $\mathbf{X}$ as the input and outputs a tensor having the same size with $\mathbf{X}$ to decide the activation weights of each element. However, such a architecture requires a large amount of parameters, making the whole network difficult to train. Moreover, since an activation function could be deployed at different locations of a CNN (*i.e.*, from the shallow layers to deep ones), the inputs $\mathbf{X}$ would be diverse (*e.g.*, the $\mathbf{X}$ from shallow layers mainly contain spatial details while the deep ones focus on semantic information). Hence, a dynamic architecture that could tune the attentive network according to the input is desired. To this end, we propose the *dynamic spatial-channel-attentive network* as $\phi_\theta(\cdot)$

$$\phi_\theta(\mathbf{X}) = \text{Sigmoid}(\phi_{\theta_s}(\mathbf{X}) \times \phi_{\theta_c}(\mathbf{X})), \tag{5}$$

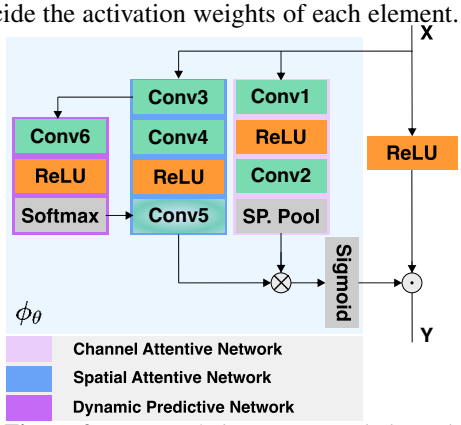

Figure 2: Proposed *dynamic spatial-channel-attentive network* ($\phi_\theta(\cdot)$) containing 3 sub-networks, *i.e.*, channel attentive net ($\phi_{\theta_c}(\cdot)$), spatial attentive net ($\phi_{\theta_s}(\cdot)$), and dynamic predictive net ($\phi_{\theta_d}(\cdot)$), where $\theta = \{\theta_s, \theta_c, \theta_d\}$.

where $\phi_{\theta_s}(\cdot)$ and $\phi_{\theta_c}(\cdot)$ denote the spatial-attentive network and channel-attentive[1] network, respectively. When we have $\mathbf{X} \in \mathbb{R}^{H\times W\times C}$, $\phi_{\theta_s}(\mathbf{X}) \in \mathbb{R}^{H\times W}$ is the spatial attentive map across all channels and $\phi_{\theta_c}(\mathbf{X}) \in \mathbb{R}^{1\times 1\times C}$ denotes the channel attentive vector across all spatial positions. Moreover, we construct a dynamic predictive network to predict partial parameters of θ_s according to the input $\mathbf{X}$. We show the whole architecture in Figure 2, which contains three sub-networks where the Conv1, Conv2, Conv3, and Conv6 are with the size of '$C \times C \times 3 \times 3$', Conv4 is with the size of '$C \times 1 \times 3 \times 3$', and Conv5 is with the size of '$1 \times 1 \times 3 \times 3$'. In particular, the parameters of Conv5 are estimated from the dynamic predictive network. We will further discuss the influence of different architectures in the Sec. 2.3.

2.3 ANALYSIS OF SPATIALLY ATTENTIVE ACTIVATION FUNCTION

In this section, we aim to analyze and validate the proposed method by comparing with the basic activation function, *i.e.*, ReLU in Sec. 2.1, and answer the following questions: ❶ whether does the proposed SPARTA help achieve higher adversarial robustness under *standard training* and *adversarial training*, respectively? ❷ whether does the advantages stem from the spatial-wise, dynamic, and attentive architectures?

2.3.1 SETUP

For comprehensive analysis of the proposed activation function, we use ResNet-18 (He et al., 2016) as the backbone network and modify it by replacing the last ReLU layers of the four groups in ResNet-18 with our SPARTA. We further discuss the influence of replacement strategies in Appendix A.1. Then, we conduct the image classification task on CIFAR-10 dataset, comparing the top-1 error rate of the raw ResNet18 and the modified one under both standard training and adversarial training. For adversarial training, we follow the setups in (Xie et al., 2019) and perform the targeted Projected Gradient Descent (PGD) attack (Madry et al., 2018) to generate adversarial examples with the step size of 1.0, and the maximum perturbation of 16.0. We implement three PGD attacks according to the iteration number of 10, 30, and 50 and denote them as PGD-10, PGD-30, and PGD-50, respectively. Note that, in all sub-sequence experiments, the top-1 error rate on adversarial images (adv. images) means that we first generate adversarial examples by using PGD to attack the evaluated DNN and calculate the classification error rate on these adversarial images. For the DNN updating, we set the learning rate to be 0.1 with the $10\times$ attenuation at the 30th and 60th epochs, and the weight decay is set to be $1e{-}4$.

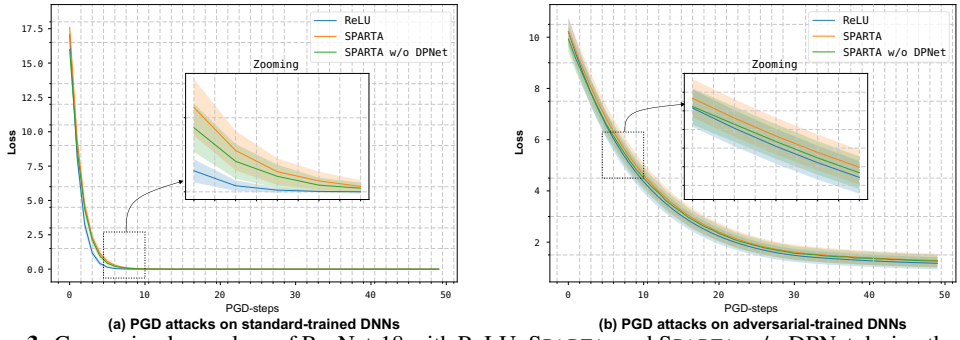

Figure 3: Comparing loss values of ResNet-18 with ReLU, SPARTA, and SPARTA-w/o-DPNet during the PGD attack under the standard training (a) and adversarial training (b), respectively.

2.3.2 DYNAMIC AND ATTENTIVE ACTIVATION BENEFITS ADVERSARIAL ROBUSTNESS

Table 2 shows the top-1 error rates of three versions of ResNet-18 on the adversarial and clean images of CIFAR-10 under both adversarial training and standard training, from which we have the following observations and conclusions: ❶ Compared with ReLU, SPARTA does help the DNN achieve much better adversarial robustness (*i.e.*, lower top-1 error rate on adversarial examples from white-box PGD attacks) under adversarial training while further improving the accuracy on clean images, concluding that SPARTA *improves adversarial robustness without the sacrifice of classification accuracy for clean images*. ❷ In terms of the standard training, the SPARTA leads to lower top-1 error rate (*i.e.*, 100% for ReLU vs. 99.85% for SPARTA) under the PGD-10 attack while achieving much higher accuracy (*i.e.*, lower error rate on clean images), demonstrating that SPARTA *does not rely on adversarial training and still benefits to both adversarial robustness and accuracy under standard training.* ❸ Compared with SPARTA-w/o-DPNet where the DPNet in $\phi_\theta(\cdot)$ is removed, SPARTA achieves lower top-1 errors under all PGD attacks in the cases of adversarial and standard training, confirming that *the dynamic activation function via the proposed DPNet helps the DNN achieve higher adversarial robustness.* ❹ When further comparing ReLU with SPARTA-w/o-DPNet, we see that ResNet-18 with SPARTA-w/o-DPNet has lower top-1 error rates on most of the PGD attacks and clean images. Under the standard training, SPARTA-w/o-DPNet always improves the DNN with lower error rates on clean images. These results demonstrate *attentive activation introduced by the SANet and CANet does enhance the DNNs' adversarial robustness and accuracy and also benefit the standard training for higher accuracy*.

Table 2: Comparing ResNet-18s equipped with ReLU, SPARTA-w/o-DPNet, and SPARTA, respectively.

	ResNet-18 with	Top-1 error on Adv. Images PGD-10	PGD-30	PGD-50	Top-1 error on Clean Images
Adv. Train. on PGD-10	ReLU	31.54%	68.93%	75.64%	15.66%
	SPARTA-w/o-DPNet	29.43%	66.13%	72.35%	15.44%
	SPARTA	29.31%	65.81%	72.55%	15.48%
Std. Train.	ReLU	100.0%	100.0%	100.0%	7.71%
	SPARTA-w/o-DPNet	99.94%	100.0%	100.0%	6.98%
	SPARTA	99.85%	100.0%	100.0%	6.90%

To better understand the above results, we further conduct an experiment to compare the loss values of pre-trained DNNs during PGD attacks, to validate whether the proposed activation function makes the adversarial attack harder as explained in Sec. 2.3.2. Specifically, we perform PGD-50 attack on 20% examples of CIFAR-10 testing dataset and collect the loss values during the optimization process. Then, we calculate the mean and standard variation of loss values at each iteration step across all examples, and draw three plots of the ResNet-18s with ReLU, SPARTA-w/o-DPNet, and SPARTA, respectively. As shown in Figure 3, we see that the loss values of DNNs based on SPARTA and SPARTA-w/o-DPNet are always larger than that of ReLU-based DNN along the iteration steps. It demonstrates that *the proposed attentive and dynamic activation function does make the optimization of adversarial attack harder for both adversarial and standard trained DNNs.*

2.3.3 SPATIAL-WISE ACTIVATION BENEFITS ADVERSARIAL ROBUSTNESS

In addition to the attentive and dynamic properties, we further analyze the importance of spatial-wise activation. To this end, we set spatial-neighboring elements of $\mathbf{X}$ sharing the same attentive scores and control the neighboring size to study the influence of spatial-wise activation. We reformulate

[1] For brevity, *spatially attentive* and *channel-wise attentive* networks are sometimes paraphrased as *spatial-attentive* and *channel-attentive* networks.

Eq. 3 to represent the above process as

$$\mathbf{Y}_p = \max(\mathbf{X}_p, 0) \cdot \phi_\theta(\mathbf{X})[\lfloor \frac{p}{N} \rfloor], \ \forall p \in \mathcal{P}, \tag{6}$$

where we set the output size of $\phi_\theta(\mathbf{X})$ to be $\frac{H}{N} \times \frac{W}{N} \times C$ and N controls the neighboring size. For example, when we set $N = 2$, every four elements in $\mathbf{X}$ share the same attentive scores; when we have $N = 1$, Eq. 6 becomes Eq. 3, that is, each element has its exclusive attentive score. To analyze the effects of different N, we modify ResNet-18 by replacing the fourth block's ReLU with SPARTA and get three DNNs by setting $N = 1, 2, 4$. Then, we perform the adversarial training and evaluate the robustness and accuracy, respectively. We present the results in Table 3 and have following observations: ❶ for all three PGD attacks, the top-1 error rate on adversarial images increases as the N becomes larger (*i.e.*, more elements share the same attentive score), indicating that *spatial-wise activation benefits the adversarial robustness.* ❷ In terms of the results on clean images, the DNN with $N = 1$ has higher top-1 error rate than the ones with $N = 2$ and $N = 4$, which indicates the spatial-wise activation could reduce the accuracy to some extent. Even then, SPARTA with $N = 1$ still enables the DNN to achieve lower error rate than ReLU.

Table 3: Comparing ResNet-18s equipped with ReLU, SPARTA-w/o-DPNet, and SPARTA, respectively.

ResNet-18 with SPARTA	Top-1 error on Adv. Images PGD-10	PGD-30	PGD-50	Top-1 error on Clean Images
$N = 1$	29.73%	65.93%	72.98%	15.25%
$N = 2$	30.11%	66.27%	73.40%	14.98%
$N = 4$	30.17%	66.58%	73.45%	15.07%

3 RELATIONSHIP TO EXISTING ADVERSARIAL TRAINING (AT) METHODS

3.1 RELATIONSHIP TO SMOOTH AND SEARCH-BASED ACTIVATION FOR AT AND BEYOND

More recently, Xie et al. (2020) identify the importance of the smooth activation function for adversarial training and show the search-based activation function, *i.e.*, Swish (Ramachandran et al., 2017), which achieves the state-of-the-art adversarial robustness and can be represented as

$$\mathbf{Y}_p = \mathbf{X}_p \cdot \text{Sigmoid}(\mathbf{X}_p), \ \forall p \in \mathcal{P}. \tag{7}$$

Meanwhile, we can reformulate the SPARTA by combining Eq. 3 and 5 and get

$$\mathbf{Y}_p = \max(\mathbf{X}_p, 0) \cdot \text{Sigmoid}((\phi_{\theta_s}(\mathbf{X}) \times \phi_{\theta_c}(\mathbf{X}))[p]), \ \forall p \in \mathcal{P}, \tag{8}$$

Comparing Eq. 7 with Eq. 8, we can see that Swish is very similar to our SPARTA, but having two main differences: ❶ the first term $\mathbf{X}_p$ in the right part of Eq. 7 is further processed via ReLU in Eq. 8. ❷ In terms of the variable in Sigmoid$(\cdot)$, Eq. 7 uses the $\mathbf{X}_s$ itself while Eq. 8 adopts the spatial and channel attentions that consider all elements in $\mathbf{X}$. We have demonstrated the advantages of the spatial and channel attentions in Sec. 2.3.2. Here, we further study the influence of the first difference and show that the *adversarial robustness of Swish can be further enhanced by simply adding the ReLU to Eq. 7.* As shown in Table 4, when equipping Swish with ReLU (*i.e.*, Swish-ReLU), the top-1 error rates on all PGD attacks decrease.

Table 4: Comparing ResNet-18s with Swish, Swish-ReLU, and SPARTA under 3 PGD attacks.

ResNet-18 with	Top-1 error on Adv. Images PGD-10	PGD-30	PGD-50
Swish	30.38%	68.65 %	76.21%
Swish-ReLU	30.10%	67.70%	75.07%
SPARTA	29.31%	65.81%	72.55%

3.2 RELATIONSHIP TO FEATURE-DENOISING-BASED ADVERSARIAL TRAINING

Xie et al. (2019) propose to improve the adversarial robustness of DNNs by adding extra blocks for feature denoising, inspired by the fact that the pixel-level adversarial noise poses large perturbations to the deep features and leads to noisy activation overwhelming the true ones, resulting in erroneous predictions. Actually, SPARTA can also be regarded as a denoising block and has the capability of feature denoising, since the perturbed elements in $\mathbf{X}$ are selectively activated or suppressed according to the predictive results of $\phi_\theta(\mathbf{X})$. As shown in Figure 1, in terms of ResNet-18-SPARTA, we see obvious noise patterns before the activation, which are suppressed after the activation. As a result, the feature map after activation becomes similar to the clean one. Compared with the method of (Xie et al., 2019), SPARTA has the following difference as well as advantages: ❶ from the viewpoint of denoising, SPARTA uses multiplication for denoising and dynamically tunes the parameters via DPNet according to the inputs while the feature denoising method adopts the addition with fixed denoising operations. The higher level of flexibility of SPARTA helps DNNs achieve much better adversarial robustness. Please find the quantitative analysis in Sec. 4.2. ❷ SPARTA is a new kind of activation function that can directly replace existing ReLUs in a DNN without changing its original architecture. On the other hand, the feature denoising method needs to add new blocks to existing DNNs, requiring extra adaption costs, which are often non-trivial and expensive.

Table 6: Comparing SPARTA with ReLU, ELU, GELU, feature denoising operation (FD), Dynamic ReLU (DyReLU), and Smooth ReLU (SmReLU) by equipping them to ResNet-18 and ResNet-34 for adversarial training and standard training on CIFAR-10 dataset.

	ResNet-18					ResNet-34				
	Adv. Training Error on Adv. Imgs			Error on Clean Imgs	Std. Training Error on Clean Imgs	Adv. Training Error on Adv. Imgs			Error on Clean Imgs	Std. Training Error on Clean Imgs
	PGD-10	PGD-30	PGD-50			PGD-10	PGD-30	PGD-50		
ReLU	31.54%	68.93%	75.64%	15.66%	7.71%	31.00%	67.88%	75.15%	17.18 %	7.36%
ELU	31.44%	67.57%	72.98%	17.08%	7.99%	31.31%	67.67%	74.02%	17.16%	7.88%
GELU	30.27%	68.95%	75.30%	14.55%	6.85%	29.66%	65.95%	73.68%	15.03%	6.52%
FD	31.81%	67.03%	73.52%	16.81%	7.99%	30.71 %	65.72 %	73.90 %	17.49%	8.05%
DyReLU	32.57%	68.54%	75.08%	16.97%	7.40%	31.56%	68.40%	75.84%	17.63%	8.78%
SmReLU	30.38%	68.65%	76.21%	14.26%	7.00%	32.20%	67.51%	74.42%	16.74%	7.28%
SPARTA	29.31%	65.81%	72.55%	15.48%	6.90%	29.17%	64.13%	72.91%	14.47%	6.78%

4 EXPERIMENTS

4.1 SETUP

Following the setup in Sec. 2.3.1, we further consider ResNet-18 and ResNet-34 (He et al., 2016) as the backbones and evaluate on CIFAR-10 Krizhevsky et al. (2009) and SVHN datasets (Netzer et al., 2011), mainly investigating the following questions: How is the performance of SPARTA compared with state-of-the-art activation functions, including ReLU (Nair & Hinton, 2010), ELU (Clevert et al., 2016), GELU (Hendrycks & Gimpel, 2016), Swish (Ramachandran et al., 2017; Xie et al., 2020), Dynamic ReLU (DyReLU) (Chen et al., 2020), and the feature denoising method (FD) (Xie et al., 2019)? Can SPARTA be shared across DNNs? Can SPARTA be shared across Datasets? Moreover, in the appendix, we discuss how to perform replacement with SPARTA in a DNN.

4.2 COMPARISON WITH STATE-OF-THE-ART ACTIVATION FUNCTIONS

We compare our SPARTA with five state-of-the-art activations and the feature denoising method through ResNet-18 and ResNet-34 architectures. Note that, we implement all baseline activations according to their public released codes. As the results on CIFAR-10 shown in Table 6, we have the following observations: ❶ Under adversarial training, DNNs with SPARTA achieve the lowest top-1 error on all three levels of PGD attack as well as lower top-1 error than DNNs with ReLU, demonstrating that *the proposed activation does help DNNs realize better adversarial robustness without sacrificing the accuracy.* ❷ Under standard training, DNNs with SPARTA have the second best accuracy (*i.e.*, second lowest top-1 error on clean images), hinting that *the proposed activation architecture not only benefits to adversarial training for better robustness but also helps achieve higher accuracy.* Then, we further conduct the comparison with more recent baselines, *i.e.*, FD, DyReLU, and SmReLU, on SVHN dataset and present the results in Table 5. Similar with the results on CIFAR-10, our SPARTA achieves the lowest top-1 error rates on the three PGD attacks.

Table 5: Comparing SPARTA with four baseline activations on SVHN datasets.

ResNet-18 with	Top-1 error on Adv. Images		
	PGD-10	PGD-20	PGD-30
ReLU	18.27%	63.71%	71.42%
FD	16.45%	61.85%	69.45%
DyReLU	17.11%	63.50%	70.71%
SmReLU	16.41%	61.00%	68.97%
SPARTA	15.30%	59.29%	67.29%

4.3 CAN SPARTA BE SHARED ACROSS DNNS?

In this part, we study the transferability of SPARTA across DNNs, that is, we regard the pre-trained SPARTA borrowed from one DNN as the activation function for another DNN and see whether it helps achieve better adversarial robustness. To this end, we take ResNet-18 and ResNet-34 as the backbones and conduct the following steps based on CIFAR-10 dataset: *First*, we adversarially train a DNN (*e.g.*, ResNet-34) equipped with SPARTA and get the pre-trained SPARTA at different blocks. We denote the pre-trained activations as $\text{SPARTA}_{\text{ResNet-34}}$. *Second*, we equip another DNN (*e.g.*, ResNet-18) with $\text{SPARTA}_{\text{ResNet-34}}$ and perform the adversarial training without updating the parameters of $\text{SPARTA}_{\text{ResNet-34}}$. Similarly, we can also train ResNet-34 by using the pre-trained SPARTA from ResNet-18 (*i.e.*, $\text{SPARTA}_{\text{ResNet-18}}$). As shown in Table 7, we see that: ❶ DNNs (*i.e.*, ResNet-18 and ResNet-34) with the transferred SPARTA achieve lower top-1 error rate than DNNs using ReLU under all three attacks (*i.e.*, PGD-10, 30, and 50) and the clean images. Such results demonstrate that

Table 7: Comparing the transferred SPARTA with the standard SPARTA and ReLU.

Backbone	Activations	Top-1 error on Adv. Images			Top-1 error on Clean Images
		PGD-10	PGD-30	PGD-50	
ResNet-18	ReLU	31.54%	68.93%	75.64%	15.66%
	$\text{SPARTA}_{\text{ResNet-34}}$	31.06%	68.11%	75.03%	15.17%
	$\text{SPARTA}_{\text{ResNet-18}}$	29.31%	65.81%	72.55%	15.48%
ResNet-34	ReLU	31.00%	67.88%	75.15%	17.18%
	$\text{SPARTA}_{\text{ResNet-18}}$	28.96%	65.51%	72.40%	14.60%
	$\text{SPARTA}_{\text{ResNet-34}}$	27.73%	42.23%	46.40%	23.84%

pre-trained SPARTA *has the transferability to some extent and can help other DNNs achieve better adversarial robustness and accuracy than the ones using ReLU.* ❷ Compared with the DNNs with standard SPARTA (*e.g.*, ResNet-34 with $\text{SPARTA}_{\text{ResNet-34}}$) whose parameters are jointly updated during adversarial training, the DNNs with transferred SPARTA (*e.g.*, ResNet-34 with $\text{SPARTA}_{\text{ResNet-18}}$) achieves worse adversarial robustness (*i.e.*, higher error rate under the three attacks) but much better accuracy (*i.e.*, lower error rate on clean images). For example, ResNet-34 with $\text{SPARTA}_{\text{ResNet-34}}$ obtains much lower top-1 error rates than ResNet-34 with $\text{SPARTA}_{\text{ResNet-18}}$ under all three PGD attacks but gets much higher error on clean images, *i.e.*, 23.84% vs. 14.60%.

4.4 CAN SPARTA BE SHARED ACROSS DATASETS?

We further study the transferability of SPARTA across datasets. Specifically, we first adversarially train a ResNet-18 with SPARTA on CIFAR-10 and get the pre-trained SPARTA denoted as $\text{SPARTA}_{\text{CIFAR}}$. Then, we regard $\text{SPARTA}_{\text{CIFAR}}$ as the activation function for another random initialized ResNet-18 and adversarially train it on SVHN to see whether the ResNet-18 with $\text{SPARTA}_{\text{CIFAR}}$ achieves better adversarial robustness or higher accuracy than the one with ReLU and standard SPARTA that are jointly trained with ResNet-18. As shown in Table 8, we have the following observations: ❶ On both CIFAR-10 and SVHN, ResNet-18 with the transferred SPARTA achieves lower top-1 error rate than the one using ReLU under all three attacks. Such results demonstrate that *pre-trained* SPARTA *on one dataset still works on another dataset, helping DNN achieve better adversarial robustness than the one using ReLU.* ❷ Compared with the standard case where the parameters of ResNet-18 and SPARTA are jointly updated during adversarial training, ResNet-18 with transferred SPARTA has higher top-1 error rates on adversarial images but lower error rate on clean images.

Table 8: Transferability of SPARTA across Datasets.

Datasets	ResNet-18 with	Top-1 error on Adv. Images PGD-10	PGD-30	PGD-50	Top-1 error on Clean Images
CIFAR-10	ReLU	31.54%	68.93%	75.64%	15.66%
	$\text{Sparta}_{\text{SVHN}}$	30.68%	66.93%	74.47%	15.36%
	$\text{Sparta}_{\text{CIFAR}}$	29.31%	65.81%	72.55%	15.48%
SVHN	ReLU	18.27%	63.71%	71.42%	6.45%
	$\text{Sparta}_{\text{CIFAR}}$	16.59%	61.88%	69.45%	6.52%
	$\text{Sparta}_{\text{SVHN}}$	15.30%	59.29%	67.29%	7.05%

5 RELATED WORK

Apart from adversarial training, numerous studies have shown to be effective, to some extent, towards enhancing adversarial robustness of DNNs. These methods can be roughly fall into the following four categories. (1) The ones that involve non-differentiable operators, intentionally or unintentionally. The introduced non-differentiability and numeric instability lead to incorrect and degenerate gradients such as applying the thermometer encoding (Buckman et al., 2018), performing various image transformations (cropping, bit-depth reduction, *etc.*) (Guo et al., 2017), and using local intrinsic dimensionality to characterize adversarial subspaces (Ma et al., 2018). However, they may be circumvented by computing the backward pass using a differentiable approximation of the function (Athalye et al., 2018). (2) The ones involve either a randomized network such as (Dhillon et al., 2018; Liu et al., 2018; Wang et al., 2018) or randomly transformed inputs such as (Xie et al., 2017; Bhagoji et al., 2018; Guo et al., 2017), which hinder correct estimate of the true gradient when using a single sample of the randomness. However, they may be countered by computing the gradient correctly over the expected transformation to the input (Athalye et al., 2018). (3) The ones involve adversarial input data purification such as high-level representation guided denoiser Liao et al. (2018), pixel deflection (Prakash et al., 2018), PixelDefend (Song et al., 2017), and Defense-GAN (Samangouei et al., 2018). However, re-parameterization can greatly diminish these attempts for improving the adversarial robustness of the DNNs (Athalye et al., 2018). (4) Others such as defensive distillation (Papernot et al., 2016) and adversarially robust architecture (Dong et al., 2020).

6 CONCLUSIONS

In this paper, we have designed a novel activation function, named SPARTA, which is designed to be spatially attentive and adversarially robust. It enables DNNs to achieve higher robustness and accuracy than the DNNs based on the state-of-the-art activation functions. We have further investigated the relationships between our SPARTA and the state-of-the-art search-based activation function, *i.e.*, Swish, and feature denoising method, providing insights and discussion about the advantages of our method. Furthermore, comprehensive evaluation has performed, the results of which demonstrated two important properties of our method: *superior transferability across DNNs* and *superior transferability across datasets*. Backed by extensive experiments, these properties have highlighted the flexibility and versatility of SPARTA.

REFERENCES

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

Arjun Nitin Bhagoji, Daniel Cullina, Chawin Sitawarin, and Prateek Mittal. Enhancing robustness of machine learning systems via data transformations. In *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–5. IEEE, 2018.

Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations*, 2018.

Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic relu. In *ECCV*, 2020.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *ICLR*, 2016.

Guneet S Dhillon, Kamyar Azizzadenesheli, Zachary C Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Anima Anandkumar. Stochastic activation pruning for robust adversarial defense. *arXiv preprint arXiv:1803.01442*, 2018.

Minjing Dong, Yanxi Li, Yunhe Wang, and Chang Xu. Adversarially robust neural architectures. *arXiv preprint arXiv:2009.00902*, 2020.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, volume 9, pp. 249–256, 2010.

Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C. Courville, and Yoshua Bengio. Maxout networks. In *ICML (3)*, volume 28, pp. 1319–1327, 2013.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pp. 1026–1034, 2015.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, 2018.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *University of Toronto, Technical Report*, 2009.

Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1778–1787, 2018.

Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 369–385, 2018.

Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E Houle, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613*, 2018.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, pp. 3, 2013.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pp. 807–814, 2010.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597. IEEE, 2016.

Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. Deflecting adversarial attacks with pixel deflection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8571–8580, 2018.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018.

Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Siyue Wang, Xiao Wang, Pu Zhao, Wujie Wen, David Kaeli, Peter Chin, and Xue Lin. Defensive dropout for hardening deep neural networks under adversarial attacks. In *Proceedings of the International Conference on Computer-Aided Design*, pp. 1–8, 2018.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*, 2017.

Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L. Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Cihang Xie, Mingxing Tan, Boqing Gong, Alan Yuille, and Quoc V. Le. Smooth adversarial training. In *arXiv:2006.14536*, 2020.

A Appendix

A.1 Which ReLU layer in a DNN should we replace with the Sparta?

Since Sparta contains extra parameters for the three sub-networks in Figure 2, it is ideal to perform as fewer ReLU replacements as possible to avoid heavy costs. To this end, we study the influence of replacement positions based on the widely used ResNet architecture and take the representative ResNet-18 as a representative case to study. Specifically, ResNet-18 contains four groups and each group has two blocks. We focus on replacing the last ReLU layer of each block and set the following strategies: *First*, we replace the last ReLU layer of the second block of each group with our Sparta and get four DNNs denoted as ResNet-18-$\text{G}_i.\text{B}_2$ where i denotes the ith group of ResNet-18. This setup helps explore the influence of Sparta at different depths of a DNN.

Second, we perform the replacements on all groups simultaneously and study whether more substitutions lead to better adversarial robustness. In particular, we consider two versions denoted as $G_{\{1,2,3,4\}}.B_{\{1,2\}}$ and $G_{\{1,2,3,4\}}.B_2$, respectively. The first one replaces the last ReLU layers of the two blocks of all groups, while the second one only conducts replacement on the second block of all groups.

Table 9: Comparing ResNet-18s with SPARTA employed at different depths with different amounts.

ResNet-18: replacing ReLU with SPARTA at	Top-1 error on Adv. Images PGD-10	PGD-30	PGD-50	Top-1 error on Clean Images
$G_1.B_2$	32.33%	67.23%	74.38%	17.41%
$G_2.B_2$	31.61%	67.76%	75.50%	15.68%
$G_3.B_2$	31.15%	68.12%	75.01%	15.77%
$G_4.B_2$	29.73%	65.93%	72.98%	15.25%
$G_{\{1,2,3,4\}}.B_{\{1,2\}}$	31.32%	65.83%	72.28%	16.70%
$G_{\{1,2,3,4\}}.B_2$	29.31%	65.81%	72.55%	15.48%

According to the results in Table 9, we have the following observations: ❶ In general, using SPARTA at deeper groups helps to achieve better adversarial robustness (*i.e.*, lower top-1 error on adversarial examples) as well as better accuracy (*i.e.*, lower top-1 error rate on clean examples). For example, ResNet-18-$G_4.B_2$ achieves the lowest top-1 error on both adversarial and clean examples. ❷ Replacing more ReLU layers is not helpful to obtain even better adversarial robustness or accuracy. For example, ResNet-18-$G_{\{1,2,3,4\}}.B_{\{1,2\}}$ has eight SPARTA layers but gets higher error rates than $G_{\{1,2,3,4\}}.B_2$ with only four SPARTA layers. Moreover, replacing the last ReLU layers of all groups, *i.e.*, $G_{\{1,2,3,4\}}.B_2$, achieves the best adversarial robustness and the second best accuracy among all variants, indicating the importance of the output activation layers of ResNet groups for adversarial training.