Pattern Recognition Letters 186 (2024) 184–190

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Adaptive feature alignment for adversarial training

Kai Zhao [a,b,*], Tao Wang [a], Ruixin Zhang [a], Wei Shen [c]

[a] Tencent Youtu Lab, China
[b] University of California, Los Angeles, United States of America
[c] Shanghai Jiaotong University, China

ARTICLE INFO

Editor: Jiwen Lu

Keywords:
Adversarial training
Adversarial attack
Image classification

ABSTRACT

Recent studies reveal that Convolutional Neural Networks (CNNs) are typically vulnerable to adversarial attacks. Many adversarial defense methods have been proposed to improve the robustness against adversarial samples. Moreover, these methods can only defend adversarial samples of a specific strength, reducing their flexibility against attacks of varying strengths. Moreover, these methods often enhance adversarial robustness at the expense of accuracy on clean samples. In this paper, we first observed that features of adversarial images change monotonically and smoothly w.r.t the rising of attacking strength. This intriguing observation suggests that features of adversarial images with various attacking strengths can be approximated by interpolating between the features of adversarial images with the strongest and weakest attacking strengths. Due to the monotonicity property, the interpolation weight can be easily learned by a neural network. Based on the observation, we proposed the adaptive feature alignment (AFA) that automatically align features to defense adversarial attacks of various attacking strengths. During training, our method learns the statistical information of adversarial samples with various attacking strengths using a dual batchnorm architecture. In this architecture, each batchnorm process handles samples of a specific attacking strength. During inference, our method automatically adjusts to varying attacking strengths by linearly interpolating the dual-BN features. Unlike previous methods that need to either retrain the model or manually tune hyper-parameters for a new attacking strength, our method can deal with arbitrary attacking strengths with a single model without introducing any hyper-parameter. Additionally, our method improves the model robustness against adversarial samples without incurring much loss of accuracy on clean images. Experiments on CIFAR-10, SVHN and tiny-ImageNet datasets demonstrate that our method outperforms the state-of-the-art under various attacking strengths and even improve accuracy on clean samples. Code will be made open available upon acceptance.

1. Introduction

Recent studies reveal that convolutional neural networks (CNNs) are vulnerable to adversarial attacks on many tasks such as classification [1,2] and biometric identification [3–6]. Adding human imperceptible noise, a.k.a adversarial perturbations, to the original inputs can fool a well-trained network into producing incorrect predictions. This poses a threat to security-sensitive applications, such as biometric identification [7] and self-driving [8].

A line of works have been proposed to enhance model robustness against adversarial samples. Among them the adversarial training (AT) methods are able to achieve strong performance under a variety of attacking configurations [9–11]. Though its performance on improving accuracy on adversarial samples (adversarial accuracy), AT-based methods notoriously sacrifice accuracy on clean samples (standard accuracy) [12,13].

The contradictory between clean and adversarial accuracies encourages researchers to develop techniques that can handle various attacking strengths [10,14]. Zhang et al. [10] introduce a hyper-parameter λ to control the attacking strength during training. A model trained with different λ can be deployed in various environments with different attacking strengths. However, this method needs to retrain the model whenever deploying to a new environment. Wang et al. [14] then propose to embed the parameter λ as an input of the model. During testing, the attacking strength is fed into the model as part of the input. Consequently, mode trained only once can deal with various attacking strengths during testing. However, in real-world applications the attacking strength is unknown to the model (see Fig. 1).

As pointed by previous research [14,17,18], the contradiction between standard and adversarial accuracy may be caused by the

* Corresponding author at: University of California, Los Angeles, United States of America.
E-mail addresses: kz@kaizhao.net, kaizhao@ucla.edu (K. Zhao), tobinwawng@tencent.com (T. Wang), ruixinzhang@tencent.com (R. Zhang), wei.shen@sjtu.edu.cn (W. Shen).

https://doi.org/10.1016/j.patrec.2024.10.004
Received 17 May 2023; Received in revised form 11 June 2024; Accepted 3 October 2024
Available online 9 October 2024

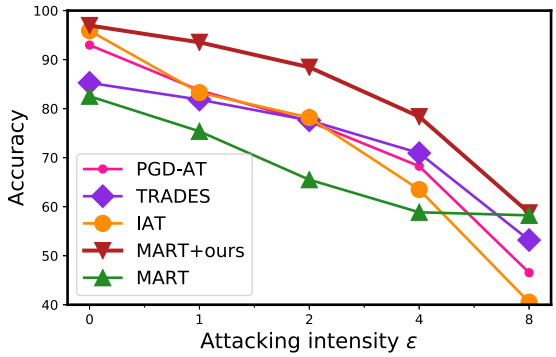

Fig. 1. Performance of various methods under different attacking strength (ϵ). Models are implemented based on the WRN-16-8 [15] architecture and trained on the SVHN [16] dataset. Our method outperforms other competitors under various attacking strength ($\epsilon > 0$) while maintaining the standard accuracy ($\epsilon = 0$).

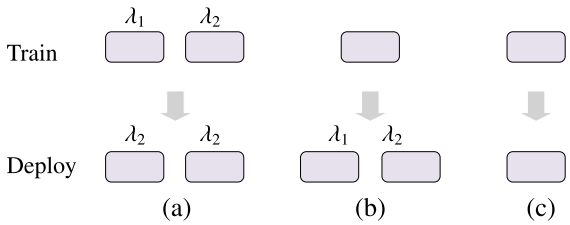

Fig. 2. Schemas of different methods to deal with various attacking strengths. (a) TRADES [10] trains multiple networks with different hyper-parameters, *e.g.*λ. (b) OAT [14] trains only one network and then deploy it with various hyper-parameters. (c) Our proposed method needs only one network in both training and deployment.

misaligned statistics between standard and adversarial features. We conducted an in-depth study on adversarial samples of various attacking strengths and found that the feature statistics undergo a smooth and monotonical transfer with the rising of attacking strength. In other words, features of various attacking strengths can be viewed as a continuous domain transfer [19]. This observation suggests that features of arbitrary attacking strengths can be approximated through linear interpolation between two basic attacking strengths, which we will illustrate in Section 3. For example, features of adversarial samples with intermediate strengths can be approximated by interpolating between the features of the weakest and strongest adversarial samples. Because of the property of monotonicity, the neural network can learn the interpolation weight, which is then automatically adjusted during inference.

Inspired by our observation, we develop the novel adaptive feature alignment (AFA) method that utilizes the dual-BN [14,18] architecture to automatically adjust features for images of arbitrary attacking strengths. In contrast to existing methods that adjust to various attacking strengths through either manual hyper-parameter tuning [14] or model retraining [10], our method is fully automatic and adapts to different attacking strengths using a single model.

The experiments on CIFAR-10, SVHN and tiny-imagenet datasets demonstrate that our proposed method surpasses the prior methods under various attacking strengths. The contributions of this paper are summarized as below:

1. We observe that feature statistics of various attacking strengths undergo *smooth* and *monotonical* transfer.
2. Inspired by the observation, we proposed the adaptive feature alignment (AFA) method that adaptively processes images of various attacking strengths by interpolating the features of two basic attacking strengths.

2. Related work

2.1. Adversarial attacking

Adversarial attacks fool the neural networks by applying imperceptible perturbations to the original inputs and consequently mislead the deep neural network to give an incorrect prediction. Roughly, adversarial attacks can be divided into two groups: (a) minimizing perturbation magnitude given that the image is misclassified, and (b) maximizing the attack success rate given a limited perturbation budget. Next we will introduce several representative methods of these two groups (see Fig. 2).

Szegedy et al. [1] propose the first adversarial attack method that uses box-constrained L-BFGS to find minimal perturbations that can fool the neural networks. Carlini and Wagner (C&W) attack [20] is similar to L-BFGS, but with different loss function applied. Carlini and Wagner investigate different loss functions and conclude that the loss that maximizes the gap between the target class logit and highest non-target class logit results in superior performance. DeepFool [21] iteratively update the perturbations by moving the inputs towards the decision boundary. Ying et al. [22] propose to perturbate in the YUV color space for reversible data hiding. With the goal of finding minimal effective perturbation, group (a) has the disadvantage of being cumbersome and slow.

In contrast, group (b) that maximizes the attack success rate given a limited perturbation budget is more straightforward and widely used in many recent studies [2,23]. The Fast Gradient Sign Method (FGSM) method [2] imposes the gradient to the inputs to increase prediction error. This method is simple but less effective [24]. Iterative FGSM (I-FGSM) [23] iteratively performs the FGSM attack. In each iteration, only a part of the allowed perturbations limit is added, which contributes to its higher attack effect compared to FGSM. Another attacking method, PGD [9], is almost the same as I-FGSM and the only difference is that the PGD attack initializes the perturbation with random noise while I-FGSM just initializes the perturbation with zero values. Recently, Gragnaniello et al. [25] the perceptual preserving back-box attacking method which preserves the perceptual quality of perturbated images. Deng et al. [26] propose to perturbate only small 'meaningful areas' of the image.

2.2. Adversarial training

Adversarial training improves robustness by training models with adversarial samples. Goodfellow et al. [2] use adversarial examples generated by FGSM as training data. Kurakin et al. [23] propose to use a multiple-step FGSM to further improve the performance. Zhang et al. [10] propose TRADES that balances the adversarial accuracy with standard accuracy. Several improvement of PGD adversarial training have also been proposed, such as [27,28] and [29]. Despite many researchers pay much attention to adversarial training and achieve substantial achievement, existing methods either sacrifice accuracy on clean inputs or Xie et al. [18] found that adversarial training can improve the performance in image classification. Adversarial training is also proved to be useful for contrast learning [30]. The most related works to ours are [18] and [14]. The former regard adversarial examples and normal examples as different domain, and utilizing adversarial examples to improve the performance on normal datasets, regardless the robustness of adversarial attacks. The later proposes a model-conditional training framework, which can joint trade-off between accuracy and robustness by a hyper-parameter without re-training. However, such a method has to know whether the test data is adversarial example in advance, which is unpractical in real-world applications.

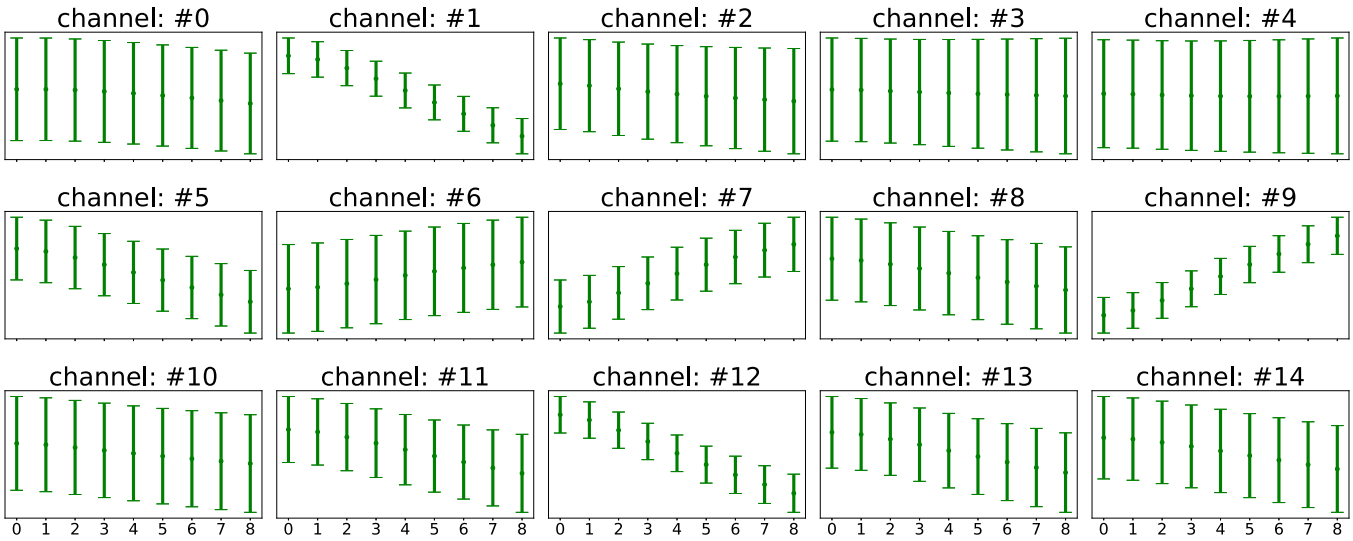

Fig. 3. Per-channel mean and variance of intermediate convolution features under different attacking strengths. Features under different attacking strengths undergo a *continuous* domain transfer.

3. Observation and motivation

In this section, we first describe the observation and illustrate how we are motivated to develop the Adaptive Feature Alignment method. Then we introduce the details of our proposed method.

3.1. Observation and motivation

Many recent studies reveal that features of standard/adversarial samples belong to two separate domains [14,18], and then developed the dual-BN architecture to process standard/adversarial samples separately. While their methods [14,18] neglect the inherent variances across adversarial samples of different attacking strengths. Our observation experiments demonstrate that feature statistics of variance attacking strengths undergo a *continuous* and *monotonical* domain transfer.

We train the Wide-Resnet-28 [15] model on the CIFAR-10 [31] dataset. We adopt the dual-BN architecture proposed in [18] which splits normal and adversarial samples into multiple parallel batchnorm branches. The FGSM [2] attacking method is used in this experiment. After training, we analyze the statistics of features of various attacking strength on the testing set. Concretely, we visualize the per-channel mean and variance of features from an intermediate convolution layer, results are shown in Fig. 3.

The results in Fig. 3 clearly demonstrate that the features are transferring *smoothly* and *monotonically* with the rising of attacking strength. This hints us that features of an arbitrary attacking strength can be approximated by a linear combination few "base" intensities. For instance, a medium attacking strength can be represented by a combination of a slight attacking and a strong attacking.

Based on this observation, we are motivated to put forward the *adaptive feature align* framework that accepts arbitrary attacking intensities through adaptive feature interpolation. We will detail the proposed method in Section 4.2.

4. Methodology

4.1. Overall framework

The overall architecture of our proposed framework is illustrated in Fig. 4. We denote $I_{\epsilon=k}$ as an adversarial image where ϵ is the attacking strength, and I represents an image of arbitrary unknown attacking strength. Commonly these attacking methods apply a gradient ascent to the original inputs to obtain adversarial samples. The attacking strength ϵ is closely related to the gradient magnitude. Specifically, we refer to $I_{\epsilon=0}$ as clean samples without adversarial attacking. Let $x_{\epsilon=k}$ be the convolutional feature extracted from an intermediate layer. Our overall pipeline is extended from the dual-BN architecture that is proposed in [14,18]. We first extend it into a *e.g.* K-BN ($K \geq 2$) architecture (Fig. 4(a)), and then drop some of the BN branches to form the final dual-BN architecture (Fig. 4(b)). The training procedure of our framework includes two stages.

In the first stage (Stage-I), we train a K-BN network with K parallel batch normalization branches and each branch $\mathrm{BN}_k (k = 0, 1, \ldots, K-1)$ deals with samples of specific attacking strength. Note that the attacking strength ϵ is known to the model and each sample x_ϵ will only go through one of the BN branches according to the attacking strength ϵ. Fig. 4(a) demonstrates the training of stage-I.

In the second stage (stage-II), we drop the intermediate BN branches and only keep two branches corresponding to the strongest and lowest attacking strengths, *e.g.* BN_0 and *e.g.* BN_{K-1}. Then we freeze all the model parameters and train our proposed Adaptive Feature Alignment (AFA) module to automatically adjust fusing weight between BN_0 and BN_{K-1}. Fig. 4(b) demonstrates the training of stage-II.

Next, we will detail the proposed 'adaptive feature alignment' module.

4.2. Adaptive feature alignment

Let x be the image feature of an arbitrary sample, and BN_0 and BN_{k-1} be the remaining BN branches as shown in Fig. 4(b).

Given the input x, the proposed AFA first generates two fusing weights W_0, W_1 with a subnetwork termed 'weight generator' (WG). The weight generator network consists of 2 sequential Conv-BN-ReLU blocks and then followed by the global average pooling (AVG) and two linear layers. Finally, the output is normalized by the sigmoid function. Fig. 5 detailly illustrates the architecture of the WG subnetwork. Consequently, the fusing weights are output of the weight generator:

$$\begin{aligned} W_0 &= \mathrm{WG}(x) \\ W_1 &= 1 - W_0. \end{aligned} \tag{1}$$

Given input x and fusing weights W_0, W_1, the out put of AFA $\hat{x}$ is:

$$\hat{x} = W_0 \cdot \mathrm{BN}_0(x) + W_1 \cdot \mathrm{BN}_{k-1}(x), \tag{2}$$

where BN_0 and BN_{k-1} are the batchnorm operator:

$$\mathrm{BN}(x) = \frac{x - \mu}{\sigma}.$$

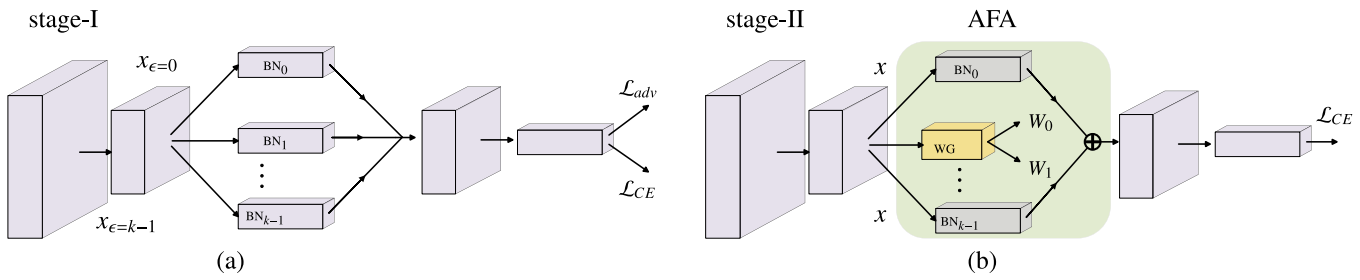

Fig. 4. The overall architecture. (a) In the first stage, we train a multi-branch network where each parallel BN branch corresponds to samples of a specific attacking strength. (b) In the second stage, we drop intermediate BN branches and only keep the outermost branches, *e.g.*BN_0 and BN_{k-1}. And the weight generator (WG) in the adaptive feature alignment (AFA) module will generate the fusing weight of the two remaining branches.

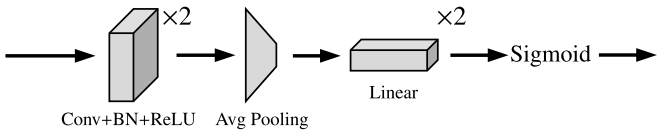

Fig. 5. Architecture of the weight generator (WG). The cube represents the 'Convolution → batchnorm → ReLU' sequential, the trapezoid represents the 'global average pooling' and the bar represents the 'Linear → ReLU' combo. The output is normalized by the sigmoid function.

4.3. Model optimization

As aforementioned, the proposed framework contains two training stages. In the first stage, we train a K-BN network with both classification loss and the adversarial loss. We use the cross-entropy loss (CE-loss) as the classification loss. Note that our method does not constrain the specific form of adversarial loss. In the second stage, we drop all middle BN-branches and only preserve two outermost ones. Then we fix all network parameters and only optimize the **weight generator**, as illustrated in Fig. 4(b). Since the attacking strength is unknown to the model, we use only the classification loss in stage-II.

Stage 1: Training basic model. In the first stage, we train the K-BN architecture as illustrate in Fig. 4(a). Note that each BN branch only accepts samples of a specific attacking strength and all other parameters are shared across all samples. Given a sample x with attacking strength ϵ, the loss function can be formulated as:

$$\mathcal{L}_1(x_\epsilon) = \mathcal{L}_{CE} + 1(\epsilon > 0) \cdot \mathcal{L}_{adv} \tag{3}$$

where $1(\cdot)$ is a indicator function evaluating to 1 when the condition is true and 0 elsewise. The optimization procedure of stage-I is summarized in Algorithm 1, which is extended from AdvProp [18].

Stage 2: Training the weight generator. In the second stage, as illustrated in Fig. 4(b), we drop all other BN branches and only preserve the two outermost branches: BN_0 and BN-$_{k-1}$.

Then we place the weight generator (WG) in a intermediate layer to generate adaptive fusing weights. In stage-II, the attacking strength is unknown to the model and we use only the cross-entropy loss as supervision in stage-II.

Algorithm 1: Training procedure of basic model

Input: The batch of natural samples x_c with label y, the path quantity K, the attack strength vector of each path ξ, the adversarial samples generating algorithm G, loss function for natural training L_c, loss function for adversarial training L_a, the network parameters Θ

iteration number $i \leftarrow 0$, total loss $L_i \leftarrow 0$;
while *not converged* **do**
  $L_i \leftarrow 0$;
  for $k \leftarrow 2$ **to** K **do**
    Switch to kth path to enable kth BN;
    Obtain corresponding attack strength value from the vector $\xi_k \leftarrow \xi[k]$;
    Generate adversarial samples with attacking strength ξ_k by $x_a \leftarrow G(\Theta, x_c, y, \xi_k)$;
    Calculate the adversarial loss of kth path by $l_k \leftarrow L_a(\Theta, x_c, x_a, y)$;
    Accumulate the total loss $L_i \leftarrow L_i + l_k$;
  end
  Calculate the clean loss $l_1 \leftarrow L_c(\Theta, x_c, y)$;
  Accumulate the total loss $L_i \leftarrow L_i + l_1$;
  Compute the gradients of Θ;
  Update the parameters Θ;
end
Output: Θ

5. Experiments

5.1. Experiments setup

Implementation details. We conduct experiments on three datasets: CIFAR-10 [31], SVHN [16] and tiny-ImageNet [32]. The feature alignment is performed at the second last convolution layer of the network. Following several previous works [14,33], we adopt the WRN-28-10 for the experiments on CIFAR-10, and WRN-16-8 on SVHN and tiny-ImageNet. We use $K = 5$ in all our experiments since we found that $K = 5$ achieves a good balance between training efficiency and model performance. The attacking strength of the five branches are: 0/255, 2/255, 4/255, 6/255, 8/255.

As aforementioned, the training procedure is seperated into two stages, On SVHN, the initial learning rates of the two stages are 0.01 and 0.001, respectively. And the initial learning rates on CIFAR and tiny-ImageNet dataset is ten times that of the SVHN dataset. We train 100 epochs for the first stage and 20 epochs for the second stage. During training, we decay the learning rate with factor of 0.9. In the first stage, the learning rate decays at the 75th and the 90th epoch; in the second stage, the learning rate decays at the 10th epoch.

Attacking methods. We adopt 3 adversarial attack algorithms: Fast Gradient Sign Method (FGSM) [2], Projected Gradient Descent (PGD) [9] and C&W [20]. In Table 2, Table 1 and Table 3, we compare our method with other competitors using the PGD attacking method under various attacking strengths. And in Table 4 we use various attacking methods under a fixed attacking strength.

Adversarial training. We compare our proposed method with several recent adversarial training methods, *e.g.*OAT [14], PGD-AT [9], TRADES [10], IAT [34], MART [35].

On the CIFAR-10 and MNIST datasets, we adhered to the original configurations of each method, including the number of perturbation steps and the step size. For the SVHN and mini-ImageNet datasets, we applied the configurations used for the CIFAR-10 dataset.

Adversarial loss functions. As demonstrated in Eq. (3), our method does not constrain the specific form of adversarial loss function. Therefore, our method can be integrated with many adversarial training methods.

Table 1
Accuracy of adversarially trained network on the SVHN dataset. All models are implemented with WRN-16-10 architecture.

Method	Steps	Step size	Acc under different ϵ					
			0	1	2	4	8	Avg
Standard			96.5	73.7	39.1	6.5	0.2	43.2
PGD-AT	7	$\epsilon/4$	93.4	87.3	81.8	72.1	51.3	77.2
PGD-AT+ours			**97.5**	**96.1**	**89.6**	**79.3**	**52.6**	**83.0**
PGD-AT	20	$\epsilon/4$	93.0	83.7	77.4	68.3	**46.5**	73.8
PGD-AT+ours			**98.1**	**93.1**	**87.8**	**75.4**	46.4	**80.0**
TRADES	10	$\epsilon/4$	85.3	81.8	77.6	70.9	53.2	73.8
TRADES+ours			**97.0**	**89.0**	**88.4**	**77.6**	**54.2**	**81.4**
IAT	20	$\epsilon/4$	95.9	83.3	78.2	63.5	40.5	72.3
IAT+ours			**97.4**	**91.6**	**87.9**	**75.9**	**42.3**	**79.0**
MART	10	$\epsilon/4$	82.5	75.4	65.5	58.9	58.2	68.1
MART+ours			**97.0**	**93.5**	**88.5**	**78.4**	**58.7**	**83.2**

Table 2
Accuracy of adversarially trained network on the CIFAR-10 dataset. All models are implemented with WRN-16-10 architecture.

Method	Steps	Acc under different ϵ					
		0	1	2	4	8	Avg
Standard		95.2	30.5	3.0	0.0	0.0	25.7
OAT ($\lambda = 0.2$)	10	90.4	85.9	79.9	65.5	34.6	71.3
OAT ($\lambda = 0.5$)		88.7	84.5	79.2	67.4	40.4	72.1
PGD-AT	20	86.9	81.5	78.5	68.8	46.2	72.4
PGD-AT+ours		**95.8**	**87.0**	**83.9**	**72.6**	**50.0**	**77.9**
TRADES	10	83.2	80.5	77.6	71.3	**53.8**	73.3
TRADES+ours		**95.4**	**86.8**	**82.7**	**72.1**	52.6	**77.9**
IAT	20	92.9	**88.3**	**84.4**	73.5	46.2	**77.0**
IAT+ours		**96.1**	86.2	81.7	**73.7**	**46.2**	76.7
MART	10	83.6	81.0	78.2	72.3	55.3	74.1
MART+ours		**95.9**	**84.3**	**84.2**	**73.5**	**56.1**	**78.8**

5.2. Quantitative comparisons

Quantitative results on CIFAR-10, SVHN and tiny-ImageNet datasets are summarized in Table 1, Table 2 and Table 3, respectively.

Performance on SVHN. The quantitative results on the SVHN dataset is shown in Table 1. Our method consistently outperforms others and the margins are more significant compared to that on the CIFAR dataset, demonstrating the superior of our method. Moreover, our method can even promote the performance of the standard model, as evidenced by that all method surpass the standard model when $\epsilon = 0$.

Performance on CIFAR-10. Table 2 summarizes the comparison results on the CIFAR-10 dataset. Our method consistently outperforms others with a very significant margin, especially under large attacking strengths. Moreover, our method can even promote the performance of a standard, as evidenced by that all method surpass the standard model when $\epsilon = 0$.

Performance on tiny-ImageNet. The comparison results on tiny-ImageNet are reported in Table 3. The results are consistent with the previous conclusion, which shows a stable promotion of our method on a larger scale dataset.

Other attacking methods. In this experiment, we assess the performance of adversarially trained models using different attack methods, rather than the method used to generate the adversarial training samples. We tested following attacking methods: PGD [9] with excessive number of iterations (200), FGSM [2], Auto-PGD (APGD) [36], and C&W [20]. The attacking strength is fixed to $\epsilon = 8$.

The results in Table 4 reveals that our method consistently improve various adversarial training methods when attacked by different attackers, demonstrating the strong generalization of our method.

Adaptive attack. Since our method introduces an additional model prediction, the fusing weight W in Eq. (1). This prediction can bring addition vulnerability to the network. To assess the robustness of our method, we tested the performance of adaptive attack [37].

Table 3
Quantitative results on the tiny-ImageNet dataset. All models are implemented with WRN-16-8 architecture.

Method	Steps	Step size	Acc under different ϵ					
			0	1	2	4	8	Avg
Standard			63.5	6.7	1.0	0.2	0.0	14.3
PGD-AT	20	2	50.4	44.6	38.8	28.0	13.5	35.1
PGD-AT+ours			**64.5**	**46.8**	**35.3**	**28.1**	**13.8**	**37.7**
TRADES	20		42.1	38.2	34.0	26.6	15.1	31.2
TRADES+ours			**62.5**	**47.9**	**39.2**	**26.7**	**15.1**	**38.3**
MART	10		41.0	37.3	33.6	27.1	16.4	31.1
MART+ours			**63.9**	**44.8**	**38.6**	**27.8**	**16.5**	**38.3**

We test our method on the adaptive attack setting by exposing the fusing weight, *e.g.*W_0, to the attacker. Specifically, we expose not only the classification logits, but also the fusing weight W_0 to the attackers. Let $\hat{y}$ be the classification prediction from the model, $\mathcal{L}_{CE}$, $\mathcal{L}_{BCE}$ be the cross entropy loss and binary cross entropy loss. The total loss is:

$$\mathcal{L} = \alpha\mathcal{L}_{CE} + \beta\mathcal{L}_{BCE},$$

where α and β are the loss weights for the two terms. Finally, adaptive attacking methods apply gradient ascent to the inputs to generate adversarial samples.

We tested 7 different loss weights between CE and BCE (10:1, 5:1, 2:1, 1:1, 1:2, 1:5, 1:10) and report the best attacking results (worst adversarial accuracies) in the last column of Table 4.

The results in Table 4 show that the fusing weight is robust against adversarial attack and our method does not bring much additional vulnerability to the model.

Black-box robustness. Black-box attacks generate adversarial samples by attacking a surrogate model. Following setting in [35,38], the ResNet-50 model with Standard adversarial training is adopted as surrogate model for evaluation. The black-box robustness on CIFAR-10 is reported in Table 5.

The results in Table 5 indicate that the robustness gained by our method is not caused by the so-called 'obfuscated gradients' [39]. It can be verified by following evidence: (1) our method has higher accuracy under weak attacks (*e.g.*FGSM) than strong attacks (*e.g.*PGD). (2) our method has higher accuracy under black-box attacks than white-box attacks.

5.3. Ablation study

In this section we do several ablation experiments on the CIFAR-10 dataset to verify& interpret our proposed framework.

Choice of K. Here we ablate the choice of K when training the K-BN architecture in stage-I as illustrated in Fig. 4(a). Results in Table 6 tells that model performance benefits from a larger K and $K = 5$ is a sweet-point for sake of efficiency and performance.

Relationship between fusion weight and ϵ In order to further probe the behavior of AFA, we analyze the relationship between fusion weights (W_1) and the attacking strength (ϵ). Specifically, we train a ResNet-34 model on CIFAR-10 to generate adversarial samples of random attacking strengths. As shown in Fig. 6, the AFA is biased towards the adversarial branch (BN_{K-1}) when the attacking strength rises up, as evidenced by the increase of W_1.

6. Conclusion

We proposed a novel method to defend adversarial samples of various attacking strength. We first observed that the feature statistics of adversarial features change smoothly and monotonically with the

Table 4
Adversarial accuracy of different attacking methods with a fixed attacking strength of $\epsilon = 8$ on the CIFAR-10 dataset. The model is trained with respect adversarial training method and evaluated with various attacking methods. We used the default attacking configurations (step size and number of perturbation steps) of the original papers. For Auto-PGD, we used 200 perturbation steps with a step size of 2. For PGD [9], we used the excessive 200 iterations to verify the performance under strong attacks.

Method	Acc under different attacks					
	Standard	PGD [9] (200 steps)	Auto-PGD [36]	FGSM [2]	C&W [20]	Adaptive [37]
Standard	95.2	0.0	0.0	24.2	0.0	–
PGD-AT	86.9	41.3	44.75	56.1	45.3	–
PGD-AT+ours	**95.8**	**43.3**	**46.72**	**81.1**	**48.3**	48.2
TRADES	83.2	46.9	46.15	64.6	**52.5**	–
TRADES+ours	**95.4**	**47.1**	**48.31**	**80.3**	51.0	52.2
IAT	92.9	41.7	45.1	65.7	43.4	–
IAT+ours	**96.1**	**42.9**	**45.9**	**80.6**	**43.5**	46.1
MART	83.6	49.1	47.1	64.3	53.4	–
MART+ours	**95.9**	**51.0**	**48.7**	**81.3**	**53.6**	**55.6**

Table 5
Black-box performance of a WRN-28-10 network on the CIFAR-10 dataset.

Method	Acc under different ϵ					
	0	1	2	4	8	Avg
PGD-AT	86.9	85.2	83.1	78.2	66.8	80.0
PGD-AT+ours	**95.8**	**94.6**	**90.5**	**91.9**	**67.9**	**88.1**
TRADES	83.2	81.5	79.0	74.1	65.3	76.6
TRADES+ours	**95.4**	**94.7**	**92.7**	**82.4**	**65.9**	**86.2**
IAT	92.9	91.2	88.7	84.5	71.8	85.8
IAT+ours	**96.1**	**95.2**	**93.6**	**87.0**	**74.0**	**89.2**
MART	83.6	82.0	80.0	76.1	66.0	77.6
MART+ours	**95.9**	**95.3**	**93.4**	**83.7**	**67.7**	**87.2**

Table 6
Performance using different number of branches K.

# BNs	Acc under different ϵ					
	0	1	2	4	8	Avg
$K = 2$	95.0	85.1	82.2	70.4	48.1	**76.2**
$K = 3$	95.5	86.7	83.2	72.3	48.6	**77.3**
$K = 5$	95.8	87.0	83.9	72.6	50.0	**77.9**
$K = 9$	95.6	87.2	84.5	72.9	50.4	**78.1**

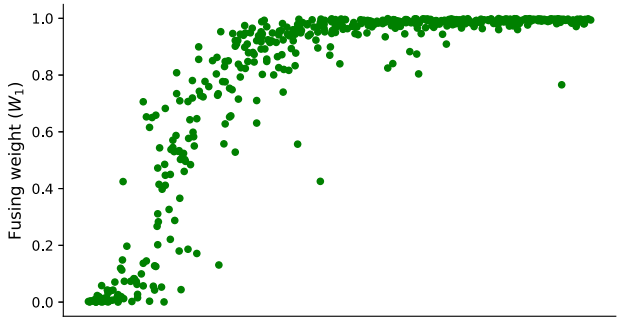

Fig. 6. Relationship between fusion weight W_1 and attacking strength ϵ. With the rising of ϵ, W_1 increases, the features are biased towards the adversarial BN branch, *e.g.*BN_{k-1}.

attacking strength. Based on the observation, we proposed adaptive feature fuse mechanism to automatically generate features for arbitrary unknown attacking strength. Compared to previous works, our method is able to defend samples of different attacking strengths on the fly with a single model, showing its potential in real-world applications. Experiments on several open benchmarks demonstrate the superior of our method on defending adversarial attacks of various strengths. Additionally, our method is easy to embed in other methods to help the original method defending wide range of attacking strengths. Our method needs adversarial samples of various strength during model training and thus introduce computational footprint during training, especially for those slow attacking methods, *e.g.*. query-based attack. Possible future works include improve the training efficiency of the framework and improve the practical application.

CRediT authorship contribution statement

Kai Zhao: Conceptualization, Data curation, Formal analysis, Investigation. **Tao Wang:** Data curation, Formal analysis. **Ruixin Zhang:** Data curation, Funding acquisition, Investigation, Methodology. **Wei Shen:** Project administration, Resources, Supervision, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

All datasets we ued are openly available.

References

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, in: Int. Conf. Learn. Represent., 2013.
[2] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in: Int. Conf. Learn. Represent., 2015.
[3] F.V. Massoli, F. Falchi, G. Amato, Cross-resolution face recognition adversarial attacks, Pattern Recognit. Lett. 140 (2020) 222–229.
[4] C. Bisogni, L. Cascone, J.L. Dugelay, C. Pero, Adversarial attacks through architectures and spectra in face recognition, Pattern Recognit. Lett. 147 (2021) 55–62.
[5] S. Marrone, C. Sansone, On the transferability of adversarial perturbation attacks against fingerprint based authentication systems, Pattern Recognit. Lett. 152 (2021) 253–259.
[6] M. Liu, J. Mu, Z. Yu, K. Ruan, B. Shu, J. Yang, Adversarial learning and decomposition-based domain generalization for face anti-spoofing, Pattern Recognit. Lett. 155 (2022) 171–177.
[7] O.M. Parkhi, A. Vedaldi, A. Zisserman, Deep face recognition, 2015.
[8] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L.D. Jackel, M. Monfort, U. Muller, J. Zhang, et al., End to end learning for self-driving cars, Adv. Neural Inf. Process. Syst. (2016).
[9] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in: Int. Conf. Learn. Represent., 2017.
[10] H. Zhang, Y. Yu, J. Jiao, E.P. Xing, L.E. Ghaoui, M.I. Jordan, Theoretically principled trade-off between robustness and accuracy, 2019.
[11] A. Agarwal, M. Vatsa, R. Singh, N. Ratha, Cognitive data augmentation for adversarial defense via pixel masking, Pattern Recognit. Lett. 146 (2021) 244–251.
[12] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, A. Madry, Robustness may be at odds with accuracy, in: Int. Conf. Learn. Represent., 2019.

[13] C. Xie, Y. Wu, L.v.d. Maaten, A.L. Yuille, K. He, Feature denoising for improving adversarial robustness, in: IEEE Conf. Comput. Vis. Pattern Recog., 2019, pp. 501–509.
[14] H. Wang, T. Chen, S. Gui, T.K. Hu, J. Liu, Z. Wang, Once-for-all adversarial training: In-situ tradeoff between robustness and accuracy for free, in: Adv. Neural Inform. Process. Syst., 2020.
[15] S. Zagoruyko, N. Komodakis, Wide residual networks, in: Brit. Mach. Vis. Conf., 2016.
[16] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A.Y. Ng, Reading digits in natural images with unsupervised feature learning, 2011.
[17] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, A. Madry, Adversarial examples are not bugs, they are features, in: Adv. Neural Inform. Process. Syst., 2019, pp. 125–136.
[18] C. Xie, M. Tan, B. Gong, J. Wang, A.L. Yuille, Q.V. Le, Adversarial examples improve image recognition, in: IEEE Conf. Comput. Vis. Pattern Recog., 2020, pp. 819–828.
[19] H. Wang, H. He, D. Katabi, Continuously indexed domain adaptation, 2020.
[20] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in: 2017 Ieee Symposium on Security and Privacy, Sp, IEEE, 2017, pp. 39–57.
[21] S.M. Moosavi-Dezfooli, A. Fawzi, 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.
[22] C. Ying, Y. Qiaoben, X. Zhou, H. Su, W. Ding, J. Ai, Consistent attack: Universal adversarial perturbation on embodied vision navigation, Pattern Recognit. Lett. 168 (2023) 57–63.
[23] A. Kurakin, I. Goodfellow, S. Bengio, Adversarial machine learning at scale, 2017.
[24] U. Ozbulak, M. Gasparyan, W. De Neve, A. Van Messem, Perturbation analysis of gradient-based adversarial attacks, Pattern Recognit. Lett. 135 (2020) 313–320.
[25] D. Gragnaniello, F. Marra, L. Verdoliva, G. Poggi, Perceptual quality-preserving black-box attack against deep learning image classifiers, Pattern Recognit. Lett. 147 (2021) 142–149.
[26] T. Deng, Z. Zeng, Generate adversarial examples by spatially perturbing on the meaningful area, Pattern Recognit. Lett. 125 (2019) 632–638.
[27] Z. Yan, Y. Guo, C. Zhang, Deep defense: Training dnns with improved adversarial robustness, in: Adv. Neural Inform. Process. Syst., 2018, pp. 419–428.
[28] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, N. Usunier, Parseval networks: Improving robustness to adversarial examples, in: ICML, 2017.
[29] F. Farnia, J.M. Zhang, D. Tse, Generalizable adversarial training via spectral normalization, in: Int. Conf. Learn. Represent., 2019.
[30] C. Xu, D. Li, M. Yang, Adversarial momentum-contrastive pre-training, Pattern Recognit. Lett. 160 (2022) 172–179.
[31] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images, 2009.
[32] J. Deng, W. Dong, R. Socher, L.J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: CVPR, Ieee, 2009, pp. 248–255.
[33] G.W. Ding, Y. Sharma, K.Y.C. Lui, R. Huang, Mma training: Direct input space margin maximization through adversarial training, in: Int. Conf. Learn. Represent., 2019.
[34] A. Lamb, V. Verma, J. Kannala, Y. Bengio, Interpolated adversarial training: Achieving robust neural networks without sacrificing too much accuracy, in: Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, 2019, pp. 95–103.
[35] Y. Wang, D. Zou, J. Yi, J. Bailey, X. Ma, Q. Gu, Improving adversarial robustness requires revisiting misclassified examples, in: Int. Conf. Learn. Represent., 2019.
[36] F. Croce, M. Hein, Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks, in: International Conference on Machine Learning, PMLR, 2020, pp. 2206–2216.
[37] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, A. Kurakin, On evaluating adversarial robustness, 2019, arXiv preprint arXiv:1902.06705.
[38] Y. Wang, X. Ma, J. Bailey, J. Yi, B. Zhou, Q. Gu, On the convergence and robustness of adversarial training, in: ICML, 2019, p. 2.
[39] A. Athalye, N. Carlini, D. Wagner, Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples, in: ICML, 2018.