Adversarial Vertex Mixup: Toward Better Adversarially Robust Generalization

Saehyung Lee Hyungyu Lee Sungroh Yoon*
Electrical and Computer Engineering, ASRI, INMC, and Institute of Engineering Research
Seoul National University, Seoul 08826, South Korea
{halo8218, rucy74, sryoon}@snu.ac.kr

Abstract

Adversarial examples cause neural networks to produce incorrect outputs with high confidence. Although adversarial training is one of the most effective forms of defense against adversarial examples, unfortunately, a large gap exists between test accuracy and training accuracy in adversarial training. In this paper, we identify Adversarial Feature Overfitting (AFO), which may cause poor adversarially robust generalization, and we show that adversarial training can overshoot the optimal point in terms of robust generalization, leading to AFO in our simple Gaussian model. Considering these theoretical results, we present soft labeling as a solution to the AFO problem. Furthermore, we propose Adversarial Vertex mixup (AVmixup), a soft-labeled data augmentation approach for improving adversarially robust generalization. We complement our theoretical analysis with experiments on CIFAR10, CIFAR100, SVHN, and Tiny ImageNet, and show that AVmixup significantly improves the robust generalization performance and that it reduces the trade-off between standard accuracy and adversarial robustness.

1. Introduction

Deep neural networks (DNNs) have produced impressive results for various machine learning tasks, including computer vision [15] and natural language processing [10]. Neural networks, however, can be easily fooled by small adversarial perturbations of their input with a high degree of confidence [34]. This vulnerability of DNNs has led to the proposal of several methods to defend adversarial attacks [27, 21, 30, 41]. Despite these attempts, many of these defenses have been defeated by strong adversarial attacks [16, 18, 3], or were eventually found to rely on obfuscated gradients [1].

Adversarial training [18] is one of the most effective adversarial defense methods which substitutes adversarial examples for the training samples. Given a dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ as an example in the d-dimensional input space and y_i as its associated label, the goal of adversarial training is to train models by using adversarial empirical risk minimization [18]:

$$\min_{\theta} \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\sim D} \left[\max_{\delta \in S} \mathcal{L}(\boldsymbol{x}+\boldsymbol{\delta}, y; \theta) \right]. \tag{1}$$

Here, $\mathcal{L}(\boldsymbol{x}+\boldsymbol{\delta}, y; \theta)$ is the loss function on adversarial examples, and S represents the set of perturbations an adversary can apply to deceive the model, which is normally the set of ℓ_p-bounded perturbations.

Many studies of the properties of these adversarial perturbations have been reported. Gilmer *et al*. [6] noted that the phenomenon of adversarial examples appears because most high dimensional data points in the data distribution are very close to the points that could be adversarial examples. Schmidt *et al*. [31] proved that robust training requires significantly larger sample complexity than that of standard training, postulating that the difficulty of robust training originates from the large sample complexity. Tsipras *et al*. [35] showed that a trade-off may exist between adversarial robustness and standard accuracy. They argued that the features learned during adversarial training differ from those learned during standard training, and attributed the trade-off to this difference.

Recently, Ilyas *et al*. [12] demonstrated that the features used to train deep learning models can be divided into adversarially robust features and non-robust features, and the problem of adversarial examples may arise from these non-robust features. Then, if adversarial examples are features, rather than bugs, it is natural to wonder: *Could we take into account the generalization between "adversarial features" in our adversarial training?* If so, *is the large gap between test accuracy and training accuracy under adversarial perturbations during adversarial training caused by the failure of adversarial feature generalization?*

Motivated by these questions, we present a theoretical model which demonstrates the robust generalization performance changes during adversarial training. Specifically, we

*Correspondence to: Sungroh Yoon sryoon@snu.ac.kr.

identify a generalization problem of adversarial training and show that our proposed method can alleviate the generalization problem. In summary, our paper makes the following contributions:

- We present a theoretical analysis which demonstrates the extent to which the change in the variance of the feature representations affects the robust generalization.
- We uncover *Adversarial Feature Overfitting* (AFO), the phenomenon of the model overfitting to the adversarial features during adversarial training leading to poor robust generalization.
- We propose *Adversarial Vertex mixup* (AVmixup), a soft-labeled data augmentation approach for adversarial training in a collaborative fashion.
- We analyze our proposed method with the results of experiments on CIFAR10, CIFAR100, SVHN, and Tiny Imagenet, and show that AVmixup substantially increases the effectiveness of state-of-the-art adversarial training methods.

2. Background

2.1. Adversarially Robust Generalization

Schmidt *et al*. [31] showed that the sample complexity for robust generalization can be much larger than the sample complexity for standard generalization by constructing a toy example as follows:

Example 1. (Schmidt *et al*.) *Let $\theta^\star \in \mathbb{R}^d$ be the per-class mean vector and let $\sigma > 0$ be the variance parameter. Then the $(\theta^\star, \sigma)$-Gaussian model is defined by the following distribution over $(\boldsymbol{x}, y) \in \mathbb{R}^d \times \{\pm 1\}$*

$$y \overset{u.a.r.}{\sim} \{-1, +1\}, \quad \boldsymbol{x} \overset{i.i.d.}{\sim} \mathcal{N}(y \cdot \theta^\star, \sigma^2 I). \tag{2}$$

Here, the difficulty of the binary classification task is controlled by adjusting the variance parameter σ which implies the amount of overlap between the two classes.

To characterize robust generalization, the definitions of standard and robust classification error are defined as follows (Schmidt *et al*.):

Definition 1. *Let $Q : \mathbb{R}^d \times \{\pm 1\} \to \mathbb{R}$ be a distribution. Then the standard classification error β of a classifier $f : \mathbb{R}^d \to \{\pm 1\}$ is defined as $\beta = \mathbb{P}_{(\boldsymbol{x},y)\sim Q}[f(\boldsymbol{x}) \neq y]$.*

Definition 2. *Let $Q : \mathbb{R}^d \times \{\pm 1\} \to \mathbb{R}$ be a distribution and let $S \in \mathbb{R}^d$ be a perturbation set that the adversary could apply to fool the model. Then the S-robust classification error β of a classifier $f : \mathbb{R}^d \to \{\pm 1\}$ is defined as $\beta = \mathbb{P}_{(\boldsymbol{x},y)\sim Q}[\exists \boldsymbol{\delta} \in S : f(\boldsymbol{x} + \boldsymbol{\delta}) \neq y]$.*

Hence, the ℓ_p^ϵ-robustness is defined as robustness with respect to the perturbation set $S = \{\boldsymbol{\delta} \in \mathbb{R}^d \mid \|\delta\|_p \leq \epsilon\}$. In our work, we focus on ℓ_∞-bounded perturbations, because this is the most common type in the context of adversarial perturbations [18, 16, 41, 40].

To calculate the sample complexities for robust and standard generalization, Schmidt *et al*. [31] used the following linear classifier model:

Definition 3. (Schmidt *et al*.) *Let $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \{\pm 1\}$ be drawn i.i.d. from a $(\theta^\star, \sigma)$-Gaussian model with $\|\theta^\star\|_2 = \sqrt{d}$. Let the weight vector $\boldsymbol{w} \in \mathbb{R}^d$ be the unit vector in the direction of $\bar{\boldsymbol{z}} = \frac{1}{n}\sum_{i=1}^n y_i \boldsymbol{x}_i$. Then the linear classifier $f_{n,\sigma}$ is defined as*

$$f_{n,\sigma} = sgn(\boldsymbol{w}^\top \boldsymbol{x}). \tag{3}$$

It was shown that the linear classifier can achieve satisfactory generalization performance even with a single sample when the variance of the data distribution is small. The upper ℓ_∞-bound of adversarial perturbations was also derived for a certain ℓ_∞^ϵ-robust classification error under the same conditions with standard classification.

2.2. Robust and Non-robust Features

Recent studies [35, 12] considered the adversarial robustness in the existence of a distinction between robust features and non-robust features. They noted that adversarial examples can arise from the non-robust features of input data which are useful for standard classification but have an adverse effect on robust classification [12]. They provided evidence to support the hypothesis by showing that non-robust features alone are sufficient for standard classification but not for robust classification. They also demonstrated that standard training on the set of robust features yields a fairly small robust classification error.

Tsipras *et al*. [35] indicated that the existence of a provable trade-off between standard accuracy and its robustness. They theoretically showed the possibility that adversarial robustness is incompatible with standard accuracy in a simple setting using a Gaussian model. In addition, they emphasized that adversarial training may reduce the contribution of non-robust features to zero with the following lemma:

Lemma 1. (Tsipras *et al*.) *Minimizing the adversarial empirical risk results in a classifier that assigns 0 weight to non-robust features.*

2.3. Soft Labeling

Szegedy *et al*. [33] proposed label-smoothing as a mechanism to regularize the classifier. They argued that maximizing the log-likelihood of the correct label may result in overfitting, and label-smoothing can alleviate the overfitting problem.

Zhang *et al*. [39] introduced a novel data augmentation method named Mixup. Mixup constructs virtual training examples as follows:

$$\tilde{x} = \alpha x_i + (1-\alpha)x_j, \quad \tilde{y} = \alpha y_i + (1-\alpha)y_j. \tag{4}$$

(x_i, y_i) and (x_j, y_j) are two examples drawn at random from the training data, and $\alpha \in [0, 1]$. They showed that Mixup improves generalization on various tasks.

3. Methods

3.1. Theoretical Motivation

In this section, we theoretically analyze the statistical aspects of robust generalization. First, a simple Gaussian data model is used to demonstrate the need to minimize feature representation variance for robust generalization. It is then shown that the optimal model parameter in terms of robust generalization differs from the model parameter which minimizes the adversarial empirical risk using data which consist of robust and non-robust features. Ultimately, we provide evidence that most deep neural networks are not free from AFO by showing that even in our simple Gaussian data model, the robust generalization performance is degraded as the model is overly trained on adversarial examples.

Based on Example 1 and the linear classifier defined in Definition 3, we prove the following theorem:

Theorem 1. *For the variance parameters σ_r and σ_s (subscript r for robust and s for standard), let $\sigma_r = \nu\sigma_s$ where $\nu \in [0, 1]$. Then, the upper bound on the standard classification error of f_{n,σ_s} and the upper bound on the ℓ_∞^ϵ-robust classification error of f_{n,σ_r} be equal with probability at least $\left(1 - 2\exp(-\frac{d}{8(\sigma_s^2+1)})\right) \cdot \left(1 - 2\exp(-\frac{d}{8(\sigma_r^2+1)})\right)$ if*

$$\epsilon \leq \frac{(2\sqrt{n} - 1)(1 - \nu)}{2\sqrt{n} + 4\sigma_s}. \tag{5}$$

(All the proofs of the theorems and corollaries in our work can be found in the supplementary material.) We can see that the theorem is consistent with our intuition. For example, when $\nu = 1$, *i.e.*, when both variances are equal, the probability that the robust generalization ability for $\epsilon > 0$ is the same as the standard generalization ability effectively becomes zero. Thus, to ensure that our model shows robust generalization at the same level as standard generalization, a smaller variance of feature representations is required than that of standard learning.

Corollary 1. *For the variance parameters σ_r and σ_s, let $\sigma_r = \nu\sigma_s$ where $\nu \in [0, 1]$. Let the upper bound on the standard classification error of f_{n,σ_s} and the upper bound on the ℓ_∞^ϵ-robust classification error of f_{n,σ_r} be equal. Then, as σ_r decreases, the upper bound of ϵ increases in proportion to π_{n,σ_s}, which is given by*

$$\pi_{n,\sigma_s} = \frac{2\sqrt{n} - 1}{\sigma_s(2\sqrt{n} + 4\sigma_s)}. \tag{6}$$

Hence, the smaller the variance of feature representations, the more effective the robust generalization performance of the model.

Next, we show the change in the variance of feature representations as we train the model to minimize the adversarial empirical risk. Specifically, we utilize the concept of robust and non-robust features, and show the way in which adversarial training results in AFO in a model similar to that used before [35].

Example 2. *Let $0 < \sigma_A \ll \sigma_B$. Then, the distribution Ψ_{true} is defined by the following distribution over $(\boldsymbol{x}, y) \in \mathbb{R}^{d+1} \times \{\pm 1\}$:*

$$\begin{gathered} y \overset{u.a.r}{\sim} \{-1, +1\} \quad \textit{and} \\ x_1 \sim \mathcal{N}(y, \sigma_A^2), \quad x_2, \ldots, x_{d+1} \overset{i.i.d.}{\sim} \mathcal{N}(\eta y, \sigma_B^2). \end{gathered} \tag{7}$$

Here, x_1 is a robust feature that is strongly correlated with the label, and the other features $x_2, \ldots, x_{d+1}$ are non-robust features that are weakly correlated with the label. Here, $\eta < 1$ is a non-negative constant, which is small but sufficiently large such that a simple classifier attains a small standard classification error.

The difficulty associated with robust learning is that a significantly large sample complexity is required [31]. Given this postulation, we extend Example 2 to Example 3 with the following assumption:

Assumption 1. *Assume the number of non-robust features in our data is N. Then, because of the lack of data samples in robust learning, M features out of N non-robust features form a sample distribution which is far from the true distribution.*

In Assumption 1, we refer to M non-robust features as "insufficient" non-robust features. Contrarily, the other non-robust features are referred to as "sufficient " non-robust features.

Example 3. *Let $0 < c < d$. Then the sample distribution $\Psi_{sample,c}$ which is formed by the sampled input-label pairs $(\boldsymbol{x}, y) \overset{i.i.d.}{\sim} \Psi_{true}$ is defined as follows:*

$$\begin{gathered} y \overset{u.a.r.}{\sim} \{-1, +1\}, \quad x_1 \sim \mathcal{N}(y, \sigma_A^2), \\ x_2, \ldots, x_{c+1} \overset{i.i.d.}{\sim} \mathcal{N}(y, \sigma_A^2), \\ x_{c+2}, \ldots, x_{d+1} \overset{i.i.d.}{\sim} \mathcal{N}(\eta y, \sigma_B^2). \end{gathered} \tag{8}$$

In Example 3, our data has a true distribution as in Example 2. However, the Gaussian distribution is changed for the insufficient non-robust features $x_2, \ldots, x_{c+1}$ in our sampled data according to Assumption 1. For simplicity, in this example, we suppose that the insufficient non-robust features form the same sample distribution as that of the robust features.

We show the variance of feature representations during adversarial training on $\Psi_{sample,c}$ by using the following linear classifier:

Definition 4. *Let Z be a function set. Let $\boldsymbol{w}$ be the weight vector of the classifier. Let ζ_f be the objective function of the linear classifier $f_{\boldsymbol{w}}$. Then our linear classifier f_Z is defined as*

$$f_Z \in \{f_{\boldsymbol{w}} \mid \zeta_f \in Z, \boldsymbol{w} \in \mathbb{R}_+^{d+1}, \|\boldsymbol{w}\|_1 = 1\}. \quad (9)$$

In our model, it is reasonable to investigate the variance of $\boldsymbol{w}^\top \boldsymbol{x}$ to show the extent to which adversarial training affects robust generalization. Based on Example 3 and the linear classifier defined in Definition 4, we can prove the following theorem:

Theorem 2. *Let $\boldsymbol{w}_B \in \mathbb{R}^{d-c}$ be the weight vector for the sufficient non-robust features of $\Psi_{sample,c}$. Let Z_{sc} be a set of strictly convex functions. Then, when the classifier $f_{Z_{sc}}$ is trained on $\Psi_{sample,c}$, the $\boldsymbol{w}_B^\star$ which minimizes the variance of $\boldsymbol{w}^\top \boldsymbol{x}$ with respect to $\Psi_{sample,c}$ is*

$$\boldsymbol{w}_B^\star = \vec{0}. \quad (10)$$

This result is consistent with that of [35], which presumed that the number of samples for all the non-robust features is sufficiently large. However, we have a limited number of samples for the non-robust features in Example 3 and this may cause the result to differ from that of Theorem 2. Therefore, we need to find $\boldsymbol{w}_B^\star$ with respect to the true distribution Ψ_{true} for the purpose of showing robust generalization ability for our model.

Theorem 3. *Let $\boldsymbol{w}_B \in \mathbb{R}^{d-c}$ be the weight vector for sufficient non-robust features of $\Psi_{sample,c}$. Let Z_{sc} be a set of strictly convex functions. Then, when the classifier $f_{Z_{sc}}$ is trained on $\Psi_{sample,c}$, the $\boldsymbol{w}_B^\star$ that minimizes the variance of $\boldsymbol{w}^\top \boldsymbol{x}$ with respect to Ψ_{true} is*

$$\boldsymbol{w}_B^\star = \frac{c}{cd + 2c + 1} \cdot \vec{1}. \quad (11)$$

For simplicity, we assume that the classifier assigns the same weight value to features with the same distribution in Theorem 3, and the limited feasible set does not change the optimal weight of the classifier. As a result, we can predict the robust generalization performance of the classifier by observing $\boldsymbol{w}_B$ in the robust learning procedure. Note that Lemma 1 also applies with our classifier. Therefore, if our sampled data have insufficient non-robust features, $\boldsymbol{w}_B$ approaches $\vec{0}$ during adversarial training, even though the optimal $\boldsymbol{w}_B^\star$ is not $\vec{0}$ in terms of robust generalization. We refer to this phenomenon as *Adversarial Feature Overfitting (AFO)*.

AFO is caused by the relation between the weight values for the features in our data. In this regard, most deep neural networks involve intertwined features, suggesting that most deep neural networks are also adversely affected by the problem we point out in the example.

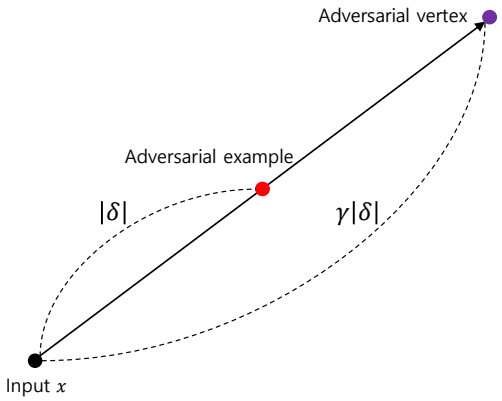

Figure 1: **Adversarial vertex**. The adversarial vertex is located in the same direction as the adversarial example but γ times farther away.

3.2. Adversarial Vertex Mixup

AFO arises when the model is overly optimized only for sufficient non-robust features, when the training data have many types of insufficient non-robust features. From this point of view, we can think of several methods to address AFO. First, the diversity of the algorithm that constructs adversarial examples during training could be increased. This may be a fundamental solution to overcome the poor robust generalization caused by the large sample complexity. Second, when the large sample complexity of robust learning cannot be satisfied, label-smoothing can directly regularize the overfitting problem as in [33]. Essentially, soft labeling can be employed to prevent the weights for the sufficient non-robust features from becoming zero. In this paper, we present a method to improve the robust generalization using soft labeling.

Several algorithms that use soft-labeled data to improve the generalization performance have been proposed [33, 39, 28]. Among them, Mixup [39] trains a model by utilizing linear interpolation between training data. This method can be seen as a variant of the label-smoothing method, because it linearly interpolates both input vectors and their labels. Inspired by Mixup, we propose Adversarial Vertex mixup (AVmixup), which is a soft-labeled data augmentation method designed to improve robust generalization.

AVmixup, similar to Mixup, also extends the training distribution by using linear interpolation. Unlike Mixup, however, for each raw input vector, AVmixup defines a virtual vector in the adversarial direction and extends the training distribution via linear interpolation of the virtual vector and the raw input vector. We refer to the virtual vector as an *adversarial vertex* (see Figure 1). Formally, the adversarial vertex is defined as follows:

Definition 5. *Let $\delta_{\boldsymbol{x}} \in \mathbb{R}^d$ be the adversarial perturbation for the raw input vector $\boldsymbol{x} \in \mathbb{R}^d$. Then, for a scaling factor $\gamma \geq 1$, adversarial vertex $\boldsymbol{x}_{av}$ is defined as*

$$\boldsymbol{x}_{av} = \boldsymbol{x} + \gamma \cdot \delta_{\boldsymbol{x}}. \tag{12}$$

Figure 1 shows how the adversarial vertex is found. After we obtain the adversarial vertex, AVmixup constructs virtual training examples as follows:

Definition 6. *Let $(\boldsymbol{x}, \boldsymbol{y})$ be the raw input-label pair. Let ϕ be a label-smoothing function. Then, for the real value α sampled from a uniform distribution $\mathcal{U}(0,1)$ and the label-smoothing parameters $\lambda_1 \in \mathbb{R}$ and $\lambda_2 \in \mathbb{R}$, the virtual input vector $\hat{\boldsymbol{x}} \in \mathbb{R}^d$ and its associated label $\hat{\boldsymbol{y}} \in \mathbb{R}^k$ are constructed by*

$$\begin{aligned} \hat{\boldsymbol{x}} &= \alpha \boldsymbol{x} + (1-\alpha)\boldsymbol{x}_{av}, \\ \hat{\boldsymbol{y}} &= \alpha\phi(\boldsymbol{y}, \lambda_1) + (1-\alpha)\phi(\boldsymbol{y}, \lambda_2). \end{aligned} \tag{13}$$

For the label-smoothing function ϕ, we use an existing label-smoothing method [33]. Specifically, in the case of k classes, the algorithm assigns $\lambda \in (0,1)$ to the true class and equally distributes $\frac{1-\lambda}{k-1}$ to the other classes.

In summary, the overall procedure of adversarial training with AVmixup is described in Algorithm 1.

Algorithm 1 Adversarial Training with AVmixup

Require: Dataset D, batch size n, training epochs T, learning rate τ, scaling factor γ, label-smoothing factors λ_1, λ_2
Require: Label-smoothing function ϕ
Require: Adversarial perturbation function $\mathcal{G}$
for $t = 1$ **to** T **do**
  for mini-batch $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^n \sim D$ **do**
    $\boldsymbol{\delta}_i \leftarrow \mathcal{G}(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{\theta})$
    AVmixup:
    $\bar{\boldsymbol{x}}_i \leftarrow \boldsymbol{x}_i + \gamma \cdot \boldsymbol{\delta}_i, \quad \alpha_i \sim \mathcal{U}(0,1)$
    $\hat{\boldsymbol{x}}_i \leftarrow \alpha_i \boldsymbol{x}_i + (1-\alpha_i)\bar{\boldsymbol{x}}_i$
    $\hat{\boldsymbol{y}}_i \leftarrow \alpha_i \phi(\boldsymbol{y}_i, \lambda_1) + (1-\alpha_i)\phi(\boldsymbol{y}_i, \lambda_2)$
    model update:
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \tau \cdot \frac{1}{n} \sum_{i=1}^n \nabla_\theta \mathcal{L}(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{y}}_i; \boldsymbol{\theta})$
  end for
end for
Output: robust model parameter $\boldsymbol{\theta}$

4. Related Work

4.1. Adversarial Attack Methods

Adversarial attacks confuse the trained deep neural networks with adversarial examples. The Fast Gradient Sign Method (FGSM) [7] is an efficient one-step attack method. Projected Gradient Descent (PGD) [18] constructs adversarial examples by applying a multi-step variant of FGSM. The Carlini & Wagner (CW) attack [3] uses a specific objective function to create adversarial examples under various conditions. Apart from these attacks, many adversarial attacks exist in white-box settings [26, 23, 16, 22]. In black-box settings, adversarial attacks are conducted using substitute models, according to which adversarial examples are generated from the substitute models [25]. Additionally, black-box attacks which only rely on the prediction score or the decision of the model have been proposed [4, 2, 32, 11, 8].

4.2. Adversarial Defense Methods

Various adversarial defense methods have been employed to make DNNs robust to adversarial attacks. Adversarial training [7] uses adversarial examples as training data to train the robust network. Many approaches [13, 21, 29, 41, 40] improve the model robustness through regularizers or variants of adversarial training. Various techniques [20, 19, 37, 30, 36] can defend adversarial attacks by denoising adversarial perturbations from input data or detect adversarial examples from among the input data. We cover further related works in the supplementary material.

5. Experimental Results And Discussion

In this section, we show that label-smoothing [33] and AVmixup improve the robust generalization with extensive experiments across many benchmark datasets including CIFAR10 [14], CIFAR100 [14], SVHN [24] and Tiny Imagenet [5]. Especially, we note that the combination of AVmixup with the state-of-the-art adversarial defense method [40], would enable us to *significantly outperform existing defense methods*. A description of the datasets used in the experiments is summarized in the supplementary material.

5.1. Implementation Details

We use WRN-34-10 [38] for the experiments on CIFAR, WRN-16-8 [38] for the experiments on SVHN, and PreActResNet18 [9] for the experiments on Tiny Imagenet. We run 80k training steps on CIFAR and SVHN and 50k training steps on Tiny Imagenet. The initial learning rate for CIFAR and Tiny Imagenet is set to 0.1 and 0.01 for SVHN. The learning rate decay is applied at 50% and 75% of total training steps with decay factor 0.1, and weight decay factor is set to $2\mathrm{e}{-4}$. We use the same adversarial perturbation budget $\epsilon = 8$ as in [18]. To evaluate adversarial defense methods, we apply several adversarial attacks including FGSM [7], PGD [18], CW [3] (PGD approach with CW loss) and transfer-based black-box attack [25]. We mainly compare the following settings in our experiments:

1. Standard: The model which is trained with the original dataset.

Table 1: Comparison of the accuracy of our proposed approach AVmixup with that of PGD [18] and LSλ ($\lambda \in \{0.8, 0.9\}$) [33] against white-box attacks on CIFAR10.

Model	Clean	FGSM	PGD10	PGD20	CW20
Standard	95.48	7.25	0.0	0.0	0.0
PGD	86.88	62.68	47.69	46.34	47.35
LS0.8	87.28	66.09	53.49	50.87	50.60
LS0.9	87.64	65.96	52.82	50.29	50.30
AVmixup	**93.24**	**78.25**	**62.67**	**58.23**	**53.63**

Figure 2: **CIFAR10 accuracy curves**. The robustness of the PGD model (blue line) overfits after 40k steps. The AVmixup model, on the other hand, shows a steady increase in robustness (red line).

2. PGD: The model trained using adversarial examples from PGD [18] with step size $= 2$, iterative steps $= 10$.
3. LSλ: With the PGD-based approach [18], we apply the label-smoothing method [33] for the model with label-smoothing factor λ.
4. AVmixup: We apply our proposed method for the model with the PGD-based approach [18].

Note that PGD and CW attacks with T iterative steps are denoted as PGDT and CWT, respectively, and the original test set is denoted as Clean.

5.2. CIFAR10

Because the CIFAR10 dataset is the most commonly used dataset for adversarial robustness studies [18, 37, 41, 40], we analyze our method in both white-box and black-box settings, and compare our method to a state-of-the-art defense method, TRADES [41], on CIFAR10. We set the scaling factor $\gamma = 2.0$ and label-smoothing factors $\lambda_1 = 1.0$ and $\lambda_2 = 0.1$ in the following experiments.

Empirical evidence for AFO We provide Figure 2 in support of our theoretical analysis and the effectiveness of AVmixup. In Figure 2, the validation accuracy curve against PGD10 of the PGD model shows that the model starts to overfit from about 40k steps, while the AVmixup model continues to improve.

White-box setting We conduct white-box attacks on the models trained with baseline methods and our proposed method AVmixup. We set the step size $= 2.0$ for PGD and CW attacks. We first evaluate the models on Clean to compare the trade-off between accuracy and robustness of the models. Then, we evaluate the models against FGSM, PGD10, PGD20, and CW20. The results are summarized in Table 1.

The results in Table 1 indicate that models trained with soft labels are more accurate in all attacks including clean data than the model trained with one-hot labels, which is consistent with our theoretical analysis. In particular, the accuracy on PGD20 of the AVmixup model is 11.89%p higher than that of the PGD model, with a decrease of only 2.24%p in accuracy on Clean compared to the Standard model.

Black-box setting Athalye *et al.* [1] indicated that obfuscated gradients, a phenomenon that leads to non-true adversarial defenses, can be identified in several ways. One such way is black-box attack evaluation.

In black-box settings, we apply transfer-based black-box attacks to the models [25]. After constructing adversarial examples from each of the trained models, we apply these adversarial examples to the other models and evaluate the performances. The results are summarized in Table 2, and more results can be found in the supplementary material. The columns represent the attack models of the transfer-based black-box attacks, and the rows represent the defense models which are evaluated. The results in Table 2 indicate that the AVmixup model is the most robust against black-box attacks from all of the attack models with significant margins. We also observe that the model trained with AVmixup shows higher accuracy against black-box attacks than against white-box attacks. Thus, we confirm that our proposed method improves the adversarial defense performance as a result of an increase in the robustness of the model rather than with obfuscated gradients [1].

Comparison We compare our method with a recently proposed defense method, TRADES [41], which uses a regularization-based adversarial training approach. TRADES requires approximately twice as much GPU memory as conventional adversarial training to calculate the additional regularization term which processes both natural examples and adversarial examples simultaneously. In contrast, AVmixup hardly incurs additional cost and can be implemented with only a few lines of code. In this experiment, we implement AVmixup based on the official Py-

Table 2: Accuracy comparisons against transfer-based black-box attacks (PGD20).

Defense model	Attack model			
	Standard	PGD	LS0.8	LS0.9
PGD	85.6	-	65.70	64.91
LS0.8	86.03	63.60	-	64.83
LS0.9	86.40	63.74	65.78	-
AVmixup	**89.53**	**68.51**	**71.48**	**70.50**

Table 3: Accuracy comparisons with TRADES [41].

Models	Clean	PGD20
PGD [41]	87.3	47.04
TRADES ($1/\lambda = 1$) [41]	88.64	49.14
TRADES ($1/\lambda = 6$) [41]	84.92	56.61
AVmixup	**90.36**	**58.27**

Torch code of TRADES [41] and train the model with the same configurations as [18]. The results are listed in Table 3, which shows that our proposed method has superior robustness with a smaller trade-off than TRADES.

Discussion In Tabel 1, in contrast with FGSM, PGD10 and PGD20, the AVmixup model does not show significant improvement against the CW20 attack, and this trend becomes more severe for challenging datasets that have a larger number of classes and smaller number of training examples per class such as CIFAR100. We can infer that this property appears as AVmixup uses linear interpolations. In other words, algorithms that utilize virtual data constructed using linear interpolation between data points tightly generalize only the features observed in the training steps. We confirm this explanation by a simple experiment, the details and further discussion of which can be found in the supplementary material. It implies that while AVmixup shows a high level of robustness against adversarial attacks used in adversarial training, it may not be able to withstand other types of attacks. Therefore, the diversity of adversarial examples generated during the adversarial training procedure is even more important for AVmixup. We thus report the results of AVmixup combined with the PGD-based approach by focusing on PGD-based attacks. The results using an algorithm that constructs diverse adversarial examples are discussed in Section 5.4.

5.3. Other Datasets

We also verify the effectiveness of our method on CIFAR100, SVHN and Tiny Imagenet. We specify the same hyperparameters for AVmixup as in the CIFAR10 experiments. The results from these experiments are provided in Table 4.

Table 4: Comparisons of AVmixup on SVHN [24], CIFAR100 [14], and Tiny ImageNet [5].

Dataset	Model	Clean	FGSM	PGD20
CIFAR100	PGD	61.29	46.01	25.17
	LS0.8	62.1	52.33	28.81
	LS0.9	61.77	53.17	27.13
	AVmixup	**74.81**	**62.76**	**38.49**
SVHN	PGD	92.4	75.31	58.22
	LS0.8	92.15	75.84	59.75
	LS0.9	92.34	76.14	59.28
	AVmixup	**95.59**	**81.83**	**61.90**
Tiny ImageNet	PGD	41.67	20.30	13.14
	LS0.8	42.89	22.75	15.43
	LS0.9	41.71	20.96	14.03
	AVmixup	**54.27**	**35.46**	**20.31**

CIFAR100 Tabel 4 shows that the accuracy of AVmixup increases by 13.52%p and 13.32%p for Clean and PGD20, respectively, compared to the PGD model. The results of additional experiments on CIFAR100 can be found in the supplementary material.

SVHN The SVHN image classification task is much easier than the image classification tasks with more complicated input images such as CIFAR and Tiny Imagenet. As shown previously [31], generalization problems with poor robustness are less common for simple image datasets such as MNIST than for complex image datasets such as CIFAR. Thus, it is possible to predict that our proposed method, starting from the robust generalization problem, would be less effective on the SVHN dataset than on other datasets, and it is indeed observed from Table 4. The accuracy of AVmixup improves by 3.19%p and 3.68%p compared to the PGD model for Clean and PGD20, respectively, which are small improvements compared to those observed on the other datasets that are tested.

Tiny Imagenet Tabel 4 shows an improvement in accuracy of 12.6%p and 7.17%p compared to the PGD model for Clean and PGD20, respectively.

5.4. When AVmixup Meets Diversity

As discussed in 5.2, the diversity of adversarial examples during adversarial training is important to enable AVmixup to be effective against various adversarial attacks. In this sense, we utilize a recent method [40] (Feature Scatter) which promotes data diversity by taking the inter-sample relationships into consideration during adversarial training. We combine Feature Scatter with our method AVmixup,

Table 5: Comparisons of AVmixup with feature scattering-based approach [40]. For PGD, we refer to the accuracy of [40]. For Feature Scatter, we reproduce and evaluate the model at the end of the training.

Dataset	Model	Clean	FGSM	PGD20	PGD100	CW20	CW100
CIFAR10	PGD [40]	85.7	54.9	44.9	44.8	45.7	45.4
	Feature Scatter	90.22	78.19	69.74	67.35	60.77	58.29
	Feature Scatter + AVmixup	**92.37**	**83.49**	**82.31**	**81.88**	**71.88**	**69.50**
CIFAR100	PGD [40]	59.9	28.5	22.6	22.3	23.2	23.0
	Feature Scatter	74.9	72.99	45.29	42.77	27.35	24.89
	Feature Scatter + AVmixup	**78.62**	**78.92**	**47.28**	**46.29**	**33.20**	**31.22**
SVHN	PGD [40]	93.9	68.4	47.9	46.0	48.7	47.3
	Feature Scatter	**96.42**	**95.92**	58.67	46.98	51.23	38.89
	Feature Scatter + AVmixup	96.07	95.26	**73.65**	**70.24**	**67.06**	**62.01**

Table 6: Sensitivity of the combination of AVmixup and Feature Scatter to label-smoothing factors ($\gamma = 1$) on CIFAR10.

λ_1 / λ_2	Clean	FGSM	PGD20	CW20
0.1 / 0.5	91.94	80.09	74.43	62.87
0.5 / 0.3	92.82	77.81	57.67	55.18
0.5 / 0.7	92.37	**83.49**	**82.31**	**71.88**
1.0 / 0.5	**93.07**	79.55	53.42	56.72

and evaluate the performance of the model on CIFAR10, CIFAR100 and SVHN.

We implement AVmixup on the PyTorch code of Feature Scatter released in [40], hence we use the same model architecture and configuration as in this report [40]. For CIFAR10 and SVHN, we set $(\gamma = 1.0, \lambda_1 = 0.5, \lambda_2 = 0.7)$. For CIFAR100, we set $(\gamma = 1.5, \lambda_1 = 0.3, \lambda_2 = 0.42)$. We evaluate the models at the end of the training. The results are summarized in Table 5.

The joint application of AVmixup with Feature Scatter results in significantly higher accuracy than with Feature Scatter alone. Specifically, on CIFAR10, the combination shows powerful adversarial robustness of 82.31% and 81.88% for PGD20 and PGD100, respectively. Furthermore, our experiments on SVHN demonstrate state-of-the-art robustness against the PGD and CW attacks. Moreover, in contrast with the experimental results of the models trained with the PGD-based approach, the combination of AVmixup and Feature Scatter shows a significant improvement not only for PGD attacks but also for CW attacks.

Note that the results on CIFAR100 differ from those on CIFAR10 or SVHN. The combination also provides state-of-the-art accuracy in all respects, but the increase in accuracy for PGD and CW is small compared to that for other datasets. We can infer the reason for these results from Table 6, which indicates that the combination is sensitive to the label-smoothing factors. In this respect, as the number of labels of the dataset increases, the sensitivity of the combination to soft label values increases, which may destabilize the effect of AVmixup. In addition, we can see that the accuracy on FGSM is slightly higher than that on Clean. This is because of the property of AVmixup, not because of label leaking [17], since the feature scattering-based approach prevents label leaking. Further discussions of the results can be found in the supplementary material.

6. Conclusion

In this work, we identified AFO, the phenomenon that leads to poor robust generalization, and used both theoretical and empirical approaches to show the extent to which soft labeling can help improve robust generalization. We also introduced AVmixup, a soft-labeled data augmentation method, and demonstrated its outstanding performance through extensive experiments. Although AVmixup has shown its excellence in various experiments, AVmixup has the disadvantage of being sensitive to its hyperparameters. This forces the appropriate hyperparameters for AVmixup to be found by line search or exhaustive search, and this task will be time consuming if there are limited resources available. Therefore, we aim to develop advanced algorithms by analyzing in detail the meaning and effects of linear interpolation in AVmixup for future research.

Acknowledgements: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science and ICT) [2018R1A2B3001628], the Brain Korea 21 Plus Project in 2020, Samsung Research Funding & Incubation Center of Samsung Electronics under Project Number SRFC-IT1901-12, and AIR Lab (AI Research Lab) in Hyundai Motor Company through HMC-SNU AI Consortium Fund.

References

[1] 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. 1, 6

[2] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017. 5

[3] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017. 1, 5

[4] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 15–26. ACM, 2017. 5

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. https://tiny-imagenet.herokuapp.com/. 5, 7

[6] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018. 1

[7] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 5

[8] Chuan Guo, Jacob R Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Q Weinberger. Simple black-box adversarial attacks. *arXiv preprint arXiv:1905.07121*, 2019. 5

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016. 5

[10] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012. 1

[11] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. *arXiv preprint arXiv:1804.08598*, 2018. 5

[12] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175*, 2019. 1, 2

[13] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018. 5

[14] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 5, 7

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[16] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016. 1, 2, 5

[17] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016. 8

[18] 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. 1, 2, 5, 6, 7

[19] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–147. ACM, 2017. 5

[20] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017. 5

[21] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018. 1, 5

[22] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017. 5

[23] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016. 5

[24] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 5, 7

[25] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519. ACM, 2017. 5, 6

[26] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. IEEE, 2016. 5

[27] 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)*, pages 582–597. IEEE, 2016. 1

[28] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017. 4

[29] Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018. 5

[30] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018. 1, 5

[31] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems*, pages 5014–5026, 2018. 1, 2, 3, 7

[32] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 2019. 5

[33] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 2, 4, 5, 6

[34] 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. 1

[35] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018. 1, 2, 3, 4

[36] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 501–509, 2019. 5

[37] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017. 5, 6

[38] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 5

[39] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 2, 4

[40] Haichao Zhang and Jianyu Wang. Defense against adversarial attacks using feature scattering-based adversarial training. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 1829–1839. Curran Associates, Inc., 2019. https://github.com/Haichao-Zhang/FeatureScatter. 2, 5, 6, 7, 8

[41] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7472–7482, Long Beach, California, USA, 09–15 Jun 2019. PMLR. https://github.com/yaodongyu/TRADES. 1, 2, 5, 6, 7