Boost Off/On-Manifold Adversarial Robustness for Deep Learning with Latent Representation Mixup

Mengdie Huang
mdhuang1@stu.xidian.edu.cn
Xidian University
Xi'an, China

Yi Xie
xieyi@stu.xidian.edu.cn
Xidian University
Xi'an, China

Xiaofeng Chen*
xfchen@xidian.edu.cn
Xidian University
Xi'an, China

Jin Li
lijin@gzhu.edu.cn
Guangzhou University
Guangzhou, China

Changyu Dong
changyu.dong@newcastle.ac.uk
Newcastle University
Newcastle, UK

Zheli Liu
liuzheli@nankai.edu.cn
Nankai University
Tianjin, China

Willy Susilo
wsusilo@uow.edu.au
University of Wollongong
Wollongong, Australia

ABSTRACT

Deep neural networks excel at solving intuitive tasks that are hard to describe formally, such as classification, but are easily deceived by maliciously crafted samples, leading to misclassification. Recently, it has been observed that the attack-specific robustness of models obtained through adversarial training does not generalize well to novel or unseen attacks. While data augmentation through mixup in the input space has been shown to improve the generalization and robustness of models, there has been limited research progress on mixup in the latent space. Furthermore, almost no research on mixup has considered the robustness of models against emerging on-manifold adversarial attacks. In this paper, we first design a latent-space data augmentation strategy called dual-mode manifold interpolation, which allows for interpolating disentangled representations of source samples in two modes: convex mixing and binary mask mixing, to synthesize semantic samples. We then propose a resilient training framework, *LatentRepresentationMixup* (LarepMixup), that employs mixed examples and softlabel-based cross-entropy loss to refine the boundary. Experimental investigations on diverse datasets (CIFAR-10, SVHN, ImageNet-Mixed10) demonstrate that our approach delivers competitive performance in training models that are robust to off/on-manifold adversarial example attacks compared to leading mixup training techniques.

CCS CONCEPTS

• **Security and privacy → Formal methods and theory of security**; • **Computing methodologies → Machine learning**.

KEYWORDS

deep neural networks, adversarial attack, adversarial robustness, representation learning

ACM Reference Format:
Mengdie Huang, Yi Xie, Xiaofeng Chen*, Jin Li, Changyu Dong, Zheli Liu, and Willy Susilo. 2023. Boost Off/On-Manifold Adversarial Robustness for Deep Learning with Latent Representation Mixup. In *ACM ASIA Conference on Computer and Communications Security (ASIA CCS '23), July 10–14, 2023, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3579856.3595786

1 INTRODUCTION

Deep neural networks (DNNs) have achieved outstanding success in complex machine learning tasks, including computer vision, speech recognition, and natural language processing. However, recent studies have demonstrated that DNNs are susceptible to adversarial examples, which are created using imperceptible perturbations to cause misclassification by the classifier [5, 16, 38, 42]. Adversarial attacks can be categorized into off-manifold and on-manifold attacks based on the space where perturbations are generated [47]. The manifold is a geometric object representing the dataset's underlying distribution, capturing its latent factors. Off-manifold attacks, like FGSM [16], PGD [36], and AutoAttack[10], aim to manipulate input features, while on-manifold attacks, such as OM-FGSM and OM-PGD, target representations in the latent space. Adversarial training (AT) [16] is a key proactive defense mechanism against adversarial attacks that integrates defender-generated adversarial examples into the original training set. AT defenses are divided into off-manifold and on-manifold variants, aiming to construct respective adversarial examples to enhance model robustness [35, 47]. However, AT relies on prior knowledge of attacks, limiting its generalization against novel or unseen attacks.

Motivated by solving this challenge, we focus on generalizing model robustness to various potential adversarial attacks without training with adversarial examples in advance. Previous efforts, such as InputMixup [56], AdaMix [19], AdvMix [34], CutMix [54],

ASIA CCS '23, July 10–14, 2023, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0098-9/23/07…$15.00
https://doi.org/10.1145/3579856.3595786

and PuzzleMixup [29], have used mixed examples and mixed labels to train neural networks for image classification, achieving enhanced robustness. Mixup has also been applied to text classification for improving generalization [7, 18] and robustness [57]. Unlike adversarial training, mixup training does not assume the defender's knowledge about the attack method. However, most research focuses on input-space mixup, synthesizing mixed examples by combining source samples in the input space. Consequently, the resulting sampels may lack realistic semantics, negatively impacting the model's ability to learn meaningful representations. Additionally, such mixed samples might not effectively improve robustness against adversarial attacks, as they may not capture subtle differences between original samples exploited by attacks.

To synthesize mixed examples that satisfy the underlying feature structure of a given dataset, we consider mixing latent representations and then mapping back to the high-dimensional input space. Limited work exists on latent-space mixup training, besides ManifoldMixup [52] and PatchUp [14], which utilize mixed feature maps from a classifier's randomly selected hidden layer as extra training signals. These methods consider off-manifold adversarial attacks but neglect on-manifold adversarial attacks. More critically, the hidden layer of a classifier struggles to capture the full complexity of the underlying data manifold due to limited expressivity. Mixing entangled features may not correspond to real input samples and could disrupt boundary learning. Moreover, the necessary alterations to hidden layers architecture make it difficult to apply these methods to different models flexibly. To tackle these issues, we create mixed examples using interpolation on a manifold captured by an external generative model, which better represents the dataset.

We propose LarepMixup, a learning framework that uses mixed examples to improve general robustness against off/on-manifold adversarial attacks. First, we extract an approximately exact data manifold coordinate system using a generative adversarial network, allowing training and test samples to be projected onto less entangled latent representations. Second, we adopt a mixing mode like convex mixup or binary mask mixup to synthesize on-manifold and off-manifold mixed samples by combining representations in the low-dimensional manifold. Lastly, we fine-tune all layers of the target classifier using an augmented dataset containing mixed examples and original training examples with a softlabel-based cross-entropy loss function. We evaluate the performance of LarepMixup on various DNNs using CIFAR-10, SVHN, and ImageNet-Mixed10. Results demonstrate our method effectively boosts robustness against multiple attacks, such as FGSM, PGD, AutoAttack, DeepFool, CW, OM-FGSM, OM-PGD, Fog, Snow, Elastic, and JPEG.

Our contributions are summarized as follows.

- We design a flexible data augmentation strategy, dual-mode manifold interpolation, for synthesizing mixed examples using convex or binary mask mixing modes. We interpret the rationality of mixed examples in improving robustness in terms of their relative position to adversarial examples.
- We propose LarepMixup, the first mixup-based training framework addressing the threats from off/on-manifold adversarial attacks simultaneously. It boosts the model robustness against perturbations in the input and latent spaces without relying on any prior knowledge of the adversary.
- We capture the approximate manifold of the data distribution $p(x, y|z)$ by learning the latent variable space $\mathcal{Z}$ of the StyleGAN-ADA model. The on-manifold datasets created by projecting high-dimensional inputs to disentangled low-dimensional representations are open-sourced.
- Extensive evaluations on different DNNs and datasets show that our method improves off/on-manifold robustness compared to previous mixup training methods. Notably, we are the first to focus on the performance of the mixup trainied model regarding on-manifold attacks and perceptual attacks, which are recommended for evaluating the generalized robustness of DNNs on unseen regular/adversarial examples.

2 RELATED WORK

2.1 Off-manifold Adversarial Attack

Starting from the adversarial example first shown [50], most existing adversarial attack algorithms focus on input-space perturbations, including optimization-based attacks (e.g., L-BFGS [50], CW [5]), gradient-based attacks (e.g., FGSM [16], BIM [33], PGD [36], MI-FGSM [12], DI-FGSM [53], JSMA [42], DeepFool [38]), and generative model-based attacks (e.g., UAE [46], ATN [2]). Moreover, AutoAttack [10], a strong and reliable attack has gained attention. It's an ensemble of diverse parameter-free attacks, including two white-box PGD versions [10], white-box FAB [9], and black-box [1]. As David et al. [47] showed that regular adversarial examples using input-space perturbations leave the manifold orthogonally, we categorize these attacks as off-manifold adversarial attacks here.

2.2 On-manifold Adversarial Attack

On-manifold adversarial examples were first proposed by David et al. [47], which are crafted by adding perturbations to representations in the latent space. Ajil et al. [24] considered finding the representation pairs that can map to similar pixel-level samples but with different predicted labels. Recently, Lin et al. [35] designed On-Manifold FGSM and On-Manifold PGD attacks. These works have demonstrated that on-manifold adversarial attacks could easily fool DNN classifiers trained by off-manifold adversarial training.

2.3 Input-space Mixup

Mixup training, first proposed by Zhang et al. [56], trains classifiers using convex combinations of pixel-level examples (samples and labels). However, mixed examples created in the input space through linear combination (e.g., AdvMix [34], MI [41]) or binary mask combination (e.g., CutMix [54], CutMix [54], PuzzleMixup [29]) are perceptually unnatural and can't be considered samples drawn from the underlying data distribution. Moreover, it is challenging to effectively use input-space interpolation ratio information in the feature space to modify the decision boundary.

2.4 Latent-space Mixup

To smooth the decision boundary of deep neural networks, Verma et al. propose to combine features maps of different inputs in the random selected hidden layer of a classifier via ManifoldMixup [52] or PatchUp [14]. In their work, representations of the data manifold are roughly described as features maps in the DNN. To this end, we further study latent interpolation work in unsupervised learning, including autoencoder-based [3, 4, 6] and GAN-based [26–28] methods. Ultimately, to the best of our knowledge, mixup training work focusing on on-manifold adversarial robustness has not yet been presented in the fields of image classification, text classification [7, 18, 57] and speech classification [13]. Thus, we were motivated to solve this issue.

3 PRELIMINARIES AND THREAT MODEL

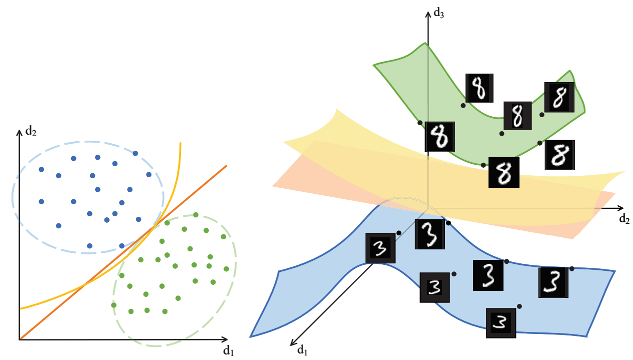
3.1 Object Manifold and Decision Boundary

3.1.1 Object Manifold. According to the manifold hypothesis, high-dimensional data in the real world lie on low-dimensional manifolds embedded within the high-dimensional space [15, 22]. For example, 28×28 pixels samples in the MNIST dataset can be seen as data points on a low-dimensional data manifold embedded in a 784-dimentional feature space, which is supported by the underlying distribution of the dataset. In this work, an object manifold refers to a dataset consisting of samples belonging to the same class, which is consistent with the explanation of object manifolds in the human visual hierarchy [8]. The data points determined by two features (d_1, d_2) and three features (d_1, d_2, d_3) are illustrated in Fig.1 (a) and Fig.1 (b), respectively. The dimension of the data manifold depends on the degree of freedom that can be varied for generating the dataset [44]. When the dataset can be generated by changing the rotation angle, the corresponding object manifold will be a 1-dimentional curve embedded in the feature space. Similarly, when the dataset can be generated by changing the rotation angle and scaling transformation, the corresponding object manifold will be a 2-dimentional hypersurface embedded in the feature space.

3.1.2 Decision Boundary. In the two-class classification task, the feature space learned by the classifier will be partitioned into two subspaces by the decision boundary, one subspace for each class. For the feature space embedded with 1-dimentional object manifolds, the decision boundary of linear classifiers and non-linear classifiers will be a straight line and a curve, respectively, as shown in Fig.1 (a). For the feature space in which at least one 2-dimentional object manifold is embedded, the decision boundary of linear classifiers and non-linear classifiers will be a hyperplane and a hypersurface, respectively, as shown in Fig.1 (b). Similarly, when the problem is extended to a multi-class classification task, assuming $|\mathcal{Y}|$ categories, the feature space will be partitioned into $|\mathcal{Y}|$ subspaces by the decision boundary, one subspace for each class.

3.2 Threat Model

Our threat model focuses on untargeted adversarial attacks against deep learning models. These attacks aim to deceive the model into misclassifying input samples without targeting any specific class or output. Potential attackers include white-box (with network architecture and weight access), grey-box (knowing only network architecture), and black-box (lacking architecture and weight information) adversaries. This work mainly focuses on white-box and gray-box attackers, since they are more powerful from the perspective of the adversary. In addition, we also consider a special attack, AutoAttack, which is a collection of two versions of white-box PGD attack, white-box FAB attack and black-box SquareAttack.



(a) 2D feature space **(b) 3D feature space**

Object Manifold of '3' Decision Line / Hyperplane
Object Manifold of '8' Decision Curve / Hypersurface

Figure 1: Interpreting object manifolds and decision boundaries in the binary classification task.

There are two attack surfaces in this threat model: the input interface of DNN and the corresponding high-level representation of the input, which are respectively formalized into the following two types of attacks: off-manifold adversarial example attack and on-manifold adversarial example attack.

3.2.1 Off-manifold Adversarial Attack. An adversarial example x_{adv} in an off-manifold adversarial attack is created by adding imperceptible adversarial perturbation δ to the original image $x \in \mathcal{X} := \mathbb{R}^{H \times W \times C}$ in the input space. Formally, the objective of untargeted attacks is

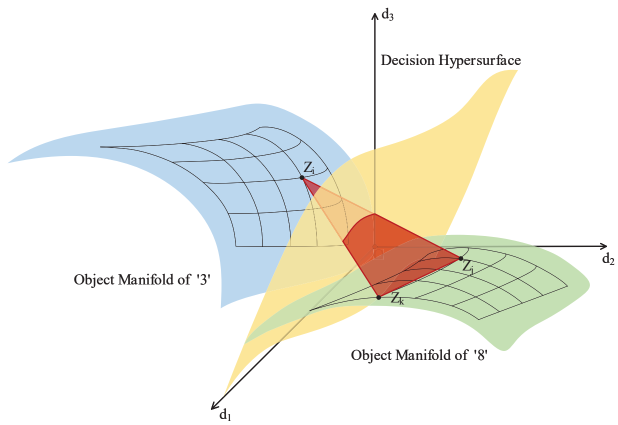
$$\max_{\delta} L(f_\theta(x + \delta), y_{true}), \tag{1}$$

where $\|\delta\|_p < \epsilon$. f_θ denotes a classifier model w.r.t the network parameters θ, y_{true} denotes the ground truth label, y_{target} denotes the target label adversary desired, and ϵ denotes the norm bound of the perturbation in the p-norm bounded attacks, such as L_0 [48], L_2 [2, 5, 37, 38, 42, 50], and L_∞ [5, 16, 32, 33, 37, 42] attacks. Most conventional adversarial examples leave the original object manifolds and can be generated by searching for adversarial perturbations in the input space using various techniques.

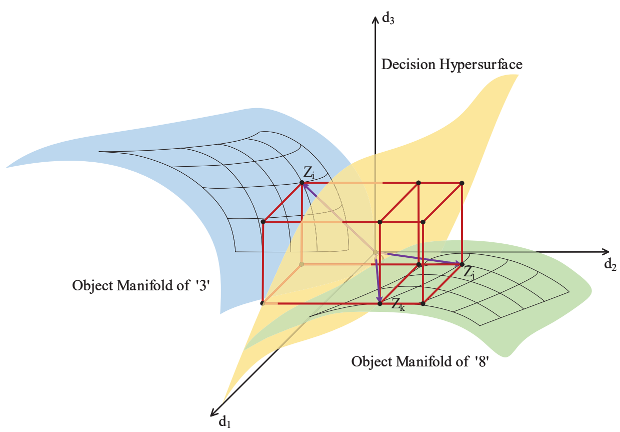
3.2.2 On-manifold Adversarial Attack. Novel on-manifold adversarial attacks aim at adding slight adversarial perturbation ζ to the n-dimentional latent representation $z \in \mathcal{Z} := \mathbb{R}^n$ corresponding to the original image x. Formally, the objective of untargeted attacks is

$$\max_{\zeta} L(f_\theta(G_\varphi(z + \zeta)), y_{true}), \tag{2}$$

where $\|\zeta\|_p < \eta$. G_φ denotes a generative model w.r.t the network parameters φ, that can map any latent representation in $\mathcal{Z}$ to its corresponding input-space sample in $\mathcal{X}$, and η denotes the norm bound of the adversarial perturbation ζ in on-manifold attacks. Typical attacks in this realm include [24, 35, 47]. On-manifold adversarial examples are essentially generalization errors and can be computed using an approximation of the data manifold corresponding to the underlying data distribution of the given dataset.



(a) Convex Combination-based Manifold Interpolation.



(b) Binary Mask Combination-based Manifold Interpolation.

Figure 2: Interpreting the spatial relationship of interpolation points and object manifolds.

4 MANIFOLD INTERPOLATION STRATEGY

In this section, we design a data augmentation strategy to synthesize on-manifold and off-manifold mixed examples.

4.1 Dual-mode Manifold Interpolation

In this work, manifold interpolation refers to the approach of constructing new representation points by combining disentangled representations on object manifolds embedded in the latent space. To infer the manifold coordinate system consisting of degrees of freedom as accurately as possible, a generative adversarial network is utilized for projecting samples in the input space $x \in \mathcal{X}$ to the latent representation in the embedding space $z \in \mathcal{Z}$: $x = G(z)$ and synthesizing high-dimensional mixed samples from the low-dimensional mixed representation z_{mix}: $x_{mix} = G(z_{mix})$. We propose two kinds of mixing modes: convex combination and binary mask combination.

4.1.1 Convex Combination-based Manifold Interpolation. We first design a convex combination-based manifold interpolation, which targets continuously creating mixed representation points along a certain direction in the latent feature space, as shown in Fig.2 (a). For z_i, z_j, interpolations constructed by dual convex combination are located on the line segment between z_i and z_j. For z_i, z_j, z_k, interpolations constructed by ternary convex combination are located on the plane enclosed by the z_i, z_j, z_k.

Dual Convex Combination. For latent representations z_i, z_j corresponding to any two samples x_i, x_j in the training set, the mixed latent representation (z_{mix}, y_{mix}) is created as

$$\begin{aligned} z_{mix} &= \alpha z_i + (1-\alpha) z_j, \\ y_{mix} &= \alpha y_i + (1-\alpha) y_j, \end{aligned} \tag{3}$$

where the coefficient scalar $\alpha \in [0, 1]$ is randomly sampled from the Beta(β) distribution. We work out the mixed label using the same coefficient, which follows the prior knowledge that linear interpolations of feature vectors should lead to linear interpolations of the associated labels.

Multivariate Convex Combination. For latent representations $z_1, \dots, z_k$ corresponding to any k samples $x_1, \dots, x_k$ in the training set, the mixed latent representation (z_{mix}, y_{mix}) is created as

$$\begin{aligned} z_{mix} &= \alpha_1 z_1 + \dots + \alpha_k z_k, \\ y_{mix} &= \alpha_1 y_1 + \dots + \alpha_k y_k, \end{aligned} \tag{4}$$

where the coefficient vector $\alpha \in A := \{R^k : \alpha_i \in [0, 1], \sum_{i=1}^{k} \alpha_i = 1\}$ is sampled from the Dirichlet(γ) distribution with $dim(\gamma) = k$.

4.1.2 Binary Mask Combination-based Manifold Interpolation. We further design binary mask combination-based manifold interpolation, which targets recombining the components of source representation vectors to synthesize the mixed samples, as shown in Fig.2 (b). For z_i, z_j, interpolations constructed by dual binary mask combination are located on the vertices of the polyhedrons formed by the components of z_i, z_j. For z_i, z_j, z_k, interpolations constructed by ternary binary mask combination are located on the vertices of the polyhedrons formed by the components of z_i, z_j, z_k.

Dual Binary Mask Combination. For n-dimensional latent representations z_i, z_j corresponding to any two samples x_i, x_j in the training set, the mixed representation (z_{mix}, y_{mix}) is created as

$$\begin{aligned} z_{mix} &= m \odot z_i + (1_B - m) \odot z_j, \\ y_{mix} &= \lambda y_i + (1-\lambda) y_j, \end{aligned} \tag{5}$$

where the coefficient vector $m \in B := \{0, 1\}^n$ is randomly sampled from the n-fold Bernoulli(p) distribution, the coefficient scalar $\lambda = \frac{n_{m_i=1}}{n}$ is worked out according to the proportion of the number of non-zero elements $n_{m_i=1}$ in the binary coefficient vector m to the dimension n of itself, 1_B denotes a binary mask filled with ones, and $\odot$ denotes the element-wise multiplication.

Multivariate Binary Mask Combination. For n-dimensional representations $z_1, \dots, z_k$ corresponding to k samples $x_1, \dots, x_k$ in the training set, the mixed representation (z_{mix}, y_{mix}) is created as

$$\begin{aligned} z_{mix} &= m_1 \odot z_1 + \dots + m_k \odot z_k, \\ y_{mix} &= \lambda_1 y_1 + \dots + \lambda_k y_k, \end{aligned} \tag{6}$$

where the coefficient vectors $m_i \in B := \{0, 1\}^n$ and $\sum_{i=1}^{k} m_i = 1_B$. m_1 is firstly sampled from the n-fold Bernoulli(p) distribution, and then q non-zero elements in the vector $1_B - m_1$ are replaced with binary values sampled from the q-fold Bernoulli distribution, to obtain the vector m_2. Subsequent m_i is sampled in the same way.

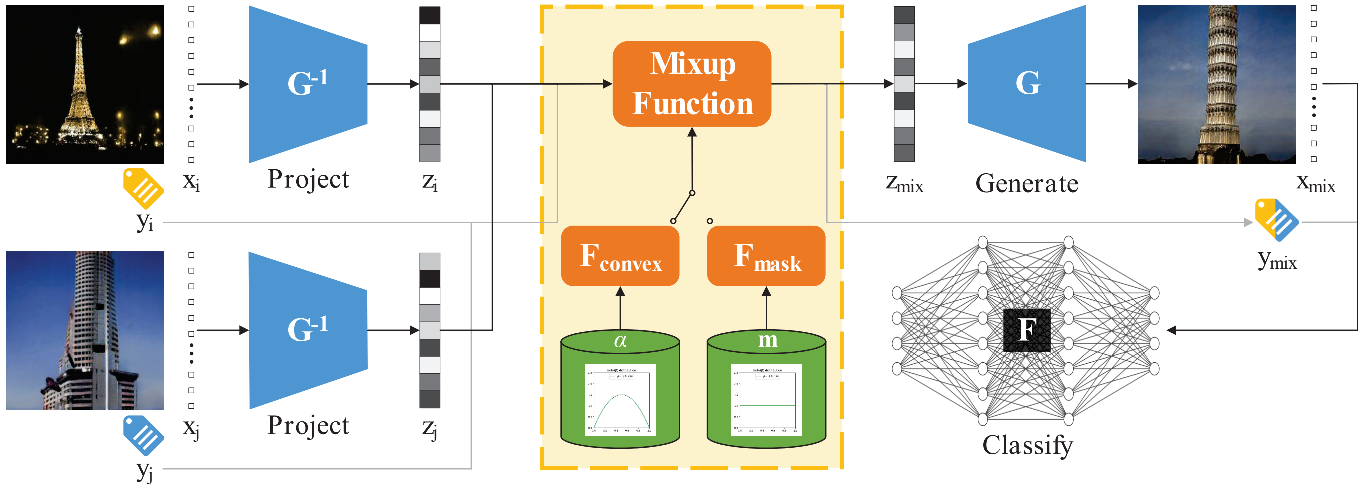


Figure 3: Framework for Latent Representation Mixup (LarepMixup) Training. It consists of three main stages: low-dimensional manifold embedding (left), latent representations mixup (middle), and softlabel-based training (right).

4.2 Interpretation of Mixed Examples in Improving Off/On-manifold Robustness

4.2.1 Off-Manifold Adversarial Robustness. For source representations located on different object manifolds, that is to say, when the source samples belong to different categories, the mixed sample formed by the manifold interpolation strategy will leave all source object manifolds and be closer to the decision boundary than at least one of the source samples. Since conventional adversarial attacks essentially generate off-manifold adversarial examples [35, 47], the mixed samples augmented based on the proposed interpolation strategy can cover some off-manifold adversarial samples in the consistent feasible region. Thus, by learning from the off-manifold mixed samples and corresponding mixed softlabels, the decision boundary of the classifier will be encouraged to yield lower champion confidences for points lying in regions between the object manifolds, presenting smoother. Particularly, the area covered by interpolation points is not restricted to any specific attack, so the robustness improvement can be generalized to some unseen attacks.

4.2.2 On-Manifold Adversarial Robustness. For source representations within the same object manifold, that is to say, when the source samples belong to the same category, the mixed sample formed by the manifold interpolation strategy will be close to or lies within the source object manifold, which can be regarded as the unseen samples meeting the underlying data distribution, such as the on-manifold adversarial sampels [24, 35, 47]. Thus, by training on on-manifold mixed examples, the classifier will be encouraged to learn an approximate manifold that is closer to the underlying manifold of the dataset, that is to say, the hidden layer of the classifier will be encouraged to learn high-level representations that are closer to the real latent variables that support the underlying data distribution of the given datset. On-manifold adversarial robustness is essentially the generalization of a DNN model to unseen samples within a manifold, thus, fine-tuning with the on-manifold mixed examples can be beneficial to boost the on-manifold robustness.

5 LAREPMIXUP TRAINING FRAMEWORK

A geometric illustration of the Latent Representation Mixup (LarepMixup) training framework is shown in Fig.3. Raw samples (x_i, x_j) are projected into latent representations (z_i, z_j) at first. Then, source representations and labels are separately combined in the interpolation module using a mixup function with optional mixing modes. Finally, the target model F is fine-tuned using softlabel-based cross-entropy loss on mixed labels y_{mix} and samples x_{mix}, which are synthesized from mixed representation z_{mix}.

5.1 Low-dimensional Manifold Embedding

In our work, the StyleGAN2-ADA network [26] is adopted to project images into the latent space, which excels at learning disentangled variance factors to represent the latent space of complex training datasets [27]. We use 1000 iterations of gradient descent to find the disentangled latent code z, which is mapped from the randomly sampled code z_{ran} through the mapping network F_{map} in the StyleGAN. The low-dimensional manifold embedding method in LarepMixup is summarized in the Algorithm 1. The loss term for optimizing the representation is defined as the combination of the image quality term and regularization term $L_{total}(G(z), x) = L_{image} + R_{noise}$, following the definition in the original work, where L_{image} signifies the LPIPIS distance between x and $G(z)$, and R_{noise} indicates the sum of squares of the noise map resolution autocorrelation coefficients. In LarepMixup, on-manifold datatset is denoted as $D_M = \{G(z_i), y_i))\}_{i=1}^{N}$, where N is the number of samples selected from training set D_{tra}, $z_i = G^{-1}(x_i)$ is the result of projecting $x_i \in D_{tra}$ into the latent space via synthesis network G, and y_i is the ground truth label corresponding to x_i.

5.2 Latent Representations Mixup

We implemented dual mixup and the ternary mixup interfaces, each of which supports both convex and binary mask mixing modes. To enhance off/on-manifold adversarial robustness concurrently, we mix source samples from different and identical classes.

Algorithm 1 Low-dimensional Manifold Embedding

Input: examples $(x, y) \in D_{tra}$, iteration number T of the optimization, dynamic learning rate η.
Output: n-dim representations $(z, y) \in Z_{tra}$, on-manifold examples $(G(z), y) \in D_M$.

pretrain G on D_{tra}
for $i = 1$ to N **do**
  $t \leftarrow 0$
  sample $z_{ran,i} \sim \text{Normal}(0, 1)$
  $z_{i,t} \leftarrow F_{map}(z_{ran,i})$
  while $t < T$ **do**
    generate $G(z_{i,t})$
    $z_{i,t+1} \leftarrow z_{i,t} - \eta(\nabla_{z(i,t)} L_{total}(G(z_{i,t}), x_i))$
    $t \leftarrow t + 1$
  end while
  $z_i \leftarrow z_{i,t+1}$
  add (z_i, y_i) to Z_{tra}
  add $(G(z_i), y_i)$ to D_M
  $i \leftarrow i + 1$
end for

Algorithm 2 Dual Latent Representations Mixup

Input: batch of n-dim representations (Z_{ori}, Y_{ori}), mixing mode e, index shuffle function F_{shu}, mixing coefficient transform function F_{tra}.
Output: batch of mixed examples (X_{mix}, Y_{mix}).

$(Z_{shu}, Y_{shu}) \leftarrow F_{shu}(Z_{ori}, Y_{ori})$
if $e = ConvexMixup$ **then**
  sample $\alpha \sim \text{Beta}(\beta)$
  $Z_{mix} \leftarrow \alpha Z_{ori} + (1 - \alpha) Y_{ori}$
  $Y_{mix} \leftarrow \alpha Y_{ori} + (1 - \alpha) Y_{shu}$
end if
if $e = MaskMixup$ **then**
  sample $p \sim \text{Uniform}(0, 1)$
  sample $m \sim n$-fold Bernoulli(p)
  $Z_{mix} \leftarrow m \odot Z_{ori} + (1_B - m) \odot Z_{shu}$
  $\lambda \leftarrow F_{tra}(m)$
  $Y_{mix} \leftarrow \lambda Y_{ori} + (1 - \lambda) Y_{shu}$
end if
$X_{mix} \leftarrow G(Z_{mix})$
output (X_{mix}, Y_{mix})

5.2.1 Dual Latent Representations Mixup. Algorithm 2 presents the dual representations mixup method. For a batch of representations with the batch size of *batchsize*, it combines with its shuffled version, enabling a mixing space of $batchsize^2$. The mixing mode is specified by the enumerated parameter e.

5.2.2 Ternary Latent Representations Mixup. Ternary latent representations mixup method is given in the Algorithm 3. A batch of representations will be combined with the objects obtained by shuffling itself twice, so the mixing space can reach $batchsize^3$. Relative to dual mixup, three-sample interpolation spans a broader area, like the triangle in Fig.2 (a). Moreover, k-source latent representation mixup expands the mixing space to a larger volume of $batchsize^k$.

Algorithm 3 Ternary Latent Representations Mixup

Input: batch of n-dim representations (Z_{ori}, Y_{ori}), mixing mode e, index shuffle function F_{shu}, nonzero element counting function $F_{nonzero}$, function $F_{rep}(a, b)$ to replace nonzero elements in a with b.
Output: batch of mixed examples (X_{mix}, Y_{mix}).

$(Z_{shu1}, Y_{shu1}) \leftarrow F_{shu}(Z_{ori}, Y_{ori})$
$(Z_{shu2}, Y_{shu2}) \leftarrow F_{shu}(Z_{ori}, Y_{ori})$
if $e = ConvexMixup$ **then**
  sample $\alpha = (\alpha_1, \alpha_2, \alpha_3) \sim \text{Dirichlet}(\gamma)$
  $Z_{mix} \leftarrow \alpha_1 Z_{ori} + \alpha_2 Z_{shu1} + \alpha_3 Z_{shu2}$
  $Y_{mix} \leftarrow \alpha_1 Y_{ori} + \alpha_2, Y_{shu1} + \alpha_3 Y_{shu2}$
end if
if $e = MaskMixup$ **then**
  $n_1 \leftarrow n$
  sample $p_1 \sim \text{Uniform}(0, 1)$
  sample $m_1 \sim n_1$-fold Bernoulli(p_1)
  $num_{nonzero} \leftarrow F_{nonzero}(1_B - m_1)$
  $n_2 \leftarrow num_{nonzero}$
  sample $p_2 \sim \text{Uniform}(0, 1)$
  sample $temp \sim n_2$-fold Bernoulli(p_2)
  $m_2 \leftarrow F_{rep}(1_B - m_1, temp)$
  $m_3 \leftarrow 1_B - m_1 - m_2$
  $z_{mix} \leftarrow m_1 \odot Z_{ori} + m_2 \odot Z_{shu1} + m_3 \odot Z_{shu2}$
  $\lambda_1, \lambda_2, \lambda_3 \leftarrow F_{tra}(m_1, m_2, m_3)$
  $y_{mix} \leftarrow \lambda_1 Y_{ori} + \lambda_2 Y_{shu1} + \lambda_3 Y_{shu2}$
end if
$X_{mix} \leftarrow G(Z_{mix})$
output (X_{mix}, Y_{mix})

5.3 Softlabel-based Training

The vanilla classifier, trained on normal samples, is designed to be fine-tuned on an augmented dataset containing mixed examples $(x_{mix}, y_{mix}) \in D_{mix}$ and original examples $(x_{ori}, y_{ori}) \in D_{tra}$ to learn a robust decision boundary while avoiding overfitting to the mixed examples and knowledge loss on the original examples. For the augmented example x_{mix} with the soft mixed label y_{mix} (label vectors having two or three non-zero elements summing to 1), cross-entropy loss based on the one-hot label is inapplicable. Instead, we separately calculate the cross-entropy loss for mixed examples on multiple target labels and combine them with the same coefficient α used for the sample mixing. The objective of the softlabel-based training is formalized as

$$\min_{\theta} \mathbb{E}_{(x,y) \sim D_{tra} \cup D_{mix}} L_{soft}(f_\theta(x), y). \tag{7}$$

LarepMixup, proposed from the perspective of implicit regularization based on data augmentation, is broadly applicable to common deep neural networks as it does not depend on any modification of the network structure. While we use images to illustrate our framework, by replacing the StyleGAN-based manifold embedding method designed for images in 5.1 with a suitable representation learning algorithm for other input instances, such as the Bert-based representation encoding method for text features, our approach can be readily extended to other input domains.

6 EXPERIMENTS

We first present our constructed on-manifold CIFAR-10 and the perception of mixed samples. Then, we analyze the effect of varying perturbation budgets on robustness improvements. Following this, we compare our method with state-of-the-art defense methods, including mixup training with the same defensive capability assumptions as ours, and adversarial training with stronger defensive capability assumptions than ours, using CIFAR-10 and SVHN datasets. Next, we evaluate the robustness to unseen attacks simulated by perceptual attacks. Finally, we demonstrate the adaptability of the proposed method on higher-dimensional datasets based on ImageNet, and analyze the impact of different mixing modes on improving model robustness. Source code and on-manifold dataset are available: https://github.com/2022Submit/LarepMixup.

6.1 Experimental Setup

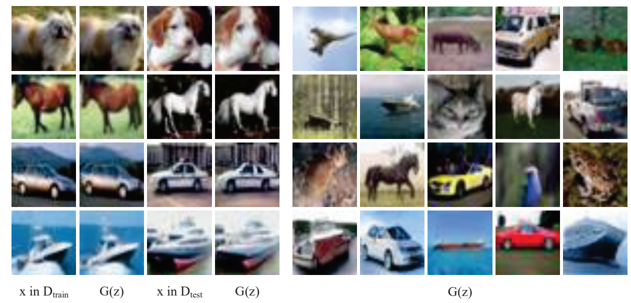
6.1.1 Testbed. We developed the project using PyTorch 1.8.1 [43] and CUDA V11.1.74. Experiments were conducted on an NVIDIA GV102 GPU. Off-manifold attacks were implemented with the Adversarial Robustness Toolbox [40], while on-manifold attacks were achieved by aggregating styleGAN and advertorch [11].

6.1.2 Dataset. Standard color-channel datasets, CIFAR-10 [30], SVHN [39] and ImageNet-Mixed10 [35] are used in our experiments. CIFAR-10 consists of 3×32×32 samples of 10 categories $\{airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck\}$, each of which has 50, 000 training samples and 10, 000 test samples. SVHN consists of 3×32×32 samples of 10 categories { '1' , '2' , '3' , '4' , '5' , '6' , '7' , '8' , '9' , '0' }, including 73, 257 training samples and 26, 032 testing samples. ImageNet-Mixed10 consists of 3×256×256 samples of 10 categories $\{dog, bird, insect, monkey, feline, truck, fruit, horse, fungus, boat\}$ picked from the ImageNet dataset [31], including 77, 237 training samples and 3, 000 testing samples.

6.1.3 Classifier Architectures. To analyze the universality of our method on different classifier architectures, we used a series of base models implemented in the Torchvision library[43], including convolutional block-based networks (Alexnet [31] and VGG [45]), residual block-based network (ResNet [20] and DenseNet [23]), and inception block-based network (GoogLeNet [49]). To maintain fairness when comparing various DNNs on the same dataset or different datasets on a single model, we did not modify any base architecture and used uniform parameters during dataset preprocessing. Additionally, we conducted experiments on the PreActResNet18/34/50 [21] and WideResNet28-10 [55] adopted in the compared mixup training schemes. More training details of the initial model are given in the Appendix.

6.2 Perception Analysis

Realistic perception is an essential requirement for augmented examples generated by mixup methods because unnatural semantic information in mixed examples can mislead the classifier and weaken the generalization of the model[29, 54]. Experimental results show that the manifold learned by LarepMixup is almost identical to the underlying data manifold of the CIFAR-10 dataset, and the mixed examples synthesized by LarepMixup have meaningful semantics.



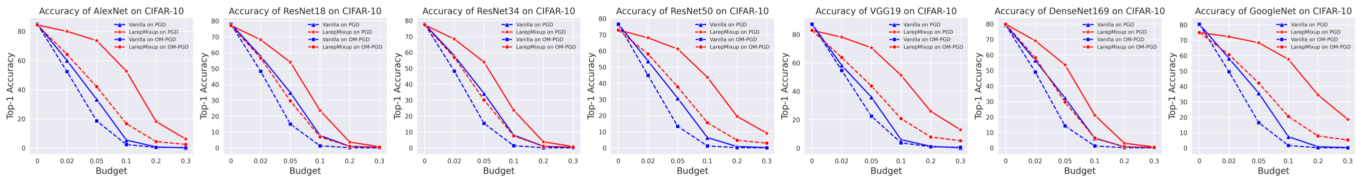
(a) CIFAR-10 dataset projection. **(b) On-manifold CIFAR-10 sampling.**

(c) Dual LarepMixup. **(d) Ternary LarepMixup.**

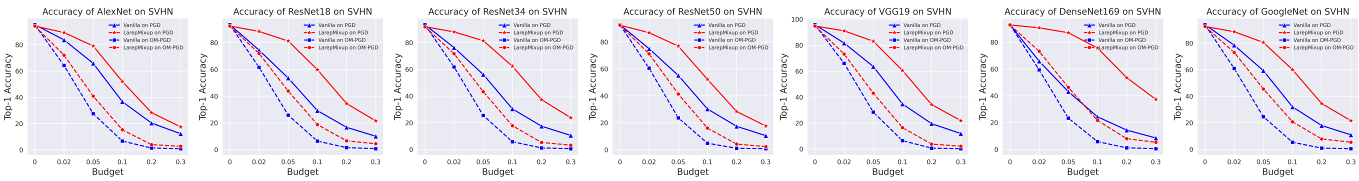
Figure 4: On-manifold dataset and mixed samples.

6.2.1 On-manifold Dataset. We train a StyleGAN2-ADA network with a 512-dimensional latent space on CIFAR-10. We then optimize a 512-dimensional latent representation vector for each training and testing sample to build a manifold representation set for CIFAR-10. Similarly, we also build respective on-manifold representation sets for SVHN and ImageNet-Mixed10 respectively. Taking CIFRA-10 as an example, it can be seen from Fig.4 (a) that when the testing samples are projected into the latent space learned on the training set, the reconstructed samples from latent representations are almost the same as the original test samples. This indicates that the data distribution supported by our learned manifold is close to the true data distribution. Moreover, we generate unknown on-manifold samples by randomly sampling representations in the manifold embedding space, as shown in Fig.4 (b). The natural semantics of synthesized samples also proves that the manifold we constructed approximates the underlying data manifold.

6.2.2 Mixed Examples. The perception of convex mixed samples and binary mask mixed samples are shown in Fig.4 (c) and (d), respectively. For convex mixup, the synthesized examples show more smooth mixed characteristics between source samples, like luma, color, and contour, since the combination coefficient α can take a value from the continuous range, $[0, 1]$. Each specific feature in the convex mixed image that corresponds to a dimension of the latent representation will show the merged value of the scaled features of the source samples with a high probability. For binary mask mixup, the synthesized examples show fewer transitions between source features, because the combination coefficient m is discrete and can only be taken from the binary set $\{0, 1\}^n$. Each specific feature in the binary mask mixed image preserves either the feature of one source sample or the other.

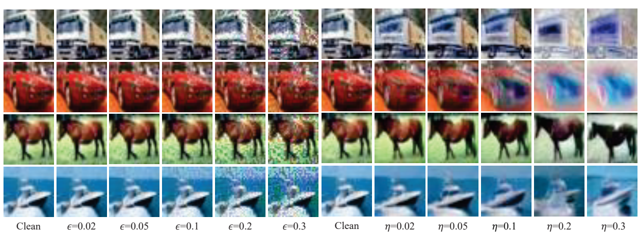


(a) CIFAR-10

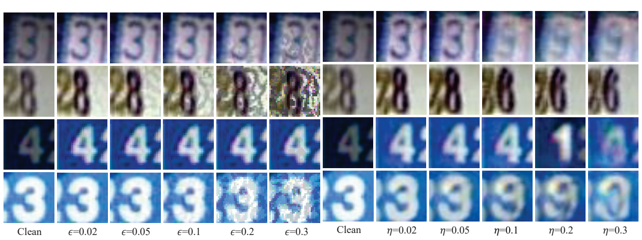


(b) SVHN

Figure 5: Accuracy of various LarepMixup trained models under different attack budgets. PGD budget ϵ and OM-PGD budget η are set sequentially as $\{0.02, 0.05, 0.1, 0.2, 0.3\}$.



(a) PGD on CIFAR-10 (b) OM-PGD on CIFAR-10



(c) PGD on SVHN (d) OM-PGD on SVHN

Figure 6: Visualization of PGD and OM-PGD examples.

6.3 Evaluations on Different Attack Budgets

To verify the effectiveness of LarepMixup in improving off/on-manifold adversarial robustness of the model under different adversary attack strengths, we evaluate the top-1 accuracy of the classifier on PGD and OM-PGD adversarial examples with different budgets of L_∞-bounded perturbations. For PGD attack, the single-step budget is 0.02; for OM-PGD attack, it's 0.005.

Fig.6 shows the perception of PGD and OM-PGD adversarial examples under varying perturbation strengths. In Fig.6 (a) and (c), off-manifold samples (PGD) display granular noise, , which is caused by the adversarial perturbation directly superimposed on the pixels. In Fig.6 (b) and (d), on-manifold samples (OM-PGD) exhibit smooth noise due to perturbations on low-dimensional latent representations, affecting high-level features like direction and style. As the perturbation budget increases, semantic changes become more drastic. However, as emphasized in [47], care needs to be taken that when implementing on-manifold attacks, label invariance should be considered. Careful control of perturbation budget is needed to avoid changing the original class manifold, which would create invalid on-manifold adversarial samples, as shown in the last two columns of Fig.6 (d).

It can be seen from Fig.5 that LarepMixup training notably enhances robustness against different off/on-manifold adversarial attack strengths on the CIFAR-10 and SVHN datasets. For PGD on CIFAR-10 with ϵ set to 0.02, 0.05, 0.1, 0.2, 0.3, the average accuracy of the seven models improves by 14.54%, 28.36%, 32.32%, 14.57%, 6.78%, respectively. For OM-PGD on CIFAR-10 with η set to 0.02, 0.05, 0.1, 0.2, 0.3, the average accuracy of the seven models improves by 10.18%, 19.93%, 11.64%, 3.60%, 2.26%, respectively. Under the same settings, for PGD on SVHN, the average classification accuracy of the seven models improves by 12.51%, 24.67%, 29.27%, 18.13%, 12.38%, respectively. For OM-PGD on SVHN, the average classification accuracy of the seven models improves by 10.27%, 17.93%, 12.21%, 4.53%, 3.05%, respectively.

An remarkable observation is that when the budget η is too large, e.g., exceeds 0.1, the improvement in robust accuracy for on-manifold attacks diminishes. In conjunction with Fig.6, we deduce that this occurs because an excessive attack budget generates some invalid OM-PGD attack samples. Consequently, in our following experiments, we employed on-manifold adversarial examples with a 0.1 budget, under which invalid OM-FGSM and OM-PGD samples are seldom observed in CIFAR-10 and SVHN datasets.

Table 1: Adversary and defender setups in compared Work

Method	Attack Surfaces	Attack Algorithm	Augmentation
PGD-AT[36]	Off-manifold	Known	Input Space
PGD-DMAT[35]	Off/On-manifold	Known	Input/Latent Space
InputMixup[56]	Off-manifold	Unknown	Input Space
CutMix[54]	Off-manifold	Unknown	Input Space
PuzzleMixup[29]	Off-manifold	Unknown	Input Space
ManifoldMixup[52]	Off-manifold	Unknown	Latent Space
PatchUp[14]	Off-manifold	Unknown	Latent Space
LarepMixup(Ours)	Off/On-manifold	Unknown	Latent Space

Table 2: Accuracy (%) of CIFAR-10 classification models on off/on-manifold adversarial examples

PreActResNet18

Method	Clean	FGSM	PGD	AutoAttack	DeepFool	CW	OM-FGSM	OM-PGD	Known Attacker	Modify Network
Vanilla	**87.37±0.00**	32.07±0.00	28.93±0.00	7.59±0.00	10.36±0.00	2.60±0.00	51.02±0.00	21.68±0.00		
InputMixup[56]	84.48±1.45	63.58±3.36	68.12±3.46	56.63±10.20	37.97±2.58	41.11±2.10	58.53±0.43	44.11±1.34	✗	✗
CutMix[54]	82.14±3.00	65.51±1.03	69.67±1.34	64.41±3.55	36.79±2.60	39.74±3.10	57.59±0.31	43.50±1.71	✗	✗
PuzzleMixup[29]	83.11±1.64	65.73±2.46	70.35±2.60	64.03±6.06	38.86±1.53	41.83±1.74	57.80±0.77	43.68±2.19	✗	✗
ManifoldMixup[52]	71.10±4.17	49.26±1.34	52.49±1.91	44.08±1.60	25.33±2.76	27.19±2.53	50.16±1.66	38.64±0.80	✗	✓
PatchUp[14]	72.02±4.10	51.35±2.13	55.91±2.29	44.61±2.56	28.81±3.35	30.94±3.13	52.22±2.32	41.33±1.24	✗	✓
Ours-Convex	84.02±1.77	**68.86±2.88**	**72.65±3.59**	**66.98±5.93**	39.03±2.16	42.03±2.31	**60.02±0.91**	**46.72±1.52**	✗	✗
Ours-Mask	84.60±1.27	66.56±1.50	71.22±1.93	63.69±4.61	**39.27±2.97**	**42.54±2.74**	58.36±0.60	44.80±0.73	✗	✗

PreActResNet34

Method	Clean	FGSM	PGD	AutoAttack	DeepFool	CW	OM-FGSM	OM-PGD	Known Attacker	Modify Network
Vanilla	**83.57±0.00**	31.37±0.00	25.71±0.00	5.27±0.00	12.27±0.00	1.89±0.00	49.23±0.00	17.05±0.00		
InputMixup[56]	68.42±7.38	62.19±4.22	63.84±4.98	63.79±4.99	26.36±4.07	29.77±4.16	54.68±3.84	47.18±2.29	✗	✗
CutMix[54]	71.21±6.16	62.45±2.71	64.61±3.50	64.30±3.16	28.88±2.07	32.12±2.38	55.65±2.56	46.40±0.99	✗	✗
PuzzleMixup[29]	67.06±7.62	60.89±4.99	62.55±5.76	62.66±5.84	25.89±2.98	28.96±3.37	54.04±3.87	46.31±2.05	✗	✗
ManifoldMixup[52]	73.69±1.78	49.65±1.94	52.24±2.08	43.75±2.04	31.09±3.13	32.81±3.18	52.99±0.24	39.47±1.34	✗	✓
PatchUp[14]	72.71±2.96	49.53±1.44	52.76±2.80	42.31±1.80	32.35±3.66	34.10±3.45	53.03±2.37	39.38±1.63	✗	✓
Ours-Convex	78.44±1.60	**67.81±1.04**	**71.12±1.08**	**70.60±1.30**	**33.98±1.04**	**37.42±1.03**	**58.96±0.67**	**47.99±1.16**	✗	✗
Ours-Mask	77.13±3.17	66.16±1.58	68.90±1.62	68.40±2.16	32.95±2.26	36.38±2.23	58.31±0.96	47.30±1.06	✗	✗

Table 3: Accuracy (%) of SVHN classification models on off/on-manifold adversarial examples

PreActResNet18

Method	Clean	FGSM	PGD	AutoAttack	DeepFool	CW	OM-FGSM	OM-PGD	Known Attacker	Modify Network
Vanilla	**95.97±0.00**	57.29±0.00	34.57±0.00	29.21±0.00	22.51±0.00	21.54±0.00	41.04±0.00	6.78±0.00		
InputMixup[56]	94.39±0.79	68.77±2.03	58.81±2.34	51.25±2.22	**60.50±3.33**	64.42±2.16	44.58±0.86	18.48±1.04	✗	✗
CutMix[54]	94.19±1.07	68.78±2.01	59.52±3.28	52.50±3.64	57.45±3.26	63.62±1.52	44.31±1.02	17.87±0.91	✗	✗
PuzzleMixup[29]	94.54±0.66	67.55±1.79	58.79±3.34	51.65±3.48	55.87±2.22	63.42±1.51	43.63±0.62	16.00±1.15	✗	✗
ManifoldMixup[52]	89.15±4.22	67.21±1.85	60.32±1.94	53.60±3.21	52.95±3.15	60.57±1.97	43.32±1.52	**22.19±2.01**	✗	✓
PatchUp[14]	89.87±1.78	66.44±0.78	58.96±1.90	52.36±2.82	54.68±2.69	61.54±1.68	43.40±0.91	21.51±1.05	✗	✓
Ours-Convex	94.38±0.61	**70.62±1.35**	**63.35±0.67**	**56.66±1.22**	58.14±0.75	**64.45±0.54**	45.24±0.44	19.59±0.57	✗	✗
Ours-Mask	94.42±0.93	70.22±1.30	60.02±1.72	53.34±2.02	57.98±2.44	64.36±1.08	**45.26±0.54**	19.90±0.71	✗	✗

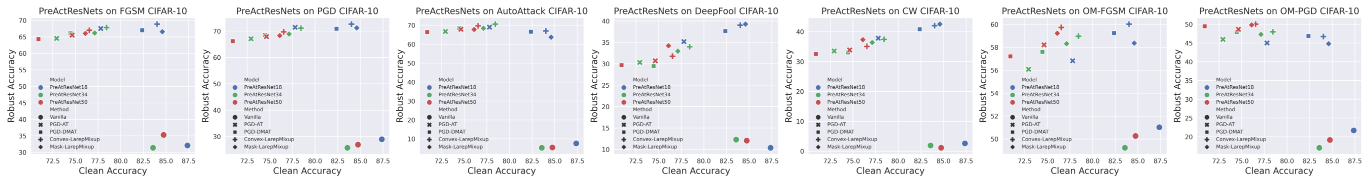
PreActResNet34

Method	Clean	FGSM	PGD	AutoAttack	DeepFool	CW	OM-FGSM	OM-PGD	Known Attacker	Modify Network
Vanilla	**95.75±0.00**	57.11±0.00	35.57±0.00	29.80±0.00	19.94±0.00	25.62±0.00	36.62±0.00	5.01±0.00		
InputMixup[56]	93.41±1.85	66.14±0.85	60.42±6.52	52.82±7.44	49.76±3.32	62.47±1.10	39.97±0.97	17.07±0.85	✗	✗
CutMix[54]	93.36±2.74	65.71±0.56	60.09±7.25	53.39±8.66	49.26±2.00	61.83±1.35	39.81±1.09	16.25±0.88	✗	✗
PuzzleMixup[29]	92.53±4.79	65.12±0.82	61.06±7.05	54.17±8.54	48.65±3.22	61.63±2.37	39.24±1.89	15.89±2.15	✗	✗
ManifoldMixup[52]	81.27±2.68	61.63±2.07	**63.61±3.10**	**59.19±1.94**	44.88±4.40	56.29±3.92	36.11±1.07	21.68±1.26	✗	✓
PatchUp[14]	68.39±9.86	51.94±4.91	55.01±6.31	52.17±5.91	36.07±2.41	47.47±5.47	31.81±2.20	**22.19±2.72**	✗	✓
Ours-Convex	94.94±0.31	**68.37±0.76**	61.75±3.65	53.55±4.05	**52.21±1.67**	**64.61±1.27**	**41.13±0.41**	16.88±0.38	✗	✗
Ours-Mask	93.63±1.13	67.69±0.52	63.21±5.39	55.74±5.69	52.10±2.75	64.27±1.30	40.70±0.60	17.01±0.47	✗	✗

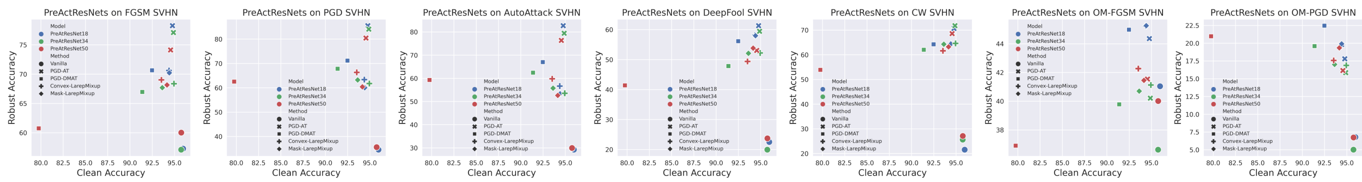
6.4 Comparison with Mixup Training Methods

We tested the performance of our proposed method in improving general robustness compared to state-of-the-art mixup training methods. For each attack, we ran each robust training method six times with the same settings and averaged the results. The initial learning rate, epochs, and batch size were 0.01, 256, and 40. All mixup training methods based on beta distribution sampling had parameters (1.0, 1.0) of the beta distribution. Experiments were conducted on PreActResNet models, with CIFAR-10 results in Table 2 and SVHN results in Table 3. The experimental results on PreActResNet-50 are shown in Appendix B. We bolded the best prediction accuracy and underlined the runner-up for each column.

To evaluate off-manifold adversarial robustness, we conduct defense tests against five out-of-manifold adversarial attacks, including FGSM, PGD, AutoAttack, DeepFool, and CW. Among these, FGSM and DeepFool are single-step attacks, PGD is a multi-step attack, and AutoAttack is an enhanced version of PGD attacks, which has been detailed in related work. For CIFAR-10, budgets of DeepFool, FGSM, PGD, OM-FGSM, and OM-PGD are 0.02, 0.05, 0.05, 0.05, 0.05, respectively. For SVHN, all budgets are set to 0.1. Refer to Table 5 in Appendix for more details on parameters, such as norm type, step size, number of iterations, and confidence. As seen in Table 2, our method achieves excellent defense results in most cases on the CIFAR-10 dataset. Both Convex-LarepMixup and ManifoldMixup use the linear interpolation strategy, while Mask-LarepMixup and PatchUp employ a binary mask mixing strategy. On the SVHN dataset, we observe from Table 3 that LarepMixup and Manifold Mixup have their own areas of expertise. Manifold Mixup is competitive in PGD-related attacks, while our algorithm has a stable advantage in FGSM, DeepFool, CW, and OM-FGSM.



(a) CIFAR-10



(b) SVHN

Figure 7: Accuracy (%) of robust trained PreActResNets on various attacks.

6.5 Comparison with Adversarial Training Methods

To further compare the difference between the improved robustness based on the proposed method and the improved robustness based on adversarial training, we compared LarepMixup with two powerful adversarial training methods, PGD-AT [36] and PGD-DMAT [35], on the CIFAR-10 and SVHN datasets. In PGD-AT, the defender generates the same number of white-box PGD examples as the original training samples for training. In PGD-DMAT, the defender generates PGD and OM-PGD adversarial examples each with half the number of original training samples for training. Attack budgets of adversarial example used for adversarial training are all set to 0.05. We used three kinds of the PreActeResNet models.

From Fig.7 (a) we can see that for most of the adversarial attacks on CIFAR-10, whether it is convex mixing or binary mask mixing, LarepMixup has achieved a slightly higher robustness improvement that AT. And it is also worth noting that our method has higher accuracy on clean samples, which is very close to the original clean accuracy. Fig.7 (b) shows taht our method still maintains good clean accuracy in SVHN, especially compared to the DMAT work. For off-manifold attacks, robustness from PGD-AT is greater, but when faced with adversarial attacks on the manifold, LarepMixup regains its advantage. Overall, LapreMixup achieves comparable robust performance to adversarial training without actively generating adversarial examples for training.

6.6 Evaluations on Perceptual Attack Examples

As adversarial attack methods evolve, it's vital to test the robustness of a model against unknown types of potential attacks. Perceptual attacks have been identified as a means to evaluate model robustness against new or unseen attacks [25, 35]. These attacks primarily use global color shifts and image filtering on normal images to create perturbed images. We consider four perceptual attacks: Fog, Snow, Elastic, and JPEG. For each perceptual attack, we conduct LarepMixup training thrice with the same settings and average the results. The initial learning rate, epochs number, batch size, and beta distribution parameters are 0.01, 40, 256, (1.0, 1.0), respectively.

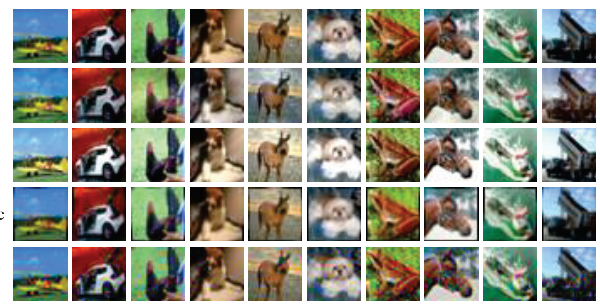
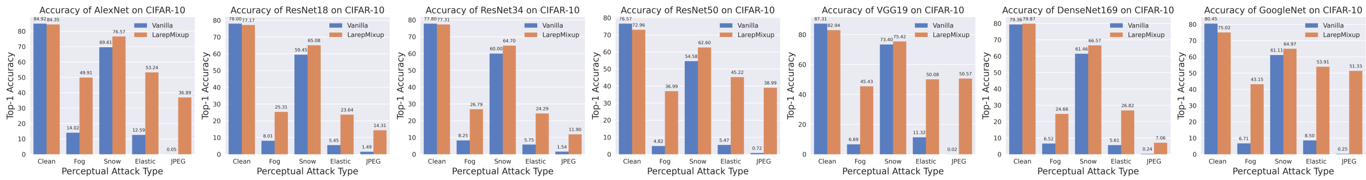


Figure 8: Percaptual attack examples.

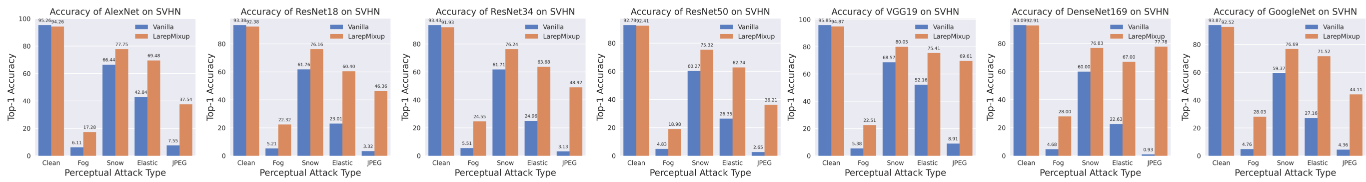
We conduct experiments on seven models using the CIFAR-10 and SVHN datasets. Taking the AlexNet as an example, the perception of four types of perceptual attack examples on CIFAR-10 are shown in Fig.8. According to Fig.9 (a), the accuracy of the classifiers on CIFAR-10 perceptual attacks has been greatly improved with LarepMixup, with the average accuracy of seven classifiers on Fog, Snow, Elastic, and JPEG samples increased by 28.17%, 5.19%, 31.79%, and 29.53%, respectively. At the same time, the accuracy of the seven classifiers on the clean test set dropped slightly, with an average reduction of 2.11%. Additionally, Fig.9 (b) shows the improvement of the robustness of the classifiers on SVHN perceptual attacks, with the average accuracy of seven classifiers on Fog, Snow, Elastic, and JPEG samples increased by 17.89%, 14.42%, 35.87%, and 47.10%, respectively. Since natural samples are constructed by superimposing perturbations on feature vectors in input space, it is reasonable to regard them as unseen attack samples outside the manifold. It can be seen that the model trained by LarepMixup achieves generalized robustness to unseen off-manifold attacks.

6.7 Evaluations on Different Mixing Modes

To evaluate the efficacy of LarepMixup in improving adversarial robustness across different mixing modes, we alternate between dual/ternary convex mixing and dual/ternary mask mixing. Furthermore, we conduct experiments using the ImageNet-Mixed10 dataset to verify the proposed method's applicability to high-dimensional

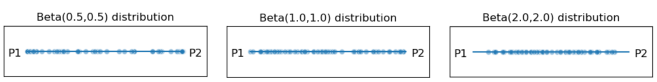


(a) CIFAR-10

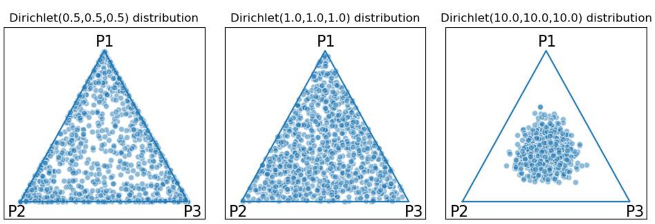


(b) SVHN

Figure 9: Accuracy (%) of various LarepMixup trained models on perceptual attacks.



(a) Two source data points P_1, P_2 and 60 mixed data points.



(b) Three source data points P_1, P_2, P_3 and 4000 mixed data points

Figure 10: Effect of sampling distribution on the position of interpolation points.

datasets. For each adversarial attack, we conduct proposed training three times under the same settings and take the average as the final result. The initial learning rate, epoch number, and batch size are 0.01, 40, and 32, respectively. The adversarial perturbation budget is 0.02. In our work, the parameters β of the Beta(β) distribution and γ of the Dirichlet(γ) distribution are hyperparameters, set by default to (1.0, 1.0) and (1.0, 1.0, 1.0), respectively. The positional relationship between source data points and mixed data points constructed using different coefficients is illustrated in Fig.10.

Experimental results evaluated in different modes are shown in Table 4. For off-manifold adversarial attacks, the robustness improvement from LarepMixup is not much different in convex mixing and mask mixing. But for on-manifold adversarial attacks, the advantages of convex mixing are obvious. In terms of the number of mixed source samples, $\{Dual, Ternary\}$, there is little difference in accuracy improvement between them. In general, for FGSM, PGD, AutoAttack, DeepFool, CW, OM-FGSM, and OM-PGD attacks, the accuracy rate of the four mixing modes increased by 2.90%, 3.15%, 3.67%, 75.67%, 83.71%, 16.62%, 22.27%, respectively.

Table 4: Robust accuracy (%) of PreActResNet18 under different mixing modes (ImageNet-Mixed10)

Method	Vanilla	Dual-LarepMixup		Ternary-LarepMixup	
		Convex	Mask	Convex	Mask
Clean	90.47	90.57±0.55	**90.89±0.35**	90.67±0.21	90.24±1.25
FGSM	13.93	17.09±0.29	16.21±0.14	16.71±0.34	**17.29±0.94**
PGD	2.00	5.38±0.81	4.68±0.45	4.73±0.69	**5.81±1.32**
AutoAttack	0.00	**3.74±0.19**	3.68±0.29	3.60±0.18	3.66±0.04
DeepFool	8.87	**85.38±0.19**	83.98±0.42	84.89±0.18	83.93±1.00
CW	0.10	**84.61±0.30**	83.16±0.52	84.19±0.47	83.28±0.62
OM-FGSM	26.90	**59.91±1.30**	28.61±5.58	57.36±1.89	28.21±0.98
OM-PGD	20.43	**58.76±1.30**	27.99±5.92	56.59±1.87	27.47±1.44

7 CONCLUSION

In this paper, we investigate the off/on-manifold robustness of DNNs. The main idea of our work is to mix latent representations lying on the low-dimensional manifold of the training set to synthesize mixed samples that capture latent variation factors in the dataset, and use them as augmented examples to train a model that can stably recognize data points adjacent to the decision boundary. Extensive evaluations show that even without any adversary information, our method can significantly alleviate the sensitivity of the model to multiple attacks in the input space and latent space. This paper concentrates on image classification, deferring other applications, such as employing LarepMixup for robust text classification with BERT-based text representations, to potential future work.

ACKNOWLEDGMENTS

This work was supported by the National Nature Science Foundation of China under Grant 61960206014 and Grant 62032012.

REFERENCES

[1] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. 2020. Square Attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision (ECCV 2020)*. 486–501.

[2] Shumeet Baluja and Ian Fischer. 2017. Adversarial Transformation Networks: Learning to Generate Adversarial Examples. *arXiv preprint arXiv:1703.09387* (2017).

[3] Christopher Beckham, Sina Honari, Vikas Verma, Alex Lamb, Farnoosh Ghadiri, R Devon Hjelm, Yoshua Bengio, and Christopher Pal. 2019. On Adversarial Mixup

Resynthesis. In *33rd Annual Conference on Neural Information Processing Systems (NIPS 2019)*.
[4] David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. 2019. Understanding and Improving Interpolation in Autoencoders via an Adversarial Regularizer. In *7th International Conference on Learning Representations (ICLR 2019)*.
[5] Nicholas Carlini and David Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *38th IEEE Symposium on Security and Privacy (SP 2017)*.
[6] Taylan Cemgil, Sumedh Ghaisas, Krishnamurthy Dj Dvijotham, and Pushmeet Kohli. 2020. Adversarially Robust Representations with Smooth Encoders. In *8th International Conference on Learning Representations (ICLR 2020)*.
[7] Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. MixText: Linguistically-Informed Interpolation of Hidden Space for Semi-Supervised Text Classification. In *58th Annual Meeting of the Association for Computational Linguistics*. 2147–2157.
[8] Uri Cohen, SueYeon Chung, Daniel D Lee, and Haim Sompolinsky. 2020. Separability and geometry of object manifolds in deep neural networks. *Nature Communications* 11, 1 (2020), 1–13.
[9] Francesco Croce and Matthias Hein. 2020. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *37th International Conference on Machine Learning (ICML 2020)*. 2196–2205.
[10] Francesco Croce and Matthias Hein. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *37th International Conference on Machine Learning (ICML 2020)*. 2206–2216.
[11] Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. 2019. AdverTorch v0. 1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623* (2019).
[12] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. 2018. Boosting Adversarial Attacks With Momentum. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2018)*. 9185–9193.
[13] Qingkai Fang, Rong Ye, Lei Li, Yang Feng, and Mingxuan Wang. 2022. STEMM: Self-learning with Speech-text Manifold Mixup for Speech Translation. In *Annual Meeting of the Association for Computational Linguistics*. 7050–7062.
[14] Mojtaba Faramarzi, Mohammad Amini, Akilesh Badrinaaraayanan, Vikas Verma, and Sarath Chandar. 2022. PatchUp: A Feature-Space Block-Level Regularization Technique for Convolutional Neural Networks. *AAAI Conference on Artificial Intelligence (AAAI 2022)* (2022).
[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.
[16] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *3rd International Conference on Learning Representations (ICLR 2015)*.
[17] Sadaf Gulshad, Jan Hendrik Metzen, and Arnold Smeulders. 2020. Adversarial and Natural Perturbations for General Robustness. *arXiv preprint arXiv:2010.01401* (2020).
[18] Hongyu Guo. 2020. Nonlinear mixup: Out-of-manifold data augmentation for text classification. In *AAAI Conference on Artificial Intelligence (AAAI 2020)*.
[19] Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019. MixUp as Locally Linear Out-of-Manifold Regularization. In *33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*. 3714–3722.
[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2016)*. 770–778.
[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV 2016)*. Springer, 630–645.
[22] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the Dimensionality of Data with Neural Networks. *science* 313, 5786 (2006), 504–507.
[23] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2017)*. 4700–4708.
[24] Ajil Jalal, Andrew Ilyas, Constantinos Daskalakis, and Alexandros G Dimakis. 2017. The Robust Manifold Defense: Adversarial Training using Generative Models. *arXiv preprint arXiv:1712.09196* (2017).
[25] Daniel Kang, Yi Sun, Dan Hendrycks, Tom Brown, and Jacob Steinhardt. 2019. Testing robustness against unforeseen adversaries. *arXiv preprint arXiv:1908.08016* (2019).
[26] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. 2020. Training Generative Adversarial Networks with Limited Data. In *34th Annual Conference on Neural Information Processing Systems (NIPS 2020)*.
[27] Tero Karras, Samuli Laine, and Timo Aila. 2019. A Style-Based Generator Architecture for Generative Adversarial Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2019)*. 4401–4410.
[28] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. Analyzing and Improving the Image Quality of StyleGAN. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2020)*. 8107–8116.
[29] Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. 2020. Puzzle Mix: Exploiting Saliency and Local Statistics for Optimal Mixup. In *37th International Conference on Machine Learning (ICML 2020)*. 5275–5285.
[30] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning Multiple Layers of Features from Tiny Images. (2009).
[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2017. Imagenet Classification with Deep Convolutional Neural Networks. *Commun. ACM* (2017).
[32] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial Machine Learning at Scale. In *5th International Conference on Learning Representations (ICLR 2017)*.
[33] Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. 2017. Adversarial examples in the physical world. In *5th International Conference on Learning Representations (ICLR 2017)*.
[34] Saehyung Lee, Hyungyu Lee, and Sungroh Yoon. 2020. Adversarial Vertex Mixup: Toward Better Adversarially Robust Generalization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2020)*. 272–281.
[35] Wei-An Lin, Chun Pong Lau, Alexander Levine, Rama Chellappa, and Soheil Feizi. 2020. Dual Manifold Adversarial Robustness: Defense against Lp and non-Lp Adversarial Attacks. In *34th Annual Conference on Neural Information Processing Systems (NIPS 2020)*.
[36] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *6th International Conference on Learning Representations (ICLR 2018)*.
[37] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal Adversarial Perturbations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2017)*. 1765–1773.
[38] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2016)*. 2574–2582.
[39] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. 2011. Reading Digits in Natural Images with Unsupervised Feature Learning. (2011).
[40] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, et al. 2018. Adversarial Robustness Toolbox v1. 0.0. *arXiv preprint arXiv:1807.01069* (2018).
[41] Tianyu Pang, Kun Xu, and Jun Zhu. 2020. Mixup Inference: Better Exploiting Mixup to Defend Adversarial Attacks. In *8th International Conference on Learning Representations (ICLR 2020)*.
[42] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The Limitations of Deep Learning in Adversarial Settings. In *IEEE European symposium on security and privacy (EuroS&P 2016)*.
[43] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An Imperative Style, High-performance Deep Learning Library. *33rd Annual Conference on Neural Information Processing Systems (NIPS 2019)* 32 (2019).
[44] H Sebastian Seung and Daniel D Lee. 2000. The manifold ways of perception. *science* 290, 5500 (2000), 2268–2269.
[45] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
[46] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. 2018. Constructing Unrestricted Adversarial Examples with Generative Models. In *32nd Annual Conference on Neural Information Processing Systems (NIPS 2018)*. 8322–8333.
[47] David Stutz, Matthias Hein, and Bernt Schiele. 2019. Disentangling Adversarial Robustness and Generalization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2019)*. 6976–6987.
[48] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One Pixel Attack for Fooling Deep Neural Networks. *IEEE Transactions on Evolutionary Computation* 23, 5 (2019), 828–841.
[49] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2015)*. 1–9.
[50] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations (ICLR 2014)*.
[51] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. 2020. Measuring robustness to natural distribution shifts in image classification. *arXiv preprint arXiv:2007.00644* (2020).
[52] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold Mixup: Better Representations by Interpolating Hidden States. In *36th International Conference on Machine Learning (ICML 2019)*. 6438–6447.
[53] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille. 2019. Improving Transferability of Adversarial Examples With Input Diversity. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2019)*. 2730–2739.
[54] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features. In *IEEE/CVF International Conference on Computer*

Vision (ICCV 2019). 6023–6032.
[55] Sergey Zagoruyko and Nikos Komodakis. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146* (2016).
[56] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2018. mixup: Beyond Empirical Risk Minimization. In *6th International Conference on Learning Representations (ICLR 2018)*.
[57] Jiahao Zhao, Penghui Wei, and Wenji Mao. 2021. Robust Neural Text Classification and Entailment via Mixup Regularized Adversarial Training. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1778–1782.

A EXPERIMENT SETUP DETAILS

A.1 Parameters in Attacks

A.1.1 Parameters in Adversarial Attacks. We use two categories of adversarial attack methods: off-manifold and on-manifold, detailed in Table 5. We normalize input sample ranges across datasets to the $[-1, 1]$ interval, which is passed as a clip parameter to attack interfaces. Table 5 presents parameters for experiments in this work that don't focus on evaluating the impact of perturbation strength. ϵ and η represent norm bounds for off-manifold and on-manifold perturbations, respectively, in p-norm bounded attacks. ϵ_s and η_s indicate single-step upper bounds for ϵ and η. n_i refers to the maximum iteration rounds. We use the default confidence of 0. For the CW attack based on optimization, there is no configuration parameter about the perturbation threshold, but the confidence parameter k needs to be configured. All CW attacks used in this work adopt the default confidence of 0. The reason for choosing these attack parameters in the evaluation experiments is that the perceptibility of adversarial perturbations under these attack configurations can be relatively well balanced with the attack success rate of the vanilla model.

A.1.2 Parameters in Perceptual Attacks. The perceptual attack methods we use can be divided into three categories: weather conditions (Fog, Snow), elastic transformation, and digital compression (JPEG), as shown in the Table 6. The reason for using perceptual attacks is to evaluate the generalization ability of the robust model to unseen attacks, which is adopted in [17, 25, 35, 51]. Our parameter configuration mainly refers to the perceptual attack parameters used in the DMAT [35]. All these attacks use 200 Gradient Descent iterations.

A.2 Parameters in Defense

A.2.1 Parameters in Standard Training. The models we employ primarily take two forms: one originates from the Torchvision library without any structural modifications, which includes AlexNet, ResNet18/34/50, DenseNet169, VGG19, and GoogleNet; the other is independently implemented, including PreActResNet18/34/50. Furthermore, preprocessing across all datasets is consistent. The input range of samples for all datasets is normalized to the $[-1, 1]$ interval using a normalization function with mean and standard deviation of 0.5. During standard training, the initial learning rate, 0.01, is reduced to one-tenth of the original every 10 epochs.

A.2.2 Parameters in Mixup Training. We evaluate mixup training methods, including input-space mixup (InputMixup, CutMix, PuzzleMixup) that directly mix input samples, and latent-space mixup (ManifoldMixup, PatchUp, LarepMixup) that mix latent samples. The maximum number of epochs is set to 40 for all mixup training methods. The initial learning rate is 0.01, reduced by a factor of

Table 5: Parameters in adversarial attacks

Dataset	Perturbation Space	Name	Norm	Configuration ϵ (η)	ϵ_s (η_s)	n_i	k
CIFAR-10	Off-Manifold	FGSM	L_∞	0.05	-	-	-
		PGD	L_∞	0.05	0.1	100	-
		AutoAttack	L_∞	0.05	0.1	-	-
		DeepFool	L_2	0.02	-	100	-
		CW	L_2	-	-	-	0
	On-Manifold	OM-FGSM	L_∞	0.05	-	-	-
		OM-PGD	L_∞	0.05	0.01	40	-
SVHN	Off-Manifold	FGSM	L_∞	0.1	0.1	-	-
		PGD	L_∞	0.1	0.1	100	-
		AutoAttack	L_∞	0.1	0.1	-	-
		DeepFool	L_2	0.1	-	100	-
		CW	L_2	-	-	-	0
	On-Manifold	OM-FGSM	L_∞	0.1	-	-	-
		OM-PGD	L_∞	0.1	0.01	40	-
ImageNet-Mixed10	Off-Manifold	FGSM	L_∞	0.02	0.1	-	-
		PGD	L_∞	0.02	0.1	100	-
		AutoAttack	L_∞	0.02	0.1	-	-
		DeepFool	L_2	0.02	-	100	-
		CW	L_2	-	-	-	0
	On-Manifold	OM-FGSM	L_∞	0.02	-	-	-
		OM-PGD	L_∞	0.02	0.01	40	-

Table 6: Parameters in perceptual attacks

Dataset	Perturbation Space	Name	Norm	Configuration ϵ (in pixel)	ϵ_s	n_i
CIFAR-10	Off-Manifold	Fog	L_∞	eps=128	0.002	200
		Snow		eps=0.0625	0.002	200
		Elastic		eps=0.5	0.035	200
		JPEG		eps=32	2.25	200
SVHN	Off-Manifold	Fog	L_∞	eps=128	0.002	200
		Snow		eps=0.0625	0.002	200
		Elastic		eps=0.5	0.035	200
		JPEG		eps=32	2.25	200

10 every 10 epochs. All augmented datasets used for mixup training consisted of mixed examples and an equal number of clean training examples. The number of mixed examples is consistent with the length of the training set. Dual-convex mixing mode is adopted in all experiments except for experiments evaluating evaluating the effect of mixing modes on the robustness performance. For InputMixup, ManifoldMixup, PuzzleMixup, and CutMix, the sampling distribution is set to Beta(1.0,1.0). For PatchUp, the sampling distribution is set to bernoulli distribution. The parameters of LarepMixup training are shown in Table 7. Since we choose to use a 512-dimensional vector to describe the latent representation, 512 Bernoulli trials are performed to determine whether the mask value of each dimension of the latent representation is 1 or 0 for binary mask mixing mode. The probability that the mask value of each dimension takes a value of 1 in each Bernoulli trial follows a uniform distribution. It is worth mentioning that the Bernoulli3$(512, p)$ distribution used for ternary mask mixing refers to conducting Bernoulli$(512, p)$ sampling of three source samples successively.

A.2.3 Parameters in Adversarial Training. In adversarial training that is also based on data augmentation, we ensure fairness in performance comparison by setting the same number of epochs, batch size, learning rate, and augmented examples as mixup training. The augmented examples used in adversarial training consist of adversarial examples and an equal number of clean training examples.

Table 7: Parameters in LarepMixup training

Dataset	Defense					
	Epochs	BatchSize	Initial Lr	Mixed Examples	LarepMixup Mode	Sampling Distribution
CIFAR-10	40	256	0.01	50,000	Dual Convex Dual Mask	Beta(1.0, 1.0) Bernoulli(512, p), p~U(0,1)
SVHN	40	256	0.01	73,257	Dual Convex Dual Mask	Beta(1.0, 1.0) Bernoulli(512, p), p~U(0,1)
ImageNet-Mixed10	40	32	0.01	77,237	Dual Convex Dual Mask Ternary Convex Ternary Mask	Beta(1.0, 1.0) Bernoulli(512, p), p~U(0,1) Dirichlet(1.0, 1.0, 1.0) Bernoulli3(512, p), p~U(0,1)

Table 8: Comparison with mixup training methods on PreActResNet50

CIFAR-10

Method	Clean	FGSM	PGD	AutoAttack	DeepFool	CW	OM-FGSM	OM-PGD	Known Attacker	Modify Network
Vanilla	**84.74±0.00**	35.27±0.00	26.89±0.00	5.43±0.00	12.03±0.00	1.13±0.00	50.26±0.00	19.13±0.00		
InputMixup[56]	74.93±2.22	65.28±1.42	67.42±1.67	67.57±2.04	31.35±2.59	34.39±2.40	58.92±1.28	49.81±1.23	✗	✗
CutMix[54]	75.27±3.01	64.68±2.18	66.98±2.46	66.97±2.30	30.70±2.77	33.84±2.78	58.40±1.00	48.61±1.24	✗	✗
PuzzleMixup[29]	67.35±5.41	59.96±2.92	61.18±3.36	61.41±2.79	26.84±2.06	29.58±2.03	55.72±1.91	48.46±1.39	✗	✗
ManifoldMixup[52]	76.17±3.03	54.54±2.59	56.76±2.88	47.64±6.91	29.97±4.24	32.81±4.04	55.26±0.93	40.93±2.34	✗	✓
PatchUp[14]	74.26±2.86	54.32±1.67	56.16±1.73	46.87±6.63	28.96±2.96	31.40±2.82	55.63±1.17	42.30±2.87	✗	✓
Ours-Convex	76.54±2.08	**66.99±1.87**	**69.73±1.92**	**69.69±1.49**	31.75±2.06	35.06±1.91	**59.73±1.00**	**50.04±0.52**	✗	✗
Ours-Mask	76.10±4.38	66.04±1.89	68.29±2.20	67.76±2.28	**34.21±3.45**	**37.32±3.67**	59.22±1.04	49.82±1.10	✗	✗

SVHN

Method	Clean	FGSM	PGD	AutoAttack	DeepFool	CW	OM-FGSM	OM-PGD	Known Attacker	Modify Network
Vanilla	**95.76±0.00**	60.02±0.00	35.61±0.00	29.94±0.00	23.72±0.00	27.09±0.00	40.01±0.00	6.73±0.00		
InputMixup[56]	94.45±0.29	66.80±1.09	58.42±2.83	50.07±3.26	49.19±1.79	60.38±1.30	42.04±0.37	17.17±0.58	✗	✗
CutMix[54]	94.48±0.31	65.83±1.79	57.87±1.85	49.99±1.21	45.26±1.06	58.98±0.44	41.73±0.46	15.37±0.75	✗	✗
PuzzleMixup[29]	94.13±1.63	66.82±0.82	61.97±4.94	54.42±5.66	47.50±1.59	61.34±0.72	41.46±0.96	15.63±0.98	✗	✗
ManifoldMixup[52]	77.84±9.38	61.22±5.05	63.16±4.59	**60.21±3.74**	44.15±7.24	54.42±6.05	36.84±3.29	**22.97±1.79**	✗	✓
PatchUp[14]	78.36±1.65	58.62±3.41	60.46±3.63	57.70±3.59	43.14±3.84	54.24±2.05	36.25±1.74	21.95±1.75	✗	✓
Ours-Convex	93.53±1.96	**69.02±0.70**	**66.33±5.88**	59.78±7.54	49.39±2.03	61.59±1.07	**42.27±0.87**	17.57±1.23	✗	✗
Ours-Mask	94.16±1.73	68.15±0.46	60.39±2.93	52.62±3.15	**53.83±2.38**	**63.21±1.32**	41.45±3.86	19.34±3.09	✗	✗

Table 9: Comparison with adversarial training methods on CIFAR-10

PreActResNet18

Method	Clean	FGSM	PGD	AutoAttack	DeepFool	CW	OM-FGSM	OM-PGD	Known Attacker	Modify Network
Vanilla	**87.37±0.00**	32.07±0.00	28.93±0.00	7.59±0.00	10.36±0.00	2.60±0.00	51.02±0.00	21.68±0.00		
PGD-AT[36]	77.80±6.39	67.54±3.70	71.44±4.62	**68.98±4.38**	35.18±6.43	37.83±6.41	56.82±1.86	45.01±2.77	✓	✗
PGD-DMAT[35]	82.37±1.07	67.00±2.48	70.88±2.97	66.56±4.55	37.66±2.38	40.85±2.20	59.24±1.35	**46.89±1.48**	✓	✗
Ours-Convex	84.02±1.77	**68.86±2.88**	**72.65±3.59**	**66.98±5.93**	39.03±2.16	42.03±2.31	**60.02±0.91**	**46.72±1.52**	✗	✗
Ours-Mask	84.60±1.27	66.56±1.50	71.22±1.93	63.69±4.61	**39.27±2.97**	**42.54±2.74**	58.36±0.60	44.80±0.73	✗	✗

PreActResNet34

Method	Clean	FGSM	PGD	AutoAttack	DeepFool	CW	OM-FGSM	OM-PGD	Known Attacker	Modify Network
Vanilla	**83.57±0.00**	31.37±0.00	25.71±0.00	5.27±0.00	12.27±0.00	1.89±0.00	49.23±0.00	17.05±0.00		
PGD-AT[36]	72.93±5.95	64.54±2.67	67.08±3.72	66.72±3.46	30.32±3.74	33.53±3.87	56.08±2.50	45.97±0.49	✓	✗
PGD-DMAT[35]	74.46±1.88	66.11±1.03	68.54±0.88	68.58±0.60	29.44±2.53	33.03±2.40	57.61±0.79	47.96±0.62	✓	✗
Ours-Convex	78.44±1.60	**67.81±1.04**	**71.12±1.08**	**70.60±1.30**	**33.98±1.04**	**37.42±1.03**	**58.96±0.67**	**47.99±1.16**	✗	✗
Ours-Mask	77.13±3.17	66.16±1.58	68.90±1.62	68.40±2.16	32.95±2.26	36.38±2.23	58.31±0.96	47.30±1.06	✗	✗

PreActResNet50

Method	Clean	FGSM	PGD	AutoAttack	DeepFool	CW	OM-FGSM	OM-PGD	Known Attacker	Modify Network
Vanilla	**84.74±0.00**	35.27±0.00	26.89±0.00	5.43±0.00	12.03±0.00	1.13±0.00	50.26±0.00	19.13±0.00		
PGD-AT[36]	74.64±3.30	65.53±2.02	67.92±2.26	67.89±1.92	30.69±1.83	33.88±1.54	58.21±0.92	48.65±0.90	✓	✗
PGD-DMAT[35]	70.92±2.79	64.33±1.77	66.18±2.08	66.42±2.03	29.66±2.17	32.59±2.08	57.20±1.50	49.47±1.13	✓	✗
Ours-Convex	76.54±2.08	**66.99±1.87**	**69.73±1.92**	**69.69±1.49**	31.75±2.06	35.06±1.91	**59.73±1.00**	**50.04±0.52**	✗	✗
Ours-Mask	76.10±4.38	66.04±1.89	68.29±2.20	67.76±2.28	**34.21±3.45**	**37.32±3.67**	59.22±1.04	49.82±1.10	✗	✗

we generate on-manifold adversarial examples and off-manifold adversarial examples each with half the number of original training samples to ensure that the total number of augmented examples is consistent no matter in mixup training or adversarial training.

B ADDITION EXPERIMENTAL RESULTS

B.1 Evaluation on Higher Capacity Models

To observe the applicability of the model on higher capacity classification models, we trained a CIFAR-10 classification model with higher initial accuracy by changing the basic model structure and adjusting the training strategy. On this modified ResNet18 model

Table 10: Comparison with adversarial training methods on SVHN

PreActResNet18										
Method	Clean	FGSM	PGD	AutoAttack	DeepFool	CW	OM-FGSM	OM-PGD	Known Attacker	Modify Network
Vanilla	**95.97±0.00**	57.29±0.00	34.57±0.00	29.21±0.00	22.51±0.00	21.54±0.00	41.04±0.00	6.78±0.00		
PGD-AT[36]	94.77±0.32	**78.26±2.07**	**85.50±2.34**	**82.69±3.18**	**61.32±2.23**	**70.65±1.39**	44.36±0.40	17.80±1.07	✓	✗
PGD-DMAT[35]	92.48±0.68	70.65±2.78	71.16±4.92	66.99±5.53	56.15±2.72	64.23±1.74	44.98±1.23	**22.46±2.15**	✓	✗
Ours-Convex	94.38±0.61	**70.62±1.35**	**63.35±0.67**	**56.66±1.22**	58.14±0.75	**64.45±0.54**	45.24±0.44	19.59±0.57	✗	✗
Ours-Mask	94.42±0.93	70.22±1.30	60.02±1.72	53.34±2.02	57.98±2.44	64.36±1.08	**45.26±0.54**	19.90±0.71	✗	✗
PreActResNet34										
Method	Clean	FGSM	PGD	AutoAttack	DeepFool	CW	OM-FGSM	OM-PGD	Known Attacker	Modify Network
Vanilla	**95.75±0.00**	57.11±0.00	35.57±0.00	29.80±0.00	19.94±0.00	25.62±0.00	36.62±0.00	5.01±0.00		
PGD-AT[36]	94.88±0.19	**77.09±1.76**	**84.07±2.26**	**79.42±2.91**	**59.45±2.33**	**71.77±1.29**	40.21±0.32	15.85±0.63	✓	✗
PGD-DMAT[35]	91.38±1.67	66.96±0.85	67.83±2.71	62.43±3.57	47.85±2.40	62.01±1.56	39.78±0.76	19.59±0.85	✓	✗
Ours-Convex	94.94±0.31	**68.37±0.76**	61.75±3.65	53.55±4.05	**52.21±1.67**	**64.61±1.27**	**41.13±0.41**	16.88±0.38	✗	✗
Ours-Mask	93.63±1.13	67.69±0.52	63.21±5.39	55.74±5.69	52.10±2.75	64.27±1.30	40.70±0.60	17.01±0.47	✗	✗
PreActResNet50										
Method	Clean	FGSM	PGD	AutoAttack	DeepFool	CW	OM-FGSM	OM-PGD	Known Attacker	Modify Network
Vanilla	**95.76±0.00**	60.02±0.00	35.61±0.00	29.94±0.00	23.72±0.00	27.09±0.00	40.01±0.00	6.73±0.00		
PGD-AT[36]	94.56±0.61	**74.12±3.43**	**80.42±6.22**	**76.34±7.32**	53.06±2.98	**68.57±2.51**	41.54±0.43	16.15±0.35	✓	✗
PGD-DMAT[35]	79.76±12.46	60.75±6.39	62.53±4.83	59.28±3.56	41.42±6.48	53.89±8.10	36.90±4.29	20.97±3.31	✓	✗
Ours-Convex	93.53±1.96	**69.02±0.70**	**66.33±5.88**	59.78±7.54	49.39±2.03	61.59±1.07	**42.27±0.87**	17.57±1.23	✗	✗
Ours-Mask	94.16±1.73	68.15±0.46	60.39±2.93	52.62±3.15	**53.83±2.38**	**63.21±1.32**	41.45±3.86	19.34±3.09	✗	✗

with 96.41% clean accuracy on CIFAR-10, we further evaluate the performance of our robust training algorithm, LarepMixup, on PGD adversarial examples with different perturbation budgets. The experimental results are shown in the Table 11 and Table 12, which demonstrate that the accuracy of the initial vanilla model does not substantially affect the effectiveness of our algorithm. Even on the modified ResNet18 model with higher accuracy, an improvement of robust accuracy under different attack budgets is observed.

Table 11: Accuracy(%) of enhanced ResNet18 on white-box adversarial examples

Input	ϵ	ϵ_s	n_i	Vanilla	Ours	Improve
Clean	-	-	-	96.41	96.29	-0.12
PGD	0.031	0.0078	7	21.53	26.36	4.83
PGD	0.031	0.0078	10	15.58	20.23	4.65
PGD	0.031	0.0078	20	10.10	13.52	3.42
PGD	0.051	0.0078	20	2.62	4.95	2.33
PGD	0.1	0.0078	20	0.29	0.89	0.60
PGD	0.2	0.0078	20	0.11	0.48	0.37

Table 12: Accuracy(%) of enhanced ResNet18 on grey-box adversarial examples

Input	ϵ	ϵ_s	n_i	Vanilla	Ours	Improve
Clean	-	-	-	96.41	96.29	-0.12
PGD	0.031	0.0078	7	21.53	36.29	14.76
PGD	0.031	0.0078	10	15.58	31.05	15.47
PGD	0.031	0.0078	20	10.10	25.91	15.81
PGD	0.051	0.0078	20	2.62	9.54	6.92
PGD	0.1	0.0078	20	0.29	2.12	1.83
PGD	0.2	0.0078	20	0.11	1.24	1.13

B.2 Time Cost of Our Work

In our scheme, the training of the styleGAN model is separated from the training of the robust classifier. Once a styleGAN model has been trained well on a given dataset, thereafter it will only be used as a mapping function from low-dimensional representation to high-dimensional input to participating in the LarepMixup training of any target network. Taking the CIFAR-10 dataset as an example, we trained the StyleGAN model for 280 epochs on the CIFAR-10 training set, each epoch took about 218 seconds. We spent a total of 16.9 hours training a styleGAN model, realizing the final effect that a latent variable randomly sampled in the hidden space of styleGAN can be mapped to a sample with real semantics in the input space. Then, we constructed the on-manifold CIFAR-10 datasets consisting of latent representations z and corresponding labels y. From the perspective of training a robust network, the above process can be regarded as a preprocessing process. After constructing the latent representation dataset of CIFAR-10, we then use it for building various robust networks on CIFAR-10. Taking the WideResNet28-10 as an example, the time cost of LarepMixup training is shown in Fig.11. The time cost of LarepMixup training was almost 700 seconds per epoch. We trained the robust WideResNet28-10 model for 40 epochs, a total using almost 7.7 hours.

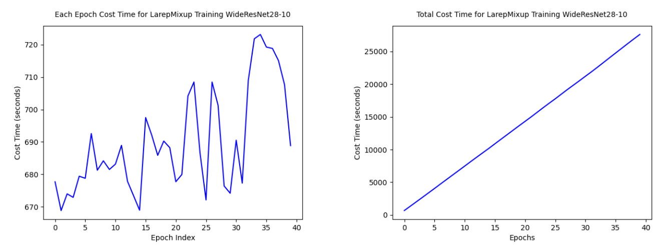


Figure 11: Time cost for LarepMixup training.