GenLabel: Mixup Relabeling using Generative Models

Jy-yong Sohn [1] **Liang Shang** [1] **Hongxu Chen** [1] **Jaekyun Moon** [2] **Dimitris Papailiopoulos** [1] **Kangwook Lee** [1]

Abstract

Mixup is a data augmentation method that generates new data points by mixing a pair of input data. While mixup generally improves the prediction performance, it sometimes degrades the performance. In this paper, we first identify the main causes of this phenomenon by theoretically and empirically analyzing the mixup algorithm. To resolve this, we propose *GenLabel*, a simple yet effective relabeling algorithm designed for mixup. In particular, *GenLabel* helps the mixup algorithm correctly label mixup samples by learning the class-conditional data distribution using generative models. Via theoretical and empirical analysis, we show that mixup, when used together with *GenLabel*, can effectively resolve the aforementioned phenomenon, improving the accuracy of mixup-trained model.

1. Introduction

Mixup (Zhang et al., 2018) is a widely adopted data augmentation algorithm used when training a classifier, which generates synthetic samples by linearly interpolating two randomly chosen samples. Each mixed sample is *soft*-labeled, i.e., it is labeled as a mixture of two (possibly same) classes of the chosen samples. The rationale behind the mixup algorithm is that such mixed samples can fill up the void space in between different class manifolds, effectively regularizing the model behavior. Mixup has been shown to improve generalization on multiple benchmark image datasets. Various variants of mixup have been proposed in the past few years such as Manifold-mixup (Verma et al., 2019), which apply mixup in the latent feature space, and several computer vision-specific variants (Yun et al., 2019; Kim et al., 2020; Uddin et al., 2021; Kim et al., 2021). Recent studies also provide some theoretical explanations for the success of mixup (Zhang et al., 2021; Carratino et al., 2020).

[1]Department of Electrical and Computer Engineering, University of Wisconsin, Madison, USA [2]School of Electrical Engineering, Daejeon, KAIST. Correspondence to: Kangwook Lee <kw1jjang@gmail.com>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

Mixup, however, does *not* always improve accuracy, and sometimes it even hurts. Guo et al. (2019) observed that the accuracies of mixup is up to 1.8% worse than vanilla training on some image classification tasks. Greenewald et al. (2021) reported a similar finding; the original mixup degrades the accuracy of vanilla training up to 2.5% on an UCI classification task (Dua & Graff, 2017). Despite such empirical findings, (1) why mixup fails and (2) how we can prevent them are not fully understood yet.

1.1. Main contributions

In this work, we first take a closer look at the failure scenarios of mixup, particularly focusing on the low-dimensional input setting. As a result, we identify two main reasons behind mixup's failure scenarios. The first one is *manifold intrusion*, which was defined in (Guo et al., 2019). Samples generated by mixing two classes may intrude the manifold of another class, causing label conflicts with the samples from the intruded class. We perform analysis of the effect of manifold intrusion. The second reason we identify is *linear labeling*. Mixup assigns a mixed sample with a linear combination of the two one-hot encoded labels. We prove that, focusing on a specific softmax regression setting, such linear labeling results in a strictly suboptimal margin and accuracy.

After identifying the key reasons behind mixup's failure, we propose a simple yet effective fix for mixup. Our proposed algorithm *GenLabel* is a relabeling algorithm designed for mixup. The idea is strikingly simple – *GenLabel* relabels a mixed sample using the likelihoods that are estimated with learned generative models.

Consider a three-way classification problem in Fig. 1. *GenLabel* learns generative models to estimate the likelihood of a mixed sample. Let $p_c(\boldsymbol{x})$ be the likelihood of sample $\boldsymbol{x}$ drawn from class c, and let $\widehat{p_c}(\boldsymbol{x})$ be the estimated likelihood. Given a mixed sample $\boldsymbol{x}^{\text{mix}}$, *GenLabel* first computes all three likelihoods $\widehat{p_c}(\boldsymbol{x}^{\text{mix}})$ for class $c \in \{1, 2, 3\}$, and then assigns the mixed sample the following label: $\boldsymbol{y}^{\text{gen}} = \text{softmax}(\log \widehat{p_1}(\boldsymbol{x}^{\text{mix}}), \log \widehat{p_2}(\boldsymbol{x}^{\text{mix}}), \log \widehat{p_3}(\boldsymbol{x}^{\text{mix}}))$. Note that the c^{th} entry of $\boldsymbol{y}^{\text{gen}}$ is $\frac{\widehat{p_c}(\boldsymbol{x}^{\text{mix}})}{\sum_{c'=1}^{3} \widehat{p_{c'}}(\boldsymbol{x}^{\text{mix}})}$, which equals the posterior probability estimate $\widehat{\mathbb{P}}(y = c|\boldsymbol{x}^{\text{mix}})$,

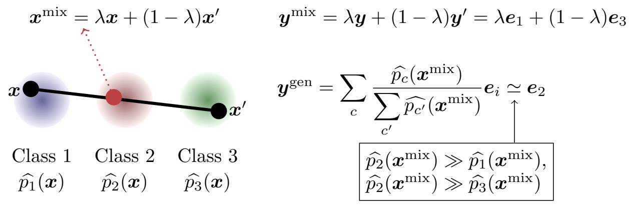

$\boldsymbol{x}^{\text{mix}} = \lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{x}'$
$\boldsymbol{y}^{\text{mix}} = \lambda \boldsymbol{y} + (1-\lambda)\boldsymbol{y}' = \lambda \boldsymbol{e}_1 + (1-\lambda)\boldsymbol{e}_3$
$\boldsymbol{x}$
$\boldsymbol{x}'$
$\boldsymbol{y}^{\text{gen}} = \sum_c \frac{\widehat{p_c}(\boldsymbol{x}^{\text{mix}})}{\sum_{c'} \widehat{p_{c'}}(\boldsymbol{x}^{\text{mix}})} \boldsymbol{e}_i \simeq \boldsymbol{e}_2$
$\widehat{p_2}(\boldsymbol{x}^{\text{mix}}) \gg \widehat{p_1}(\boldsymbol{x}^{\text{mix}})$, $\widehat{p_2}(\boldsymbol{x}^{\text{mix}}) \gg \widehat{p_3}(\boldsymbol{x}^{\text{mix}})$
Class 1 $\widehat{p_1}(\boldsymbol{x})$
Class 2 $\widehat{p_2}(\boldsymbol{x})$
Class 3 $\widehat{p_3}(\boldsymbol{x})$

Figure 1: Visualization of *GenLabel* applied to mixup. Consider two labeled samples $(\boldsymbol{x}, \boldsymbol{y})$ and $(\boldsymbol{x}', \boldsymbol{y}')$. Mixup generates $\boldsymbol{x}^{\text{mix}} = \lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{x}'$ for some $\lambda \in [0,1]$ and label it as $\boldsymbol{y}^{\text{mix}} = \lambda \boldsymbol{y} + (1-\lambda)\boldsymbol{y}' = \lambda \boldsymbol{e}_1 + (1-\lambda)\boldsymbol{e}_3$, where $\boldsymbol{e}_c$ is the c-th standard basis vector. This induces label noise as $\boldsymbol{x}^{\text{mix}}$ lies on the manifold of class 2. On the other hand, *GenLabel* properly relabel it as $\boldsymbol{y}^{\text{gen}} \simeq \boldsymbol{e}_2$, which is computed using the learned distribution for each class c ($\widehat{p_c}(\boldsymbol{x})$) and the relabeling formula in the figure.

when we have a balanced dataset. Thus, *GenLabel* can be viewed as a labeling method that assigns the posterior probability of the label y given a mixed sample $\boldsymbol{x}^{\text{mix}}$. This property of *GenLabel* allows us to fix the issue of the conventional labeling method in mixup, as shown in Fig. 1.

The suggested *GenLabel* has been analyzed in diverse perspectives. First, we empirically show that *GenLabel* fixes the manifold intrusion issue on toy datasets. Second, we provide a problem setup where *GenLabel* combined with mixup maximizes the margin of a classifier, while mixup alone leads to a much smaller margin. Third, we show that *GenLabel* improves the adversarial robustness of mixup in logistic regression models and fully-connected (FC) networks with ReLU activations. Finally, we tested *GenLabel* on 133 low-dimensional real datasets in OpenML (Vanschoren et al., 2013). Our experimental results show that the suggested *GenLabel* helps mixup improve the accuracy in various low-dimensional datasets. We also found that adversarial robustness can be improved by applying *GenLabel*.

1.2. Preliminaries

Notations We focus on k-way classification tasks. A dataset with n data points is denoted by $S = \{\boldsymbol{z}_i\}_{i=1}^n$, where the i-th data point is represented by $\boldsymbol{z}_i = (\boldsymbol{x}_i, \boldsymbol{y}_i)$ composed of the input feature $\boldsymbol{x}_i \in \mathbb{R}^d$ and the label $\boldsymbol{y}_i \in [0,1]^k$. We use one-hot encoding for the label, *i.e.*, the label of class-c data points is represented as $\boldsymbol{y} = \boldsymbol{e}_c$, where $\boldsymbol{e}_c$ is the standard basis vector with a 1 in the c-th coordinate and 0's elsewhere. Mixed samples can have *soft label*, *e.g.*, $0.5\boldsymbol{e}_1 + 0.5\boldsymbol{e}_2$ denotes that the mixed sample is equally likely to be from class 1 and 2. The set of input features is denoted by $X = \{\boldsymbol{x}_i\}_{i=1}^n$. We assume that each data point $\boldsymbol{x}$ in class c is generated from (unknown) probability distribution $p_c(\boldsymbol{x}) := p_{\mathsf{x}|\mathsf{y}}(\boldsymbol{x}|\boldsymbol{e}_c)$. For a positive integer k, we define $[k] := \{1, 2, \cdots, k\}$. Given a distance metric, $d(x_1, x_2)$ denotes the distance between x_1 and x_2, and $d(x, A) = \min_{a \in A} d(x, a)$ denotes the minimum of the distances between x and the points in a closed set A.

Mixup Mixup (Zhang et al., 2018) generates synthetic data points by applying a linear combination of two samples. Given samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, it generates $\boldsymbol{x}^{\text{mix}} = \lambda \boldsymbol{x}_i + (1-\lambda)\boldsymbol{x}_j$ having a mixed label $\boldsymbol{y}^{\text{mix}} = \lambda \boldsymbol{y}_i + (1-\lambda)\boldsymbol{y}_j$, for randomly sampled $\lambda \sim \text{Beta}(\alpha, \alpha)$ for a given $\alpha > 0$.

Gaussian mixture model Gaussian mixture (GM) model is a generative model, which assumes that samples $\boldsymbol{x}$ in each class (say class c) follow a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$ for some mean $\boldsymbol{\mu}_c$ and covariance matrix $\boldsymbol{\Sigma}_c$. One can estimate the model parameters by computing the within-class sample mean $\widehat{\boldsymbol{\mu}_c}$ and the within-class sample covariance matrix $\widehat{\boldsymbol{\Sigma}_c}$ for each class. A GM model of two classes, say class 1 and 2, can be modeled as $\pi_1 \mathcal{N}(\widehat{\boldsymbol{\mu}_1}, \widehat{\boldsymbol{\Sigma}_1}) + (1-\pi_1)\mathcal{N}(\widehat{\boldsymbol{\mu}_2}, \widehat{\boldsymbol{\Sigma}_2})$, where $\pi_1 = \mathbb{P}(\boldsymbol{y} = \boldsymbol{e}_1)$.

Kernel density estimator Kernel density estimator (KDE) is a non-parametric density estimator that makes use of a kernel function. For example, KDE with Gaussian kernel estimates the distribution of class c as $\frac{1}{n_c}\sum_{i=1}^{n_c} \mathcal{N}(\boldsymbol{x}_i, h^2 \widehat{\boldsymbol{\Sigma}_c})$ for a given bandwidth h, where $\{\boldsymbol{x}_i\}_{i=1}^{n_c}$ is the set of samples in class c and $\widehat{\boldsymbol{\Sigma}_c}$ is the sample covariance matrix of class c. One can use KDE as a generative model, creating new samples from the estimated density.

Nearest neighbor (NN) based density estimator Suppose we have n_c data points in class c. Then, k_c-NN density estimator provides the estimated class-conditional density $p_c(\boldsymbol{x}) = \frac{k_c}{n_c V}$ for a given sample $\boldsymbol{x}$, where V is the minimum volume of a sphere centered at $\boldsymbol{x}$ and containing k_c data points belonging to class c (Bishop, 2006).

2. Related Works

Mixup and variants Mixup and its variants have been considered as promising data augmentation schemes improving the generalization and robustness performance in various image classification tasks (Zhang et al., 2018; Verma et al., 2019; Tokozume et al., 2018; Inoue, 2018; Shimada et al., 2019; Hendrycks et al., 2020; Yun et al., 2019; Kim et al., 2020; Uddin et al., 2021; Kim et al., 2021; Zhang et al., 2021; Park et al., 2022; Liu et al., 2021). However, the performance of mixup for low-dimesional datasets have been rarely observed in previous works. This paper focuses on the failures of mixup in low-dimensional datasets, and provide a simple label correction method to solve this issue, which improves generalization performance in various real datasets.

Manifold intrusion Guo et al. (2019) observed that mixup samples of two classes may intrude the manifold of a third class. The authors dubbed this label conflict phenomenon as *manifold intrusion*. Hwang & Whang (2021) found that a similar label conflict problem becomes even more salient in the regression setting. To resolve manifold intrusion issue, previous works have suggested mixing

Table 1: Clean accuracy (%) on various synthetic and real datasets. Mixup performs worse than vanilla training on these datasets.

Dataset	Moon	Four-circle	3D cube	OpenML-48	OpenML-307	OpenML-818
Vanilla	**98.7**	**91.3**	**93.0**	**39.1**	**66.8**	**100.0**
Mixup	97.0 (1.7↓)	57.7 (33.6↓)	89.4 (3.6↓)	30.9 (8.2↓)	54.4 (12.4↓)	92.3 (7.7↓)

strategies which avoid generating mixup samples causing the label conflict. Guo et al. (2019) suggested learning a mixing policy network that prohibits generating the in-manifold mixup samples. Greenewald et al. (2021) suggested mixing adjacent data samples by using the concept of optimal transport. This allows both the mixed sample and the corresponding data sample pair lie on the same manifold, which avoids facing the label-conflicting scenarios. Focusing on the regression setting, Hwang & Whang (2021) suggested learning a mixing policy by measuring how helpful mixing each pair is. Although all these regularization techniques prohibit generating mixup points that incur label conflicts, they also inherently give up the potential benefits of using such label-conflicting mixup samples by properly re-labeling them. In this paper, for the first time, we solve the manifold intrusion issue by *re-labeling* the mixup samples, with the aid of generative models.

Generative models for classification Generative models have been widely used for classification tasks for several decades. A generative classifier (Ng & Jordan, 2002) predicts label $\boldsymbol{y}$ based on the class-conditional density $p(\boldsymbol{x}|\boldsymbol{y})$ estimated by generative models, and has been developed until recent years (Schott et al., 2018; Ju & Wagner, 2020). The present paper also makes use of generative models for classification task, but we use them for re-labeling augmented data, while existing works use them for the prediction itself. Some previous works proposed generative model-based data augmentation schemes (Antoniou et al., 2017; Perez & Wang, 2017; Tanaka & Aranha, 2019). While these schemes use the learned distribution to create on-manifold synthetic data, we use the learned distribution to re-label both on-manifold and out-of-manifold mixup samples. There have been extensive works on using generative models to improve the robustness against adversarial attacks and out-of-distribution samples (Ilyas et al., 2017; Xiao et al., 2018; Samangouei et al., 2018; Song et al., 2018; Schott et al., 2018; Li et al., 2019; Ghosh et al., 2019; Serrà et al., 2020; Choi et al., 2018; Lee et al., 2018). Though looking similar, these are not data augmentation algorithms and hence only tangentially related to our algorithm. Our method can be used together with any of these algorithms.

3. Failure of Mixup on Low-Dimensional Data

In this section, we observe the failure scenarios of mixup, *i.e.*, when mixup performs even worse than vanilla training, especially focusing on the low-dimensional data setting. Table 1 shows the scenarios when mixup has a lower accuracy

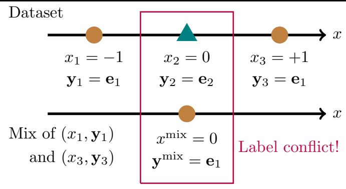

Dataset
x
$x_1 = -1$
$\mathbf{y}_1 = \mathbf{e}_1$
$x_2 = 0$
$\mathbf{y}_2 = \mathbf{e}_2$
$x_3 = +1$
$\mathbf{y}_3 = \mathbf{e}_1$
x
Mix of $(x_1, \mathbf{y}_1)$
and $(x_3, \mathbf{y}_3)$
$x^{\text{mix}} = 0$
$\mathbf{y}^{\text{mix}} = \mathbf{e}_1$
Label conflict!

Figure 2: Example 1 visualization. Top: dataset $S = \{x_i, \boldsymbol{y}_i\}_{i=1}^3$, Bottom: A mixup point generated with x_1 and x_3 with $\lambda = 0.5$. This incurs the manifold intrusion because of x_2 at the same location but with a different label. Our analysis shows that this phenomenon is what reduces the margin from $1/2$ to $7/16$.

than vanilla training, for synthetic datasets[1] and OpenML datasets (Vanschoren et al., 2013). For example, in the Four-circle dataset, the performance gap between mixup and vanilla training is larger than 30%. Why does mixup have such failure scenarios? Here we identify and analyze two main reasons. First, we show the manifold intrusion defined in (Guo et al., 2019) may degrade the margin/accuracy of mixup. Second, we show that even when there is no manifold intrusion issue, the labeling method used in mixup may harm the margin/accuracy of a classifier.

3.1. Manifold intrusion reduces margin and accuracy

In (Guo et al., 2019), the manifold intrusion (MI) is defined as the case when the mixup sample $\boldsymbol{x}^{\text{mix}}$, generated by mixing data in class c_1 and c_2, collides with a real data sample having label $c_3 \notin \{c_1, c_2\}$. Below we provide a concrete problem instance where the margin reduction property of manifold intrusion is rigorously proved.

Example 1. *Consider binary classification on the dataset* $S = \{(x_i, \boldsymbol{y}_i)\}_{i=1}^3 = \{(-1, \boldsymbol{e}_1), (0, \boldsymbol{e}_2), (+1, \boldsymbol{e}_1)\}$ *in Fig. 2, where each data point in class 1 is represented as brown circle, and each data point in class 2 is shown as blue triangle. We consider the classifier f_θ defined as $f_\theta(x) = 1$ if $|x| \geq 2\theta$, and $f_\theta(x) = \frac{|x|}{2\theta}$ otherwise. This classifier estimates the label of a given feature x as $\hat{\boldsymbol{y}} = [f_\theta(x), 1 - f_\theta(x)]$, and the margin of this classifier is represented as* $\text{margin}(f_\theta) = \min\{\theta, 1-\theta\}$.

As in Fig. 2, applying mixup on this dataset suffers from manifold intrusion (MI); mixing x_1 and x_3 with coefficient $\lambda = 0.5$ generates $x^{\text{mix}} = 0$ with label $\boldsymbol{y}^{\text{mix}} = \boldsymbol{e}_1$, overlapping with $x_2 = 0$ having different label $\boldsymbol{y}_2 = \boldsymbol{e}_2$. Here we compare (1) mixup and (2) mixup-without-MI. To avoid MI, we set the scheme (2) to mix only samples with different classes. Here, we sample $\lambda \sim \text{Beta}(1,1)$. For each scheme $s \in \{\text{mixup}, \text{mixup-without-MI}\}$, let θ_s be the parameter θ that minimizes the MSE loss $\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \|\hat{\boldsymbol{y}} - \boldsymbol{y}\|_2^2$. We have $\text{margin}(\theta_{\text{mixup}}) = \frac{7}{16}$ *and* $\text{margin}(\theta_{\text{mixup-without-MI}}) = \frac{1}{2}$ *as in Section C.1 of Appendix.*

This shows that we can achieve the maximum margin $\frac{1}{2}$ if we remove mixup points having manifold intrusion, while

[1] See Section E in Appendix for the details of synthetic datasets.

Table 2: The effect of manifold intrusion (MI) on logistic regression accuracy, for 7 balanced low-dimensional OpenML datasets, having more than two classes. For six out of seven, one can increase the accuracy by discarding the mixup points which incur manifold intrusion, demonstrating MI's effect on accuracy.

OpenML dataset ID	18	36	307	468	1413	1499	40984
Mixup-without-MI (%)	**73.70**	**89.21**	**56.77**	82.73	**93.33**	**96.19**	**84.88**
Mixup (%)	72.93	88.98	54.41	**83.64**	88.00	95.56	83.90

vanilla mixup degrades the margin to $\frac{7}{16}$, implying the margin reduction effect of manifold intrusion.

Now we empirically show how the manifold intrusion affects the classification accuracy of mixup for real datasets in OpenML. Similar to the previous example, we consider two schemes: (1) mixup and (2) mixup-without-MI, where the second scheme is defined as a usual mixup with the exclusion of mixup points that incur the manifold intrusion. Here we decide whether a mixup point is suffering from manifold intrusion, using a relaxed version of the definition suggested in (Guo et al., 2019):

Definition 1. *For a real dataset $D = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^n$, we call a mixed point $\boldsymbol{x}^{\text{mix}} = \lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2$ has manifold intrusion if the label of the nearest neighbor $\boldsymbol{x}^{\text{nn}} = \arg\min_{\boldsymbol{x}\in X} d(\boldsymbol{x}, \boldsymbol{x}^{\text{mix}})$ is different from $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$.*

Table 2 compares the classification accuracy of (1) mixup and (2) mixup-without-MI, for 7 balanced low-dimensional datasets in OpenML having more than two classes, and having less than 20 features. We trained a logistic regression model for classification, and the result is averaged out over five trials. It turns out that for 6 out of 7 datasets, mixup-without-MI has a higher accuracy than mixup. This shows that excluding the manifold-intruding mixup points is beneficial for improving the classification accuracy, for a large portion of tested low-dimensional real datasets.

3.2. Labeling method in mixup is sub-optimal

Here we show that for datasets not having manifold intrusion, the labeling method used in mixup is sub-optimal in terms of margin and accuracy. Note that for a given mixed point $\boldsymbol{x}^{\text{mix}} = \lambda \boldsymbol{x}_i + (1-\lambda)\boldsymbol{x}_j$, the conventional method uses a linear interpolation of labels of original samples, represented as $\boldsymbol{y}^{\text{lin}} = \lambda \boldsymbol{y}_i + (1-\lambda)\boldsymbol{y}_j$. We call this method as *linear labeling*. Here we compare this with an alternative labeling dubbed as *logistic labeling*, represented as $\boldsymbol{y}^{\text{log}} = \rho \boldsymbol{y}_i + (1-\rho)\boldsymbol{y}_j$ where $\rho = \frac{1}{1+\exp\{-2(\lambda-1/2)/\sigma^2\}}$ for some $\sigma > 0$. We show that mixup with *linear* labeling performs worse than mixup with *logistic* labeling for some datasets, implying the sub-optimality of conventional label.

Below we start with analyzing the margin of mixup with linear/logistic labeling methods for a synthetic dataset.

Example 2. *Consider a dataset with three data points in Fig. 3a, where the feature-label pairs are defined as* $(\boldsymbol{x}_1, \boldsymbol{y}_1) = ([-d, +d], \boldsymbol{e}_1)$, $(\boldsymbol{x}_2, \boldsymbol{y}_2) = ([+d, +d], \boldsymbol{e}_2)$, *and* $(\boldsymbol{x}_3, \boldsymbol{y}_3) = ([-d, -d], \boldsymbol{e}_3)$ *with* $d = 5$. *We train softmax regression model* $\boldsymbol{W} = [\boldsymbol{w}_1^T; \boldsymbol{w}_2^T; \boldsymbol{w}_3^T] \in \mathbb{R}^{3\times 2}$ *for vanilla training and mixup with linear/logistic labeling. Given* $\boldsymbol{x} = [x^{(1)}, x^{(2)}]$, *the prediction score is denoted as* $\boldsymbol{p} = [p_1, p_2, p_3] = \frac{1}{\sum_{i=1}^3 \exp(\boldsymbol{w}_i^T \boldsymbol{x})}[e^{\boldsymbol{w}_1^T \boldsymbol{x}}, e^{\boldsymbol{w}_2^T \boldsymbol{x}}, e^{\boldsymbol{w}_3^T \boldsymbol{x}}]$. *Figs. 3b, 3c, 3d compare the decision boundaries. Mixup with linear label harms the margin, while mixup with logistic label achieves the maximum margin.*

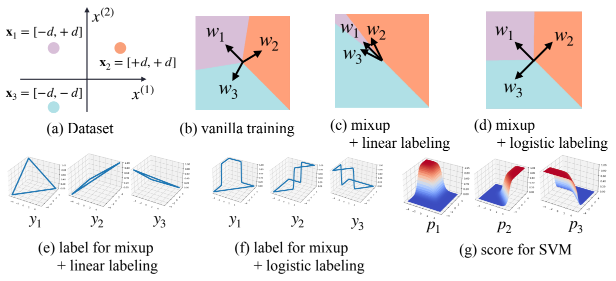


Figure 3: Comparison of linear/logistic labeling for a dataset in (a). From (b)–(d), one can see that logistic labeling results in the max-margin classifier while linear labeling does worse than vanilla training. This implies the need for nonlinear labeling for mixup. See Section 3.2 for more detailed explanations.

The effect of linear/logistic labeling methods on the margin can be explained as follows. Recall that the softmax regression finds the model that minimizes the cross entropy loss between the prediction $\boldsymbol{p}$ and the label $\boldsymbol{y}$, and we achieve the minimum when $\boldsymbol{p} = \boldsymbol{y}$ holds. In Fig. 3a, consider mixing $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ with coefficient $\lambda \in [0, 1]$ along the line $x^{(2)} = x^{(1)}$, generating $\boldsymbol{x}^{\text{mix}} = \lambda \boldsymbol{x}_2 + (1-\lambda)\boldsymbol{x}_3 = [(2\lambda-1)d, (2\lambda-1)d]$. As in Fig. 3e, mixup with linear labeling assigns the label $\boldsymbol{y}^{\text{lin}} = [y_1, y_2, y_3] = [0, \lambda, 1-\lambda]$ for the mixup points on the line $x^{(2)} = x^{(1)}$. The model $\boldsymbol{W}$ is trained in a way that $\boldsymbol{p}$ resembles $\boldsymbol{y}^{\text{lin}}$, *i.e.*, set $p_1 = 0$ and set both p_2 and p_3 as a linear function of λ along the line $x^{(2)} = x^{(1)}$. This is true when $\boldsymbol{w}_2$ and $\boldsymbol{w}_3$ are close enough and symmetric about the line $x^{(2)} = -x^{(1)}$, as in Fig. 3c; if we set $\boldsymbol{w}_2 = [-1+\frac{1}{2d}, 1+\frac{1}{2d}]$ and $\boldsymbol{w}_3 = [-1-\frac{1}{2d}, 1-\frac{1}{2d}]$, then using $\exp(\boldsymbol{w}_2^T \boldsymbol{x}^{\text{mix}}) \simeq 1 + \boldsymbol{w}_2^T \boldsymbol{x}^{\text{mix}}$, we have $p_2 \simeq \frac{1}{2}(1 + \boldsymbol{w}_2^T \boldsymbol{x}^{\text{mix}}) = \lambda = y_2$ and similarly $p_3 \simeq 1 - \lambda = y_3$. This implies that the model $\boldsymbol{W}$ trained to set $\boldsymbol{p} = \boldsymbol{y}^{\text{lin}}$ will look like the solution in Fig. 3c, especially when d is large. Thus, fitting the softmax regression model to mixup with linear labeling reduces the margin in this toy dataset.

We now explain how the logistic labeling enjoys a large margin as in Fig. 3d. Again, consider mixup points $\boldsymbol{x}^{\text{mix}} = \lambda \boldsymbol{x}_2 + (1-\lambda)\boldsymbol{x}_3 = [(2\lambda-1)d, (2\lambda-1)d]$. As in Fig. 3f, the logistic labeling with $\sigma = \frac{1}{2\sqrt{d}}$ assigns the label $\boldsymbol{y}^{\text{log}} = [y_1, y_2, y_3] = [0, \rho, 1-\rho]$ for these mixup points, where $\rho = \frac{1}{1+\exp(-8d(\lambda-1/2))}$. Then, one can confirm that $\boldsymbol{p} = \boldsymbol{y}^{\text{log}}$ holds for the support vector machine (SVM) solution having $\boldsymbol{w}_1 = [-1, 1]$, $\boldsymbol{w}_2 = [1, 1]$, and $\boldsymbol{w}_3 = [-1, -1]$. Note that the logistic label $\boldsymbol{y}$ in Fig. 3f resembles the score $\boldsymbol{p}$ of SVM

Algorithm 1 GenLabel

```
Input Dataset $S = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$, learning rate $\eta$, loss ratio $\gamma$
Output Discriminative model $f_\theta(\cdot)$
1: $\theta \leftarrow$ Random initial model parameter
2: $p_c(\cdot) \leftarrow$ Estimated density for class $c \in [k]$
3: for $(\boldsymbol{x}_i, \boldsymbol{y}_i), (\boldsymbol{x}_j, \boldsymbol{y}_j) \in S$ do
4:     $(\boldsymbol{x}^{\text{mix}}, \boldsymbol{y}^{\text{mix}}) = (\lambda \boldsymbol{x}_i + (1-\lambda)\boldsymbol{x}_j, \lambda \boldsymbol{y}_i + (1-\lambda)\boldsymbol{y}_j)$
5:     $\boldsymbol{y}^{\text{gen}} \leftarrow \sum_{c=1}^k \frac{p_c(\boldsymbol{x}^{\text{mix}})}{\sum_{c'=1}^k p_{c'}(\boldsymbol{x}^{\text{mix}})} \boldsymbol{e}_c$
6:     $\theta \leftarrow \theta - \eta \nabla_\theta \{\gamma \cdot \ell_{\text{CE}}(\boldsymbol{y}^{\text{gen}}, f_\theta(\boldsymbol{x}^{\text{mix}}))$
                  $+ (1-\gamma) \cdot \ell_{\text{CE}}(\boldsymbol{y}^{\text{mix}}, f_\theta(\boldsymbol{x}^{\text{mix}}))\}$
7: end for
```

solution in Fig. 3g, which corroborates the fact that logistic labeling guides us to achieve the maximum margin.

The above analysis shows that the *linear labeling* method is harming the margin of a mixup-trained classifier, while the *logistic labeling* method is allowing mixup to enjoy the maximum margin. This clearly shows that the conventional labeling method of mixup is sub-optimal, and an appropriate re-labeling method improves the margin significantly.

We also tested the classification accuracy of mixup with linear/logistic labeling, for 133 real datsets in OpenML (Vanschoren et al., 2013) having less than 20 features, when we use the logistic regression model. In order to decouple the effect of labeling and the effect of manifold intrusion, we removed the mixed points incurring the manifold intrusion by following the criterion in Definition 1. It turns out that the accuracy gain by using logistic labeling is positive for 47.4%, zero for 16.5%, and negative for 36.1% of the tested datasets. In other words, using logistic labeling instead of linear labeling improves the accuracy for around half of the tested real datasets in OpenML. This shows that the conventional method of labeling mixup points is not optimal in terms of accuracy in many low-dimensional real datasets.

4. *GenLabel*

In the previous section, we observed two main issues of mixup. First issue is manifold intrusion, which occurs since mixup blindly interpolates randomly chosen two samples, without knowing the underlying data distribution. Second, the conventional method of labeling mixup samples is sub-optimal, in terms of margin and accuracy.

Motivated by these observations, we propose *GenLabel*, a method of re-labeling mixup samples based on the data distribution estimated by generative models. *GenLabel* contains three steps. First, we estimate the class-conditional data distribution $\widehat{p_c}(\boldsymbol{x})$ for each class c. Second, we apply the mixup-based data augmentation, generating mixup sample $\boldsymbol{x}^{\text{mix}}$ originally labeled as $\boldsymbol{y}^{\text{mix}}$. Finally, we relabel $\boldsymbol{x}^{\text{mix}}$ based on $\widehat{p_c}(\boldsymbol{x})$, *i.e.*, we define the new label as $\boldsymbol{y}^{\text{gen}} = \text{softmax}(\log \widehat{p_1}(\boldsymbol{x}^{\text{mix}}), \cdots, \log \widehat{p_k}(\boldsymbol{x}^{\text{mix}}))$. This new label is called *GenLabel* since it makes use of generative models for labeling. Note that the suggested *GenLabel* is a re-labeling method, and we follow the mixing strategy of mixup by default. Throughout the paper, the scheme called "*GenLabel*" refers to "mixup+*GenLabel*".

Now we give a formal description of *GenLabel*. Given a dataset S, we first train class-conditional generative model, thereby getting estimates on the underlying data distribution $\widehat{p_c}(\boldsymbol{x})$. Then, for randomly chosen data pair $(\boldsymbol{x}_i, \boldsymbol{y}_i), (\boldsymbol{x}_j, \boldsymbol{y}_j) \in S$, we apply mixup scheme, generating the mixed feature $\boldsymbol{x}^{\text{mix}} = \lambda \boldsymbol{x}_i + (1-\lambda)\boldsymbol{x}_j$ and the mixed label $\boldsymbol{y}^{\text{mix}} = \lambda \boldsymbol{y}_i + (1-\lambda)\boldsymbol{y}_j$. Here, the mixing coefficient follows the beta distribution, i.e., $\lambda \sim \text{Beta}(\alpha, \alpha)$ for some $\alpha > 0$. Finally, we re-label this augmented data $\boldsymbol{x}^{\text{mix}}$ as

$$\boldsymbol{y}^{\text{gen}} = \sum_{c=1}^{k} \frac{\widehat{p_c}(\boldsymbol{x}^{\text{mix}})}{\sum_{c'=1}^k \widehat{p_{c'}}(\boldsymbol{x}^{\text{mix}})} \boldsymbol{e}_c, \tag{1}$$

which is the posterior probability estimate $\widehat{\mathbb{P}}(y = c|\boldsymbol{x}^{\text{mix}})$ when we have a balanced dataset.

Since our generative model is imperfect, $\boldsymbol{y}^{\text{gen}}$ may be incorrect for some samples. Thus, we can use a combination of mixup labeling and the suggested labeling, *e.g.*, define the label of mixed point as $\gamma \boldsymbol{y}^{\text{gen}} + (1-\gamma)\boldsymbol{y}^{\text{mix}}$ for some $\gamma \in [0, 1]$. Note that our scheme reduces to the mixup when $\gamma = 0$. Using the relabeled augmented data, the algorithm trains the classifier $f_\theta : \mathbb{R}^n \to [0, 1]^k$ that predicts the label $\boldsymbol{y} = [y_1, \cdots, y_k]$ of the input data. Here, the cross-entropy loss $\ell_{\text{CE}}(\cdot)$ is used while training the classifier. The pseudocode of *GenLabel* is provided in Algorithm 1.

In summary, the proposed scheme is a novel label correction method for mixup, which first learns the data distributions for each class using class-conditional generative models, and then re-label the mixup data based on the conditional likelihood of the mixup data sampled from each class. More precisely, *GenLabel* sets the label of a mixup data as the softmax of the class-conditional log-likelihood, which matches with the posterior probability for the balanced datasets.

Remark 1. *GenLabel described in Algorithm 1 assumes that the generative model learns the data distribution in the input feature space. However, we can easily extend our algorithm to cases when we train generative models in the latent feature space. Due to the space limitation, the detailed algorithm description is in Section A.1 at Appendix.*

5. Analysis of *GenLabel*

We analyze the effect of *GenLabel* in various perspectives. In Section 5.1, we visualize *GenLabel* for toy datasets, empirically showing that *GenLabel* fixes the manifold intrusion issue of mixup. Section 5.2 provides a concrete problem instance where *GenLabel* solves the margin reduction issue of linear labeling in mixup. Finally, Section 5.3 shows that *GenLabel* improves the robustness of mixup on logistic regression models and fully-connected ReLU networks.

5.1. GenLabel solves the manifold intrusion issue

Consider the datasets given in Fig. 4a: we have nine classes of 2-dimensional Gaussian dataset on the top row, and two classes having two circles at each class on the bottom row; we call the top one as "9-class Gaussian" and the bottom one as "Four-circle" dataset. Note that both datasets contain numerous mixed points suffering from the manifold intrusion, e.g., the mixed point of blue and orange samples lie on the black class in 9-class Gaussian dataset.

Given a soft-label $\boldsymbol{y} = [y_1, \cdots, y_k]$, define the top-1 label as $y^{\text{top-1}} = \arg\max_{c\in[k]} y_c$. Fig. 4b and Fig. 4c illustrate the top-1 label of a mixed point, for the conventional labeling in mixup and the suggested *GenLabel* scheme, respectively. For the 9-class Gaussian data, we set the mixing coefficient as $\lambda = 0.6$ for the purpose of illustration. As shown in Fig. 4b, the conventional labeling method causes the label conflict issue for a large number of mixup samples. This issue has been resolved by *GenLabel* as shown in Fig. 4c: the label of mixed points assigned by *GenLabel* matches with the label of maximum margin classifier for each dataset.

This label correction method also increases the margin of a classifier. Figs. 4d, 4e, 4f and 4g show the decision boundaries of vanilla training, mixup (with original labeling), *GenLabel*, and kernel SVM using radial basis function kernel $\exp(-2\|\boldsymbol{x} - \boldsymbol{x}'\|^2)$, respectively. While vanilla training and mixup have small margin for some samples, *GenLabel* achieves the ideal margin of kernel SVM.

5.2. GenLabel improves the margin

Here we provide a concrete problem instance where switching from the *linear label* to *GenLabel* of mixup sample will increase the margin to its maximum value, *i.e.*,

$$\begin{aligned}\text{margin(SVM)} &= \text{margin(GenLabel)}\\ &> \text{margin(vanilla)} > \text{margin(mixup)},\end{aligned}$$

where SVM represents the support vector machine (Cortes & Vapnik, 1995) achieving the maximum L_2 margin.

Example 3. *Consider a dataset $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n+2}$, where the feature $\boldsymbol{x}_i \in \mathbb{R}^2$ and the label $y_i \in \{+1, -1\}$ of each point is specified in Fig. 5a. Let $\boldsymbol{\theta} = (r\cos\phi, r\sin\phi)$ be the model parameter for a fixed $r > 0$. Fig. 5b shows $\phi^\star = \arg\min_\phi \ell(r,\phi)$ for various r, where ℓ is the logistic loss applied to the (augmented) dataset. It turns out that the optimal $\phi^\star$ of GenLabel approaches to the SVM solution $\phi_{\text{svm}} = 0.25\pi$ as r increases. The detailed derivation of $\phi^\star$ for each scheme is given in Section C.2 in Appendix.*

Remark 2. *For large $r = \|\boldsymbol{\theta}\|$, the original mixup does not converge to the max-margin solution, while the mixup relabeled by GenLabel approaches to the max-margin solution.*

In summary, for Examples 2 and 3, (1) the original mixup method has a smaller margin than vanilla training, and (2) simply changing the label of the mixed points (using *GenLabel*) fixes this issue and achieves the maximum margin.

5.3. GenLabel improves the robustness of mixup

We provide mathematical analysis on the effect of *GenLabel* on the adversarial robustness. Below theorem shows that *GenLabel* is improving the robustness of mixup in logistic regression model and FC ReLU networks, for Gaussian data. Due to the space limitation, we put the formal statement and proofs at Sections B.1 and C in Appendix.

Theorem (informal)**.** *Consider binary classification problem where each class follows a Gaussian distribution. For logistic regression model and fully-connected ReLU networks parameterized by $\boldsymbol{\theta}$, we have*

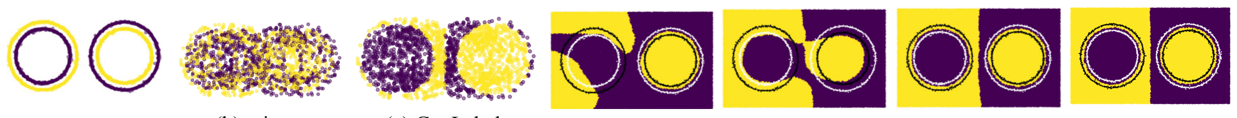
$$\ell_{\text{mixup}}(\theta) \geq \ell_{\text{GenLabel}}(\theta) \geq \ell_{\text{adv}}(\theta).$$

This shows that the adversarial loss of a model is upper bounded by the *GenLabel* loss of the model, i.e., if we find a model with *GenLabel* loss smaller than or equal to a threshold l_{th}, then the adversarial loss of this model is at most l_{th}. Compared with mixup loss, *GenLabel* loss is a tighter upper bound on the adversarial loss. This implies that *GenLabel* improves the robustness of mixup.

6. Experimental Results

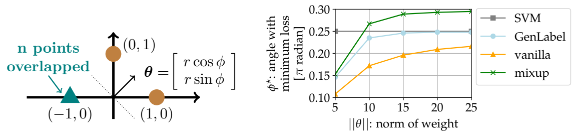
Now we investigate the effect of *GenLabel* on real datasets. We consider two models: logistic regression and fully-connected (FC) ReLU networks with 2 hidden layers. We focus on OpenML (Vanschoren et al., 2013) having various real tabular datasets. Especially, we consider 160 low-dimensional OpenML datasets having less than 20 features and less than 5000 data points. Among such 160 datasets, we filter out 27 redundant datasets with different versions, for datasets named as iris, seeds, and chscase-census, thus having 133 datasets for testing. For logistic regression model, we reported the result for 82 datasets which fit well on either KDE or GM model; we used a dataset if either of the generative model has more than 95% of train accuracy[2]. Recall that we allow the combination of the mixup labeling $\boldsymbol{y}^{\text{mix}}$ and the suggested labeling $\boldsymbol{y}^{\text{gen}}$, i.e., re-label the mixed point by $\gamma\boldsymbol{y}^{\text{gen}} + (1-\gamma)\boldsymbol{y}^{\text{mix}}$ for $\gamma \in [0, 1]$. Here we choose the optimal mixing ratio γ using cross-validations. All algorithms are implemented

[2]It is clear that our algorithm will not work as expected if the chosen generative model has a poor fit to the data. By using a larger class of generative models, such as flow (Kingma & Dhariwal, 2018) and diffusion models (Kingma et al., 2021), one should be able to handle the omitted datasets, but we leave it as future work.



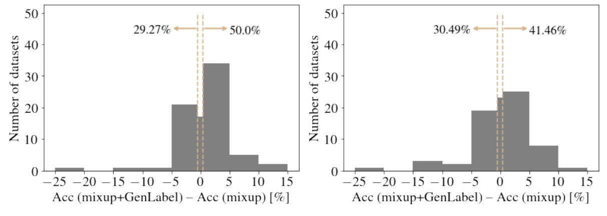
(a) dataset (b) mixup, top-1 label of mixed points (c) GenLabel, top-1 label of mixed points (d) vanilla training, decision boundary (e) mixup, decision boundary (f) GenLabel, decision boundary (g) kernel SVM, decision boundary

Figure 4: Comparison of vanilla training, mixup and the suggested *GenLabel* for Four-circle dataset. (a): dataset. (b), (c): mixup points and their top-1 labels of $\boldsymbol{y}^{\text{mix}}$ and $\boldsymbol{y}^{\text{gen}}$. One can see that $\boldsymbol{y}^{\text{mix}}$ causes label conflicts while $\boldsymbol{y}^{\text{gen}}$ does not. (d), (e), (f), (g): decision boundaries. The margin of *GenLabel* classifier is larger than those of vanilla training and mixup and matches that of the kernel SVM.



(a) Dataset (b) Comparison of $\phi^\star$

Figure 5: (a) Logistic regression on $(\boldsymbol{x}_1, y_1) = ([1,0], +1)$, $(\boldsymbol{x}_2, y_2) = ([0,1], +1)$, and $(\boldsymbol{x}_i, y_i) = ([-1,0], -1)$ for $i = 3, 4, \cdots, n+2$. (b) *GenLabel* quickly converges to the SVM solution as $\|\boldsymbol{\theta}\|$ increases much faster than the vanilla scheme. The original mixup converge to a suboptimal value.



(a) $\alpha = 1$ (b) $\alpha = 2$

Figure 6: The accuracy increase (%) of *GenLabel* (CV) compared with mixup for 82 OpenML datasets, where $\lambda \sim \text{Beta}(\alpha, \alpha)$. For both α values, a large portion of tested datasets enjoy the accuracy improvement by combining *GenLabel* with mixup.

in PyTorch (Paszke et al., 2017), and the experimental details are summarized in Section E in Appendix. All results are averaged out over 5 trials. The github repo for reproducible code is given in `https://github.com/UW-Madison-Lee-Lab/GenLabel_official`.

Suggested schemes For OpenML datasets, we considered three types of generative models: Gaussian mixture (GM), kernel density estimator (KDE) with Gaussian kernel, and 1-nearest-neighbor (NN) density estimator (Bishop, 2006). We denote each scheme by *GenLabel* (GM), *GenLabel* (KDE), and *GenLabel* (NN), respectively. We also considered using cross-validation (CV) to choose either GM or KDE: this scheme is denoted by *GenLabel* (CV). For logistic regression, we used *GenLabel* on the input feature space, and for FC ReLU networks, we used *GenLabel* on the latent feature space at the penultimate layer. For image datasets (MNIST, CIFAR-10, CIFAR-100 and TinyImageNet), we tested *GenLabel* on mixup and manifold-mixup, the results of which are given in the Appendix.

Compared schemes For all datasets, we compared our scheme with vanilla training, mixup, and adamixup (Guo et al., 2019). For OpenML datasets, we added the comparison with (1) generative classifier using Gaussian mixture (GM) as the generative model and (2) mixup+*excluding MI*, which excludes the mixup points with manifold intrusion (MI). Here, we detected mixup points with MI using Definition 1. For image datasets, we tested mixup and manifold-mixup and compared them with the performances of mixup+*GenLabel* and "manifold-mixup"+*GenLabel*.

6.1. Results on clean accuracy

We compare the accuracy of *GenLabel* with various baselines, on the logistic regression model.

Table 3: Accuracies of *GenLabel* and baselines training a logistic regression model, for OpenML datasets. We tested on (a) 82 datasets where generative model well fits the data, and (b) 10 balanced datasets among them. Each cell has two values: the result for case (a), and the result for case (b) in the parenthesis. For example, among 82 datasets, *GenLabel* has higher accuracy than mixup for 54.88% of tested datasets. The number of datasets where *GenLabel* outperforms each baseline is larger than or equal to the number of datasets where *GenLabel* is worse than the baseline.

Mixup+GenLabel (CV) versus	**Higher**	**On-par**	**Lower**
Vanilla	**47.56**% (**50**%)	7.32% (30%)	45.12% (20%)
Adamixup	**45.12**% (**50**%)	9.76% (20%)	**45.12**% (30%)
Mixup	**54.88**% (**40**%)	7.32% (30%)	35.37% (30%)
Mixup + Excluding MI	**53.66**% (**50**%)	6.10% (20%)	39.02% (30%)
Generative Classifier (GM)	**52.42**% (**70**%)	3.66% (10%)	41.46% (20%)

Statistics of *GenLabel* vs baselines Fig. 6 compares the accuracies of *GenLabel* (CV) and mixup on 82 OpenML datasets. The x-axis is $(A^{\text{cln}}_{\text{GenLabel}} - A^{\text{cln}}_{\text{mixup}})$, the accuracy increase with the aid of *GenLabel*, and the y-axis is the number of datasets having the accuracy gain. Recall that $\lambda \sim \text{Beta}(\alpha, \alpha)$ for given $\alpha = 1$ or $\alpha = 2$. For both settings, the accuracy of *GenLabel* is greater than or equal to that of mixup for more than 69% of the tested datasets.

Table 3 compares accuracies of *GenLabel* and baselines, for OpenML datasets. Each cell shows the percentage of datasets satisfying the condition for (1) 82 datasets and (2) 10 balanaced datasets, where result for (2) is in the parenthesis. For example, for 54.88% of 82 datasets, *GenLabel* has higher accuracy than mixup. The number of datasets where *GenLabel* (CV) outperforms each baseline is larger than or equal to the number of datasets where the baseline outperforms *GenLabel* (CV). The results for 10 balanced datasets in the parenthesis in Table 3 show similar behavior.

Table 4: Accuracy (%) comparison on selected OpenML datasets, for the logistic regression model. For dataset IDs 830 and 1413, *GenLabel* has over 8% accuracy gain than mixup. Note that the comparisons for all tested datasets are in Table 3.

Methods \ Dataset ID	721	777	792	830	855	913	1413
Generative Classifier (GM)	78.33	53.33	74.67	78.67	65.33	71.33	95.56
Vanilla	79.67	58.67	73.20	77.60	63.33	70.80	95.56
AdaMixup	80.33	**64.00**	73.87	78.40	66.67	70.53	92.44
Mixup	79.33	62.67	73.47	76.27	66.00	69.87	88.00
Mixup + Excluding MI	79.67	62.67	74.53	78.13	66.40	71.47	93.33
Mixup + GenLabel (GM)	**81.00**	58.67	75.47	**86.13**	66.40	71.47	96.00
Mixup + GenLabel (KDE)	79.67	58.67	**75.87**	77.33	**67.60**	72.67	96.00
Mixup + GenLabel (CV)	80.33	**64.00**	75.60	84.53	**67.33**	**73.20**	**96.44**

Table 5: Robustness against a black-box attack (Brendel et al., 2018) with radius $\varepsilon = 0.1$, on 13 balanced OpenML datasets having less than 500 data samples and less than 20 features, for FC ReLU networks with 2 hidden layers. (a): *GenLabel* (best among NN and GM) versus baselines. The portion of datasets where *GenLabel* outperforms is over 53%. (b): Robust accuracy (%) for 6 datasets. *GenLabel* achieves 2–10% improvement over mixup. See Table 7 in Appendix for the full result.

(a) Statistics for 13 balanced datasets

Mixup+GenLabel (Best) versus	**Higher**	**On-par**	**Lower**
Vanilla	**53.85**%	23.08%	23.08%
Adamixup	**61.54**%	7.69%	30.77%
Mixup	**69.23**%	0.00%	30.77%
Mixup + Excluding MI	**69.23**%	0.00%	30.77%

(b) 6 selected datasets

Methods \ OpenML ID	446	468	683	755	763	1413
Vanilla	29.67	34.55	51.11	41.05	64.27	68.00
AdaMixup	30.33	37.27	51.11	37.89	63.20	67.11
Mixup	30.67	37.27	50.00	36.84	65.07	67.56
Mixup + Excluding MI	31.67	31.82	**52.22**	38.95	63.20	70.67
Mixup + GenLabel (GM)	37.00	**42.73**	**52.22**	**43.16**	61.87	71.11
Mixup + GenLabel (NN)	**38.00**	32.73	46.67	**43.16**	**66.93**	**77.33**

Performance of *GenLabel* on selected datasets Table 4 shows some selected datasets when *GenLabel* performs better than mixup, *e.g.*, 8% accuracy gain for dataset IDs 830 and 1413. The best generative model (GM or KDE) for *GenLabel* varies depending on the dataset. Moreover, *GenLabel* (CV) successfully chooses the appropriate generative model and outperforms other baselines in all datasets in Table 4.

Interestingly, for dataset IDs 721, 830, 913 and 1413, mixup performs worse than vanilla, but mixup combined with *GenLabel* overcomes the limitation of mixup and outperforms vanilla. For other datasets (IDs 777, 792 and 855) where mixup outperforms vanilla, *GenLabel* with an appropriate choice of generative model further improves mixup. Table 4 also shows that *GenLabel* (CV) is performing better or equal to other baselines (adaMixup, generative classifer, mixup with excluding MI), for the selected datasets. Note that the comparison for all datasets are given in Table 3.

6.2. Results on adversarial robustness

GenLabel has some improvement on robustness, compared with baselines. We tested a black-box attack (Brendel et al., 2018) on 13 balanced OpenML datasets; black-box attack is used to avoid the gradient obfuscation issue (Athalye et al., 2018) of gradient-based attacks (e.g., FGSM/PGD).

Table 5 shows the robust accuracy of *GenLabel* and baselines. *GenLabel* improves the robust accuracy of mixup for more than 69% of tested datasets. In Table 5(b), *GenLabel* improves the robustness by 2–10 %. Still, we remark that mixup approaches should not be considered as a standalone approach as they are less effective at obtaining robust models compared to adversarial training algorithms.

6.3. Extension to high-dimensional image datasets

We also tested *GenLabel* on MNIST, CIFAR-10, CIFAR-100 and TinyImageNet-200. *GenLabel* has a marginal gain. See Section A.2 in Appendix for the results.

7. Discussions

Regarding the suggested *GenLabel*, one can think of the following discussion topics: (1) using generative models with implicit/approximate density for *GenLabel*, (2) using generative models not only for *labeling*, but also *mixing*.

7.1. Extension to other generative models

Note that our method can be applied to generative models do not having explicit density. For generative models with approximated density $\tilde{p}_c(\boldsymbol{x})$, as in VAEs (Kingma et al., 2019), we can replace $p_c(\boldsymbol{x})$ by $\tilde{p}_c(\boldsymbol{x})$ in Algorithm 1. For GANs (Xia et al., 2021) having implicit density, we use a proxy to the density $p_c(\boldsymbol{x})$ by inverting the generator (Creswell & Bharath, 2018). Let $\mathcal{M}_c = \{G(\boldsymbol{w}, c) : \boldsymbol{w} \in \mathbb{R}^d\}$ be the data manifold generated by generator G for class c. Assuming the spherical Gaussian noise model used for manifold learning (Hastie & Stuetzle, 1989; Chang & Ghosh, 2001), we can estimate $p_c(\boldsymbol{x}) = \int p(\boldsymbol{x}|G(\boldsymbol{w}, c))p(G(\boldsymbol{w}, c))d\boldsymbol{w} \simeq \frac{1}{n}\sum_{i=1}^{n} \exp(-d(\boldsymbol{x}, G(\boldsymbol{w}_i, c))$ by choosing n random samples of $\boldsymbol{w}$. Then, we approximate this summation with the dominant term, expressed as $\max_i \exp(-d(\boldsymbol{x}, G(\boldsymbol{w}_i, c)) = \exp(-d(\boldsymbol{x}, \mathcal{M}_c))$. Thus, we replace $p_c(\boldsymbol{x})$ by $\exp(-d(\boldsymbol{x}, \mathcal{M}_c))$ in Algorithm 1.

7.2. Using generative models for mixing and labeling

In *GenLabel*, mixed points $\boldsymbol{x}^{\text{mix}}$ are obtained by existing mixing strategies, and generative models are used only for re-labeling the mixed points. Can we also use generative models for making better mixed points? We suggest the following. For a target class pair c_1 and c_2, we first choose a mixing coefficient $\lambda \in [0, 1]$, e.g., using a Beta distribution. Then, we find $\boldsymbol{x}^{\text{mix}}$ satisfying $\frac{p_{c_1}}{p_{c_1}+p_{c_2}} = \lambda$ and label it as $\boldsymbol{y}^{\text{mix}} = \frac{p_{c_1}}{p_{c_1}+p_{c_2}}\boldsymbol{e}_{c_1} + \frac{p_{c_2}}{p_{c_1}+p_{c_2}}\boldsymbol{e}_{c_2}$, where $p_{c_1} = p_{c_1}(\boldsymbol{x}^{\text{mix}})$ and $p_{c_2} = p_{c_2}(\boldsymbol{x}^{\text{mix}})$. In Section F in Appendix, we provide this algorithm for finding mixed point $\boldsymbol{x}^{\text{mix}}$ using generative models (Gaussian mixture models and GANs) and the experimental results of this algorithm on synthetic/real datasets.

8. Conclusion

In this paper, we closely examined the failure scenarios of mixup for low dimensional data, and specify two main issues of mixup: (1) the manifold intrusion reduces both margin and accuracy, and (2) even when there is no manifold intrusion, the linear labeling method harms the margin and accuracy. Motivated by these observations, we proposed *GenLabel*, a novel way of labeling mixup points by using generative models. We provided concrete examples where *GenLabel* solves the main issues of mixup and achieves maximum margin. We also provided empirical results showing that *GenLabel* helps improving the accuracy of mixup on a large number of low-dimensional OpenML datasets.

Acknowledgements

This work was supported by an American Family Insurance grant via American Family Insurance Data Science Institute at University of Wisconsin-Madison.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Antoniou, A., Storkey, A., and Edwards, H. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, pp. 274–283. PMLR, 2018.

Bishop, C. M. Pattern recognition. *Machine learning*, 128 (9), 2006.

Brendel, W., Rauber, J., and Bethge, M. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations*, 2018.

Carratino, L., Cissé, M., Jenatton, R., and Vert, J.-P. On mixup regularization. *arXiv preprint arXiv:2006.06049*, 2020.

Chang, K.-Y. and Ghosh, J. A unified model for probabilistic principal surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):22–41, 2001.

Choi, H., Jang, E., and Alemi, A. A. Waic, but why? generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392*, 2018.

Cortes, C. and Vapnik, V. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Creswell, A. and Bharath, A. A. Inverting the generator of a generative adversarial network. *IEEE transactions on neural networks and learning systems*, 2018.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, pp. 2206–2216. PMLR, 2020.

Donahue, J., Krähenbühl, P., and Darrell, T. Adversarial feature learning. In *International Conference on Learning Representations*, 2017.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., and Courville, A. Adversarially learned inference. In *International Conference on Learning Representations*, 2017.

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Tran, B., and Madry, A. Adversarial robustness as a prior for learned representations. *arXiv preprint arXiv:1906.00945*, 2019.

Feurer, M., van Rijn, J. N., Kadra, A., Gijsbers, P., Mallik, N., Ravi, S., Müller, A., Vanschoren, J., and Hutter, F. Openml-python: an extensible python api for openml. *arXiv:1911.02490*, 2019.

Ghojogh, B. and Crowley, M. Linear and quadratic discriminant analysis: Tutorial. *arXiv preprint arXiv:1906.02590*, 2019.

Ghosh, P., Losalka, A., and Black, M. J. Resisting adversarial attacks using gaussian mixture variational autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 541–548, 2019.

Greenewald, K., Gu, A., Yurochkin, M., Solomon, J., and Chien, E. k-mixup regularization for deep learning via optimal transport. *arXiv preprint arXiv:2106.02933*, 2021.

Guo, H., Mao, Y., and Zhang, R. Mixup as locally linear out-of-manifold regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3714–3722, 2019.

Hastie, T. and Stuetzle, W. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989.

Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., and Lakshminarayanan, B. Augmix: A simple data processing method to improve robustness and uncertainty. In *International Conference on Learning Representations*, 2020.

Hwang, S.-H. and Whang, S. E. MixRL: Data mixing augmentation for regression using reinforcement learning. *arXiv preprint arXiv:2106.03374*, 2021.

Ilyas, A., Jalal, A., Asteri, E., Daskalakis, C., and Dimakis, A. G. The robust manifold defense: Adversarial training using generative models. *arXiv preprint arXiv:1712.09196*, 2017.

Inoue, H. Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*, 2018.

Ju, A. and Wagner, D. E-abs: Extending the analysis-by-synthesis robust classification model to more complex image domains. In *Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security*, pp. 25–36, 2020.

Kim, J.-H., Choo, W., and Song, H. O. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *International Conference on Machine Learning*, pp. 5275–5285. PMLR, 2020.

Kim, J.-H., Choo, W., Jeong, H., and Song, H. O. Co-mixup: Saliency guided joint mixup with supermodular diversity. In *International Conference on Learning Representations*, 2021.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, 2018.

Kingma, D. P., Welling, M., et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

Kingma, D. P., Salimans, T., Poole, B., and Ho, J. Variational diffusion models. In *Advances in Neural Information Processing Systems*, 2021.

Lee, K., Lee, K., Lee, H., and Shin, J. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, 2018.

Li, Y., Bradshaw, J., and Sharma, Y. Are generative classifiers more robust to adversarial attacks? In *International Conference on Machine Learning*, pp. 3804–3814. PMLR, 2019.

Liu, Z., Li, S., Wu, D., Chen, Z., Wu, L., Guo, J., and Li, S. Z. Unveiling the power of mixup for stronger classifiers. *arXiv preprint arXiv:2103.13027*, 2021.

Ng, A. Y. and Jordan, M. I. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems*, pp. 841–848, 2002.

Park, J., Yang, J. Y., Shin, J., Hwang, S. J., and Yang, E. Saliency grafting: Innocuous attribution-guided mixup with calibrated label mixing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Perez, L. and Wang, J. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

Samangouei, P., Kabkab, M., and Chellappa, R. Defense-gan: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018.

Schott, L., Rauber, J., Bethge, M., and Brendel, W. Towards the first adversarially robust neural network model on mnist. *arXiv preprint arXiv:1805.09190*, 2018.

Serrà, J., Álvarez, D., Gómez, V., Slizovskaia, O., Núñez, J. F., and Luque, J. Input complexity and out-of-distribution detection with likelihood-based generative models. In *International Conference on Learning Representations*, 2020.

Shimada, T., Yamaguchi, S., Hayashi, K., and Kobayashi, S. Data interpolating prediction: Alternative interpretation of mixup. *arXiv preprint arXiv:1906.08412*, 2019.

Song, Y., Kim, T., Nowozin, S., Ermon, S., and Kushman, N. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *International Conference on Learning Representations*, 2018.

Soudry, D., Hoffer, E., Nacson, M. S., and Srebro, N. The implicit bias of gradient descent on separable data. In *International Conference on Learning Representations*, 2018.

Tanaka, F. H. K. d. S. and Aranha, C. Data augmentation using gans. *arXiv preprint arXiv:1904.09135*, 2019.

Tokozume, Y., Ushiku, Y., and Harada, T. Learning from between-class examples for deep sound recognition. In *International Conference on Learning Representations*, 2018.

Uddin, A., Monira, M., Shin, W., Chung, T., Bae, S.-H., et al. Saliencymix: A saliency guided data augmentation strategy for better regularization. In *International Conference on Learning Representations*, 2021.

Vanschoren, J., van Rijn, J. N., Bischl, B., and Torgo, L. Openml: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013. doi:10.1145/2641190.2641198. URL `http://doi.acm.org/10.1145/2641190.2641198`.

Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., and Bengio, Y. Manifold mixup: Better representations by interpolating hidden states. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6438–6447, 2019.

Xia, W., Zhang, Y., Yang, Y., Xue, J.-H., Zhou, B., and Yang, M.-H. GAN inversion: A survey. *arXiv preprint arXiv:2101.05278*, 2021.

Xiao, C., Li, B., Zhu, J. Y., He, W., Liu, M., and Song, D. Generating adversarial examples with adversarial networks. In *27th International Joint Conference on Artificial Intelligence, IJCAI 2018*, pp. 3905–3911. International Joint Conferences on Artificial Intelligence, 2018.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6023–6032, 2019.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

Zhang, L., Deng, Z., Kawaguchi, K., Ghorbani, A., and Zou, J. How does mixup help with robustness and generalization? In *International Conference on Learning Representations*, 2021.

Algorithm 2 *GenLabel* (using generative models for the latent feature)

```
Input Dataset $S = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$, input feature set $X = \{\boldsymbol{x}_i\}_{i=1}^n$, learning rate $\eta$, loss ratio $\gamma$
Output Trained discriminative model $f_{\theta,\phi}$
1: $\theta, \phi \leftarrow$ Vanilla-trained parameter for model $f_{\theta,\phi} = f_\theta^{\text{cls}} \circ f_\phi^{\text{feature}}$
2: $p_c(\boldsymbol{z}) \leftarrow$ Density estimated by generative model for latent feature $\boldsymbol{z} \in f_\phi^{\text{feature}}(X)$, conditioned on class $c \in [k]$
3: for $(\boldsymbol{x}_i, \boldsymbol{y}_i), (\boldsymbol{x}_j, \boldsymbol{y}_j) \in S$ do
4:     $(\boldsymbol{x}^{\text{mix}}, \boldsymbol{y}^{\text{mix}}) \leftarrow (\lambda \boldsymbol{x}_i + (1-\lambda)\boldsymbol{x}_j, \lambda \boldsymbol{y}_i + (1-\lambda)\boldsymbol{y}_j)$
5:     $\boldsymbol{z}^{\text{mix}} \leftarrow f_\phi^{\text{feature}}(\boldsymbol{x}^{\text{mix}})$
6:     $\boldsymbol{y}^{\text{gen}} \leftarrow \sum_{c=1}^k \frac{p_c(\boldsymbol{z}^{\text{mix}})}{\sum_{c'=1}^k p_{c'}(\boldsymbol{z}^{\text{mix}})} \boldsymbol{e}_c$
7:     $\theta \leftarrow \theta - \eta \nabla_\theta \{\gamma \cdot \ell_{\text{CE}}(\boldsymbol{y}^{\text{gen}}, f_\theta^{\text{cls}}(\boldsymbol{z}^{\text{mix}})) + (1-\gamma) \cdot \ell_{\text{CE}}(\boldsymbol{y}^{\text{mix}}, f_\theta^{\text{cls}}(\boldsymbol{z}^{\text{mix}}))\}$
8: end for
```

A. Additional algorithms and experimental results

A.1. GenLabel in the latent feature space

Here we illustrate how *GenLabel* provided in Algorithm 1 can be extended to the case of learning generative models in the latent feature space. The procedure is in three steps. First, we use existing training algorithm (vanilla training or mixup training) to learn a classifier $f_{\theta,\phi} = f_\theta^{\text{cls}} \circ f_\phi^{\text{feature}}$ where f_ϕ^{feature} is the feature extractor part and f_θ^{cls} is the classification part. Second, given the trained feature extractor f_ϕ^{feature}, we train a class-conditional generative model for the latent feature $\boldsymbol{z} = f_\phi^{\text{feature}}(\boldsymbol{x})$, and denote the learned density for class c by $p_c(\boldsymbol{z})$. Third, we freeze f_ϕ^{feature} and fine-tune f_θ^{cls} using the following manner. We first follow the mixup process: for $(\boldsymbol{x}_i, \boldsymbol{y}_i), (\boldsymbol{x}_j, \boldsymbol{y}_j) \in S$, we generate the augmented data $\boldsymbol{x}^{\text{mix}} = \lambda \boldsymbol{x}_i + (1-\lambda)\boldsymbol{x}_j$ having label $\boldsymbol{y}^{\text{mix}} = \lambda \boldsymbol{y}_i + (1-\lambda)\boldsymbol{y}_j$. Then, we re-label this augmented data by $\boldsymbol{y}^{\text{gen}} = \sum_{c=1}^k \frac{p_c(\boldsymbol{z}^{\text{mix}})}{\sum_{c'=1}^k p_{c'}(\boldsymbol{z}^{\text{mix}})} \boldsymbol{e}_c$, where $\boldsymbol{z}_{\text{mix}} = f_\phi^{\text{feature}}(\boldsymbol{x}_{\text{mix}})$. Finally, we train f_θ^{cls} using $(\boldsymbol{z}^{\text{mix}}, \boldsymbol{y}^{\text{gen}})$, the hidden representation of mixed sample and the suggested label for the mixed feature. The pseudocode of this *GenLabel* variant (for the latent feature) is given in Algorithm 2.

A.2. GenLabel on high dimensional image datasets

In the main manuscript, we focused on the result for low-dimensional datasets. Here, we provide our experimental results on high dimensional image datasets, including MNIST, CIFAR-10, CIFAR-100, and TinyImageNet-200.

We test GenLabel combined with existing data augmentation schemes of mixup (Zhang et al., 2018) and manifold-mixup (Verma et al., 2019). We compare our schemes with mixup and manifold-mixup. We also compare with AdaMixup, which avoids the manifold intrusion of mixup, similar to our work. For measuring the adversarial robustness, we test under AutoAttack (Croce & Hein, 2020), which is developed to overcome gradient obfuscation (Athalye et al., 2018), containing four white/black-box attack schemes (including auto-PGD) that does not need any specification of free parameters. The attack radius ε for each dataset is specified in Section E.

GenLabel variant used for image datasets For image datasets, we learn generative models in the latent space. To be specific, we use a variant of GenLabel, which learns the generative model (Gaussian mixture model) and the discriminative model at the same time. The pseudocode of this variant is given in Algorithm 3, and below we explain the details of this algorithm.

Consider a neural network $f_\theta = f_\theta^{\text{cls}} \circ f_\theta^{\text{feature}}$ parameterized by θ, which is composed of the feature extractor part $f_\theta^{\text{feature}}$ and the classification part f_θ^{cls}. We train a Gaussian mixture (GM) model on $f_\theta^{\text{feature}}(\boldsymbol{x})$, the hidden representation of input $\boldsymbol{x}$. In this algorithm, we consider updating the estimated GM model parameters (mean and covariance) at each batch training. At each iteration t, we randomly choose B batch samples $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^B$ from the dataset D. Then, we estimate the class-conditional mean and covariance of GM model in the hidden feature space. In other words, we compute the mean $\boldsymbol{\mu}_c^{(t)} = \frac{1}{|S_c|} \sum_{i \in S_c} f_\theta^{\text{feature}}(\boldsymbol{x}_i)$ and the covariance $\boldsymbol{\Sigma}_c^{(t)} = \frac{1}{|S_c|} \sum_{i \in S_c} (f_\theta^{\text{feature}}(\boldsymbol{x}_i) - \boldsymbol{\mu}_c^{(t)})(f_\theta^{\text{feature}}(\boldsymbol{x}_i) - \boldsymbol{\mu}_c^{(t)})^T$ for each class $c \in [k]$, where $S_c = \{i : \boldsymbol{y}_i = \boldsymbol{e}_c\}$ is the set of samples with label c within the batch. For simplicity, we approximate the covariance matrix as a multiple of identity matrix, by setting $\boldsymbol{\Sigma}_c^{(t)} \leftarrow \frac{1}{d}\text{trace}(\boldsymbol{\Sigma}_c^{(t)})\boldsymbol{I}_d$. Here, we consider making use of the parameters (mean and covariance) estimated in the previous batches as well, by introducing a memory ratio factor $\beta \in [0,1]$. Formally, the rule for updating mean $\boldsymbol{\mu}_c^{(t)}$ and covariance $\boldsymbol{\Sigma}_c^{(t)}$ are represented as $\boldsymbol{\mu}_c^{(t)} \leftarrow (1-\beta)\boldsymbol{\mu}_c^{(t)} + \beta \boldsymbol{\mu}_c^{(t-1)}$

Algorithm 3 GenLabel (learning generative/discriminative models at the same time)

```
Input Data $D$, mix function $\text{mix}(\cdot)$, learning rate $\eta$, loss ratio $\gamma$, memory ratio $\beta$, batch size $B$, max iteration $T$
Output Trained model $f_\theta = f_\theta^{\text{cls}} \circ f_\theta^{\text{feature}}$
  $\theta \leftarrow$ Random initial model parameter,          $\pi \leftarrow$ Permutation of $[B]$
  $(\boldsymbol{\mu}_c^{(0)}, \boldsymbol{\Sigma}_c^{(0)}) \leftarrow (\mathbf{0}, \boldsymbol{I}_d)$ for $c \in [k]$
  for iteration $t = 1, 2, \cdots, T$ do
    $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^B \leftarrow$ Randomly chosen batch samples in $D$
    for class $c \in [k]$ do
      $S_c \leftarrow \{i : \boldsymbol{y}_i = \boldsymbol{e}_c\}$
      $\boldsymbol{\mu}_c^{(t)} \leftarrow \frac{1}{|S_c|}\sum_{i\in S_c} f_\theta^{\text{feature}}(\boldsymbol{x}_i)$,          $\boldsymbol{\mu}_c^{(t)} \leftarrow (1-\beta)\boldsymbol{\mu}_c^{(t)} + \beta\boldsymbol{\mu}_c^{(t-1)}$
      $\boldsymbol{\Sigma}_c^{(t)} \leftarrow \frac{1}{|S_c|}\sum_{i\in S_c}(f_\theta^{\text{feature}}(\boldsymbol{x}_i) - \boldsymbol{\mu}_c^{(t)})(f_\theta^{\text{feature}}(\boldsymbol{x}_i) - \boldsymbol{\mu}_c^{(t)})^T$
      $\boldsymbol{\Sigma}_c^{(t)} \leftarrow \frac{1}{d}\text{trace}(\boldsymbol{\Sigma}_c^{(t)})\boldsymbol{I}_d$,          $\boldsymbol{\Sigma}_c^{(t)} \leftarrow (1-\beta)\boldsymbol{\Sigma}_c^{(t)} + \beta\boldsymbol{\Sigma}_c^{(t-1)}$
    end for
    for sample index $i \in [B]$ do
      $(\boldsymbol{x}_i^{\text{mix}}, \boldsymbol{y}_i^{\text{mix}}) \leftarrow \text{mix}((\boldsymbol{x}_i, \boldsymbol{y}_i), (\boldsymbol{x}_{\pi(i)}, \boldsymbol{y}_{\pi(i)}))$
      $p_c \leftarrow \det(\boldsymbol{\Sigma}_c^{(t)})^{-1/2}\exp\{-(f_\theta^{\text{feature}}(\boldsymbol{x}_i^{\text{mix}}) - \boldsymbol{\mu}_c^{(t)})^T(\boldsymbol{\Sigma}_c^{(t)})^{-1}(f_\theta^{\text{feature}}(\boldsymbol{x}_i^{\text{mix}}) - \boldsymbol{\mu}_c^{(t)})\}$ for $c \in [k]$
      $c_1 \leftarrow \arg\min_{c\in[k]} p_c$,          $c_2 \leftarrow \arg\min_{c\in[k]\setminus\{c_1\}} p_c$
      $\boldsymbol{y}_i^{\text{gen}} \leftarrow \frac{p_{c_1}}{p_{c_1}+p_{c_2}}\boldsymbol{e}_{c_1} + \frac{p_{c_2}}{p_{c_1}+p_{c_2}}\boldsymbol{e}_{c_2}$
    end for
    $\theta \leftarrow \theta - \eta\sum_{i\in[B]}\nabla_\theta\{\gamma\cdot\ell_{\text{CE}}(\boldsymbol{y}_i^{\text{gen}}, f_\theta(\boldsymbol{x}_i^{\text{mix}})) + (1-\gamma)\cdot\ell_{\text{CE}}(\boldsymbol{y}_i^{\text{mix}}, f_\theta(\boldsymbol{x}_i^{\text{mix}}))\}$
  end for
```

and $\boldsymbol{\Sigma}_c^{(t)} \leftarrow (1-\beta)\boldsymbol{\Sigma}_c^{(t)} + \beta\boldsymbol{\Sigma}_c^{(t-1)}$. When $\beta = 0$, it reduces to the memoryless estimation. To avoid cluttered notation, we discard t unless necessary.

In the second stage, we apply conventional mixup-based data augmentation. We first permute the batch data $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^B$ and obtain $\{(\boldsymbol{x}_{\pi(i)}, \boldsymbol{y}_{\pi(i)})\}_{i=1}^B$. Afterwards, for each $i \in [B]$, we select the data pair, $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ and $(\boldsymbol{x}_{\pi(i)}, \boldsymbol{y}_{\pi(i)})$, and apply a mixup-based data augmentation scheme denoted by $\text{mix}(\cdot)$, to generate mixed point $\boldsymbol{x}_i^{\text{mix}}$ labeled by $\boldsymbol{y}_i^{\text{mix}}$. One can use any data augmentation as $\text{mix}(\cdot)$, *e.g.*, mixup, manifold-mixup. For example, for vanilla mixup, we have

$$\boldsymbol{x}_i^{\text{mix}} = \lambda\boldsymbol{x}_i + (1-\lambda)\boldsymbol{x}_{\pi(i)}, \quad \boldsymbol{y}_i^{\text{mix}} = \lambda\boldsymbol{y}_i + (1-\lambda)\boldsymbol{y}_{\pi(i)}$$

where $\lambda \sim \text{Beta}(\alpha, \alpha)$ for some $\alpha > 0$.

In the third stage, we re-label this augmented data based on the estimated GM model parameters. To be specific, we compute the likelihood of the mixed data sampled from class c, denoted by $p_c = \det(\boldsymbol{\Sigma}_c)^{-1/2}\exp\{-(f_\theta^{\text{feature}}(\boldsymbol{x}_i^{\text{mix}}) - \boldsymbol{\mu}_c)^T\boldsymbol{\Sigma}_c^{-1}(f_\theta^{\text{feature}}(\boldsymbol{x}_i^{\text{mix}}) - \boldsymbol{\mu}_c)\}$. Then, we sort k classes in a descending order of p_c, and select the top-2 classes c_1 and c_2 satisfying $p_{c_1} \geq p_{c_2} \geq p_c$ for $c \in [k]\setminus\{c_1, c_2\}$. Then, we label the mixed point $\boldsymbol{x}_i^{\text{mix}}$ as

$$\boldsymbol{y}_i^{\text{gen}} = \frac{p_{c_1}}{p_{c_1}+p_{c_2}}\boldsymbol{e}_{c_1} + \frac{p_{c_2}}{p_{c_1}+p_{c_2}}\boldsymbol{e}_{c_2}.$$

Since our generative model is an imperfect estimate on the data distribution, $\boldsymbol{y}_i^{\text{gen}}$ may be incorrect for some samples. Thus, we can use a combination of vanilla labeling and the suggested labeling, *i.e.*, define the label of mixed point as $\gamma\boldsymbol{y}_i^{\text{gen}} + (1-\gamma)\boldsymbol{y}_i^{\text{mix}}$ for some $\gamma \in [0, 1]$. Note that our scheme reduces to the vanilla labeling scheme when $\gamma = 0$. Using the augmented data with updated label, the algorithm trains the classification model $f_\theta : \mathbb{R}^n \to [0, 1]^k$ that predicts the label $\boldsymbol{y} = [y_1, \cdots, y_k]$ of the input data, using the cross-entropy loss $\ell_{\text{CE}}(\cdot)$.

Results Table 6 shows the summary of results. Here, we tried two different validation schemes: one is to choose the best robust model against AutoAttack, and the other is to select the model with the highest clean accuracy. For each scheme $X \in \{\text{mixup}, \text{manifold-mixup}\}$, it is shown that "$X$ + GenLabel" has a minor improvement on both clean and robust accuracies, for all image datasets. It is also shown that the suggested *GenLabel* achieves a higher clean accuracy than AdaMixup, which requires 3x higher computational complexity than our method.

Table 6: Clean and robust accuracies on real image datasets. We consider two types of validation: selecting the best robust model based on AutoAttack, or the best model based on the clean accuracy. Here, Mixup+GenLabel indicates that we applied the suggested labeling method to the augmented points generated by mixup. GenLabel helps mixup-based data augmentations in terms of both robust accuracy and clean accuracy.

Methods	MNIST		CIFAR-10		CIFAR-100		TinyImageNet-200	
	Robust	Clean	Robust	Clean	Robust	Clean	Robust	Clean
Vanilla	48.17 $\pm$ 13.1	99.34 $\pm$ 0.03	16.89 $\pm$ 0.98	94.57 $\pm$ 0.25	17.19 $\pm$ 0.20	74.48 $\pm$ 0.28	13.19 $\pm$ 0.19	58.13 $\pm$ 0.09
AdaMixup	-	99.32 $\pm$ 0.05	-	95.45 $\pm$ 0.13	-	-	-	-
Mixup	55.44 $\pm$ 1.80	99.27 $\pm$ 0.03	11.65 $\pm$ 1.96	95.68 $\pm$ 0.06	18.44 $\pm$ 0.45	77.65 $\pm$ 0.30	14.91 $\pm$ 0.48	59.46 $\pm$ 0.30
Mixup+GenLabel	56.54 $\pm$ 1.03	99.36 $\pm$ 0.06	14.32 $\pm$ 1.23	**96.09** $\pm$ 0.01	**19.58** $\pm$ 0.71	78.04 $\pm$ 0.21	**15.34** $\pm$ 0.30	59.78 $\pm$ 0.09
Manifold mixup	55.56 $\pm$ 1.53	99.32 $\pm$ 0.04	18.14 $\pm$ 1.88	94.78 $\pm$ 0.49	19.25 $\pm$ 0.61	78.61 $\pm$ 0.17	14.78 $\pm$ 0.28	59.87 $\pm$ 0.63
Manifold mixup+GenLabel	**56.62** $\pm$ 1.31	**99.37** $\pm$ 0.07	**18.91** $\pm$ 1.26	95.10 $\pm$ 0.10	19.28 $\pm$ 1.04	**78.99** $\pm$ 0.54	15.19 $\pm$ 0.22	**60.02** $\pm$ 0.25

Table 7: Robust accuracy (%) under decision-based black-box attack (Brendel et al., 2018) and clean accuracy (%), on 6 selected datasets. The robust accuracy is same as what has been reported in Table 5, and we added clean accuracy in the parenthesis for comparison. One can confirm the huge gap between robust accuracy and clean accuracy, showing the effectiveness of the black-box attack used in the paper.

Methods \ OpenML ID	446	468	683	755	763	1413
Vanilla	29.67 (99.33)	34.55 (84.55)	51.11 (81.11)	41.05 (67.37)	64.27 (85.33)	68.00 (92.00)
AdaMixup	30.33 (100.00)	37.27 (89.09)	51.11 (81.11)	37.89 (71.58)	63.20 (85.87)	67.11 (92.00)
Mixup	30.67 (100.00)	37.27 (84.55)	50.00 (81.11)	36.84 (70.53)	65.07 (86.13)	67.56 (88.89)
Mixup + Excluding MI	31.67 (100.00)	31.82 (89.09)	**52.22** (83.33)	38.95 (71.58)	63.20 (85.33)	70.67 (92.89)
Mixup + GenLabel (GM)	37.00 (100.00)	**42.73** (82.73)	**52.22** (80.00)	**43.16** (73.68)	61.87 (85.33)	71.11 (93.78)
Mixup + GenLabel (NN)	**38.00** (100.00)	32.73 (81.82)	46.67 (82.22)	**43.16** (69.47)	**66.93** (85.33)	**77.33** (96.44)

A.3. Supplementary for robustness experiments

In Table 5, we showed the robust accuracies of selected datasets. Table 7 shows the full version including the robust accuracy as well as clean accuracy. This shows the effectiveness of the attack scheme used in this paper.

A.4. Visualization of GenLabel

For the Four-circle dataset and 9-class Gaussian data, we visualized the label of mixup samples and decision boundaries for *GenLabel*, mixup, and vanilla training in Fig. 4. This is a full version of Fig. 7.

A.5. GenLabel on CutMix (Yun et al., 2019)

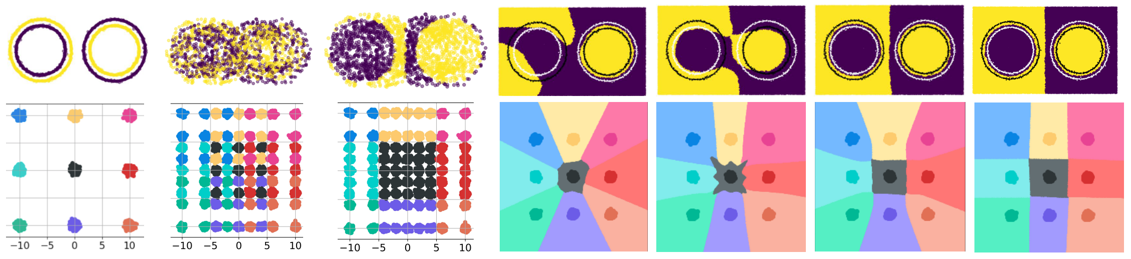
In the main manuscript, we have provided the effect of GenLabel on mixup only, but GenLabel can be also applied to other mixup variants. Now the question is, whether GenLabel is effective in other mixup variants as well? Note that the latest mixup variants (PuzzleMix, Co-Mixup, SaliencyMix) are mostly tailored for image datasets, while we are focusing on tabular datasets. Thus, we consider CutMix, the only method (among the variants listed above) that is applicable to tabular datasets. Table 8 compares the performance of mixup and mixup+GenLabel, as well as that of CutMix and CutMix+GenLabel. Our results show that CutMix+GenLabel is having gain compared with CutMix in terms of accuracy.

Table 8: Test accuracies (%) of baselines and GenLabel. For both mixup and CutMix, using GenLabel improves the accuracy.

Dataset / Scheme	Vanilla	Mixup	Mixup+GenLabel	CutMix	CutMix+GenLabel
`Iris`	95.56	88.00	96.44 (**8.44** $\uparrow$)	94.67	95.11 (**0.44** $\uparrow$)
`Wine`	92.96	94.81	96.67 (**1.86** $\uparrow$)	96.30	97.40 (**1.10** $\uparrow$)
`Kidney`	62.61	65.22	71.30 (**6.08** $\uparrow$)	63.48	69.57 (**6.09** $\uparrow$)

A.6. Effect of generative models on the performance of GenLabel

Here we empirically show how the quality of generative models affect the performance of GenLabel. Table 9 shows the **test accuracy** of vanilla/mixup/GenLabel, as well as the **train accuracy** of GenLabel using Gaussian Mixture (GM) model. One can see that GenLabel fails when the train accuracy of GM classifier is low, *i.e.*, when GM model cannot fit the data.



(a) dataset (b) mixup, top-1 label of mixed points (c) GenLabel, top-1 label of mixed points (d) vanilla training, decision boundary (e) mixup, decision boundary (f) GenLabel, decision boundary (g) kernel SVM, decision boundary

Figure 7: Comparison of vanilla training, mixup and the suggested *GenLabel*. Top: Four-circle dataset, Bottom: 9-class Gaussian data. For the purpose of illustration, we set the mixing coefficient as $\lambda = 0.6$ for the 9-class Gaussian data. (a): training data points. (b): mixup points and their top-1 labels (c): mixup points and their top-1 label for the suggested *GenLabel* $\boldsymbol{y}^{\text{gen}}$. Figures in (b) and (c) show that the conventional labeling method $\boldsymbol{y}^{\text{mix}}$ causes the label conflict issue for a large number of mixup points, while the suggested label $\boldsymbol{y}^{\text{gen}}$ does not suffer from the conflict issue. (d), (e), (f), (g): decision boundaries of vanilla training, mixup, *GenLabel*, and kernel SVM. In the decision boundary plots, we can find that several classes/samples have small margins for vanilla training and mixup, while the suggested *GenLabel* does not have such issue and approaching to the ideal margin obtained by kernel SVM.

Table 9: Test accuracies (%) of three schemes (vanilla, mixup, mixup+GenLabel) and train accuracy of Gaussian Mixture (GM) classifier, for four datasets. For `Iris` and `Wine` datasets which GM fits the data well (*i.e.*, train accuracy of GM classifier is high), GenLabel has high test accuracy. On the other hand, for `Abalone` and `Glass` datasets having small train accuracy of GM classifier, GenLabel does not improve the performance of mixup.

Dataset	Vanilla	Mixup	Mixup+GenLabel (GM)	**Train acc** of GM classifier
`Iris`	95.56	88.00	96.44 (8.44 ↑)	98.00
`Wine`	92.96	94.81	96.67 (1.86 ↑)	100.00
`Abalone`	63.01	62.42	62.39 (0.03 ↓)	62.00
`Glass`	60.00	64.92	60.31 (4.61 ↓)	64.00

B. Additional mathematical results

B.1. Formal statement on the adversarial robustness of GenLabel

Here we analyze the adversarial robustness of a model trained by mixup+*GenLabel*, and show that *GenLabel* is beneficial for improving the robustness of mixup under the logistic regression models and the fully-connected (FC) ReLU networks. We first describe the basic setting considered in our analysis, and then provide the results. All proofs are given in Section C in Appendix.

Basic setting and notations Consider d-dimensional Gaussian dataset defined as $(\boldsymbol{x}|y=0) \sim \mathcal{N}(-\boldsymbol{e}_1, \frac{\boldsymbol{\Sigma}}{\sigma_1^2})$ and $(\boldsymbol{x}|y=1) \sim \mathcal{N}(\boldsymbol{e}_1, \frac{\boldsymbol{\Sigma}}{\sigma_2^2})$, where $\Sigma_{ij} = 1$ for $i = j$ and $\Sigma_{ij} = \tau$ for $i \neq j$. Here we assume that $-1 < \tau < 1$, $\tau \notin \{\frac{-1}{d-1}, \frac{-1}{d-2}\}$ and $\sigma_2 = c\sigma_1$ with $2-\sqrt{3} < c < 2+\sqrt{3}$. We consider the loss function $\ell(\boldsymbol{\theta}, (\boldsymbol{x}, y)) = h\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})\right) - y f_{\boldsymbol{\theta}}(\boldsymbol{x})$, where $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ is the prediction of a model parameterized by θ for a given input $\boldsymbol{x}$, and $h(w) = \log(1+\exp(w))$.

We assume the following labeling setting: when we mix $(\boldsymbol{x}_i, y_i)$ and $(\boldsymbol{x}_j, y_j)$ which generates the mixed point $\boldsymbol{x}_{ij}^{\text{mix}} = \lambda \boldsymbol{x}_i + (1-\lambda)\boldsymbol{x}_j$, we label it as $y_{ij}^{\text{mix}} = y$ if $y_i = y_j = y$, and we label it as y_{ij}^{gen} in (1) otherwise. We assume the mixing coefficient follows the uniform distribution $\lambda \sim \text{Unif}[0,1]$, *i.e.*, $\lambda \sim \text{Beta}(\alpha, \alpha)$ with $\alpha = 1$.

For a given model parameter $\boldsymbol{\theta}$ and the dataset S, we define the notations for several losses as below. The standard loss is denoted by $L_n^{\text{std}}(\boldsymbol{\theta}, S) = \frac{1}{n}\sum_{i=1}^n \ell(\boldsymbol{\theta}, \boldsymbol{z}_i)$. The mixup loss and *GenLabel* loss are denoted by $L_n^{\text{mix}}(\boldsymbol{\theta}, S) = \frac{1}{n^2}\sum_{i,j=1}^n \mathbb{E}_\lambda[\ell(\boldsymbol{\theta}, \boldsymbol{z}_{ij}^{\text{mix}})]$ and $L_n^{\text{gen}}(\boldsymbol{\theta}, S) = \frac{1}{n^2}\sum_{i,j=1}^n \mathbb{E}_\lambda[\ell(\boldsymbol{\theta}, \boldsymbol{z}_{ij}^{\text{gen}})]$, respectively, where $\boldsymbol{z}_{ij}^{\text{mix}} = (\boldsymbol{x}_{ij}^{\text{mix}}, y_{ij}^{\text{mix}})$ and $\boldsymbol{z}_{ij}^{\text{gen}} = (\boldsymbol{x}_{ij}^{\text{mix}}, y_{ij}^{\text{gen}})$. The adversarial loss with L_2 attack of radius $\varepsilon\sqrt{d}$ is defined as $L_n^{\text{adv}}(\boldsymbol{\theta}, S) = \frac{1}{n}\sum_{i=1}^n \max_{\|\boldsymbol{\delta}_i\|_2 \leq \varepsilon\sqrt{d}} \ell\left(\boldsymbol{\theta}, (\boldsymbol{x}_i + \boldsymbol{\delta}_i, y_i)\right)$.

Mathematical results Before stating our result, we denote the Taylor approximation of mixup loss by $\tilde{L}_n^{\text{mix}}(\boldsymbol{\theta}, S)$, the expression of which is given in Lemma 8 in Appendix. Similarly, the Taylor approximation of each term in the adversarial loss is denoted by $\tilde{\ell}_{\text{adv}}(\varepsilon\sqrt{d}, (\boldsymbol{x_i}, y_i))$, which is expressed in Lemma 9 in Appendix. Finally, the approximation of *GenLabel* loss, denoted by $\tilde{L}_n^{\text{gen}}(\boldsymbol{\theta}, S)$, is expressed in Lemma 1 in Appendix.

In the theorem below, we state the relationship between Taylor approximations of mixup loss, *GenLabel* loss, and adversarial loss, for the logistic regression models. In this theorem, we consider the set of model parameters

$$\Theta := \{\boldsymbol{\theta} \in \mathbb{R}^d : (2y_i - 1) f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \geq 0 \text{ for all } i = 1, 2, \cdots, n\}$$

which contains the set of all $\boldsymbol{\theta}$ with zero training errors.

Theorem 1. *Consider the logistic regression setting having $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x}$. Suppose there exists a constant $c_x > 0$ such that $\|\boldsymbol{x}_i\|_2 \geq c_x$ for all $i \in \{1, 2, \cdots, n\}$. Then, in the asymptotic regime of large σ_1, for any $\boldsymbol{\theta} \in \Theta$, we have $\tilde{L}_n^{\text{mix}}(\boldsymbol{\theta}, S) > \tilde{L}_n^{\text{gen}}(\boldsymbol{\theta}, S) \geq \frac{1}{n}\sum_{i=1}^n \tilde{\ell}_{\text{adv}}(\delta_{\text{gen}}, (\boldsymbol{x_i}, y_i))$. Here, $\delta_{\text{gen}} = R \cdot c_x A^i_{\sigma_1,c,\tau,d}$ with $R = \min_{i\in\{1,\cdots,n\}} |\cos(\boldsymbol{\theta}, \boldsymbol{x_i})|$, where $A^i_{\sigma_1,c,\tau,d}$ is defined in (24) in the Appendix.*

Fig. 8 compares the second-order Taylor approximation of mixup loss $\tilde{L}_n^{\text{mix}}(\boldsymbol{\theta}, S)$, *GenLabel* loss $\tilde{L}_n^{\text{gen}}(\boldsymbol{\theta}, S)$, and adversarial loss $\frac{1}{n}\sum_{i=1}^n \tilde{\ell}_{\text{adv}}(\delta_{\text{gen}}, (\boldsymbol{x}_i, y_i))$, for the logistic regression model $\boldsymbol{\theta} = (10\cos\phi, 10\sin\phi)$ parameterized by the angle ϕ. Here, we use the dataset $S = \{(\boldsymbol{x}_i^+, +1), (\boldsymbol{x}_i^-, -1)\}_{i=1}^{20}$, where each sample at class $+1$ and -1 follows the distribution of $\boldsymbol{x}_i^+ \sim \mathcal{N}([+1, 0], \frac{1}{100}\boldsymbol{I}_2)$ and $\boldsymbol{x}_i^- \sim \mathcal{N}([-1, 0], \frac{1}{100}\boldsymbol{I}_2)$, respectively. One can confirm that the model $\boldsymbol{\theta} = (10, 0)$, which corresponds to $\phi = 0$, has the smallest mixup/*GenLabel*/adversarial loss. In every angle $\phi \in [-\frac{\pi}{4}, \frac{\pi}{4}]$, the *GenLabel* loss is strictly smaller than mixup loss, which coincides with the result of Theorem 1.

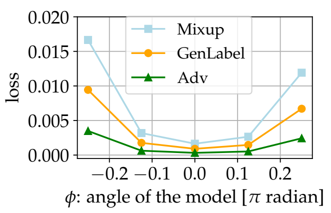


Figure 8: Comparison between the mixup loss, *GenLabel* loss and adversarial loss, for the logistic regression model $\boldsymbol{\theta} = (10\cos\phi, 10\sin\phi)$ on a two-dimensional Gaussian dataset. This plot coincides with the result in Theorem 1.

We can also extend the result of Theorem 1 to fully-connected ReLU networks as below.

Theorem 2. *Consider fully-connected ReLU network $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\beta}^T \sigma(\boldsymbol{W}_{N-1} \cdots (\boldsymbol{W}_2 \sigma(\boldsymbol{W}_1 \boldsymbol{x})))$ where σ is the activation function and the parameters contain matrices $\boldsymbol{W}_i$ and a vector $\boldsymbol{\beta}$. Suppose there exists a constant $c_x > 0$ such that $\|\boldsymbol{x}_i\|_2 \geq c_x$ for all $i \in \{1, 2, \cdots, n\}$. Then, in the asymptotic regime of large σ_1, for any $\boldsymbol{\theta} \in \Theta$, we have $\tilde{L}_n^{\text{mix}}(\boldsymbol{\theta}, S) > \tilde{L}_n^{\text{gen}}(\boldsymbol{\theta}, S) \geq \frac{1}{n}\sum_{i=1}^n \tilde{\ell}_{\text{adv}}(\delta_{\text{gen}}, (\boldsymbol{x}_i, y_i))$. Here, $\tilde{\ell}_{\text{adv}}(\delta, (\boldsymbol{x}, y)) = \ell(\boldsymbol{\theta}, (\boldsymbol{x}, y)) + \delta\, |g(f_{\boldsymbol{\theta}}(\boldsymbol{x})) - y|\, \|\nabla f_{\boldsymbol{\theta}}(\boldsymbol{x})\|_2 + \frac{\delta^2 d}{2}\, |h''(f_{\boldsymbol{\theta}}(\boldsymbol{x}))|\, \|\nabla f_{\boldsymbol{\theta}}(\boldsymbol{x})\|_2^2$ is the Taylor approximation of adversarial loss for ReLU network, and we have $\delta_{\text{gen}} = R c_x A^i_{\sigma_1,c,\tau,d}$ and $R = \min_{i\in\{1,\cdots,n\}} |\cos(\nabla f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{x}_i)|$, where $A^i_{\sigma_1,c,\tau,d}$ is in (24) and $g(x) = e^x/(1+e^x)$.*

B.2. Approximation of GenLabel loss

Below we state the approximation of GenLabel loss $L_n^{\text{gen}}(\boldsymbol{\theta}, S)$. The proof of this lemma is in Section C.6.

Lemma 1. *The second order Taylor approximation of the GenLabel loss is given by*

$$\tilde{L}_n^{\text{gen}}(\boldsymbol{\theta}, S) = L_n^{\text{std}}(\boldsymbol{\theta}, S) + R_1^{\text{gen}}(\boldsymbol{\theta}, S) + R_2^{\text{gen}}(\boldsymbol{\theta}, S) + R_3^{\text{gen}}(\boldsymbol{\theta}, S),$$

where

$$R_1^{\text{gen}}(\boldsymbol{\theta}, S) = \frac{1}{n}\sum_{i=1}^n A_{\sigma_1,c,\tau,d}(h'(f_{\boldsymbol{\theta}}(\boldsymbol{x_i})) - y_i)\nabla f_{\boldsymbol{\theta}}(\boldsymbol{x_i})^T \mathbb{E}_{\boldsymbol{r_x}\sim D_X}[\boldsymbol{r_x} - \boldsymbol{x_i}],$$

$$R_2^{\text{gen}}(\boldsymbol{\theta}, S) = \frac{1}{2n}\sum_{i=1}^n B_{\sigma_1,c,\tau,d} h''(f_{\boldsymbol{\theta}}(\boldsymbol{x_i}))\nabla f_{\boldsymbol{\theta}}(\boldsymbol{x_i})^T \mathbb{E}_{\boldsymbol{r_x}\sim D_X}[(\boldsymbol{r_x} - \boldsymbol{x_i})(\boldsymbol{r_x} - \boldsymbol{x_i})^T]\nabla f_{\boldsymbol{\theta}}(\boldsymbol{x_i}),$$

$$R_3^{\text{gen}}(\boldsymbol{\theta}, S) = \frac{1}{2n}\sum_{i=1}^n B_{\sigma_1,c,\tau,d}(h'(f_{\boldsymbol{\theta}}(\boldsymbol{x_i})) - y_i)\mathbb{E}_{\boldsymbol{r_x}\sim D_X}[(\boldsymbol{r_x} - \boldsymbol{x_i})^T \nabla^2 f_{\boldsymbol{\theta}}(\boldsymbol{x_i})(\boldsymbol{r_x} - \boldsymbol{x_i})],$$

where $A_{\sigma_1,c,\tau,d}$ and $B_{\sigma_1,c,\tau,d}$ are two constants defined in (24). When $\sigma_1 \to \infty$, we have $\lim_{\sigma_1\to\infty} A_{\sigma_1,c,\tau,d} = \frac{c^2+1}{2(c+1)^2} < \frac{1}{3}$, $\lim_{\sigma_1\to\infty} B_{\sigma_1,c,\tau,d} = \frac{c^2-c+1}{3(c+1)^2} < \frac{1}{6}$.

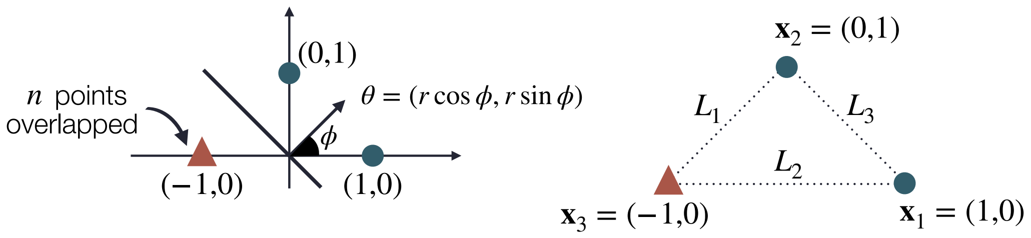


Figure 9: Left: the dataset S used in Example 3. The feature-label pairs are defined as $(\boldsymbol{x}_1, y_1) = ([1,0], +1)$, $(\boldsymbol{x}_2, y_2) = ([0,1], +1)$, and $(\boldsymbol{x}_i, y_i) = ([-1,0], -1)$ for $i = 3, 4, \cdots, n+2$. Right: the line segments connecting training data points.

C. Proof of mathematical results

C.1. Proof for Example 1

We start with showing $\theta^\star_{\text{mixup}} = \frac{7}{16}$. First, we the prediction of the classifier can be represented as

$$f_\theta(x) = \begin{cases} \frac{1}{2\theta}|x|, & \text{if } 0 \le |x| \le 2\theta, \\ 1, & \text{if } 2\theta \le |x| \le 1. \end{cases}$$

Since we have three data points, we have $\binom{3}{2} = 3$ different way of mixing the data points: (1) mixing x_1 and x_2, (2) mixing x_2 and x_3, (3) mixing x_3 and x_1. We denote the loss value of i-th mix pair as L_i. We first compute L_2, the loss of mixing x_2 and x_3. The mixed point is $x^{\text{mix}} = \lambda x_3 + (1-\lambda)x_2 = \lambda$, which has label $\boldsymbol{y}^{\text{mix}} = \lambda \boldsymbol{e}_1 + (1-\lambda)\boldsymbol{e}_2$ for $\lambda \in [0,1]$. Then,

$$\begin{aligned} L_2 &= \int_0^1 \left\| \boldsymbol{y}^{\text{mix}} - \begin{bmatrix} f_\theta(x^{\text{mix}}) \\ 1 - f_\theta(x^{\text{mix}}) \end{bmatrix} \right\|_2^2 d\lambda = 2\int_0^1 (\lambda - f_\theta(x^{\text{mix}}))^2 d\lambda = 2\int_0^{2\theta} \lambda^2 (1 - \frac{1}{2\theta})^2 \mathrm{d}\lambda + 2\int_{2\theta}^1 (\lambda - 1)^2 \mathrm{d}\lambda \\ &= \frac{2}{3}(2\theta - 1)^2. \end{aligned}$$

Since $f_\theta(x)$ is symmetric, we have $L_1 = L_2$. Now, we compute L_3. The mixed point is represented as $x^{\text{mix}} = \lambda x_3 + (1-\lambda)x_1 = 2\lambda - 1$, which is labeled as $\boldsymbol{y}^{\text{mix}} = \lambda \boldsymbol{e}_1 + (1-\lambda)\boldsymbol{e}_1 = \boldsymbol{e}_1$, for $\lambda \in [0,1]$. Then,

$$\begin{aligned} L_3 &= \int_0^1 \left\| \boldsymbol{y}^{\text{mix}} - \begin{bmatrix} f_\theta(x^{\text{mix}}) \\ 1 - f_\theta(x^{\text{mix}}) \end{bmatrix} \right\|_2^2 d\lambda = \int_0^1 (1 - f_\theta(x^{\text{mix}}))^2 d\lambda \\ &= \int_{\frac{1}{2}-\theta}^{\frac{1}{2}+\theta} (1 - \frac{1}{2\theta}|2\lambda - 1|)^2 \mathrm{d}\lambda = 2\int_{\frac{1}{2}}^{\frac{1}{2}+\theta} (1 - \frac{1}{2\theta}(2\lambda - 1))^2 \mathrm{d}\lambda = \frac{2}{3}\theta. \end{aligned}$$

Thus, $\frac{\mathrm{d}}{\mathrm{d}\theta}(\frac{3}{2}(L_1 + L_2 + L_3)) = \frac{\mathrm{d}}{\mathrm{d}\theta}(8\theta^2 - 7\theta + 2) = 0$ when $\theta = \frac{7}{16}$. This completes the proof of $\theta^\star_{\text{mixup}} = \frac{7}{16}$.

Finally, $\theta^\star_{\text{mixup-without-MI}} = \frac{1}{2}$ is trivial from the fact that $\frac{\mathrm{d}}{\mathrm{d}\theta}(\frac{3}{4}(L_1 + L_2)) = \frac{\mathrm{d}}{\mathrm{d}\theta}(2\theta - 1)^2 = 0$ when $\theta = \frac{1}{2}$.

C.2. Proof for Example 3

Consider the problem of classifying $n+2$ data points $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n+2}$, where the feature $\boldsymbol{x}_i \in \mathbb{R}^2$ and the label $y_i \in \{+1, -1\}$ of each point is specified in Fig. 9. We use the one-hot label $\boldsymbol{y}_i = [1, 0]$ for class $+1$ and $\boldsymbol{y}_i = [0, 1]$ for class -1. Consider applying logistic regression to this problem, where the solution is represented as $\boldsymbol{\theta} = [r\cos\phi, r\sin\phi]$. Here we compare three different schemes: (1) vanilla training, (2) mixup, and (3) mixup with GenLabel (dubbed as new-mixup). The first scheme is nothing but training only using the given training data S. Both mixup and new-mixup generate mixed points using linear combination of data points, *i.e.*, $\boldsymbol{x}_{ij} = \lambda \boldsymbol{x}_i + (1-\lambda)\boldsymbol{x}_j$ for some $\lambda \sim \text{Beta}(\alpha, \alpha)$, while the labeling method is different. The original mixup uses $\boldsymbol{y}_{ij} = \lambda \boldsymbol{y}_i + (1-\lambda)\boldsymbol{y}_j$, whereas the new-mixup uses $\boldsymbol{y}_{ij} = \rho \boldsymbol{y}_i + (1-\rho)\boldsymbol{y}_j$ where $\rho = \frac{1}{1+\exp\{-(\lambda - 1/2)/\sigma^2\}}$ for some small σ , assuming the class +1 is modeled as Gaussian mixture. We analyze the solutions of these schemes, denoted by $\boldsymbol{\theta}_{\text{vanilla}}$, $\boldsymbol{\theta}_{\text{mixup}}$ and $\boldsymbol{\theta}_{\text{new-mixup}}$, and compare it with the L_2 max-margin classifier obtained from support vector machine (SVM), represented as $\boldsymbol{\theta}_{\text{svm}} = (\cos\frac{\pi}{4}, \sin\frac{\pi}{4})$. Here, we denote the angle of SVM solution by $\phi_{\text{svm}} = \pi/4$. Below we first analyze the loss of vanilla training, and then provide analysis on the loss of the mixup scheme (using either original linear labeling or the suggested GenLabel).

Vanilla training Consider the vanilla training which learns $\boldsymbol{\theta}$ (or the corresponding ϕ) by only using the given data. In this case, the sum of logistic loss over all samples can be represented as

$$\ell_{\text{vanilla}} = \sum_{i=1}^{n+2} \log(1 + e^{-y_i \boldsymbol{\theta}^T \boldsymbol{x}_i}),$$

where the exponential term for each data is

$$\begin{aligned}
-y_1\boldsymbol{\theta}^T\boldsymbol{x}_1 &= -(r\cos\phi, r\sin\phi)^T(1,0) = -r\cos\phi, \\
-y_2\boldsymbol{\theta}^T\boldsymbol{x}_2 &= -(r\cos\phi, r\sin\phi)^T(0,1) = -r\sin\phi, \\
-y_i\boldsymbol{\theta}^T\boldsymbol{x}_i &= (r\cos\phi, r\sin\phi)^T(-1,0) = -r\cos\phi, \qquad i = 3,4,\cdots,n+2
\end{aligned}$$

Then, the loss of vanilla training is

$$\ell_{\text{vanilla}} = (n+1)\log(1 + e^{-r\cos\phi}) + \log(1 + e^{-r\sin\phi}). \tag{2}$$

The derivative of the loss with respect to ϕ is given as

$$R(\phi) \coloneqq \frac{d}{d\phi}\ell_{\text{vanilla}} = (n+1)\frac{r\sin\phi\exp\{-r\cos\phi\}}{1+\exp\{-r\cos\phi\}} + \frac{-r\cos\phi\exp\{-r\sin\phi\}}{1+\exp\{-r\sin\phi\}}.$$

By plugging in $\phi_{\text{svm}} = \pi/4$ in this expression, we have

$$R(\phi = \phi_{\text{svm}}) = \frac{nr}{\sqrt{2}} \cdot \frac{\exp\{-r/\sqrt{2}\}}{1+\exp\{-r/\sqrt{2}\}} = \frac{n}{\sqrt{2}} \cdot \frac{r}{1+e^{r/\sqrt{2}}} \neq 0,$$

meaning that vanilla training cannot achieve the max-margin classifier $\boldsymbol{\theta}_{\text{svm}}$ for a fixed $r > 0$. Note that $R(\phi = \phi_{\text{svm}}) \to 0$ holds when $\|\boldsymbol{\theta}\| = r \to \infty$, *i.e.*, the vanilla gradient descent training achieves the SVM solution. This coincides with the result of (Soudry et al., 2018) which showed that for linearly separable data, the model parameter $\boldsymbol{w}(t)$ updated by gradient descent satisfies both $\lim_{t\to\infty}\|\boldsymbol{\theta}(t)\| = \infty$ and $\lim_{t\to\infty}\boldsymbol{\theta}(t)/\|\boldsymbol{\theta}(t)\| = \boldsymbol{\theta}_{\text{svm}}$.

Mixup Now we analyze the case of mixup + GenLabel (or new-mixup). Here we briefly recap how the suggested data augmentation works. Basically, following the vanilla mixup scheme, we randomly sample data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and generate augmented data $\boldsymbol{x}_{ij} = \lambda\boldsymbol{x}_i + (1-\lambda)\boldsymbol{x}_j$ where $\lambda \sim \text{Beta}(\alpha,\alpha)$ for some $\alpha > 0$. Then, we label this augmented data as $\boldsymbol{y}_{ij} = \rho\boldsymbol{y}_i + (1-\rho)\boldsymbol{y}_j$ where $\rho = \lambda$ for vanilla mixup with linear labeling, and $\rho = \frac{1}{1+\exp\{-(\lambda-1/2)/\sigma^2\}}$ for new labeling, where σ is a small positive number. Since there are total $n+2$ points in the training set, we have $(n+2)^2$ pairs of $\boldsymbol{x}_i, \boldsymbol{x}_j \in X$. The sum of loss values of all pairs can be represented as

$$\ell_{\text{mixup}} = 2n\int_{L_1\cup L_2} \ell(\boldsymbol{y}_{ij}, \hat{\boldsymbol{y}}_{ij}) + n^2\ell(\boldsymbol{y}_3, \hat{\boldsymbol{y}}_3) + 2\int_{L_3} \ell(\boldsymbol{y}_{ij}, \hat{\boldsymbol{y}}_{ij}) + \ell(\boldsymbol{y}_1, \hat{\boldsymbol{y}}_1) + \ell(\boldsymbol{y}_2, \hat{\boldsymbol{y}}_2) \tag{3}$$

where the line segments L_1, L_2, L_3 are illustrated in Fig. 9. Note that each line segment can be represented as the set of following $(\boldsymbol{x}_{ij}, \boldsymbol{y}_{ij})$ pairs for $\lambda \in [0,1]$:

$$\begin{aligned}
L_1 &: \boldsymbol{x}_{ij} = (-\lambda, 1-\lambda), \quad \boldsymbol{y}_{ij} = [1-\rho, \rho] \\
L_2 &: \boldsymbol{x}_{ij} = (2\lambda-1, 0), \quad \boldsymbol{y}_{ij} = [\rho, 1-\rho] \\
L_3 &: \boldsymbol{x}_{ij} = (1-\lambda, \lambda), \quad \boldsymbol{y}_{ij} = [1, 0]
\end{aligned}$$

Recall that for a given random data $\boldsymbol{x}$, the label estimated by logistic regression model $\boldsymbol{\theta}$ is represented as $\hat{\boldsymbol{y}} = [\hat{y}^{(0)}, \hat{y}^{(1)}] = [\frac{1}{1+\exp(-\boldsymbol{\theta}^T\boldsymbol{x})}, \frac{1}{1+\exp(+\boldsymbol{\theta}^T\boldsymbol{x})}]$. If this sample has true one-hot encoded label $\boldsymbol{y} = [y^{(0)}, y^{(1)}]$, then the logistic loss of this model (regarding the specific sample $(\boldsymbol{x}, \boldsymbol{y})$) is given as

$$\ell(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -y^{(0)}\log\hat{y}^{(0)} - y^{(1)}\log\hat{y}^{(1)}$$

Thus, each loss term in (3) can be represented as

$$\begin{aligned}
\int_{L_1} \ell(\boldsymbol{y}_{ij}, \hat{\boldsymbol{y}}_{ij}) &= \int_0^1 \{(1-\rho)\log(1+e^{\lambda r\cos\phi-(1-\lambda)r\sin\phi}) + \rho\log(1+e^{-\lambda r\cos\phi+(1-\lambda)r\sin\phi})\}p(\lambda)\mathrm{d}\lambda,\\
\int_{L_2} \ell(\boldsymbol{y}_{ij}, \hat{\boldsymbol{y}}_{ij}) &= \int_0^1 \{(1-\rho)\log(1+e^{(2\lambda-1)r\cos\phi}) + \rho\log(1+e^{-(2\lambda-1)r\cos\phi})\}p(\lambda)\mathrm{d}\lambda,\\
\int_{L_3} \ell(\boldsymbol{y}_{ij}, \hat{\boldsymbol{y}}_{ij}) &= \int_0^1 \log(1+e^{-(1-\lambda)r\cos\phi-\lambda r\sin\phi}) \quad p(\lambda)\mathrm{d}\lambda,\\
\ell(\boldsymbol{y}_1, \hat{\boldsymbol{y}}_1) &= \ell(\boldsymbol{y}_3, \hat{\boldsymbol{y}}_3) = \log(1+e^{-r\cos\phi}),\\
\ell(\boldsymbol{y}_2, \hat{\boldsymbol{y}}_2) &= \log(1+e^{-r\sin\phi}),
\end{aligned}$$

where $p(\lambda)$ is the probability density function for sampling λ.

Based on the expression of the loss $\ell(r,\phi)$ for each scheme given in (2) and (3), we numerically plotted $\phi^\star = \arg\min_\phi \ell(r,\phi)$ for various r in Fig. 5. It turns out that the optimal $\phi^\star_{\text{new-mixup}}$ of new-mixup approaches to the SVM solution $\phi_{\text{svm}} = \pi/4$ as r increases. Using the standard definition of margin denoted by $\text{margin}(\boldsymbol{\theta}) = \min_{(\boldsymbol{x}_i, y_i)\in D} \frac{y_i \boldsymbol{\theta}^T \boldsymbol{x}_i}{\|\boldsymbol{\theta}\|} = \min\{\cos\phi, \sin\phi\}$, we have

$$\text{margin}(\boldsymbol{\theta}_{\text{svm}}) = \text{margin}(\boldsymbol{\theta}_{\text{new-mixup}}) > \text{margin}(\boldsymbol{\theta}_{\text{vanilla}}) > \text{margin}(\boldsymbol{\theta}_{\text{mixup}})$$

according to Fig. 5.

C.3. Proof of Theorem 1

Following the proof of Theorem 3.1 of (Zhang et al., 2021), when $\boldsymbol{\theta} \in \Theta$, we have

$$(h'(f_{\boldsymbol{\theta}}(\boldsymbol{x_i})) - y_i)\nabla f_{\boldsymbol{\theta}}(\boldsymbol{x_i})^T \mathbb{E}_{\boldsymbol{r_x}\sim D_X}[\boldsymbol{r_x} - \boldsymbol{x_i}] \geq 0,$$

$$h''(f_{\boldsymbol{\theta}}(\boldsymbol{x_i}))\nabla f_{\boldsymbol{\theta}}(\boldsymbol{x_i})^T \mathbb{E}_{\boldsymbol{r_x}\sim D_X}[(\boldsymbol{r_x} - \boldsymbol{x_i})(\boldsymbol{r_x} - \boldsymbol{x_i})^T]\nabla f_{\boldsymbol{\theta}}(\boldsymbol{x_i}) \geq 0.$$

The first inequality in Theorem 1 is directly obtained by combining Lemma 8 and the fact that $A^i_{\sigma_1,c,\tau,d} < 1/3$ and $B^i_{\sigma_1,c,\tau,d} < \frac{1}{6}$ holds, which is proven in Lemma 1. The second inequality in Theorem 1 is obtained by applying Theorem 3.1 of (Zhang et al., 2021) into the Taylor approximation of GenLabel loss $\tilde{L}^{\text{gen}}_n(\theta, S)$ in Lemma 1.

C.4. Proof of Theorem 2

Similar to the proof of Theorem 1, the first inequality is directly from Lemma 1. The second inequality is obtained by applying Theorem 3.3 of (Zhang et al., 2021) into the Taylor approximation of GenLabel loss $\tilde{L}^{\text{gen}}_n(\theta, S)$ in Lemma 1.

C.5. Lemmas used for proving Lemma 1

We here provide lemmas that are used in the proof of Lemma 1, which is given in Section C.6. Before stating our first lemma, recall that the covariance matrix of each class-conditional data distribution is a scalar factor of $\boldsymbol{\Sigma}$, which is defined as

$$\boldsymbol{\Sigma} = \begin{pmatrix} 1 & \tau & \tau & \cdots & \tau \\ \tau & 1 & \tau & \cdots & \tau \\ \vdots & \tau & \ddots & \tau & \vdots \\ \tau & \tau & \cdots & 1 & \tau \\ \tau & \tau & \cdots & \tau & 1 \end{pmatrix}. \tag{4}$$

Below we provide the inverse matrix of $\boldsymbol{\Sigma}$.

Lemma 2. *When $\tau \notin \{\frac{-1}{d-1}, \frac{-1}{d-2}\}$ and $-1 < \tau < 1$, the matrix in (4) is invertible. The inverse is given by*

$$\boldsymbol{\Sigma}^{-1} = c_d \begin{pmatrix} 1 & -\tau_d & -\tau_d & \cdots & -\tau_d \\ -\tau_d & 1 & -\tau_d & \cdots & -\tau_d \\ \vdots & -\tau_d & \ddots & -\tau_d & \vdots \\ -\tau_d & -\tau_d & \cdots & 1 & -\tau_d \\ -\tau_d & -\tau_d & \cdots & -\tau_d & 1 \end{pmatrix}, \tag{5}$$

where

$$c_d = \frac{1}{1-\tau}\frac{(d-2)\tau+1}{(d-1)\tau+1} \tag{6}$$

and

$$\tau_d = \frac{\tau}{(d-2)\tau+1}. \tag{7}$$

Proof. We prove the lemma by verifying $\boldsymbol{\Sigma} \times (5) = \boldsymbol{I}_d$.

Clearly the diagonal element in $\boldsymbol{\Sigma} \times (5)$ reads

$$\begin{aligned} c_d[1-(d-1)\tau\tau_d] &= \frac{1}{1-\tau}\frac{(d-2)\tau+1}{(d-1)\tau+1}[1-(d-1)\frac{\tau^2}{(d-2)\tau+1}] \\ &= \frac{1}{1-\tau}\frac{(d-2)\tau+1}{(d-1)\tau+1}\frac{(d-2)\tau+1-\tau^2(d-1)}{(d-2)\tau+1} \\ &= \frac{1}{1-\tau}\frac{(d-2)\tau+1-\tau^2(d-1)}{(d-1)\tau+1} = \frac{1}{1-\tau}\frac{(1-\tau)(\tau(d-1)+1)}{(d-1)\tau+1} = 1. \end{aligned}$$

The off-diagonal element in $\boldsymbol{\Sigma} \times (5)$ reads

$$c_d(\tau-\tau_d-(d-2)\tau\tau_d) = c_d[\frac{(d-2)\tau^2+\tau-\tau}{(d-2)\tau+1} - \frac{(d-2)\tau^2}{(d-2)\tau+1}] = 0.$$

Then we conclude the proof.

□

Lemma 3. *For $\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$, we have the following formula for $\boldsymbol{Z}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{Z}$:*

$$\begin{aligned} \boldsymbol{Z}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{Z} = c_d\Big[&A_1[Z_1 - B_1(\sum_{i=2}^{d} Z_i)]^2 + A_2[Z_2 - B_2(\sum_{i=3}^{d} Z_i)]^2 \\ &+ \cdots + A_{d-1}[Z_{d-1} - B_{d-1}(\sum_{i=d}^{d} Z_i)]^2 + A_d Z_d^2\Big], \end{aligned}$$

where A_n, B_n are constants that satisfy the following recurrence relation for $n \le d$

$$\begin{aligned} &A_n = A_{n-1} - B_{n-1}^2 A_{n-1}, \quad B_n = \frac{A_{n-1}B_{n-1} + B_{n-1}^2 A_{n-1}}{A_n}, \\ &A_1 = 1, \quad B_1 = \tau_d, \end{aligned} \tag{8}$$

and $\boldsymbol{Z} = [Z_1, \cdots, Z_d]$.

Proof. Using (5) we have

$$\boldsymbol{Z}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{Z} = c_d\underbrace{[\sum_{i=1}^{d} Z_i^2 - 2\tau_d \sum_{i\neq j} Z_i Z_j]}_{(*)}.$$

We focus on (*). We claim the following induction formula:

Claim: for any $n < d$, and A_n, B_n satisfying (8), we can decompose $(*)$ into

$$\begin{aligned}(*) &= A_1[Z_1 - B_1(\sum_{i=2}^{d} Z_i)]^2 + \cdots + A_n[Z_n - B_n(\sum_{i=n+1}^{d} Z_i)]^2 \\ &+ A_{n+1}\sum_{i=n+1}^{d} Z_i^2 - 2A_{n+1}B_{n+1}\sum_{i,j=n+1,i\neq j}^{d} Z_iZ_j. \end{aligned} \tag{9}$$

The lemma immediately follows by setting $n = d-1$ in the claim. Now we use induction to prove the claim.

Base case: when $n = 1$, we complete the square for Z_1 and obtain

$$\begin{aligned}(*) &= [Z_1 - \tau_d(Z_2 + \cdots + Z_d)]^2 - \tau_d^2(Z_2 + \cdots + Z_d)^2 + \sum_{i=2}^{d} Z_i^2 - 2\tau_d \sum_{i,j=2,i\neq j} Z_iZ_j \\ &= [Z_1 - \tau_d(Z_2 + \cdots + Z_d)]^2 + (1-\tau_d^2)\sum_{i=2}^{d} Z_i^2 - 2(\tau_d + \tau_d^2)\sum_{i,j=2,i\neq j} Z_iZ_j.\end{aligned}$$

We conclude the base case with A_1, B_1, A_2, B_2 satisfying (8) as:

$$A_1 = 1, \quad B_1 = \tau_d, \quad A_2 = 1 - \tau_d^2 = A_1 - B_1^2A_1,$$

$$A_2B_2 = \tau_d + \tau_d^2 = A_2\frac{A_1B_1 + B_1^2A_1}{A_2} = A_1B_1 + B_1^2A_1.$$

Induction hypothesis: we assume the claim holds true for n. We want to show the claim also holds true for $n+1$. We focus on the second line of the claim: (9). We further complete the square and have

$$\begin{aligned}(9) &:= A_{n+1}\sum_{i=n+1}^{d} Z_i^2 - 2A_{n+1}B_{n+1}\sum_{i,j=n+1,i\neq j}^{d} Z_iZ_j \\ &= A_{n+1}[Z_{n+1} - B_{n+1}\sum_{i=n+2}^{d} Z_i]^2 - A_{n+1}B_{n+1}^2(\sum_{i=n+2}^{d} Z_i)^2 \\ &+ A_{n+1}\sum_{i=n+2}^{d} Z_i^2 - 2A_{n+1}B_{n+1}\sum_{i,j=n+2,i\neq j}^{d} Z_iZ_j \\ &= A_{n+1}[Z_{n+1} - B_{n+1}\sum_{i=n+2}^{d} Z_i]^2 \\ &+ (A_{n+1} - A_{n+1}B_{n+1}^2)\sum_{i=n+2}^{d} Z_i^2 - 2(A_{n+1}B_{n+1} + A_{n+1}B_{n+1}^2)\sum_{i,j=n+2,i\neq j} Z_iZ_j \\ &= A_{n+1}[Z_{n+1} - B_{n+1}\sum_{i=n+2}^{d} Z_i]^2 + A_{n+2}\sum_{i=n+2}^{d} Z_i^2 - 2A_{n+2}B_{n+2}\sum_{i,j=n+2,i\neq j} Z_iZ_j.\end{aligned}$$

Thus the claim holds true for $n+1$ with $A_{n+2}, B_{n+2}, A_{n+1}, B_{n+1}$ satisfying (8) as

$$A_{n+2} = A_{n+1} - A_{n+1}B_{n+1}^2, \quad B_{n+2} = \frac{A_{n+1}B_{n+1} + A_{n+1}B_{n+1}^2}{A_{n+2}}.$$

Then we conclude the claim and the lemma.

□

Lemma 4. *Denote $\boldsymbol{Y} = (Y_1, \cdots, Y_j)$, and $Y_j = Z_j - B_j(\sum_{i=j+1}^{d} Z_i)$ with B_j, Z_j defined in Lemma 3 for $1 \le j \le d$, then Y_j follows a 1-D Gaussian distribution:*

$$Y_j \sim \mathcal{N}(0, \frac{1}{c_d A_j}).$$

Proof. Applying Lemma 3, we compute the cumulative density function of Y_j as

$$\begin{aligned}
&P(Y_j < x) = P(Z_j - B_j(\sum_{i=j+1}^{d} Z_i) < x) = \int_{(Z_{j+1}, Z_{j+2}, \cdots, Z_d) \in \mathbb{R}^{d-j}} \\
&\times \int_{-\infty}^{x + B_j(Z_{j+1} + \cdots + Z_d)} \int_{(Z_1, Z_2, \cdots, Z_{j-1}) \in \mathbb{R}^{j-1}} (2\pi)^{-\frac{d}{2}} \det(\boldsymbol{\Sigma})^{-\frac{1}{2}} \exp\{-\frac{1}{2} Z^T \boldsymbol{\Sigma}^{-1} Z\} \mathrm{d}\boldsymbol{Z} \\
&= \int_{(Z_{j+1}, Z_{j+2}, \cdots, Z_d) \in \mathbb{R}^{d-j}} (2\pi)^{-\frac{d-j}{2}} \exp\{-\frac{c_d}{2}[A_{j+1} Y_{j+1}^2 + \cdots + A_d Y_d^2]\} \\
&\times \int_{-\infty}^{x + B_j(Z_{j+1} + \cdots + Z_d)} (2\pi)^{-\frac{1}{2}} \exp\{-\frac{c_d}{2} A_j Y_j^2\} \\
&\times \int_{(Z_1, Z_2, \cdots, Z_{j-1}) \in \mathbb{R}^{j-1}} (2\pi)^{-\frac{j-1}{2}} \det(\boldsymbol{\Sigma})^{-\frac{1}{2}} \exp\{-\frac{c_d}{2}[A_1 Y_1^2 + \cdots + A_{j-1} Y_{j-1}^2]\} \mathrm{d}\boldsymbol{Z},
\end{aligned} \tag{10}$$

where we used $Y_d = Z_d$. Note that $Y_j = Z_j - B_j(\sum_{i=j+1}^{m} Z_i)$, we apply change of variable

$$(Z_1 - B_1(\sum_{i=2}^{d} Z_i), \cdots, Z_d) \to (Y_1, \cdots, Y_d). \tag{11}$$

The corresponding Jacobian matrix $|\frac{\partial(Y_1, \cdots, Y_d)}{\partial(Z_1, \cdots, Z_d)}|$ is an upper triangular matrix with diagonal element 1. Thus the Jacobian is 1, and we conclude

$$\begin{aligned}
(10) &= \int_{(Y_{j+1}, Y_{j+2}, \cdots, Y_d) \in \mathbb{R}^{d-j}} (2\pi)^{-\frac{d-j}{2}} \exp\{-\frac{c_d}{2}[A_{j+1} Y_{j+1}^2 + \cdots + A_d Y_d^2]\} \\
&\times \int_{-\infty}^{x} (2\pi)^{-\frac{1}{2}} \exp\{-\frac{c_d}{2} A_j Y_j^2\} \\
&\times \int_{(Y_1, Y_2, \cdots, Y_{j-1}) \in \mathbb{R}^{j-1}} (2\pi)^{-\frac{j-1}{2}} \det(\boldsymbol{\Sigma})^{-\frac{1}{2}} \exp\{-\frac{c_d}{2}[A_1 Y_1^2 + \cdots + A_{j-1} Y_{j-1}^2]\} \mathrm{d}\boldsymbol{Y} \\
&= C \times \frac{1}{(c_d A_j)^{1/2}} \times \frac{1}{2}[1 + \mathrm{erf}(\frac{x\sqrt{c_d A_j}}{\sqrt{2}})],
\end{aligned} \tag{12}$$

where C a constant that corresponds to the integration in the first and third line. Here C does not depend on x. Note that $\frac{1}{2}[1 + \mathrm{erf}(\frac{x\sqrt{c_d A_j}}{\sqrt{2}})]$ is the cdf of $\mathcal{N}(0, \frac{1}{c_d A_j})$, let $x \to \infty$, we conclude $C \times \frac{1}{(c_d A_j)^{1/2}} = 1$, thus $Y_j \sim \mathcal{N}(0, \frac{1}{c_d A_j})$.

□

Lemma 5. *Y_j and Y_k are independent for $j \neq k$, where Y_j is defined in Lemma 4.*

Proof. We prove the lemma by showing the joint cdf of Y_j, Y_k can be written as the product of cdf of Y_j and cdf of Y_k. Without loss of generality, we assume $j < k$. We focus on computing the joint cdf $P(Y_j < x, Y_k < y)$. Following the same procedure of (10), we apply the change of variable (11), then the integration becomes:

$$\begin{aligned} P(Y_j < x, Y_k < y) = & \int_{(Y_{k+1}, Y_{k+2}, \cdots, Y_d) \in \mathbb{R}^{d-k}} (2\pi)^{-\frac{d-k}{2}} \exp\{-\frac{c_d}{2}[A_{k+1}Y_{k+1}^2 + \cdots + A_d Y_d^2]\} \\ & \times \int_{-\infty}^{y} (2\pi)^{-\frac{1}{2}} \exp\{-\frac{c_d}{2} A_k Y_k^2\} \\ & \times \int_{(Y_{j+1}, Y_{j+2}, \cdots, Y_{k-1}) \in \mathbb{R}^{k-j-1}} (2\pi)^{-\frac{k-j-1}{2}} \exp\{-\frac{c_d}{2}[A_{j+1}Y_{j+1}^2 + \cdots + A_{k-1}Y_{k-1}^2]\} \\ & \times \int_{-\infty}^{x} (2\pi)^{-\frac{1}{2}} \exp\{-\frac{c_d}{2} A_j Y_j^2\} \\ & \times \int_{(Y_1, Y_2, \cdots, Y_{j-1}) \in \mathbb{R}^{j-1}} (2\pi)^{-\frac{j-1}{2}} \det(\mathbf{\Sigma})^{-\frac{1}{2}} \exp\{-\frac{c_d}{2}[A_1 Y_1^2 + \cdots + A_{j-1}Y_{j-1}^2]\} \mathrm{d}\boldsymbol{Y} \\ = & C \times \frac{1}{(c_d A_j)^{1/2}} \frac{1}{2}[1 + \mathrm{erf}(\frac{x\sqrt{c_d A_j}}{\sqrt{2}})] \times \frac{1}{(c_d A_k)^{1/2}} \frac{1}{2}[1 + \mathrm{erf}(\frac{y\sqrt{c_d A_k}}{2})]. \end{aligned}$$

Similar to (12), C is a constant that corresponds to the first, third and fifth line, and C does not depend on x, y. Note that $\frac{1}{2}[1 + \mathrm{erf}(\frac{x\sqrt{c_d A_j}}{2})]$ and $\frac{1}{2}[1 + \mathrm{erf}(\frac{y\sqrt{c_d A_k}}{2})]$ are the cdf of $\mathcal{N}(0, \frac{1}{c_d A_j})$ and $\mathcal{N}(0, \frac{1}{c_d A_k})$. Let $x, y \to \infty$, we conclude that the constant terms combine to be $C \times \frac{1}{c_d\sqrt{A_j A_k}} = 1$. Thus the joint cdf is

$$P(Y_j < x, Y_k < y) = \frac{1}{2}[1 + \mathrm{erf}(\frac{x\sqrt{c_d A_j}}{2})] \times \frac{1}{2}[1 + \mathrm{erf}(\frac{y\sqrt{c_d A_k}}{2})].$$

This equals to $P(Y_j < x) \times P(Y_k < y)$ by directly applying Lemma 4. Then we conclude the lemma.

□

Lemma 6. *Suppose $\boldsymbol{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ with $\mathbf{\Sigma}$ given by (4), then*

$$\boldsymbol{e}_1^T \mathbf{\Sigma}^{-1} \boldsymbol{Z} \sim \mathcal{N}(0, c_d), \tag{13}$$

where c_d is defined in (7). Here (13) corresponds to a 1-D Gaussian distribution.

Proof. By Lemma 2 we have

$$\boldsymbol{e}_1^T \mathbf{\Sigma}^{-1} = c_d(1, -\tau_d, -\tau_d, \cdots, -\tau_d)^T.$$

Thus

$$\boldsymbol{e}_1^T \mathbf{\Sigma}^{-1} \boldsymbol{Z} = c_d[Z_1 - \tau_d Z_2 - \tau_d Z_3 - \cdots - \tau_d Z_d].$$

Then the lemma follows by directly applying Lemma 4 with $A_1 = 1, B_1 = \tau_d$ in (8).

□

Lemma 7. *Suppose $\boldsymbol{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ with $\mathbf{\Sigma}$ given by (4), then*

$$\boldsymbol{Z}^T \mathbf{\Sigma}^{-1} \boldsymbol{Z} \textit{ has the same distribution as } \chi^2(d),$$

where $\chi^2(d)$ is the Chi-square distribution with freedom d.

Proof. Applying Lemma 3, Lemma 4 and Lemma 5 we have

$$\boldsymbol{Z}^T \mathbf{\Sigma}^{-1} \boldsymbol{Z} = \sum_{i=1}^{d} c_d A_i Y_i^2,$$

where A_i is defined in (8), Y_i is defined in Lemma 4. Here Y_i, Y_j are independent for $i \neq j$ and $c_d A_i Y_i^2 = (\sqrt{c_d A_i} Y_i)^2$. Then we apply Lemma 4 to get $(\sqrt{c_d A_i} Y_i) \sim \mathcal{N}(0, 1)$ is a standard normal distribution. Then by the definition of the Chi-square distribution we conclude the lemma.

□

C.6. Proof of Lemma 1

Proof. Denote the mixed point by $\tilde{\boldsymbol{x}}_{ij}(\lambda) = \lambda\boldsymbol{x_i} + (1-\lambda)\boldsymbol{x}_j$. In order to estimate the second order Taylor expansion of $L_n^{\text{gen}}(\boldsymbol{\theta}, S)$, we first compute the GenLabel y_{ij}^{gen}. Next we use expression of y_{ij}^{gen} to estimate $L_n^{\text{gen}}(\boldsymbol{\theta}, S)$. Then we derive the second order Taylor expansion and the correspond coefficients $A^i_{\sigma_1,c,\tau,d}$, $B^i_{\sigma_1,c,\tau,d}$. Last we consider the asymptotic limit $\sigma_1 \to \infty$.

Step 1: compute y_{ij}^{gen}.

Recall that when $y_i = y_j$, we set the label of mixed point as $y_{ij}^{\text{mix}} = y_i$. For such case, we have $y_{ij}^{\text{mix}} = \lambda_1 y_i + (1-\lambda_1)y_i = \lambda_1 y_i + (1-\lambda_1)y_j$ for any $\lambda_1 \in \mathbb{R}$. When $y_i \neq y_j$, we use the suggested GenLabel y_{ij}^{gen} in (1). Without loss of generality, we assume $\boldsymbol{x_i} \sim \mathcal{N}(-\boldsymbol{e}_1, \frac{\boldsymbol{\Sigma}}{\sigma_1^2})$ and $\boldsymbol{x}_j \sim \mathcal{N}(\boldsymbol{e}_1, \frac{\boldsymbol{\Sigma}}{\sigma_2^2})$. Thus the correspond labels are $y_i = 0, y_j = 1$. We compute the mixed point $\tilde{x}_{ij}$ as follows:

$$\begin{aligned} \tilde{\boldsymbol{x}}_{ij}(\lambda) &= \lambda\boldsymbol{x_i} + (1-\lambda)\boldsymbol{x}_j = \lambda(-\boldsymbol{e}_1 + \boldsymbol{Z}_i) + (1-\lambda)(\boldsymbol{e}_1 + \boldsymbol{Z}_j) = (1-2\lambda)\boldsymbol{e}_1 + \boldsymbol{Z}_{ij} \\ \boldsymbol{Z}_{ij} &= \lambda\boldsymbol{Z}_i + (1-\lambda)\boldsymbol{Z}_j \sim \mathcal{N}(\boldsymbol{0}, \frac{\lambda^2\boldsymbol{\Sigma}}{\sigma_1^2} + \frac{(1-\lambda)^2\boldsymbol{\Sigma}}{\sigma_2^2}). \end{aligned} \tag{14}$$

where $\boldsymbol{Z}_i = \boldsymbol{x_i} + \boldsymbol{e}_1$ and $\boldsymbol{Z}_j = \boldsymbol{x}_j - \boldsymbol{e}_1$. Now we compute the GenLabel y_{ij}^{gen} and express it as a convex combination of y_i and y_j. To compute the y_{ij}^{gen}, we denote the density function of $\mathcal{N}(-\boldsymbol{e}_1, \frac{\boldsymbol{\Sigma}}{\sigma_1^2})$ as

$$p(\boldsymbol{x}) = (2\pi)^{-\frac{d}{2}}\det(\boldsymbol{\Sigma})^{-\frac{1}{2}}\sigma_1^d e^{-\frac{\sigma_1^2}{2}(\boldsymbol{x}+\boldsymbol{e}_1)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}+\boldsymbol{e}_1)},$$

and we denote the density function of $\mathcal{N}(\boldsymbol{e}_1, \frac{\boldsymbol{\Sigma}}{\sigma_2^2})$ as

$$q(\boldsymbol{x}) = (2\pi)^{-\frac{d}{2}}\det(\boldsymbol{\Sigma})^{-\frac{1}{2}}\sigma_2^d e^{-\frac{\sigma_2^2}{2}(\boldsymbol{x}-\boldsymbol{e}_1)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{e}_1)}.$$

Then the GenLabel y_{ij}^{gen} in (1) is given by the ratio:

$$\begin{aligned} y_{ij}^{\text{gen}} &= \frac{q(\tilde{\boldsymbol{x}}_{ij}(\lambda))}{p(\tilde{\boldsymbol{x}}_{ij}(\lambda)) + q(\tilde{\boldsymbol{x}}_{ij}(\lambda))} = \frac{1}{1 + \frac{p(\tilde{\boldsymbol{x}}_{ij}(\lambda))}{q(\tilde{\boldsymbol{x}}_{ij}(\lambda))}} \\ &= \frac{1}{1 + \frac{\sigma_1^d}{\sigma_2^d}\exp\{-\frac{\sigma_1^2}{2}[(\tilde{\boldsymbol{x}}_{ij}(\lambda)+\boldsymbol{e}_1)^T\boldsymbol{\Sigma}^{-1}(\tilde{\boldsymbol{x}}_{ij}(\lambda)+\boldsymbol{e}_1) - \frac{\sigma_2^2}{\sigma_1^2}(\tilde{\boldsymbol{x}}_{ij}(\lambda)-\boldsymbol{e}_1)^T\boldsymbol{\Sigma}^{-1}(\tilde{\boldsymbol{x}}_{ij}(\lambda)-\boldsymbol{e}_1)]\}}. \end{aligned}$$

We use $\tilde{\boldsymbol{x}}_{ij}(\lambda) = (1-2\lambda)\boldsymbol{e}_1 + \boldsymbol{Z}_{ij}$ in (14) to express the exponential term in the denominator as

$$\begin{aligned} &\exp\{-\frac{\sigma_1^2}{2}[(2-2\lambda)^2\boldsymbol{e}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{e}_1 + 4(1-\lambda)\boldsymbol{e}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{Z}_{ij} + \boldsymbol{Z}_{ij}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{Z}_{ij}]\} \\ &\times \exp\{\frac{\sigma_1^2}{2}[4\lambda^2\boldsymbol{e}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{e}_1 - 4\lambda\boldsymbol{e}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{Z}_{ij} + \boldsymbol{Z}_{ij}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{Z}_{ij}]\} \\ &\times \exp\{\frac{\sigma_2^2 - \sigma_1^2}{2}[4\lambda^2\boldsymbol{e}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{e}_1 - 4\lambda\boldsymbol{e}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{Z}_{ij} + \boldsymbol{Z}_{ij}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{Z}_{ij}]\} \\ &= \exp\{-\sigma_1^2[(2-4\lambda)\boldsymbol{e}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{e}_1 + 2\boldsymbol{e}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{Z}_{ij}]\}\exp\{(\sigma_2^2 - \sigma_1^2)[2\lambda^2\boldsymbol{e}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{e}_1 - 2\lambda\boldsymbol{e}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{Z}_{ij} + \frac{\boldsymbol{Z}_{ij}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{Z}_{ij}}{2}]\}. \end{aligned}$$

Now we apply previous lemmas to estimate all terms in the exponent.

For $\boldsymbol{e}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{e}_1$, we apply Lemma 2 to $\boldsymbol{\Sigma}^{-1}$ and conclude

$$(2-4\lambda)\boldsymbol{e}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{e}_1 = (2-4\lambda)c_d,$$

where c_d is defined in (7).

For the other two terms, we define $Z'_{ij} := \boldsymbol{e}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{Z}_{ij}$ and $\bar{z}_{ij} := \boldsymbol{Z}_{ij}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{Z}_{ij}$. From (14), $\boldsymbol{Z}_{ij} = \sqrt{\frac{\lambda^2}{\sigma_1^2} + \frac{(1-\lambda)^2}{\sigma_2^2}}\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{0}, [\frac{\lambda^2}{\sigma_1^2} + \frac{(1-\lambda)^2}{\sigma_2^2}]\boldsymbol{\Sigma})$, with $\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$. Then we apply Lemma 6 to Z'_{ij} and have

$$Z'_{ij} = \boldsymbol{e}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{Z}_{ij} \sim \mathcal{N}(0, [\frac{\lambda^2}{\sigma_1^2} + \frac{(1-\lambda)^2}{\sigma_2^2}]c_d). \tag{15}$$

For $\bar{z}_{ij}$, we apply Lemma 7 and have

$$\bar{z}_{ij} = \boldsymbol{Z}_{ij}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{Z}_{ij} = [\frac{\lambda^2}{\sigma_1^2} + \frac{(1-\lambda)^2}{\sigma_2^2}] \boldsymbol{Z}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{Z} \sim [\frac{\lambda^2}{\sigma_1^2} + \frac{(1-\lambda)^2}{\sigma_2^2}] \chi^2(d) \tag{16}$$

where $\chi^2(d)$ is the Chi-square distribution with freedom d. Thus we conclude that the GenLabel reads

$$y_{ij}^{\text{gen}} = \frac{1}{1 + \frac{\sigma_1^d}{\sigma_2^d} \exp\{-\sigma_1^2[(2-4\lambda)c_d + 2Z'_{ij}]\} \exp\{(\sigma_2^2 - \sigma_1^2)[2\lambda^2 c_d - 2\lambda Z'_{ij} + \frac{\bar{z}_{ij}}{2}]\}}. \tag{17}$$

In other words, y_{ij}^{gen} can be written as a convex combination of y_i and y_j as follows:

$$\begin{aligned} y_{ij}^{\text{gen}} &= \lambda_1 y_i + (1-\lambda_1) y_j = 1 - \lambda_1 = (17), \\ \lambda_1 &= 1 - (17) = \frac{1}{1 + \frac{\sigma_2^d}{\sigma_1^d} \exp\{\sigma_1^2[(2-4\lambda)c_d + 2Z'_{ij}]\} \exp\{(\sigma_1^2 - \sigma_2^2)[2\lambda^2 c_d - 2\lambda Z'_{ij} + \frac{\bar{z}_{ij}}{2}]\}}. \end{aligned} \tag{18}$$

Step 2: estimate $L_n^{\text{gen}}(\boldsymbol{\theta}, S)$.

Now we plug the expression of y_{ij}^{gen} (18) into the GenLabel loss, we have

$$\begin{aligned} L_n^{\text{gen}}(\boldsymbol{\theta}, S) &= \frac{1}{n^2} \mathbb{E}_{\lambda \sim \text{Unif}([0,1])} \sum_{i,j=1}^n [h(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij}(\lambda))) - (\lambda_1 y_i + (1-\lambda_1) y_j)] f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij}(\lambda)) \\ &= \frac{1}{n^2} \mathbb{E}_{\lambda \sim \text{Unif}([0,1])} \sum_{i,j=1}^n \Bigg\{ \mathbb{E}_{B \sim \text{Bern}(\lambda_1)} [B[h(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij})) - y_i f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij})] \\ &\quad + (1-B)[h(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij})) - y_j f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij})]] \Bigg\}. \end{aligned} \tag{19}$$

For $\lambda \sim \text{Unif}([0,1])$, $B|\lambda \sim \text{Bern}(\lambda_1)$, we can exchange them in order and have

$$B \sim \text{Bern}(a_{ij}), \quad \lambda | B \sim \begin{cases} \mathcal{F}_{ij}^1, & B = 1; \\ \mathcal{F}_{ij}^2, & B = 0. \end{cases}$$

$$a_{ij} = \int_0^1 \lambda_1 \mathrm{d}\lambda = \int_0^1 \frac{1}{1 + \frac{\sigma_2^d}{\sigma_1^d} \exp\{\sigma_1^2[(2-4\lambda)c_d + 2Z'_{ij}]\} \exp\{(\sigma_1^2 - \sigma_2^2)[2\lambda^2 c_d - 2\lambda Z'_{ij} + \frac{\bar{z}_{ij}}{2}]\}} \mathrm{d}\lambda.$$

$\mathcal{F}_{ij}^1$ has density function

$$\mathcal{F}_{ij}^1 \sim \frac{\lambda_1}{a_{ij}} = \frac{1}{a_{ij}} \frac{1}{1 + \frac{\sigma_2^d}{\sigma_1^d} \exp\{\sigma_1^2[(2-4\lambda)c_d + 2Z'_{ij}]\} \exp\{(\sigma_1^2 - \sigma_2^2)[2\lambda^2 c_d - 2\lambda Z'_{ij} + \frac{\bar{z}_{ij}}{2}]\}}.$$

$\mathcal{F}_{ij}^2$ has density function

$$\mathcal{F}_{ij}^2 \sim \frac{1-\lambda_1}{1-a_{ij}} = \frac{1}{1-a_{ij}} \frac{1}{1 + \frac{\sigma_1^d}{\sigma_2^d} \exp\{-\sigma_1^2[(2-4\lambda)c_d + 2Z'_{ij}]\} \exp\{(\sigma_2^2 - \sigma_1^2)[2\lambda^2 c_d - 2\lambda Z'_{ij} + \frac{\bar{z}_{ij}}{2}]\}}.$$

After changing the order of λ and B in (19), we get

$$(19) = \frac{1}{n^2} \Bigg\{ \sum_{i,j=1}^n a_{ij} \mathbb{E}_{\lambda \sim \mathcal{F}_{ij}^1} [h(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij}(\lambda))) - y_i f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij}(\lambda))] \tag{20}$$

$$+ \sum_{i,j=1}^n (1-a_{ij}) \mathbb{E}_{\lambda \sim \mathcal{F}_{ij}^2} [h(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij}(\lambda))) - y_j f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij}(\lambda))] \Bigg\}. \tag{21}$$

Since $\tilde{\boldsymbol{x}}_{ij}(\lambda) = \tilde{\boldsymbol{x}}_{ji}(1-\lambda)$, we can rewrite (21) as

$$(21) = \frac{1}{n^2}\sum_{i,j=1}^{n}(1-a_{ij})\mathbb{E}_{\lambda\sim\mathcal{F}_{ij}^3}[h(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij}(\lambda))) - y_i f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij}(\lambda))]. \tag{22}$$

Here $\mathcal{F}_{ij}^3$ has density function $\mathcal{F}_{ij}^2(1-\lambda)$:

$$\mathcal{F}_{ij}^3 \sim \frac{1}{1-a_{ij}}\frac{1}{1+\frac{\sigma_1^d}{\sigma_2^d}\exp\{-\sigma_1^2[(4\lambda-2)c_d+2Z'_{ij}]\}\exp\{(\sigma_2^2-\sigma_1^2)[2(1-\lambda)^2c_d-2(1-\lambda)Z'_{ij}+\frac{\bar{z}_{ij}}{2}]\}}.$$

From (20) and (22), we denote $\mathcal{F}_{ij}$ as a mixture distribution:

$$\mathcal{F}_{ij} = a_{ij}\mathcal{F}_{ij}^1 + (1-a_{ij})\mathcal{F}_{ij}^3.$$

Then (19) reads

$$\begin{aligned}(19) &= \frac{1}{n^2}\sum_{i,j=1}^{n}\mathbb{E}_{\lambda\sim\mathcal{F}_{ij}}[h(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij}(\lambda))) - y_i f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij}(\lambda))] \\ &= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{\lambda\sim\mathcal{F}_i}\mathbb{E}_{\boldsymbol{r_x}\sim D_X}\ell_{\check{\boldsymbol{x}}_i,y_i}(\boldsymbol{\theta}). \end{aligned}\tag{23}$$

Here we defined $\mathcal{F}_i$ as a mixture distribution:

$$\mathcal{F}_i = \frac{1}{n}\sum_{j=1}^{n}\mathcal{F}_{ij}.$$

D_X is the empirical distribution induced by training samples and $\check{\boldsymbol{x}}_i = \lambda\boldsymbol{x_i} + (1-\lambda)\boldsymbol{r_x}$.

Step 3: derive the second order Taylor expansion.

Given the expression of $L_n^{\text{gen}}(\boldsymbol{\theta}, S)$ in (23), we follow the proof of Lemma 8 and conclude that the second order Taylor expansion is given by Lemma 1, with the coefficients $A_{\sigma_1,c,\tau,d}^i, B_{\sigma_1,c,\tau,d}^i$ given by

$$A_{\sigma_1,c,\tau,d}^i = \mathbb{E}_{\lambda\sim\mathcal{F}_i}[1-\lambda], \quad B_{\sigma_1,c,\tau,d}^i = \mathbb{E}_{\lambda\sim\mathcal{F}_i}[(1-\lambda)^2]. \tag{24}$$

Here $\mathcal{F}_i$ has density function

$$\begin{aligned}\frac{1}{n}\sum_{j=1}^{n}\{&\frac{1}{1+\frac{\sigma_2^d}{\sigma_1^d}\exp\{\sigma_1^2[(2-4\lambda)c_d+2Z'_{ij}]\}\exp\{(\sigma_1^2-\sigma_2^2)[2\lambda^2c_d-2\lambda Z'_{ij}+\frac{\bar{z}_{ij}}{2}]\}} \\ &+\frac{1}{1+\frac{\sigma_1^d}{\sigma_2^d}\exp\{-\sigma_1^2[(4\lambda-2)c_d+2Z'_{ij}]\}\exp\{(\sigma_2^2-\sigma_1^2)[2(1-\lambda)^2c_d-2(1-\lambda)Z'_{ij}+\frac{\bar{z}_{ij}}{2}]\}}\},\end{aligned}$$

where Z'_{ij}, $\bar{z}_{ij}$ and c_d are defined in (15), (16) and (6) respectively.

It remains to prove that when $\sigma_1\to\infty$, these coefficients satisfy the properties mentioned in Lemma 1.

Step 4: asymptotic analysis for $\sigma_1\to\infty$

Now we prove $\lim_{\sigma_1\to\infty}A_{\sigma_1,c,\tau,d}^i = \frac{c^2+1}{2(c+1)^2}$ and $\lim_{\sigma_1\to\infty}B_{\sigma_1,c,\tau,d}^i = \frac{c^2-c+1}{3(1+c)^2}$. Recall that GenLabel y_{ij}^{gen} is given in (18). When $\sigma_1\to\infty$, we have $Z'_{ij}, \bar{z}_{ij}\to 0$, then λ_1 in (18) becomes

$$\begin{aligned}\lambda_1 &= \frac{1}{1+c^d\exp\{\sigma_1^2[(2-4\lambda)]c_d+2(\sigma_1^2-c^2\sigma_1^2)\lambda^2c_d\}} \\ &= \frac{1}{1+c^d\exp\{2\sigma_1^2c_d[(1-c^2)\lambda^2-2\lambda+1]\}} \\ &= \begin{cases}\frac{1}{1+c^d\exp\{2\sigma_1^2c_d(1-c^2)(\lambda-\frac{1}{1-c})(\lambda-\frac{1}{1+c})\}}, & c\neq 1; \\ \frac{1}{1+c^d\exp\{2\sigma_1^2c_d(1-2\lambda)\}}, & c=1.\end{cases}\end{aligned}$$

We have three cases regarding c.

If $c > 1$, then $\frac{1}{1-c} < 0$, $1 - c^2 < 0$, which implies

$$(1-c^2)(\lambda - \frac{1}{1-c})(\lambda - \frac{1}{1+c}) \begin{cases} > 0, & \frac{1}{1-c} < 0 \le \lambda < \frac{1}{1+c}; \\ < 0, & \frac{1}{1+c} < \lambda \le 1. \end{cases}$$

If $0 < c < 1$, then $\frac{1}{1-c} > 1$, $1 - c^2 > 0$, which implies

$$(1-c^2)(\lambda - \frac{1}{1-c})(\lambda - \frac{1}{1+c}) \begin{cases} > 0, & 0 \le \lambda < \frac{1}{1+c}; \\ < 0, & \frac{1}{1+c} < \lambda \le 1 < \frac{1}{1-c}. \end{cases}$$

If $c = 1$, we have $1 - 2\lambda > 0$ for $0 \le \lambda < \frac{1}{2}$ and $1 - 2\lambda < 0$ for $\frac{1}{2} < \lambda \le 1$.

When $\sigma_1 \to \infty$, we combine all three cases above and conclude

$$\lambda_1 = \begin{cases} 0, & 0 \le \lambda < \frac{1}{1+c}; \\ 1, & \frac{1}{1+c} < \lambda \le 1. \end{cases} \qquad y_{ij}^{\text{gen}} = \begin{cases} y_j, & 0 \le \lambda < \frac{1}{1+c}; \\ y_i, & \frac{1}{1+c} < \lambda \le 1. \end{cases}$$

With the GenLabel given by the above equation, we compute the GenLabel loss as

$$\begin{aligned} L_n^{\text{gen}}(\boldsymbol{\theta}, S) &= \frac{1}{n^2} \mathbb{E}_{\lambda \sim \text{Unif}([0,1])} \sum_{i,j=1}^n [h(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij}(\lambda))) - y_{ij}^{\text{gen}} f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij}(\lambda))] \\ &= \frac{1}{(c+1)n^2} \mathbb{E}_{\lambda \sim \text{Unif}([0,1/(1+c)])} \sum_{i,j=1}^n \Big\{ h(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij}(\lambda))) - y_j f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij}(\lambda)) \Big\} \\ &+ \frac{c}{(c+1)n^2} \mathbb{E}_{\lambda \sim \text{Unif}([1/(1+c),1])} \sum_{i,j=1}^n \Big\{ h(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij}(\lambda))) - y_i f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij}(\lambda)) \Big\}. \end{aligned} \tag{25}$$

Since $1 - \text{Unif}([0, 1/(1+c)])$ and $\text{Unif}([1 - 1/(1+c), 1])$ are of the same distribution and $\tilde{\boldsymbol{x}}_{ij}(1-\lambda) = \tilde{\boldsymbol{x}}_{ji}(\lambda)$, we have

$$(25) = \frac{1}{(c+1)n^2} \mathbb{E}_{\lambda \sim \text{Unif}([c/(c+1),1])} \sum_{i,j=1}^n \Big\{ h(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij}(\lambda))) - y_i f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij}(\lambda)) \Big\}.$$

Using the above equation, the GenLabel loss reads

$$\begin{aligned} L_n^{\text{gen}}(\boldsymbol{\theta}, S) &= \frac{1}{n^2} \mathbb{E}_{\lambda \sim \frac{c}{c+1}\text{Unif}([1/(c+1),1]) + \frac{1}{c+1}\text{Unif}([c/(c+1),1])} \sum_{i,j=1}^n \Big\{ h(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij}(\lambda))) - y_i f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{ij}(\lambda)) \Big\} \\ &= \frac{1}{n} \sum_{i=1}^n \mathbb{E}_{\lambda \sim \frac{c}{c+1}\text{Unif}([1/(c+1),1]) + \frac{1}{c+1}\text{Unif}([c/(c+1),1])} \mathbb{E}_{\boldsymbol{r_x} \sim D_X} \ell_{\tilde{\boldsymbol{x}}_i, y_i}(\boldsymbol{\theta}). \end{aligned}$$

Following the proof of Lemma 8, we conclude that when $\sigma_1 \to \infty$, the coefficients $A^i_{\sigma_1,c,\tau,d}$, $B^i_{\sigma_1,c,\tau,d}$ are given by

$$\begin{aligned} \lim_{\sigma_1 \to \infty} A^i_{\sigma_1,c,\tau,d} =& \mathbb{E}_{\lambda \sim \frac{c}{c+1}\text{Unif}([1/(c+1),1]) + \frac{1}{c+1}\text{Unif}([c/(c+1),1])} [1 - \lambda] \\ &= \frac{c}{c+1} \int_{\frac{1}{c+1}}^1 \frac{c+1}{c} (1-\lambda) \mathrm{d}\lambda + \frac{1}{c+1} \int_{\frac{c}{c+1}}^1 \frac{c+1}{1} (1-\lambda) \mathrm{d}\lambda \\ &= \int_{\frac{1}{c+1}}^1 (1-\lambda) \mathrm{d}\lambda + \int_{\frac{c}{c+1}}^1 (1-\lambda) \mathrm{d}\lambda = \frac{c^2}{2(c+1)^2} + \frac{1}{2(c+1)^2} = \frac{c^2+1}{2(c+1)^2}. \end{aligned}$$

$$\begin{aligned} \lim_{\sigma_1 \to \infty} B^i_{\sigma_1,c,\tau,d} =& \mathbb{E}_{\lambda \sim \frac{c}{c+1}\text{Unif}([1/(c+1),1]) + \frac{1}{c+1}\text{Unif}([c/(c+1),1])} [(1 - \lambda)^2] \\ &= \int_{\frac{1}{c+1}}^1 (1-\lambda)^2 \mathrm{d}\lambda + \int_{\frac{c}{c+1}}^1 (1-\lambda)^2 \mathrm{d}\lambda = \frac{c^3}{3(c+1)^3} + \frac{1}{3(c+1)^3} = \frac{c^2-c+1}{3(c+1)^2}. \end{aligned}$$

From direct computation, we conclude that when $2-\sqrt{3} < c < 2+\sqrt{3}$,

$$\frac{c^2+1}{2(c+1)^2} < \frac{1}{3}, \quad \frac{c^2-c+1}{3(c+1)^2} < \frac{1}{6} \iff c^2-4c+1<0.$$

We conclude the lemma. $\square$

D. Mathematical results in (Zhang et al., 2021)

Lemma 8 (Lemma 3 of (Zhang et al., 2021))**.** *The second order Taylor approximation of the mixup loss $L_n^{\text{mix}}(\boldsymbol{\theta}, S)$ is given by*

$$\tilde{L}_n^{\text{mix}}(\boldsymbol{\theta}, S) = L_n^{\text{std}}(\boldsymbol{\theta}, S) + R_1^{\text{mix}}(\boldsymbol{\theta}, S) + R_2^{\text{mix}}(\boldsymbol{\theta}, S) + R_3^{\text{mix}}(\boldsymbol{\theta}, S),$$

where

$$R_1^{\text{mix}}(\boldsymbol{\theta}, S) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{3}(h'(f_{\boldsymbol{\theta}}(\boldsymbol{x_i}))-y_i)\nabla f_{\boldsymbol{\theta}}(\boldsymbol{x_i})^T\mathbb{E}_{\boldsymbol{r_x}\sim D_X}[\boldsymbol{r_x}-\boldsymbol{x_i}],$$

$$R_2^{\text{mix}}(\boldsymbol{\theta}, S) = \frac{1}{2n}\sum_{i=1}^{n}\frac{1}{6}h''(f_{\boldsymbol{\theta}}(\boldsymbol{x_i}))\nabla f_{\boldsymbol{\theta}}(\boldsymbol{x_i})^T\mathbb{E}_{\boldsymbol{r_x}\sim D_X}[(\boldsymbol{r_x}-\boldsymbol{x_i})(\boldsymbol{r_x}-\boldsymbol{x_i})^T]\nabla f_{\boldsymbol{\theta}}(\boldsymbol{x_i}),$$

$$R_3^{\text{mix}}(\boldsymbol{\theta}, S) = \frac{1}{2n}\sum_{i=1}^{n}\frac{1}{6}(h'(f_{\boldsymbol{\theta}}(\boldsymbol{x_i}))-y_i)\mathbb{E}_{\boldsymbol{r_x}\sim D_X}[(\boldsymbol{r_x}-\boldsymbol{x_i})^T\nabla^2 f_{\boldsymbol{\theta}}(\boldsymbol{x_i})(\boldsymbol{r_x}-\boldsymbol{x_i})].$$

Lemma 9 (Lemma 3.2 of (Zhang et al., 2021))**.** *Consider the logistic regression model having $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^T\boldsymbol{x}$. The second order Taylor approximation of $L_n^{\text{adv}}(\boldsymbol{\theta}, S)$ is $\frac{1}{n}\sum_{i=1}^{n}\tilde{\ell}_{\text{adv}}(\varepsilon\sqrt{d}, (\boldsymbol{x_i}, y_i))$, where for any $\eta > 0, \boldsymbol{x} \in \mathbb{R}^d$ and $y \in \{0,1\}$,*

$$\tilde{\ell}_{\text{adv}}(\eta, (\boldsymbol{x}, y)) = \ell(\boldsymbol{\theta}, (\boldsymbol{x}, y)) + \eta\left|g\left(\boldsymbol{\theta}^T\boldsymbol{x}\right)-y\right|\cdot\|\boldsymbol{\theta}\|_2 + \frac{\eta^2}{2}\cdot g\left(\boldsymbol{\theta}^T\boldsymbol{x}\right)\left(1-g\left(\boldsymbol{\theta}^T\boldsymbol{x}\right)\right)\cdot\|\boldsymbol{\theta}\|_2^2$$

and $g(x) = e^x/(1+e^x)$ is the logistic function.

E. Detailed experiments setup

Here we provide a detailed description on our experimental settings.

E.1. Synthetic datasets

Datasets The 2D cube dataset with 2 classes (class 0 and 1) is defined as follows. Consider two adjacent squares centered at $\boldsymbol{\mu}_0 = (-1,0)$ and $\boldsymbol{\mu}_1 = (1,0)$, respectively, where the length of each side of each square is 2. We define the support of class i as the area of each square. In other words, the support of class 0 is $X_0 = \{\boldsymbol{x} \in \mathbb{R}^2 : \|\boldsymbol{\mu}_0 - \boldsymbol{x}\|_\infty \leq 1\}$ where $\|\cdot\|_\infty$ is the L_∞ norm operator. Similarly, the support of class 1 is $X_1 = \{\boldsymbol{x} \in \mathbb{R}^2 : \|\boldsymbol{\mu}_1 - \boldsymbol{x}\|_\infty \leq 1\}$. The data point $\boldsymbol{x}$ for class $i \in \{0,1\}$ is uniform-randomly sampled from the square X_i.

The 3D cube dataset with 8 classes is defined as below. Consider 8 adjacent cubes, each of which is located at each octant, where the center of each cube is $\boldsymbol{\mu} = (\mu^{(1)}, \mu^{(2)}, \mu^{(3)})$ for $\mu^{(1)}, \mu^{(2)}, \mu^{(3)} \in \{-1,1\}$ and the length of each side of each cube is 2. We define the support of class i as the volume of each cube. For example, the class 0 corresponds to the cube centered at $\boldsymbol{\mu}_0 = (-1,-1,-1)$, and the support of class 0 is $X_0 = \{\boldsymbol{x} \in \mathbb{R}^3 : \|\boldsymbol{\mu}_0 - \boldsymbol{x}\|_\infty \leq 1\}$. Similarly, we define the support of class $i \in \{0,1,\cdots,7\}$. The data point $\boldsymbol{x}$ for class i is uniform-randomly sampled from the cube X_i.

The 9-class Gaussian dataset used in Fig. 4 is defined as follows. We generate 9 Gaussian clusters having the covariance matrix of $\boldsymbol{\Sigma} = \frac{1}{10}\boldsymbol{I}_2$ and centered at $\boldsymbol{\mu} = (\mu^{(1)}, \mu^{(2)})$ for $\mu^{(1)}, \mu^{(2)} \in \{-10,0,10\}$. For example, cluster 0 (or class 0) is centered at $\boldsymbol{\mu}_0 = (-10,-10)$ and cluster 8 (or class 8) is centered at $\boldsymbol{\mu}_0 = (10,10)$.

The Circle and Moon datasets used in Table 1 are from scikit-learn (Pedregosa et al., 2011) combined with Laplacian noise, where the exponential decay λ of Laplacian noise is set to 0.1 for Moon and 0.02 for Circle.

The Four-circle dataset used in Table 1 is generated as follows. We first generate a Circle dataset from scikit-learn (Pedregosa et al., 2011) combined with Laplacian noise, where the exponential decay λ of Laplacian noise is set to 0.01. Then, we generate another (second) Circle dataset under the same setting (but having different realization), shift it to the right, and flip the label of the second Circle dataset. In this way, we get two adjacent Circle datasets with flipped label.

Training setting For synthetic datasets, the hyperparameters used in our experiments are summarized in Table 10. For both 2D and 3D cube datasets, we randomly generate 20 data samples from uniform distribution for each class as training data, and evaluate the decision boundary by another 10000 randomly generated data samples for each class. For 9-class Gaussian dataset, each cluster has 5000 randomly generated samples as the training data. For Moon and Circle datasets, we randomly generate 1000 data samples for both training and testing. For Four-circle dataset, we randomly generate 1000 data samples for each Circle dataset for both training and testing. For 2D and 3D cube datasets, we use a 3-layer fully connected network, which has 64 neurons in the first hidden layer and 128 neurons in the second hidden layer. For Moon, Circle and Four-circle datasets, we use a 4-layer fully connected network, which has 64 neurons in the first hidden layer and 128 neurons in the remaining hidden layers. For all the datasets, we use the SGD optimizer and the multi-step learning rate decay. We measure the clean validation accuracy at each epoch and choose the best model having the highest clean accuracy.

Algorithms For mixup (Zhang et al., 2018), we followed the code from the official github repository: `https://github.com/facebookresearch/mixup-cifar10`. For our GenLabel scheme on 9-class Gaussian datasets, we use the ground-truth mean and identity covariance to estimate the Gaussian mixture (GM) models at the input layer.

E.2. Real datasets

Datasets We use OpenML datasets from (Vanschoren et al., 2013) , MNIST, CIFAR-10 and CIFAR-100 datasets from PyTorch (Paszke et al., 2017), and Tiny-Imagenet-200 dataset from `http://cs231n.stanford.edu/tiny-imagenet-200.zip`.

For experiments on OpenML datasets, we first accessed all datasets from Python OpenML API (Feurer et al., 2019). Afterwards, we filtered out the datasets having more than 20 features, datasets with more than 5000 data samples. We tested our GenLabel on the remaining datasets.

Training setting The hyperparameters used in our experiments are summarized in Table 11, 12, 13 and 14. When we train mixup+GenLabel on OpenML datasets, we used a 6-fold cross-validation for choosing the best loss ratio $\gamma \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. For the clean validation runs, we measured the clean validation accuracy at each epoch and choose the best model having the highest clean accuracy. For the robust validation runs, we measured the robust validation accuracy at every 5 epochs and choose the best model having the highest robust accuracy. For OpenML datasets, we tested training methods on both the logistic regression model and the neural network with 2 hidden layers. For the latter, we followed the same architecture used in mixup (Zhang et al., 2018) which has 128 neurons in each hidden layer. For MNIST and CIFAR-10 datasets, we used LeNet-5 and ResNet-18, respectively. For both CIFAR-100 and Tiny-Imagenet-200 datasets, we used PreActResNet-18. We tested on NVIDIA Tesla V100 GPUs in Amazon Web Service (AWS) and local NVIDIA RTX2080 GPU machines.

Algorithms For mixup (Zhang et al., 2018) and manifold-mixup (Verma et al., 2019), we followed the code from the official github repository: `https://github.com/facebookresearch/mixup-cifar10` and `https://github.com/vikasverma1077/manifold_mixup`. Note that the mixup github repository contains license: see `https://github.com/facebookresearch/mixup-cifar10/blob/master/LICENSE`. For adamixUp (Guo et al., 2019), we used two methods: (1) for OpenML, we implemented by ourselves in PyTorch, (2) for MNIST and CIFAR-10, we cloned the source code in `https://github.com/SITE5039/AdaMixUp` implemented in TensorFlow (Abadi et al., 2015), and made slight modifications to make their experimental settings and models consistent with ours. For our GenLabel schemes, we estimated and updated the Gaussian mixture (GM) models at the penultimate layer.

F. Generative model-based mixup algorithm (GenMix)

In Section 7.2 of the main manuscript, we suggested a new way of mixing data points using generative models. Here, we formally define the algorithm for such "generative model-based mixup", which is dubbed as *GenMix*. Our algorithm first trains a class-conditional generative model. One can use any generative models off-the-shelf, *e.g.*, Gaussian mixture models, GANs. Based on the learned class-conditional distribution $p_c(\boldsymbol{x})$'s, our algorithm augments the training dataset with data points $\boldsymbol{x}^{\text{mix}}$ that satisfy $p_{c_1}(\boldsymbol{x}^{\text{mix}}) : p_{c_2}(\boldsymbol{x}^{\text{mix}}) = (1-\lambda) : \lambda$ for arbitrary pre-defined $\lambda \in [0,1]$. It then trains a model via a standard (non-adversarial) training algorithm with the augmented dataset. The key idea behind GenMix is that such augmented data points can act as an implicit regularizer, promoting larger margins for the classification boundary of the trained model, which in turn guarantees robustness with good generalization.

Table 10: Hyperparameters and models used for clean validation in synthetic dataset experiments.

General settings	Optimizer	Momentum	Weight decay	Batch size
	SGD	0.9	0.0001	128

Datasets	Methods	Model	Training epochs	Learning rate	Loss ratio γ
2D cube	**Vanilla**	3-layer FC net	40	0.1	-
	Mixup	3-layer FC net	40	0.1	-
	Mixup+GenLabel	3-layer FC net	40	0.1	1
3D cube	**Vanilla**	3-layer FC net	40	0.1	-
	Mixup	3-layer FC net	40	0.1	-
	Mixup+GenLabel	3-layer FC net	40	0.1	0.8
Moon	**Vanilla**	4-layer FC net	100	0.1	-
	Mixup	4-layer FC net	100	0.1	-
	Mixup+GenLabel	4-layer FC net	100	0.1	1
Circle	**Vanilla**	4-layer FC net	100	0.1	-
	Mixup	4-layer FC net	100	0.1	-
	Mixup+GenLabel	4-layer FC net	100	0.1	0.8
Four-circle	**Vanilla**	4-layer FC net	100	0.1	-
	Mixup	4-layer FC net	100	0.1	-
	Mixup+GenLabel	4-layer FC net	100	0.1	1

Table 11: Hyperparameters and models used for clean validation in OpenML datasets experiments.

General settings	Training epochs	Optimizer	Weight decay	Batch size
	100	Adam	0.0001	128

Datasets	Methods	Model	Learning rate	Loss ratio γ
OpenML	**Baselines**	Logistic Regression	Chosen by cross-validation (among 0.1, 0.01, 0.001, and 0.0001)	-
	Mixup+GenLabel	Logistic Regression	Chosen by cross-validation (among 0.1, 0.01, 0.001, and 0.0001)	Chosen by cross-validation

The rest of this section is organized as follows. We first provide a formal description of the GenMix framework. Then, we propose two specific instances of our framework, namely, GenMix+GM and GenMix+GAN, which use Gaussian mixture (GM) and GANs for generative modeling, respectively.

F.1. General framework

Let $D_c = \{(\boldsymbol{x}_c^{(m)}, \boldsymbol{e}_c)\}_{m=1}^{n_c}$ be the training data for class $y \in [k]$, where $\boldsymbol{x}_c^{(m)}$ is the feature vector for m-th data point, $\boldsymbol{e}_c$ is the one-hot encoded label vector for any data points in class c, and n_c is the number of data points with class y. The training data is denoted by $D = \cup_{c\in[k]} D_c$. In the first stage, it trains class-conditional generative model using the given training data $X_c = \{\boldsymbol{x}_c^{(m)}\}_{m=1}^{n_c}$, thereby learning the underlying data distribution $p_c(\boldsymbol{x})$.

In the second stage, we randomly sample mixing coefficient $\lambda \in [0, 1]$. For each class pair $i, j \in [k]$, we generate augmented points $X_{\text{mixup}} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{n_{\text{aug}}}\}$, each of which satisfies $p_i(\boldsymbol{x}^{\text{mix}}) : p_j(\boldsymbol{x}^{\text{mix}}) = (1-\lambda) : \lambda$. In other words, the goal is to find virtual data $\boldsymbol{x}$'s which satisfy

$$\left| \frac{p_j(\boldsymbol{x})}{p_i(\boldsymbol{x}) + p_j(\boldsymbol{x})} - \lambda \right| \leq \varepsilon \tag{26}$$

for a pre-defined small margin $\varepsilon > 0$. Depending on the generative model used in the algorithm, we use different methods to find these mixup points X_{mixup}. The detailed description of these methods are given in the following subsections. In both schemes, we check whether the generated mixup points X_{mixup} incur manifold intrusion (Guo et al., 2019), and discard the mixup points having such issues. To be specific, for the case of mixing class i and j, we decide that the manifold intrusion does not occur for a mixup point $\boldsymbol{x} \in X_{\text{mixup}}$ if classes i and j are the two most probable classes of $\boldsymbol{x}$, *i.e.*, $\min\{p_i(\boldsymbol{x}), p_j(\boldsymbol{x})\} \geq p_\ell(\boldsymbol{x})$ holds for all other classes $\ell \in [k]\backslash\{i, j\}$. For augmented data $\boldsymbol{x}$ without such manifold intrusion issue, we soft-label it as $\boldsymbol{y} = \frac{p_i}{p_i+p_j}\boldsymbol{e}_i + \frac{p_j}{p_i+p_j}\boldsymbol{e}_j$ where $p_i = p_i(\boldsymbol{x})$ is the probability that $\boldsymbol{x}$ is sampled from class i. We denote the set of data-label pair as $D_{\text{mixup}} = \{(\boldsymbol{x}, \boldsymbol{y})\}$ for $\boldsymbol{x} \in X_{\text{mixup}}$.

Given n_{aug} data points obtained in the second stage, the algorithm finally trains the classification model $f : \mathbb{R}^n \to [0, 1]^k$ that predicts the label $\boldsymbol{y} = [y_1, \cdots, y_k]$ of the input data. Here, the cross-entropy loss is used while optimizing the model. In our GenMix scheme, the model is trained by using not only the given training data $D = \cup_{c\in[k]}\{D_c\}$, but also the augmented dataset D_{mixup}. The pseudocode of the GenMix algorithm is given in Algorithm 4.

Table 12: Hyperparameters and models used for robust validation in OpenML dataset experiments.

General settings	Training epochs	Optimizer	Momentum	Weight decay	Batch size	Black-box attack radius
	100	SGD	0.9	0.0001	128	0.1 * (max value - min value)

Datasets	Methods	Model	Learning rate	Loss ratio γ
OpenML	**Baselines**	FC ReLU networks with 2 hidden layers	Chosen by cross-validation (among 0.1, 0.01, 0.001, and 0.0001)	-
	Mixup+GenLabel	FC ReLU networks with 2 hidden layers	Chosen by cross-validation (among 0.1, 0.01, 0.001, and 0.0001)	Chosen by cross-validation

Table 13: Hyperparameters and models used for clean validation in image dataset experiments.

General settings	Training epochs	Learning rate scheduler	Optimizer	Momentum	Weight decay	Batch size
	200	multi-step decay	SGD	0.9	0.0001	128

Datasets	Methods	Model	Learning rate	Attack radius	Loss ratio γ	Memory ratio β
MNIST	**Vanilla**	LeNet-5	0.1	0.05	-	-
	AdaMixup	LeNet-5	0.1	0.05	-	-
	Mixup	LeNet-5	0.1	0.05	-	-
	Mixup+GenLabel	LeNet-5	0.1	0.05	0.15	0.95
	Manifold mixup	LeNet-5	0.1	0.05	-	-
	Manifold mixup+GenLabel	LeNet-5	0.1	0.05	0.15	0.99
CIFAR-10	**Vanilla**	ResNet-18	0.1	2/255	-	-
	AdaMixup	ResNet-18	0.1	2/255	-	-
	Mixup	ResNet-18	0.1	2/255	-	-
	Mixup+GenLabel	ResNet-18	0.1	2/255	0.1	0.95
	Manifold mixup	ResNet-18	0.1	2/255	-	-
	Manifold mixup+GenLabel	ResNet-18	0.1	2/255	0.1	0.95
CIFAR-100	**Vanilla**	PreAct ResNet-18	0.1	1/255	-	-
	Mixup	PreAct ResNet-18	0.1	1/255	-	-
	Mixup+GenLabel	PreAct ResNet-18	0.1	1/255	0.1	0.97
	Manifold mixup	PreAct ResNet-18	0.1	1/255	-	-
	Manifold mixup+GenLabel	PreAct ResNet-18	0.1	1/255	0.1	0.97
Tiny ImageNet	**Vanilla**	PreAct ResNet-18	0.1	1/255	-	-
	Mixup	PreAct ResNet-18	0.1	1/255	-	-
	Mixup+GenLabel	PreAct ResNet-18	0.1	1/255	0.05	0.995
	Manifold mixup	PreAct ResNet-18	0.1	1/255	-	-
	Manifold mixup+GenLabel	PreAct ResNet-18	0.1	1/255	0.05	0.995

Algorithm 4 GenMix

```
Input Training data $D$, Number of augmented data $n_{\text{aug}}$, likelihood-ratio margin $\varepsilon > 0$, mixing coefficient $\lambda \in [0,1]$
Output Trained model $f(\cdot)$, Augmented data $D_{\text{mixup}}$
  $p_c \leftarrow$ Data distribution of class $c$ learned by generative model
  $D_{\text{mixup}} \leftarrow \{\}$
  for classes $i \in [k]$ and $j \in [k] \backslash \{i\}$ do
    $n \leftarrow 0$
    while $n < n_{\text{aug}}$ do
      Find point $\boldsymbol{x}$ satisfying $\left|\frac{p_j(\boldsymbol{x})}{p_i(\boldsymbol{x})+p_j(\boldsymbol{x})} - \lambda\right| \leq \varepsilon$
      $p_\ell \leftarrow p_\ell(\boldsymbol{x})$ for $\ell \in [k]$
      if $\min\{p_i, p_j\} \geq p_\ell \quad \forall \ell \in [k] \backslash \{i,j\}$ then
        $D_{\text{mixup}} \leftarrow D_{\text{mixup}} \cup \{(\boldsymbol{x}, \frac{p_i}{p_i+p_j}\boldsymbol{e}_i + \frac{p_j}{p_i+p_j}\boldsymbol{e}_j)\}$
        $n \leftarrow n + 1$
      end if
    end while
  end for
  $f \leftarrow$ model training with $D \cup D_{\text{mixup}}$
```

In summary, the proposed scheme is a novel data augmentation technique that first learns the data distributions for each class using class-conditional generative models, and then augments the train data with soft-labeled data points X_{mixup}, each of which has the likelihood ratio of $\lambda \in [0,1]$ with respect to a target class pair.

Table 14: Hyperparameters and models used for AutoAttack validation in image dataset experiments.

General settings	Training epochs	Learning rate scheduler	Optimizer	Momentum	Weight decay	Batch size
	50	multi-step decay	SGD	0.9	0.0001	128

Datasets	Methods	Model	Learning rate	Attack radius	Loss ratio γ	Memory ratio β
MNIST	**Vanilla**	LeNet-5	0.001	0.1	-	-
	AdaMixup	LeNet-5	0.001	0.1	-	-
	Mixup	LeNet-5	0.001	0.1	-	-
	Mixup+GenLabel	LeNet-5	0.001	0.1	0.15	0.97
	Manifold mixup	LeNet-5	0.001	0.1	-	-
	Manifold mixup+GenLabel	LeNet-5	0.001	0.1	0.15	0.97
CIFAR-10	**Vanilla**	ResNet-18	0.001	2/255	-	-
	AdaMixup	ResNet-18	0.001	2/255	-	-
	Mixup	ResNet-18	0.001	2/255	-	-
	Mixup+GenLabel	ResNet-18	0.001	2/255	0.15	0.9
	Manifold mixup	ResNet-18	0.001	2/255	-	-
	Manifold mixup+GenLabel	ResNet-18	0.001	2/255	0.15	0.9
CIFAR-100	**Vanilla**	PreAct ResNet-18	0.001	1/255	-	-
	Mixup	PreAct ResNet-18	0.001	1/255	-	-
	Mixup+GenLabel	PreAct ResNet-18	0.001	1/255	0.15	0.97
	Manifold mixup	PreAct ResNet-18	0.001	1/255	-	-
	Manifold mixup+GenLabel	PreAct ResNet-18	0.001	1/255	0.15	0.97
Tiny ImageNet	**Vanilla**	PreAct ResNet-18	0.002	1/255	-	-
	Mixup	PreAct ResNet-18	0.002	1/255	-	-
	Mixup+GenLabel	PreAct ResNet-18	0.002	1/255	0.15	0.995
	Manifold mixup	PreAct ResNet-18	0.002	1/255	-	-
	Manifold mixup+GenLabel	PreAct ResNet-18	0.002	1/255	0.15	0.995

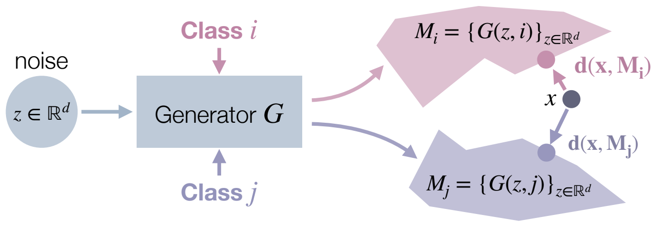


Figure 10: Finding the augmented point $\boldsymbol{x} \in X_{\text{mixup}}$ satisfying $d(\boldsymbol{x}, \mathcal{M}_j) - d(\boldsymbol{x}, \mathcal{M}_i) \simeq \log(\frac{1}{\lambda} - 1)$ in GenMix+GAN, for arbitrary classes $i \neq j$ and a pre-defined mixing coefficient $\lambda \in [0,1]$. Given the manifold $\mathcal{M}_c = \{G(\boldsymbol{z}, c)\}_{\boldsymbol{z}\in\mathbb{R}^d}$ for class $c \in [k]$ estimated by class-conditional GAN, the distance $d(\boldsymbol{x}, \mathcal{M}_c)$ is measured by inverting the generator of GAN (Creswell & Bharath, 2018).

F.2. GenMix+GM

We first suggest GenMix+GM, a data augmentation scheme which uses the Gaussian mixture (GM) model for generative modeling. Here, we provide a formal description on how GenMix+GM finds the augmented points $\boldsymbol{x}$ satisfying the likelihood ratio condition (26). Given training samples, GenMix+GM algorithm first estimates the parameters of Gaussian distribution for each class. To be specific, it computes the sample mean and the sample covariance of class c, represented as $\widehat{\boldsymbol{\mu}_c} = \frac{1}{n_c}\sum_{m=1}^{n_c} \boldsymbol{x}_c^{(m)}$ and $\widehat{\boldsymbol{\Sigma}_c} = \frac{1}{n_c}\sum_{m=1}^{n_c}(\boldsymbol{x}_c^{(m)} - \widehat{\boldsymbol{\mu}_c})(\boldsymbol{x}_c^{(m)} - \widehat{\boldsymbol{\mu}_c})^T$, respectively. Then, the (estimated) probability of point $\boldsymbol{x}$ sampled from class c is $p_c(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^k \det(\widehat{\boldsymbol{\Sigma}_c})}} e^{-(\boldsymbol{x}-\widehat{\boldsymbol{\mu}_c})^T \widehat{\boldsymbol{\Sigma}_c}^{-1}(\boldsymbol{x}-\widehat{\boldsymbol{\mu}_c})/2}$. Now, the question is how to find the virtual data points $\boldsymbol{x}$ satisfying (26). This can be solved by applying quadratic discriminant analysis (QDA) (Ghojogh & Crowley, 2019), which gives us the closed-form solution for $\boldsymbol{x}$ satisfying $|\log \frac{p_j(\boldsymbol{x})}{p_i(\boldsymbol{x})+p_j(\boldsymbol{x})}| \simeq \lambda$, for given target classes i, j.

F.3. GenMix+GAN

The Gaussian mixture (GM) model is a simple generative model that works well when the data distribution is similar to Gaussian, but it cannot learn other distributions. In such cases, GANs are useful for learning the underlying distribution. Thus, here we suggest GenMix+GAN which uses GANs for generative modeling. As discussed in Section 7.1, we can replace $p_c(\boldsymbol{x})$ by $\exp(-d(\boldsymbol{x}, \mathcal{M}_c))$ in Algorithm 4 and apply GenMix scheme. Note that the condition in (26) reduces to $d(\boldsymbol{x}, \mathcal{M}_j) - d(\boldsymbol{x}, \mathcal{M}_i) \simeq \log(\frac{1}{\lambda} - 1)$. Thus, the goal is to solve $\min_{\boldsymbol{x}}(d(\boldsymbol{x}, \mathcal{M}_j) - d(\boldsymbol{x}, \mathcal{M}_i) - \log(\frac{1}{\lambda} - 1))^2$.

We use an iterative method to find points $\boldsymbol{x}$ that satisfy this condition. One key observation that helps us to design an efficient optimization algorithm is that if $\boldsymbol{m}^\star = \arg\min_{\boldsymbol{m}\in\mathcal{M}} d(\boldsymbol{x}, \boldsymbol{m})$, then $d(\boldsymbol{x}+\boldsymbol{\delta}, \mathcal{M}) \approx d(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{m}^\star)$ if $\boldsymbol{\delta}$ is small.

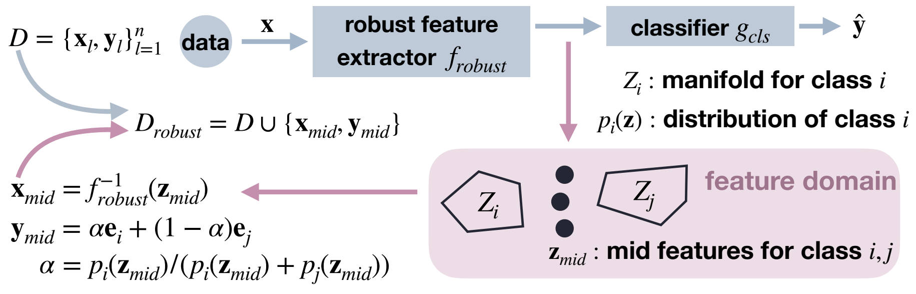

$D = \{\mathbf{x}_l, \mathbf{y}_l\}_{l=1}^n$
data
$\mathbf{x}$
robust feature extractor f_{robust}
classifier g_{cls}
$\hat{\mathbf{y}}$
Z_i : manifold for class i
$p_i(\mathbf{z})$: distribution of class i
$D_{robust} = D \cup \{\mathbf{x}_{mid}, \mathbf{y}_{mid}\}$
$\mathbf{x}_{mid} = f_{robust}^{-1}(\mathbf{z}_{mid})$
$\mathbf{y}_{mid} = \alpha \mathbf{e}_i + (1-\alpha)\mathbf{e}_j$
$\alpha = p_i(\mathbf{z}_{mid})/(p_i(\mathbf{z}_{mid}) + p_j(\mathbf{z}_{mid}))$
feature domain
Z_i
Z_j
$\mathbf{z}_{mid}$: mid features for class i,j

Figure 11: How to generate augmented dataset D_{robust} by applying GenMix in the hidden feature space. For target classes i, j, we first apply GenMix+GM in the feature space to find the mid hidden feature $\boldsymbol{z}_{\text{mid}}$, and then invert it back to get mid input feature $\boldsymbol{x}_{\text{mid}} = f_{\text{robust}}^{-1}(\boldsymbol{z}_{\text{mid}})$. Finally, we add mid input features in the original dataset D to obtain D_{robust}. Here, we make use of the invertibility of f_{robust} suggested in (Engstrom et al., 2019).

That is, once we have a projection of $\boldsymbol{x}$ onto a manifold $\mathcal{M}$, say $\boldsymbol{m}^\star$, the distance between $\boldsymbol{x}+\boldsymbol{\delta}$ and the same manifold can be safely approximated by the distance between $\boldsymbol{x}+\boldsymbol{\delta}$ and $\boldsymbol{m}^\star$, without recomputing the projection.

To formally prove this, from triangle inequality,

$$\begin{aligned} d(\boldsymbol{x}+\boldsymbol{\delta}, \mathcal{M}) &= \min_{\boldsymbol{m}\in\mathcal{M}} d(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{m}) \\ &\leq \min_{\boldsymbol{m}\in\mathcal{M}} [d(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{x}) + d(\boldsymbol{x}, \boldsymbol{m})] = \min_{\boldsymbol{m}\in\mathcal{M}} d(\boldsymbol{x}, \boldsymbol{m}) + d(\boldsymbol{x}, \boldsymbol{x}+\boldsymbol{\delta}) \\ &= d(\boldsymbol{x}, \boldsymbol{m}^\star) + d(\boldsymbol{x}, \boldsymbol{x}+\boldsymbol{\delta}) \leq d(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{m}^\star) + 2d(\boldsymbol{x}, \boldsymbol{x}+\boldsymbol{\delta}) \end{aligned}$$

holds. Similarly, we have $d(\boldsymbol{x}+\boldsymbol{\delta}, \mathcal{M}) \geq d(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{m}^\star) - 2d(\boldsymbol{x}, \boldsymbol{x}+\boldsymbol{\delta})$. This implies that when $d(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{m}^\star) \gg d(\boldsymbol{x}, \boldsymbol{x}+\boldsymbol{\delta})$, we have $d(\boldsymbol{x}+\boldsymbol{\delta}, \mathcal{M}) \approx d(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{m}^\star)$.

Using this approximation, we propose the following sequential optimization algorithm, as illustrated in Fig. 10. Starting from a random initial point $\boldsymbol{x} \in \mathbb{R}^n$, we first compute its projection on k class-conditional manifolds, finding $\boldsymbol{m}_c^\star = \arg\min_{\boldsymbol{m}\in\mathcal{M}_c} d(\boldsymbol{x}, \boldsymbol{m})$ for each $c \in [k]$. Each of these projections can be approximately computed by solving a respective optimization problem $\min_{\boldsymbol{z}\in\mathbb{R}^d} d(\boldsymbol{x}, G(\boldsymbol{z}, c))$. Now, we select two target classes i, j which are closest to the initial point, *i.e.*, $d(\boldsymbol{x}, \boldsymbol{m}_i^\star) \leq d(\boldsymbol{x}, \boldsymbol{m}_j^\star) \leq d(\boldsymbol{x}, \boldsymbol{m}_l^\star)$ for all $l \in [k] \setminus \{i, j\}$, and consider the following optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{\delta}} |d(\boldsymbol{x}+\boldsymbol{\delta}, \mathcal{M}_j) - d(\boldsymbol{x}+\boldsymbol{\delta}, \mathcal{M}_i) - \log(\frac{1}{\lambda} - 1)|^2 \\ &\text{such that } d(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{m}_c^\star) \gg d(\boldsymbol{x}, \boldsymbol{x}+\boldsymbol{\delta}), \quad c \in \{i, j\} \end{aligned}$$

That is, we find the best direction $\boldsymbol{\delta}$ that minimizes the objective function, within a small set around $\boldsymbol{x}$. By the aforementioned approximation, the target function can be rewritten as $|d(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{m}_j^\star) - d(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{m}_i^\star) - \log(\frac{1}{\lambda} - 1)|^2$. Since $\boldsymbol{m}_i^\star$ and $\boldsymbol{m}_j^\star$ are given, we can compute the gradient of this objective function with respect to $\boldsymbol{\delta}$ and run a gradient descent algorithm. The solution to this sub-optimization problem is now defined as $\boldsymbol{x}$, and we repeat the whole procedure until $|\frac{1}{1+\exp(d(\boldsymbol{x}, \boldsymbol{m}_j^\star) - d(\boldsymbol{x}, \boldsymbol{m}_i^\star))} - \lambda| \leq \varepsilon$, and obtain the augmented data point $\boldsymbol{x}$. We label this augmented data as $\boldsymbol{y} = \frac{\exp(-d_i)}{\exp(-d_i)+\exp(-d_j)}\boldsymbol{e}_i + \frac{\exp(-d_j)}{\exp(-d_i)+\exp(-d_j)}\boldsymbol{e}_j$ where $d_i = d(\boldsymbol{x}, \boldsymbol{m}_i^\star)$.

F.4. GenMix in the hidden feature space

As illustrated in Fig. 11, the suggested GenMix can be also defined in the hidden feature space. Below we describe the details of using GenMix in the hidden space.

Let f_{robust} be the robust feature extractor suggested in (Engstrom et al., 2019). Note that this feature extractor is approximately *invertible*, *i.e.*, the input data $\boldsymbol{x}$ can be well estimated by the representation $\boldsymbol{z} = f_{\text{robust}}(\boldsymbol{x})$ in the feature space. We first apply GenMix in the feature space to find the middle features $\boldsymbol{z}_{\text{mid}}$ satisfying $p_i(\boldsymbol{z}_{\text{mid}}) \simeq p_j(\boldsymbol{z}_{\text{mid}})$ for target classes i, j. Then, using the invertibility of f_{robust}, we compute $\boldsymbol{x}_{\text{mid}} = f_{\text{robust}}^{-1}(\boldsymbol{z}_{\text{mid}}) = \arg\min_{\boldsymbol{x}} \|\boldsymbol{z}_{\text{mid}} - f_{\text{robust}}(\boldsymbol{x})\|$. Afterwards, we define the augmented dataset as $D_{\text{robust}} = D \cup \{(\boldsymbol{x}_{\text{mid}}, \boldsymbol{y}_{\text{mid}})\}$, where $\boldsymbol{y}_{\text{mid}} = \alpha\boldsymbol{e}_i + (1-\alpha)\boldsymbol{e}_j$ for $\alpha = \frac{p_i(\boldsymbol{z}_{\text{mid}})}{p_i(\boldsymbol{z}_{\text{mid}})+p_j(\boldsymbol{z}_{\text{mid}})}$.

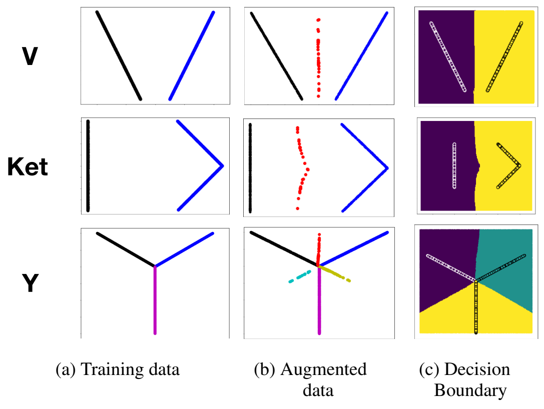


Figure 12: Result of GenMix+GAN on V, Ket and Y datasets. (a): Training data (black: X_1, blue: X_2, magenta: X_3), (b) Augmented data X_{mixup} including middle points (red, yellow, cyan), (c): Decision boundaries (DBs) of GenMix+GAN. The region with same color represents the set of points classified as the same class.

F.5. Experimental results on GenMix

We evaluate the generalization and robustness performances of GenMix+GAN, GenMix+GM and existing algorithms. We tested on synthetic datasets (Circle, Moon in scikit-learn (Pedregosa et al., 2011) and V, Ket, Y datasets designed by us) and a real dataset (MNIST with digits 7 and 9). The V, Ket, Y-datasets are illustrated in Fig. 12a. We compare our schemes with mixup (Zhang et al., 2018) and manifold-mixup (Verma et al., 2019).

F.5.1. GenMix enjoys large margins

Fig. 12 shows the result of GenMix+GAN for three synthetic datasets. Here, we set the mixing coefficient as $\lambda = 0.5$, so that GenMix generates mixup data that are equiprobable to target classes. One can confirm that the equiprobable points help the trained model to enjoy large margins in all datasets.

In Fig. 13, we visualize the suggested mixup points and the model trained by the suggested data augmentation on various synthetic datasets, and compare them with those found by vanilla mixup. Here, we set the mixing coefficient $\lambda = 0.5$, meaning that the suggested mixup points are equally probable to be sampled by two target classes.

First, we show the result for 2D Gaussian dataset with 4 classes, where each data in class c is sampled from a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_c, \Sigma_c)$. Trivially, Gaussian mixture (GM) model fits well with this data, so we use GM to estimate $p_c(\boldsymbol{x})$ in this dataset. The middle points $\boldsymbol{x}^{\text{mix}}$ generated by the suggested mixup are illustrated in (a). Note that the mid points lie on the equiprobable regime for each class pair. Here, the suggested mixup learns to *not* mix class-1 data (red) and class-2 data (blue), since mixing these classes incur manifold intrusion. In (b) and (c), we show the decision boundary found by suggested mixup and vanilla mixup. One can see that the suggested mixup, which makes use of the underlying distribution to generate proper middle points, achieves large margins for all classes. On the other hand, the standard mixup interpolates samples without considering the overall data distribution, resulting in smaller margins around the class-0 data.

Second, we show the result for circle and moon datasets defined in (Pedregosa et al., 2011). Since the Gaussian mixture model is not suitable for these datasets, we use GANs to estimate the underlying distribution $p_c(\boldsymbol{x})$. As described in the discussion section for applying GenLabel to "implicit density", we inverted GAN and used the projected distance as a proxy to the negative log likelihood. In (a) of circle and moon datasets, the mixed points satisfying $p_0(\boldsymbol{x}^{\text{mix}}) = p_1(\boldsymbol{x}^{\text{mix}})$ are colored as red, which are indeed at the middle of two manifolds of black and blue. Using these mixed points, the decision boundary has a larger margin compared with vanilla mixup, as shown in (b) and (c).

Note that in Fig. 13 we used L_2 norm for generating middle points in Moon dataset, but we can also generate middle points for L_1 or L_∞ norms. Fig. 14 illustrates the mixup points generated for Moon dataset, when L_1, L_2, and L_∞ distance metrics are used. Here, we set the mixing coefficient as $\lambda = 0.5$, i.e., the goal is to find equidistant points to target manifolds. From the figures, we can conclude that GenMix+GAN successfully finds the points that are equidistant to both manifolds, for various L_p distance settings.

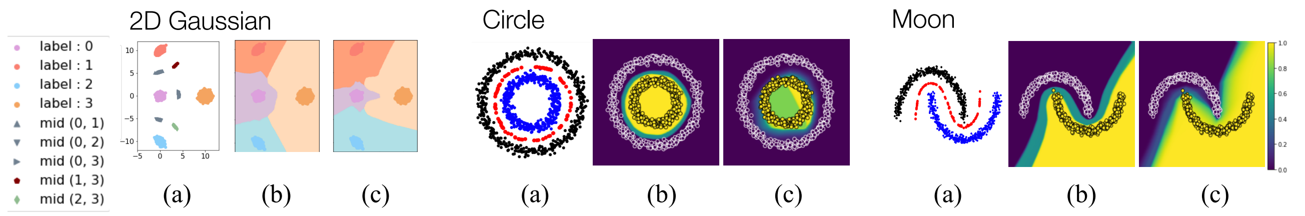


Figure 13: Comparison between generative-model based mixup (suggested mixup) and vanilla mixup for 2D datasets. For 2D Gaussian data, class 0,1,2,3 are colored as violet, red, blue, and yellow, respectively. For circle and moon datasets, class 0 and 1 are colored as black and blue, respectively, and the middle point obtained in the suggested mixup is colored as red. For each dataset, we show three results: (a) the mid points generated by the suggested mixup, and the decision boundaries of (b) suggested mixup and (c) vanilla mixup. In (a), one can confirm that the mid points of the suggested mixup lie on the equiprobable regime for the target class pair. As shown in (b) and (c), the suggested mixup enjoys larger margins for all classes than vanilla mixup.

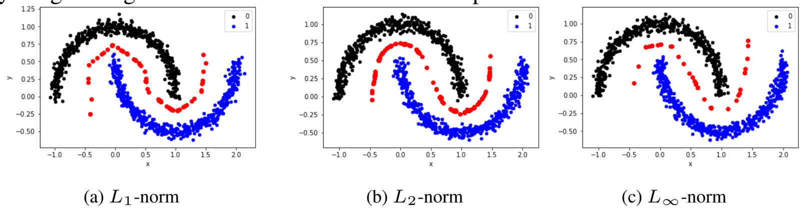


Figure 14: Illustration of data points generated by GenMix+GAN in various L_p norm setup. Black and blue points correspond to each class. The red points represent the mixup points generated by Algorithm 4.

F.5.2. GenMix helps generalization

Here we compare GenMix with mixup and manifold-mixup in terms of generalization performance. Table 15 compares the performance for circle and MNIST datasets. For MNIST, we used binary classification of digits 7 and 9 using only $n_{\text{train}} = 500$ samples at each class, to show the scenarios with large gap between GenMix and existing schemes. One can confirm that GenMix+GAN strictly outperforms the other data augmentation schemes in terms of generalization performances. This shows that depending on how we generate middle points (*i.e.*, how we mix data), generalization performance varies significantly. One can confirm that GenMix outperforms conventional ways of mixing data, by making use of the underlying data distribution learned by generative models.

F.5.3. GenMix in the hidden feature space

Recall that in Section F.4, we have suggested GenMix in the hidden feature space. Fig. 15 shows the result of GenMix+GM applied for the hidden feature space, tested on CIFAR-10 dataset. Note that each generated image contains the features of both classes c_1, c_2 written in the caption, showing that the mid features obtained by the suggested mixup indeed lies in between the target class manifolds.

F.6. Reducing the computational complexity of GenMix+GAN

Here we discuss methods for reducing the complexity of GenMix+GAN, which used inverting the generator of GAN. We can reduce the complexity of inverting the generator of GAN, by using alternative GAN architectures that simultaneously learn the inverse mapping during training, *e.g.*, bidirectional GAN (Donahue et al., 2017) and ALIGAN (Dumoulin et al., 2017). One can also consider using flow-based generative models, *e.g.*, (Kingma & Dhariwal, 2018).

Table 15: Classification errors (%). GenMix+GAN has a better generalization performance than other schemes.

Schemes / Datasets	Circle (2D)	Circle (3D)	MNIST 7/9 (n_{train}=500)
Vanilla Training	8.60 $\pm$ 4.84	1.40 $\pm$ 0.54	2.72 $\pm$ 0.20
Mixup	7.98 $\pm$ 2.94	5.22 $\pm$ 1.99	2.32 $\pm$ 0.40
Manifold-mixup	7.34 $\pm$ 1.43	0.94 $\pm$ 0.75	3.88 $\pm$ 0.53
GenMix+GAN	**4.90** $\pm$ 0.12	**0.22** $\pm$ 0.06	**2.13** $\pm$ 0.12

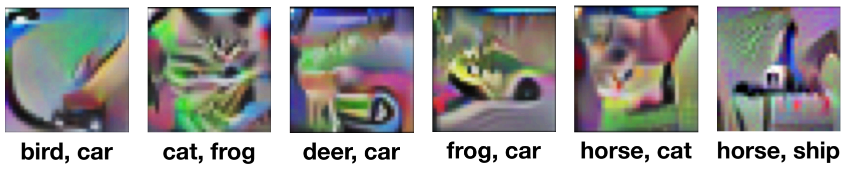


Figure 15: Generative model-based mixup in the hidden feature space for CIFAR-10. Each image $\boldsymbol{x}^{\text{mix}}$ contains features of both classes c_1, c_2 in the caption, supporting that it lies in between the manifold of target classes.