# MIXUP TRAINING AS THE COMPLEXITY REDUCTION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Machine learning models often suffer from the problem of over-fitting. Many data augmentation methods have been proposed to tackle such a problem, and one of them is called mixup. Mixup is a recently proposed regularization procedure, which linearly interpolates a random pair of training examples. This regularization method works very well experimentally, but its theoretical guarantee is not adequately discussed. In this study, we aim to discover why mixup works well from the aspect of the statistical learning theory. In addition, we reveal how the effect of mixup changes in each situation. Furthermore, we also investigated the effects of changes in the parameter of mixup. Our work contributes to searching for the optimal parameters and estimating the effects of the parameters currently used. The results of this study provide a theoretical clarification of when and how effective regularization by mixup is.

## 1 INTRODUCTION

Machine learning has achieved remarkable results in recent years due to the increase in the number of data and the development of computational resources (Michie et al., 1994; Bishop, 2006; Goldberg, 2017; Deng et al., 2009; Everingham et al., 2010). However, despite such excellent performance, machine learning models often suffer from the problem of over-fitting (Hawkins, 2004; Lawrence & Giles, 2000; Dietterich, 1995). In recent years, a concept called mixup (Zhang et al., 2018) or BC-Learning (Tokozume et al., 2018b) has attracted attention as one of the powerful regularization methods for machine learning models. The main idea of these regularization methods is to prepare $(\tilde{\boldsymbol{x}}_{ij}, \tilde{y}_{ij}) = (\lambda \boldsymbol{x}_i + (1-\lambda)\boldsymbol{x}_j, \lambda y_i + (1-\lambda)y_j)$ mixed with random pairs $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ of input vectors and their corresponding labels $(y_i, y_j)$ and use them as training data. This regularization method is very powerful and has been applied in various fields such as image recognition (Tokozume et al., 2018a; Inoue, 2018) or speech recognition (Medennikov et al., 2018; Xu et al., 2018). Despite these strong experimental results, there is not enough discussion about why this method works well.

In this paper, we give theoretical guarantees for regularization by mixup and reveal how regularization changes in each setting. Our main idea is that there must be some different quantities before and after the regularization. We focus on the Rademacher complexity and the smoothness of the convex function characterizing the Bregman divergence, which are measures of model richness, as such a quantity. In other words, the model's complexity should change with the mixup regularization, and by observing how these changes, we can theoretically clarify the effects of mixup. Furthermore, we also investigated the effects of changes in mixup's parameter $\lambda$. This contributes to searching for the optimal parameters and estimating the effects of the parameters currently used.

To summarize our results, mixup regularization leads to the following effects:

- For linear classifiers, the effect of regularization is higher when the sample size is small, and the sample standard deviation is large.
- For neural networks, the effect of regularization is higher when the number of samples is small, and the training dataset contains outliers.
- When the parameter $\lambda$ is close to 0 or 1, mixup can reduce the variance of the estimator, but this will be affected by bias.
- When the parameter $\lambda$ has near the optimal value, mixup can reduce both the bias and variance of the estimator.

- Geometrically, mixup reduces the second-order derivative of the convex function that characterizes the Bregman divergence.

## 2 Related Works

### 2.1 Mixup Variants

Mixup is originaly proposed by (Xu et al., 2018). The main idea of these regularization methods is to prepare $(\tilde{\boldsymbol{x}}_{ij}, \tilde{y}_{ij}) = (\lambda \boldsymbol{x}_i + (1-\lambda)\boldsymbol{x}_j, \lambda y_i + (1-\lambda)y_j)$ mixed with random pairs $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ of input vectors and their corresponding labels $(y_i, y_j)$ and use them as training data, where $\lambda \sim Beta(\alpha, \alpha)$, for $\alpha \in (0, \infty)$.

Because of its power and ease of implementation, several variants have been studied. Verma et al. (2019) proposed the Manifold mixup, which is a method to mix up the output of an intermediate layer of neural networks (including the input layer) instead of the input data. Berthelot et al. (2019) proposed MixMatch, a heuristic method that combines the ideas of mixup and semi-supervised learning. Puzzle Mix (Kim et al., 2020) leverages the saliency information while respecting the underlying local statistics of the data, and Nonlinear Mixup Guo (2020) relaxes the constraint of convex combination in mixup. There are several other variants, but many are heuristic methods and have insufficient theoretical explanations (Yun et al., 2019; Lim et al., 2019; Sohn et al., 2020).

On the other hand, there are several theoretical analyses of the effects of mixup. Archambault et al. (2019) suggested that mixup training is connected to adversarial training.

Carratino et al. (2020) have further shown that mixup amounts to empirical risk minimization on modified points plus multiple regularization terms through a Taylor approximation.

## 3 Notations and Preliminaries

We consider a binary classification problem in this paper. However, our analysis can easily be applied to a multi-class case.

Let $\mathscr{X}$ be the input space, $\mathscr{Y} = \{-1, +1\}$ be the output space, and $\mathscr{C}$ be a set of concepts we may wish to learn, called concept class. We assume that each input vector $\boldsymbol{x} \in \mathbb{R}^d$ is of dimension $d$. We also assume that examples are independently and identically distributed (i.i.d) according to some fixed but unknown distribution $D$.

Then, the learning problem is formulated as follows: we consider a fixed set of possible concepts $H$, called hypothesis set. We receive a sample $B = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ drawn i.i.d. according to $D$ as well as the labels $(c(\boldsymbol{x}_1), \ldots, c(\boldsymbol{x}_n))$, which are based on a specific target concept $c \in \mathscr{C} : \mathscr{X} \mapsto \mathscr{Y}$. Our task is to use the labeled sample $B$ to find a hypothesis $h_B \in H$ that has a small generalization error with respect to the concept $c$. The generalization error $\mathscr{R}(h)$ is defined as follows.

**Definition 1.** (Generalization error) Given a hypothesis $h \in H$, a target concept $c \in \mathscr{C}$, and unknown distribution $D$, the generalization error of $h$ is defined by

$$\mathscr{R}(h) = \mathbb{E}_{x \sim D}\left[\mathbb{1}_{h(\boldsymbol{x}) \neq c(\boldsymbol{x})}\right], \tag{1}$$

where $\mathbb{1}_\omega$ is the indicator function of the event $\omega$.

The generalization error of a hypothesis $h$ is not directly accessible since both the underlying distribution $D$ and the target concept $c$ are unknown Then, we have to measure the empirical error of hypothesis $h$ on the observable labeled sample $B$. The empirical error $\hat{\mathscr{R}}(h)$ is defined as follows.

**Definition 2.** (Empirical error) Given a hypothesis $h \in H$, a target concept $c \in \mathscr{C}$, and a sample $B = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, the empirical error of $h$ is defined by

$$\hat{\mathscr{R}}(h) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{h(\boldsymbol{x}_i) \neq c(\boldsymbol{x}_i)}. \tag{2}$$

In learning problems, we are interested in how much difference there is between empirical and generalization errors. Therefore, in general, we consider the relative generalization error $\hat{\mathscr{R}}(h) -$

$\mathscr{R}(h)$. The Rademacher complexity and the learning bound using it can be used to provide useful information about the relative generalization error.

**Definition 3.** (Empirical Rademacher complexity) Given a hypothesis set $H$ and a sample $B = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, the empirical Rademacher complexity of $H$ is defined as:

$$\hat{\mathfrak{R}}_B(H) = \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{h \in H} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(\boldsymbol{x}_i)\right], \tag{3}$$

where $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)^T$ with Rademacher variables $\sigma_i \in \{-1, +1\}$ which are independent uniform random variables.

**Definition 4.** (Rademacher complexity) Let $D$ denote the distribution according to which samples are drawn. For any sample size $n \geq 1$, the Rademacher complexity of $H$ is the expectation of the empirical Rademacher complexity over all samples of size $n$ drawn according to $D$:

$$\mathfrak{R}_n(H) = \mathbb{E}_{B \sim D^n}\left[\hat{\mathfrak{R}}_B(H)\right]. \tag{4}$$

Intuitively, this discribes the richeness of hypothesis class $H$.

The Rademacher complexity is a very useful tool for investigating hypothesis class $H$. By the following theorem, we can quantify the relative generalization error.

**Theorem 1.** Given a hypothesis $h \in H$ and the distribution $D$ over the input space $\mathscr{X}$, we assume that $\hat{\mathfrak{R}}_B(H)$ is the empirical Rademacher complexity of the hypothesis class $H$. Then, for any $\delta > 0$, with probability at least $1 - \delta$ over a sample $B$ of size $n$ drawn according to $D$, each of the following holds over $H$ uniformly:

$$\mathscr{R}(h) - \hat{\mathscr{R}}(h) \quad \leq \quad \hat{\mathfrak{R}}_n(H) + \sqrt{\frac{\log\frac{1}{\delta}}{2m}}, \tag{5}$$

$$\mathscr{R}(h) - \hat{\mathscr{R}}(h) \quad \leq \quad \hat{\mathfrak{R}}_B(H) + 3\sqrt{\frac{\log\frac{2}{\delta}}{2m}}. \tag{6}$$

For a proof of this theorem, see Appendix A. This theorem provides a generalization bound based on the Rademacher complexity. We can see that this bound is data-dependent due to the fact that empirical Rademacher complexity $\hat{\mathfrak{R}}_B(H)$ is a function of the specific sample $B$.

From the above discussion, we can see that if we can quantify the change of empirical Rademacher complexity before and after mixup, we can evaluate the relative generalization error of the hypothesis class $H$. Our main idea is to clarify the effects of the mixup regularization by examining how these Rademacher complexity changes before and after regularization. Note that we are not interested in the tightness of the bound, but only in the difference in the bound.

## 4 Complexity Reduction of Linear Classifiers with Mixup

In this section, we assume that $H_\ell$ is a class of linear functions:

$$h(\boldsymbol{x}) \in H_\ell = \left\{\boldsymbol{x} \mapsto \boldsymbol{w}^T \boldsymbol{x} \;\middle|\; \boldsymbol{w} \in \mathbb{R}^d,\ \|\boldsymbol{w}\|_2 \leq \Lambda\right\}, \tag{7}$$

where $\boldsymbol{w}$ is the weight vector and $\Lambda$ is a constant that regularizes the L2 norm of the weight vector.

The following theorem provides a relaxation of the Rademacher complexity of the linear classifier by mixup.

**Theorem 2.** Given a hypothesis set $H_\ell$ and a sample $B = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, we assume that $\hat{\mathfrak{R}}_B(H_\ell)$ is the empirical Rademacher complexity of the hypothesis class $H_\ell$ and $\hat{\mathfrak{R}}^*_B(H_\ell)$ is the empirical Rademacher complexity of $H_\ell$ when mixup is applied. The difference between the two Rademacher complexity $\hat{\mathfrak{R}}_B(H_\ell) - \hat{\mathfrak{R}}^*_B(H_\ell)$ is less than or equal to a constant multiple of the sample variance of the norm of the input vectors:

$$\hat{\mathfrak{R}}_B(H_\ell) - \hat{\mathfrak{R}}^*_B(H_\ell) \leq \frac{C^\Lambda_\lambda}{\sqrt{n}} \sqrt{s^2 \|\boldsymbol{x}\|_2}, \tag{8}$$

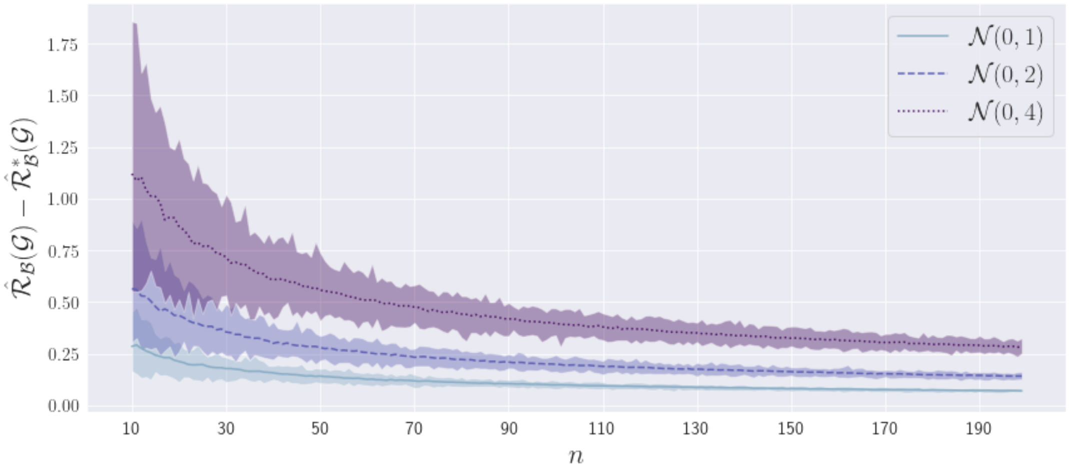

Figure 1: The relationship between $\hat{\mathfrak{R}}_B(H_\ell) - \hat{\mathfrak{R}}^*_B(H_\ell)$ and the number of samples $n$ and variance $\sigma^2$ when mixup is applied. Each data point was sampled from the normal distribution $\mathscr{N}(0,\sigma^2)$ and the constant part was set to 1. It can be seen that as the number of samples n increases, the effect of complexity mitigation by mixup decreases. We also find that the greater the variance in the distribution of the data, the higher the effect of mixup.

where $C^\Lambda_\lambda$ is a constant that depends on the parameter $\lambda$ of mixup and $s^2$ is the sample variance computed from the sample set.

As can be seen from the equation 8, the complexity relaxation by mixup decreases as the number of samples $n$ increases (this can be seen by taking the right-hand side of the theorem to the limit for $n$, see Figre 1).

*Proof.* Let $\tilde{\boldsymbol{x}}_i = \mathbb{E}_{\boldsymbol{x}_j}[\lambda \boldsymbol{x}_i + (1-\lambda)\boldsymbol{x}_j]$ be the expectation of the linear combination of input vectors by mixup, where $\lambda$ is a parameter in mixup and is responsible for adjusting the weights of the two vectors. Then, we have

$$\begin{aligned} \hat{\mathfrak{R}}_B(H_\ell) - \hat{\mathfrak{R}}^*_B(H_\ell) &\leq \frac{\Lambda}{n}\left(\sum_{i=1}^{n} \|\boldsymbol{x}_i\|_2^2\right)^{\frac{1}{2}} - \frac{\Lambda}{n}\left(\sum_{i=1}^{n} \left\| \mathbb{E}_{x_j}\left[\lambda \boldsymbol{x}_i + (1-\lambda)\boldsymbol{x}_j\right]\right\|_2^2\right)^{\frac{1}{2}} && (9) \\ &= \frac{\Lambda|1-\lambda|}{\sqrt{n}}\sqrt{s^2(\|\boldsymbol{x}\|_2)} \geq 0, && (10) \end{aligned}$$

Here, let $C^\Lambda_\lambda = \Lambda|1-\lambda|$ and we can obtain equation 8. □

For a complete proof, see Appendix B.

The above results are in line with our intuition and illustrate well how mixup depends on the shape of the data distribution. In the next section, we discuss neural networks as a more general application destination for the mixup.

## 5 Complexity Reduction of neural networks with mixup

Let $H_{L,\boldsymbol{W}_L}$ be the function class of a neural network:

$$h(\boldsymbol{x}) \in H_{L,\boldsymbol{W}_L} = \left\{h : \|\boldsymbol{v}\|_2 = 1, \prod_{i=1}^{L} \|\boldsymbol{W}_i\|_F \leq \boldsymbol{W}_L\right\}, \tag{11}$$

where $L$ is the number of layers, $\boldsymbol{W}_i$ is the weight matrix, $\boldsymbol{v} \in \mathbb{R}^{M_L}$ represents the normalized linear classifier operating on the output of the neural networks with input vector $\boldsymbol{x}$ and $\|\boldsymbol{A}\|_F$ is the

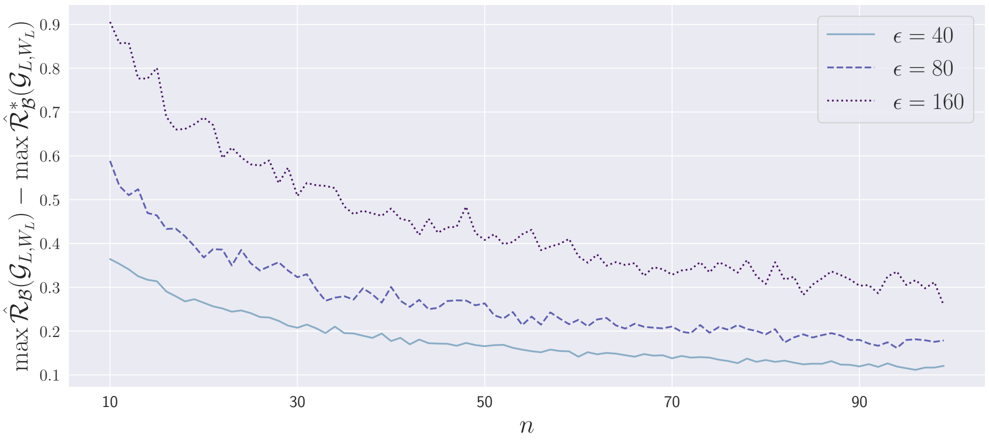

Figure 2: The relationship between $\hat{\mathfrak{R}}_B(H_{L,\boldsymbol{W}_L}) - \hat{\mathfrak{R}}^*_B(H_{L,\boldsymbol{W}_L})$ and the number of samples $n$ and the noise of the outliers $\boldsymbol{\epsilon}$. When there are extreme outliers in the sample, we can see that mixup allows the neural network to make robust estimation. In addition, we can see that the effect of regularization decreases as the sample size $n$ increases.

Frobenius norm of the matrix $\boldsymbol{A} = (a_{ij})$.

$$\|\boldsymbol{A}\|_F = \sqrt{\sum_{ij} a_{ij}^2}.$$

The following theorem provides relaxation of the Rademacher complexity of the neural network by mixup.

**Theorem 3.** Given a hypothesis set $H_{L,\boldsymbol{W}_L}$ and a sample $B = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$, we assume that $\hat{\mathfrak{R}}_B(H_{L,\boldsymbol{W}_L})$ is the empirical Rademacher complexity of the hypothesis class $H_{L,\boldsymbol{W}_L}$ and $\hat{\mathfrak{R}}^*_B(H_{L,\boldsymbol{W}_L})$ is the empirical Rademacher complexity of $H_{L,\boldsymbol{W}_L}$ when mixup is applied. In addition, we assume that each sample $\boldsymbol{x}_i$ occurs with the population mean $\boldsymbol{\mu_x}$ plus the some noise $\boldsymbol{\epsilon}_i$. In other words, we assume that $\boldsymbol{x}_i = \boldsymbol{\mu_x} + \boldsymbol{\epsilon}_i$. The difference between the maximum of two Rademacher complexity $\hat{\mathfrak{R}}_B(H_{L,\boldsymbol{W}_L}) - \hat{\mathfrak{R}}^*_B(H_{L,\boldsymbol{W}_L})$ is less than or equal to a constant multiple of the maximum value of noise in a sample of training data when the number of samples $n$ is sufficiently large:

$$\max \hat{\mathfrak{R}}_B(H_{L,\boldsymbol{W}_L}) - \max \hat{\mathfrak{R}}^*_B(H_{L,\boldsymbol{W}_L}) \leq \frac{C_\lambda^L}{\sqrt{n}} \max_i \|\boldsymbol{\epsilon}_i\|, \tag{12}$$

where $C_\lambda^L$ is a constant that depends on the parameter $\lambda$ of mixup and the number of layers $L$ of neural networks.

This theorem shows that mixup regularization for neural networks is more effective when there are outliers in the sample.

*Proof.* The upper bound of the Rademacher complexity of the neural network with ReLU as the activation function and regularization by the constant $\boldsymbol{W}_L$ for the norm of each weight is bounded as follows (Neyshabur et al., 2015):

$$\hat{\mathfrak{R}}_B(H_{L,\boldsymbol{W}_L}) \leq \frac{1}{\sqrt{n}} 2^{L+\frac{1}{2}} \boldsymbol{W}_L \max_i \|\boldsymbol{x}_i\|. \tag{13}$$

Rademacher complexity of $H_{L,\boldsymbol{W}_L}$ with mixup is

$$\begin{aligned} \hat{\mathfrak{R}}^*_B(H_{L,\boldsymbol{W}_L}) &\leq \frac{1}{\sqrt{n}} 2^{L+\frac{1}{2}} \boldsymbol{W}_L \max_i \|\mathbb{E}_j[\lambda \boldsymbol{x}_i + (1-\lambda)\boldsymbol{x}_j]\| \\ &\leq \frac{1}{\sqrt{n}} 2^{L+\frac{1}{2}} \boldsymbol{W}_L \max_i \left\{ \lambda \|\boldsymbol{x}_i\| + (1-\lambda)\|\mathbb{E}_j[\boldsymbol{x}_j]\| \right\}. \end{aligned} \tag{14}$$

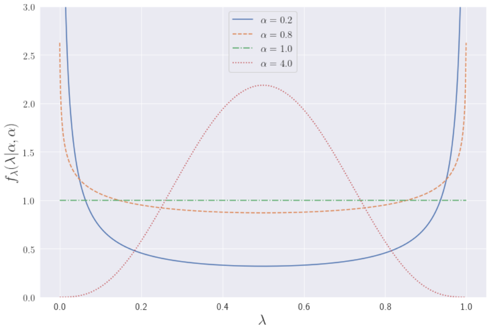

Figure 3: Beta distribution $Beta(\alpha, \alpha)$ for each $\alpha$. Here, the probability density function of the Beta distribution is $f(x;\alpha,\beta) = \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}$, where $B(\alpha,\beta)$ is the beta function. From this figure, it can be seen that when $\alpha = 1$, it is equivalent to a uniform distribution, and when $\alpha > 1$, it becomes bell-shaped. we can also see that when $\alpha < 1$, sampled $\lambda$ is close to 0 or 1.

Then,

$$\begin{aligned}
\max \hat{\mathfrak{R}}_B(H_{L,\boldsymbol{W}_L}) - \max \hat{\mathfrak{R}}^*_B(H_{L,\boldsymbol{W}_L}) &\leq \frac{1}{\sqrt{n}} 2^{L+\frac{1}{2}} \boldsymbol{W}_L \max_i \left\{ \|\boldsymbol{x}_i\| - (\lambda \|\boldsymbol{x}_i\| + (1-\lambda)\|\mathbb{E}_j[\boldsymbol{x}_j]\|\right\} \\
&\leq \frac{1-\lambda}{\sqrt{n}} 2^{L+\frac{1}{2}} \boldsymbol{W}_L \max_i \left\{ \|\boldsymbol{\mu}_{\boldsymbol{x}}\| + \|\boldsymbol{\epsilon}_i\| - \|\bar{\boldsymbol{x}}\| \right\} && (15) \\
&= \frac{1-\lambda}{\sqrt{n}} 2^{L+\frac{1}{2}} \boldsymbol{W}_L \max_i \|\boldsymbol{\epsilon}_i\|. && (16)
\end{aligned}$$

The inequality in equation 15 is guaranteed by the subadditivity nature of the norm, and the equality in equation 16 is guaranteed by the law of large numbers. Here, let $C^L_\lambda = (1-\lambda)2^{L+\frac{1}{2}}\boldsymbol{W}_L$ and we can obtain equation 12. □

For a complete proof, see Appendix B.

While neural networks have wealthy representational power due to their ability to approximate complicated functions, they are prone to over-fitting into training samples. In other words, it approximates a function that fits well for unusual examples that occur accidentally in the training sample $b$. According to the above theorem, mixup allows the neural networks robust learning for outliers with accidentally large noise $\boldsymbol{\epsilon}$ in the training sample $B$.

## 6 The Optimal Parameters of Mixup

In this section, we consider the optimal parameter of mixup. Here, we let the parameter $\lambda \in (0,1)$. From equation 10 and equation 16, we can see that a large $1-\lambda$ has a good regularization effect. In other words, if the weight of one input vector is more extreme than the other, the mixup effect is more significant. By swapping $i$ and $j$, we can see that $\lambda$ should be close to 0 or 1.

In the original mixup paper (Zhang et al., 2018), the parameter $\lambda$ is sampled from the Beta distribution $Beta(\alpha,\alpha)$, where $\alpha$ is another parameter. Figure 3 shows some shapes of the *Beta* distribution changing $\alpha$. From this figure, we can see that when $\alpha < 1$, $\lambda$ is sampled such that one of the input vectors has a high weight (in other words, $\lambda$ is close to 0 or 1). We treated $\lambda$ as a constant in the above discussion, but if we treat it as a random variable $\lambda \sim Beta(\alpha,\alpha)$, we can obtain the following:

$$\begin{aligned}
\mathbb{E}[\lambda] &= \frac{\alpha}{\alpha+\alpha} = \frac{1}{2}, \\
Var(\lambda) &= \frac{\alpha^2}{(\alpha+\alpha)^2(\alpha+\alpha+1)} = \frac{\alpha^2}{4\alpha^2(2\alpha+1)} = \frac{1}{4(2\alpha+1)},
\end{aligned}$$

where $\alpha > 0$. Since the $\mathbb{E}[\lambda]$ is a constant, we can see that when the weight parameter $\lambda$ is close to 0 or 1, $\alpha$ is expected to be close to 0.

Figure 4 shows the experimental results for CIFAR-10 (Krizhevsky et al., 2009). We use ResNet-18 (He et al., 2016) as a classifier and apply mixup with each parameter $\alpha$ for $\lambda \sim Beta(\alpha,\alpha)$. This shows that the generalization performance is higher when the parameter $\alpha$ is a small value. The right side of Figure 4 shows a plot of the training loss and test loss of the classifier and their differences for each $\alpha$. We can see that when the value of parameter $\alpha$ is small, the difference between train loss

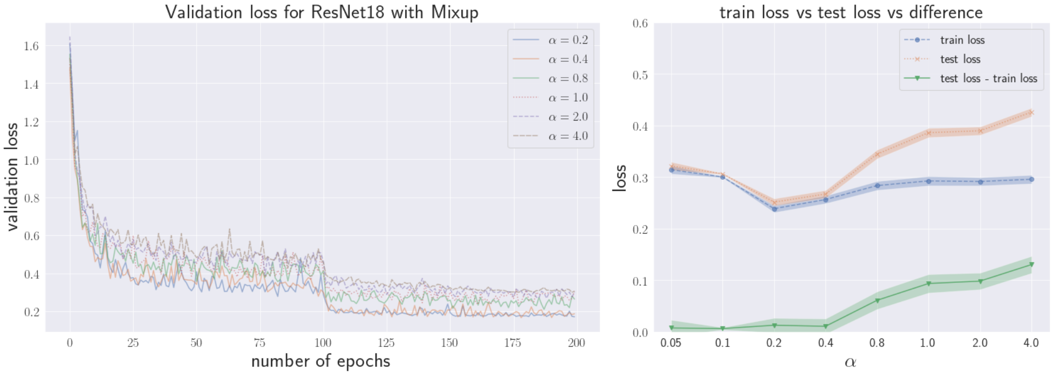

Figure 4: Experimental results for CIFAR-10 dataset.We use ResNet-18 as a classifier and apply mixup with each parameter $\alpha$ for $\lambda \sim Beta(\alpha, \alpha)$. Left: Learning curve of ResNet-18 with mixup. The generalization performance is higher when the parameter $\alpha$ is small value. Right: Plot of train loss, test loss, and their differences for each $\alpha$.

and test loss is small. Appendix D shows the details of the experiments and additional experimental results.

## 7 Geometric Perspective of Mixup Training: Parameter Space Smoothing

In this section, we consider the effect of mixup geometrically on the space of the parameters to be searched. The following theorem suggests that mixup's regularization contributes to the smoothing of convex functions corresponding to the parameter space.

**Theorem 4.** Let $p(\boldsymbol{x};\boldsymbol{\theta})$ be the exponential distribution family that depends on the unknown parameter vector $\boldsymbol{\theta}$. When mixup is applied, the second-order derivative $\nabla\nabla\psi_\lambda(\boldsymbol{\theta})$ of $\psi_\lambda(\boldsymbol{\theta})$ that characterizes the Bregman divergence between the parameter $\boldsymbol{\theta}$ and $\boldsymbol{\theta} + d\boldsymbol{\theta}$, which is a slight change of the parameter, satisfies the following:

$$\nabla\nabla\psi_\lambda(\boldsymbol{\theta}) = \lambda^2(\nabla\nabla\psi(\boldsymbol{\theta})), \tag{17}$$

where $\psi(\boldsymbol{\theta})$ is a convex function of the original data distribution and $\lambda \in (0,1)$ is a parameter of the mixup.

In optimization, the smaller the change in the gradient of the convex function, the more likely it is to avoid falling into a local solution. This means that mixup reduces the complexity in the context of the parameter search.

*Proof.* An exponential family of probability distributions is written as

$$p(\boldsymbol{x};\boldsymbol{\theta}) = \exp\left\{\sum \theta_i x_i + k(\boldsymbol{x}) - \psi(\boldsymbol{\theta})\right\}, \tag{18}$$

where $p(\boldsymbol{x};\boldsymbol{\theta})$ is the probability density function of random variable vector $\boldsymbol{x}$ specified by parameter vector $\boldsymbol{\theta}$ and $k(\boldsymbol{x})$ is a function of $\boldsymbol{x}$. Also, $\psi(\boldsymbol{\theta})$ can be written as

$$\psi(\boldsymbol{\theta}) = \log \int \exp\left\{\sum \theta_i x_i + k(\boldsymbol{x})\right\} d\boldsymbol{x}. \tag{19}$$

By differentiating equation 19, we can confirm that the Hessian becomes a positive definite matrix, which means that $\psi(\boldsymbol{\theta})$ is a convex function. Here, the Bregman divergence from $\boldsymbol{\xi}$ to $\boldsymbol{\xi}'$ is defined by using the convex function $\varphi(\boldsymbol{\xi})$:

$$D_\varphi[\xi : \xi'] = \varphi(\xi) - \varphi(\xi') - \nabla\varphi(\xi')\cdot(\xi - \xi') \tag{20}$$

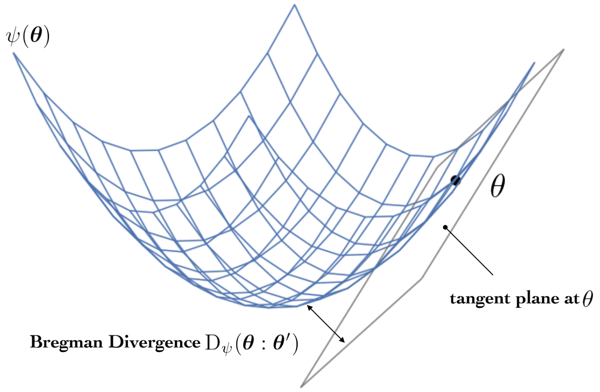
$\psi(\boldsymbol{\theta})$
$\theta$
tangent plane at $\theta$
Bregman Divergence $\mathrm{D}_{\psi}(\boldsymbol{\theta} : \boldsymbol{\theta}')$

Figure 5: Bregman divergence from $\boldsymbol{\theta}'$ to $\boldsymbol{\theta}$. This divergence derived from the convex function $\psi(\boldsymbol{\theta})$ and its supporting hyperplane with normal vector $\nabla\psi(\boldsymbol{\theta}_0)$.

Let $\psi(\cdot) = \varphi(\cdot)$ and $\boldsymbol{\theta} = \boldsymbol{\xi}$, then we can naturally define a Bregman divergence for $\psi(\cdot)$ and $\boldsymbol{\theta}$. Differentiating equation 18, we can obtain

$$\begin{aligned} 0 &= \frac{\partial}{\partial \theta_i} \int \left\{ \sum_i \theta_i x_i + k(\boldsymbol{x}) - \psi(\boldsymbol{\theta}) \right\} d\boldsymbol{x} \\ \therefore \nabla\psi(\boldsymbol{x}) &= \mathbb{E}[\boldsymbol{x}]. \end{aligned} \tag{21}$$

Differentiating again,

$$\begin{aligned} 0 &= -\frac{\partial^2}{\partial\theta_i\partial\theta_j}\psi(\boldsymbol{\theta}) + \int (x_i - \mathbb{E}[x_i])(x_j - \mathbb{E}[x_j])p(\boldsymbol{x};\boldsymbol{\theta})d\boldsymbol{x} \\ \therefore \nabla\nabla\psi(\boldsymbol{\theta}) &= Var(\boldsymbol{x}). \end{aligned} \tag{22}$$

Here, if we adopt the linear combination $\tilde{\boldsymbol{x}} = \lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{x}_j$ to find the parameter $\boldsymbol{\theta}$, we can obtain

$$\nabla\psi_\lambda(\boldsymbol{\theta}) = \mathbb{E}[\tilde{\boldsymbol{x}}] = \mathbb{E}[\lambda\boldsymbol{x} + (1-\lambda)\mathbb{E}[\boldsymbol{x}]] = \mathbb{E}[\boldsymbol{x}], \tag{23}$$

$$\nabla\nabla\psi_\lambda(\boldsymbol{\theta}) = Var(\lambda\boldsymbol{x} + (1-\lambda)\mathbb{E}[\boldsymbol{x}]) = \lambda^2\psi(\boldsymbol{\theta}), \tag{24}$$

where $\psi_\lambda(\cdot)$ is defined by

$$p(\tilde{\boldsymbol{x}};\boldsymbol{\theta}) = \exp\left\{ \sum \theta_i \tilde{x}_i + k(\tilde{\boldsymbol{x}}) - \psi_\lambda(\boldsymbol{\theta}) \right\}. \tag{25}$$

From Bayes theorem, we would be computing the probability of a parameter given the likelihood of some data: $p(\tilde{\boldsymbol{x}};\boldsymbol{\theta}) = p(\tilde{\boldsymbol{x}};\boldsymbol{\theta})p(\boldsymbol{\theta})/\sum'_{\boldsymbol{\theta}} p(\tilde{\boldsymbol{x}};\boldsymbol{\theta}')p(\boldsymbol{\theta}')$, and applying mixup means $p(\boldsymbol{x};\boldsymbol{\theta}) \to p(\tilde{\boldsymbol{x}};\boldsymbol{\theta})$. And then, we can obtain equation 17. □

For a complete proof, see Appendix C.

Bregman divergence is a generalization of KL-divergence, which is frequently used in probability distribution spaces, such as loss functions for parameter search. The above theorem means that the magnitude of the gradient of the convex function characterizing the Bregman divergence can be smoothed by using the mixup.

## 8 Conclusion and Discussion

In this paper, we provided a theoretical analysis of mixup regularization for linear classifiers and neural networks with ReLU activation functions. Our results show that a theoretical clarification of when and how effective regularization by mixup is.

Our future work includes considering whether similar arguments can be made for some variants of mixup (Verma et al., 2019; Berthelot et al., 2019; Yun et al., 2019; Lim et al., 2019; Sohn et al., 2020). Because of the simplicity of the idea and ease of implementation, there are many variants of mixup, but most of them are heuristic approaches.

Also, Tokozume et al. (Tokozume et al., 2018b) suggest that BC-Leaning, a concept roughly equivalent to the mixup, behaves in a way that increases the Fisher's criterion (Fisher, 1936). This claim is impressive, and they provide experimental support for this hypothesis, but the theoretical arguments are insufficient. It is worth considering to show theoretically that data augmentation by mixup contributes to the increase of Fisher's criterion, and to clarify how much this changes the value.

Another possible future study is a theoretical consideration of mixing data on manifolds (Verma et al., 2019). Taking data as a point in the manifold, it would lead to more advanced research to investigate how mixup training behaves on the manifold.

We believe it would be useful to divert the discussion we have had in this paper to clarify whether such modifications improve mixup and, if so, to what extent.

## References

Guillaume P Archambault, Yongyi Mao, Hongyu Guo, and Richong Zhang. Mixup as directional adversarial training. *arXiv preprint arXiv:1906.06875*, 2019.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, pp. 5050–5060, 2019.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

Luigi Carratino, Moustapha Cissé, Rodolphe Jenatton, and Jean-Philippe Vert. On mixup regularization. *arXiv preprint arXiv:2006.06049*, 2020.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223, 2011.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Tom Dietterich. Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3):326–327, 1995.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2): 303–338, 2010.

Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7 (2):179–188, 1936.

Yoav Goldberg. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309, 2017.

Hongyu Guo. Nonlinear mixup: Out-of-manifold data augmentation for text classification. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 4044–4051. AAAI Press, 2020. URL `https://aaai.org/ojs/index.php/AAAI/article/view/5822`.

Douglas M Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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

Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *International Conference on Machine Learning (ICML)*, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Steve Lawrence and C Lee Giles. Overfitting and neural networks: conjugate gradient and backpropagation. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 1, pp. 114–119. IEEE, 2000.

Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. In *Advances in Neural Information Processing Systems*, pp. 6662–6672, 2019.

Ivan Medennikov, Yuri Y Khokhlov, Aleksei Romanenko, Dmitry Popov, Natalia A Tomashenko, Ivan Sorokin, and Alexander Zatvornitskiy. An investigation of mixup training strategies for acoustic models in asr. In *Interspeech*, pp. 2903–2907, 2018.

Donald Michie, David J Spiegelhalter, CC Taylor, et al. Machine learning. *Neural and Statistical Classification*, 13(1994):1–298, 1994.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

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

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pp. 1376–1401, 2015.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020.

Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. Between-class learning for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5486–5494, 2018a.

Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. Learning from between-class examples for deep sound recognition. In *International Conference on Learning Representations*, 2018b. URL https://openreview.net/forum?id=B1Gi6LeRZ.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6438–6447, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/verma19a.html.

Kele Xu, Dawei Feng, Haibo Mi, Boqing Zhu, Dezhi Wang, Lilun Zhang, Hengxing Cai, and Shuwen Liu. Mixup-based acoustic scene classification using multi-channel convolutional neural network. In *Pacific Rim Conference on Multimedia*, pp. 14–23. Springer, 2018.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6023–6032, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1Ddp1-Rb.

# A Rademacher Complexities Bounds

## A.1 Theorem 1

**Lemma 1.** Let $\mathscr{G} : \mathscr{Z} = \mathscr{X} \times \mathscr{Y} \mapsto [0,1]$ be a family of functions. Then, for any $\delta > 0$, with probability at least $1-\delta$, the following holds for all $g \in \mathscr{G}$:

$$\mathbb{E}\Bigg[g(\boldsymbol{z}) \quad \leq \quad \frac{1}{n}\sum_{i=1}^{n} g(\boldsymbol{z}_i) + 2\mathfrak{R}_n(G) + \sqrt{\frac{\log\frac{1}{\delta}}{2m}}\Bigg] \tag{26}$$

$$\mathbb{E}\Bigg[g(\boldsymbol{z}) \quad \leq \quad \frac{1}{n}\sum_{i=1}^{n} g(\boldsymbol{z}_i) + 2\mathfrak{R}_B(G) + 3\sqrt{\frac{\log\frac{2}{\delta}}{2m}}\Bigg]. \tag{27}$$

*Proof.* (Lemma 1) For any sample $B = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n)$ and for any $g \in \mathscr{G}$, we denote by $\hat{\mathbb{E}}_B[g]$ the empirical average of $g$ over $B : \hat{\mathbb{E}}_B[g] = \frac{1}{n}\sum_{i=1}^{n} g(\boldsymbol{z}_i)$. We define the function $\Phi(\cdot)$ for any sample $B$ as follows:

$$\Phi(B) = \sup_{g \in \mathscr{G}} \mathbb{E}[g] - \hat{\mathbb{E}}_B[g]. \tag{28}$$

Let $B$ and $B'$ be two samples differing by exactly one point, which mean $\boldsymbol{z}_n \in B \wedge \boldsymbol{z}_n \notin B'$ and $\boldsymbol{z}'_n \in B' \wedge \boldsymbol{z}'_n \notin B$. Then, we have

$$\Phi(B') - \Phi(B) \quad \leq \quad \sup_{g \in \mathscr{G}} \hat{\mathbb{E}}_B[g] - \hat{\mathbb{E}}_{B'}[g] = \sup_{g \in \mathscr{G}} \frac{g(\boldsymbol{z}_n) - g(\boldsymbol{z}'_n)}{n} \leq \frac{1}{n} \tag{29}$$

$$\Phi(B) - \Phi(B') \quad \leq \quad \sup_{g \in \mathscr{G}} \hat{\mathbb{E}}_{B'}[g] - \hat{\mathbb{E}}_B[g] = \sup_{g \in \mathscr{G}} \frac{g(\boldsymbol{z}'_n) - g(\boldsymbol{z}_n)}{n} \leq \frac{1}{n} \tag{30}$$

$$\therefore \Big|\Phi(B) - \Phi(B')\Big| \quad \leq \quad \frac{1}{n}. \tag{31}$$

Then, by McDiarmid's inequality, for any $\delta > 0$, with probability at least $1 - \frac{\delta}{2}$, the following holds:

$$\Phi(B) \quad \leq \quad \mathbb{E}_B[\Phi(B)] + \sqrt{\frac{\log\frac{2}{\delta}}{2n}} \tag{32}$$

$$\mathbb{E}_B[\Phi(B)] \quad \leq \quad \mathbb{E}_{\boldsymbol{\sigma},B,B'}\left[\sup_{g \in \mathscr{G}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i(g(\boldsymbol{z}'_i) - g(\boldsymbol{z}_i))\right] \tag{33}$$

$$= \quad 2\mathbb{E}_{\boldsymbol{\sigma},B}\left[\sup_{g \in \mathscr{G}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i g(z_i)\right] = 2\mathfrak{R}_n(\mathscr{G}). \tag{34}$$

Then, using MacDiarmid's inequality, with probability $1 - \frac{\delta}{2}$ the following holds:

$$\mathfrak{R}_n(\mathscr{G}) \leq \hat{\mathscr{R}}_B(\mathscr{G}) + \sqrt{\frac{\log\frac{2}{\delta}}{2n}}. \tag{35}$$

Finally, we use the union bound and we can have the result of this lemma. □

**Lemma 2.** Let $H$ be a family of functions taking values in $\{-1,+1\}$ and let $\mathscr{G}$ be the family of loss functions associated to $H$: $\mathscr{G} = \{(x,y) \mapsto \mathbb{1}_{h(x) \neq y} : h \in H\}$. For any samples $B = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n))$, let $\mathscr{S}_{\mathscr{X}}$ denote the its projection over $\mathscr{X} : \mathscr{S}_{\mathscr{X}} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$. Then, the following relation holds between the empirical Rademacher complexities of $\mathscr{G}$ and $H$:

$$\hat{\mathfrak{R}}_B(\mathscr{G}) = \frac{1}{2}\hat{\mathfrak{R}}_{\mathscr{S}_{\mathscr{X}}}(H). \tag{36}$$

*Proof.* (Lemma 2) For any sample $B = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_2, y_2))$ of elements in $\mathscr{X} \times \mathscr{Y}$, the empirical Rademacher complexity of $\mathscr{G}$ can be written as:

$$\begin{aligned}
\hat{\mathfrak{R}}_B(\mathscr{G}) &= \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{h \in H} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \mathbb{1}_{h(\boldsymbol{x}_i) \neq y_i}\right] && (37)\\
&= \frac{1}{2}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{h \in H} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(\boldsymbol{x}_i)\right] && (38)\\
&= \frac{1}{2}\hat{\mathfrak{R}}_{\mathscr{S}_{\mathscr{X}}}(H). && (39)
\end{aligned}$$

□

*Proof.* (Theorem 1) From Lemma 1 and Lemma 2, we can have the result of Theorem 1 immediately. □

For more details, see textbooks on statistical learning theory (e.g., Shalev-Shwartz & Ben-David (2014); Mohri et al. (2018)).

## B Proof of the Complexity Reduction

In this section, we show the proofs of the theorems of the Rademacher complexity reduction. First we prove the theorem on linear discriminators, then we prove the theorem on neural networks.

### B.1 Theorem 2

*Proof.* By the Definition 3, empirical Rademacher complexity of $h(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$ is as follows:

$$\begin{aligned}
\hat{\mathfrak{R}}_B(H) &= \mathbb{E}_\sigma\left[\frac{1}{n} \sup_{\|\boldsymbol{w}\|_2 \leq \Lambda} \sum_{i=1}^{n} \sigma_i \boldsymbol{w}^T \boldsymbol{x}_i\right]\\
&= \mathbb{E}_\sigma\left[\frac{1}{n} \sup_{\|\boldsymbol{w}\|_2 \leq \Lambda} \boldsymbol{w}^T \sum_{i=1}^{n} \sigma_i \boldsymbol{x}_i\right]\\
&= \frac{1}{n}\mathbb{E}_\sigma\left[\sup_{\|\boldsymbol{w}\|_2 \leq \Lambda} \boldsymbol{w}^T \sum_{i=1}^{n} \sigma_i \boldsymbol{x}_i\right]\\
&= \frac{1}{n}\mathbb{E}_\sigma\left[\Lambda \left\|\sum_{i=1}^{n} \sigma_i \boldsymbol{x}_i\right\|_2\right] \quad (\because \text{Cauchy–Schwarz's inequality})\\
&\leq \frac{\Lambda}{n}\left(\mathbb{E}_\sigma\left[\left\|\sum_{i=1}^{n} \sigma_i \boldsymbol{x}_i\right\|_2^2\right]\right)^{\frac{1}{2}} \quad (\because \text{Jensen's inequality})\\
&= \frac{\Lambda}{n}\left(\sum_{i=1}^{n} \|\boldsymbol{x}_i\|_2^2\right)^{\frac{1}{2}}. && (40)
\end{aligned}$$

Let $\tilde{\boldsymbol{x}}_i = \mathbb{E}_{\boldsymbol{x}_j}[\lambda \boldsymbol{x}_i + (1-\lambda)\boldsymbol{x}_j]$ be the expectation of the linear combination of input vectors by mixup, where $\lambda$ is a parameter in mixup and is responsible for adjusting the weights of the two vectors. Then, we have

$$\begin{aligned}
\hat{\mathfrak{R}}^*_B(H) &= \frac{\Lambda}{n}\left(\sum_{i=1}^{n}\|\tilde{\boldsymbol{x}}_i\|_2^2\right)^{\frac{1}{2}} \\
&= \frac{\Lambda}{n}\left(\sum_{i=1}^{n}\left\|\mathbb{E}_{x_j}\left[\lambda\boldsymbol{x}_i+(1-\lambda)\boldsymbol{x}_j\right]\right\|_2^2\right)^{\frac{1}{2}} \\
&= \frac{\Lambda}{n}\left(\sum_{i=1}^{n}\left\|\lambda\boldsymbol{x}_i+(1-\lambda)\mathbb{E}_{x_j}\left[\boldsymbol{x}_j\right]\right\|_2^2\right)^{\frac{1}{2}} \quad (\because \text{Linearity of expectation}) \\
&\leq \frac{\Lambda}{n}\left(\sum_{i=1}^{n}\left(\|\lambda\boldsymbol{x}_i\|_2^2+\left\|(1-\lambda)\mathbb{E}_{\boldsymbol{x}_j}[\boldsymbol{x}_j]\right\|_2^2\right)\right)^{\frac{1}{2}} \quad (\because \text{Subadditivity of norm}) \\
&= \frac{\Lambda}{n}\left(\lambda^2\sum_{i=1}^{n}\|\boldsymbol{x}_i\|_2^2+(1-\lambda)^2\sum_{i=1}^{n}\left\|\mathbb{E}_{\boldsymbol{x}_j}[\boldsymbol{x}_j]\right\|_2^2\right)^{\frac{1}{2}}. \qquad (41)
\end{aligned}$$

From equation 40 and equation 41, we can have

$$\begin{aligned}
\hat{\mathfrak{R}}_B(H)-\hat{\mathfrak{R}}^*_B(H) &\leq \frac{\Lambda}{n}\left(\sum_{i=1}^{n}\|\boldsymbol{x}_i\|_2^2-\lambda^2\sum_{i=1}^{n}\|\boldsymbol{x}_i\|_2^2-(1-\lambda)^2\sum_{i=1}^{n}\left\|\mathbb{E}_{\boldsymbol{x}_j}[\boldsymbol{x}_j]\right\|_2^2\right)^{\frac{1}{2}} \\
&\leq \frac{\Lambda}{n}\left((1-\lambda)^2\sum_{i=1}^{n}\|\boldsymbol{x}_i\|_2^2-(1-\lambda)^2\sum_{i=1}^{n}\left\|\mathbb{E}_{\boldsymbol{x}_j}[\boldsymbol{x}_j]\right\|_2^2\right)^{\frac{1}{2}} \\
&= \frac{\Lambda|1-\lambda|}{n}\left(\sum_{i=1}^{n}\|\boldsymbol{x}_i\|_2^2-\sum_{i=1}^{n}\left\|\mathbb{E}_{\boldsymbol{x}_j}[\boldsymbol{x}_j]\right\|_2^2\right)^{\frac{1}{2}} \qquad (42) \\
&= \frac{\Lambda|1-\lambda|}{\sqrt{n}}\left(\frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{x}_i\|_2^2-\frac{1}{n}\sum_{i=1}^{n}\|\bar{\boldsymbol{x}}\|_2^2\right)^{\frac{1}{2}} \quad (\because \text{i.i.d.}) \qquad (43) \\
&= \frac{\Lambda|1-\lambda|}{\sqrt{n}}\left(s^2(\|\boldsymbol{x}\|_2)+\|\bar{\boldsymbol{x}}\|_2^2-\|\bar{\boldsymbol{x}}\|_2^2\right)^{\frac{1}{2}} \qquad (44) \\
&= \frac{\Lambda|1-\lambda|}{\sqrt{n}}\sqrt{s^2(\|\boldsymbol{x}\|_2)}\geq 0. \qquad (45)
\end{aligned}$$

□

### B.2 Theorem 3

*Proof.* By the upper bound of (Neyshabur et al., 2015), empirical Rademacher complexity of $h(x)\in H_{L,\boldsymbol{W}_L}$ is as follows:

$$\hat{\mathfrak{R}}_B(H_{L,\boldsymbol{W}_L})\leq\frac{1}{\sqrt{n}}2^{L+\frac{1}{2}}\boldsymbol{W}_L\max_i\|\boldsymbol{x}_i\|. \qquad (46)$$

Let $\tilde{\boldsymbol{x}}_i=\mathbb{E}_{\boldsymbol{x}_j}[\lambda\boldsymbol{x}_i+(1-\lambda)\boldsymbol{x}_j]$ be the expectation of the linear combination of input vectors by mixup, where $\lambda$ is a parameter in mixup and is responsible for adjusting the weights of the two vectors. Then, we have

$$\begin{aligned}
\hat{\mathfrak{R}}^*_B(H_{L,\boldsymbol{W}_L}) &\leq \frac{1}{\sqrt{n}}2^{L+\frac{1}{2}}\boldsymbol{W}_L\max_i\|\mathbb{E}_j[\lambda\boldsymbol{x}_i+(1-\lambda)\boldsymbol{x}_j]\| \\
&= \frac{1}{\sqrt{n}}2^{L+\frac{1}{2}}\boldsymbol{W}_L\max_i\|\lambda\boldsymbol{x}_i+(1-\lambda)\mathbb{E}_j[\boldsymbol{x}_j]\| \\
&\leq \frac{1}{\sqrt{n}}2^{L+\frac{1}{2}}\boldsymbol{W}_L\max_i\left\{\lambda\|\boldsymbol{x}_i\|+(1-\lambda)\|\mathbb{E}_j[\boldsymbol{x}_j]\|\right\}. \quad (\because \text{Subadditivity of norm})
\end{aligned}$$

(47)

Now we consider to bound the difference between the maximum values of each quantity,

$$\begin{aligned}
\max\left\{\hat{\mathfrak{R}}_B(H_{L,\boldsymbol{W}_L})\right\} &= \frac{1}{\sqrt{n}}2^{L+\frac{1}{2}}\boldsymbol{W}_L \max_i \|\boldsymbol{x}_i\|, \\
\max\left\{\hat{\mathfrak{R}}^*_B(H_{L,\boldsymbol{W}_L})\right\} &= \frac{1}{\sqrt{n}}2^{L+\frac{1}{2}}\boldsymbol{W}_L \max_i \left\{\lambda\|\boldsymbol{x}_i\| + (1-\lambda)\|\mathbb{E}_j[\boldsymbol{x}_j]\|\right\},
\end{aligned}$$

and then, from equation 46 and equation 47, we can have

$$\begin{align*}
\max\hat{\mathfrak{R}}_B(H_{L,\boldsymbol{W}_L}) - \max\hat{\mathfrak{R}}^*_B(H_{L,\boldsymbol{W}_L}) &= \frac{1}{\sqrt{n}}2^{L+\frac{1}{2}}\boldsymbol{W}_L\left\{\max_i\|\boldsymbol{x}_i\| - \max_i\left\{\lambda\|\boldsymbol{x}_i\| + (1-\lambda)\|\mathbb{E}_j[\boldsymbol{x}_j]\|\right\}\right\} \\
&\leq \frac{1}{\sqrt{n}}2^{L+\frac{1}{2}}\boldsymbol{W}_L \max_i\left|\|\boldsymbol{x}_i\|_2 - (\lambda\|\boldsymbol{x}_i\|_2 + (1-\lambda)\|\mathbb{E}_j[\boldsymbol{x}_j]\|_2\right| \\
&= \frac{1}{\sqrt{n}}2^{L+\frac{1}{2}}\boldsymbol{W}_L \max_i\left|(1-\lambda)\|\boldsymbol{x}_i\|_2 - (1-\lambda)\|\bar{\boldsymbol{x}}\|_2\right| \\
&= \frac{1-\lambda}{\sqrt{n}}2^{L+\frac{1}{2}}\boldsymbol{W}_L \max_i\left|\|\boldsymbol{x}_i\|_2 - \|\bar{\boldsymbol{x}}\|_2\right| \\
&= \frac{1-\lambda}{\sqrt{n}}2^{L+\frac{1}{2}}\boldsymbol{W}_L \max_i\left|\|\boldsymbol{\mu}_{\boldsymbol{x}} + \boldsymbol{\epsilon}_i\|_2 - \|\bar{\boldsymbol{x}}\|_2\right| \\
&\leq \frac{1-\lambda}{\sqrt{n}}2^{L+\frac{1}{2}}\boldsymbol{W}_L \max_i\left|\|\boldsymbol{\mu}_{\boldsymbol{x}}\|_2 + \|\boldsymbol{\epsilon}_i\|_2 - \|\bar{\boldsymbol{x}}\|_2\right| \ (\because \text{Subadditivity of norm}) \\
&= \frac{1-\lambda}{\sqrt{n}}2^{L+\frac{1}{2}}\boldsymbol{W}_L \max_i\|\boldsymbol{\epsilon}_i\|_2 \ (\because \text{Law of large numbers}) \tag{48} \\
&\geq 0 \quad (\because 1-\lambda \geq 0, \|\boldsymbol{\epsilon}_i\|_2 \geq 0),
\end{align*}$$

here equation 48 is supported by the law of large numbers.

$$\lim_{n\to\infty} P\Big(\Big|\bar{X}-\mu\| > \varepsilon\Big) = 0 \ \ (\forall\varepsilon > 0). \tag{49}$$

□

## C Effect of Mixup on the Convex Function Characterizing the Bregman Divergence

In this section, we show the proof of the theorem of the Effect of mixup on the convex function characterinzing the Bregman divergence.

### C.1 Definitions

**Definition 5.** (Bregman divergence) For some convex function $\varphi(\cdot)$ and $d$-dimensional parameter vector $\boldsymbol{\xi} \in \mathbb{R}^d$, the Bregman divergence from $\boldsymbol{\xi}$ to $\boldsymbol{\xi}'$ is defined as follows:

$$D_\varphi[\xi : \xi'] = \varphi(\xi) - \varphi(\xi') - \nabla\varphi(\xi')\cdot(\xi - \xi'). \tag{50}$$

### C.2 Theorem 4

*Proof.* An exponential family of probability distributions is written as

$$p(\boldsymbol{x};\boldsymbol{\theta}) = \exp\left\{\sum \theta_i x_i + k(\boldsymbol{x}) - \psi(\boldsymbol{\theta})\right\}, \tag{51}$$

where $p(\boldsymbol{x};\boldsymbol{\theta})$ is the probability density function of random variable vector $\boldsymbol{x}$ specified by parameter vector $\boldsymbol{\theta}$ and $k(\boldsymbol{x})$ is a function of $\boldsymbol{x}$. Since $\int p(\boldsymbol{x};\boldsymbol{\theta}) = 1$, the normalization term $\psi(\boldsymbol{\theta})$ can be written as:

$$\psi(\boldsymbol{\theta}) = \log\int \exp\left\{\sum_i \theta_i\, x_i + k(\boldsymbol{x})\right\} d\boldsymbol{x} \tag{52}$$

which is known as the cumulant generating function in statistics. By differentiating equation 52, we can confirm that the Hessian becomes a positive definite matrix, which means that $\psi(\boldsymbol{\theta})$ is a convex function. Here, the Bregman divergence from $\boldsymbol{\xi}$ to $\boldsymbol{\xi}'$ is defined by using the convex function $\varphi(\boldsymbol{\xi})$:

$$D_\varphi[\xi:\xi'] = \varphi(\xi) - \varphi(\xi') - \nabla\varphi(\xi')\cdot(\xi-\xi') \tag{53}$$

Let $\psi(\cdot) = \varphi(\cdot)$ and $\boldsymbol{\theta} = \boldsymbol{\xi}$, then we can naturally define the Bregman divergence for $\psi(\cdot)$ and $\boldsymbol{\theta}$. Differentiating equation 51, we can obtain

$$\begin{align*}
0 &= \frac{\partial}{\partial\theta_i}\int \exp\left\{\sum_i \theta_i x_i + k(\boldsymbol{x}) - \psi(\theta)\right\} d\boldsymbol{x} \\
&= \int\left\{x_i - \frac{\partial}{\partial\theta_i}\psi(\boldsymbol{\theta})\right\} p(\boldsymbol{x};\boldsymbol{\theta})d\boldsymbol{x} \tag{54} \\
&= \int x_i p(\boldsymbol{x};\boldsymbol{\theta})d\boldsymbol{x} - \frac{\partial}{\partial\theta_i}\psi(\boldsymbol{\theta}) \\
\therefore \frac{\partial}{\partial\theta_i}\psi(\boldsymbol{\theta}) &= \int x_i p(\boldsymbol{x};\boldsymbol{\theta})d\boldsymbol{x} = \mathbb{E}[x_i] \\
\nabla\psi(\boldsymbol{x}) &= \mathbb{E}[\boldsymbol{x}]. \tag{55}
\end{align*}$$

Differentiating it again,

$$\begin{align*}
0 &= \int \frac{\partial}{\partial\theta_j}\left\{x_i - \frac{\partial}{\partial\theta_i}\psi(\boldsymbol{\theta})\right\} p(\boldsymbol{x};\boldsymbol{\theta}) + \left\{x_i - \frac{\partial}{\partial\theta_i}\psi(\boldsymbol{\theta})\right\}\frac{\partial}{\partial\theta_j}p(\boldsymbol{x};\boldsymbol{\theta})d\boldsymbol{x} \\
&= \int -\frac{\partial^2}{\partial\theta_i\partial\theta_j}\psi(\boldsymbol{\theta})d\boldsymbol{x} + \int\left\{x_i - \frac{\partial}{\partial\theta_i}\psi(\boldsymbol{\theta})\right\}\left\{x_j - \frac{\partial}{\partial\theta_j}\psi(\boldsymbol{\theta})\right\} p(\boldsymbol{x};\boldsymbol{\theta})d\boldsymbol{x} \\
&= -\frac{\partial^2}{\partial\theta_i\partial\theta_j}\psi(\boldsymbol{\theta}) + \int (x_i - \mathbb{E}[x_i])(x_j - \mathbb{E}[x_j])p(\boldsymbol{x};\boldsymbol{\theta})d\boldsymbol{x} \\
&= -\frac{\partial^2}{\partial\theta_i\partial\theta_j}\psi(\boldsymbol{\theta}) + \mathbb{E}[(x_i - \mathbb{E}[x_i])(x_j - \mathbb{E}[x_j])] \\
\therefore \nabla\nabla\psi(\boldsymbol{\theta}) &= Var(\boldsymbol{x}). \tag{56}
\end{align*}$$

Here, if we adopt the linear combination $\tilde{\boldsymbol{x}} = \lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{x}_j$ to find the parameter $\boldsymbol{\theta}$, we can obtain

$$\begin{align*}
\nabla\psi_\lambda(\boldsymbol{\theta}) &= \mathbb{E}[\tilde{\boldsymbol{x}}] = \mathbb{E}[\lambda\boldsymbol{x} + (1-\lambda)\mathbb{E}[\boldsymbol{x}]] = \mathbb{E}[\boldsymbol{x}], \tag{57} \\
\nabla\nabla\psi_\lambda(\boldsymbol{\theta}) &= Var(\lambda\boldsymbol{x} + (1-\lambda)\mathbb{E}[\boldsymbol{x}]) \\
&= \lambda^2 Var(\boldsymbol{x}) + (1-\lambda)^2 Var(\mathbb{E}[\boldsymbol{x}]) \\
&= \lambda^2 Var(\boldsymbol{x}) = \lambda^2\psi(\boldsymbol{\theta}) \tag{58}
\end{align*}$$

where $\psi_\lambda(\cdot)$ is defined by

$$p(\tilde{\boldsymbol{x}};\boldsymbol{\theta}) = \exp\left\{\sum \theta_i\tilde{x}_i + k(\tilde{\boldsymbol{x}}) - \psi_\lambda(\boldsymbol{\theta})\right\}. \tag{59}$$

From Bayes theorem, we would be computing the probability of a parameter given the likelihood of some data:

$$p(\tilde{\boldsymbol{x}};\boldsymbol{\theta}) = \frac{p(\tilde{\boldsymbol{x}};\boldsymbol{\theta})p(\boldsymbol{\theta})}{\sum_{\boldsymbol{\theta}}' p(\tilde{\boldsymbol{x}};\boldsymbol{\theta}')p(\boldsymbol{\theta}')}, \tag{60}$$

and applying mixup means $p(\boldsymbol{x};\boldsymbol{\theta}) \to p(\tilde{\boldsymbol{x}};\boldsymbol{\theta})$.

And then, we can obtain equation 17. □

## D Experimental Results for Generalization Error

In this section, we introduce additional experimental results on the relationship between the mixup's parameter $\alpha$ and the generalization error. In these experiments, we used ResNet-18 as the network with $lr = 0.1$, $epochs = 200$. In addition, we performed 10 trials with different random seeds and reported the mean values of the trials.

Table 1 shows the effect of the parameter $\alpha$ on the generalization gap between train and test loss for each dataset. We can see that the smaller the value of $\alpha$, the smaller the gap between training loss and test loss.

Table 1: Effect of the parameter $\alpha$ on the generalization gap between train and test loss for each dataset.

| dataset | $\alpha = 0.1$ | $\alpha = 0.2$ | $\alpha = 0.4$ | $\alpha = 0.8$ | $\alpha = 1.0$ | $\alpha = 2.0$ | $\alpha = 4.0$ |
|---|---|---|---|---|---|---|---|
| CIFAR10 (Krizhevsky et al., 2009) | **0.0061** | 0.0126 | 0.0106 | 0.0610 | 0.0935 | 0.0982 | 0.1303 |
| CIFAR100 (Krizhevsky et al., 2009) | **0.1825** | 0.2592 | 0.2778 | 0.2923 | 0.3485 | 0.5965 | 0.6951 |
| STL10 (Coates et al., 2011) | **0.0137** | 0.0215 | 0.0296 | 0.0901 | 0.1210 | 0.1206 | 0.1691 |
| SVHN (Netzer et al., 2011) | **0.0499** | 0.0508 | 0.0571 | 0.0623 | 0.0875 | 0.1330 | 0.1828 |