OVER-TRAINING WITH MIXUP MAY HURT GENERALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixup, which creates synthetic training instances by linearly interpolating random sample pairs, is a simple and yet effective regularization technique to boost the performance of deep models trained with SGD. In this work, we report a previously unobserved phenomenon in Mixup training: on a number of standard datasets, the performance of Mixup-trained models starts to decay after training for a large number of epochs, giving rise to a U-shaped generalization curve. This behavior is further aggravated when the size of original dataset is reduced. To help understand such a behavior of Mixup, we show theoretically that Mixup training may introduce undesired data-dependent label noises to the synthesized data. Via analyzing a least-square regression problem with a random feature model, we explain why noisy labels may cause the U-shaped curve to occur: Mixup improves generalization through fitting the clean patterns at the early training stage, but as training progresses, Mixup becomes over-fitting to the noise in the synthetic data. Extensive experiments are performed on a variety of benchmark datasets, validating this explanation.

1 INTRODUCTION

Mixup has empirically shown its effectiveness in improving the generalization and robustness of deep classification models (Zhang et al., 2018; Guo et al., 2019a;b; Thulasidasan et al., 2019; Zhang et al., 2022b). Unlike the vanilla empirical risk minimization (ERM), in which networks are trained using the original training set, Mixup trains the networks with synthetic examples. These examples are created by linearly interpolating both the input features and labels of instance pairs randomly sampled from the original training set.

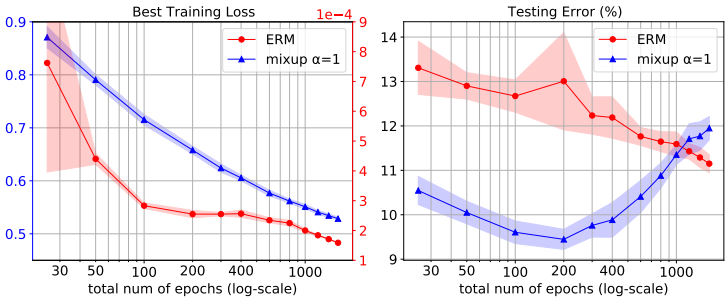


Figure 1: Over-training ResNet18 on CIFAR10. It shows that over-training reduces both the ERM and Mixup training loss, but the testing error of the Mixup-trained ResNet18 decreases first and then gradually increases, while that of the ERM-trained ResNet18 keeps decreasing.

Owning to Mixup's simplicity and its effectiveness in boosting the accuracy and calibration of deep classification models, there has been a recent surge of interest attempting to better understand Mixup's working mechanism, training characteristics, regularization potential, and possible limitations. For example, Thulasidasan et al. (2019) empirically show that Mixup helps improve the calibration of the trained networks. Guo et al. (2019a) identify the manifold intrusion issue in Mixup, where the synthetic data "intrudes" the data manifolds of the real data. Zhang et al. (2021) theoretically explain the effectiveness via analyzing an upper bound of loss function used in Mixup. Zhang et al. (2022b) suggest that the calibration effect of Mixup is correlated with the capacity of the network. In this work, we carry out an exploration along these research lines. In this paper, we further investigate the generalization properties of Mixup training.

We first report a previously unobserved phenomenon in Mixup training. Through extensive experiments on various benchmark datasets, we observe that over-training the networks with Mixup may result in significant degradation of the networks' generalization performance. As a result, along with training epochs, the generalization performance of the network measured by its testing error may exhibit a U-shaped curve. Figure 1 shows such a curve obtained from over-training ResNet18 with Mixup on CIFAR10. As can be seen from Figure 1, when training with Mixup for a long time, both ERM and Mixup keep decreasing their training loss, but the testing accuracy of the Mixup-trained ResNet18 gradually reduces, while that of the ERM-trained ResNet18 keeps decreasing.

Motivated by this observation, we conduct a theoretical analysis, aiming to better understand the aforementioned behavior of Mixup training. We show theoretically that Mixup training may introduce undesired data-dependent label noises to the synthesized data. Then by analyzing gradient-descent dynamics of training a random feature model for a least-square regression problem, we explain why noisy labels may cause the U-shaped curve to occur: under label noise, the early phase of training is primarily driven by the clean data pattern, which moves the model parameter closer to the correct solution. But as training progresses, the effect of label noise accumulates through iterations and gradually over-weighs that of the clean pattern and dominates the training process. In this phase, the model parameter gradually moves away from the correct solution until it is sufficient apart and approaches a location depending on the noise realization.

2 Related Work

Mixup Improves Generalization After the initial work of Zhang et al. (2018), a series of the Mixup's variants have been proposed (Guo et al., 2019a; Verma et al., 2019; Yun et al., 2019; Kim et al., 2020; Greenewald et al., 2021; Han et al., 2022; Sohn et al., 2022). For example, *AdaMixup* (Guo et al., 2019a) trains an extra network to dynamically determine the interpolation coefficient parameter α. *Manifold Mixup* (Verma et al., 2019) performs the linear mixing on the hidden states of the neural networks. Aside of the application aspects of Mixup, its working mechanism and the possible limitations are also being explored constantly. For example, Zhang et al. (2021) demonstrate that Mixup yields an upper bound of the Rademacher complexity of the class of functions that the network fits, which in turn bounds the generalization error of the network. Thulasidasan et al. (2019) show that Mixup helps to improve the calibration of the trained networks. Zhang et al. (2022b) theoretically justify that the calibration effect of Mixup is correlated with the capacity of the network. Additionally, manifold intrusion as introduced in Guo et al. (2019a) refers to a phenomenon in Mixup training where the synthetic data "intrudes" the data manifolds of the real data.

Training on Random Labels, Epoch-Wise Double Descent and Robust Overfitting The thought-provoking work of Zhang et al. (2017) highlights that neural networks are able to fit data with random labels. After that, the generalization behavior on corrupted label dataset has been widely investigated (Arpit et al., 2017; Liu et al., 2020; Feng & Tu, 2021; Wang & Mao, 2022; Liu et al., 2022). Specifically, Arpit et al. (2017) observe that neural networks will learn the clean pattern first before fitting to data with random labels. This is further explained by Arora et al. (2019a) where they demonstrate that under the overparameterization regime, the convergence of loss depends on the projections of labels on the eigenvectors of some Gram matrix where true labels and random labels have different projections. As a parallel research line, an *epoch-wise* double descent behavior of testing loss of deep neural networks is observed in Nakkiran et al. (2020). While the *model-wise* double descent is largely studied (Belkin et al., 2019; Hastie et al., 2022; Mei & Montanari, 2022; Ba et al., 2020), theoretical works studying the *epoch-wise* double descent are still very limited (Heckel & Yilmaz, 2021; Stephenson & Lee, 2021; Pezeshki et al., 2022). Among the related works, the work of Advani et al. (2020) inspires our theoretical analysis of the U-sharped curve of Mixup training. Moreover, robust overfitting (Rice et al., 2020) is also another yet related research line, In particular, robust overfitting is referred to a phenomenon in adversarial training that robust accuracy will first increase then decrease after a long training time. Chen et al. (2020) shows that robust overfitting is deemed to the early part of epoch-wise double descent due to the *implicit label noise* induced by adversarial training. Since Mixup training has been connected to adversarial training or adversarial robustness in the previous works (Archambault et al., 2019; Zhang et al., 2021), the work of Chen et al. (2020) indeed motivates us to study the label noise induced by Mixup training.

3 PRELIMINARIES

Consider a C-class classification setting with input space $\mathcal{X} = \mathbb{R}^{d_0}$ and label space $\mathcal{Y} := \{1, 2, \ldots, C\}$. Denote by $\mathcal{P}(\mathcal{Y})$ the space of distributions over $\mathcal{Y}$. Let $S = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ be a training set, where each $\mathbf{y}_i \in \mathcal{Y}$ may also be treated as a one-hot vector in $\mathcal{P}(\mathcal{Y})$. Let Θ denote the model parameter space, and for each $\theta \in \Theta$, let $f_\theta : \mathcal{X} \to [0,1]^C$ denote the predictive function associated with θ, which maps an input feature to a distribution in $\mathcal{P}(\mathcal{Y})$. For any pair $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{P}(\mathcal{Y})$, let $\ell(\theta, \mathbf{x}, \mathbf{y})$ denote the loss of the prediction $f_\theta(\mathbf{x})$ with respect to the target label $\mathbf{y}$. The empirical risk of the predictor θ on S is defined as

$$\hat{R}_S(\theta) := \frac{1}{n} \sum_{i=1}^n \ell(\theta, \mathbf{x}_i, \mathbf{y}_i).$$

When training with Empirical Risk Minimization (ERM), one sets out to find a θ^* to minimize this risk. It is evident that if $\ell(\cdot)$ is taken as the cross-entropy loss, the empirical risk $\hat{R}_S(\theta)$ is non-negative, where $\hat{R}_S(\theta) = 0$ precisely when $f_\theta(\mathbf{x}_i) = \mathbf{y}_i$ for every $i = 1, 2, \ldots, n$.

In Mixup, instead of using the original training set S, the training is performed on a synthetic dataset $\widetilde{S}$ obtained by interpolating training examples in S. Without loss of generality, we consider the synthetic dataset for each interpolating parameter $\lambda \in [0,1]$ as the set $\widetilde{S}_\lambda$, which is defined by

$$\widetilde{S}_\lambda := \{(\lambda \mathbf{x} + (1-\lambda)\mathbf{x}', \lambda \mathbf{y} + (1-\lambda)\mathbf{y}') : (\mathbf{x}, \mathbf{y}) \in S, (\mathbf{x}', \mathbf{y})' \in S\} \tag{1}$$

Note that in practice, $\lambda \in [0,1]$ is drawn from some prescribed distribution, independently across for all example pairs, which would correspond to sampling from $\widetilde{S}_\lambda$ for a random λ. The optimization objective, or the "Mixup loss", is then

$$\mathbb{E}_\lambda \hat{R}_{\widetilde{S}_\lambda}(\theta) := \mathbb{E}_\lambda \frac{1}{|\widetilde{S}_\lambda|} \sum_{(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \in \widetilde{S}_\lambda} \ell(\theta, \tilde{\mathbf{x}}, \tilde{\mathbf{y}})$$

Most often, the interpolating parameter λ is drawn from a symmetric Beta distribution, $\mathrm{Beta}(\alpha, \alpha)$. The default option is to take $\alpha = 1$. In this case, the following can be proved.

Lemma 3.1. *Let $\ell(\cdot)$ be the cross-entropy loss, and $\{\lambda\}$ is drawn i.i.d. from $\mathrm{Beta}(1,1)$ (or the uniform distribution on $[0,1]$). Then for all $\theta \in \Theta$ and for any given training set S that is balanced,*

$$\mathbb{E}_\lambda \hat{R}_{\widetilde{S}_\lambda}(\theta) \geq \frac{C-1}{2C},$$

where the equality holds if and only if $f_\theta(\tilde{\boldsymbol{x}}) = \tilde{\boldsymbol{y}}$ for each synthetic example $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}) \in \widetilde{S}_\lambda$.

This lower bound enables us to understand the Mixup loss during training. For example, for 10-class classification tasks, the bound has value 0.45. Then only when the Mixup loss approaches this value, we may conclude that an optimum is founded (assuming the model has sufficient capacity).

4 EMPIRICAL OBSERVATIONS

We here empirically show that over-training with Mixup may hurt the networks' generalization.

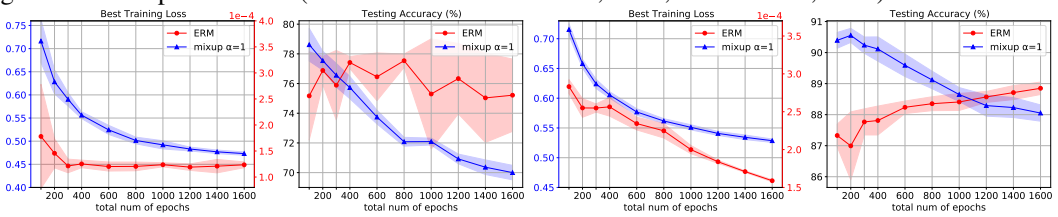
We conduct experiments using CIFAR10, CIFAR100 and SVHN. For CIFAR10 and SVHN, we adopt both the original training set and a balanced subset of it containing 30% of the data. For CIFAR100, we only use the original training set, since downsampling CIFAR100 appears to result in testing performance baselines with high variances. We train ResNet networks on the three datasets using both ERM and Mixup, while adopting SGD with weight decay. No data augmentation is used.

In each training trial, we record the achieved training loss and the corresponding epoch, as well as the network's testing accuracy at that epoch. Local loss landscape (where "loss" refers to the empirical risk defined using the real data) around the found solution is visualized in 2D following Li et al. (2018). More experimental details are given in Appendix A.

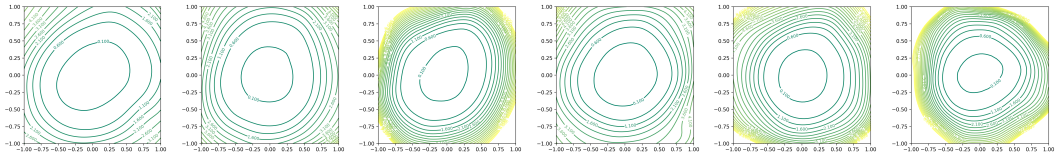
4.1 RESULTS

ResNet18 is used for the CIFAR10 and SVHN datasets.

Training is performed for up to 1600 epochs for CIFAR10, and the results for CIFAR10 are shown in Figure 2. For both the 30% dataset and the full dataset, we see clearly that after some number of epochs (e.g, epoch 200 for the full dataset), the test accuracy of the Mixup-trained network starts decreasing and this trend continues. This confirms that over-training with Mixup hurts the network's generalization. One would observe a U-shaped curve, as shown in Figure 1 (right), if we were to plot test error and include results from earlier epochs. Notably, this phenomenon is not observed in ERM. We also found that over-training with Mixup tends to force the network to learn a solution located at the sharper local minima on the loss landscape, a phenomenon correlated with degraded generalization performance (Hochreiter & Schmidhuber, 1997; Keskar et al., 2016).

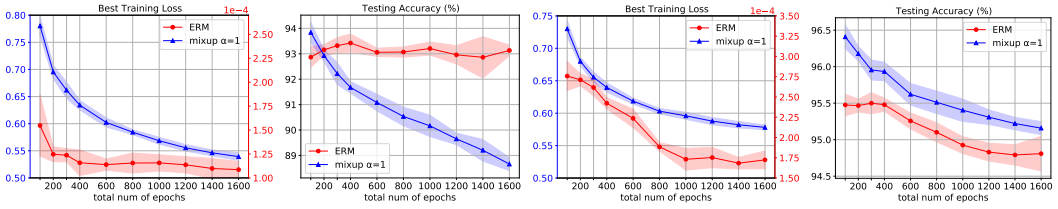


(a) Train loss (30% data) (b) Test acc (30% data) (c) Train loss (100% data) (d) Test acc (100% data)

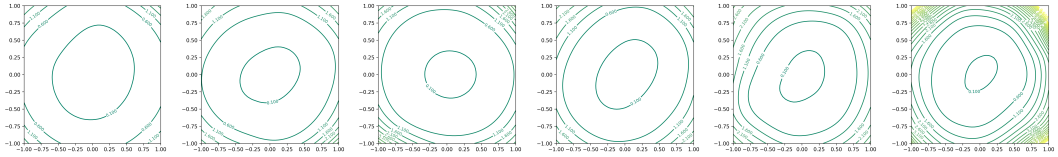


(e) 200 epochs (f) 400 epochs (g) 800 epochs (h) 200 epochs (i) 400 epochs (j) 800 epochs

Figure 2: Results of training ResNet18 on CIFAR10 training set (100% data and 30% data) without data augmentation. Top row: training loss and test accuracy for ERM and Mixup. Bottom row: loss landscape of the Mixup-trained ResNet18 at various training epochs; left 3 figures are for the 30% CIFAR10 dataset, and the right 3 are for the full CIFAR10 dataset.



(a) Train loss (30% data) (b) Test acc (30% data) (c) Train loss (100% data) (d) Test acc (100% data)

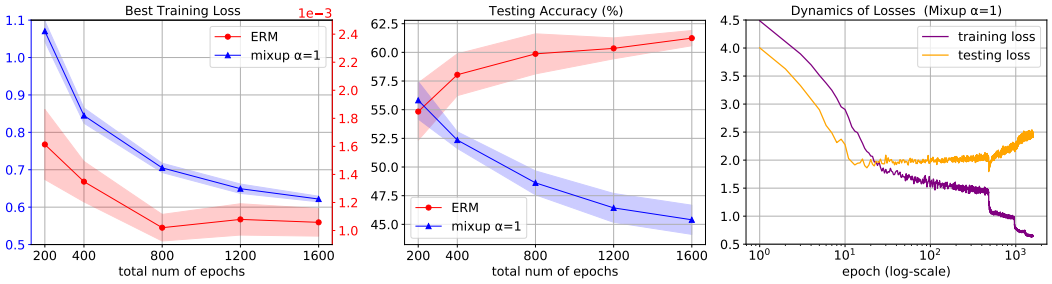


(e) 200 epochs (f) 400 epochs (g) 800 epochs (h) 200 epochs (i) 400 epochs (j) 800 epochs

Figure 3: Results of training ResNet18 on SVHN training set (100% data and 30% data) without data augmentation. Top raw: training loss and testing accuracy for ERM and Mixup. Bottom raw: loss landscape of the Mixup-trained ResNet18 at various training epochs: the left 3 figures are for the 30% SVHN dataset, and the right 3 are for the full SVHN dataset.

Training is performed for up to 1000 epochs for SVHN, and the results are presented in figure 3. Mixup exhibits a similar phenomenon as it does for CIFAR10. What differs notably is that over-training with ERM on the original SVHN training set appears to also lead to worse test accuracy. However, this does not occur on the 30% SVHN training set[1].

[1]This might be related to the epoch-wise double descent behavior of ERM training. That is, when over-training ResNet18 on the whole training set with a total of 1000 epochs, the network is still in the first stage



(a) Train loss (b) Test acc (c) U-shaped curve

Figure 4: Training loss, test accuracy and a U-shaped test loss curve (subfigure (c), yellow) of training ResNet34 on CIFAR100 (100% training data) without data augmentation.

ResNet34 is used for the more challenging task CIFAR100. This choice allows Mixup training to drive its loss to lower values, closer to the lower bound given in Lemma 3.1. Training is performed for up to 1600 epochs. The results are plotted in Figure 4. The results again confirm that over-training with Mixup hurts the generalization capability of the learned model. A U-shaped testing loss curve (obtained from a single trial) is also observed in Figure 4c.

Additional results of training ResNet34 on CIFAR10 and SVHN are provided in Appendix B.

5 Theoretical Explanation

5.1 Mixup Induces Label Noise

We will use the capital letters X and Y to denote the random variables corresponding to the input feature and output label, while reserving the notations $\mathbf{x}$ and $\mathbf{y}$ to denote their realizations respectively. In particular, we consider that each true label $\mathbf{y}$ is an element, i.e, a token, in $\mathcal{Y}$, not a one-hot vector in $\mathcal{P}(\mathcal{Y})$. Let $P(Y|X)$ be the ground-truth conditional distribution of the label Y given input feature X. For simplicity of notation, we will also express $P(Y|X)$ using a vector-valued function $f : \mathcal{X} \to \mathbb{R}^C$, where $f_j(\mathbf{x}) \triangleq P(Y = j|X = \mathbf{x})$ for each dimension $j \in \mathcal{Y}$ and input $\mathbf{x}$. Under this ground truth, the correct hard-assignment of label for $\mathbf{x}$ is $\arg\max_{j\in\mathcal{Y}} f_j(\mathbf{x})$.

For simplicity, we consider Mixup with a fixed $\lambda \in [0, 1]$; extension to random λ is straight-forward. Let $\widetilde{X}$ and $\widetilde{Y}$ be the random variables corresponding to the synthetic feature and synthetic label respectively. Then $\widetilde{X} \triangleq \lambda X + (1 - \lambda)X'$. Let $P(\widetilde{Y}|\widetilde{X})$ be the conditional distribution of the synthetic label conditioned on the synthetic feature, induced by Mixup, namely, $P(\widetilde{Y} = j|\widetilde{X}) = \lambda f_j(X) + (1-\lambda) f_j(X')$ for each j. Then for a synthetic feature $\widetilde{X}$, there are two ways to assign to it a hard label. The first is based on the ground truth, assigning $\widetilde{Y}_{\mathrm{h}}^* \triangleq \arg\max_{j\in\mathcal{Y}} f_j(\widetilde{X})$. The second is based on the Mixup-induced conditional $P(\widetilde{Y}|\widetilde{X})$, assigning $\widetilde{Y}_{\mathrm{h}} \triangleq \arg\max_{j\in\mathcal{Y}} P(\widetilde{Y} = j|\widetilde{X})$. When the two assignments disagree, or $\widetilde{Y}_{\mathrm{h}} \neq \widetilde{Y}_{\mathrm{h}}^*$, we say that the Mixup-assigned label $\widetilde{Y}_{\mathrm{h}}$ is noisy.

Theorem 5.1. *For any fixed X, X' and $\widetilde{X}$ related by $\widetilde{X} = \lambda X + (1 - \lambda)X'$ for a fixed $\lambda \in [0, 1]$, the probability of assigning a noisy label is lower bounded by*

$$P(\widetilde{Y}_{\mathrm{h}} \neq \widetilde{Y}_{\mathrm{h}}^*|\widetilde{X}) \geq \mathrm{TV}(P(\widetilde{Y}|\widetilde{X}), P(Y|X)) \geq \frac{1}{2} \sup_{j\in\mathcal{Y}} \left| f_j(\widetilde{X}) - [(1 - \lambda)f_j(X) + \lambda f_j(X')] \right|,$$

where $\mathrm{TV}(\cdot, \cdot)$ is total variation distance (see Appendix D).

Remark 5.1. *This lower bound hints that label noises induced by Mixup training depends on the distribution of original data P_X, the convexity of $f(X)$ and the value of λ. Clearly, Mixup will not create any noisy label almost surely when f_j is linear for each j.*

Remark 5.2. *In practice, we often use the one-hot vector to denote the real data label, that is to say, we let $\max_{j\in\mathcal{Y}} f_j(X) = 1$ and $\sum_{j=1}^{C} f_j(X) = 1$. Thus, the probability of assigning noisy label to a*

of over-fitting the training data, while when over-training the network on 30% of the training set, the network learns faster on the training data due to the smaller sample size, thus it passes the turning point of the double descent curve earlier.

given synthetic data can be discussed in three situations: i) if $\widetilde{Y}_{\mathrm{h}}^* \notin \{Y, Y'\}$, where Y could be the same with Y', then $\widetilde{Y}$ is a noisy label with probability one; ii) if $\widetilde{Y}_{\mathrm{h}}^* \in \{Y, Y'\}$ where $Y \neq Y'$, then a noisy label is assigned with probability at least λ or $1-\lambda$; iii) if $\widetilde{Y}_{\mathrm{h}}^* = Y = Y'$, then $\widetilde{Y}_{\mathrm{h}}^* = \widetilde{Y}$.

As shown in some previous works (Arpit et al., 2017; Arora et al., 2019a), when neural networks are trained with a fraction of random labels, they will first learn the clean data and then will overfit to the data with noisy labels. In the Mixup training case, we indeed create much more data than traditional ERM training (e.g., n^2 for a fixed λ). Thus, Mixup training will give higher testing performance in the first stage of learning (where neural networks learn the true pattern of data (Arpit et al., 2017)), and then performance will be impaired due to overfitting to noisy data. In some cases, if $\widetilde{Y}_{\mathrm{h}}^* \notin \{Y, Y'\}$ happens with a high chance, a phenomenon known as "manifold intrusion" (Guo et al., 2019a), then synthetic dataset contains too many noisy labels, Mixup training may be unable to give better performance than ERM training.

To this end, Theorem 5.1 suggests that for any classification problem, Mixup training induces label noise. Next, we will provide a theoretical analysis using a regression setup to explain that such label noise may result in the U-shape learning curve. The choice of a regression setup in this analysis is due to the difficulty in directly analyzing classification problems (under the cross-entropy loss). We note that the regression setting may not perfectly explain the U-shaped curve in classification datasets, we however believe that they give adequate insight illuminating such scenarios as well. Indeed, due to the difficulty of analyzing the cross-entropy loss, most of the analytic works for deep learning study regression problems under the square-error loss and use insights obtain this way to explain the behaviour of deep neural net in classification settings. For example, Arora et al. (2019b) uses a regression setup to analyze the optimization and generalization property of overpparameterized neural networks. Yang et al. (2020) theoretically analyze the bias-variance trade-off in deep network generalization using a regression problem.

5.2 Regression Setting With Random Feature Models

To further illustrate the training dynamics on dataset with noisy labels, we now consider a simple least squares regression problem. Let $\mathcal{Y} = \mathbb{R}$ and let $f : \mathcal{X} \to \mathcal{Y}$ be the ground-truth labelling function. Let $(\widetilde{X}, \widetilde{Y})$ be a synthetic pair obtained by mixing (X, Y) and (X', Y'). Let $\widetilde{Y}^* = f(\widetilde{X})$ and $Z \triangleq \widetilde{Y} - \widetilde{Y}^*$. Then Z can be regarded as noise introduced by Mixup, which may be data-dependent. For example, if f is strongly convex with some parameter $\rho > 0$, then $Z \geq \frac{\rho}{2}\lambda(1-\lambda)||X - X'||_2^2$.

Given a synthesized training dataset $\widetilde{S} = \{(\widetilde{X}_i, \widetilde{Y}_i)\}_{i=1}^m$, consider a random feature model, $\theta^T \phi(X)$, where $\phi : \mathcal{X} \to \mathbb{R}^d$ and $\theta \in \mathbb{R}^d$. We will train the model using gradient descent on the MSE loss

$$\hat{R}_{\widetilde{S}}(\theta) \triangleq \frac{1}{2m} \left|\left|\theta^T \widetilde{\Phi} - \widetilde{\mathbf{Y}}^T\right|\right|_2^2,$$

where $\widetilde{\Phi} = [\phi(\widetilde{X}_1), \phi(\widetilde{X}_2), \ldots, \phi(\widetilde{X}_m)] \in \mathbb{R}^{d \times m}$ and $\widetilde{\mathbf{Y}} = [\widetilde{Y}_1, \widetilde{Y}_2, \ldots, \widetilde{Y}_m] \in \mathbb{R}^m$. Notably, we will consider ϕ as fixed and only update θ.

For a fixed λ, Mixup can create $m = n^2$ synthesized examples. Thus it is reasonable to assume $m > d$ (e.g., under-parameterized regime) in Mixup training. For example, ResNet-50 has less than 30 million parameters while the square of CIFAR10 training dataset size is larger than 200 million without using other data augmentation techniques. Then the gradient flow is

$$\dot{\theta} = -\eta \nabla \hat{R}_{\widetilde{S}}(\theta_t) = \frac{\eta}{m} \widetilde{\Phi}\widetilde{\Phi}^T \left(\widetilde{\Phi}^\dagger \widetilde{\mathbf{Y}} - \theta_t\right), \tag{2}$$

where η is learning rate and $\widetilde{\Phi}^\dagger = (\widetilde{\Phi}\widetilde{\Phi}^T)^{-1}\widetilde{\Phi}$ is the Moore–Penrose inverse of $\widetilde{\Phi}^T$ (only possible when $m > d$). Thus, we have the following important lemma.

Lemma 5.1. *Let $\theta^* = \widetilde{\Phi}^\dagger \widetilde{\mathbf{Y}}^*$ and $\theta^{\mathrm{noise}} = \widetilde{\Phi}^\dagger \mathbf{Z}$ wherein $\mathbf{Z} = [Z_1, Z_2, \ldots, Z_m] \in \mathbb{R}^m$, the ODE of Eq. (2) has the following closed form solution*

$$\theta_t - \theta^* = (\theta_0 - \theta^*) e^{-\frac{\eta}{m}\widetilde{\Phi}\widetilde{\Phi}^T t} + (\mathbf{I}_d - e^{-\frac{\eta}{m}\widetilde{\Phi}\widetilde{\Phi}^T t})\theta^{\mathrm{noise}}. \tag{3}$$

Remark 5.3. *Lemma 5.1 indicates that the dynamics of θ gives a U-shaped curve in each dimension, and the increasing behavior results from the second term that contains the noise $\mathbf{Z}$. More precisely,*

the first term in Eq. (3) is monotonically decreasing and it dominates the dynamics of θ_t in the early phase of learning. Remarkably $\theta^* = \widetilde{\Phi}^\dagger \widetilde{\mathbf{Y}}^*$ may be understood as the "clean pattern" of the training data. Then we see that the model, in the early phase, is learning the "clean pattern", which generalizes to the unseen data (i.e. (X, Y)). In the later training phase, the second term in Eq. (3) gradually dominates the trajectory of θ_t, and the model learns the"noisy pattern", namely converging to $\theta^* + \theta^{\text{noise}}$. This then hurts generalization. It is noteworthy that $\theta^* + \theta^{\text{noise}}$ is also the closed-form solution for the regression problem (under Mixup labels). This suggests that the optimization problem associated with the Mixup loss has a "wrong" solution, but it is possible to benefit from only solving this problem partially, using gradient descent without over-training.

For a given synthesized dataset $\widetilde{S}$, the expected population risk as a function of time step t is

$$R_t \triangleq \mathbb{E}_{\theta_t, X, Y} \left|\left|\theta_t^T \phi(X) - Y\right|\right|_2^2 .$$

The following theorem shows the dynamics of the population risk under mild assumptions.

Theorem 5.2 (Dynamic of Population Risk)**.** *Given a synthesized dataset $\widetilde{S}$, assume $\theta_0 \sim \mathcal{N}(0, \xi^2 \mathrm{I}_d)$, $||\phi(X)||^2 \leq C_1/2$ for some constant $C_1 > 0$ and $|Z| \leq \sqrt{C_2}$ for some constant $C_2 > 0$, then we have the following upper bound*

$$R_t - R^* \leq C_1 \sum_{k=1}^{d} \left[\left(\xi_k^2 + \theta_k^{*2}\right) e^{-2\eta\mu_k t} + \frac{C_2}{\mu_k} \left(1 - e^{-\eta\mu_k t}\right)^2 \right] + 2\sqrt{C_1 R^* \zeta},$$

where $R^* = \mathbb{E}_{X,Y} \left|\left|Y - \theta^{*T}\phi(X)\right|\right|_2^2$, $\zeta = \sum_{k=1}^{d} \max\{\xi_k^2 + \theta_k^{*2}, \frac{C_2}{\mu_k}\}$ and μ_k is the kth eigenvalue of the matrix $\frac{1}{m}\widetilde{\Phi}\widetilde{\Phi}^T$.

Remark 5.4. *The additive noise Z is usually assumed as a zero mean Gaussian in the literature of generalization dynamics analysis (Advani et al., 2020; Pezeshki et al., 2022; Heckel & Yilmaz, 2021). We note that the boundness assumption of the data-dependent noise in the theorem is easily satisfied as long as the output of f is bounded, while there is no clue to assume Z is Gaussian.*

Remark 5.5. *If we further let $\xi = 0$ (i.e. using zero initialization) and assume that the eigenvalues of the matrix $\frac{1}{m}\widetilde{\Phi}\widetilde{\Phi}^T$ are all equal to μ, then the summation part in the bound above can be re-written as $C_1 \left|\left|\theta^*\right|\right|^2 e^{-2\eta\mu t} + C_2/\mu \left(1 - e^{-\eta\mu t}\right)^2$, then it is clear that the magnitude of the curve is controlled by the norm of θ^*, the norm of the representation, the noise level and μ.*

Theorem 5.2 indicates that the population risk will first decrease due to the first term (i.e. $\left(\xi_k^2 + \theta_k^{*2}\right) e^{-2\eta\mu_k t}$) then it will grow due to the existence of label noises (i.e. $\frac{C_2}{\mu_k}\left(1 - e^{-\eta\mu_k t}\right)^2$) . Overall, the population risk will be endowed with the U-shaped curve. Notice that the quantity $\eta\mu_k$ plays a key role in the upper bound, the larger of $\eta\mu_k$, the inflection point of the U-shaped curve comes earlier. This may have an interesting application, justifying a multi-stage training strategy where the learning rate is reduced at each new stage. Suppose that with the initial learning rate, at epoch T, the test error has dropped to the bottom of the U-curve corresponding to this learning rate. If the learning rate is decreased at this point, then the U-shape curve corresponding to the new learning rate may have a lower minimum error and its bottom shifted to the right. In this case, the new learning rate allows the testing error to move to the new U-shaped curve and further decay.

6 Empirical Verification

6.1 A Teacher-Student Toy Setup

To empirically verify our theoretical results discussed in Section 5.2, we invoke a simple teacher-student setting. Consider the original data $\{X_i\}_{i=1}^n$ are drawn i.i.d. from a standard Gaussian $\mathcal{N}(0, \mathrm{I}_{d_0})$, and the teacher network is a two-layer neural networks with **Tanh** as the activation function. The student network is also a two-layer neural network with **Tanh**, where we fix the parameters in the first layer and only train the second layer. The output dimension of the first layer is 100 (i.e. $d = 100$) and we use full-batch gradient descent to train the student network with MSE as the loss function. For the value of λ, we consider

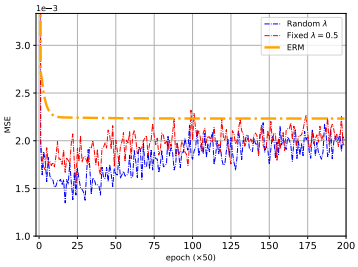


Figure 5: Dynamics of testing loss in the toy example.

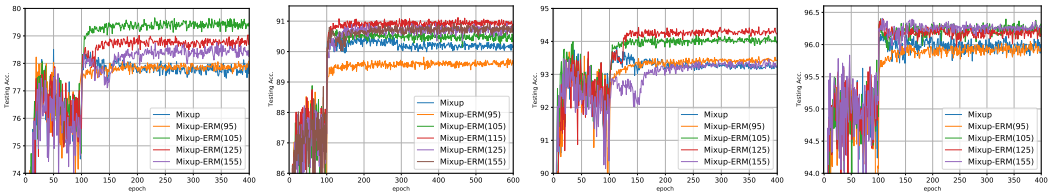
two cases: a fixed value with $\lambda = 0.5$ and random values drawn from a $\mathrm{Beta}(1, 1)$ distribution at each epoch. As a comparison, we also present the result of ERM training in an overparameterized regime (i.e., $n < d$).

All the testing loss dynamics are presented in Figure 5. We first note that Mixup still outperforms ERM in this regression problem, but clearly, only Mixup training has a U-shaped curve while the testing loss of ERM training converges to a constant value. Furthermore, the testing loss of Mixup training is endowed with a U-shaped behavior for both two λ scenarios, namely constant value of 0.5 and $\mathrm{Beta}(1, 1)$. This indeed justifies that our analysis does not depend on specific λ distribution. Moreover, Figure 5 indicates that when λ is fixed to 0.5, the increasing stage of the U-shaped curve comes earlier than that of λ with $\mathrm{Beta}(1, 1)$. This is again consistent with our theoretical results in Section 5.2. That is, owning to the fact that λ with a constant value of 0.5 provides larger noise level for Mixup, the noise domination effect for Mixup training comes earlier.

6.2 Using Mixup Only in the Early Stage of Training

We here aim to empirically verify that Mixup can induce label noises as discussed in Section 5.1.

If Mixup training learns "clean patterns" in the early stage of the training and then overfits the "noisy patterns" in the latter stage of training, then we can stop using Mixup after a certain number of epochs. Doing so, we can prevent the training from overfitting the noises induced by Mixup. We present the results of utilizing such training schema for both CIFAR10 and SVHN in Figure 6.



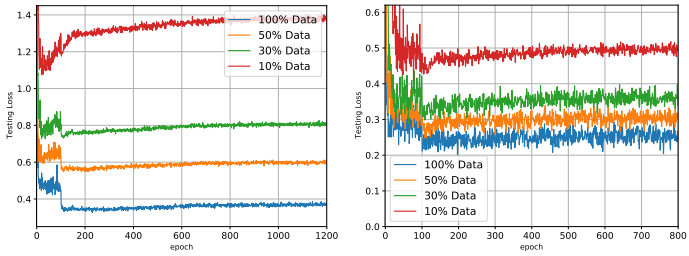
(a) CIFAR10 (30%) (b) CIFAR10 (100%) (c) SVHN (30%) (d) SVHN (100%)

Figure 6: Switching from Mixup training to ERM training. The number in the bracket is the epoch number where we let $\alpha = 0$ (i.e. Mixup training becomes ERM training).

Results in Figure 6 clearly indicate that switching from Mixup to ERM at a propertied time will successfully avoid the generalization degradation of Mixup training. Figure 6 also suggests that it may not boost the model performance if we change Mixup to ERM before the clean samples created by Mixup have large effect. In addition, if we change Mixup to ERM too late then the memorization of noisy data may already has negative effect on generalization. We note that our results here can be regarded as a complement to (Golatkar et al., 2019), where the authors show that regularization techniques only matter during the early phase of learning.

7 Further Investigation

Impact of Data Size on U-shaped Curve Although the U-shaped behavior occurs for using both 100% and 30% of the original data of CIFAR10 and SVHN, we notice that smaller size datasets facilitate the U-shaped behavior to present. We present such experimental results in Figure 7.



(a) CIFAR10 (b) SVHN

Figure 7: Over-training on Different Number of Samples.

It may be tempting to think that we can apply the usual analysis of generalization dynamics from the existing literature (Liao & Couillet, 2018; Advani et al., 2020; Stephenson & Lee, 2021). For example, one can analyze the distribution of the eigenvalues in Theorem 5.2. Specifically, if entries in Φ are independent identically distributed with zero mean, then

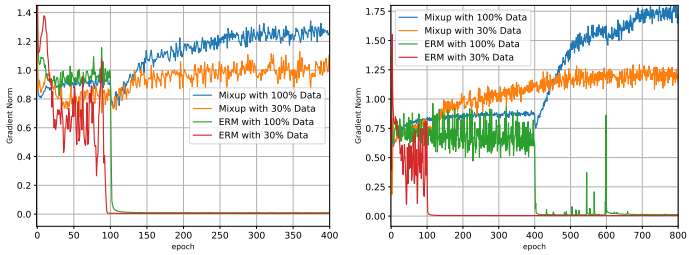
the eigenvalues $\{\mu_k\}_{k=1}^d$ follow the Marchenko-Pasteur (MP) distribution (Marčenko & Pastur, 1967) in the limit $d, m \to \infty$ with $d/m = \gamma \in (0, +\infty)$, which is defined as

$$P^{MP}(\mu|\gamma) = \frac{1}{2\pi}\frac{\sqrt{(\gamma_+ - \mu)(\mu - \gamma_-)}}{\mu\gamma}\mathbf{1}_{\mu\in[\gamma_-,\gamma_+]},$$

where $\gamma_\pm = (1 \pm \gamma)^2$. Note that the P^{MP} are only non-zero when $\mu = 0$ or $\mu \in [\gamma_-, \gamma_+]$. When γ is close to one, the probability of extremely small eigenvalues is immensely increased. From Theorem 5.2, when μ_k is small, the second term that contains noise will badly dominate the behavior of population risk and converge to a larger value. Thus, by letting $d \ll m$ will alleviate the domination of the noise term in Theorem 5.2. However, it is important to note that such analysis is not rigorous enough since columns in Φ are not independent (each two columns might come from the linear combination of the same two original instances). To apply the similar analysis here, one may need to remove or relax the independence conditions for the MP distribution to hold, for example, by invoking some techniques developed in Bryson et al. (2021). This is beyond the scope of this paper, and we would like to leave it for future study.

Gradient Norm in Mixup Training Does Not Vanish Normally, ERM training obtains zero gradient norm at the end of training, which indicates that SGD finds a local minimum. However, We observe that the gradient norm of Mixup training does not converge to zero, as shown in Figure 8.

Indeed, gradient norm in the Mixup training even increases until converging to a maximum value instead of converging to zero. When models are trained on random labels, this increasing trend of gradient norm is also observed in the previous works (Feng & Tu, 2021; Wang & Mao, 2022). Specifically, in Wang & Mao (2022), such increasing behavior is interpreted as a sign that the training of SGD enters a "memorization regime", and after the overparameterized neural networks memorize all the noisy labels, the gradient norm (or gradient dispersion in Wang & Mao (2022)) will decrease again until it converges to zero. In Mixup training, since the size of synthetic dataset is usually larger than the number of parameters (i.e., $m > d$), neural networks may not be able to memorize all the noisy labels in this case. Notice that m is larger than n^2 in practice since λ is not fixed for all the pairs of original data.



(a) Gradient Norm on CIFAR10 (b) Gradient Norm on SVHN

Figure 8: Dynamics of gradient norm.

Notably, although ERM training is able to find a local minimum in the first 130 epochs on CIFAR10, Figure 1 indicates that Mixup training outperforms ERM in the first 400 epochs. Similar observation also holds for SVHN. This result indeed suggests that Mixup can generalize well without converging to any stationary points. Notice that there is a close observation in the recent work of Zhang et al. (2022a), where they show that large-scale neural networks generalize well without vanishing of the gradient norm during training. Additionally, by switching Mixup training to ERM training, as what we did in Figure 6, the gradient norm will instantly become zero (see Figure 15 in Appendix B.4). This further justifies that the "clean patterns" are already learned by Mixup trained neural networks at the early stage of training, and the original data may not provide any useful gradient signal.

8 Concluding Remarks

We discovered a novel phenomenon in Mixup: over-training with Mixup may give rise to a U-shaped generalization curve. We theoretically show that this is due to the data-dependent label noises introduced to the synthesized data, and suggest that Mixup improves generalization through fitting the clean patterns at the early training stage, but over-fits the noise as training proceeds. The effectiveness of Mixup and the fact it works by only partially optimizing its loss function without reaching convergence, as are validated by our analysis and experiments, seem to suggest that the dynamics of the iterative learning algorithm and an appropriate criteria for terminating the algorithm might be more essential than the loss function or the solutions to the optimization problem. Exploration in the space of iterative algorithms (rather than the space of loss functions) may lead to fruitful discoveries.

REFERENCES

Madhu S Advani, Andrew M Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020.

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

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pp. 322–332. PMLR, 2019a.

Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *ICML*, volume 97, pp. 322–332. PMLR, 2019b.

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International conference on machine learning*, pp. 233–242. PMLR, 2017.

Jimmy Ba, Murat Erdogdu, Taiji Suzuki, Denny Wu, and Tianzong Zhang. Generalization of two-layer neural networks: An asymptotic viewpoint. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1gBsgBYwH.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Jennifer Bryson, Roman Vershynin, and Hongkai Zhao. Marchenko–pastur law with relaxed independence conditions. *Random Matrices: Theory and Applications*, 10(04):2150040, 2021.

Tianlong Chen, Zhenyu Zhang, Sijia Liu, Shiyu Chang, and Zhangyang Wang. Robust overfitting may be mitigated by properly learned smoothening. In *International Conference on Learning Representations*, 2020.

Yu Feng and Yuhai Tu. Phases of learning dynamics in artificial neural networks in the absence or presence of mislabeled data. *Machine Learning: Science and Technology*, 2(4):043001, 2021.

Aditya Sharad Golatkar, Alessandro Achille, and Stefano Soatto. Time matters in regularizing deep networks: Weight decay and data augmentation affect early learning dynamics, matter little near convergence. *Advances in Neural Information Processing Systems*, 32, 2019.

Kristjan Greenewald, Anming Gu, Mikhail Yurochkin, Justin Solomon, and Edward Chien. k-mixup regularization for deep learning via optimal transport, 2021. URL https://arxiv.org/abs/2106.02933.

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

Hongyu Guo, Yongyi Mao, and Richong Zhang. Augmenting data with mixup for sentence classification: An empirical study. *CoRR*, abs/1905.08941, 2019b. URL http://arxiv.org/abs/1905.08941.

Xiaotian Han, Zhimeng Jiang, Ninghao Liu, and Xia Hu. G-mixup: Graph data augmentation for graph classification. *arXiv preprint arXiv:2202.07179*, 2022.

Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *The Annals of Statistics*, 50(2):949–986, 2022.

Reinhard Heckel and Fatih Furkan Yilmaz. Early stopping in deep networks: Double descent and how to eliminate it. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=tlV90jvZbw.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural computation*, 9(1):1–42, 1997.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

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.

David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

Zhenyu Liao and Romain Couillet. The dynamics of learning: A random matrix approach. In *International Conference on Machine Learning*, pp. 3072–3081. PMLR, 2018.

Sheng Liu, Jonathan Niles-Weed, Narges Razavian, and Carlos Fernandez-Granda. Early-learning regularization prevents memorization of noisy labels. *Advances in neural information processing systems*, 33:20331–20342, 2020.

Sheng Liu, Zhihui Zhu, Qing Qu, and Chong You. Robust training under label noise by overparameterization. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 14153–14172. PMLR, 17–23 Jul 2022.

Shangyun Lu, Bradley Nott, Aaron Olson, Alberto Todeschini, Hossein Vahabi, Yair Carmon, and Ludwig Schmidt. Harder or different? a closer look at distribution shift in dataset reproduction. In *ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2020.

Vladimir A Marčenko and Leonid Andreevich Pastur. Distribution of eigenvalues for some sets of random matrices. *Mathematics of the USSR-Sbornik*, 1(4):457, 1967.

Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, 75(4):667–766, 2022.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1g5sA4twr.

Mohammad Pezeshki, Amartya Mitra, Yoshua Bengio, and Guillaume Lajoie. Multi-scale feature learning dynamics: Insights for double descent. In *International Conference on Machine Learning*, pp. 17669–17690. PMLR, 2022.

Francesco Pinto, Harry Yang, Ser-Nam Lim, Philip HS Torr, and Puneet K Dokania. Regmixup: Mixup as a regularizer can surprisingly improve accuracy and out distribution robustness. *arXiv preprint arXiv:2206.14502*, 2022.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do cifar-10 classifiers generalize to cifar-10? *arXiv preprint arXiv:1806.00451*, 2018.

Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pp. 8093–8104. PMLR, 2020.

Jy-yong Sohn, Liang Shang, Hongxu Chen, Jaekyun Moon, Dimitris Papailiopoulos, and Kangwook Lee. Genlabel: Mixup relabeling using generative models. *arXiv preprint arXiv:2201.02354*, 2022.

Cory Stephenson and Tyler Lee. When and how epochwise double descent happens. *arXiv preprint arXiv:2108.12006*, 2021.

Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

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. PMLR, 09–15 Jun 2019.

Ziqiao Wang and Yongyi Mao. On the generalization of models trained with SGD: Information-theoretic bounds and implications. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=oWZsQ8o5EA.

Zitong Yang, Yaodong Yu, Chong You, Jacob Steinhardt, and Yi Ma. Rethinking bias-variance trade-off for generalization of neural networks. In *ICML*, 2020.

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/CVF international conference on computer vision*, pp. 6023–6032, 2019.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Sy8gdB9xx.

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.

Jingzhao Zhang, Haochuan Li, Suvrit Sra, and Ali Jadbabaie. Neural network weights do not converge to stationary points: An invariant measure perspective. In *International Conference on Machine Learning*, pp. 26330–26346. PMLR, 2022a.

Linjun Zhang, Zhun Deng, Kenji Kawaguchi, Amirata Ghorbani, and James Zou. How does mixup help with robustness and generalization? In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=8yKEo06dKNo.

Linjun Zhang, Zhun Deng, Kenji Kawaguchi, and James Zou. When and how mixup improves calibration. In *International Conference on Machine Learning*, pp. 26135–26160. PMLR, 2022b.

A EXPERIMENTAL SETUPS OF OVER-TRAINING

In each training trial, we train the network for in total a fixed number of epochs. We record the minimal training loss achieved by the network during the training process, and we also record the network's testing accuracy of the epoch at which the minimal training loss is achieved. Additionally, we visualize the local loss landscape around the solution found by the network at the aforementioned training epoch. We gradually increase the total number of the training epochs in different trials of training so as to gradually over-train the network.

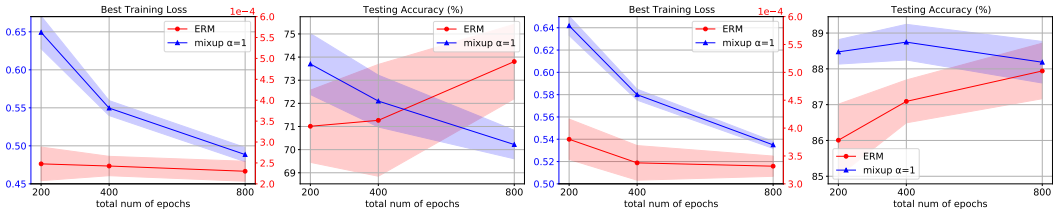
We repeat each training trial for 10 times (using 10 different random seeds) and we average the recorded training losses and testing accuracies. For example, Figure 2a illustrates the results of training ResNet18 on 30% CIFAR10 data. Each point represents the average of the recorded training losses in a training trial, and its label on the horizonal axis represents the setting of total number of epochs of that trial. The width of the shade beside each point reflects the deviation of the recorded results in the corresponding trial.

B ADDITIONAL EXPERIMENTAL RESULTS

B.1 ADDITIONAL RESULTS OF OVER-TRAINING

Besides CIFAR100, ResNet34 is also used for the CIFAR10 and the SVHN datasets.

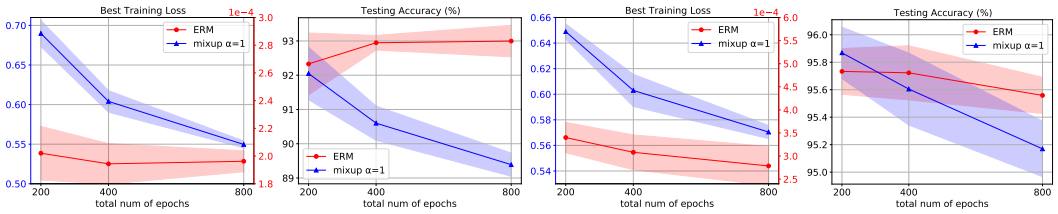
Training is performed on both CIFAR10 and SVHN for in total 200, 400 and 800 epochs respectively. The results for CIFAR10 are shown in Figure 9. For both the 30% dataset and the original dataset, Mixup exhibits a similar phenomenon as it does in training ResNet18 on CIFAR10. The difference is that over-training ResNet34 with ERM let the testing accuracy gradually increase on both the 30% dataset and the original dataset.



(a) Train loss (30% data) (b) Test acc (30% data) (c) Train loss (100% data) (d) Test acc (100% data)

Figure 9: Results of the recorded training losses and testing accuracies of training ResNet34 on CIFAR10 training set (100% data and 30% data) without data augmentation.

The results for SVHN are shown in Figure 10. These results are also in accordance with those of training ResNet18 on both 30% and 100% of the SVHN dataset.

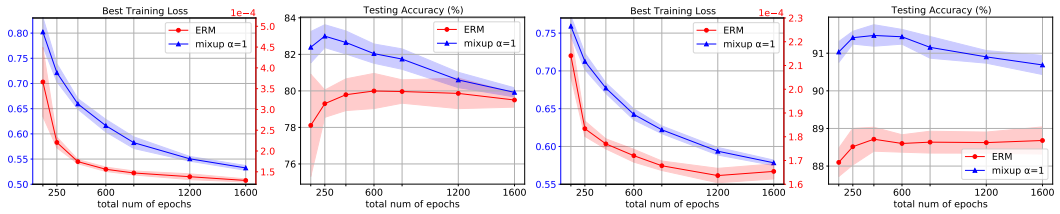


(a) Train loss (30% data) (b) Test acc (30% data) (c) Train loss (100% data) (d) Test acc (100% data)

Figure 10: Results of the recorded training losses and testing accuracies of training ResNet34 on SVHN training set (100% data and 30% data) without data augmentation.

In addition, we have trained VGG16 on the CIFAR10 training set (100% data and 30% data) for up to in total 1600 epochs without data augmentation. The results are provided in Figure 11. In

both cases, over-training VGG16 with either ERM or Mixup can gradually reduce the best achieved training loss. However, the testing accuracy of the Mixup-trained network also decreases, while that of the ERM-trained network has no significant change.

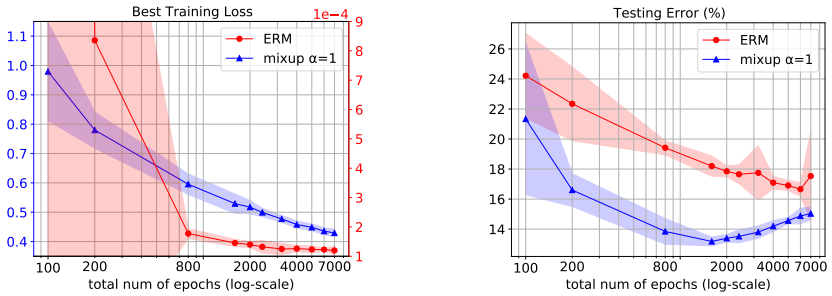


(a) Train loss (30% data) (b) Test acc (30% data) (c) Train loss (100% data) (d) Test acc (100% data)

Figure 11: Results of the recorded training losses and testing accuracies of training VGG16 on CIFAR10 training set (30% data and 100% data) without data augmentation.

B.2 Results of Over-Training with Data Augmentation

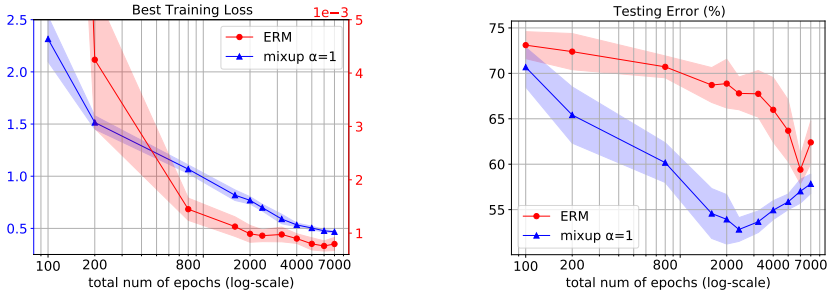
We have also carried out the over-training experiments with data augmentation applied including "random crop" and "horizontal flip". We train ResNet18 on 10% of the CIFAR10 training set with data augmentation for up to in total 7000 epochs. The results are given in Figure 12. In this case, the Mixup-trained models also produce a U-shaped generalization curve along with the total numbers of epochs. However, the inflection point of the U-shaped curve comes much later compared to that in any of the previous experiments where data augmentation is not applied.



(a) Training Loss (10% data) (b) Testing Error (10% data)

Figure 12: Results of the recorded training losses and testing errors of over-training ResNet18 on 10% of the CIFAR10 training set with data augmentation.

Also, the results of over-training ResNet34 on 10% of the CIFAR100 training set for up to in total 7000 epochs are given in Figure 13. In this case, the similar phenomenons as in the previous case has also been observed.

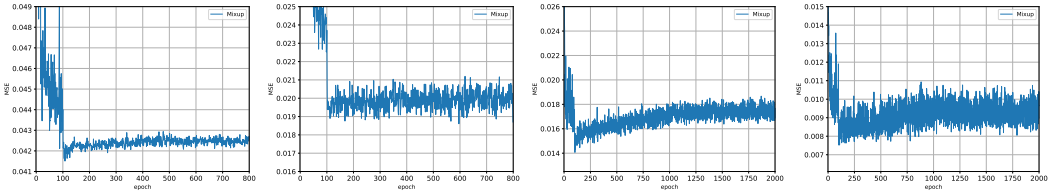


(a) Training Loss (10% data) (b) Testing Error (10% data)

Figure 13: Results of the recorded training losses and testing errors of over-training ResNet34 on 10% of the CIFAR100 training set with data augmentation.

B.3 RESULTS OF MEAN SQUARE ERROR LOSS

We also conduct Mixup training experiments by using mean square error loss on CIFAR10 and SVHN. Figure 14 shows that the U-shaped behavior also holds for the MSE loss. The learning rate is divided by 10 at epoch 100 and 150.

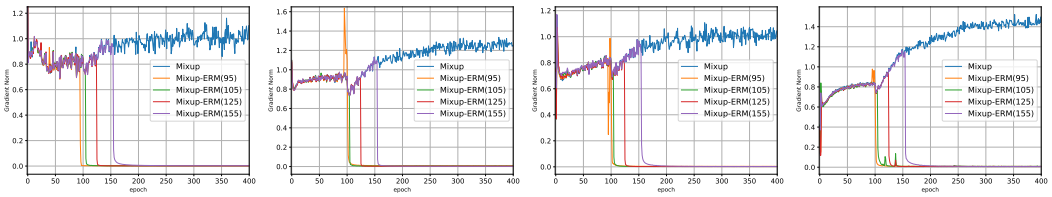


(a) CIFAR10 (30%) (b) CIFAR10 (100%) (c) SVHN (30%) (d) SVHN (100%)

Figure 14: Dynamics of MSE during Mixup training.

B.4 GRADIENT NORM VANISHES WHEN CHANGING MIXUP TO ERM

In Figure 8, we know that the gradient norm of Mixup training will not vanish at the end of training and will even explode to a very high value. In contrast, ERM will have zero gradient norm at the end. Figure 15 shows that the gradient norm will instantly become zero. This is because the "clean patterns" are already learned by Mixup trained neural networks and the original data will not provide any useful gradient signal. This further justifies that the latter stage of Mixup training is only for memorizing noisy data.



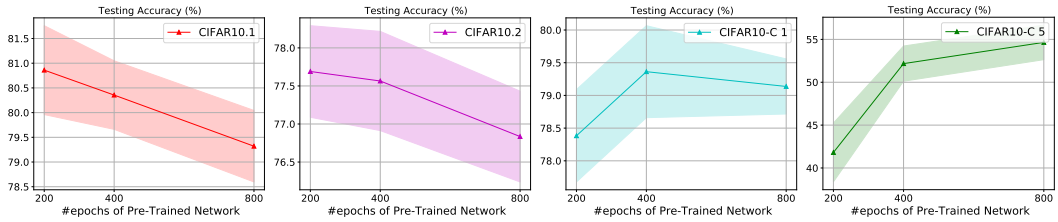
(a) CIFAR10 (30%) (b) CIFAR10 (100%) (c) SVHN (30%) (d) SVHN (100%)

Figure 15: Dynamics of Gradient Norm when changing Mixup training to ERM training.

B.5 VALIDATION RESULTS ON COVARIANCE-SHIFT DATASETS

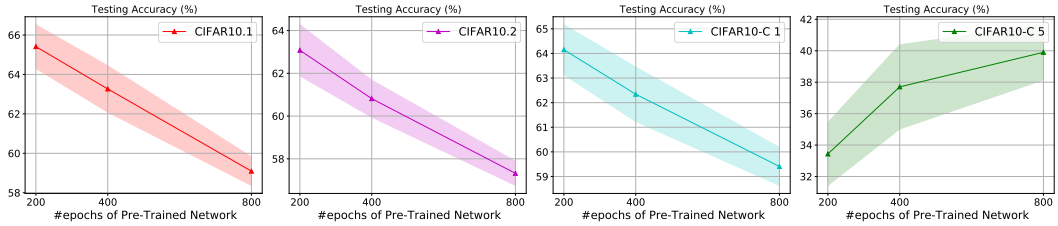
Recall Figures 2 and 3, we have seen that as we increase the training epochs ($200 \to 400 \to 800$), the local minima on the loss landscape (measure by the real training data) of the Mixup-trained model gradually becomes sharper. To validate the regular pattern of the relationship between the minima flatness and the generalization behavior on the covariate-shift datasets, we have ran some of the Mixup-trained ResNet18 networks and tested their accuracies on CIFAR10.1 (Recht et al., 2018), CIFAR10.2 (Lu et al., 2020) and CIFAR10-C (Hendrycks & Dietterich, 2019) using Gaussian noise with severity 1 and 5 (denoted by CIFAR10-C-1 and CIFAR10-C-5). The results of the models pre-trained on 100% CIFAR10 are given in Figure 16, and the results of the models pre-trained on 30% CIFAR10 are given in Figure 17.

From the results, it is seen that with training epochs increases, the testing performance on the models on CIFAR10.1 and CIFAR10.2 decreases, taking a similar trend as our results in standard testing sets (i.e., the original CIFAR10 testing sets without covariate shift.) But on CIFAR10-C, this behaviour is not observed. In particular, the performance on CIFAR10-C-5 continues to improve over training iterations. This seems to suggest that the flatness of empirical-risk loss landscape may impact generalization to covariate-shift datasets in more complex ways, possibly depending on the nature and structure of the covariate shift.



(a) CIFAR10.1 (b) CIFAR10.2 (c) CIFAR10-C-1 (d) CIFAR10-C-5

Figure 16: Models Pre-Trained on 100% CIFAR10 (without data augmentation)



(a) CIFAR10.1 (b) CIFAR10.2 (c) CIFAR10-C-1 (d) CIFAR10-C-5

Figure 17: Models Pre-Trained on 30% CIFAR10 (without data augmentation)

B.6 Investigation of the Impact of RegMixup in Over-Training

RegMixup is a variant algorithm of Mixup proposed by (Pinto et al., 2022). For each synthetic example $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ formulated by $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}', \mathbf{y}')$, RegMixup minimizes the following loss

$$\ell_{\text{CE}}(\theta, \mathbf{x}, \mathbf{y}) + \eta \ell_{\text{CE}}(\theta, \tilde{\mathbf{x}}, \tilde{\mathbf{y}})$$

where $\ell_{\text{CE}}(\cdot)$ denotes the cross-entropy loss, and η is non-negative. The authors show that RegMixup can improve generalization on both in-distribution and covariate-shift datasets, and that it can also improve the out-of-distribution robustness. To validate the performance of RegMixup in the over-training scenario, we have trained ResNet18 using RegMixup with a few different settings of η. The network is trained on CIFAR10 without data augmentation for up to in total 1200 epochs. The results are given in Figure 18.

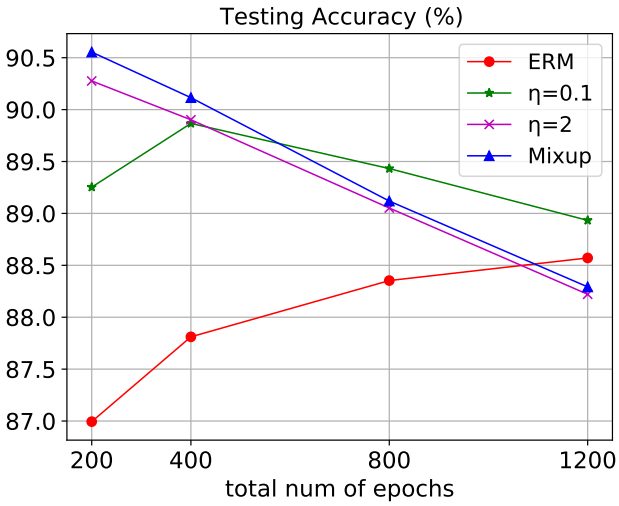


Figure 18: Results of Over-Training ResNet18 on CIFAR10 (without data augmentation) with RegMixup. Green: $\eta = 0.1$; Purple: $\eta = 2$; Red: ERM; Blue: standard Mixup.

The results show that when $\eta = 2$, RegMixup performs nearly identically as standard Mixup in the over-training scenario. When $\eta = 0.1$, RegMixup postpones the presenting of the turning point, and in the large epochs it outperforms standard Mixup. However, the phenomenon that the generalization performance of the trained model degrades with over-training still exists.

C Experiment Settings for the Teacher-Student Toy Example

The dimension of the input feature $d_0 = 10$. The teacher network is a two-layer neural networks with **Tanh** as the activation function, and the width of hidden layer is 5. The student network is also a two-layer neural network with **Tanh**, where we fix the parameters in the first layer and only train the second layer. The hidden layer dimension is 100 (i.e. $d = 100$). For the value of λ, we either draw from **Beta(1,1)** distribution in each epoch or fix it to 0.5 in each epoch. We choose $n = 20$ (so that $n < d$ is the overparameterized regime and $m \geq n^2 > d$ is the underparameterized regime) and the learning rate is 0.1. We use full-batch gradient descent to train the student network with MSE. Notice that here the "full-batch" indicates the batch size is equal to n, so that we can fairly compare the fixed λ and random λ. More details can be found in our codes.

C.1 Ablation Study: Effect of Fixed Mixing Coefficient

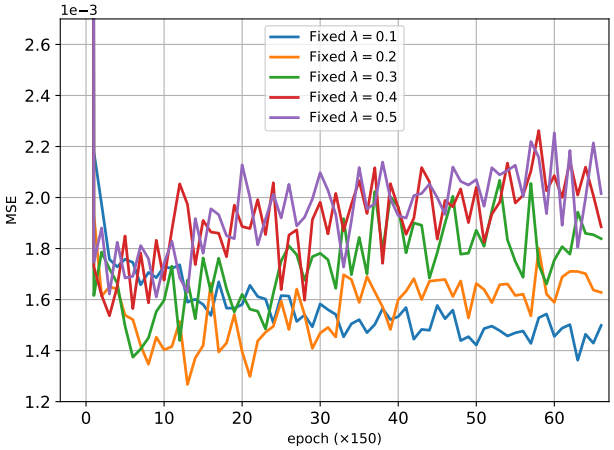


Figure 19: Results of the ablation study on λ.

We compare different fixed values of λ for the teacher-student setting, and the results are shown in Figure 19. Notably, when λ is close to 0.5, i.e. having higher noise level, the inflection point of testing error will come earlier, which is consistent with our theoretical results.

D Omitted Definitions and Proofs

Definition D.1 (Total Variation)**.** *The total variation between two probability measures P and Q is $\mathrm{TV}(P, Q) \triangleq \sup_E |P(E) - Q(E)|$, where the supremum is over all measurable set E.*

Lemma D.1 ((Levin & Peres, 2017, Proposition 4.2))**.** *Let P and Q be two probability distributions on $\mathcal{X}$. If $\mathcal{X}$ is countable, then*

$$\mathrm{TV}(P, Q) = \frac{1}{2} \sum_{x \in \mathcal{X}} |P(x) - Q(x)| .$$

Lemma D.2 (Coupling Inequality (Levin & Peres, 2017, Proposition 4.7))**.** *Given two random variables X and Y with probability distributions P and Q, any coupling $\hat{P}$ of P, Q satisfies*

$$\mathrm{TV}(P, Q) \leq \hat{P}(X \neq Y).$$

D.1 PROOF OF LEMMA 3.1

Proof. We first prove the closed-form of the cross-entropy loss's lower bound. For any two discrete distributions P and Q defined on the same probability space $\mathcal{Y}$, the KL divergence of P from Q is defined as follows:

$$D_{\mathrm{KL}}(P\|Q) := \sum_{y\in\mathcal{Y}} P(y)\log\left(\frac{P(y)}{Q(y)}\right). \tag{4}$$

It is non-negative and it equals 0 if and only if $P = Q$.

Let's denote the ith element in $f_\theta(x)$ by $f_\theta(x)_i$. By adapting the definition of the cross-entropy loss, we have:

$$\begin{aligned}
\ell(\theta,(\mathbf{x},\mathbf{y})) &= -\mathbf{y}^{\mathrm{T}}\log\big(f_\theta(\mathbf{x})\big)\\
&= -\sum_{i=1}^{K} y_i \log\big(f_\theta(\mathbf{x})_i\big)\\
&= -\sum_{i=1}^{K} y_i \log\left(\frac{f_\theta(\mathbf{x})_i}{y_i} y_i\right)\\
&= -\sum_{i=1}^{K} y_i \log\frac{f_\theta(\mathbf{x})_i}{y_i} - \sum_{i=1}^{C} y_i \log y_i\\
&= D_{\mathrm{KL}}\big(\mathbf{y}\|f_\theta(\mathbf{x})\big) + \mathcal{H}(\mathbf{y})\\
&\geq \mathcal{H}(\mathbf{y}),
\end{aligned} \tag{5}$$

where the equality holds if and only if $f_\theta(\mathbf{x}) = \mathbf{y}$. Here $\mathcal{H}(\mathbf{y}) := \sum_{i=1}^{C} y_i \log y_i$ is the entropy of the discrete distribution $\mathbf{y}$. Particularly in ERM training, since $\mathbf{y}$ is one-hot, by definition its entropy is simply 0. Therefore the lower bound of the empirical risk is given as follows.

$$\hat{R}_S(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(\theta,(\mathbf{x},\mathbf{y})) \geq 0 \tag{6}$$

The equality holds if $f_\theta(\mathbf{x}_i) = \mathbf{y}_i$ is true for each $i \in \{1, 2, \cdots, n\}$.

We then prove the lower bound of the expectation of empirical Mixup loss. From Eq. (5), the lower bound of the general Mixup loss for a given λ is also given by:

$$\begin{aligned}
\ell(\theta,(\tilde{\mathbf{x}},\tilde{\mathbf{y}})) &\geq \mathcal{H}(\tilde{\mathbf{y}})\\
&= -\sum_{i=1}^{C} y_i \log y_i\\
&= -\big(\lambda\log\lambda + (1-\lambda)\log(1-\lambda)\big).
\end{aligned} \tag{7}$$

if $(\tilde{\mathbf{x}},\tilde{\mathbf{y}})$ is formulated via cross-class mixing. Recall the definition of the Mixup loss,

$$\hat{R}_{\tilde{S}}(\theta,\alpha) = \mathop{\mathbb{E}}_{\lambda\sim Beta(\alpha,\alpha)} \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} \ell(\theta,(\tilde{\mathbf{x}},\tilde{\mathbf{y}})), \tag{8}$$

we can exchange the computation of the expectation and the empirical average:

$$\hat{R}_{\tilde{S}}(\theta,\alpha) = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} \mathop{\mathbb{E}}_{\lambda\sim Beta(\alpha,\alpha)} \ell(\theta,(\tilde{\mathbf{x}},\tilde{\mathbf{y}})) \tag{9}$$

Note that when $\alpha = 1$, $Beta(\alpha,\alpha)$ is simply the uniform distribution in the interval $[0,1]$: $U(0,1)$. Using the fact that the probability density of $U(0,1)$ is constantly 1 in the interval $[0,1]$, the lower

bound of $\mathop{\mathbb{E}}_{\lambda\sim Beta(1,1)} \ell(\theta,(\tilde{\mathbf{x}},\tilde{\mathbf{y}}))$ where $\mathbf{y}\neq\mathbf{y}'$ is given by:

$$\begin{aligned}
\mathop{\mathbb{E}}_{\lambda\sim Beta(1,1)} \ell(\theta,(\tilde{\mathbf{x}},\tilde{\mathbf{y}})) &\geq -\mathop{\mathbb{E}}_{\lambda\sim U(0,1)} \big(\lambda\log\lambda+(1-\lambda)\log(1-\lambda)\big) \\
&= -\int_0^1 \lambda\log\lambda+(1-\lambda)\log(1-\lambda)\,d\lambda \\
&= -2\int_0^1 \lambda\log\lambda\,d\lambda \\
&= -2\left(\log\lambda\int_0^1\lambda\,d\lambda-\int_0^1\frac{1}{\lambda}\left(\int_0^1\lambda\,d\lambda\right)d\lambda\right) \\
&= -2\left(\frac{\lambda^2\log\lambda}{2}-\frac{\lambda^2}{4}\right)\Bigg|_0^1 \\
&= 0.5
\end{aligned} \tag{10}$$

Note that if the synthetic example is formulated via in-class mixing, the synthetic label is still one-hot, thus the lower bound of its general loss is 0. In a balanced C-class training set, with probability $\frac{1}{C}$ the in-class mixing occurs. Therefore, the lower bound of the overall Mixup loss is given as follows,

$$\hat{R}_{\widetilde{S}}(\theta,\alpha=1)\geq\frac{C-1}{2C} \tag{11}$$

The equality holds if $f_\theta(\tilde{\mathbf{x}})=\tilde{\mathbf{y}}$ is true for each synthetic example $(\tilde{\mathbf{x}},\tilde{\mathbf{y}})\in\widetilde{S}$. This completes the proof. □

D.2 Proof of Theorem 5.1

Proof. By the coupling inequality i.e. Lemma D.2, we have

$$\mathrm{TV}(P(\widetilde{Y}_{\mathrm{h}}|\widetilde{X}),P(\widetilde{Y}^*_{\mathrm{h}}|\widetilde{X}))\leq P(\widetilde{Y}_{\mathrm{h}}\neq\widetilde{Y}^*_{\mathrm{h}}|\widetilde{X}),$$

Since $\mathrm{TV}(P(\widetilde{Y}_{\mathrm{h}}|\widetilde{X}),P(Y|X))=\mathrm{TV}(P(\widetilde{Y}_{\mathrm{h}}|\widetilde{X}),P(\widetilde{Y}^*_{\mathrm{h}}|\widetilde{X}))$, then the first inequality is straightforward.

For the second inequality, by Lemma D.1, we have

$$\begin{aligned}
\mathrm{TV}(P(\widetilde{Y}_{\mathrm{h}}|\widetilde{X}),P(\widetilde{Y}^*_{\mathrm{h}}|\widetilde{X})) =&\frac{1}{2}\sum_{j=1}^{C}\left|P(\widetilde{Y}^*=j|\widetilde{X})-P(\widetilde{Y}=j|\widetilde{X})\right| \\
=&\frac{1}{2}\sum_{j=1}^{C}\left|f_j(\widetilde{X})-((1-\lambda)f_j(X)+\lambda f_j(X'))\right| \\
\geq&\sup_j\frac{1}{2}\left|f_j(\widetilde{X})-((1-\lambda)f_j(X)+\lambda f_j(X'))\right|.
\end{aligned}$$

This completes the proof. □

D.3 Proof of Lemma 5.1

Proof. The proof here is trivial. The ordinary differential equation of Eq. (2) (Newton's law of cooling) has the closed form solution

$$\theta_t=\widetilde{\Phi}^\dagger\widetilde{\mathbf{Y}}+(\theta_0-\widetilde{\Phi}^\dagger\widetilde{\mathbf{Y}})e^{-\frac{\eta}{m}\widetilde{\Phi}\widetilde{\Phi}^Tt}. \tag{12}$$

Recall that $\widetilde{\mathbf{Y}}=\widetilde{\mathbf{Y}}^*+\mathbf{Z}$,

$$\begin{aligned}
\theta_t =&\widetilde{\Phi}^\dagger\left(\widetilde{\mathbf{Y}}^*+\mathbf{Z}\right)+(\theta_0-\widetilde{\Phi}^\dagger\left(\widetilde{\mathbf{Y}}^*+\mathbf{Z}\right))e^{-\frac{\eta}{m}\widetilde{\Phi}\widetilde{\Phi}^Tt} \\
=&\widetilde{\Phi}^\dagger\widetilde{\mathbf{Y}}^*+\widetilde{\Phi}^\dagger\mathbf{Z}+\left(\theta_0-\widetilde{\Phi}^\dagger\widetilde{\mathbf{Y}}^*\right)e^{-\frac{\eta}{m}\widetilde{\Phi}\widetilde{\Phi}^Tt}-\widetilde{\Phi}^\dagger\mathbf{Z}e^{-\frac{\eta}{m}\widetilde{\Phi}\widetilde{\Phi}^Tt} \\
=&\theta^*+(\theta_0-\theta^*)e^{-\frac{\eta}{m}\widetilde{\Phi}\widetilde{\Phi}^Tt}+(\mathbf{I}_d-e^{-\frac{\eta}{m}\widetilde{\Phi}\widetilde{\Phi}^Tt})\theta^{\mathrm{noise}},
\end{aligned}$$

which concludes the proof. □

D.4 PROOF OF THEOREM 5.2

Proof. We first notice that

$$
\begin{aligned}
R_t =& \mathbb{E}_{\theta_t,X,Y} \left|\left|\theta_t^T \phi(X) - Y\right|\right|_2^2 \\
=& \mathbb{E}_{\theta_t,X,Y} \left|\left|\theta_t^T \phi(X) - \theta^{*T}\phi(X) + \theta^{*T}\phi(X) - Y\right|\right|_2^2 \\
=& \mathbb{E}_{\theta_t,X} \left|\left|\theta_t^T \phi(X) - \theta^{*T}\phi(X)\right|\right|_2^2 + \mathbb{E}_{X,Y} \left|\left|\theta^{*T}\phi(X) - Y\right|\right|_2^2 + 2\mathbb{E}_{\theta_t,X,Y}\langle \theta_t^T\phi(X) - \theta^{*T}\phi(X), \theta^{*T}\phi(X) - Y\rangle \\
\leq& \mathbb{E}_X \left|\left|\phi(X)\right|\right|_2^2 \mathbb{E}_{\theta_t} \left|\left|\theta_t^T - \theta^{*T}\right|\right|_2^2 + R^* + 2\sqrt{\mathbb{E}_{\theta_t,X} \left|\left|\theta_t^T\phi(X) - \theta^{*T}\phi(X)\right|\right|_2^2 \mathbb{E}_{X,Y} \left|\left|\theta^{*T}\phi(X) - Y\right|\right|_2^2} \\
\leq& \frac{C_1}{2}\mathbb{E}_{\theta_t} \left|\left|\theta_t^T - \theta^{*T}\right|\right|_2^2 + R^* + 2\sqrt{\frac{C_1 R^*}{2}\mathbb{E}_{\theta_t} \left|\left|\theta_t^T - \theta^{*T}\right|\right|_2^2}, \qquad (13)
\end{aligned}
$$

where the first inequality is by the Cauchy–Schwarz inequality and the second inequality is by the assumption.

Recall Eq. (3),

$$
\theta_t - \theta^* = (\theta_0 - \theta^*)e^{-\frac{\eta}{m}\widetilde{\Phi}\widetilde{\Phi}^T t} + (\mathbf{I}_d - e^{-\frac{\eta}{m}\widetilde{\Phi}\widetilde{\Phi}^T t})\widetilde{\Phi}^\dagger \mathbf{Z}.
$$

By eigen-decomposition we have

$$
\frac{1}{m}\widetilde{\Phi}\widetilde{\Phi}^T = V\Lambda V^T = \sum_{k=1}^{d} \mu_k v_k v_k^T,
$$

where $\{v_k\}_{k=1}^d$ are orthonormal eigenvectors and $\{\mu_k\}_{k=1}^d$ are corresponding eigenvectors.

Then, for each dimension k,

$$
(\theta_{t,k} - \theta_k^*)^2 \leq 2(\theta_{0,k} - \theta_k^*)^2 e^{-2\eta\mu_k t} + 2(1 - e^{-\eta\mu_k t})^2 \frac{mC_2}{m\mu_k},
$$

Taking expectation over θ_0 for both side, we have

$$
\mathbb{E}_{\theta_0}(\theta_{t,k} - \theta_k^*)^2 \leq 2(\xi_k^2 + \theta_k^{*2})e^{-2\eta\mu_k t} + 2(1 - e^{-\eta\mu_k t})^2 \frac{C_2}{\mu_k}. \qquad (14)
$$

Notich that the RHS in Eq. 14 first monotonically decreases and then monotonically increases, so the maximum value of RHS is achieved either at $t = 0$ or $t \to \infty$. That is,

$$
\mathbb{E}_{\theta_0} \left|\left|\theta_t^T - \theta^{*T}\right|\right|_2^2 \leq \sum_{k=1}^{d} 2\max\{\xi_k^2 + \theta_k^{*2}, \frac{C_2}{\mu_k}\}. \qquad (15)
$$

Plugging Eq. 14 and Eq. 15 into Eq. 13, we have

$$
\begin{aligned}
R_t \leq& \frac{C_1}{2}\mathbb{E}_{\theta_t} \left|\left|\theta_t^T - \theta^{*T}\right|\right|_2^2 + R^* + 2\sqrt{\frac{C_1 R^*}{2}\mathbb{E}_{\theta_t} \left|\left|\theta_t^T - \theta^{*T}\right|\right|_2^2} \\
\leq& R^* + C_1 \sum_{k=1}^{d} (\xi_k^2 + \theta_k^{*2})e^{-2\eta\mu_k t} + (1 - e^{-\eta\mu_k t})^2 \frac{C_2}{\mu_k} + 2\sqrt{C_1 R^* \zeta},
\end{aligned}
$$

where $\zeta = \sum_{k=1}^d \max\{\xi_k^2 + \theta_k^{*2}, \frac{C_2}{\mu_k}\}$. This concludes the proof. □