MixupE: Understanding and Improving Mixup from Directional Derivative Perspective

Yingtian Zou [*1] **Vikas Verma** [*2,3] **Sarthak Mittal**[2] **Wai Hoh Tang**[1] **Hieu Pham**[4] **Juho Kannala**[3]
Yoshua Bengio[2] **Arno Solin**[3] **Kenji Kawaguchi**[1]

[1]National University of Singapore, Singapore
[2]Universite de Montreal, Mila, Canada
[3]Aalto University, Finland
[4]Google Brain, USA

Abstract

Mixup is a popular data augmentation technique for training deep neural networks where additional samples are generated by linearly interpolating pairs of inputs and their labels. This technique is known to improve the generalization performance in many learning paradigms and applications. In this work, we first analyze Mixup and show that it implicitly regularizes infinitely many directional derivatives of all orders. Based on this new insight, we propose an improved version of Mixup, theoretically justified to deliver better generalization performance than the vanilla Mixup. To demonstrate the effectiveness of the proposed method, we conduct experiments across various domains such as images, tabular data, speech, and graphs. Our results show that the proposed method improves Mixup across multiple datasets using a variety of architectures, for instance, exhibiting an improvement over Mixup by 0.8% in ImageNet top-1 accuracy. The code is available at https://github.com/oneHuster/MixupE.

1 INTRODUCTION

Deep Neural Networks (DNNs) represent a class of very powerful function approximators, and large-scale DNNs have achieved state-of-the-art performance in many application areas such as computer vision [Krizhevsky et al., 2012], natural language understanding [Devlin et al., 2018], speech recognition [Hinton et al., 2012], reinforcement learning [Silver et al., 2016], and natural sciences [Jumper et al., 2021]. In a supervised learning setting, DNNs are typically trained to minimize their average error on the training samples. This training principle is known as Empirical Risk Minimization (ERM) [Vapnick, 1998].

[*]indicates equal contribution.

Although being a simple training principle, training neural networks with ERM has a major problem: in the absence of regularization techniques, instead of learning meaningful concepts, neural networks trained with ERM are prone to *memorize* training data [Arpit et al., 2017]. This results in poor generalization to test samples, which come from a distribution slightly different from the training samples. To address this limitation of ERM, Mixup [Zhang et al., 2018] has recently been proposed as an alternative training principle. In a nutshell, instead of training a neural network on individual samples and their corresponding outputs, Mixup trains a neural network on the linear interpolation of the samples and the corresponding linear interpolation of the outputs. It fosters a smoother decision boundary and reduces the risk of overfitting. Therefore, understanding its implicit regularization helps shed light on generalization.

Mathematically, let us suppose that $\mathbf{x}_i$ and $\mathbf{x}_j$ are input vectors corresponding to two randomly drawn samples i and j from the training distribution, and $\mathbf{y}_i$ and $\mathbf{y}_j$ are their one-hot encoded labels. Then, Mixup constructs a training sample as $\tilde{\mathbf{x}} = \lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_j$ and $\tilde{\mathbf{y}} = \lambda \mathbf{y}_i + (1-\lambda)\mathbf{y}_j$, where $\lambda \in [0, 1]$. Training with this kind of synthetic samples encourages the model to learn a function where linear interpolation in the input vectors leads to the linear interpolation of the corresponding targets. This kind of constraint limits the model complexity, thus limiting their ability to memorize training samples. Mixup can be interpreted as a data-agnostic *data augmentation* technique that does not require expert knowledge to create additional training samples. Mixup can also be interpreted from the viewpoint of the *Vicinal Risk Minimization* (VRM) principle [Chapelle et al., 2000]. In this view, Mixup proposes a generic vicinal distribution based on the interpolation of training samples and their associated targets, and the additional training samples are drawn from such vicinal distribution around each training sample [Zhang et al., 2018].

Despite its simplicity and minimal computation overhead, Mixup and its variants have been shown to achieve state-of-the-art in many tasks such as but not limited to, image

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

classification [Yun et al., 2019, Kim et al., 2020, Faramarzi et al., 2020], object detection [Jeong et al., 2021], speech recognition [Lam et al., 2020, Tomashenko et al., 2018], text classification [Guo et al., 2019, Zhang et al., 2020], and medical image segmentation [Panfilov et al., 2019]. Recently, Mixup was theoretically analyzed and shown to be approximately equivalent to adding a second-order regularization term to the standard loss function [Zhang et al., 2021]. However, if the benefit of Mixup can be explained by a second-order regularization, the following natural question arises: why can we not replace Mixup with this second-order regularization directly? Unfortunately, the answer is no because the second-order terms are input-specific, thus yielding a complicated form.

In this paper, we show that Mixup is equivalent to implicitly adding infinitely many regularization terms on the directional derivatives of all orders instead of a complex second-order form as [Zhang et al., 2021] for ERM. Our analysis provides a feasible insight to design the regularization in practice. Based on this novel insight, this paper proposes to explicitly enhance the implicit regularization effect of Mixup on the directional derivatives. Instead of computing all infinite regularization terms, we efficiently approximate the dominant term using accessible results during each forward propagation, ensuring computational efficiency. We name this method as *MixupE* (**Mixup E**nhanced). Furthermore, we give a generalization guarantee of *MixupE*, which reveals that it achieves lower complexity compared to vanilla Mixup. Figure 1 shows the training and test loss of ERM, Mixup, and *MixupE*. We can see that *MixupE* has higher training loss, implying that it works as a stronger regularizer than Mixup and ERM. This subsequently results in better generalization (i.e. lower test loss) than Mixup and ERM.

To understand the benefits of *MixupE* empirically, we conduct experiments on a variety of datasets, such as images, tabular data and speech data, using various architectures such as LeNet [LeCun et al., 1998], VGG [Simonyan and Zisserman, 2014], ResNet [He et al., 2016], Vision Transformer [ViT, Dosovitskiy et al., 2020], and CoAtNet [Dai et al., 2021]. In our experiments, we consistently see that *MixupE* has better generalization error than Mixup and ERM, as well as the robustness of test set deformation.

2 METHODS

In this section, we derive our method with a new mathematical understanding of Mixup. We begin in Section 2.1 with the notation used to present our theory and method. Mixup is then shown to implicitly regularize infinitely many directional derivatives of all orders in Section 2.2. This theoretical insight allows us to enhance the regularization effect on the directional derivatives. We also demonstrate that the proposed explicit regularization reduces the algorithmic complexity, thus improving the generalization. In Section 2.3, we present an algorithm to strengthen the regularization effect of Mixup based on a theoretical derivation and justify the theoretical improvement over Mixup.

2.1 NOTATION

We denote the input and output pair as $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ and $\mathbf{y} \in \mathcal{Y} \subseteq \mathbb{R}^C$, respectively. Let $f_\theta(\mathbf{x}) \in \mathbb{R}^C$ be the output of the logits (i.e., the last layer before the softmax or sigmoid) of the model parameterized by θ. We use $\ell(\theta, (\mathbf{x}, \mathbf{y})) = h(f_\theta(\mathbf{x})) - \mathbf{y}^\top f_\theta(\mathbf{x})$ to denote the loss function where

$$h(f_\theta(\mathbf{x})) = \begin{cases} \log\left(\sum_j \exp(f_\theta(\mathbf{x})_{(j)})\right) & \text{Softmax} \\ \log\left(1 + \exp\left(f_\theta(\mathbf{x})\right)\right) & \text{Sigmoid} \end{cases} \tag{1}$$

where $\exp(\cdot)$ is the exponential function applied to every element. Let $g(\cdot)$ be the activation function. We use $\mathbf{x}_{(i)}$ to index i-th element of the vector $\mathbf{x}$ and $\mathbf{x}_j$ to represent j-th variable in a set. $\mathbf{J}^k$ denotes the k-th Jacobian operator.

Mixup Given a training dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ of size n with $\mathbf{x}_i \in \mathcal{X}$ and $\mathbf{y}_i \in \mathcal{Y}$, we define the Mixup version of the input and output pair by $\tilde{\mathbf{x}}_{i,j}(\lambda) = \lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_j$ and $\tilde{\mathbf{y}}_{i,j}(\lambda) = \lambda \mathbf{y}_i + (1-\lambda)\mathbf{y}_j$ with the Mixup coefficient $\lambda \in [0, 1]$. Then, we denote the standard empirical loss by $L_n^{std}(\theta, S) = \frac{1}{n}\sum_{i=1}^n l(\theta, (\mathbf{x}_i, \mathbf{y}_i))$ and the Mixup loss by

$$L_n^{\text{mix}}(\theta, S) := \frac{1}{n^2} \sum_{i,j=1}^n \mathop{\mathbb{E}}_{\lambda \sim \text{Beta}(\alpha,\beta)} l(\theta, \tilde{\mathbf{x}}_{i,j}(\lambda), \tilde{\mathbf{y}}_{i,j}(\lambda))$$

where $\text{Beta}(\alpha, \beta)$ represents the beta distribution with its parameters $\alpha, \beta > 0$. We define a mixture of beta distributions as $\mathcal{D}_\lambda = \frac{\alpha}{\alpha+\beta}\text{Beta}(\alpha+1, \beta) + \frac{\beta}{\alpha+\beta}\text{Beta}(\beta+1, \alpha)$ where the coefficients are the drawing probabilities. Let $a_\lambda = 1 - \lambda$ and $[n] = \{1, \ldots, n\}$.

2.2 MOTIVATION FROM IMPLICIT REGULARIZATION OF MIXUP

Here we show a theorem that optimizing the vanilla Mixup loss induces an implicit regularization on directional derivatives of the model f_θ.

Theorem 1. *Let $\ell(\theta, (\mathbf{x}, \mathbf{y})) \triangleq h(f_\theta(\mathbf{x})) - \mathbf{y}^\top f_\theta(\mathbf{x})$ be the loss function and $\forall \theta \in \Theta$ functions $f_\theta(\cdot)$ in a C^K manifold. Then the implicit regularization of Mixup is:*

$$\begin{aligned} L_n^{mix}(\theta, S) &= L_n^{std}(\theta, S) \\ &+ \frac{1}{n}\sum_{i=1}^n \mathbb{E}_{\substack{\lambda \sim \mathcal{D}_\lambda \\ \mathbf{x}' \sim \mathcal{D}_X}} \Bigg(\sum_{k=1}^K \frac{a_\lambda^k}{k!} \mathbf{J}^k_{h \circ f_\theta}(\mathbf{x}_i) \Delta_i^{\otimes k} \\ &- a_\lambda \mathbf{y}_i^\top \Delta_i + a_\lambda^K \hat{\psi}_{i,\mathbf{x}'}(a_\lambda) \Bigg) \end{aligned} \tag{2}$$

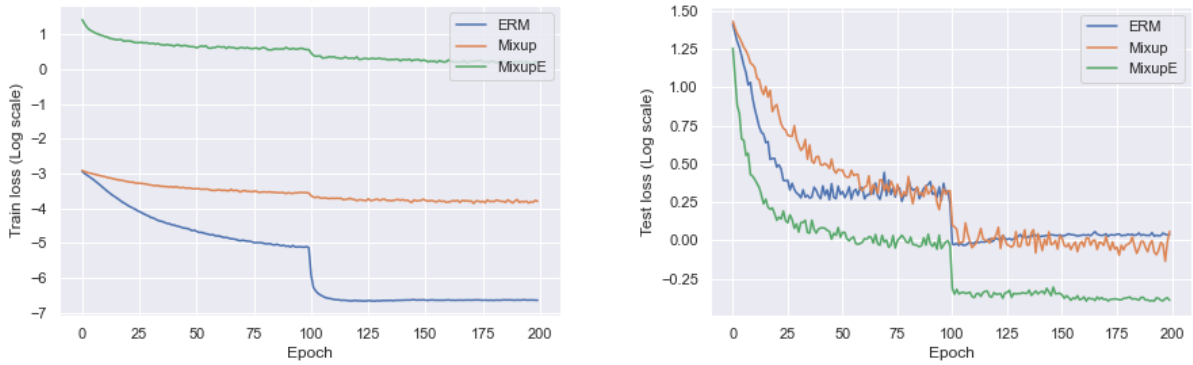

Figure 1: Comparison of train and test loss of ERM, Mixup, and *MixupE* trained with Wide-Resnet-28-10 [Zagoruyko and Komodakis, 2016] architecture. We can see that *MixupE* has a higher training loss but lower test loss than Mixup and ERM.

where $\mathbf{J}_{h\circ f_\theta}(\mathbf{x}_i) = g(f_\theta(\mathbf{x}_i))^\top \mathbf{J}_{f_\theta}(\mathbf{x}_i)$ *and*

$$\Delta_i = \sum_{k=1}^{K} \frac{a_\lambda^{k-1}}{k!} \mathbf{J}^k_{f_\theta}(\mathbf{x}_i)(\mathbf{x}' - \mathbf{x}_i)^{\otimes k} + a_\lambda^{K-1} \psi_{i,\mathbf{x}'}(a_\lambda). \quad (3)$$

Remark 1. $\hat{\psi}_{i,\mathbf{x}'}$ and $\psi_{i,\mathbf{x}'}$ are the remainder terms in Taylor expansion of order $\mathcal{O}(K)$ and with probability 1, $\lim_{a_\lambda \to 0} \hat{\psi}_{i,\mathbf{x}'}(a_\lambda) = 0$, $\lim_{a_\lambda \to 0} \psi_{i,\mathbf{x}'}(a_\lambda) = 0$. For cross-entropy loss, given input $\mathbf{z} \in \mathbb{R}^d$, $h(\mathbf{z}) = \log\left(\sum_j \exp(\mathbf{z}_{(j)})\right)$, the derivative of each element is $\frac{\partial h(\mathbf{z})}{\partial \mathbf{z}_{(t)}} = [\sum_j \exp(\mathbf{z}_{(j)})]^{-1} \exp(\mathbf{z}_{(t)}) = g(\mathbf{z})_{(t)}$, Similarly, the logistic loss has same derivative form $\frac{\partial h(\mathbf{z})}{\partial \mathbf{z}_{(i)}} = (1 + \exp(\mathbf{z}_{(i)}))^{-1} \exp(\mathbf{z}_{(i)}) = g(\mathbf{z})_{(i)}$. Therefore, for both cases, we have the Jacobian w.r.t $\mathbf{x}_i$ that $\mathbf{J}_{h\circ f_\theta}(\mathbf{x}_i) = g(f_\theta(\mathbf{x}_i))^\top \mathbf{J}_{f_\theta}(\mathbf{x}_i) \in \mathbb{R}^{1\times d}$.

The proof of Theorem 1 is given in Appendix B. Theorem 1 provides the following novel insights: (1) Implicit regularization of Mixup is to add a series of directional derivatives with ascending orders to ERM. (2) To minimize the error brought by remainder terms, we need a large expansion order K, or even infinite. Obviously, in this case, explicitly computing the regularizer involves $\mathcal{O}(K^2)$ high-order derivative terms and thus suffers a heavy computational burden. Therefore, instead of replacing Mixup with all explicit regularizers, it is more advantageous to retain Mixup with an extra regularization as it provides a computationally efficient alternative.

In this view, Theorem 1 provides a theoretical motivation to further improve Mixup by enhancing its regularization effect in terms of directional derivatives $D^1_{\theta,S}, ..., D^K_{\theta,S}$. For computational efficiency, we propose to strengthen the first-order term while letting Mixup implicitly take care of the higher-order terms. In Theorem 1, the regularization effect of Mixup on the first-order directional derivatives ($k = 1$) is captured by $D^1_{\theta,S} := \frac{1}{n}\mathbb{E}_{\lambda\sim\mathcal{D}_\lambda}[a_\lambda] \sum_{i=1}^n q(\mathbf{x}_i)$, where

$$q(\mathbf{x}_i) = (g(f_\theta(\mathbf{x}_i)) - \mathbf{y}_i)^\top \mathbf{J}_{f_\theta}(\mathbf{x}_i)(\mathbb{E}[\mathbf{x}'] - \mathbf{x}_i) \quad (4)$$

and $\mathbf{J}_{f_\theta}(\mathbf{x}_i) \in \mathbb{R}^{C\times d}$. Since $0 < a_\lambda < 1$ and small enough, the first-order derivative $D^1_{\theta,S}$ dominates the rest terms $D^k_{\theta,S}$ ($k > 1$). Therefore, it turns out to be a stitch in time saves nine if we only use $D^1_{\theta,S}$ as the regularization term. Unfortunately, computing Jacobian in deep models at each iteration step is still impractical. Furthermore, $q(\mathbf{x}_i)$ can be approximated by

$$\hat{q}(\mathbf{x}_i) = (\mathbf{y}_i - g(f_\theta(\mathbf{x}_i)))^\top f_\theta(\mathbf{x}_i), \quad (5)$$

which lessens the computational burden by removing the derivatives of f_θ. This approximation $q(\mathbf{x}_i) = \hat{q}(\mathbf{x}_i)$ holds true when $\mathbb{E}_{\mathbf{x}'\sim\mathcal{D}_X}[\mathbf{x}'] = 0$ and $\mathbf{J}_{f_\theta}(\mathbf{x}_i)\mathbf{x}_i = f_\theta(\mathbf{x}_i)$. To this end, we can normalize the training dataset with zero means to realize the first condition. The second condition can be guaranteed from the linear model, i.e. deep neural networks with ReLU activation. Thus we have the approximation that $q(\mathbf{x}_i) \approx \hat{q}(x_i)$ and $\mathbf{J}_{f_\theta}(\mathbf{x}_i)\mathbf{x}_i \approx f_\theta(\mathbf{x}_i)$.

However, there is an issue of negativity in the first-order regularization term $D^1_{\theta,S}$ of Mixup. Let $\alpha_{k,i} = (g(f(\mathbf{x}_i)_{(k)}) - \mathbf{y}_i)\zeta_{k,i}$, (4) can be rewritten as

$$q(\mathbf{x}_i) = \sum_{j=1}^{C} \alpha_{j,i} \|\mathbf{J}_{f_\theta}(\mathbf{x}_i)_{(j)}\|_2 \|\mathbb{E}_{\mathbf{x}'\sim\mathcal{D}_X}[\mathbf{x}'] - \mathbf{x}_i\|_2 \quad (6)$$

where $f(\mathbf{x}_i)_{(j)}$ is the j-th coordinate of $f(\mathbf{x}_i)$ and coefficient $\zeta_{j,i}$ is the cosine similarity between j-th row vector $\mathbf{J}_{f_\theta}(\mathbf{x}_i)_{(j)}$ and $\mathbb{E}_{\mathbf{x}'\sim\mathcal{D}_X}[\mathbf{x}'] - \mathbf{x}_i$. If $\alpha_{j,i}$ is positive, then Mixup tends to minimize all first-order directional derivatives $\|\mathbf{J}_{f_\theta}(\mathbf{x}_i)_{(j)}\|_2, j \in [C]$. However, if $\alpha_{j,i}$ is negative, Mixup has an unintended effect of maximizing $\|\mathbf{J}_{f_\theta}(\mathbf{x}_i)_{(j)}\|_2$. Figure 2 shows that the minimum values of α are negative for some sample $\mathbf{x}_i$ and coordinate k in the initial phase of Mixup training. We show these values for both Preactresnet18 and Preactresnet50 architectures on the CIFAR-10 dataset.

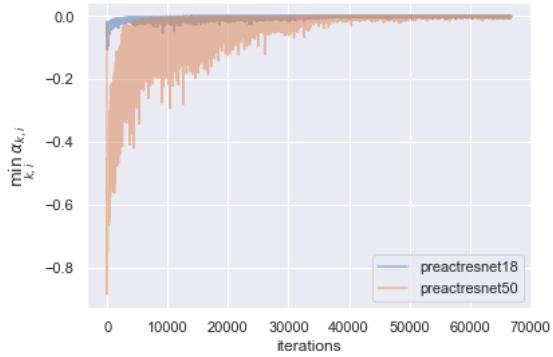

Figure 2: Minimum value of α over the coordinate k and sample i for different iterations during the training.

Algorithm 1 Algorithm of *MixupE*

Require: Model $f_\theta : \mathbb{R}^d \to \mathbb{R}^C$. Hyperparameters α, β for beta distribution (mean $\bar{\lambda}$). Loss function ℓ.

while Training epochs $< N$ **do**
  for Batch of data $X \in \mathbb{R}^{m\times d}, Y \in \mathbb{R}^{m\times C}$ in training set of size m **do**
    Sample $\lambda \sim \text{Beta}(\alpha, \beta)$
    Mixup data with $\tilde{X}, \tilde{Y} \leftarrow \lambda(X, Y) + a_\lambda \text{Permute}(X, Y)$
    Mixup Loss $L_n^{mix}(\theta, X) = \ell(f_\theta(\tilde{X}), \tilde{Y})$
    Compute first-order directional derivatives that $\hat{q}(X) = f_\theta(X) \otimes (Y - \text{Softmax}(f_\theta(X)))$
    Get additional loss $R(\theta, X)$ via (5)
    $\mathcal{L} \leftarrow \hat{\eta}\left(L_n^{mix}(\theta, S) + \eta R(\theta, S)\right)$ via (7)
    Optimize parameters θ with loss $\mathcal{L}$
  end for
end while

2.3 PROPOSED MIXUPE

To avoid the unintended effect of maximizing $\|\mathbf{J}_{f_\theta}(\mathbf{x}_i)_{(j)}\|_2$ in the initial phase of Mixup training, the proposed method uses the following alternative form to ensure the positivity and strengthen the vanilla Mixup:

$$R(\theta, S) = \frac{\mathbb{E}_{\lambda\sim\mathcal{D}_\lambda}[a_\lambda]}{n} \sum_{i=1}^{n} |\tilde{q}(\mathbf{x}_i)|, \tag{7}$$

where $\tilde{q} = q$ for the accurate version and $\tilde{q} = \hat{q}$ for the approximate version. The functions q and $\hat{q}$ are defined in equations (4) and (5). The approximate version does not require computation of the directional derivatives, and the additional computational cost is negligible because $f_\theta(\mathbf{x}_i)$ is known for the original loss and the value of $\mathbb{E}_{\lambda\sim\mathcal{D}_\lambda}[a_\lambda]$ is a fixed number over epochs: e.g., $\mathbb{E}_{\lambda\sim\mathcal{D}_\lambda}[a_\lambda] = 1 - \frac{\alpha+1}{2\alpha+1}$ when $\text{Beta}(\alpha, \alpha)$ is used for Mixup. To justify the rationality of our proposed regularization, we consider a degenerate case. Given a mixup distribution $\text{Beta}(\alpha, \beta)$, if $\alpha, \beta \to 0$, we have that $R(\theta, S) \to 0$ as $\mathbb{E}_{\lambda\sim\mathcal{D}_\lambda}[a_\lambda] \to 0$. In this case, Mixup behaves closely to ERM (i.e., mixup coefficient λ goes to 0). Then additional term $R(\theta, S)$ vanishes and is consistent with the behavior of vanilla Mixup. However, introducing $R(\theta, S)$ brings the artifacts when computing the training loss. In line with the original mixup training, we rescale the loss magnitude as before. Overall, we propose an explicit regularization $R(\theta, S)$ for vanilla Mixup to strengthen the implicit effect, namely *MixupE*, where the total loss is defined as:

$$\mathcal{L}(\theta, S) := \hat{\eta}\left(L_n^{mix}(\theta, S) + \eta R(\theta, S)\right), \tag{8}$$

$$\hat{\eta} = \frac{|L_n^{mix}(\theta, S)|}{|L_n^{mix}(\theta, S) + \eta R(\theta, S)|}, \tag{9}$$

where $\hat{\eta}$ is a scaling factor that depends on the magnitudes of $L_n^{mix}(\theta, S)$ and $R(\theta, S)$. Note that $\eta > 0$ is the only hyperparameter of the proposed method.

The whole algorithm of *MixupE* is shown in Algorithm 1. Extending Mixup to *MixupE* requires one additional forward pass on the *original* (non-mixed) sample for computing the additional loss term and requires one additional hyperparameter η in comparison to Mixup. We note that Algorithm 1 shows how to apply *MixupE* when the training sample is in the form of a fixed-shape tensor (for example, images or tabular data). When applying *MixupE* to training samples with irregular (not fixed) topology, such as graphs, sequences, and trees, we first need to project the input samples to a fixed shape hidden states using an encoder network. After this projection, *MixupE* can be applied in the usual form.

Theoretical justification: To validate the rationality of the *MixupE*, we justify the generalization improvement over Mixup. Regarding the vanilla Mixup as an original unconstrained problem, for some $\gamma > 0$, then the constraint Θ in the dual problem of *MixupE* will be

$$\Theta = \{\mathbf{x} \to f_\theta(\mathbf{x}) | \sup_{\mathbf{x}} [\hat{q}(\mathbf{x})] \le \gamma\}. \tag{10}$$

Following Zhang et al. [2021], we consider Generalized Linear Model (GLM) $h(f_\theta(\mathbf{x})) = A(\theta^\top \mathbf{x})$ for *MixupE*. For simplicity, we assume $\mathbf{y}$ can be represented by $(\theta^*)^\top \mathbf{x}$. Thus, there exists a diagonal matrix Σ of size d such that $\theta^* = \Sigma\theta$. We proved the following generalization gap:

Theorem 2. *Suppose $A(\cdot)$ is L_A-Lipchitz continuous, $\mathcal{X}, \mathcal{Y}$ and Θ are all bounded, the label of GLM is $\mathbf{y} = (\Sigma\theta)^\top \mathbf{x}$, then there exist constants $L, B > 0$, such that for all $\theta \in \Theta$, we have*

$$\mathcal{L}(\theta, S) \le \hat{\eta} L_n^{mix}(\theta, S) + \frac{2\hat{\eta}\eta L L_A (L_A + \sqrt{\gamma\overline{\sigma}})}{\overline{\sigma}\sqrt{n}} + B\sqrt{\frac{\log(1/\delta)}{2n}} \tag{11}$$

and $\overline{\sigma} = \sup_\Sigma \frac{\text{tr}(\Sigma)}{d}$ with probability at least $1 - \delta$.

See proof in Appendix C. In the context of Rademacher complexity, the smaller size of function classes will guarantee better generalization, which is reflected by the *complexity* exemplified as the second term in (11). It is worth noting that for vanilla Mixup, a general parameter constraint can be set, such as $\|\theta\|^2 \leq \gamma$. In this case, the *complexity* term can be as high as $\sqrt{\gamma \cdot \text{MaxNorm}(\mathbf{x})/n}$, depending on the maximum norm of the data. This maximum norm can often be much larger than the second term in the abovementioned inequality. The regularization of *MixupE* enforces a constraint on $\hat{q}(\mathbf{x})$, which involves the product of the "calibration" error and the representations. On the one hand, $\overline{\sigma}$ represents the average value of the maximum correction matrix to the ground truth, which can also be large. Consequently, as $\overline{\sigma} \to \infty$, the second term tends to shrink towards zero. On the other hand, the norm of the representation scales with the norm of the input in the GLM. Therefore, $\hat{q}(\mathbf{x})$ also constrains the norm of the data, denoted as $\text{Norm}(\mathbf{x})$. In summary, these observations indicate that the regularized version of Mixup, *MixupE*, leads to better generalization due to the stronger constraint imposed on the parameter space. This validates the rationale behind *MixupE* and highlights its efficacy in promoting improved generalization performance. Further details and a comprehensive discussion of these findings are in Appendix C.

3 RELATED WORK

Mixup [Zhang et al., 2018, Tokozume et al., 2017] and its numerous variants have seen remarkable success in supervised learning problems, as well as other problems such as semi-supervised learning [Verma et al., 2021a, Berthelot et al., 2019], unsupervised learning using autoencoders [Beckham et al., 2019, Berthelot* et al., 2019], adversarial learning [Lamb et al., 2019, Lee et al., 2020, Pang* et al., 2020], graph-based learning [Verma et al., 2019b], computer vision [Yun et al., 2019, Jeong et al., 2021, Panfilov et al., 2019, Faramarzi et al., 2020], natural language [Guo et al., 2019, Zhang et al., 2020] and speech [Lam et al., 2020, Tomashenko et al., 2018].

Mixup [Zhang et al., 2018] creates synthetic training samples by linear interpolation in the input vectors and their corresponding labels. The follow-up work of Mixup can be categorized into two main categories: (a) methods that propose a *non-linear* interpolation in the input vector (or hidden vectors): examples of this category include Yun et al. [2019], Faramarzi et al. [2020], Kim et al. [2020], Zhang et al. [2020], Verma et al. [2021b]. (b) methods that extend linear interpolation based Mixup training to various learning paradigms or applications: examples of this class include Mixup based training for supervised learning [Verma et al., 2019a], semi-supervised learning [Verma et al., 2021a, Berthelot et al., 2019], for adversarial training [Lamb et al., 2019], for node classification [Verma et al., 2019b], and for natural language processing [Guo et al., 2019]. The method proposed in this work can be applied to any of the methods in the latter category, and we leave experimental evaluation of *MixupE* applied to these methods for future work.

Theoretically, Mixup has been analyzed by Zhang et al. [2021], in which the authors show that it is approximately equivalent to adding a second-order regularizer to improve robustness and generalization. However, they did not propose a method based on the theory. In contrast, this paper shows that it is equivalent to adding infinite regularizations on the directional derivatives of all orders and uses this theory to propose a new method.

4 EXPERIMENTS

We present a range of experiments to back up the methodological claims, demonstrate versatility across benchmark problems, and show practical applicability on images, tabular data, and speech problems.

4.1 IMAGE DATASETS

For small-scale image datasets, we consider the CIFAR-10, CIFAR-100, SVHN, and Tiny-ImageNet. We run our experiments using a variety of architectures, including PreActResNet18, PreActResNet34, PreActResNet50, PreActResNet101 [He et al., 2016], and Wide-Resnet-28-10 [Zagoruyko and Komodakis, 2016].

Throughout our experiments, we use SGD+Momentum optimizer with batch-size 100, learning rate 0.1, momentum 0.9 and weight-decay 10^{-4}, with step-wise learning rate decay. We train all the networks for all the datasets for 200 epochs, and the learning rate is annealed by a factor of 10 at epochs 100 and 150.

Hyperparameters α and η: For Mixup on the CIFAR-10 and CIFAR-100 datasets, we used the value $\alpha = 1.0$ as recommended by Zhang et al. [2018]. For Mixup on SVHN and Tiny-ImageNet datasets, we experimented with the α values 1.0 and 0.2, respectively, as recommended by Verma et al. [2019a]. We experimented with the $\eta \in \{0.0001, 0.001, 0.01, 0.1\}$ and obtained the best results using $\eta = 0.001$ for most of the experiments and using $\eta = 0.0001$ for the remaining experiments.

Results: We show results for the CIFAR-10 (Table 1a), CIFAR-100 (Table 1b), SVHN (Table 2a), and Tiny-ImageNet (Table 2b) datasets. We see that *MixupE* consistently outperforms baseline methods ERM and Mixup across all the datasets and architectures.

Sensitivity to hyperparameter η: To validate that the method is not overly sensitive to the newly introduced hyperparameter η, we conducted experiments for *MixupE* with the value of $\eta \in \{0.0001, 0.001, 0.01, 0.1\}$ with Preactresnet50

Table 1: Classification errors on (a) CIFAR-10 and (b) CIFAR-100. Standard deviations over five repetitions. Best performing methods in **bold**.

(a) CIFAR-10

PreActResNet50	Test Error (%)
ERM	$4.71_{\pm 0.062}$
Mixup	$4.53_{\pm 0.041}$
MixupE	$\mathbf{3.53}_{\pm 0.047}$
PreActResNet101	
ERM	$4.21_{\pm 0.069}$
Mixup	$4.43_{\pm 0.049}$
MixupE	$\mathbf{3.35}_{\pm 0.049}$
Wide-Resnet-28-10	
ERM	$4.24_{\pm 0.101}$
Mixup	$3.03_{\pm 0.091}$
MixupE	$\mathbf{2.94}_{\pm 0.048}$

(b) CIFAR-100

PreActResNet50	Test Error (%)
ERM	$24.68_{\pm 0.349}$
Mixup	$23.03_{\pm 0.471}$
MixupE	$\mathbf{20.23}_{\pm 0.507}$
PreActResNet101	
ERM	$23.20_{\pm 0.362}$
Mixup	$23.05_{\pm 0.383}$
MixupE	$\mathbf{18.86}_{\pm 0.376}$
Wide-Resnet-28-10	
ERM	$22.20_{\pm 0.108}$
Mixup	$19.38_{\pm 0.113}$
MixupE	$\mathbf{17.12}_{\pm 0.111}$

Table 2: Classification error on SVHN and classification accuracy on Tiny-Imagenet. Standard deviations over five repetitions. Best performing methods in **bold**.

(a) Classification Error on SVHN

PreActResNet50	Test Error (%)
ERM	$2.80_{\pm 0.201}$
Mixup	$2.65_{\pm 0.017}$
MixupE	$\mathbf{2.42}_{\pm 0.021}$
PreActResNet101	
ERM	$2.95_{\pm 0.019}$
Mixup	$2.79_{\pm 0.015}$
MixupE	$\mathbf{2.35}_{\pm 0.019}$
Wide-Resnet-28-10	
ERM	$2.82_{\pm 0.049}$
Mixup	$2.48_{\pm 0.117}$
MixupE	$\mathbf{2.29}_{\pm 0.168}$

(b) Classification Accuracy on Tiny-ImageNet

PreActResNet18	top-1	top-5
ERM	$54.97_{\pm 0.52}$	$72.71_{\pm 0.48}$
Mixup	$54.64_{\pm 0.43}$	$72.53_{\pm 0.51}$
MixupE	$\mathbf{62.21}_{\pm 0.39}$	$\mathbf{82.09}_{\pm 0.41}$
PreActResNet34		
ERM	$57.25_{\pm 0.48}$	$72.58_{\pm 0.53}$
Mixup	$57.79_{\pm 0.39}$	$76.15_{\pm 0.42}$
MixupE	$\mathbf{65.37}_{\pm 0.31}$	$\mathbf{83.77}_{\pm 0.35}$
PreActResNet50		
ERM	$55.91_{\pm 0.61}$	$73.50_{\pm 0.57}$
Mixup	$54.86_{\pm 0.46}$	$73.11_{\pm 0.43}$
MixupE	$\mathbf{67.22}_{\pm 0.38}$	$\mathbf{85.14}_{\pm 0.36}$

architecture and the CIFAR-100 dataset. This experiment was repeated five times with different initializations. We got the mean test error (in %) of $20.23, 20.84, 21.01, 20.87$ for the η values of $0.0001, 0.001, 0.01, 0.1$, respectively, vs the mean test-error 23.03 for Mixup. This suggests that the proposed method *MixupE* is not overly sensitive to the hyperparameter η and works better than Mixup for a large range of η values.

For a large-scale image classification dataset, we consider ImageNet [Deng et al., 2009], using three architectures: ResNet [He et al., 2016], Vision Transformer [ViT, Dosovitskiy et al., 2020], and CoAtNet [Dai et al., 2021]. In particular, we use ResNet-50, ViT-B/16, and CoAtNet-0. We choose these architectures for experiments because they are fast to train, and they respectively represent the three families of image classification models: convolution-based models, attention-based models, and hybrid models.

Except for the Mixup-related hyperparameters α and η, all training hyperparameters for these models follow their original paper. Specifically, all models are trained and evaluated at the resolution of 224x224. Our ResNet-50 is trained with a SGD+Momentum with the momentum coefficient of 0.9, while our ViT-B/16 and CoAtNet-0 are both trained with AdamW [Loshchilov and Hutter, 2018], $\beta_1 = 0.9$ and $\beta_2 = 0.99$. An L_2 weight decay of 10^{-4} is applied to our ResNet-50, while the larger weight decays of 0.05 and 0.3 are applied to our CoAtNet-0 and ViT-B/16, respectively.

All models were trained for 100K steps, with a global batch

size of 4096. Throughout these 100K training steps, the learning rate starts from 0 and warms up linearly to its peak value – which is 1.6 for ResNet-50 and 0.001 for ViT and CoAtNet – and then decreases to 1/1000 times the peak value following the cosine schedule. For models with batch normalization, i.e., ResNet-50 and CoAtNet-0, the batch statistics during training are computed globally. We also apply a Polyak moving average with the rate of 0.999 on all parameters, *including* the batch normalization cumulative statistics in the case of ResNet-50 and CoAtNet-0.

Hyperparameters α and η: we use the Mixup rate $\alpha = 0.2$ for ResNet-50, following the suggestions from Zhang et al. [2018]. Since ViT-B/16 and CoAtNet-0 were invented after Mixup, we first tune the value of α and found that $\alpha = 0.4$ offers a sweet spot for these models. Fixing $\alpha = 0.2$ for ResNet-50 and $\alpha = 0.4$ for ViT-B/16 and CoAtNet-0, we then tune the values for η with ResNet-50. We find that $\eta = 10^{-3}$ is best for ResNet-50 while the smaller value of $\eta = 5 \times 10^{-4}$ is best for ViT-B/16 and CoAtNet-0.

ImageNet results: Table 3 presents our results. We observe that *MixupE* consistently outperforms Mixup across the three architectures in our experiments. Notably, the gains of *MixupE* – in terms of top-1 accuracy – are larger for ViT-B/16 and CoAtNet-0 than for ResNet-50, *i.e.* +0.7 and +0.8 compared to +0.5, even though the top-1 accuracy of the Mixup baselines for ViT-B/16 and CoAtNet-0 are higher. We note that other extensions of Mixup, such as CutMix [Yun et al., 2019], PuzzleMix [Kim et al., 2020], and PatchUp [Faramarzi et al., 2020] use *non-linear* mixing of samples, thus they are not directly comparable with *MixupE*. We leave an experimental comparison of *MixupE* with these methods using a common implementation scheme (architecture and training/validation protocol) as future work.

Table 3: ImageNet accuracy of various models. Each experiment was run for 3 times.

Models	MixUp Type	Top-1	Top-5
ResNet-50	None	$76.2_{\pm 0.5}$	$93.6_{\pm 0.3}$
	MixUp	$77.2_{\pm 0.2}$	$94.0_{\pm 0.1}$
	MixupE	$\mathbf{77.7}_{\pm 0.2}$	$\mathbf{94.4}_{\pm 0.1}$
ViT-B/16	None	$79.1_{\pm 0.2}$	$95.1_{\pm 0.1}$
	MixUp	$79.7_{\pm 0.2}$	$95.4_{\pm 0.1}$
	MixupE	$\mathbf{80.4}_{\pm 0.1}$	$\mathbf{95.8}_{\pm 0.1}$
CoAtNet-0	None	$79.8_{\pm 0.2}$	$95.1_{\pm 0.1}$
	MixUp	$80.8_{\pm 0.3}$	$95.5_{\pm 0.1}$
	MixupE	$\mathbf{81.6}_{\pm 0.2}$	$\mathbf{95.7}_{\pm 0.2}$

A note on implementation and runtime. Despite these improvements, *MixupE* requires twice as many forward passes as normal Mixup. As shown in Algorithm 1, the extra computation stems from the forward pass through the original (non-mixed) samples X, *i.e.* $f_\theta(X)$. For larger models running on ImageNet, this cost can lead to significantly slower experiment time. To alleviate the computational burden, we "batch" this extra pass through the non-mixup data into the pass through the mixup data. Thanks to this trick, our implementation of *MixupE* is only 1.3 times slower than Mixup.

4.2 TABULAR DATASETS

We consider a number of tabular environments drawn from the UCI dataset [Lichman et al., 2013], namely *Arrhythmia, Letter, Balance-scale, Mfeat-factors, Mfeat-fourier, Mfeat-karhunen, Mfeat-morphological, Mfeat-zernike, CMC, Optdigits, Pendigits, Iris, Mnist_784, Abalone* and *Volkert*.

We consider the same setting as Zhang et al. [2018], where the network is a fully-connected multi-layer perceptron (MLP) with two hidden layers, each with 128 dimensions, with ReLU activations for non-linearity. We train this network with the Adam optimizer using the cross-entropy loss with the default learning rate of 0.001 and a batch size of 100, for 25 epochs. We feed in the categorical part of the data as one-hot inputs, and for any samples with missing features in the dataset, we fill it with the mean (for continuous) or mode (for discrete) of those features.

Table 4: Classification Test Error (%) on tabular datasets from UCI repository. Results are averaged over five trials.

Dataset	Method		
	ERM	Mixup	*MixupE*
Arrhythmia	$\mathbf{34.60}_{\pm 3.10}$	$35.49_{\pm 3.88}$	$34.85_{\pm 3.99}$
Letter	$4.56_{\pm 0.27}$	$\mathbf{3.71}_{\pm 0.18}$	$4.04_{\pm 0.20}$
Balance-scale	$3.87_{\pm 1.03}$	$3.70_{\pm 1.00}$	$\mathbf{3.68}_{\pm 0.97}$
Mfeat-factors	$2.74_{\pm 0.81}$	$\mathbf{2.44}_{\pm 0.42}$	$2.56_{\pm 0.64}$
Mfeat-fourier	$17.69_{\pm 1.76}$	$17.80_{\pm 1.56}$	$\mathbf{17.57}_{\pm 1.60}$
Mfeat-karhunen	$3.74_{\pm 0.58}$	$3.06_{\pm 0.29}$	$\mathbf{2.47}_{\pm 0.32}$
Mfeat-morph	$25.00_{\pm 2.10}$	$\mathbf{24.62}_{\pm 1.83}$	$24.66_{\pm 1.30}$
Mfeat-zernike	$17.58_{\pm 1.72}$	$\mathbf{15.19}_{\pm 1.73}$	$15.55_{\pm 0.62}$
CMC	$45.77_{\pm 1.49}$	$46.67_{\pm 1.83}$	$\mathbf{45.42}_{\pm 2.05}$
Optdigits	$1.48_{\pm 0.19}$	$\mathbf{1.15}_{\pm 0.21}$	$1.33_{\pm 0.14}$
Pendigits	$1.03_{\pm 0.25}$	$0.76_{\pm 0.19}$	$\mathbf{0.72}_{\pm 0.16}$
Iris	$9.06_{\pm 7.01}$	$8.14_{\pm 6.48}$	$\mathbf{7.29}_{\pm 6.95}$
Mnist_784	$2.83_{\pm 0.11}$	$2.57_{\pm 0.05}$	$\mathbf{2.56}_{\pm 0.14}$
Abalone	$35.05_{\pm 0.61}$	$35.07_{\pm 0.69}$	$\mathbf{34.91}_{\pm 0.70}$
Volkert	$33.26_{\pm 0.62}$	$32.74_{\pm 0.76}$	$\mathbf{32.54}_{\pm 0.61}$

Hyperparameters α and η: We consider the hyperparameters from the set $\alpha \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0\}$ and $\eta \in \{0.0001, 0.001, 0.01, 0.1, 1.0\}$ and run five seeds for each of the combinations and algorithms. Then, the value of (α, η) is chosen based

on the best validation accuracy, corresponding to which we report the test accuracy for that particular dataset.

Results: Table 4 presents our results for a subset of the tabular datasets. We observe that *MixupE* outperforms the standard Mixup as well as ERM across multiple datasets. Among these 15 datasets, MixupE surpasses the baselines substantially (9 datasets) and achieves comparable performances as the best, such as Arrhythmia, Letter, Mfeat-zernike and Mfeat-morph (4 datasets). On the whole, MixupE has demonstrated considerable improvements over vanilla Mixup by considering the relative improvements of MixupE compared to the standard ERM training scheme.

4.3 SPEECH DATASET

To have a rigorous comparison with Zhang et al. [2018], similar to their work for the speech dataset, we use the Google commands dataset [Warden, 2018]. This dataset consists of 65000 one-second long utterances of 30 short words, such as *yes, no, up, down, left, right, stop, go, on, off*, by thousands of different people. 30 short words correspond to 30 classes. We preprocess the utterances by first extracting the normalized spectrograms from the original waveform at a sampling rate of 16 kHz, followed by zero-padding the spectrograms to equalize their size at 160×101. This preprocessing step is exactly the same as Zhang et al. [2018]. Furthermore, similar to Zhang et al. [2018], we use LeNet [LeCun et al., 1998] and VGG-11 and VGG-13 [Simonyan and Zisserman, 2014] architectures. We train all the models for 20 epochs using Adam optimizer with a learning rate of 0.001 and batch size of 100.

Table 5: Classification Test Error (%) on Google Speech Command Dataset [Warden, 2018]. We run each experiment five times

Architecture	**Method**		
	ERM	Mixup	*MixupE*
LeNet	$10.43_{\pm 0.052}$	$10.12_{\pm 0.041}$	$\mathbf{10.02}_{\pm 0.042}$
VGG-11	$6.04_{\pm 0.059}$	$4.63_{\pm 0.047}$	$\mathbf{3.93}_{\pm 0.050}$
VGG-13	$5.77_{\pm 0.053}$	$4.68_{\pm 0.039}$	$\mathbf{3.84}_{\pm 0.040}$

Hyperparameters α and η: For all the architectures, we first find the best value of hyperparameter α for Mixup from the set $\alpha \in \{0.1, 0.2, 0.5, 1.0, 2.0\}$. We observed that $\alpha = 0.2$ works best consistently for all architectures. For *MixupE*, we used the best α values from Mixup and only fine-tuned the η hyperparameter using $\eta \in \{0.001, 0.01, 0.1, 1.0\}$. In our experiments, $\eta = 0.01$ works best for all the experiments.

Results: In Table 5, we observe that *MixupE* improves the test error of Mixup for different architectures. Moreover, the improvement is more significant for larger architectures such as VGG-11 and VGG-13 than LeNet.

4.4 GRAPH DATASETS

For graph classification, we consider the *MUTAG, NCI1, PTC, PROTEINS, IMDB-BINARY* and *IMDB-MULTI* datasets. We use the experimental settings defined in Xu et al. [2018] as the baseline system, where Mixup and *MixupE* are performed after encoding the graph to a fixed dimensional vector, that is, at the graph-level readout stage. Each system here relies on 5 graph neural network layers that give rise to the readout, which a non-linear MLP then operates on. The models are trained for 350 epochs using the Adam optimizer with a learning rate of 0.01, which is halved every 50 epochs. For the hyperparameters, we consider $\alpha \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0\}$ and $\eta \in \{0.0001, 0.001, 0.01, 0.1, 1.0\}$. Corresponding to each model setting, we perform 10-fold validation, identify which epoch and hyperparameters give the best test accuracy, and report the algorithm's final mean and standard deviation over the ten folds. We refer the readers to Table 6, which shows the benefits of using *MixupE* on the graph datasets.

In conclusion, *MixupE* outperforms than vanilla Mixup on different types of datasets. The proposed regularizer effectively improve the generalization of Mixup.

Table 6: Classification Test Error (%) on graph datasets from the TUDatasets benchmark when following the setup of Xu et al. [2018]. Results are obtained from 10-fold validation.

Dataset	**Method**		
	ERM	Mixup	*MixupE*
MUTAG	$10.15_{\pm 0.06}$	$10.67_{\pm 0.05}$	$\mathbf{10.06}_{\pm 0.06}$
NCI1	$17.79_{\pm 0.02}$	$18.59_{\pm 0.02}$	$\mathbf{17.74}_{\pm 0.01}$
PTC	$38.37_{\pm 0.09}$	$\mathbf{34.87}_{\pm 0.08}$	$35.50_{\pm 0.08}$
PROTEINS	$25.43_{\pm 0.04}$	$24.44_{\pm 0.04}$	$\mathbf{23.72}_{\pm 0.04}$
IMDBBINARY	$25.60_{\pm 0.03}$	$25.30_{\pm 0.03}$	$\mathbf{25.20}_{\pm 0.03}$
IMDBMULTI	$50.33_{\pm 0.03}$	$49.27_{\pm 0.04}$	$\mathbf{48.53}_{\pm 0.03}$

Table 7: Ablation experiments to understand the effect of the additional loss term in Equation 7. Each experiment was run 5 times.

Method	**Test Error**
ERM	24.68
Mixup	23.03
ERM+additional loss	22.42
Mixup+additional loss (*MixupE*)	20.23

Table 8: Test accuracy on novel deformations. All models are trained on normal CIFAR-100.

Test Set Deformation	Mixup ($\alpha = 1$)	Mixup ($\alpha = 2$)	Manifold Mixup ($\alpha = 2$)	Ours ($\alpha = 1$)
Rotation $U(-20, 20)$	55.55	56.48	60.08	**62.23**
Rotation $U(-40, 40)$	37.73	36.78	42.13	**43.08**
Shearing $U(-28.6, 28.6)$	58.16	60.01	62.85	**63.94**
Shearing $U(-57.3, 57.3)$	39.34	39.70	**44.27**	**43.87**
Zoom In (60% rescale)	13.75	13.12	11.49	**15.66**
Zoom In (80% rescale)	52.18	50.47	52.70	**54.22**
Zoom Out (120% rescale)	60.02	61.62	**63.59**	61.39
Zoom Out (140% rescale)	41.81	42.02	**45.29**	36.58

4.5 ABLATION EXPERIMENTS

In *MixupE*, we have proposed to add an additional loss term derived from the first-order derivative (7) to the Mixup Loss 8. A natural question arises: what would be the performance of adding this term to the ERM loss? We conduct an ablation study to investigate this question. Specifically, we compare the following four methods : 1) ERM, 2) Mixup, 3) ERM+additional loss, and 4) Mixup+additional loss (*MixupE*). The test error on the CIFAR-100 dataset using the Preactresnet50 architecture for the abovementioned method is shown in Table 7.

Results in Table 7 show that adding the additional loss term of Eqn (7) improves the test accuracy in Mixup. This is consistent with our argument in Section 2.2 that Mixup can have an unintended effect of maximizing $\|\mathbf{J}_{f_\theta}(\mathbf{x}_i)_{(j)}\|_2$. Furthermore, we observe that Mixup+additional loss (*MixupE*) performs better than ERM+additional loss; this indicates that the implicit regularization of higher order directional derivative through Mixup training is important for better test errors, thus justifying our proposed method.

4.6 GENERALIZATION TO NOVEL DEFORMATIONS

Following Verma et al. [2019a], we also evaluate the robustness of the representations learned by MixupE and compare it to other baselines. For our method, we use the PreActResNet18 only trained with 400 epochs instead of 1200 epochs of Manifold Mixup, which means fewer training epochs were used to obtain our results than other baselines reported in Verma et al. [2019a]. As shown in Table 8, the results indicate that our method consistently outperforms the other methods in most test set deformations. Specifically, for rotation in the range of $U(-20, 20)$ and Zoom In (Rows 1, 5, 6), our method significantly improved over all baselines, which is the highest among all methods. MixupE again outperforms the other methods in Rows 2 and 3 for the rest settings and achieves similar accuracy to the previous SOTA in Row 4. These results suggest that *MixupE* has a better generalization to novel deformation test data.

5 CONCLUSION AND LIMITATIONS

In this work, we have theoretically derived a new method to improve Mixup. Our theory shows that Mixup is a computationally efficient way to regularize directional derivatives of all orders (see Theorem 1). Based on this intuition, we propose a new Mixup variant, termed *MixupE*, a simple and one-line code modification of the original Mixup. Our proposed method is mathematically designed to strengthen the regularization effect of Mixup with a generalization improvement guarantee (see Theorem 2). Empirically, *MixupE* outperforms Mixup on several datasets, such as image, tabular, and speech datasets, trained with various networks. The improvement in test error is more significant for networks with larger capacities. As a limitation, our method requires one additional forward pass in the network during training than Mixup but only suffers an extra 30% time cost than Mixup. While we only approximate the first-order term for the computational efficiency, our results suggest a promising future research direction to enhance Mixup by studying higher-order terms in Theorem 1.

Acknowledgements

This material is based upon work supported by the Google Cloud Research Credits program with the award (6NW8-CF7K-3AG4-1WH1). The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg).

This research/project was partially supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-GC-2019-001-2A).

References

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

Proceedings of the International Conference on Machine Learning, pages 233–242. PMLR, 2017.

Christopher Beckham, Sina Honari, Vikas Verma, Alex M Lamb, Farnoosh Ghadiri, R Devon Hjelm, Yoshua Bengio, and Chris Pal. On adversarial mixup resynthesis. In *Advances in Neural Information Processing Systems*, volume 32, pages 4346–4357. Curran Associates, Inc., 2019.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. MixMatch: A Holistic Approach to Semi-Supervised Learning. *arXiv e-prints*, art. arXiv:1905.02249, May 2019.

David Berthelot*, Colin Raffel*, Aurko Roy, and Ian Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *International Conference on Learning Representations*, 2019.

Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. Vicinal risk minimization. *Advances in Neural Information Processing Systems*, 13, 2000.

Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. *Advances in Neural Information Processing Systems*, 34:3965–3977, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Mojtaba Faramarzi, Mohammad Amini, Akilesh Badrinaaraayanan, Vikas Verma, and Sarath Chandar. Patchup: A regularization technique for convolutional neural networks. *arXiv preprint arXiv:2006.07794*, 2020.

Hongyu Guo, Yongyi Mao, and Richong Zhang. Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*, 2019.

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*, pages 770–778, 2016.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdelrahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

Jisoo Jeong, Vikas Verma, Minsung Hyun, Juho Kannala, and Nojun Kwak. Interpolation-based semi-supervised learning for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11602–11611, 2021.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Zidek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, Aug 2021.

Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *Proceedings of the International Conference on Machine Learning*, pages 5275–5285. PMLR, 2020.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

M. W. Y. Lam, J. Wang, D. Su, and D. Yu. Mixup-breakdown: A consistency training method for improving generalization of speech separation models. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6374–6378, 2020.

Alex Lamb, Vikas Verma, Juho Kannala, and Yoshua Bengio. Interpolated adversarial training: Achieving robust neural networks without sacrificing too much accuracy. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, AISec'19, page 95–103, New York, NY, USA, 2019. Association for Computing Machinery.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Saehyung Lee, H. Lee, and S. Yoon. Adversarial vertex mixup: Toward better adversarially robust generalization.

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 269–278, 2020.

Moshe Lichman et al. UCI machine learning repository, 2013.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

Egor Panfilov, Aleksei Tiulpin, Stefan Klein, Miika T Nieminen, and Simo Saarakkala. Improving robustness of deep learning based knee MRI segmentation: Mixup and adversarial domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019.

Tianyu Pang*, Kun Xu*, and Jun Zhu. Mixup inference: Better exploiting mixup to defend adversarial attacks. In *International Conference on Learning Representations*, 2020.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. Between-class learning for image classification. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5486–5494, 2017.

Natalia Tomashenko, Yuri Khokhlov, and Yannick Estève. Speaker adaptive training and mixup regularization for neural network acoustic models in automatic speech recognition. In *Proceedings of Interspeech*, pages 2414–2418, 09 2018.

Vladimir N Vapnick. *Statistical Learning Theory*. Wiley, New York, 1998.

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 *Proceedings of the 36th International Conference on Machine Learning*, pages 6438–6447, 2019a.

Vikas Verma, Meng Qu, Kenji Kawaguchi, Alex Lamb, Yoshua Bengio, Juho Kannala, and Jian Tang. Graphmix: Improved training of gnns for semi-supervised learning. *arXiv preprint arXiv:1909.11715*, 2019b.

Vikas Verma, Alex Lamb, Kannala Juho, Yoshua Bengio, and David Lopez-Paz. Interpolation consistency training for semi-supervised learning. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, 2021a.

Vikas Verma, Thang Luong, Kenji Kawaguchi, Hieu Pham, and Quoc Le. Towards domain-agnostic contrastive learning. In *Proceedings of the International Conference on Machine Learning*, pages 10530–10541. PMLR, 2021b.

Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 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/CVF international conference on computer vision*, pages 6023–6032, 2019.

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

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

Linjun Zhang, Zhun Deng, and Kenji Kawaguchi. How does mixup help with robustness and generalization? In *International Conference on Learning Representations (ICLR)*, 2021.

Rongzhi Zhang, Yue Yu, and Chao Zhang. SeqMix: Augmenting active sequence labeling via sequence mixup. *arXiv preprint arXiv:2010.02322*, 2020.