A Local Mixup to Prevent Manifold Intrusion

Raphaël Baena, Lucas Drumetz, Vincent Gripon
IMT Atlantique and Lab-STICC
name.surname@imt-atlantique.fr

***Abstract*—Deployed in the context of supervised learning, Mixup is a data-dependent regularization technique that consists in linearly interpolating input samples and associated outputs. It has been shown to improve accuracy when used to train on standard machine learning datasets. However, authors have pointed out that Mixup can produce out-of-distribution virtual samples and even contradictions in the augmented training set, potentially resulting in adversarial effects. In this paper, we introduce *Local Mixup* in which distant input samples are weighted down when computing the loss. In constrained settings we demonstrate that *Local Mixup* can create a trade-off between bias and variance, with the extreme cases reducing to vanilla training and classical Mixup. Using standardized computer vision benchmarks, we also show that *Local Mixup* can improve accuracy.**

***Index Terms*—Mixup, Regularization, Manifold Intrusion**

I. Introduction

Deep Learning has become the golden standard for many tasks in the fields of machine learning and signal processing. Using a large number of tunable parameters, Deep Neural Networks (DNNs) are able to identify subtle dependencies in large training datasets to be later leveraged to perform accurate predictions on previously unseen data. Without constraints or enough samples, many models can fit the training data (high variance) and it is difficult to find the ones that would generalize correctly (low bias).

Regularization techniques have been deployed with the aim of improving generalization [1]. A popular data-dependent regularization technique consists of artificially increasing the size of the training set, which is referred to as *data augmentation* [2], [3]. In the field of computer vision, for example, it is very common to generate new samples using basic class-invariant transformations [4], [5].

In [6], the authors introduce *Mixup*, a data augmentation technique in which artificial training samples $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$, called *virtual* samples, are generated through linear interpolations between two training samples $(\mathbf{x}_i, \mathbf{y}_i)$ and $(\mathbf{x}_j, \mathbf{y}_j)$. The associated output is computed as the corresponding linear interpolation on the respective outputs. *Mixup* improves generalization error of state-of-the-art models on ImageNet, CIFAR, speech, and tabular datasets [6]. This method is also used in the context of few shot learning [7], [8].

By using linear interpolation, *virtual* samples can in some cases contradict each other, or even generate out-of-distribution inputs. This phenomenon has been described in [3] where the authors use the term *manifold intrusion*. As such, it is not clear if *Mixup* is always desirable. More generally, the question arises of whether *Mixup* could be amended to reduce the risk of generating spurious interpolations.

In this paper we introduce *Local Mixup*, where *virtual* samples are weighted in the training loss. The weight of each possible *virtual* sample depends on the distance between the endpoints of the corresponding segment $(\mathbf{x}_i, \mathbf{x}_j)$. In particular, this method can be implemented to forbid interpolations between samples that are too distant from each other in the input domain, reducing the risk of generating spurious virtual samples, as we shall discuss later in this paper.

Here are our main contributions:

- We introduce *Local Mixup*, a mixup method depending on a single parameter whose extremes correspond to classical *Mixup* and Vanilla.
- In dimension one, we prove that *Local Mixup* allows to select a bias/variance trade-off.
- In higher dimensions, we show that *Local Mixup* can help achieve more accurate models than classical *Mixup* using standard vision datasets.
- Our work contributes more broadly to better understanding the impact of *Mixup* during training.

II. Related Work

Introducing notations: In supervised Machine Learning, a training dataset $\mathcal{D}_{train}$ made of pairs of inputs and corresponding outputs is used to learn the model's parameters, and a test one $\mathcal{D}_{test}$ is used to evaluate the performance of the model on previously unseen inputs [9]. We also consider that both input and output data lie in metric spaces $(\mathcal{X}, d_X)$ and $(\mathcal{Y}, d_\mathcal{Y})$. Typically, $\mathcal{X}$ and $\mathcal{Y}$ are assumed to be Euclidean spaces with the usual metrics. We denote by $f : \mathcal{X} \to \mathcal{Y}$ the parametric model to be trained and by $\mathcal{F}$ the hypothesis set, i.e. the set containing all candidate parametrizations of the model $f \in \mathcal{F}$.

To train our model, we use an *error function* $\mathcal{L}$ that measures the discrepancy between the model outputs and expected ones. Training the model amounts to minimizing the training loss while generalization may be quantitavely evaluated by the test loss, resulting in the following simplified equation:

$$L_{vanilla} = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \mathcal{L}(f(\mathbf{x}), \mathbf{y}).$$

Data augmentation and mixup: To improve generalization one can use regularization techniques [1]. Among them, data augmentation is a form of data-dependent regularization [3]. Recently, the use of data-dependent methods relying on some sort of *mixing* has emerged [6], [10]–[19]. The pioneering mixing method is *Mixup* [6], whose mixed samples $(\tilde{x}, \tilde{y})$ are generated by linear interpolations between pairs of samples,

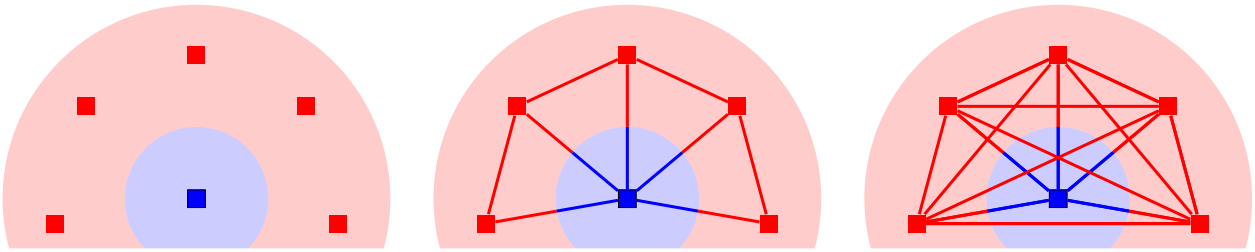

Fig. 1: Illustration of the proposed *Local Mixup* method. On the left, only vanilla samples are used, without data augmentation. Ground truth is depicted in filled regions. On the middle we depict *Local Mixup* where we only interpolate samples which are close enough, leading to no contradiction with ground thuth. On the right we depict *Mixup* in which we interpolate all samples, leading to contradictory virtual samples.

i.e. $\tilde{\mathbf{x}}_{i,j,\lambda} = \lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_j$ and $\tilde{\mathbf{y}}_{i,j,\lambda} = \lambda \mathbf{y}_i + (1-\lambda)\mathbf{y}_j$ for some training samples $(\mathbf{x}_i, \mathbf{y}_i)$ and $(\mathbf{x}_j, \mathbf{y}_j)$ and some $\lambda \in [0,1]$. The *Mixup* training criterion is defined as:

Definition II.1 (Mixup Criterion)**.** *Let* $\lambda \sim Beta[\alpha, \beta]$, i, j *discrete variables uniformly drawn with repetitions in* $\{0, \dots, n-1\}$. f^* *minimizes the Mixup criterion if:*

$$f^* = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n^2} \mathbb{E}_\lambda \left[\underbrace{\sum_{\mathcal{D}^2_{train}} \mathcal{L}\left(\tilde{\mathbf{y}}_{i,j,\lambda}, f(\tilde{\mathbf{x}}_{i,j,\lambda})\right)}_{L_{mixup}} \right].$$

As defined above *Mixup* encourages linear behavior [6]. The positive effect of this linear behavior questioned several authors who aimed at explaining theoretically and empirically *Mixup*. In [20] they shows that Mixup can be interpreted as the combination of a data transformation and a data perturbation; In [21] they highlight that *Mixup* impacts the Lipschitz constant L of the gradient of the network.

In [3] they described and introduced the term *manifold intrusion*. This phenomenon is depicted in Figure 1 on the right, where we see that virtual samples created through *Mixup* between distant red samples lie outside the manifold domain for the red class. The method of [3] called *Adamixup* uses an additional neural network to remove such interpolations.

III. Mixup in dimension 1

The detailed proofs of each theoretical result can be found in a longer version of this paper [22]. Let us consider the simple case where our model f is defined on $\mathbb{R}$. Without loss of generality, let us consider that the training set $\mathcal{D}_{train} = \{x_i, y_i\}$ is ordered by increasing input, i.e, $x_i \leq x_{i+1}$.

For a given $\tilde{x}$, *Mixup*'s loss implies that the output $f^*(\tilde{x})$ of the model is determined by the set $\mathcal{E}(\tilde{x})$ of all convex combinations that can be obtain $\tilde{x}$ from two training inputs x_i and x_j: $\mathcal{E}(\tilde{x}) = \{i, j, \lambda_{i,j} | \tilde{x} = \lambda_{ij} x_i + (1-\lambda_{ij})\ x_j\}$. It is clear that for any $\tilde{x} \in [x_0, x_{n-1}]$, $\mathcal{E}(\tilde{x})$ is non empty and finite. In practice, the distribution of λ can be uniform [6], [10] $\lambda \sim Beta(\alpha = 1, \beta = 1) = \mathcal{U}(0,1)$. In this case, we show the output $f^*(\tilde{x})$ of $x \in [x_0, x_n]$ is the barycenter of the target values corresponding to the points of $\mathcal{E}(\tilde{x})$.

Lemma III.1. $\forall \tilde{x} \in [x_0, x_{n-1}]$,

$$f^*(\tilde{x}) = \frac{1}{card(\mathcal{E}(\tilde{x}))} \sum_{(i,j,\lambda_{i,j}) \in \mathcal{E}(\tilde{x})} \lambda_{i,j} y_i + (1-\lambda_{i,j}) y_j. \quad (1)$$

A consequence of this lemma is the following theorem:

Theorem III.2. *The function* f^* *that minimizes the loss on the training set is piecewise linear on* $[x_0, x_{n-1}]$*, linear on each segment* $[x_i, x_{i+1}]$ *and defined by Equation* (1)*.*

In practice inferring a function f^* that minimizes that the Mixup Criterion is usually not desired in machine learning, and one looks for f with a sufficiently small loss to have a regularizing effect. Indeed f^* is not likely to generalize well. Still, we note that it tends to an average of convex combinations and thus leads to a model with a low variance.

IV. Local Mixup

A. Locality graphs

Consider a (training) dataset $\mathcal{D}$ made of pairs $(\mathbf{x}, \mathbf{y})$. We propose to build a graph from $\mathcal{D}$ as follows. We define $G_{\mathcal{D}} = \langle V, \mathbf{W} \rangle$ where $V = \{\mathbf{x} \mid \exists \mathbf{y}, (\mathbf{x}, \mathbf{y}) \in \mathcal{D}\}$. The symmetric real matrix $\mathbf{W}$ is based on D, where D is the pairwise distance matrix $D[i,j] = d_{\mathcal{X}}(\mathbf{x}_i, \mathbf{x}_j)$.

In this work, we consider various ways to obtain $\mathbf{W}$, but the rationale is always the same: to obtain a similarity matrix where large weights correspond to closest pairs of samples. Namely, we consider K-nearest neighbors graphs, where we set to 1 weights of target vertices corresponding to the K closest samples for a given source vertex and 0 otherwise; thresholded graphs where $\mathbf{W}[i,j] = \phi(D[i,j])$ and $\phi(d) = \mathbf{1}_{d \leq \varepsilon}$; smooth decreasing exponential graphs where $\mathbf{W}[i,j] = \exp(-\alpha \mathbf{D}[i,j])$. The loss is then weighted using $\mathbf{W}$:

$$L_{\text{local mixup}} = \sum_{\mathcal{D}^2_{train}} \mathbf{W}[i,j] \mathcal{L}\left(\tilde{\mathbf{y}}_{i,j,\lambda}, f(\tilde{\mathbf{x}}_{i,j,\lambda})\right). \quad (2)$$

For computational cost considerations, we compute a graph for each batch (random subset) of samples during stochastic gradient descent. As such, the weights associating two samples can vary depending on the chosen graph and random batch.

In the extreme case where some weights are 0, the corresponding virtual samples are discarded during gradient descent, resulting in only considering local interpolations of samples, hence the name *Local Mixup*.

B. Low dimension

In this section, we are interested in proving that *Local Mixup* allows to tune a trade-off between bias and variance on trained models. For this purpose, we simplify the problem to dimension 1 and only consider K-nearest neighbor graphs.

In this case, note that varying K can create a range of settings where $K = 0$ boils down to vanilla training and $K \geq n$ where n is the number of training samples boils down to classical *Mixup* where all combinations are allowed.

1) Local Mixup and the bias/variance trade-off: Let us first recall the definitions of the bias and variance in the context of a machine learning problem.

Definition IV.1 (Bias and Variance)**.** *Let us consider a training set $\mathcal{D}_{train}$ and a function f from $\mathcal{X}$ to $\mathcal{Y}$. We define Bias and Variance as follow:*

- Bias*:* $Bias(f)^2 = \mathbb{E}_{train}[(f(x) - y)^2]$.
- Variance*:* $Var(f) = \mathbb{E}_{train}[(f - \mathbb{E}_{train}[f])^2]$.

We consider two settings. In the first one, the input domain $\mathbb{Z}/n\mathbb{Z}$ is periodic and thus the number of samples is finite. In the second one, the input domain $\mathbb{Z}$ is infinite and outputs are independent and identically distributed (i.i.d).

Periodic setting

Let us consider that the training set $\mathcal{D}_{train}$ is made of pairs (x, y), where $\{x \mid \exists y, (x, y) \in \mathcal{D}_{train}\} = \mathbb{Z}/n\mathbb{Z}$. We also consider $d_{\mathcal{X}}(x, x') = |x - x'| \in \{0, \cdots, n-1\}$.

In this case, we can write explicit formulations of f^*_K, the function that minimizes the *Local Mixup* criterion for K-nearest neighbors graphs. Following similar arguments to those used to obtain Equation (1): for a given x_i we know that the optimal value for $f^*_K(x_i)$ would be an average of the the $\tilde{y}$ that correspond to the possible interpolations. we obtain:

$$\forall x_i \in \mathbb{Z}/n\mathbb{Z}, f^*_K(x_i) = \frac{1}{K(K+3)/2}(2Ky_i + S_K(x_i)), \quad (3)$$

where $S_K(x_i)$ is defined recursively as follows:

$$S_{K+1} = \begin{cases} 0 & \text{if } K = 0 \\ S_K(x_i) + A_{K+1}(x_i) & \forall K \geq 1 \end{cases}. \quad (4)$$

and:

$$A_K(x_i) = \frac{1}{K}\sum_{k=1}^{K-1}(K-k)\cdot y_{i-k} + k \cdot y_{i+K-k}.$$

On Figure 2 we depicted for a given x_i the different interpolations and $\tilde{y}$ that contribute to $f_K(x_i)$. In blue the interpolation between x_i and its direct neighbors, in red the interpolation between points other than x_i that happen to intersect x_i. As we increase K, the influence of S_K (red points) increases.

We obtain the following theorem:

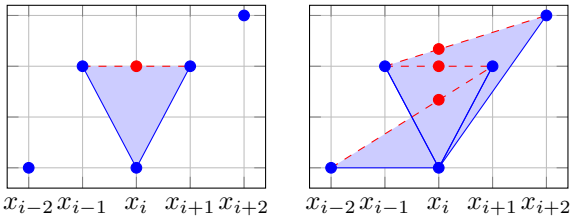

Fig. 2: We depict here the terms of $f^*_K(x_i)$ given by Eq (3) for different values of K. In blue the interpolations corresponding to $2Ky_i$ and in red the terms of the sum S_K. On the right, $K = 2$ and on the left $K = 3$.

Theorem IV.1 (Convergence of f^*_K in the periodic setting)**.** *As K grows, it holds that:*

$$\forall x_i \in \mathbb{Z}/n\mathbb{Z},$$
$$f^*_K(x_i) \to \mathbb{E}_{D_{train}}[y], \quad (5)$$
$$Bias^2(f^*_K) \to \mathbb{E}_{train}[(y_i - \mathbb{E}_{train}[y])^2], \quad (6)$$
$$Var(f^*_K) = \mathbb{E}_{train}[(f^*_K(x_i) - \mathbb{E}_{train}[f^*_K(x_i)])^2] \to 0, \quad (7)$$
$$Var(f^*_K) \text{ is eventually nonincreasing.}$$

This theorem states two main results: 1) in the case of *Mixup* the function that minimizes the loss f^* has zero variance and converges to $\mathbb{E}_{train}[y]$. 2). Eventually the variance of the function that minimizes the *Local Mixup* criterion is decreasing, showing that the proposed *Local Mixup* can indeed tune the trade-off between the bias and variance.

i.i.d random output setting

Let us now consider that the training set is made of inputs $\{x \mid \exists y, (x, y) \in \mathcal{D}_{train}\} = \mathbb{Z}$ and y_i are i.i.d. according to a random variable R of variance σ^2.

Theorem IV.2. *For a signal with i.i.d outputs, the variance is eventually bounded by:*

$$\frac{4^2\sigma^2}{K^2} \leq Var(f_K(x_i)) \leq \frac{8\sigma^2}{K}. \quad (8)$$

2) Invariance of linear models: Interestingly, we can show that both *Mixup* and *Local Mixup* lead to the same optimal linear models, as stated in the following theorem:

Theorem IV.3. *For a linear model: $f(x) = ax + b,\ a, b \in \mathbb{R}$, the function f^* that minimizes the loss of* Mixup *and* Local Mixup *is the same.*

C. High Dimension and Lipschitz constraint

The proofs given in low dimension have some limitations. Basically, the averaging effect happens since any point x within the interval $[x_1, x_n]$ can be written as at least one convex combination of pairs from the training set. Contradictions may occur as illustrated above when several combinations corresponds to x. In higher dimension such explicit contradictions are not necessarily expected. Still, the more segments we allow to interpolate, the larger becomes the virtual training set, eventually leading to potentially smaller margins between classes and thus to harder variance/trade-off compromises. We will illustrate this point in the experiments.

METHOD	CIFAR-10 / Resnet18	CIFAR-100 / Resnet18	Fashion-MNIST / Densenet	SVHN / LeNet
Baseline	4.98 ± 0.03	30.6 ± 0.27	6.20 ± 0.2	10.01 ± 0.15
Mixup	4.13 ± 0.03	29.23 ± 0.4	6.36 ± 0.16	8.31 ± 0.14
Local Mixup	4.03 ± 0.03	29.08 ± 0.34	5.97 ± 0.2	8.20 ± 0.13

TABLE I: Error rates (%) on CIFAR-10, Fashion-MNIST and SVHN. Values are averaged on 100 runs for CIFAR-10 and 10 runs for Fashion-MNIST and SVHN. Mean errors with their confidence intervals are given.

V. Experiments

Our method depends on an hyperparameter α or ϵ. The value of these parameters has been chosen for each experiments according to the ablations studies that we provide in [22].

A. Low dimension

As stated in the introduction and [3], *Mixup* leads to interpolations that may be misleading for the model. To illustrate this effect, we consider a 2d toy dataset of two coiling spirals where such interpolations occur frequently. The two coiling spirals is a binary classification dataset: each spiral corresponds to a different class. We expect to retrieve better performance for *Local Mixup* compared to *Mixup*: local interpolations are likely to stay in the same spiral and therefore avoid manifold intrusion. For this experiment we use a thresholded graph with parameter ε.

We use a fully connected neural network made of two hidden layers with 100 neurons and ReLU function as non linear function. For small values of ε many weights of the graph are zero and thus the corresponding interpolations are disregarded into the loss. This means that for a given batch only a small proportion of samples are actually considered. To avoid side effects, we vary the batch size so that in average the same number of samples are used to update the parameters.

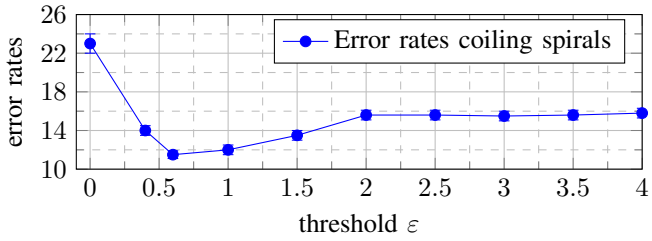

Fig. 3: Error rate (averaged over 1000 runs) as a function of ε for the two coiling spirals dataset. 1000 samples per class generated with a Gaussian noise of standard deviation $\sigma = 1.5$. Extremes correspond to Vanilla ($\varepsilon = 0$) and *Mixup* ($\varepsilon > 4$).

In Figure 3, we depict the evolution of the average error rate as a function of the parameter ε. One can note the significant benefit of *Mixup* and *Local Mixup* over Vanilla. As expected, *Local Mixup* presents a minimum error rate which is significantly smaller than *Mixup*'s error rate. We can note that the minimum is reached with a value of ε smaller than the first quantile. This means that for this dataset *Mixup* interpolations given above this threshold are either useless or misleading for the network's training.

B. Lipschitz lower bound

To illustrate the impact of ε on the optimal Lipschitz constant, we use the dataset CIFAR-10 [23]. In Figure 4, we depict the evolution of $Q(D)$ a lower bound of the Lipschitz constant that we obtain in [22].

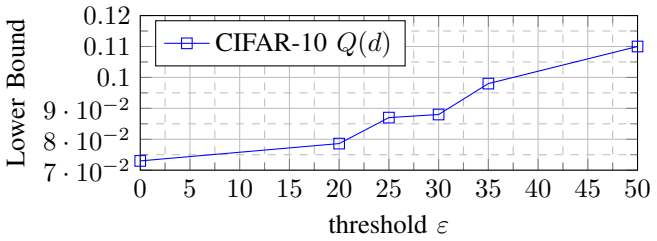

Fig. 4: Evolution of $Q(D)$ on the dataset CIFAR10. $\varepsilon = 0$ corresponds to Vanilla. $\varepsilon = 50$ corresponds to classical *Mixup*.

For classical *Mixup* we obtained $Q(D) = 0.11$ and for Vanilla $Q(D) = 0.073$. Note that these two extremes are reached with *Local Mixup* when $\varepsilon = 0$ and $\varepsilon \geq 50$.

We observe that ε can be used to smoothly tune the lower bound $Q(D)$. In practice, a lower $Q(D)$ is preferable, but this only accounts for the optimal Lipschitz constant. Larger values of ε lead to larger training sets and thus potentially better generalization.

C. High dimension: Experiments on classification dataset

The previous toy dataset is particularly suitable to generate contradictory virtual samples. We delve into more complex and real world datasets in the following subsection to illustrate the negative impact of some interpolations.

We tested our proposition on different classification datasets and architectures and report the results in table 1. We consider the datasets CIFAR-10, CIFAR-100 [23] Fashion-MNIST [24] and SVHN [25]. Fashion-MNIST is composed of clothes images of size 28x28 pixels (grayscale) . There are 60,000 images in the training set corresponding to 10 classes. SVHN is a real-world image dataset made of small cropped digits of size 32x32 pixels and 3 colors. There are 73257 digits in the training set corresponding to 10 classes. For these tests, we use a smooth decreasing exponential graph tuned by α.

We observed that *Local Mixup* showed a smaller error rate on these datasets. For Fashion MNIST, we note that *Mixup* impacts negatively the error rate, suggesting that on this dataset *Mixup* creates spurious interpolations as discussed in [3], which might be caused by a low intrinsic dimensionality.

For these experiments, we also tried to use a K-nearest neighbor graph or a thresholded graph but without being able to achieve smaller error rates compared to *Mixup* or even Vanilla. This may indicate that some segments generated by *Mixup* are important to act as a regularizer during training even if some of them may generate manifold intrusions. By tuning α, we weight the importance of this regularization.

D. Comparaison with Adamixup

We compare our proposed approach with *Adamixup*, another method presented as capable of preventing manifold intrusion. We use the GitHub Repository of the author and we changed only the Mixup part to implement our method. On CIFAR-10 we used 1400 epochs as well for Local Mixup and *Adamixup* since the authors used this number for *Adamixup*. We observe a slight advantage for *Adamixup* on MNIST: $0.49\% \pm 0.03$ for *Adamixup*, $0.54\% \pm 0.02$ for *Local Mixup* (error rates averaged over 10 runs). Still, on CIFAR-10 our method outperforms *Adamixup*: $4.11\% \pm 0.12$ for *Adamixup* and $3.89\% \pm 0.12$ for *Local Mixup* (averaged over 5 runs).

Note that *Local Mixup* does not completely discard interpolations but weighs them, contrary to *Adamixup* that prevents the interpolations which are considered as causing manifold intrusions. Thus, this may indicates that even interpolated samples causing manifold intrusion could be beneficial as long as their are not predominant in the loss.

We also note that our method seems to converge faster on MNIST. One benefit of our proposed approach is the simplicity and the small number of parameters (CIFAR-10): 836522 for *Local Mixup* and 11171146 for *Adamixup*.

VI. Conclusion

In this paper, we introduced a method called *Local Mixup*, in which pairs of samples are interpolated and weighted in the loss depending on the distance between them in the input domain. This method comes with a hyper-parameter that allows to provide a continuous range of solutions between Vanilla and classical *Mixup*. Using a simple framework, we showed that *Local Mixup* can control the bias/variance trade-off of trained models. In more general settings, we showed that *Local Mixup* can tune a lower bound on the Lipschitz constant of the trained model. We used real world datasets to prove the ability of *Local Mixup* to achieve better generalization, as measured using the test error rate, than Vanilla and classical *Mixup*.

Overall, our methodology introduces a simple way to incorporate locality notions into *Mixup*. We believe that such a notion of locality is beneficial and could be leveraged to a greater level in future work, or could be incorporated to the various *Mixup* extensions that have been proposed in the community. In future work, we would like to investigate further the choice of the graph, the choice of the hyper-parameter that comes with it, and trainable versions of *Local Mixup*. We would like to rely on quantitative information given on the topology such as the histogram of the distance or persistence diagrams [26] to tune these hyper-parameters. There are also many possibilities to improve over using the Euclidean metric, in particular the pullback metrics given by the Euclidean distance between the samples once in the feature space corresponding to the penultimate layer. Extending the theoretical results to more general contexts would definitely allow to gain intuition on the effect of locality on *Mixup*.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[2] P. Simard, Y. Lecun, J. Denker, and B. Victorri, "Transformation invariance in pattern recognition – tangent distance and tangent propagation," *International Journal of Imaging Systems and Technology*, vol. 11, 01 2001.

[3] H. Guo, Y. Mao, and R. Zhang, "Mixup as locally linear out-of-manifold regularization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3714–3722.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[6] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[7] P. Mangla, N. Kumari, A. Sinha, M. Singh, B. Krishnamurthy, and V. N. Balasubramanian, "Charting the right manifold: Manifold mixup for few-shot learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 2218–2227.

[8] G. S. Dhillon, P. Chaudhari, A. Ravichandran, and S. Soatto, "A baseline for few-shot image classification," *arXiv preprint arXiv:1909.02729*, 2019.

[9] C. M. Bishop, "Pattern recognition," *Machine learning*, vol. 128, no. 9, 2006.

[10] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, and Y. Bengio, "Manifold mixup: Better representations by interpolating hidden states," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6438–6447.

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

[12] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[13] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, "Augmix: A simple data processing method to improve robustness and uncertainty," *arXiv preprint arXiv:1912.02781*, 2019.

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

[15] H.-P. Chou, S.-C. Chang, J.-Y. Pan, W. Wei, and D.-C. Juan, "Remix: Rebalanced mixup," in *European Conference on Computer Vision*. Springer, 2020, pp. 95–110.

[16] Z. Liu, S. Li, D. Wu, Z. Chen, L. Wu, J. Guo, and S. Z. Li, "Automix: Unveiling the power of mixup," *arXiv preprint arXiv:2103.13027*, 2021.

[17] J. Chen, S. Sinha, and A. Kyrillidis, "Stackmix: A complementary mix algorithm," *arXiv preprint arXiv:2011.12618*, 2020.

[18] W. Yin, H. Wang, J. Qu, and C. Xiong, "Batchmixup: Improving training by interpolating hidden states of the entire mini-batch."

[19] A. Rame, R. Sun, and M. Cord, "Mixmo: Mixing multiple inputs for multiple outputs via deep subnetworks," *arXiv preprint arXiv:2103.06132*, 2021.

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

[21] P. K. Gyawali, S. Ghimire, and L. Wang, "Enhancing mixup-based semi-supervised learning with explicit lipschitz regularization," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 1046–1051.

[22] R. Baena, L. Drumetz, and V. Gripon, "Preventing manifold intrusion with locality: Local mixup," https://arxiv.org/abs/2201.04368, 2022.

[23] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 05 2012.

[24] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[25] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[26] L. Wasserman, "Topological data analysis," *Annual Review of Statistics and Its Application*, vol. 5, pp. 501–532, 2018.