On Mainifold Mixup For Deep Learning

Candidate Number :

Abstract

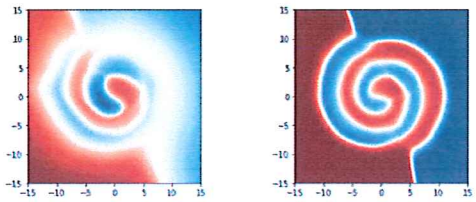
Manifold Mixup is a regularization technique introduced by Verma et al. in [11] that involves training a neural network on interpolated hidden states. In this paper we quantify the impact of applying Manifold Mixup deeper into in a network and the type of problems that the technique works best on. We also propose a method to dynamically change the strength of the regularization called ConfMix.

1 Introduction

Mixup was introduced by Zhang et al. in [13] as a data augmentation technique. The idea was subsequently generalised to Manifold Mixup in [11]. The technique has been empirically shown to shrink the generalisation gap [14] and improve robustness to adversarial examples [6]. In this paper we primarily consider classification tasks. In short Mixup consists of mixing two data points by a mixing coefficient λ and then training on the resulting data point,

$$(\tilde{x}_{i,j}, \tilde{y}_{i,j}) = \lambda(x_i, y_i) + (1 - \lambda)(x_j, y_j).$$

The intuition behind Mixup is that it encourages a smoother transition between predicting different classes and helps the network to discover the location of the decision boundary. In our preliminary experiments (figure 1) we found that input Mixup lead to a more pronounced boundary.



(a) No Mixup (b) Input Mixup

Figure 1: The decision boundary of a 4 layer ReLU network on the spiral dataset.

Our Contributions In this paper we introduce a family of toy datasets used to analyse how the performance of Manifold Mixup varies with depth and with problem difficulty. We quantify the difficulty of a classification problem by introducing the Mixup misclassification lower bound. We also propose a novel method to prescribe the hyper-parameter α (introduced in section 2) called ConfMix, which uses a partially trained classifier's confidence to dynamically increase or decrease the regularizing effect of Mixup. *Note that all code used in this paper is 100% original.*

Related Work The advent of Mixup has sparked a wave of related data augmentation techniques for images namely Puzzle Mix [5], CutMix [12] and AugMix [3]. These methods have demonstrated better empirical performance when working with natural images.

One property of Mixup is that the technique encourages safe interpolation in hidden states. The classical form of Mixup is not best suited to generative image models, recent works on this issue include [7], [8], [1] and [2]. Most of the work on Mixup is empirical, the most comprehensive review of the theory is [14].

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

2 Preliminaries

2.1 Notation

Consider a K-class classification problem and let $\mathcal{X} = \mathbb{R}^2$ be the input space with $\mathcal{Y} = \{1, \dots, K\}$ being the target space. Let $\mathcal{D} = (X, Y)$ be the joint distribution of two random variables taking values in $\mathcal{X} \times \mathcal{Y}$. Furthermore, assume that $P(Y = k) = \pi_k$ and that $X \mid Y = k \sim X_k$ where X_k has pdf $g_k(x)$.

Let $h : \mathcal{X} \to \mathcal{Y}$ denote a prediction rule and $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ denote a loss function. We use $\mathcal{R}(h) = \mathbb{E}\left[L\left(h(X), Y\right)\right]$ to denote the risk of a decision rule and $\tilde{\mathcal{R}}(h)$ to denote the empirical risk over a dataset $\tilde{\mathcal{D}} = (x_i, y_i)_{i=1}^n \subseteq \mathcal{X} \times \mathcal{Y}$. Denote the Bayes classifier $h^* = \arg\min_h \mathcal{R}(h)$.

2.2 Manifold Mixup

As introduced by [11] Manifold Mixup [1] is a technique used when training deep neural networks. Consider a neural network $h(x) = F_k\left(G_k(x)\right)$, where $G_k(x)$ denotes the activity of the k-th hidden layer and F_k is the remaining part of the network. We use $\tilde{\cdot}$ to denote quantities that have been mixed. Training h with Manifold Mixup happens as follows,

1. Process two random mini-batches of data $(x_i, y_i), (x_j, y_j)$ until reaching the k-th layer.
2. Perform input Mixup, as introduced by [13] on the intermediate mini-batches

$$\left(\tilde{G}_k(x_i, x_j), \tilde{y}_{i,j}\right) := \lambda\left(G_k(x_i), y_i\right) + (1 - \lambda)\ \left(G_k(x_j), y_j\right).$$

3. Continue to process the intermediate mini-batch, and perform a parameter update using the gradient of the loss

$$L\left(F_k\left(\tilde{G}_k(x_i, x_j)\right), \tilde{y}_{i,j}\right).$$

Here λ is called the mixing coefficient and is drawn from a Beta(α, α) distribution where $\alpha \in (0, \infty)$ is a hyper-parameter. The value of λ can either be computed on a per-batch or on a per-sample basis. Note that in the case $\alpha \to 0$, then standard empirical risk minimisation (ERM) is recovered. The risk that Manifold Mixup aims to minimise is,

$$\mathcal{R}_{\text{mix}}(h) = \mathop{\mathbb{E}}_{(x,y)} \mathop{\mathbb{E}}_{(x',y')} \mathop{\mathbb{E}}_{\lambda \sim \text{Beta}(\alpha,\alpha)} L\left[F_k\left(\tilde{G}_k\left(x, x'\right)\right), \tilde{y}\right]. \tag{1}$$

We state Theorem 1 from [11] and paraphrase Lemma 3.1 from [14] as Result 1.

Theorem 1 *Let $\mathcal{H}$ be some vector space of dimension $dim(\mathcal{H})$, and let $d \in \mathbb{N}$ denote the number of classes appearing in some dataset $\tilde{\mathcal{D}}$. Let F^*, G^* be the minimisers of (1). If $dim(\mathcal{H}) \geq d - 1$ then F^* is linear.*

Result 1 *We can write $\mathcal{R}_{mix}(h) = \mathcal{R}_{ERM}(h) + \sum_{i=1}^{3} \mathcal{R}_i(h) + \mathbb{E}_\lambda\left((1-\lambda)^2 \phi(1-\lambda)\right)$ where $\lim_{a \to 0} \phi(a) = 0$ and $\mathcal{R}_i$ are higher order loss terms depending on ∇h and $\nabla^2 h$, that vanish as $\lambda \to 1$.*

Result 1 shows that Mixup is equivalent to ERM with regularization terms, and that the strength of this regularization is controlled by λ (and hence α), as the $\mathcal{R}_i$ terms regularize the derivatives of h this explains why training under Mixup often leads to a smoother transition between classes. Theorem 1 suggests that we should not seek to minimise (1) in practice, else we could use simpler linear models. These results inform how we view Mixup and implement it programmatically. We do not optimize over all possible convex combinations of data points but instead on convex combinations of randomly sampled mini-batches, as discussed above.

[1] We use the terms Mixup and Manifold Mixup largely interchangeably throughout.

3 Analysis of Mixup with depth

In this section we seek to explore two phenomena

1. In light of Theorem 1, how does a model behave when Mixup is applied at different depths?
2. Can we understand which types of classification problems Mixup performs best on?

3.1 Toy datasets

To this end we first introduce a novel family of toy datasets, inspired by [10] section 8. We chose to analyse datasets of this nature as we can analytically compute the Bayes classifier and ([13], [11]) already have comprehensive studies of (Manifold) Mixup on benchmark datasets. In-line with the notation above, we consider classification problems where the class distributions are given by a mixture distribution,

$$X \mid Y = k \ \sim X_k \text{ with pdf } g_k(x) = \sum_{i=1}^{N} w_i \, \phi\left(\mu_i, \Sigma_i\right),$$

where $\sum w_i = 1$ and ϕ denotes the pdf of a multivariate normal distribution. In our experiments we chose $w_i = \frac{1}{N}$ and $\Sigma_i = \sigma^2 \, I$ for all i, where σ is a shared standard deviation. We sampled μ_i, σ from uniform distributions over a fixed range. Changing N and σ allows us to control the difficulty and complexity of the classification problem in question.

The key insight is that having the Bayes classifier h^* allows us to look at the excess risk $\mathcal{R}(h) - \mathcal{R}(h^*)$ of a prediction rule h. The excess risk is more meaningful to analyse as we consider problems of varying difficulty in our subsequent experiments.

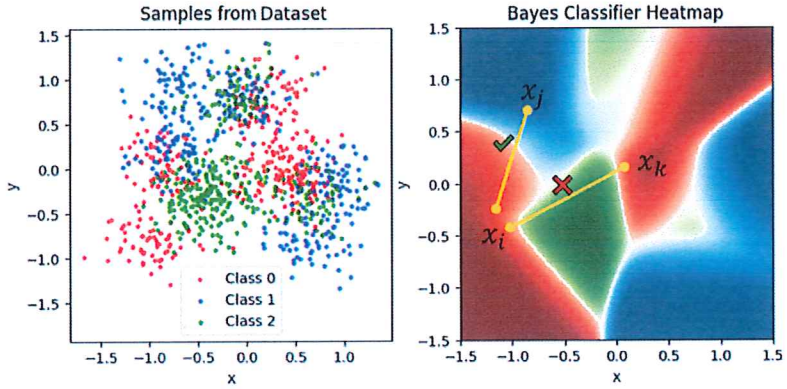


Figure 2: An example of one particular toy dataset with 3 classes and $N = 5$ (per class).

3.2 Mixup misclassification bound

In light of figure 2, we can see that performing input Mixup between data points x_i, x_j, would be beneficial as a decision boundary lies on the line segment between x_i and x_j. On the other hand, input Mixup between data points x_i and x_k would be detrimental as along this line segment $\tilde{y}_{i,k} = (1, 0, 0) \neq (0, 0, 1)$ and so would encourage the classifier to misclassify points from class 2.

To quantify this effect we introduce a lower bound for the Mixup risk, which we shall call the Mixup misclassification lower bound (MMLB). Writing $\mathcal{D}_{\text{mix}} = \mathcal{D} \times \mathcal{D} \times \text{Beta}(\alpha, \alpha)$, to mean the independent joint distribution. Starting from (1) with $k = 0$ we have,

$$\mathcal{R}_{\text{mix}}(h^*) \geq \sum_{k=1}^{K} \pi_k^2 \mathop{\mathbb{E}}_{x, x', \lambda} \left[L\left(h^*(\tilde{x}), k\right)\right] := \mathcal{R}_{\text{low}}\left(\mathcal{D}_{\text{mix}}\right). \tag{2}$$

In our computations of the MMLB we took $\alpha = 1$ and $\pi_k = \frac{1}{K}$ so that $\mathcal{R}_{\text{low}}\left(\mathcal{D}_{\text{mix}}\right)$ can be computed explicitly as a five dimensional integral that *only* depends on the class distributions X_k.

Remark. Analytic computation of the expectation in (2) is expensive, in practice we take a large number of samples to estimate this quantity.

3.3 Experimental details

We analysed 400 realisations of our toy datasets $\tilde{\mathcal{D}} = (x_i, y_i)_{i=1}^{1000}$, 100 for each $K \in \{2, 3, 4, 5\}$, each with 1000 training samples. We trained a 4-layer ReLU$(2, 512, 512, 512, K)$ network for 20 epochs with a batch size of 50, with a new random (but fixed for each type of training) initialization per dataset.

We either performed no Mixup (baseline) or Manifold Mixup on layer k (with $k = 0$ corresponding to input Mixup). We found that a width of 512 was expressive enough to capture the more complex decision boundaries that emerge for large K.

We performed Mixup on a per sample basis (One value of λ per pair of randomly chosen inputs). We implemented Mixup in the form of a custom data generator, used to generate a shuffle order and λ and then a Mixup layer with different behaviour during training and inference. This approach was chosen as it allows for Mixup to be performed on non-sequential models and multiple layers at the same time, whereas existing implementations do not [9]. To be in line with [11] we used $\alpha = 0.5$.

3.4 Experimental results

In our experiments we found that performance (under the 0-1 loss) improved as Mixup was applied deeper into the network. This aligns with Theorem 1, as the last layer of the network is either a softmax or a sigmoid layer which has linear decision boundaries.

We found that $\left(\sum \pi_k^2\right) \mathcal{R}_{\text{low}}\left(\mathcal{D}_{\text{mix}}\right)$ (which we call the normalised MMLB) is strongly correlated with $R(h^*)$, which is to be expected but suggests that the core difficulty of a classification problem can be estimated by understanding how convex the class distributions are. Figure 3 suggests that applying Mixup on the deeper layers has the most impact, and offers a greater absolute performance improvement as the normalised MMLB increases. The decision boundaries under each technique are qualitatively different, though it is hard to draw any conclusions from these (omitted) figures.

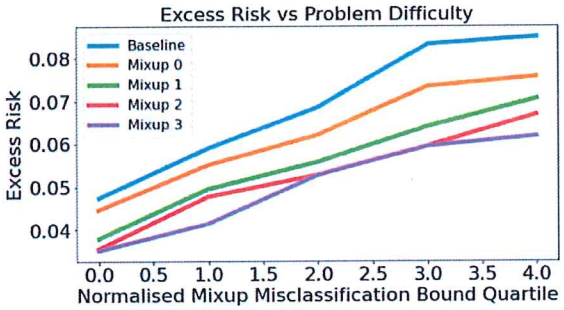


Figure 3: Excess risk against normalised MMLB quartile

Model	Avg % Excess Risk
Baseline	18.1
Mixup 0	16.5
Mixup 1	14.7
Mixup 2	13.8
Mixup 3	**13.2**

Table 1: Mixup Performance

4 ConfMix

In all existing Mixup methods the hyper-parameter α needs to be manually chosen. We propose a novel Mixup algorithm called ConfMix (confidence based mixing) to address this issue. Recall that,

- if $\alpha = 0$ then Mixup collapses to standard ERM (Empirical Risk Minimisation),
- if $\alpha = 1$ then Beta$(\alpha, \alpha) \sim \mathcal{U}([0, 1])$ and so Mixup corresponds to a uniform mixing.

4.1 Proposed method

We consider training a smooth classifier h on a K-class classification problem, by smooth we mean that h outputs a confidence rather than a class label (i.e max instead of $\arg\max$ for the final layer). We are loosely inspired by smoothMix [4] and momentum based gradient decent methods.

(Input) ConfMix Algorithm

Input : Data points $(x_i, y_i), (x_j, y_j)$ and a smooth classifier h
Output: The mixed input and target $(\tilde{x}_{i,j}, \tilde{y}_{i,j})$

$\alpha \leftarrow \frac{K}{K-1}\left[h\left(x_i\right) - \frac{1}{K}\right] \in [0, 1]$
Sample λ from Beta(α, α)
$\tilde{x}_{i,j} = \lambda x_i + (1-\lambda)x_j$
$\tilde{y}_{i,j} = \lambda y_i + (1-\lambda)y_j$
return $(\tilde{x}_{i,j}, \tilde{x}_{i,j})$

Algorithm 1: Pseudo code for (input) ConfMix

Mixup Technique	Accuracy Change
Input Mixup ($\alpha = 0.5$)	+1.9%
Input Confmix	+3.4%
$\frac{1}{2}$Input ConfMix	+2.4%
Input Mixup : α from $0 \rightarrow 1$	+3.1%
Input Mixup : α from $1 \rightarrow 0$	+3.1%
Manifold Mixup ($\alpha = 0.5$)	+3.7%
Manifold Confmix	+3.7%
$\frac{1}{2}$Manifold ConfMix	+2.9%
Manifold Mixup : α from $0 \rightarrow 1$	**+3.9%**
Manifold Mixup : α from $1 \rightarrow 0$	+2.6%

Table 2: Improvement over the baseline network

The idea behind ConfMix is as follows,

- We first train under ERM for an initial M epochs, so that h produces sensible smooth labels for most points.
- Points with high confidence can be used as 'anchors' to propagate information about the location of a decision boundary by Mixup (want $\alpha \sim 1$).
- Points with low confidence would benefit more from ERM (want $\alpha \sim 0$).

4.2 Experimental details

We look at the CIFAR-10 benchmark dataset, with $K = 10$. In our experiments we trained a network with two convolutional layers Conv$(32, 64)$ with 3×3 kernel sizes, and then with two fully connected layers ReLU(64), Softmax(10). The networks were trained for 50 epochs with batch sizes of 250. The baseline (in table 2) is the same network trained under ERM. We 'turned on' ConfMix after $M = 5$ epochs. We denote $\frac{1}{2}$ConfMix to mean exactly ConfMix but with $\lambda \sim$ Beta$(\frac{\alpha}{2}, \frac{\alpha}{2})$. We performed Manifold Mixup immediately before the first fully connected layer. Two natural benchmarks for ConfMix are schemes in which α varies from 0 to 1 (or visa versa) over the course of training. We implemented an exponential decay scheme, with a half-life of 25 epochs.

4.3 Experimental results

Firstly we found that for 4/5 of the techniques, applying the Mixup later in the network was more beneficial, which supports our discussion from section 3. During the duration of training with ConfMix we observed that during early training $\alpha \sim 0.2$ but in later training $\alpha \sim 1$. The training accuracy (which is under the Mixup labels) converged to almost 100%. This is evidence that ConfMix allows the model to reinforce its own biases.

We can see that all Mixup techniques offer an improvement over the baseline network. ConfMix and a scheme in which α varies from $0 \rightarrow 1$ during training offered the greatest performance improvement. This makes sense as these techniques are similar in practice. Overall we can see that starting with ERM and increasing the strength of Mixup regularization during training is the most beneficial to test accuracy.

5 Conclusion

Manifold Mixup is a data augmentation/regularization technique that generates new samples from convex combinations of hidden states in deep networks. In this paper we introduced a family of toy datasets and a quantity called the Manifold Mixup lower bound to quantify the difficulty of a classification problem. We gathered empirical evidence that Manifold Mixup performed deeper into the network is most beneficial and offers a greater improvement on harder problems. We also introduced ConfMix, a method for generating the value of the hyper-parameter α on a per sample basis. We found that adaptive α schedules where α transitions from $0 \rightarrow 1$ (increasing the strength of the Mixup regularization) during training were most impactful. Extensions to this work include extending section 3 to images (see [3], [12] and [5]) and investigating further schemes like ConfMix.

References

[1] Christopher Beckham, Sina Honari, Vikas Verma, Alex Lamb, Farnoosh Ghadiri, R Devon Hjelm, Yoshua Bengio, and Christopher Pal. On adversarial mixup resynthesis, 2019.

[2] David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer, Jan 2018.

[3] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty, 2020.

[4] Jongheon Jeong, Sejun Park, Minkyu Kim, Heung-Chang Lee, Do-Guk Kim, and Jinwoo Shin. Smoothmix: Training confidence-calibrated smoothed classifiers for certified robustness. *CoRR*, abs/2111.09277, Jan 2021.

[5] Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup, 2020.

[6] Alex Lamb, Vikas Verma, Kenji Kawaguchi, Savya Khosla, Juho Kannala, and Yoshua Bengio. Interpolated adversarial training: Achieving robust neural networks without sacrificing too much accuracy, 2021.

[7] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian J. Goodfellow. Adversarial autoencoders. *CoRR*, abs/1511.05644, 2015.

[8] Alon Oring, Zohar Yakhini, and Yacov Hel-Or. Autoencoder image interpolation by shaping the latent space. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8281–8290. PMLR, 18–24 Jul 2021.

[9] Rahul Rahulmadanahalli. Manifold mixup keras implementation. `https://github.com/rahulmadanahalli/manifold_mixup` 2019.

[10] B. D. Ripley. Neural networks and related methods for classification. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(3):409–437, Jan 1994.

[11] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, Aaron Courville, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. 2019.

[12] 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 (ICCV)*, October 2019.

[13] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017.

[14] Linjun Zhang, Zhun Deng, Kenji Kawaguchi, Amirata Ghorbani, and James Zou. How does mixup help with robustness and generalization?, 2021.