When does mixup promote local linearity in learned representations?

Arslan Chaudhry
DeepMind
arslanch@google.com

Aditya Krishna Menon
Google Research
adityakmenon@google.com

Andreas Veit
Google Research
aveit@google.com

Sadeep Jayasumana
Google Research
sadeep@google.com

Srikumar Ramalingam
Google Research
rsrikumar@google.com

Sanjiv Kumar
Google Research
sanjivk@google.com

Abstract

mixup is a regularization technique that artificially produces new samples using convex combinations of original training points. This simple technique has shown strong empirical performance, and has been heavily used as part of semi-supervised learning techniques such as mixmatch [1] and interpolation consistent training (ICT) [17]. In this paper, we look at mixup through a representation learning lens in a semi-supervised learning setup. In particular, we study the role of mixup in promoting linearity in the learned network representations. Towards this, we study two questions: (1) how does the mixup loss that enforces linearity in the last network layer propagate the linearity to the earlier layers?; and (2) how does the enforcement of stronger mixup loss on more than two data points affect the convergence of training? We empirically investigate these properties of mixup on vision datasets such as CIFAR-10, CIFAR-100 and SVHN. Our results show that supervised mixup training does not make all the network layers linear; in fact the intermediate layers become more non-linear during mixup training compared to a network that is trained without mixup. However, when mixup is used as an unsupervised loss, we observe that all the network layers become more linear resulting in faster training convergence.

1 Introduction

While models learned via empirical risk minimization (ERM) [15] tend to perform well on test data that are similar to training data, predictions can change significantly when the samples are chosen outside the training distribution. For improved generalization, typically data augmentation techniques are used to generate new training examples near the neighborhood of the original training samples through simple transformations [10]. Such techniques play a critical role in training deep neural networks that have shown great success [6][12][4][3]. While popular data augmentation techniques such as filtering and cropping on images tend to produce samples near the vicinity of training samples, they are domain-specific and require expert knowledge in generating augmentations.

On the contrary, mixup [19] – an augmentation technique that generates new training data by linearly interpolating the original training points – is applicable in various domains without requiring expert domain knowledge. mixup has shown strong performance in both supervised [19] and semi-supervised [17][1] learning setups by allowing the model to learn better network representations. In the context of supervised learning, the representations learned through mixup regularization are

*The work was done when the author was with Google Research. The author is now at DeepMind.
shown to improve generalization of the network and also its robustness to corrupt labels [19]. The general idea of linearly combining the two vectors in mixup can then be extended to the intermediate layers of the network resulting in even better representations [16]. Additionally, mixup training has been shown to improve network calibration for both in- and out-of-distribution data [14].

In the context of semi-supervised learning, mixup regularization, also known as interpolation consistency training (ICT), is shown to improve the performance of a learner significantly. Specifically, Verma et al. [17] showed that mixup training encourages consistency regularization which is an effective unsupervised learning signal explored in many works [9, 7, 13, 8]. Other related methods include mixmatch [11] that rely on mixing labeled and unlabeled samples, and methods relying on pseudo-labels [14] and data augmentation [19].

While mixup regularization has brought performance benefits across the board, it is less clear how network representations differ between a network trained with and without mixup. Some theoretical works have studied the improved generalization of mixup through the lens of regularization effects it brings to the network training. Carratino et al. [2] argued that mixup regularizes the Jacobian of the network resulting in a function with a low Lipschitz constant. Similarly, Zhang et al. [20] showed that, in a two-layer ReLU network, mixup training reduces the complexity of the hypothesis class leading to better generalization. While these works point to some regularization effects of mixup, the training dynamics of a network trained under mixup regularization are not fully clear.

Our work is an empirical study that aims to provide some clues as to how network representations evolve during mixup training. Specifically, 1) we show that mixup tends to make the first and the last layer of a network more linear, but does so at the expense of making the intermediate layers more non-linear, compared to a network that is trained without mixup. 2) We show that when mixup is used as an unsupervised loss, ICT, all layers tend to become more linear. Finally, 3) we show that enforcing a stronger linearity in mixup, by means of using more than two mixing points, leads to more linear representations that manifest in faster convergence to a specified test accuracy with less labeled examples in a semi-supervised setup.

2 Setup

Let $\mathcal{X} \subseteq \mathbb{R}^D$ and $\mathcal{Y} \subseteq \mathbb{R}^C$ be input and output spaces, respectively. We adopt a standard semi-supervised learning setup where we assume access to a small labeled dataset $\mathcal{D}_s = \{(x_i, y_i)\}_{i=1}^{n_s}$, and a large unlabeled dataset $\mathcal{D}_{us} = \{u_i\}_{i=1}^{n_{us}}$, with $n_s \ll n_{us}$ are the number of examples in the labeled and unlabeled datasets, respectively. The training objective is to learn a model $f : \mathcal{X} \rightarrow \mathcal{Y}$, parameterized by $\theta \in \mathbb{R}^P$ (a neural network in our case), that performs well on a held out test set $\mathcal{D}_{te}$ by minimizing the following loss:

$$\ell = \ell_s + w_1 \cdot \ell_{us},$$

where $\ell_s : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$ is a supervised loss computed on $\mathcal{D}_s$ (typically a cross-entropy loss for classification tasks), $\ell_{us}$ is an unsupervised loss defined on $\mathcal{D}_{us}$, and $w(\ell)$ is the weight of unsupervised loss at the $t$-th iteration of stochastic gradient descent (SGD). We define the mixup operation as a convex operation parameterized by $\lambda \in \mathbb{R}^{\geq 0}_K$, on a set $\mathcal{P} = \{p_1, \cdots, p_K\}$ as:

$$\text{Mix}_\lambda(\mathcal{P}) = \sum_{i=1}^{K} \lambda_i p_i,$$  \hspace{1cm} (1)

such that $\sum_{i=1}^{K} \lambda_i = 1$, and $\lambda_i \geq 0$ for all $i \in \{1, \cdots, K\}$. A sample from a $K$-th order Dirichlet distribution with parameters $\alpha_1, \cdots, \alpha_K > 0$, defines a valid $\lambda$ for the mixup operation. When the mixup operation is used on the supervised loss $\ell_s$, the examples $x \in \mathcal{X}$ and targets $y \in \mathcal{Y}$ are modified as:

$$x_{\text{us}} = \text{Mix}_\lambda(x_1, \cdots, x_K),$$

$${y}_{\text{us}} = \text{Mix}_\lambda(y_1, \cdots, y_K),$$

at each training step of the SGD. For the case of $K = 2$, the supervised loss becomes the standard mixup training as proposed by Zhang et al. [19]. Similarly, we can also use the mixup operation to define an unsupervised loss on the unlabeled examples. Concretely, given $K$ unlabeled examples $K = \{u_1, \cdots, u_K\}$, the unsupervised loss $\ell_{us}$ is defined as,

$$\ell_{\text{us}} = D \left( f(\text{Mix}_\lambda(K)) - \text{Mix}_\lambda(f(u_1), \cdots, f(u_K)) \right),$$  \hspace{1cm} (2)
where $D$ is some distance metric; $L_2$ distance is used in this work. Intuitively, the unsupervised loss enforces consistent predictions on the interpolated unlabeled examples which Verma et al. [17] found to be a good learning signal for consistency regularization in semi-supervised setups.

3 Experiments

Setup We assume standard semi-supervised learning datasets (CIFAR-10, CIFAR-100, and SVHN) and experiment with different labeled dataset sizes. For CIFAR-10 and SVHN, we consider labeled datasets of sizes $\{250, 1000, 2000, 40000\}$. For CIFAR-100, we only consider a single labeled dataset of size 10K. We use a ‘Wide ResNet-28’ architecture similar to [1]. We denote the output of the initial convolution layer as ‘Layer 0’, and the outputs of the next three Wide ResNet blocks are referred to as ‘Layer 1-3’, followed by the output of average pooling layer that is denoted by ‘Layer 4’. Finally the output is referred to as ‘Layer 5’. Under this definition of a ‘layer’, we monitor the representation at the $l$-th training step by:

$$\text{NonLin}_l(l) = \sum_{(x_1, x_2) \sim D_{ts}} D\left(\hat{f}_l^{\lambda}(\text{Mix}(x_1, x_2)) - \text{Mix}(\hat{f}_l^{\lambda}(x_1), \hat{f}_l^{\lambda}(x_2))\right),$$

where $D_{ts}$ is a held out test set, and $\hat{f}_l^{\lambda}(\cdot)$ is the normalized output of layer $l$ at the $t$-th training step.

We normalize the output of a layer to have a unit norm, i.e. $\hat{f}_l^{\lambda} = \frac{f_l^{\lambda}}{\|f_l^{\lambda}\|}$, to allow for a fair comparison of the non-linearity across different layers. Lower values imply a more linear layer.

Results Figure 1 show the evolution of non-linearity throughout network training for different baselines on CIFAR-10. These figures use a labeled dataset of size 250 (see Appendix for other datasets and settings). The total training iterations are remapped between 0 and 1000 and referred to as Train time in the plots.

First, we compare the (non-)linearity of ERM and mixup (both are supervised setups) in different layers. We see that in the first and last layer the representations from mixup are more linear compared to ERM representations. Intuitively, this could be explained by the mixup operation linearly combining both the inputs and outputs. Hence, training using mixup keeps the first and last functions in the composition $f = f_0 \circ f_1 \cdots \circ f_t$ more linear. Surprisingly, however, we do not see that mixup representations remain more linear in all the layers compared to those of ERM as suggested by the theory. For example, in Figure 1, we can see that Layers 2, 3, 4 based on mixup training are more non-linear compared to those trained with ERM. This suggests that a) the findings of theory on simplified setups do not directly translate to practical deep networks, and b) more importantly, without proper regularization in all layers, the network maintains an overall non-linearity from input to output. Further, this suggests that the linearity enforced by a regularization in some layers is counteracted by increased non-linearity in the other layers.
Second, we see that ICT tend to make all the layers more linear compared to ERM and mixup, except for the first layer where mixup is slightly more linear. This suggests that the ICT loss, as defined in equation 2, applied on the network logits, and on a larger unsupervised dataset, is a stronger regularization compared to mixup defined on one-hot encoded vectors. Further, we see from Figure 1 that using more mixup points (ICT4 vs ICT2), leads to stronger linearity across all layers. In the next section, we will see that this stronger linearity also leads to faster convergence.

3.2 Effect of stronger mixup on convergence

In the previous section, we saw that stronger enforcement of linearity, by using more mixup points, led to the network layers becoming more linear. We will now empirically show the effect of this linearity on network convergence. Specifically, we measure test accuracy given a specified number of labeled training examples. For this, we will investigate the performance of different baselines in a semi-supervised setup for varying amounts of labeled data.

Results Figure 2 shows the accuracy on a fixed test set against various labeled dataset sizes for CIFAR-10 (the results for CIFAR-100 are given in the appendix). From the Figure, we can see that the ICT-based model generally outperforms the ERM and mixup baselines across all dataset sizes. This is likely due to the additional unsupervised training performed on the additional unsupervised dataset. Among the ICT variants we further observe that the test accuracy tends to be higher as we increase the number of points used in the mixup operation. This is especially true in the regime of very small datasets and the performance of all ICT variants converges to similar performance as more datapoints become available. For example, to reach 70% test accuracy on CIFAR-10, mixup requires roughly 1800 labeled examples, whereas ICT4 only requires 500 labeled examples (even less for ICT5). The stronger regularization achieved by using 4 mixup points leads to the network becoming more linear, which results in more efficient learning from fewer labeled samples.

4 Conclusion

In this work, we explored when does (or doesn’t) mixup enforce local linearity in the learned representations. We studied this question in a semi-supervised learning setup for image classification tasks. Our experiments demonstrate that standard supervised mixup training doesn’t make the representations in all the layers locally linear. In fact, some of the intermediate layers become more
Figure 2: CIFAR-10: Test accuracy with varying numbers of labeled training examples. ERM and mixup refer to the cases where only supervised loss is used. ICT[K] refer to a setup where the ICT loss with K mixup points are used as unsupervised loss. The supervised loss with the ICT does not use mixup loss.

non-linear during supervised mixup training compared to standard empirical risk minimization (ERM) training. However, when mixup operation is used as an unsupervised loss, on a larger unlabeled dataset, network representations in all layers become more linear. As a consequence of these smoother representations, the network converges to a given test set performance faster (in terms of labeled samples) than a network that is locally less linear. These findings, which we verified empirically, sheds new light on the training dynamics of a network trained under mixup regularization.
References


Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Please see section 1.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Please see section 3.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Please see section 3.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]
Appendix

In this appendix we provide more detailed results for various sizes of labeled dataset. The results here further strengthen the claims made in the main paper.

A More Results

Figure 3: **CIFAR-10**: Linearity propagation in different layers with 2000 labeled examples and consistency weight of 100. ERM and mixup refer to the cases where only supervised loss is used. \(\text{ICT}_K\) refer to a setup where the ICT loss with \(K\) mixup points are used as unsupervised loss. The supervised loss with the ICT does not use mixup.

Figure 4: **CIFAR-10**: Linearity propagation in different layers with 4000 labeled examples and consistency weight of 100. ERM and mixup refer to the cases where only supervised loss is used. \(\text{ICT}_K\) refer to a setup where the ICT loss with \(K\) mixup points are used as unsupervised loss. The supervised loss with the ICT does not use mixup.
Figure 5: SVHN: Linearity propagation in different layers with 250 labeled examples and consistency weight of 100. ERM and mixup refer to the cases where only supervised loss is used. ICT\[X\] refer to a setup where the ICT loss with X mixup points are used as unsupervised loss. The supervised loss with the ICT does not use mixup.

Figure 6: SVHN: Linearity propagation in different layers with 2000 labeled examples and consistency weight of 100. ERM and mixup refer to the cases where only supervised loss is used. ICT\[K\] refer to a setup where the ICT loss with K mixup points are used as unsupervised loss. The supervised loss with the ICT does not use mixup.
Figure 7: **SVHN**: Linearity propagation in different layers with 4000 labeled examples and consistency weight of 100. ERM and mixup refer to the cases where only supervised loss is used. ICT$_K$ \( K \) refer to a setup where the ICT loss with \( K \) mixup points are used as unsupervised loss. The supervised loss with the ICT does not use mixup.

Figure 8: **CIFAR-100**: Linearity propagation in different layers with 100 labeled examples and consistency weight of 100. ERM and mixup refer to the cases where only supervised loss is used. ICT$_K$ refer to a setup where the ICT loss with \( K \) mixup points are used as unsupervised loss. The supervised loss with the ICT does not use mixup.
Figure 9: **SVHN**: Test accuracy when stronger linearity is enforced. ERM and mixup refer to the cases where only supervised loss is used. ICT[X] refer to a setup where the ICT loss with X mixup points are used as unsupervised loss. The supervised loss with the ICT does not use mixup loss.

Figure 10: **CIFAR-100**: Test accuracy