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.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

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 [1] that rely on mixing labeled and unlabeled samples, and methods relying on pseudo-labels [11] and data augmentation [18].

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 $\mathcal{X} \in \mathbb{R}^D$ and $\mathcal{Y} \in \mathbb{R}^C$ 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 $\mathcal{D}_{\text{us}} = \{u_i\}_{i=1}^{n_{\text{us}}}$, with $n_s \ll n_{\text{us}}$ are the number of examples in the labeled and unlabeled datasets, respectively. The training objective is to learn a model $f_\theta : \mathcal{X} \mapsto \mathcal{Y}$, parameterized by $\theta \in \mathbb{R}^P$ (a neural network in our case), that performs well on a held out test set $\mathcal{D}_{\text{te}}$ by minimizing the following loss:

$$\ell = \ell_{\text{s}} + w_t \cdot \ell_{\text{us}},$$

where $\ell_{\text{s}} : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}_{\geq 0}$ is a supervised loss computed on $\mathcal{D}_s$ (typically a cross-entropy loss for classification tasks), ℓ_{us} is an unsupervised loss defined on $\mathcal{D}_{\text{us}}$, and $w(t)$ 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 $\boldsymbol{\lambda} \in \mathbb{R}^K_{\geq 0}$, on a set $\mathcal{P} = \{p_1, \cdots, p_K\}$ as:

$$\texttt{Mix}_{\boldsymbol{\lambda}}(\mathcal{P}) = \sum_{i=1}^{K} \lambda_i p_i, \tag{1}$$

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 $\boldsymbol{\lambda}$ for the mixup operation. When the mixup operation is used on the supervised loss ℓ_{s}, the examples $x \in \mathcal{X}$ and targets $y \in \mathcal{Y}$ are modified as:

$$x_{\text{m}} = \texttt{Mix}_{\boldsymbol{\lambda}}(x_1, \cdots, x_K),$$
$$y_{\text{m}} = \texttt{Mix}_{\boldsymbol{\lambda}}(y_1, \cdots, y_K),$$

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 $\mathcal{K} = \{u_1, \cdots, u_K\}$, the unsupervised loss ℓ_{us} is defined as,

$$\ell_{\text{us}} = D\Big(f\big(\texttt{Mix}_{\boldsymbol{\lambda}}(\mathcal{K})\big) - \texttt{Mix}_{\boldsymbol{\lambda}}\big(f(u_1), \cdots, f(u_K)\big)\Big), \tag{2}$$

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 logits layer is referred to as 'Layer 5'. Under this definition of a 'layer', we monitor the representation after each layer. The batch size is set to 128 and the Adam optimizer [5] is used with learning rate = 0.002, β_1 = 0.9 and β_2 = 0.999. The weight (w_t) of the unsupervised loss follows a schedule such that its value linearly increases from 0 to a maximum of 100 in 40% of training iterations that are kept fixed for all datasets and for all dataset sizes. The linearity of network 'layers' is periodically reported throughout the network training on a held out test set. The α for the Dirichlet distribution is set to 0.5 for CIFAR-10 and CIFAR-100, and 0.1 for SVHN. Each method is run with 5 different random seeds and the performance is averaged across these runs, and error bars are reported for convergence plots.

Baselines We compare 3 different baselines. **ERM** refers to standard empirical risk minimization *without* `mixup`. `mixup` refers to training using two `mixup` points. In both these setups, only the supervised labeled set $\mathcal{D}_\mathrm{s}$ is used and $\mathcal{D}_\mathrm{us}$ is not used. **ICT[K]** refers to unsupervised `mixup`, as defined in equation 2, with $K \in \{2, 3, 4\}$ referring to the number of examples mixed together. For this setup, the ICT loss is computed on the $\mathcal{D}_\mathrm{us}$ and standard cross-entropy loss computed on $\mathcal{D}_s$.

3.1 Non-Linearity in Network Representations

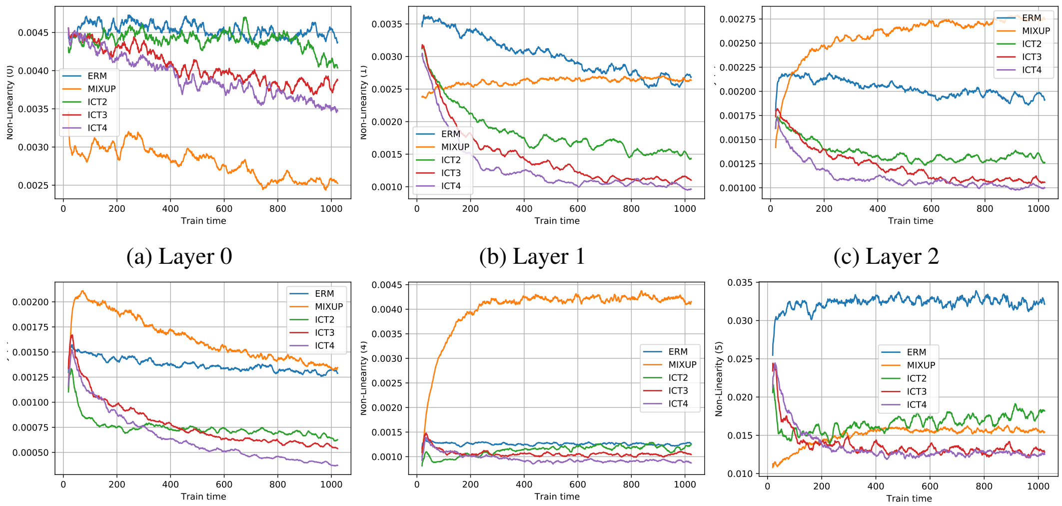
We first investigate the effect of `mixup` on the network representations at different layers during training. Specifically, we take a look at the effect of `mixup` training on the (non-) linearity of network representations. We specifically chose to study the linearity of representations because the `mixup` theory [2, 20] suggests that `mixup` trained networks tend to be more linear than their non-`mixup` counterparts. We define the non-linearity in a layer l at the t-th training step by:

$$\texttt{NonLin}_t(l) = \sum_{(x_1,x_2)\sim\mathcal{D}_{te}} D\Bigg(\hat{f}_t^l(\texttt{Mix}_{\boldsymbol{\lambda}}(x_1, x_2)) - \texttt{Mix}_{\boldsymbol{\lambda}}(\hat{f}_t^l(x_1), \hat{f}_t^l(x_2))\Bigg), \tag{3}$$

where $\mathcal{D}_\mathrm{te}$ is a held out test set, and $\hat{f}_t^l(\cdot)$ 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.* $\hat{f}_t^l = \frac{f_t^l}{\|f_t^l\|}$, 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 $f = f_0 \circ f_1 \cdots \circ f_5$ 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.



(a) Layer 0 (b) Layer 1 (c) Layer 2

(d) Layer 3 (e) Layer 4 (f) Layer 5

Figure 1: **CIFAR-10**: 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[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`.

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

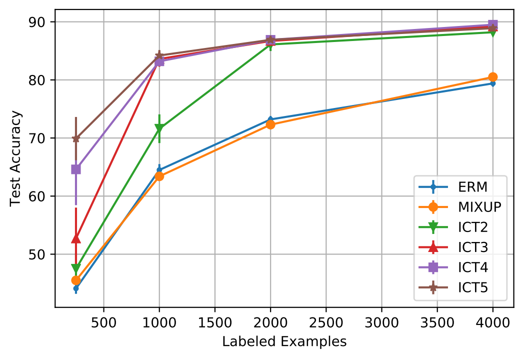


Figure 2: **CIFAR-10**: Test accuracy with varying numbers of labeled training examples. ERM and `mixup` refer to the cases where only supervised loss is used. ICT[K] refer to a setup where the ICT loss with K `mixup` points are used as unsupervised loss. The supervised loss with the ICT does not use `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

[1] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning, 2019.

[2] Luigi Carratino, Moustapha Cissé, Rodolphe Jenatton, and Jean-Philippe Vert. On mixup regularization. *arXiv preprint arXiv:2006.06049*, 2020.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, 2018.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[5] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[7] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

[8] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

[9] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *Advances in neural information processing systems*, 29:1163–1171, 2016.

[10] Patrice Y. Simard, Yann LeCun, John S. Denker, and Bernard Victorri. Transformation invariance in pattern recognition-tangent distance and tangent propagation. In *Neural Networks: Tricks of the Trade*, 1996.

[11] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence, 2020.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[13] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017.

[14] Sunil Thulasidasan, Gopinath Chennupati, Jeff Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks, 2020.

[15] V.N. Vapnik. Statistical learning theory. 1998.

[16] 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.

[17] Vikas Verma, Alex Lamb, Juho Kannala, Yoshua Bengio, and David Lopez-Paz. Interpolation consistency training for semi-supervised learning. *arXiv preprint arXiv:1903.03825*, 2019.

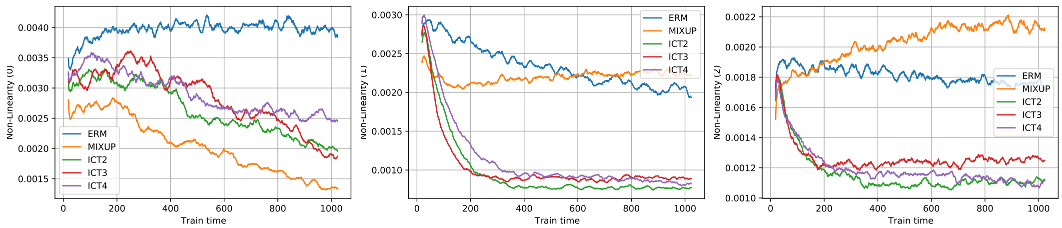
[18] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised data augmentation for consistency training. In *NeurIPS*, 2020.

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

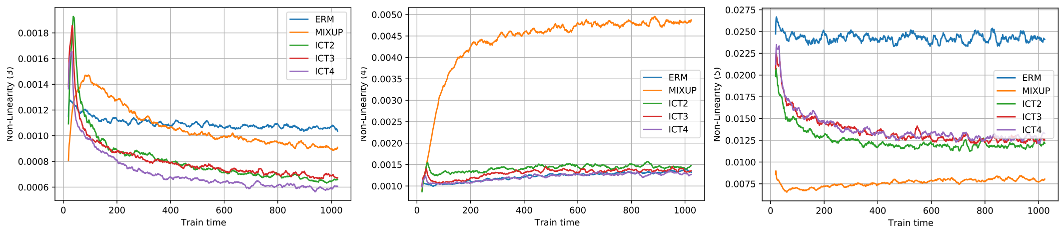
[20] Linjun Zhang, Zhun Deng, Kenji Kawaguchi, Amirata Ghorbani, and James Zou. How does mixup help with robustness and generalization? *arXiv preprint arXiv:2010.04819*, 2020.

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]

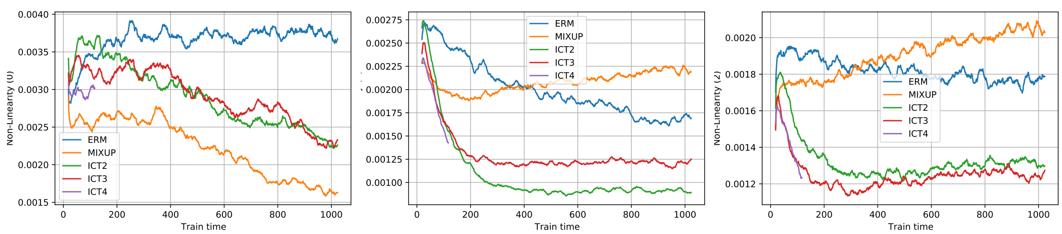


(a) Layer 0 (b) Layer 1 (c) Layer 2

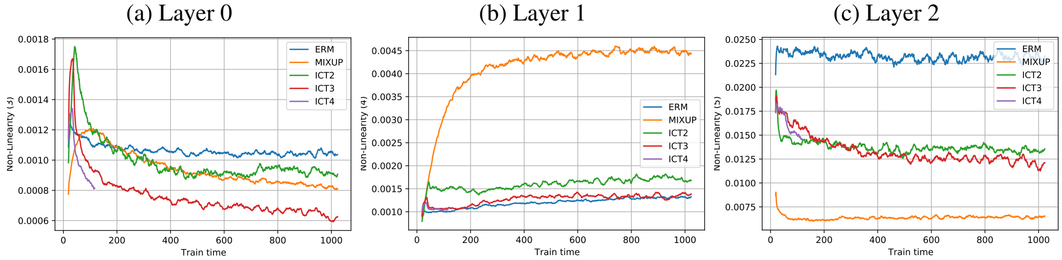


(d) Layer 3 (e) Layer 4 (f) Layer 5

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. $\texttt{ICT}_K$ 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.



(a) Layer 0 (b) Layer 1 (c) Layer 2



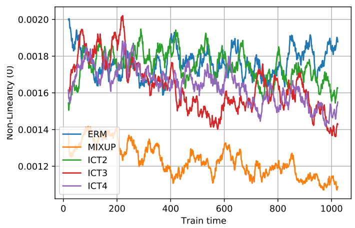
(d) Layer 3 (e) Layer 4 (f) Layer 5

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. $\texttt{ICT}_K$ 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.

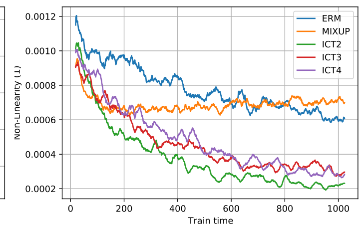
Appendix

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.

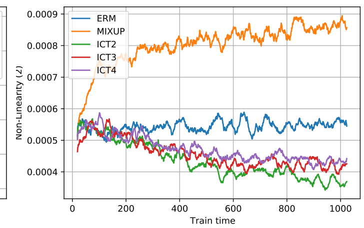
A More Results



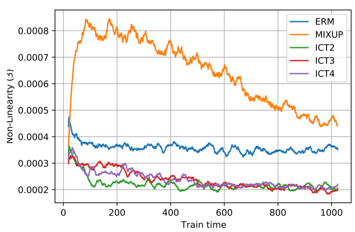
(a) Layer 0



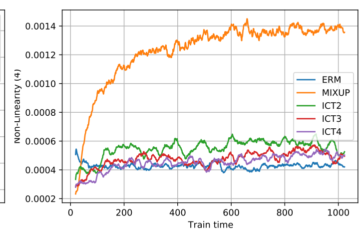
(b) Layer 1



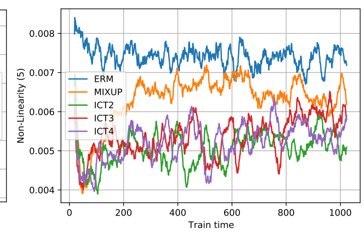
(c) Layer 2



(d) Layer 3

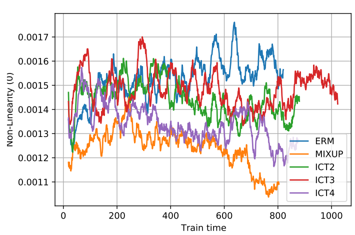


(e) Layer 4

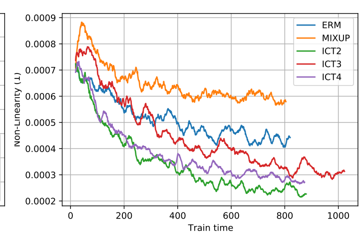


(f) Layer 5

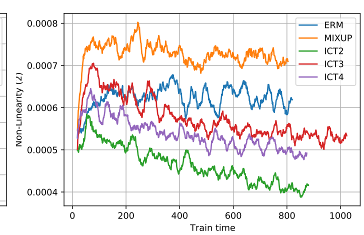
Figure 5: **SVHN**: Linearity propagation in different layers with 250 labeled examples and consistency weight of 100. ERM and mixup refer to the cases where only supervised loss is used. ICT[X] refer to a setup where the ICT loss with X mixup points are used as unsupervised loss. The supervised loss with the ICT does not use mixup.



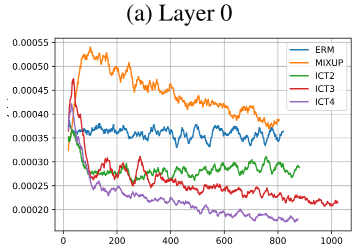
(a) Layer 0



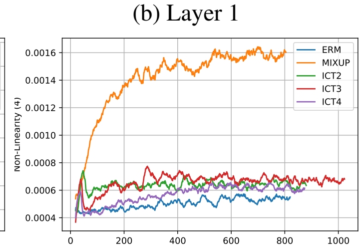
(b) Layer 1



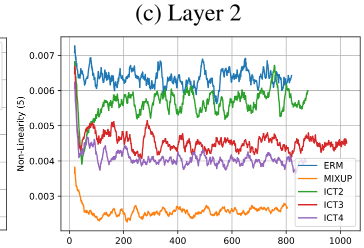
(c) Layer 2



(d) Layer 3

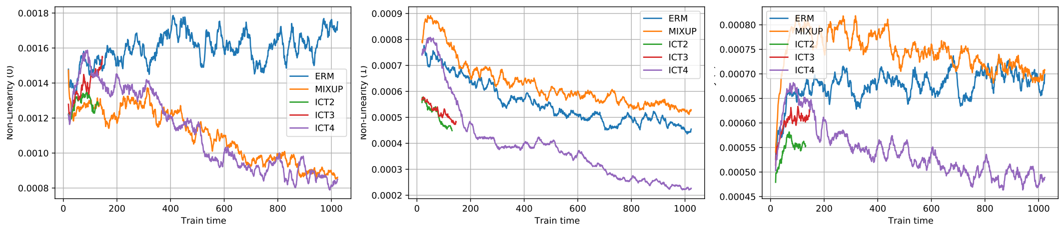


(e) Layer 4

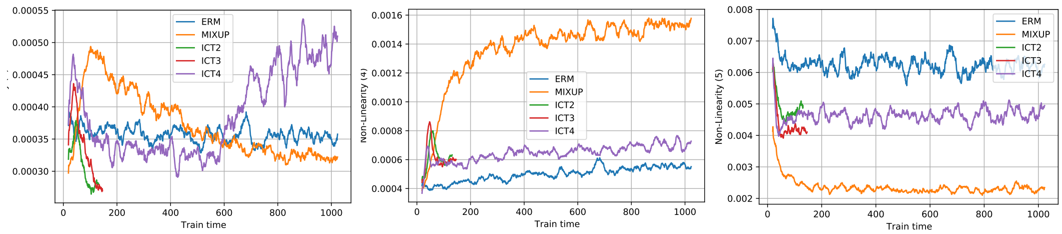


(f) Layer 5

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. $\texttt{ICT}_K$ 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.

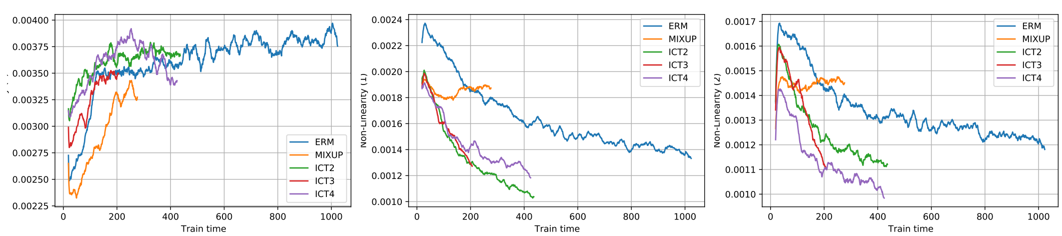


(a) Layer 0 (b) Layer 1 (c) Layer 2

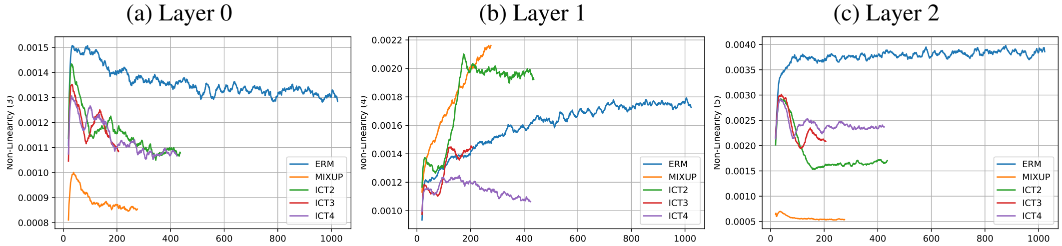


(d) Layer 3 (e) Layer 4 (f) Layer 5

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. $\texttt{ICT}_K$ 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.



(a) Layer 0 (b) Layer 1 (c) Layer 2



(d) Layer 3 (e) Layer 4 (f) Layer 5

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. $\texttt{ICT}_K$ 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.

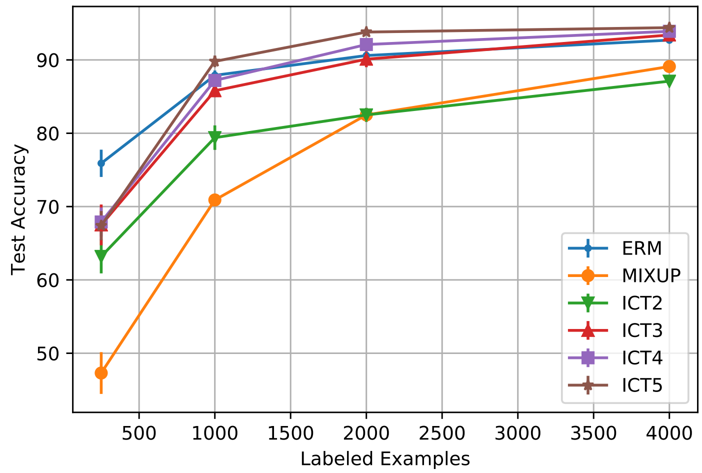


Figure 9: **SVHN**: Test accuracy when stronger linearity is enforced. ERM and mixup refer to the cases where only supervised loss is used. ICT[X] refer to a setup where the ICT loss with X mixup points are used as unsupervised loss. The supervised loss with the ICT does not use mixup loss.

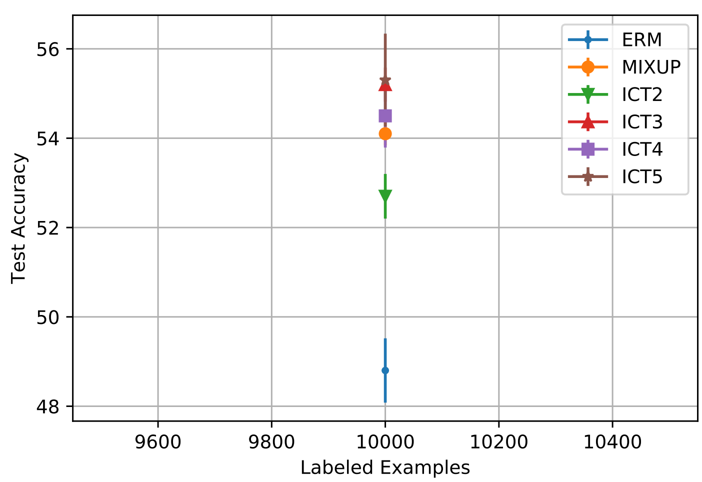


Figure 10: **CIFAR-100**: Test accuracy