Manifold Mixup: Better Representations by Interpolating Hidden States

Vikas Verma [*1 2] **Alex Lamb** [*2] **Christopher Beckham** [2] **Amir Najafi** [3] **Ioannis Mitliagkas** [2] **David Lopez-Paz** [4] **Yoshua Bengio** [2]

Abstract

Deep neural networks excel at learning the training data, but often provide incorrect and confident predictions when evaluated on slightly different test examples. This includes distribution shifts, outliers, and adversarial examples. To address these issues, we propose *Manifold Mixup*, a simple regularizer that encourages neural networks to predict less confidently on interpolations of hidden representations. *Manifold Mixup* leverages semantic interpolations as additional training signal, obtaining neural networks with smoother decision boundaries at multiple levels of representation. As a result, neural networks trained with *Manifold Mixup* learn flatter class-representations, that is, with fewer directions of variance. We prove theory on why this flattening happens under ideal conditions, validate it empirically on practical situations, and connect it to the previous works on information theory and generalization. In spite of incurring no significant computation and being implemented in a few lines of code, *Manifold Mixup* improves strong baselines in supervised learning, robustness to single-step adversarial attacks, and test log-likelihood.

1. Introduction

Deep neural networks are the backbone of state-of-the-art systems for computer vision, speech recognition, and language translation (LeCun et al., 2015). However, these systems perform well only when evaluated on instances very similar to those from the training set. When evaluated on slightly different distributions, neural networks often provide incorrect predictions with strikingly high confidence. This is a worrying prospect, since deep learning systems are being deployed in settings where data may be subject to distributional shifts. Adversarial examples (Szegedy et al., 2014) are one such failure case: deep neural networks with nearly perfect performance provide incorrect predictions with very high confidence when evaluated on perturbations imperceptible to the human eye. Adversarial examples are a serious hazard when deploying machine learning systems in security-sensitive applications. More generally, deep learning systems quickly degrade in performance as the distributions of training and testing data differ slightly from each other (Ben-David et al., 2010).

In this paper, we realize several troubling properties concerning the hidden representations and decision boundaries of state-of-the-art neural networks. First, we observe that the decision boundary is often sharp and close to the data. Second, we observe that the vast majority of the hidden representation space corresponds to high confidence predictions, both on and off of the data manifold.

Motivated by these intuitions we propose *Manifold Mixup* (Section 2), a simple regularizer that addresses several of these flaws by training neural networks on linear combinations of hidden representations of training examples. Previous work, including the study of analogies through word embeddings (e.g. king $-$ man $+$ woman $\approx$ queen), has shown that interpolations are an effective way of combining factors (Mikolov et al., 2013). Since high-level representations are often low-dimensional and useful to linear classifiers, linear interpolations of hidden representations should explore meaningful regions of the feature space effectively. To use combinations of hidden representations of data as novel training signal, we also perform the same linear interpolation in the associated pair of one-hot labels, leading to mixed examples with soft targets.

To start off with the right intuitions, Figure 1 illustrates the impact of *Manifold Mixup* on a simple two-dimensional classification task with small data. In this example, vanilla training of a deep neural network leads to an irregular decision boundary (Figure 1a), and a complex arrangement of hidden representations (Figure 1d). Moreover, every point in both the raw (Figure 1a) and hidden (Figure 1d) data representations is assigned a prediction with very high confidence.

[*]Equal contribution [1]Aalto University, Finland [2]Montréal Institute for Learning Algorithms (MILA) [3]Sharif University of Technology [4]Facebook Research Correspondence to: Alex Lamb <lambalex@iro.umontreal.ca>, Vikas Verma <vikasverma.iitm@gmail.com, vikas.verma@aalto.fi>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

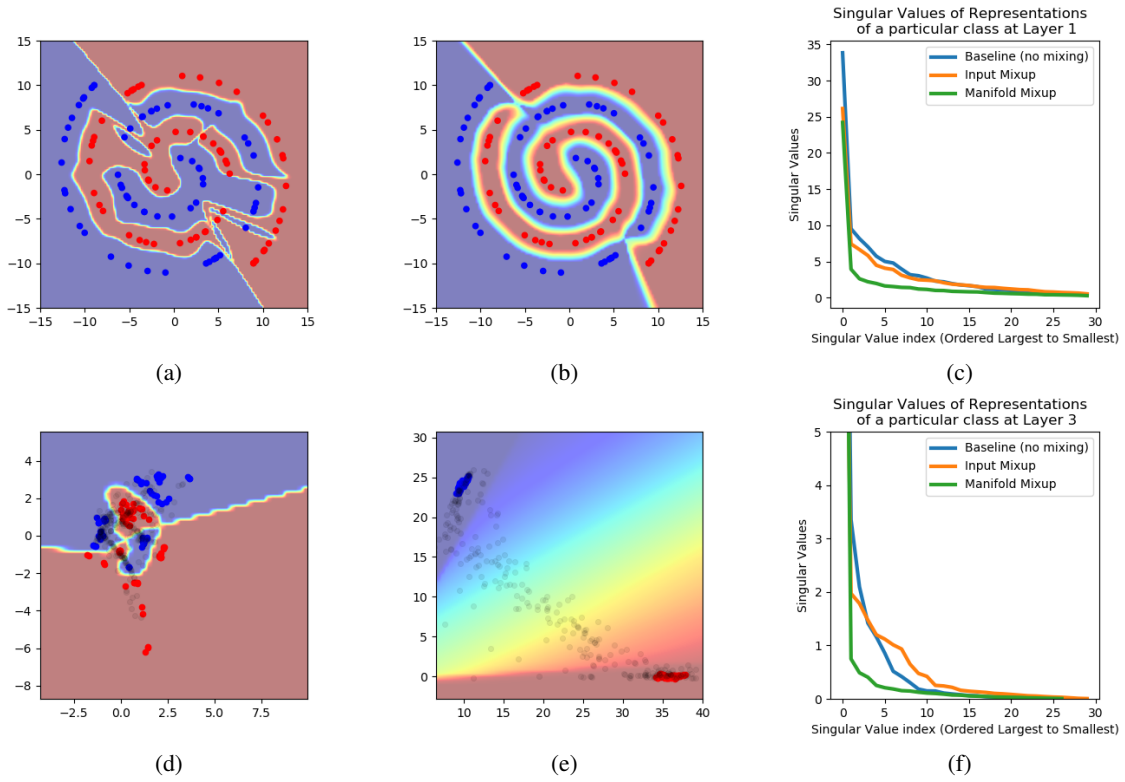

Figure 1: An experiment on a network trained on the 2D spiral dataset with a 2D bottleneck hidden state in the middle of the network. Manifold mixup has three effects on learning when compared to vanilla training. First, it smoothens decision boundaries (from a. to b.). Second, it improves the arrangement of hidden representations and encourages broader regions of low-confidence predictions (from d. to e.). Black dots are the hidden representation of the inputs sampled uniformly from the range of the input space. Third, it flattens the representations (c. at layer 1, f. at layer 3). Figure 2 shows that these effects are not accomplished by other well-studied regularizers (input mixup, weight decay, dropout, batch normalization, and adding noise to the hidden representations).

This includes points (depicted in black) that correspond to inputs from off of the data manifold! In contrast, training the same deep neural network with *Manifold Mixup* leads to a smoother decision boundary (Figure 1b) and a simpler (linear) arrangement of hidden representations (Figure 1e). In sum, the representations obtained by *Manifold Mixup* have two desirable properties: the class-representations are flattened into a minimal amount of directions of variation, and all points in-between these flat representations, most unobserved during training and off the data manifold, are assigned low-confidence predictions. This example conveys the central message of this paper:

Manifold mixup improves the hidden representations and decision boundaries of neural networks at multiple layers.

More specifically, *Manifold Mixup* improves generalization in deep neural networks because it:

- Leads to smoother decision boundaries that are further away from the training data, at multiple levels of representation. Smoothness and margin are well-established factors of generalization (Bartlett & Shawe-taylor, 1998; Lee et al., 1995).
- Leverages interpolations in deeper hidden layers, which capture higher level information (Zeiler & Fergus, 2013) to provide additional training signal.
- Flattens the class-representations, reducing their number of directions with significant variance (Section 3). This can be seen as a form of compression, which is linked to generalization by a well-established theory (Tishby & Zaslavsky, 2015; Shwartz-Ziv & Tishby, 2017).

Throughout a wide variety of experiments, we demonstrate four substantial benefits of *Manifold Mixup*:

- Better generalization than other competitive regularizers (such as Cutout, Mixup, AdaMix, and Dropout) (Section 5.1).

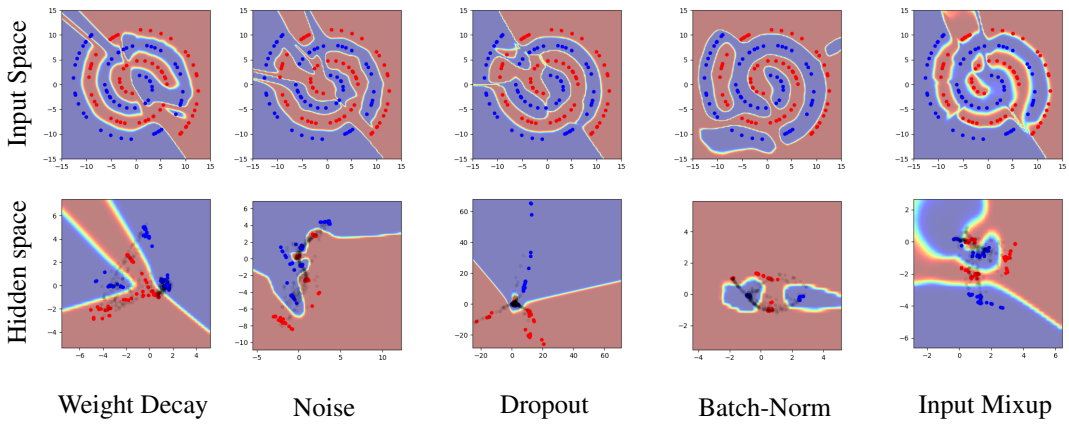

Figure 2: The same experimental setup as Figure 1, but using a variety of competitive regularizers. This shows that the effect of concentrating the hidden states for each class and providing a broad region of low confidence between the regions is not accomplished by the other regularizers (although input space mixup does produce regions of low confidence, it does not flatten the class-specific state distribution). Noise refers to gaussian noise in the input layer, dropout refers to dropout of 50% in all layers except the bottleneck itself (due to its low dimensionality), and batch normalization refers to batch normalization in all layers.

- Improved log-likelihood on test samples (Section 5.1).
- Increased performance at predicting data subject to novel deformations (Section 5.2).
- Improved robustness to single-step adversarial attacks. This is evidence *Manifold Mixup* pushes the decision boundary away from the data in some directions (Section 5.3). This is not to be confused with full adversarial robustness, which is defined in terms of moving the decision boundary away from the data in *all* directions.

2. Manifold Mixup

Consider training a deep neural network $f(x) = f_k(g_k(x))$, where g_k denotes the part of the neural network mapping the input data to the hidden representation at layer k, and f_k denotes the part mapping such hidden representation to the output $f(x)$. Training f using *Manifold Mixup* is performed in five steps. First, we select a random layer k from a set of eligible layers $\mathcal{S}$ in the neural network. This set may include the input layer $g_0(x)$. Second, we process two random data minibatches (x, y) and (x', y') as usual, until reaching layer k. This provides us with two intermediate minibatches $(g_k(x), y)$ and $(g_k(x'), y')$. Third, we perform Input Mixup (Zhang et al., 2018) on these intermediate minibatches. This produces the mixed minibatch:

$$(\tilde{g}_k, \tilde{y}) := (\text{Mix}_\lambda(g_k(x), g_k(x')), \text{Mix}_\lambda(y, y')),$$

where $\text{Mix}_\lambda(a, b) = \lambda \cdot a + (1 - \lambda) \cdot b$. Here, (y, y') are one-hot labels, and the mixing coefficient $\lambda \sim \text{Beta}(\alpha, \alpha)$ as proposed in mixup (Zhang et al., 2018). For instance, $\alpha = 1.0$ is equivalent to sampling $\lambda \sim U(0, 1)$. Fourth, we continue the forward pass in the network from layer k until the output using the mixed minibatch $(\tilde{g}_k, \tilde{y})$. Fifth, this output is used to compute the loss value and gradients that update the parameters of the neural network.

Mathematically, *Manifold Mixup* minimizes:

$$L(f) = \mathop{\mathbb{E}}_{(x,y)\sim P} \mathop{\mathbb{E}}_{(x',y')\sim P} \mathop{\mathbb{E}}_{\lambda\sim\text{Beta}(\alpha,\alpha)} \mathop{\mathbb{E}}_{k\sim\mathcal{S}} \ell(f_k(\text{Mix}_\lambda(g_k(x), g_k(x'))), \text{Mix}_\lambda(y, y')). \quad (1)$$

Some implementation considerations. We backpropagate gradients through the entire computational graph, including those layers before the mixup layer k (Section 5.1 and appendix Section B explore this issue in more detail). In the case where $\mathcal{S} = \{0\}$, *Manifold Mixup* reduces to the original mixup algorithm of Zhang et al. (2018). While one could try to reduce the variance of the gradient updates by sampling a random (k, λ) per example, we opted for the simpler alternative of sampling a single (k, λ) per minibatch, which in practice gives the same performance. As in Input Mixup, we use a single minibatch to compute the mixed minibatch. We do so by mixing the minibatch with copy of itself with shuffled rows.

3. Manifold Mixup Flattens Representations

We turn to the study of how *Manifold Mixup* impacts the hidden representations of a deep neural network. At a high level, *Manifold Mixup* flattens the class-specific representations. More specifically, this flattening reduces the number of directions with significant variance (akin to reducing their

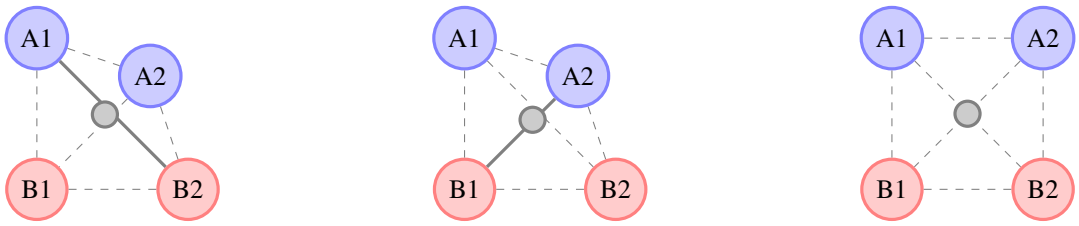

Figure 3: We show a simple case with two classes (interpolations depicted as grey lines) illustrating why Manifold Mixup learns flatter representations. The interpolation between A1 and B2 in the left panel soft-labels the black dot as 50% red and 50% blue, regardless of being very close to a blue point. In the middle panel a different interpolation between A2 and B1 soft-labels the same point as 95% blue and 5% red. However, since *Manifold Mixup learns* the hidden representations, the pressure to predict consistent soft-labels at interpolated points causes the states to become flattened (right panel).

number of principal components).

In the sequel, we first prove a theory (Section 3.1) that characterizes this behavior precisely under idealized conditions. Second, we show that this flattening also happens in practice, by performing the SVD of class-specific representations of neural networks trained on real datasets (Section 3.2). Finally, we discuss why the flattening of class-specific representations is a desirable property (Section 3.3).

3.1. Theory

We start by characterizing how the representations of a neural network are changed by *Manifold Mixup*, under a simplifying set of assumptions. More concretely, we will show that if one performs mixup in a sufficiently deep hidden layer in a neural network, then the loss can be driven to zero if the dimensionality of that hidden layer $\dim(\mathcal{H})$ is greater than the number of classes d. As a consequence of this, the resulting representations for that class will fall onto a subspace of dimension $\dim(\mathcal{H}) - d + 1$.

A more intuitive and less formal version of this argument is given in Figure 3 and Appendix F.

To this end, assume that $\mathcal{X}$ and $\mathcal{H}$ denote the input and representation spaces, respectively. We denote the label-set by $\mathcal{Y}$ and let $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. Let $\mathcal{G} \subseteq \mathcal{H}^{\mathcal{X}}$ denote the set of functions realizable by the neural network, from the input to the representation. Similarly, let $\mathcal{F} \subseteq \mathcal{Y}^{\mathcal{H}}$ be the set of all functions realizable by the neural network, from the representation to the output.

We are interested in the solution of the following problem in some asymptotic regimes:

$$J(P) = \inf_{g\in\mathcal{G}, f\in\mathcal{F}} \mathop{\mathbb{E}}_{(x,y),(x',y'),\lambda} \ell(f(\text{Mix}_\lambda(g(x), g(x'))), \text{Mix}_\lambda(y, y')). \tag{2}$$

More specifically, let P_D be the empirical distribution defined by a dataset $D = \{(x_i, y_i)\}_{i=1}^n$. Then, let $f^\star \in \mathcal{F}$ and $g^\star \in \mathcal{G}$ be the minimizers of (2) for $P = P_D$. Assume the number of layers are asymptotically increased, which results in $\mathcal{G} \to \mathcal{H}^{\mathcal{X}}$, $\mathcal{F} \to \mathcal{Y}^{\mathcal{H}}$, with $\mathcal{H}$ being a vector space. These is a direct consequence of the *universal approximation theorem* (Cybenko, 1989), which states that the mappings realizable by large neural networks are dense in the set of all continuous bounded functions. Under this setting, the objective (2) can be rewritten as:

$$J(P_D) = \inf_{h_1,\dots,h_n\in\mathcal{H}} \frac{1}{n(n-1)} \sum_{i\neq j}^{n} \left\{ \inf_{f\in\mathcal{F}} \int_0^1 \ell(f(\text{Mix}_\lambda(h_i, h_j)), \text{Mix}_\lambda(y_i, y_j))\, p(\lambda) \mathrm{d}\lambda \right\}, \tag{3}$$

where $h_i = g(x_i)$. We now propose our main theorem, which establishes sufficient conditions for Manifold Mixup to achieve zero training error.

Theorem 1. *Let $\mathcal{H}$ be a space with dimension $\dim(\mathcal{H})$, and let $d \in \mathbb{N}$ to represent the number of classes in a dataset D. If $\dim(\mathcal{H}) \geq d-1$, then $J(P_D) = 0$ and the corresponding minimizer $f^\star$ is a linear function from $\mathcal{H}$ to $\mathbb{R}^d$.*

Proof. First, we observe that the following statement is true if $\dim(\mathcal{H}) \geq d - 1$:

$$\exists\, A, H \in \mathbb{R}^{\dim(\mathcal{H})\times d}, b \in \mathbb{R}^d : A^\top H + b1_d^\top = I_{d\times d},$$

where $I_{d\times d}$ and 1_d denote the d-dimensional identity matrix and all-one vector, respectively. In fact, $b1_d^\top$ is a rank-one matrix, while the rank of identity matrix is d. Therefore, $A^\top H$ only needs to be rank $d - 1$.

Let $f^\star(h) = A^\top h + b$ for all $h \in \mathcal{H}$. Let $g^\star(x_i) = H_{\zeta_i,:}$ be the ζ_i-th column of H, where $\zeta_i \in \{1, \dots, d\}$ stands for the class-index of the example x_i. These choices minimize our objective (3) to zero, since:

$$\ell(f^\star(\text{Mix}_\lambda(g^\star(x_i), g^\star(x_j))), \text{Mix}_\lambda(y_i, y_j)) =$$
$$\ell(A^\top \text{Mix}_\lambda(H_{\zeta_i,:}, H_{\zeta_j,:}) + b, \text{Mix}_\lambda(y_{i,\zeta_i}, y_{j,\zeta_j})) =$$
$$\ell(u, u) = 0.$$

The result follows from $A^\top H_{\zeta_i,:} + b = y_{i,\zeta_i}$ for all i. □

Furthermore, if $\dim(\mathcal{H}) > d-1$, then data points in the representation space $\mathcal{H}$ have some degrees of freedom to move independently.

Corollary 1. *Consider the setting in Theorem 1 with* $\dim(\mathcal{H}) > d-1$. *Let* $g^\star \in \mathcal{G}$ *minimize* (2) *under* $P = P_D$. *Then, the representations of the training points* $g^\star(x_i)$ *fall on a* $(\dim(\mathcal{H}) - d + 1)$*-dimensional subspace.*

Proof. From the proof of Theorem 1, $A^\top H = I_{d\times d} - b1_d^\top$. The r.h.s. of this expression is a rank-$(d-1)$ matrix for a properly chosen b. Thus, A can have a null-space of dimension $\dim(\mathcal{H}) - d + 1$. This way, one can assign $g^\star(x_i) = H_{\zeta_i,:} + e_i$, where $H_{\zeta_i,:}$ is defined as in the proof of Theorem 1, and e_i are arbitrary vectors in the null-space of A, for all $i = 1, \ldots, n$. □

This result implies that if the *Manifold Mixup* loss is minimized, then the representation of each class lies on a subspace of dimension $\dim(\mathcal{H}) - d + 1$. In the extreme case where $\dim(\mathcal{H}) = d-1$, each class representation will collapse to a single point, meaning that hidden representations would not change in any direction, for each class-conditional manifold. In the more general case with larger $\dim(\mathcal{H})$, the majority of directions in $\mathcal{H}$-space will contain zero variance in the class-conditional manifold.

3.2. Empirical Investigation of Flattening

We now show that the "flattening" theory that we have just developed also holds in practice, for real neural networks networks trained on real data. To this end, we trained a collection of fully-connected neural networks on the MNIST dataset using multiple regularizers, including *Manifold Mixup*. When using *Manifold Mixup*, we mixed representations at a single, fixed hidden layer per network. After training, we performed the Singular Value Decomposition (SVD) of the hidden representations of each network, and analyzed their spectrum decay.

More specifically, we computed the largest singular value per class, as well as the sum of the all other singular values. We computed these statistics at the first hidden layer for all networks and regularizers. For the largest singular value, we obtained: 51.73 (baseline), 33.76 (weight decay), 28.83 (dropout), 33.46 (input mixup), and 31.65 (manifold mixup). For the sum of all the other singular values, we obtained: 78.67 (baseline), 73.36 (weight decay), 77.47 (dropout), 66.89 (input mixup), and 40.98 (manifold mixup). Therefore, weight decay, dropout, and input mixup all reduce the largest singular value, but only *Manifold Mixup* achieves a reduction of the sum of the all other singular values (e.g. flattening). For more details regarding this experiment, consult Appendix G.

3.3. Why is Flattening Representations Desirable?

We have presented evidence to conclude that *Manifold Mixup* leads to flatter class-specific representations, and that such flattening is not accomplished by other regularizers.

But why is this flattening desirable? First, it means that the hidden representations computed from our data occupy a much smaller volume. Thus, a randomly sampled hidden representation within the convex hull spanned by the data in this space is more likely to have a classification score with lower confidence (higher entropy). Second, compression has been linked to generalization in the information theory literature (Tishby & Zaslavsky, 2015; Shwartz-Ziv & Tishby, 2017).

4. Related Work

Regularization is a major area of research in machine learning. *Manifold Mixup* is a generalization of Input Mixup, the idea of building random interpolations between training examples and perform the same interpolation for their labels (Zhang et al., 2018; Tokozume et al., 2018).

Intriguingly, our experiments show that *Manifold Mixup* changes the representations associated to the layers before and after the mixing operation, and that this effect is crucial to achieve good results (Section 5.1, Appendix G). This suggests that *Manifold Mixup* may work for different reasons than Input Mixup.

Another line of research closely related to *Manifold Mixup* involves regularizing deep networks by perturbing their hidden representations. These methods include dropout (Hinton et al., 2012), batch normalization (Ioffe & Szegedy, 2015), and the information bottleneck (Alemi et al., 2017). Notably, (Hinton et al., 2012) and (Ioffe & Szegedy, 2015) demonstrated that regularizers that work well in the input space can also be applied to the hidden layers of a deep network, often to further improve results. We believe that *Manifold Mixup* is a complimentary form of regularization.

Zhao & Cho (2018) explored improving adversarial robustness by classifying points using a function of the nearest neighbors in a fixed feature space. This involves applying mixup between each set of nearest neighbor examples in that feature space. The similarity between (Zhao & Cho, 2018) and *Manifold Mixup* is that both consider linear interpolations of hidden representations with the same interpolation applied to their labels. However, an important difference is that *Manifold Mixup* backpropagates gradients through the earlier parts of the network (the layers before the point where mixup is applied), unlike (Zhao & Cho, 2018). In Section 3 we explain how this discrepancy significantly affects the learning process.

AdaMix (Guo et al., 2018a) is another related method which

local线性假设的那一篇

attempts to learn better mixing distributions to avoid overlap. AdaMix performs interpolations only on the input space, reporting that their method degrades significantly when applied to hidden layers. Thus, AdaMix may likely work for different reasons than *Manifold Mixup*, and perhaps the two are complementary. AgrLearn (Guo et al., 2018b) adds an information bottleneck layer to the output of deep neural networks. AgrLearn leads to substantial improvements, achieving 2.45% test error on CIFAR-10 when combined with Input Mixup (Zhang et al., 2018). As AgrLearn is complimentary to Input Mixup, it may be also complimentary to *Manifold Mixup*. Wang et al. (2018) proposed an interpolation exclusively in the output space, does not backpropagate through the interpolation procedure, and has a very different framing in terms of the Euler-Lagrange equation (Equation 2) where the cost is based on unlabeled data (and the pseudolabels at those points) and the labeled data provide constraints.

5. Experiments

We now turn to the empirical evaluation of *Manifold Mixup*. We will study its regularization properties in supervised learning (Section 5.1), as well as how it affects the robustness of neural networks to novel input deformations (Section 5.2), and adversarial examples (Section 5.3).

5.1. Generalization on Supervised Learning

We train a variety of residual networks (He et al., 2016) using different regularizers: no regularization, AdaMix, Input Mixup, and *Manifold Mixup*. We follow the training procedure of (Zhang et al., 2018), which is to use SGD with momentum, a weight decay of 10^{-4}, and a step-wise learning rate decay. Please refer to Appendix C for further details (including the values of the hyperparameter α). We show results for the CIFAR-10 (Table 1a), CIFAR-100 (Table 1b), SVHN (Table 2), and TinyImageNET (Table 3) datasets. *Manifold Mixup* outperforms vanilla training, AdaMix, and Input Mixup across datasets and model architectures. Furthermore, *Manifold Mixup* leads to models with significantly better Negative Log-Likelihood (NLL) on the test data. In the case of CIFAR-10, *Manifold Mixup* models achieve as high as 50% relative improvement of test NLL.

As a complimentary experiment to better understand why *Manifold Mixup* works, we zeroed gradient updates immediately after the layer where mixup is applied. On the dataset CIFAR-10 and using a PreActResNet18, this led to a 4.33% test error, which is worse than our results for Input Mixup and *Manifold Mixup*, yet better than the baseline. Because *Manifold Mixup* select the mixing layer at random, each layer is still being trained even when zeroing gradients, although it will receive less updates. This demonstrates that *Manifold Mixup* improves performance by updating the layers both before and after the mixing operation.

We also compared *Manifold Mixup* against other strong regularizers. For each regularizer, we selected the best hyper-parameters using a validation set. Then, training a PreActResNet50 on CIFAR-10 for 600 epochs led to the following test errors (%): no regularization (4.96 ± 0.19), Dropout (5.09 ± 0.09), Cutout (Devries & Taylor, 2017) (4.77 ± 0.38), Mixup (4.25 ± 0.11), and Manifold Mixup (3.77 ± 0.18). (Note that the results in Table 1 for PreActResNet were run for 1200 epochs, and therefore are not directly comparable to the numbers in this paragraph.)

To provide further evidence about the quality of representations learned with *Manifold Mixup*, we applied a k-nearest neighbour classifier on top of the features extracted from a PreActResNet18 trained on CIFAR-10. We achieved test errors of 6.09% (vanilla training), 5.54% (Input Mixup), and 5.16% (*Manifold Mixup*).

Finally, we considered a synthetic dataset where the data generating process is a known function of disentangled factors of variation, and mixed in this space factors. As shown in Appendix A, this led to significant improvements in performance. This suggests that mixing in the correct level of representation has a positive impact on the decision boundary. However, our purpose here is not to make any claim about when do deep networks learn representations corresponding to disentangled factors of variation.

Finally, Table 4 and Table 6 show the sensitivity of *Manifold Mixup* to the hyper-parameter α and the set of eligible layers $\mathcal{S}$. (These results are based on training a PreActResNet18 for 2000 epochs, so these numbers are not exactly comparable to the ones in Table 1.) This shows that *Manifold Mixup* is robust with respect to choice of hyper-parameters, with improvements for many choices.

5.2. Generalization to Novel Deformations

To further evaluate the quality of representations learned with *Manifold Mixup*, we train PreActResNet34 models on the normal CIFAR-100 training split, but test them on novel (not seen during training) deformations of the test split. These deformations include random rotations, random shearings, and different rescalings. Better representations should generalize to a larger variety of deformations. Table 5 shows that networks trained using *Manifold Mixup* are the most able to classify test instances subject to novel deformations, which suggests the learning of better representations. For more results see Appendix C, Table 9.

5.3. Robustness to Adversarial Examples

Adversarial robustness is related to the position of the decision boundary relative to the data. Because *Manifold Mixup* only considers some directions around data points (those

Table 1: Classification errors on (a) CIFAR-10 and (b) CIFAR-100. We include results from (Zhang et al., 2018)† and (Guo et al., 2016)‡. We run experiments five times to report the mean and the standard deviation of errors and neg-log-likelihoods.

PreActResNet18	Test Error (%)	Test NLL
No Mixup	4.83 ± 0.066	0.190 ± 0.003
AdaMix‡	3.52	NA
Input Mixup†	4.20	NA
Input Mixup ($\alpha = 1$)	3.82 ± 0.048	0.186 ± 0.004
Manifold Mixup ($\alpha = 2$)	2.95 ± 0.046	0.137 ± 0.003
PreActResNet34		
No Mixup	4.64 ± 0.072	0.200 ± 0.002
Input Mixup ($\alpha = 1$)	2.88 ± 0.043	0.176 ± 0.002
Manifold Mixup ($\alpha = 2$)	2.54 ± 0.047	0.118 ± 0.002
Wide-Resnet-28-10		
No Mixup	3.99 ± 0.118	0.162 ± 0.004
Input Mixup ($\alpha = 1$)	2.92 ± 0.088	0.173 ± 0.001
Manifold Mixup ($\alpha = 2$)	2.55 ± 0.024	0.111 ± 0.001

(a) CIFAR-10

PreActResNet18	Test Error (%)	Test NLL
No Mixup	24.01 ± 0.376	1.189 ± 0.002
AdaMix‡	20.97	n/a
Input Mixup†	21.10	n/a
Input Mixup ($\alpha = 1$)	22.11 ± 0.424	1.055 ± 0.006
Manifold Mixup ($\alpha = 2$)	20.34 ± 0.525	0.912 ± 0.002
PreActResNet34		
No Mixup	23.55 ± 0.399	1.189 ± 0.002
Input Mixup ($\alpha = 1$)	20.53 ± 0.330	1.039 ± 0.045
Manifold Mixup ($\alpha = 2$)	18.35 ± 0.360	0.877 ± 0.053
Wide-Resnet-28-10		
No Mixup	21.72 ± 0.117	1.023 ± 0.004
Input Mixup ($\alpha = 1$)	18.89 ± 0.111	0.927 ± 0.031
Manifold Mixup ($\alpha = 2$)	18.04 ± 0.171	0.809 ± 0.005

(b) CIFAR-100

Table 2: Classification errors and neg-log-likelihoods on SVHN. We run each experiment five times.

PreActResNet18	Test Error (%)	Test NLL
No Mixup	2.89 ± 0.224	0.136 ± 0.001
Input Mixup ($\alpha = 1$)	2.76 ± 0.014	0.212 ± 0.011
Manifold Mixup ($\alpha = 2$)	2.27 ± 0.011	0.122 ± 0.006
PreActResNet34		
No Mixup	2.97 ± 0.004	0.165 ± 0.003
Input Mixup ($\alpha = 1$)	2.67 ± 0.020	0.199 ± 0.009
Manifold Mixup ($\alpha = 2$)	2.18 ± 0.004	0.137 ± 0.008
Wide-Resnet-28-10		
No Mixup	2.80 ± 0.044	0.143 ± 0.002
Input Mixup ($\alpha = 1$)	2.68 ± 0.103	0.184 ± 0.022
Manifold Mixup ($\alpha = 2$)	2.06 ± 0.068	0.126 ± 0.008

Table 3: Classification accuracies on TinyImagenet.

PreActResNet18	top-1	top-5
No Mixup	55.52	71.04
Input Mixup ($\alpha = 0.2$)	56.47	71.74
Input Mixup ($\alpha = 0.5$)	55.49	71.62
Input Mixup ($\alpha = 1.0$)	52.65	70.70
Input Mixup ($\alpha = 2.0$)	44.18	68.26
Manifold Mixup ($\alpha = 0.2$)	58.70	73.59
Manifold Mixup ($\alpha = 0.5$)	57.24	73.48
Manifold Mixup ($\alpha = 1.0$)	56.83	73.75
Manifold Mixup ($\alpha = 2.0$)	48.14	71.69

Table 4: Test accuracy (%) of Input Mixup and *Manifold Mixup* for different α on CIFAR-10.

α	Input Mixup	*Manifold Mixup*
0.5	96.68	96.76
1.0	96.75	97.00
1.2	96.72	97.03
1.5	96.84	97.10
1.8	96.80	97.15
2.0	96.73	97.23

corresponding to interpolations), we would not expect the model to be robust to adversarial attacks that consider any direction around each example. However, since *Manifold Mixup* expands the set of examples seen during training, an intriguing hypothesis is that these expansions overlap with the set of possible adversarial examples, providing some degree of defense. If this hypothesis is true, *Manifold Mixup* would force adversarial attacks to consider a wider set of directions, leading to a larger computational expense for the attacker. To explore this, we consider the Fast Gradient Sign Method (FGSM, Goodfellow et al., 2015), which constructs adversarial examples in one single step, thus considering a relatively small subset of directions around examples. The performance of networks trained using *Manifold Mixup* against FGSM attacks is given in Table 7. One challenge in evaluating robustness against adversarial examples is the "gradient masking problem", in which a defense succeeds only by reducing the quality of the gradient signal. (Athalye et al., 2018) explored this issue in depth, and proposed running an unbounded search for a large number of iterations to

Table 5: Test accuracy on samples subject to novel deformations. All models were trained on normal CIFAR-100.

Deformation	No Mixup	Input Mixup ($\alpha = 1$)	Input Mixup ($\alpha = 2$)	*Manifold Mixup* ($\alpha = 2$)
Rotation U(−20°,20°)	52.96	55.55	56.48	60.08
Rotation U(−40°,40°)	33.82	37.73	36.78	42.13
Shearing U(−28.6°, 28.6°)	55.92	58.16	60.01	62.85
Shearing U(−57.3°, 57.3°)	35.66	39.34	39.7	44.27
Zoom In (60% rescale)	12.68	13.75	13.12	11.49
Zoom In (80% rescale)	47.95	52.18	50.47	52.70
Zoom Out (120% rescale)	43.18	60.02	61.62	63.59
Zoom Out (140% rescale)	19.34	41.81	42.02	45.29

Table 6: Test accuracy (%) of *Manifold Mixup* for different sets of eligible layers $\mathcal{S}$ on CIFAR-10/CIFAR-100.

$\mathcal{S}$	CIFAR-10	CIFAR-100
$\{0, 1, 2\}$	97.23	79.60
$\{0, 1\}$	96.94	78.93
$\{0, 1, 2, 3\}$	96.92	80.18
$\{1, 2\}$	96.35	78.69
$\{0\}$	96.73	78.15
$\{1, 2, 3\}$	96.51	79.31
$\{1\}$	96.10	78.72
$\{2, 3\}$	95.32	76.46
$\{2\}$	95.19	76.50
$\{\}$	95.27	76.40

Table 7: Test accuracy on white-box FGSM adversarial examples on CIFAR-10/CIFAR-100 (using a PreActResNet18 model) and SVHN (using a WideResNet20-10 model). We include the results of (Madry et al., 2018)†.

CIFAR-10	FGSM
No Mixup	36.32
Input Mixup ($\alpha = 1$)	71.51
Manifold Mixup ($\alpha = 2$)	77.50
PGD training (7-steps)†	56.10
CIFAR-100	**FGSM**
Input Mixup ($\alpha = 1$)	40.7
Manifold Mixup ($\alpha = 2$)	44.96
SVHN	**FGSM**
No Mixup	21.49
Input Mixup ($\alpha = 1$)	56.98
Manifold Mixup ($\alpha = 2$)	65.91
PGD training (7-steps)†	72.80

confirm the quality of the gradient signal. *Manifold Mixup* passes this sanity check (consult Appendix D for further details). While we found that using *Manifold Mixup* improves the robustness to single-step FGSM attack (especially over Input Mixup), we found that *Manifold Mixup* did not significantly improve robustness against stronger, multi-step attacks such as PGD (Madry et al., 2018).

6. Conclusion

Deep neural networks often give incorrect, yet extremely confident predictions on examples that differ from those seen during training. This problem is one of the most central challenges in deep learning. We have investigated this issue from the perspective of the representations learned by deep neural networks. We observed that vanilla neural networks spread the training data widely throughout the representation space, and assign high confidence predictions to almost the entire volume of representations. This leads to major drawbacks since the network will provide high-confidence predictions to examples off the data manifold, thus lacking enough incentives to learn discriminative representations about the training data. To address these issues, we introduced *Manifold Mixup*, a new algorithm to train neural networks on interpolations of hidden representations. *Manifold Mixup* encourages the neural network to be uncertain across the volume of the representation space unseen during training. This leads to concentrating the representations of the real training examples in a low dimensional subspace, resulting in more discriminative features. Throughout a variety of experiments, we have shown that neural networks trained using *Manifold Mixup* have better generalization in terms of error and log-likelihood, as well as better robustness to novel deformations of the data and adversarial examples. Being easy to implement and incurring little additional computational cost, we hope that *Manifold Mixup* will become a useful regularization tool for deep learning practitioners.

Acknowledgements

Vikas Verma was supported by Academy of Finland project 13312683 / Raiko Tapani AT kulut. We would also like to acknowledge Compute Canada for providing computing resources used in this work.

References

Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. Deep variational information bottleneck. In *International Conference on Learning Representations*, 2017.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pp. 214–223, 2017.

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 274–283, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/athalye18a.html.

Bartlett, P. and Shawe-taylor, J. Generalization performance of support vector machines and other pattern classifiers, 1998.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Devries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017. URL http://arxiv.org/abs/1708.04552.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*, 2015.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pp. 5769–5779, 2017.

Guo, H., Mao, Y., and Zhang, R. MixUp as Locally Linear Out-Of-Manifold Regularization. *ArXiv e-prints*, 2016. URL https://arxiv.org/abs/1809.02499.

Guo, H., Mao, Y., and Zhang, R. MixUp as Locally Linear Out-Of-Manifold Regularization. *ArXiv e-prints*, September 2018a.

Guo, H., Mao, Y., and Zhang, R. Aggregated Learning: A Vector Quantization Approach to Learning with Neural Networks. *ArXiv e-prints*, July 2018b.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *ECCV*, 2016.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. URL http://arxiv.org/abs/1207.0580.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436, 2015.

Lee, W. S., Bartlett, P. L., and Williamson, R. C. Lower bounds on the vc dimension of smoothly parameterized function classes. *Neural Computation*, 7(5):1040–1053, Sep. 1995. ISSN 0899-7667. doi: 10.1162/neco.1995.7.5.1040.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1QRgziT-.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

Shwartz-Ziv, R. and Tishby, N. Opening the black box of deep neural networks via information. *CoRR*, abs/1703.00810, 2017. URL http://arxiv.org/abs/1703.00810.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

Tishby, N. and Zaslavsky, N. Deep learning and the information bottleneck principle. *CoRR*, abs/1503.02406, 2015. URL http://arxiv.org/abs/1503.02406.

Tokozume, Y., Ushiku, Y., and Harada, T. Between-class learning for image classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Wang, B., Luo, X., Li, Z., Zhu, W., Shi, Z., and Osher, S. J. Deep learning with data dependent implicit activation function. *CoRR*, abs/1802.00168, 2018. URL http://arxiv.org/abs/1802.00168.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013. URL http://arxiv.org/abs/1311.2901.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1Ddp1-Rb.

Zhao, J. and Cho, K. Retrieval-augmented convolutional neural networks for improved robustness against adversarial examples. *CoRR*, abs/1802.09502, 2018. URL http://arxiv.org/abs/1802.09502.