Manifold Mixup: Encouraging Meaningful On-Manifold Interpolation as a Regularizer

Vikas Verma* †
Aalto Univeristy, Finland
vikas.verma@aalto.fi

Alex Lamb*
Montréal Institute for Learning Algorithms
lambalex@iro.umontreal.ca

Christopher Beckham
Montréal Institute for Learning Algorithms
christopher.j.beckham@gmail.com

Aaron Courville
Montréal Institute for Learning Algorithms
aaron.courville@gmail.com

Ioannis Mitliagkas
Montréal Institute for Learning Algorithms
imitliagkas@gmail.com

Yoshua Bengio
Montréal Institute for Learning Algorithms
CIFAR Senior Fellow
yoshua.umontreal@gmail.com

Abstract

Deep networks often perform well on the data manifold on which they are trained, yet give incorrect (and often very confident) answers when evaluated on points from off of the training distribution. This is exemplified by the adversarial examples phenomenon but can also be seen in terms of model generalization and domain shift. We propose *Manifold Mixup* which encourages the network to produce more reasonable and less confident predictions at points with combinations of attributes not seen in the training set. This is accomplished by training on convex combinations of the hidden state representations of data samples. Using this method, we demonstrate improved semi-supervised learning, learning with limited labeled data, and robustness to novel transformations of the data not seen during training. *Manifold Mixup* requires no (significant) additional computation. We also discover intriguing properties related to adversarial examples and generative adversarial networks. Analytical experiments on both real data and synthetic data directly support our hypothesis for why the *Manifold Mixup* method improves results.

1 Introduction

Machine learning systems have been enormously successful in domains such as vision, speech, and language and are now widely used both in research and industry. Modern machine learning systems typically only perform well when evaluated on the same distribution that they were trained on. However machine learning systems are increasingly being deployed in settings where the environment is noisy, subject to domain shifts, or even adversarial attacks. In many cases, deep neural networks

Preprint. Work in progress. * Equal contribution. † Work done while author was visiting Montreal Institute for Learning Algorithms. Code available at https://github.com/vikasverma1077/manifold_mixup

which perform extremely well when evaluated on points on the data manifold give incorrect answers when evaluated on points off the training distribution, and with strikingly high confidence.

This manifests itself in several failure cases for deep learning. One is the problem of adversarial examples (Szegedy et al., 2013), in which deep neural networks with nearly perfect test accuracy can produce incorrect classifications with very high confidence when evaluated on data points with small (imperceptible to human vision) adversarial perturbations. These adversarial examples could present serious security risks for machine learning systems. Another failure case involves the training and testing distributions differing significantly. With deep neural networks, this can often result in dramatically reduced performance.

These challenges have motivated the study of machine learning systems which generalize well at points off of the training distribution. One approach to address these issues is *data augmentation*, which involves synthesizing new points and training on them along with the original data. Perhaps the most straightforward way to produce novel data points is to perturb existing data points with noise. However in high-dimensional space, when the data distribution is concentrated near lower-dimensional manifolds, isotropic noise almost never approximates the kind of variations which can exist in real data. In high-dimensional images, for example, independent random noise can be trivially removed by exploiting strong local spatial correlations. Structured transformations (such as rotations and translations in images) have also been explored, but the effectiveness of this kind of data augmentation is domain specific. In addition, these augmentation strategies often fail to capture the kinds of variations that separate the training and test distributions.

The goal of our proposed algorithm, *Manifold Mixup* [1] , is to increase the generality and effectiveness of data augmentation by using a deep network's learned representations as a way of generating novel data points for training. For example, in a dataset of images of dogs and cats, one could consider a non-linear interpolation between a pair of dogs to produce novel examples of a dog that possesses some factors of each of the pair in a novel combination (one such factor combination would be to combine the head from one dog with the body of another dog). Some real examples of this kind of interpolation from our model are seen in figure 4.

The space of such high-level factors has lower dimensionality but still contains significant information about the inputs — through the disentangling effect of deep networks (Bengio et al., 2012) – that combining samples in a more abstract representation space is much more efficient and more of it can be explored with a relatively simple search procedure, such as linear interpolations.

We propose to take convex combinations of the hidden state representations of data samples. Previous work, including the study of analogies through word embeddings (e.g. king - man + woman $\approx$ queen), has shown that such linear interpolation between hidden states is an effective way of combining factors (Mikolov et al., 2013). Combining such factors in the higher level representations has the advantage that it is typically lower dimensional, so a simple procedure like linear interpolation between pairs of data points explores more of the space and with more of the points having meaningful semantics. Following the label interpolation approach proposed in *Mixup* (Zhang et al., 2017), when we combine the hidden representations of training examples, we also perform the same linear interpolation in the labels (seen as one-hot vectors or categorical distributions), producing new soft targets for the mixed examples.

Experimentally, we observe that *Manifold Mixup* can act as an effective regularizer. In particular, we show performance gains when training with limited labeled data and in semi-supervised learning. We also show that training with *Manifold Mixup* can increase robustness to the single step FGSM adversarial attack as well as to artificial deformations (rotations, shearing, and re-scaling) not seen during training, while requiring no additional (significant) computation. We also conduct a set of analytical experiments to explore and attempt to explain exactly how *Manifold Mixup* regularizes network training.

[1] We select the name *Manifold Mixup* because the properties of data lying on a manifold make input space data augmentation less desirable than representation-based augmentation. We make no strong claims about these representations perfectly capturing the data manifold.

2 Manifold Mixup

The *Manifold Mixup* algorithm consists of selecting a random layer (from a set of eligible layers including the input layer) k. We then process the batch without any mixup until reaching that layer, and we perform mixup at that hidden layer, and then continue processing the network starting from the mixed hidden state, changing the target vector according to the mixup interpolation. More formally, we can redefine our neural network function $y = f(x)$ in terms of k: $f(x) = g_k(h_k(x))$. Here g_k is a function which runs a neural network from the input hidden state k to the output y, and h_k is a function which computes the k-th hidden layer activation from the input x.

For the linear interpolation between factors, we define a variable λ and we sample from $p(\lambda)$. Following (Zhang et al., 2017), we always use a beta distribution $p(\lambda) = Beta(\alpha, \alpha)$. With $\alpha = 1.0$, this is equivalent to sampling from $U(0, 1)$. In general we found that $\alpha = 2.0$ was often a good choice when using *Manifold Mixup*.

We consider data augmentation in the set of layers S_k and minimize the expected *Manifold Mixup* loss.

$$L = \mathbb{E}_{(x_i,y_i),(x_j,y_j)\sim p(x,y),\lambda\sim p(\lambda),k\sim S_k} \ell(g_k(\lambda h_k(x_i) + (1-\lambda)h_k(x_j))), \lambda y_i + (1-\lambda)y_j)$$

We backpropagate gradients through the entire computational graph, including to layers before the mixup process is applied (section 4.1 and Figure 5 explore this issue directly). In the case where $k = 0$ is the input layer and $S_k = 0$, *Manifold Mixup* reduces to the mixup algorithm of (Zhang et al., 2017). With $\alpha = 2.0$, about 5% of the time λ is within 5% of 0 or 1, which essentially means that an ordinary example is presented. In the more general case, we can optimize the expectation in the *Manifold Mixup* objective using stochastic gradients, by sampling a different layer to perform mixup in on each update. We could also select a new random layer as well as a new lambda for each example in the minibatch. In this theory this should reduce the variance in the updates introduced by these random variables. However in practice we found that this didn't significantly effect the results, so we decided to sample a single lambda and randomly chosen layer per minibatch.

The original mixup is done by randomly drawing two minibatches, and mixing between them, yet in practice they only draw a single minibatch and mixup the minibatch with a shuffled version of itself. For a sufficiently large minibatch, this approaches the procedure of drawing two random minibatches. This was performed in the mixup paper (Zhang et al., 2017) and *Manifold Mixup* reuses this same procedure as it is simpler and more efficient.

The results in the Figure 1 demonstrate that *Manifold Mixup* reduces the loss calculated along hidden interpolations significantly better than Input Mixup, without significantly changing the loss calculated along visible space interpolations.

3 Related Work

Regularization is a major area of research in machine learning. *Manifold Mixup* closely builds on two threads of research. The first is the idea of linearly interpolating between different randomly drawn examples and similarly interpolating the labels (Zhang et al., 2017; Tokozume et al., 2017). This line of work builds on other data augmentation techniques as well as methods which smooth the labels and encourage higher entropy in the softmax (Pereyra et al., 2017; Szegedy et al., 2015). Intriguingly our experiments show that *Manifold Mixup* changes the behavior of the layers both before and after the mixing operation and that it's crucial to achieving good results (section 4.1 and Figure 5), suggesting that the method works for different reasons than the previously studied Input Mixup.

Another line of research closely related to *Manifold Mixup* involves regularizing deep networks by perturbing the hidden states of the network. These methods include dropout (Hinton et al., 2012), batch normalization (Ioffe & Szegedy, 2015), and the information bottleneck (Alemi et al., 2016). Notably (Hinton et al., 2012) and (Ioffe & Szegedy, 2015) both demonstrated that regularizers already demonstrated to work well in the input space (salt and pepper noise and input normalization respectively) could also be adapted to improve results when applied to the hidden layers of a deep network. We believe that the regularization effect of *Manifold Mixup* would be complementary to that of these algorithms. In particular, the additional independence of hidden units induced by the

use of dropout would likely render our use of linear interpolation in representation space even more effective.

(Zhao & Cho, 2018) explored improving adversarial robustness by classifying points using a function of the nearest neighbors in a fixed feature space. This involved applying Mixup between each set of nearest neighbor examples in that feature space. An important similarity between (Zhao & Cho, 2018) and *Manifold Mixup* is that both consider linear interpolations in hidden states with the same interpolation applied to the labels. However an important difference is that *Manifold Mixup* considers a different randomly selected hidden layer on each update and another difference is that *Manifold Mixup* considers pairs drawn from all examples when doing the interpolations, whereas (Zhao & Cho, 2018) only considers pairs drawn from the set of nearest neighbors. Another important difference is that *Manifold Mixup* backpropagates gradients through the earlier parts of the network and (Zhao & Cho, 2018) used mixup in the space of a fixed feature extractor. In 4.1 and 5 this was found to significantly change the learning process.

4 Experiments

4.1 Regularization on Supervised Learning

We present results on *Manifold Mixup* based regularization of networks using the PreActResNet architecture (He et al., 2016). We closely followed the procedure of (Zhang et al., 2017) as a way of providing direct comparisons with the Input Mixup algorithm. We used weight decay of 0.0001 and trained with SGD with momentum and multiplied the learning rate by 0.1 at regularly scheduled epochs. These results for CIFAR-10 and CIFAR-100 are in Table 1 and Table 2. We also ran experiments where took PreActResNet34 models trained on the normal CIFAR-100 data and evaluated them on test sets with artificial deformations (shearing, rotation, and zooming) and showed that *Manifold Mixup* demonstrated significant improvements (Table 3). We also show that the number of epochs needed to reach good results is not significantly affected by using *Manifold Mixup* in Figure 6.

To better understand why the method works, we performed an experiment where we trained with *Manifold Mixup* but blocked gradients immediately after the layer where we perform mixup. On CIFAR-10 PreActResNet18, this caused us to achieve 95.14% test accuracy when trained on 400 epochs and 95.67% test accuracy when trained on 1200 epochs. This is better than the baseline, but worse than *Manifold Mixup* or Input Mixup in both cases. Because we randomly select the layer to mix, each layer of the network is still being trained, although not on every update. This demonstrates that the *Manifold Mixup* method improves results by changing the layers both before and after the mixup operation is applied.

Table 1: Supervised Classification Results on CIFAR-10. We note significant improvement with *Manifold Mixup* especially in terms of likelihood. Please refer to Appendix A for details on the implementation of *Manifold Mixup* and *Manifold Mixup* All layers.

Model	Test Acc	Test NLL
PreActResNet18		
No Mixup	94.88	0.2646
Input Mixup ($\alpha = 1.0$) (Zhang et al., 2017)	96.10	n/a
Input Mixup ($\alpha = 1.0$) (ours)	96.498	0.1945
Manifold Mixup ($\alpha = 2.0$)	**97.104**	**0.1407**
PreActResNet152		
No Mixup	95.797	0.1994
Input Mixup ($\alpha = 1.0$)	96.844	0.2312
Manifold Mixup ($\alpha = 2.0$)	97.238	0.1419
Manifold Mixup all layers ($\alpha = 6.0$)	**97.622**	**0.0957**

Table 2: Supervised Classification Results on CIFAR-100. We note significant improvement with *Manifold Mixup* for both larger and smaller networks. Please refer to Appendix A for details on the implementation of *Manifold Mixup* and *Manifold Mixup* all layers.

Model	Test Acc	Test NLL
PreActResNet18		
No Mixup (Zhang et al., 2017)	74.4	n/a
No Mixup (ours)	75.32	1.284
Input Mixup ($\alpha = 1.0$) (Zhang et al., 2017)	78.9	n/a
Manifold Mixup ($\alpha = 2.0$)	**78.95**	**0.913**
PreActResNet34		
Input Mixup ($\alpha = 1.0$)	77.208	1.085
Manifold Mixup ($\alpha = 2.0$)	**79.609**	**0.930**

Table 3: Results on models trained on the normal CIFAR-100 and evaluated on a test set with novel deformations. The full version of this table and a more detailed description are in appendix table 7. *Manifold Mixup* (ours) consistently allows the model to be more robust to random shearing, rescaling, and rotation even though these deformations were not observed during training.

Test Set Deformation	No Mixup Baseline	Input Mixup α=1.0	Input Mixup α=2.0	*Manifold Mixup* α=2.0
Rotation U($-20°$,$20°$)	52.96	55.55	56.48	**60.08**
Rotation U($-60°$,$60°$)	26.77	28.47	27.53	**33.78**
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 (80% rescale)	47.95	52.18	50.47	**52.7**
Zoom Out (140% rescale)	19.34	41.81	42.02	**45.29**
Zoom Out (160% rescale)	11.12	25.48	25.85	**27.02**

4.2 Semi-Supervised Learning

Semi-supervised learning is concerned with building models which can take advantage of both labeled and unlabeled data. It is particularly useful in domains where obtaining labels is challenging, but unlabeled data is plentiful.

The *Manifold Mixup* approach to semi-supervised learning is closely related to the consistency regularization approach reviewed by Oliver et al. (2018). It involves minimizing loss on labelled samples as well as unlabeled samples by controlling the trade-off between these two losses via a consistency coefficient. In the *Manifold Mixup* approach for semi-supervised learning, the loss from labeled examples is computed as normal. For computing loss from unlabelled samples, the model's predictions are evaluated on a random batch of unlabeled data points. Then the normal manifold mixup procedure is used, but the targets to be mixed are the soft target outputs from the classifier. The detailed algorithm for both *Manifold Mixup* and Input Mixup with semi-supervised learning are given in appendix B.

(Oliver et al., 2018) performed a systematic study of semi-supervised algorithms using a fixed wide resnet architecture "WRN-28-2" (Zagoruyko & Komodakis, 2016). We evaluate *Manifold Mixup* using this same setup and achieve improvements for CIFAR-10 over the previously best performing algorithm, Virtual Adversarial Training (VAT) (Miyato et al., 2017) and Mean-Teachers (Tarvainen & Valpola, 2017). For SVHN, *Manifold Mixup* is competitive with VAT and Mean-Teachers. See Table 4. While VAT requires an additional calculation of the gradient and Mean-Teachers requires repeated model parameters averaging, *Manifold Mixup* requires no additional (non-trivial) computation.

Table 4: Results on semi-supervised learning on CIFAR-10 (4k labels) and SVHN (1k labels) (in test error %). All results use the same standardized architecture (WideResNet-28-2). Each experiment was run for 5 trials. † refers to the results reported in (Oliver et al., 2018)

SSL Approach	CIFAR-10	SVHN
Supervised †	20.26 ± 0.38	12.83 ± 0.47
Mean-Teacher †	15.87 ± 0.28	5.65 ± 0.47
VAT †	13.86 ± 0.27	5.63 ± 0.20
VAT-EM †	13.13 ± 0.39	$\mathbf{5.35 \pm 0.19}$
Semi-supervised Input Mixup	10.71 ± 0.44	6.54 ± 0.62
Semi-supervised *Manifold Mixup*	$\mathbf{10.26 \pm 0.32}$	5.70 ± 0.48

In addition, we also explore the regularization ability of *Manifold Mixup* in a fully-supervised low-data regime by training a PreResnet-152 model on 4000 labeled images from CIFAR-10. We obtained 13.64 % test error which is comparable with the fully-supervised regularized baseline according to results reported in (Oliver et al., 2018). Interestingly, we do not use a combination of two powerful regularizers ("Shake-Shake" and "Cut-out") and the more complex ResNext architecture as in (Oliver et al., 2018) and still achieve the same level of test accuracy, while doing much better than the fully supervised baseline not regularized with fancy regularizers (20.26% error).

4.3 Adversarial Examples

Adversarial examples in some sense are the "worst case" scenario for models failing to perform well when evaluated with data off of the manifold.[2]. Because *Manifold Mixup* only considers a subset of directions around data points (namely, those corresponding to interpolations), we would not expect the model to be robust to adversarial attacks which can consider any direction within an epsilon-ball of each example. At the same time, *Manifold Mixup* expands the set of points seen during training, so an intriguing hypothesis is that these overlap somewhat with the set of possible adversarial examples, which would force adversarial attacks to consider a wider set of directions, and potentially be more computationally expensive. To explore this we considered the Fast Gradient Sign Method (FGSM, Goodfellow et al., 2014) which only requires a single gradient update and considers a relatively small subset of adversarial directions. The results performance of *Manifold Mixup* against FGSM are given in Table 5. A challenge in evaluating adversarial examples comes from the gradient masking problem in which a defense succeeds solely due to 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 confirm the quality of the gradient signal. Our *Manifold Mixup* passed this sanity check, see appendix D. While we found that *Manifold Mixup* greatly improved robustness to the FGSM attack, especially over Input Mixup (Zhang et al., 2017), we found that *Manifold Mixup* did not significantly improve robustness against the stronger iterative projected gradient descent (PGD) attack (Madry et al., 2017).

4.4 Generative Adversarial Networks

The recent literature has suggested that regularizing the discriminator is beneficial to training (Salimans et al., 2016; Arjovsky et al., 2017; Gulrajani et al., 2017; Miyato et al., 2018). In a similar vein, one could add mixup to the original GAN training objective such that the extra data augmentation acts as a beneficial regularization to the discriminator, which is what was proposed in Zhang et al. (2017). Mixup proposes the following objective:

$$\max_{g} \min_{d} \mathbb{E}_{x,z,\lambda}\, \ell(d(\lambda x + (1-\lambda) g(z)), \lambda), \tag{1}$$

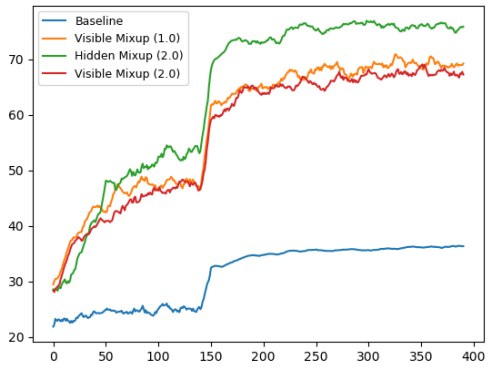
where $\lambda \sim Beta(\alpha, \alpha)$[3]. Empirically, we found that this formulation did not perform well (yielding non-crisp looking samples), and we instead propose a modification with several changes: 1) instead

[2]See the adversarial spheres (Gilmer et al., 2018) paper for a discussion of what it means to be off of the manifold.

[3]A subtle detail to add is that we clamp values of λ such that if $\lambda_0 \sim Beta(\alpha, \alpha)$ then $\lambda = \min(\lambda_0, 1-\lambda_0)$. This is done in order to avoid mixes that are close to being very real, e.g. 99% real + 1% fake.

Table 5: CIFAR-10 Test Accuracy Results on white-box adversarial attacks (higher is better) using PreActResNet18. Note that our method is close to adversarial training in performance despite not requiring any additional (significant) computation. CIFAR-10 FGSM attacks on test set, wrt training epochs (right). JB refers to (Jacob Buckman, 2018), Madry refers to (Madry et al., 2017).

CIFAR10 Models	FGSM ε=0.03
Adv. Training (Madry)	60.30
Adversarial Training + Fortified Networks	81.80
Baseline (ours)	36.32
Input Mixup ($\alpha = 1.0$)	71.51
Manifold Mixup ($\alpha = 2.0$)	77.50
CIFAR100 Models	**FGSM ε=0.03**
Input Mixup ($\alpha = 1.0$)	40.7
Manifold Mixup ($\alpha = 2.0$)	44.96



of the discriminator classifying mixes with label λ, we denote all mixes to be fake; 2) we keep the original objectives for the discriminator (to classify between real and fake); and 3) the generator does not try to fool the discriminator with mixes, and so its objective remains the same as that of a regular GAN. Therefore, our new discriminator objective is (with k denoting the layer to mix up in):

$$\min_d \mathbb{E}_{x,z}\, \ell(d_k(x), 1) + \ell(d_k(g(z), 0) + \ell(d_k(\lambda h_k(x) + (1-\lambda)h_k(g(z)), 0), \tag{2}$$

Note that one could also experiment with mixes between only real and fake examples, which would add more mixing to the training objective at the cost of computation time. We experiment with mixes between fake examples, where we add the following term to Equation 2:

$$\mathbb{E}_{z1,z2}\, \ell(d_k(\lambda h_k(g(z_1)) + (1-\lambda)h_k(g(z_2))), 0). \tag{3}$$

Analogous to the earlier parts of the paper, *Manifold Mixup* in the GAN case would simply be to perform the mixup in the hidden space of the discriminator. Like the other domains on which *Manifold Mixup* was tested, this means (with equal probability) choosing between mixing in visible space, the output of the first resblock or the second resblock of the discriminator. However, we found that this did not produce competitive results to when only Input Mixup was performed. In fact, we found that mixing in the first layer of the discriminator worked reasonably well.

We conjecture that the reason for why GAN mixup does not perform well in deep hidden spaces (in our case, the resblocks of the discriminator) is because we are more likely to create plausible mixes the deeper we go into the network (see Figures 3 and 4), and therefore, assigning hard labels to all mixes may be to the detriment of the discriminator, whereas in contrast it has the positive effect of tightening the discriminator's decision boundary between real and fake examples when done in visible space. To corroborate this, we ran a range of experiments exploring hidden mixup but using soft labels instead, and while it did not produce competitive results, it gave a slight boost in Inception scores. This suggests that that marking interpolated points in hidden space as completely fake is not an appropriate assumption to make. Lastly, our competitive manifold mixup result (in Table 6) was obtained by mixing on the output of the first convolutional layer of the discriminator, rather than the outputs of its resblocks. It is possible that mixing early enough in the hidden space of the network achieves the right balance between tightening the decision boundary of the discriminator versus producing plausible-looking mixes.

5 Analysis

Synthetic Data Task So far we have generated significant empirical improvements from manifold mixup, however these experiments only provide a limited degree of insight because they use natural

Table 6: Mixup results on GANs on CIFAR-10. For GAN-GP, γ denotes the gradient penalty coefficient term; α denotes the parameters of the Beta distribution; FR denotes fake-real mixes (equation 2) and FF fake-fake mixes (equation 3); and for spectral norm (SN) n_{dis} is the number of discriminator updates per generator update (we found that $n_{dis} = 5$ was necessary for spectral norm). All results were run thrice with Inception mean / standard deviation and FID score averaged across those runs. † refers to the results reported in Miyato et al. (2018), and †† Gulrajani et al. (2017).

Experiment	Inception	FID
GAN-GP ($\gamma = 10$)	7.30 ± 0.09	30.2
+ mixup FR ($\alpha = 0.2$)	7.34 ± 0.09	28.0
+ mixup FR+FF ($\alpha = 0.2$)	7.55 ± 0.08	25.6
SN-GAN ($n_{dis} = 5$)	7.94 ± 0.08	21.7
+ mixup FR ($\alpha = 0.2$)	8.00 ± 0.08	21.5
+ mixup FR+FF ($\alpha = 0.2$)	$\mathbf{8.13 \pm 0.10}$	**20.6**
SN-GAN + hinge ($n_{dis} = 5$)	7.97 ± 0.10	22.2
+ mixup FR ($\alpha = 0.2$)	7.92 ± 0.09	22.6
GAN + *Manifold Mixup* FR ($\gamma = 0.5, \alpha = 0.2$)	8.05 ± 0.06	20.7
WGAN-GP ††	7.86 ± 0.08	—
SN-GAN + hinge †	$\mathbf{8.22 \pm 0.05}$	**21.7**

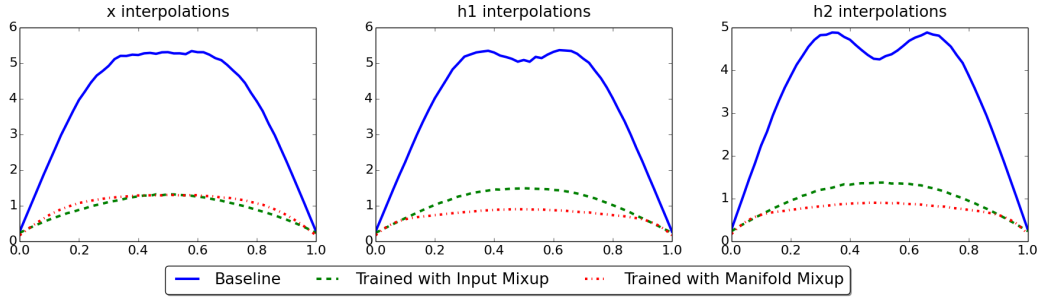


Figure 1: Study of test NLL using the interpolated target values (lower is better) on interpolated points under models trained with the baseline, mixup, and *Manifold Mixup*. *Manifold Mixup* dramatically improves performance when interpolating in the hidden states, and very slightly reduces performance when interpolating in the visible space.

data, where it is difficult to understand exactly what caused the improvement. To give better clarity into why our method helps compared to ordinary training and compared to Input Mixup, we generated a synthetic dataset (Figure 2) where the examples are generated based on a set of independent latent factors. Thus we can simulate the ideal case where mixup is performed in the true latent space. We deliberately construct a test set where the generative factors' marginal distribution differs from that during training. We found that mixing in the ground-truth attribute space dramatically improved results both in terms of accuracy and likelihood. Input Mixup improved likelihood modestly but did not improve accuracy. This is direct evidence that mixup provides the most significant improvements when it produces points which correspond to important factors of variation in the data, and that producing "nonsense" points which are very far from the manifold yields only a small improvement. More details and a full table of results on different synthetic tasks is in the appendix E.

Visualizing the effect of mixing in the hidden states An important question is what kinds of feature combinations are being explored when we perform mixup in the hidden layers as opposed to linear interpolation in visible space. To provide a qualitative study of this, we trained a small decoder

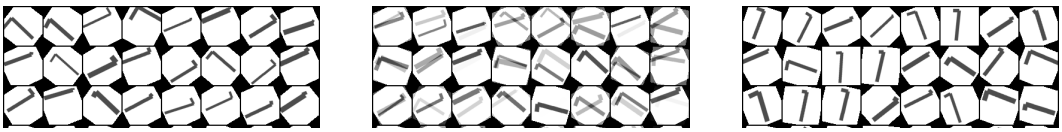


Figure 2: Synthetic task where the underlying factors are known exactly. Training images (left), images from Input Mixup (center), and images from mixing in the ground truth factor space (right).

convnet (with upsampling layers) to predict an image from the classifier's hidden representation (using a simple squared error loss in the visible space). We then performed mixup on the hidden states between two random examples, and ran this interpolated hidden state through the convnet to get an estimate of what the point would look like in x-space. Similarly to the earlier results on auto-encoders (Bengio et al., 2013), we found that these interpolated h points corresponded to images with a blend of the features from the two images, as opposed to the less-semantic pixel-wise blending resulting from Input Mixup (Figure 3). Furthermore, this justifies the training objective for examples mixed-up in the hidden layers: (1) most of the interpolated points correspond to plausible natural images, thus leading to more appropriate augmentation; and (2) none of the interpolated points between objects of two different categories A and B correspond to a third category C, thus justifying a training target which gives 0 probability on all the classes except A and B. Observation 2 can be explained by remembering that the latent space is still very high-dimensional, so that most of the probability mass is on the outside (think of the surface of a hyper-sphere) with interpolated rays between A and B going through the middle of the sphere (where no other class examples are present). Observation (1) is consistent with a flattening of the manifolds of all categories along the same directions (otherwise the interpolated points would be much more outside of the manifold of natural images).

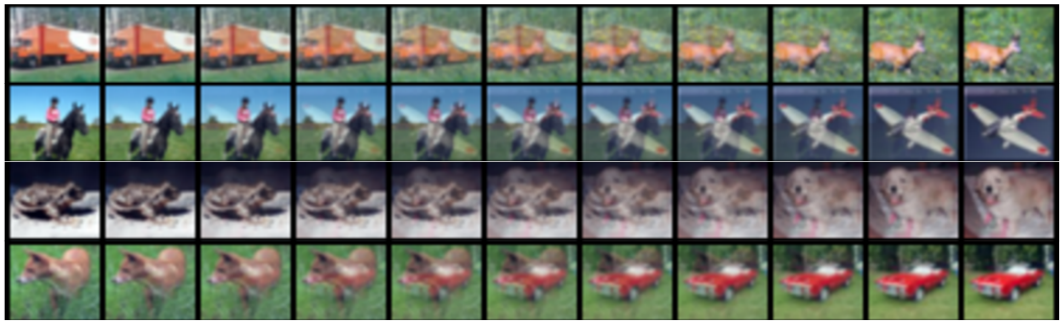


Figure 3: **Interpolations in the input space** with a mixing rate varied from 0.0 to 1.0.

Figure 4: **Interpolations in the hidden states** (using a small convolutional network trained to predict the input from the output of the second resblock). The interpolations in the hidden states show a better blending of semantically relevant features, and more of the images are visually consistent.

Analysis of how *Manifold Mixup* changes learned representations An intriguing question is how applying *Manifold Mixup* changes the representations learned by a layer. To qualitatively study this we trained on MNIST digits 0-4 using a network with two initial leaky ReLU layers, a 2D bottleneck hidden representation, followed by two additional leaky ReLU layers. We considered

training without mixup, with just Input Mixup, and with mixup only directly following that 2D bottleneck state. This is shown in Fig. 5.

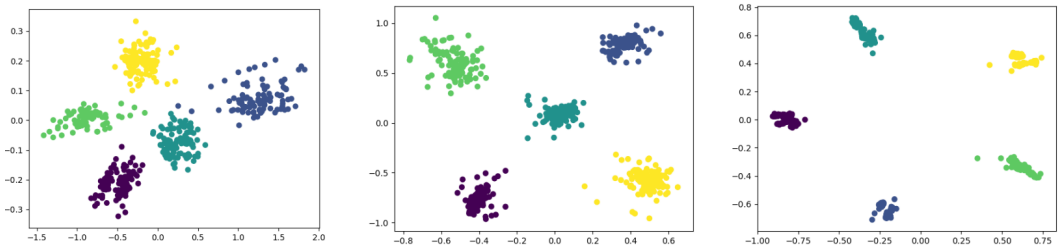


Figure 5: Representations in a 2D bottleneck layer from an MNIST classifier (digits 0-4, a version with all digits is in appendix Figure 8). No Mixup Baseline (left), Input Mixup (center). *Manifold Mixup* (right) learns more cleanly separated representations with a wider margin between classes.

6 Future Work

While it is not essential to our discussion of *Manifold Mixup*, there is an intriguing connection between this idea and a challenging problem in neuroscience. At a high level, we can imagine systems in the brain which compute predictions from a stream of changing inputs, and pass these predictions onto other modules which return some kind of feedback signal (Lee et al., 2015; Scellier & Bengio, 2017; Whittington & Bogacz, 2017; Bartunov et al., 2018) (potentially gradients or targets for the prediction). There is a delay between the output of the prediction and the point in time in which the feedback can return to the system after having travelled across the brain. Moreover, this delay could be noisy and could differ based on the type of the prediction or other conditions in the brain, as well as depending on which paths are considered (there are many skip connections between areas). This means that it could be very difficult for a system in the brain to establish a clear 1:1 correspondence between its outputs and the feedback signals that it receives over time.

While it is very preliminary, an intriguing hypothesis is that systems in the brain could be working around this limitation by averaging their states and feedback signals across multiple points in time. The empirical results from mixup suggest that such a technique may not just allow successful computation, but also act as a potent regularizer. *Manifold Mixup* buttresses this result by showing that the same regularization effect can be achieved from mixing in higher level hidden states.

7 Conclusion

We have presented *Manifold Mixup*, a new algorithm for improving the generalization of deep networks and their ability to handle data off of the manifold. This is demonstrated in improved test accuracy and dramatically improved test likelihood on classification, better robustness to adversarial examples, and improved semi-supervised learning. We also demonstrated intriguing properties of *Manifold Mixup* on generative adversarial networks. *Manifold Mixup* incurs virtually no additional computational cost and works across a variety of hyperparameters, making it appealing for practitioners. We conducted analysis that gives geometric insight into why *Manifold Mixup* works (basically by exploiting the ability of deep nets to flatten data manifolds) and constructed synthetic tasks where mixing along the ground truth factors provides much greater improvements than mixing in the input space, which supports our reasoning for why *Manifold Mixup* improves results on real data.

Acknowledgements

The authors thank Christopher Pal, Sherjil Ozair and Dzmitry Bahdanau for useful discussions and feedback. 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.

References

Alemi, Alexander A., Fischer, Ian, Dillon, Joshua V., and Murphy, Kevin. Deep variational information bottleneck. *CoRR*, abs/1612.00410, 2016. URL http://arxiv.org/abs/1612.00410.

Arjovsky, Martin, Chintala, Soumith, and Bottou, Léon. 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. *ArXiv e-prints*, February 2018.

Bartunov, Sergey, Santoro, Adam, Richards, Blake A., Hinton, Geoffrey E., and Lillicrap, Timothy. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *submitted to ICLR'2018*, 2018.

Bengio, Yoshua, Mesnil, Grégoire, Dauphin, Yann, and Rifai, Salah. Better mixing via deep representations. *CoRR*, abs/1207.4404, 2012. URL http://arxiv.org/abs/1207.4404.

Bengio, Yoshua, Mesnil, Grégoire, Dauphin, Yann, and Rifai, Salah. Better mixing via deep representations. In *ICML'2013*, 2013.

Gilmer, J., Metz, L., Faghri, F., Schoenholz, S. S., Raghu, M., Wattenberg, M., and Goodfellow, I. Adversarial Spheres. *ArXiv e-prints*, January 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and Harnessing Adversarial Examples. *ArXiv e-prints*, December 2014.

Gulrajani, Ishaan, Ahmed, Faruk, Arjovsky, Martin, Dumoulin, Vincent, and Courville, Aaron C. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pp. 5769–5779, 2017.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016. URL http://arxiv.org/abs/1603.05027.

Hinton, Geoffrey E., Srivastava, Nitish, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. URL http://arxiv.org/abs/1207.0580.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

Jacob Buckman, Aurko Roy, Colin Raffel Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S18Su--CW.

Lee, Dong-Hyun, Zhang, Saizheng, Fischer, Asja, and Bengio, Yoshua. Difference target propagation. In *Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*. 2015.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards Deep Learning Models Resistant to Adversarial Attacks. *ArXiv e-prints*, June 2017.

Mikolov, Tomas, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. URL http://arxiv.org/abs/1301.3781.

Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. Virtual Adversarial Training: a Regularization Method for Supervised and Semi-supervised Learning. *ArXiv e-prints*, April 2017.

Miyato, Takeru, Kataoka, Toshiki, Koyama, Masanori, and Yoshida, Yuichi. Spectral normalization for generative adversarial networks. *CoRR*, abs/1802.05957, 2018. URL http://arxiv.org/abs/1802.05957.

Oliver, A., Odena, A., Raffel, C., Cubuk, E. D., and Goodfellow, I. J. Realistic Evaluation of Deep Semi-Supervised Learning Algorithms. *ArXiv e-prints*, April 2018.

Pereyra, Gabriel, Tucker, George, Chorowski, Jan, Kaiser, Lukasz, and Hinton, Geoffrey E. Regularizing neural networks by penalizing confident output distributions. *CoRR*, abs/1701.06548, 2017. URL http://arxiv.org/abs/1701.06548.

Salimans, Tim, Goodfellow, Ian, Zaremba, Wojciech, Cheung, Vicki, Radford, Alec, and Chen, Xi. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

Scellier, Benjamin and Bengio, Yoshua. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11, 2017.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *ArXiv e-prints*, December 2013.

Szegedy, Christian, Vanhoucke, Vincent, Ioffe, Sergey, Shlens, Jonathon, and Wojna, Zbigniew. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. URL `http://arxiv.org/abs/1512.00567`.

Tarvainen, Antti and Valpola, Harri. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 1195–1204. Curran Associates, Inc., 2017.

Tokozume, Yuji, Ushiku, Yoshitaka, and Harada, Tatsuya. Between-class learning for image classification. *CoRR*, abs/1711.10284, 2017. URL `http://arxiv.org/abs/1711.10284`.

Whittington, James CR and Bogacz, Rafal. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 2017.

Zagoruyko, Sergey and Komodakis, Nikos. Wide residual networks. *CoRR*, abs/1605.07146, 2016. URL `http://arxiv.org/abs/1605.07146`.

Zhang, Hongyi, Cissé, Moustapha, Dauphin, Yann N., and Lopez-Paz, David. mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017. URL `http://arxiv.org/abs/1710.09412`.

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

A Supervised Regularization

Table 7: Full version of Table 3 on models trained on the normal CIFAR-100 and evaluated on a test set with novel deformations. *Manifold Mixup* (ours) consistently allows the model to be more robust to random shearing, rescaling, and rotation even though these deformations were not observed during training. For the rotation experiment, each image is rotated with an angle uniformly sampled from the given range. Likewise the shearing is performed with uniformly sampled angles. Zooming-in refers to take a bounding box at the center of the image with k% of the length and k% of the width of the original image, and then expanding this image to fit the original size. Likewise zooming-out refers to drawing a bounding box with k% of the height and k% of the width, and then taking this larger area and scaling it down to the original size of the image (the padding outside of the image is black).

Test Set Deformation	No Mixup Baseline	Input Mixup α=1.0	Input Mixup α=2.0	*Manifold Mixup* α=2.0
Rotation U($-20°$,$20°$)	52.96	55.55	56.48	**60.08**
Rotation U($-40°$,$40°$)	33.82	37.73	36.78	**42.13**
Rotation U($-60°$,$60°$)	26.77	28.47	27.53	**33.78**
Rotation U($-80°$,$80°$)	24.19	26.72	25.34	**29.95**
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**
Shearing U($-114.6°$, $114.6°$)	19.57	22.94	22.8	**24.69**
Shearing U($-143.2°$, $143.2°$)	17.55	21.66	21.22	**23.56**
Shearing U($-171.9°$, $171.9°$)	22.38	25.53	25.27	**28.02**
Zoom In (20% rescale)	2.43	1.9	**2.45**	2.03
Zoom In (40% rescale)	4.97	4.47	**5.23**	4.17
Zoom In (60% rescale)	12.68	**13.75**	13.12	11.49
Zoom In (80% rescale)	47.95	52.18	50.47	**52.7**
Zoom Out (120% rescale)	43.18	60.02	61.62	**63.59**
Zoom Out (140% rescale)	19.34	41.81	42.02	**45.29**
Zoom Out (160% rescale)	11.12	25.48	25.85	**27.02**
Zoom Out (180% rescale)	7.98	**18.11**	18.02	15.68

For supervised regularization we considered architectures within the PreActResNet family: PreActResNet18, PreActResNet34, and PreActResNet152. When using *Manifold Mixup*, we selected the layer to perform mixing uniformly at random from a set of eligible layers. In our experiments on PreActResNets in Table 1, Table 2 Table 3, Table 5 and Table 7, for *Manifold Mixup*, our eligible layers for mixing were : the input layer, the output from the first resblock, and the output from the second resblock. For PreActResNet18, the first resblock has four layers and the second resblock has four layers. For PreActResNet34, the first resblock has six layers and the second resblock has eight layers. For PreActResNet152, the first resblock has 9 layers and the second resblock has 24 layers. Thus the mixing is often done fairly deep in the network, for example in PreActResNet152 the output of the second resblock is preceded by a total of 34 layers (including the initial convolution which is not in a resblock). For *Manifold Mixup* All layers in Table 1, our eligible layers for mixing were : the input layer, the output from the first resblock, and the output from the second resblock, and the output from the third resblock. We trained all models for 1200 epochs and dropped the learning rates by a factor of 0.1 at 400 epochs and 800 epochs.

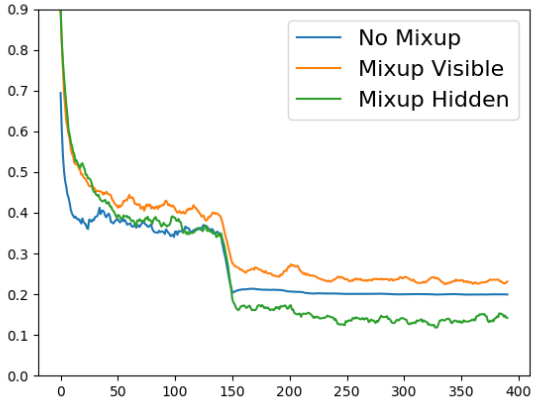


Figure 6: CIFAR-10 test set Likelihood on PreActResNet152, wrt training epochs.

B Semi-supervised Manifold Mixup and Input Mixup Algorithm

We present the procedure for Semi-supervised Manifold Mixup and Semi-supervised Input Mixup in Algorithms 1 and 3 respectively.

Algorithm 1 Semi-supervised Manifold Mixup. f_θ: Neural Network; $ManifoldMixup$: Manifold Mixup Algorithm 2; D_L: set of labelled samples; D_{UL}: set of unlabelled samples; π : consistency coefficient (weight of unlabeled loss, which is ramped up to increase from zero to its max value over the course of training); N: number of updates; $\tilde{y_i}$: Mixedup labels of labelled samples; $\hat{y_i}$: predicted label of the labelled samples mixed at a hidden layer; y_j: Psuedolabels for unlabelled samples; $\tilde{y_j}$: Mixedup Psuedolabels of unlabelled samples; $\hat{y_j}$ predicted label of the unlabelled samples mixed at a hidden layer

$k \leftarrow 0$
while $k \leq N$ **do**
  Sample $(x_i, y_i) \sim D_L$ ▷ Sample labeled batch
  $\hat{y}_i, \tilde{y}_i = ManifoldMixup(x_i, y_i, \theta)$
  $L_S = Loss(\hat{y}_i, \tilde{y}_i)$ ▷ Cross Entropy loss
  Sample $x_j \sim D_{UL}$ ▷ Sample unlabeled batch
  $y_j = f_\theta(x_j)$ ▷ Compute Pseudolabels
  $\hat{y}_j, \tilde{y}_j = ManifoldMixup(x_j, y_j, \theta)$
  $L_{US} = Loss(\hat{y}_j, \tilde{y}_j)$ ▷ MSE Loss
  L = $L_S + \pi(k) L_{US}$ ▷ Total Loss
  $g \leftarrow \nabla_\theta L$ (Gradients of the minibatch Loss)
  $\theta \leftarrow$ Update parameters using gradients g (e.g. SGD)
end while

Algorithm 2 Manifold Mixup. f_θ: Neural Network; D : dataset

Sample $(x_i, y_i) \sim D$ ▷ Sample a batch
$h_i \leftarrow$hidden state representation of Neural Network f_θ at a layer k randomly ▷ the layer k is chosen
$(h_i^{mixed}, y_i^{mixed}) \leftarrow Mixup(h_i, y_i)$
$\hat{y_i} \leftarrow$ Forward Pass the h_i^{mixed} from layer k to the output layer of f_θ
return $\hat{y_i}, \tilde{y_i}$

Algorithm 3 Semi-supervised Input Mixup. f_θ: Neural Network. $InputMixup$: Mixup process of (Zhang et al., 2017); D_L: set of labelled samples; D_{UL}: set of unlabelled samples; π : consistency coefficient (weight of unlabeled loss, which is ramped up to increase from zero to its max value over the course of training); N: number of updates; $x_i^{mixedup}$: mixed up sample; $y_i^{mixedup}$: mixed up label; $\hat{y}_i^{mixedup}$: mixed up predicted label

$k \leftarrow 0$
while $k \leq N$ **do**
    Sample $(x_i, y_i) \sim D_L$
    ▷ Sample labeled batch
    $(x_i^{mixedup}, y_i^{mixedup}) = InputMixup(x_i, y_i)$
    $L_S = Loss(f_\theta({x_i}^{mixedup}), {y_i}^{mixedup})$ ▷ CrossEntropy Loss
    Sample $x_j \sim D_{UL}$ ▷ Sample unlabeled batch
    $\hat{y_j} = f_\theta(x_j)$ ▷ Compute Pseudolabels
    $({x_j}^{mixedup}, \hat{y_j}^{mixedup}) = InputMixup(x_j, \hat{y_j})$
    $L_{US} = Loss(f_\theta({x_j}^{mixedup}), \hat{y_i}^{mixedup})$ ▷ MSE Loss
    L = $L_S + \pi(k) * L_{US}$ ▷ Total Loss
    $g \leftarrow \nabla_\theta L$ ▷ Gradients of the minibatch Loss
    $\theta \leftarrow$ Update parameters using gradients g (e.g. SGD)
end while

C Semi-supervised Experimental details

We use the WideResNet28-2 architecture used in (Oliver et al., 2018) and closely follow their experimental setup for fair comparison with other Semi-supervised learning algorithms. We used SGD with momentum optimizer in our experiments. For Cifar10, we run the experiments for 1000 epochs with initial learning rate is 0.1 and it is annealed by a factor of 0.1 at epoch 500, 750 and 875. For SVHN, we run the experiments for 200 epochs with initial learning rate is 0.1 and it is annealed by a factor of 0.1 at epoch 100, 150 and 175. The momentum parameter was set to 0.9. We used L2 regularization coefficient 0.0005 and L1 regularization coefficient 0.001 in our experiments. We use the batch-size of 100.

The data pre-processing and augmentation in exactly the same as in (Oliver et al., 2018). For CIFAR-10, we use the standard train/validation split of 45,000 and 5000 images for training and validation respectively. We use 4000 images out of 45,000 train images as labelled images for semi-supervised learning. For SVHN, we use the standard train/validation split with 65932 and 7325 images for training and validation respectively. We use 1000 images out of 65932 images as labelled images for semi-supervised learning. We report the test accuracy of the model selected based on best validation accuracy.

For supervised loss, we used α (of $\lambda \sim Beta(\alpha, \alpha)$) from the set { 0.1, 0.2, 0.3... 1.0} and found 0.1 to be the best. For unsupervised loss, we used α from the set {0.1, 0.5, 1.0, 1.5, 2.0. 3.0, 4.0} and found 2.0 to be the best.

The consistency coefficient is ramped up from its initial value 0.0 to its maximum value at 0.4 factor of total number of iterations using the same sigmoid schedule of (Tarvainen & Valpola, 2017). For CIFAR-10, we found max consistency coefficient = 1.0 to be the best. For SVHN, we found max consistency coefficient = 2.0 to be the best.

When using *Manifold Mixup*, we selected the layer to perform mixing uniformly at random from a set of eligible layers. In our experiments on WideResNet28-2 in Table 4, our eligible layers for mixing were : the input layer, the output from the first resblock, and the output from the second resblock.

D Adversarial Examples

We ran the unbounded projected gradient descent (PGD) (Madry et al., 2017) sanity check suggested in (Athalye et al., 2018). We took our trained models for the input mixup baseline and manifold mixup and we ran PGD for 200 iterations with a step size of 0.01 which reduced the mixup model's accuracy to 1% and reduced the *Manifold Mixup* model's accuracy to 0%. This is direct evidence that our defense did not improve results primarily as a result of gradient masking.

The Fast Gradient Sign Method (FGSM) Goodfellow et al. (2014) is a simple one-step attack that produces $\widetilde{x} = x + \varepsilon \operatorname{sgn}(\nabla_x L(\theta, x, y))$.

E Synthetic Experiments Analysis

We conducted experiments using a generated synthetic dataset where each image is deterministically rendered from a set of independent factors. The goal of this experiment is to study the impact of input mixup and an idealized version of *Manifold Mixup* where we know the true factors of variation in the data and we can do mixup in exactly the space of those factors. This is not meant to be a fair evaluation or representation of how *Manifold Mixup* actually performs - rather it's meant to illustrate how generating relevant and semantically meaningful augmented data points can be much better than generating points which are far off the data manifold.

We considered three tasks. In Task A, we train on images with angles uniformly sampled between (-70°, -50°) (label 0) with 50% probability and uniformly between (50°, 80°) (label 1) with 50% probability. At test time we sampled uniformly between (-30°, -10°) (label 0) with 50% probability and uniformly between (10°, 30°) (label 1) with 50% probability. Task B used the same setup as Task A for training, but the test instead used (-30°, -20°) as label 0 and (-10°, 30°) as label 1. In Task C we made the label whether the digit was a "1" or a "7", and our training images were uniformly sampled between (-70°, -50°) with 50% probability and uniformly between (50°, 80°) with 50% probability. The test data for Task C were uniformly sampled with angles from (-30°, 30°).

The examples of the data are in figure 7 and results are in table 8. In all cases we found that Input Mixup gave some improvements in likelihood but limited improvements in accuracy - suggesting that the even generating nonsensical points can help a classifier trained with Input Mixup to be better calibrated. Nonetheless the improvements were much smaller than those achieved with mixing in the ground truth attribute space.

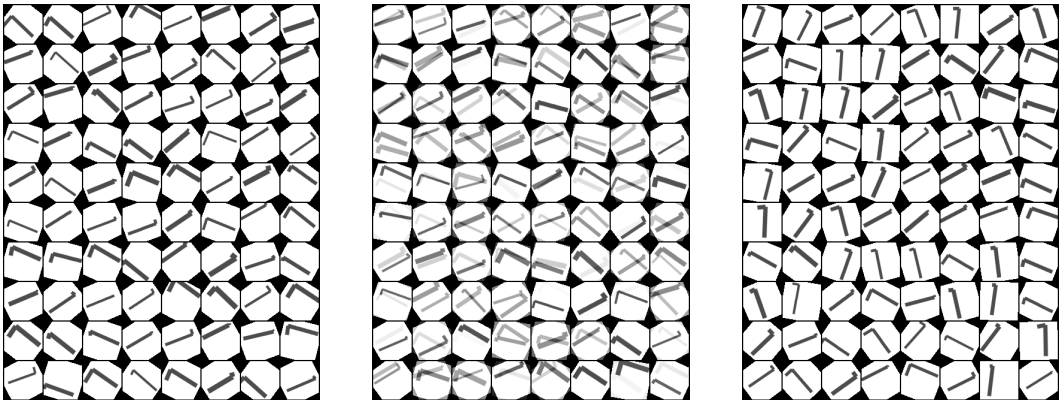


Figure 7: Synthetic task where the underlying factors are known exactly. Training images (left), images from input mixup (center), and images from mixing in the ground truth factor space (right).

Table 8: Results on synthetic data generalization task with an idealized Manifold Mixup (mixing in the true latent generative factors space). Note that in all cases visible mixup significantly improved likelihood, but not to the same degree as factor mixup.

Task	Model	Test Accuracy	Test NLL
Task A	No Mixup	1.6	8.8310
	Input Mixup (1.0)	0.0	6.0601
	Ground Truth Factor Mixup (1.0)	94.77	0.4940
Task B	No Mixup	21.25	7.0026
	Input Mixup (1.0)	18.40	4.3149
	Ground Truth Factor Mixup (1.0)	84.02	0.4572
Task C	No Mixup	63.05	4.2871
	Input Mixup	66.09	1.4181
	Ground Truth Factor Mixup	99.06	0.1279

F Analysis of how *Manifold Mixup* changes learned representations

We have found significant improvements from using *Manifold Mixup*, but a key question is whether the improvements come from changing the behavior of the layers before the mixup operation is applied or the layers after the mixup operation is applied. This is a place where *Manifold Mixup* and Input Mixup are clearly differentiated, as Input Mixup has no "layers before the mixup operation" to change. We conducted analytical experimented where the representations are low-dimensional enough to visualize. More concretely, we trained a fully connected network on MNIST with two fully-connected leaky relu layers of 1024 units, followed by a 2-dimensional bottleneck layer, followed by two more fully-connected leaky-relu layers with 1024 units.

We then considered training with no mixup, training with mixup in the input space, and training *only* with mixup directly following the 2D bottleneck. We consistently found that *Manifold Mixup* has the effect of making the representations much tighter, with the real data occupying more specific points, and with a more well separated margin between the classes.

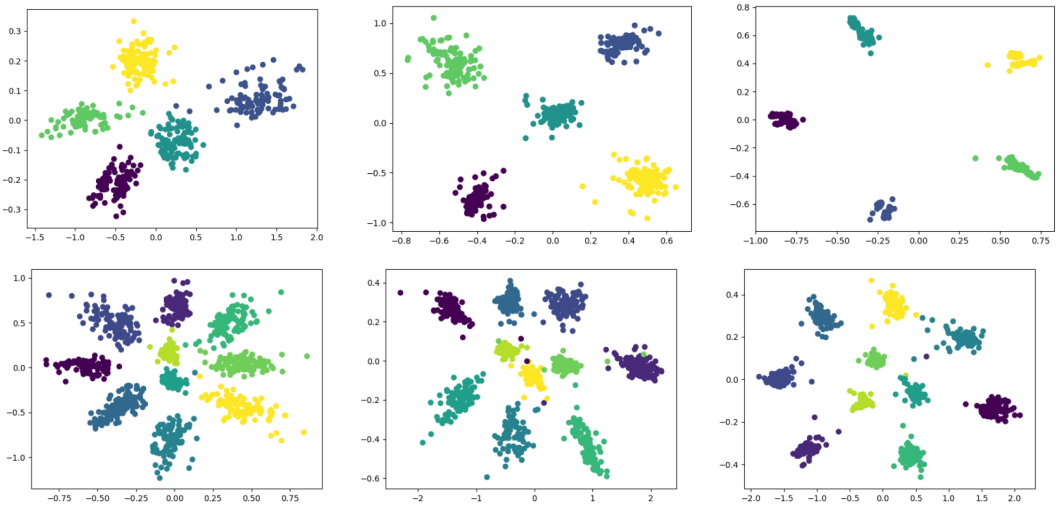


Figure 8: Representations from a classifier on MNIST (top is trained on digits 0-4, bottom is trained on all digits) with a 2D bottleneck representation in the middle layer. No Mixup Baseline (left), Input Mixup (center), *Manifold Mixup* (right).

References

Alemi, Alexander A., Fischer, Ian, Dillon, Joshua V., and Murphy, Kevin. Deep variational information bottleneck. *CoRR*, abs/1612.00410, 2016. URL http://arxiv.org/abs/1612.00410.

Arjovsky, Martin, Chintala, Soumith, and Bottou, Léon. 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. *ArXiv e-prints*, February 2018.

Bartunov, Sergey, Santoro, Adam, Richards, Blake A., Hinton, Geoffrey E., and Lillicrap, Timothy. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *submitted to ICLR'2018*, 2018.

Bengio, Yoshua, Mesnil, Grégoire, Dauphin, Yann, and Rifai, Salah. Better mixing via deep representations. *CoRR*, abs/1207.4404, 2012. URL http://arxiv.org/abs/1207.4404.

Bengio, Yoshua, Mesnil, Grégoire, Dauphin, Yann, and Rifai, Salah. Better mixing via deep representations. In *ICML'2013*, 2013.

Gilmer, J., Metz, L., Faghri, F., Schoenholz, S. S., Raghu, M., Wattenberg, M., and Goodfellow, I. Adversarial Spheres. *ArXiv e-prints*, January 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and Harnessing Adversarial Examples. *ArXiv e-prints*, December 2014.

Gulrajani, Ishaan, Ahmed, Faruk, Arjovsky, Martin, Dumoulin, Vincent, and Courville, Aaron C. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pp. 5769–5779, 2017.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016. URL http://arxiv.org/abs/1603.05027.

Hinton, Geoffrey E., Srivastava, Nitish, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. URL http://arxiv.org/abs/1207.0580.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

Jacob Buckman, Aurko Roy, Colin Raffel Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S18Su--CW.

Lee, Dong-Hyun, Zhang, Saizheng, Fischer, Asja, and Bengio, Yoshua. Difference target propagation. In *Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*. 2015.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards Deep Learning Models Resistant to Adversarial Attacks. *ArXiv e-prints*, June 2017.

Mikolov, Tomas, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. URL http://arxiv.org/abs/1301.3781.

Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. Virtual Adversarial Training: a Regularization Method for Supervised and Semi-supervised Learning. *ArXiv e-prints*, April 2017.

Miyato, Takeru, Kataoka, Toshiki, Koyama, Masanori, and Yoshida, Yuichi. Spectral normalization for generative adversarial networks. *CoRR*, abs/1802.05957, 2018. URL http://arxiv.org/abs/1802.05957.

Oliver, A., Odena, A., Raffel, C., Cubuk, E. D., and Goodfellow, I. J. Realistic Evaluation of Deep Semi-Supervised Learning Algorithms. *ArXiv e-prints*, April 2018.

Pereyra, Gabriel, Tucker, George, Chorowski, Jan, Kaiser, Lukasz, and Hinton, Geoffrey E. Regularizing neural networks by penalizing confident output distributions. *CoRR*, abs/1701.06548, 2017. URL http://arxiv.org/abs/1701.06548.

Salimans, Tim, Goodfellow, Ian, Zaremba, Wojciech, Cheung, Vicki, Radford, Alec, and Chen, Xi. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

Scellier, Benjamin and Bengio, Yoshua. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11, 2017.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *ArXiv e-prints*, December 2013.

Szegedy, Christian, Vanhoucke, Vincent, Ioffe, Sergey, Shlens, Jonathon, and Wojna, Zbigniew. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. URL `http://arxiv.org/abs/1512.00567`.

Tarvainen, Antti and Valpola, Harri. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 1195–1204. Curran Associates, Inc., 2017.

Tokozume, Yuji, Ushiku, Yoshitaka, and Harada, Tatsuya. Between-class learning for image classification. *CoRR*, abs/1711.10284, 2017. URL `http://arxiv.org/abs/1711.10284`.

Whittington, James CR and Bogacz, Rafal. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 2017.

Zagoruyko, Sergey and Komodakis, Nikos. Wide residual networks. *CoRR*, abs/1605.07146, 2016. URL `http://arxiv.org/abs/1605.07146`.

Zhang, Hongyi, Cissé, Moustapha, Dauphin, Yann N., and Lopez-Paz, David. mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017. URL `http://arxiv.org/abs/1710.09412`.

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