A. 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 by mixing in the input space.

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 4 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.

B. 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 smaller region in the hidden space, and with a more well separated margin between the classes, as shown in Figure 5

C. Supervised Regularization Experimental Details

For supervised regularization we considered following architectures: PreActResNet18, PreActResNet34, and Wide-Resnet-28-10. When using *Manifold Mixup*, we selected the layer to perform mixing uniformly at random from a set of eligible layers. In all our experiments, for the PreActResNets architectures, the eligible layers for mixing in *Manifold Mixup* were : the input layer, the output from the first resblock, and the output from the second resblock. For Wide-ResNet-20-10 architecture, the eligible layers for mixing in *Manifold Mixup* were: the input layer and the output from the first 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 Wide-Resnet-28-10, the first resblock has four layers. Thus the mixing is often done fairly deep layers in the network.

Throughout our experiments, we use SGD+Momentum optimizer with learning rate 0.1, momentum 0.9 and weight-decay 10^{-4}, with step-wise learning rate decay.

For Table 1a, Table 1b and Table 2, we train the PreActResNet18, and PreActResNet34 for 1200 epochs with learning rate annealed by a factor of 10 at epoch 400 and 800. For above Tables, we train Wide-ResNet-28-10 for 400 epochs with learning rate annealed by a factor of 10 at epoch 200 and 300. In Table 3, we train PreActResNet18 for 2000 epochs with learning rate annealed by a factor of 10 at epoch 1000 and 1500.

For Table 4 and Table 6, we train the PreActResNet18 network for 2000 epochs with learning rate annealed by a factor of 10 at epoch 1000 and 1500.

For Table 7, Table 5 and Table 9, we train the networks for 1200 epochs with learning rate annealed by a factor of 10 at epoch 400 and 800.

In Figure 6 and Figure 7, we present the training loss (Binary cross entropy) for CIFAR10 and CIFAR100 datasets respectively. We observe that performing *Manifold Mixup* in higher layers allows the train loss to go down faster as compared to the Input Mixup, which suggests that while Input Mixup may suffer from underfitting, *Manifold Mixup* alleviates this problem to some extend.

In Table 9, we present full set of experiments of Section 5.2.

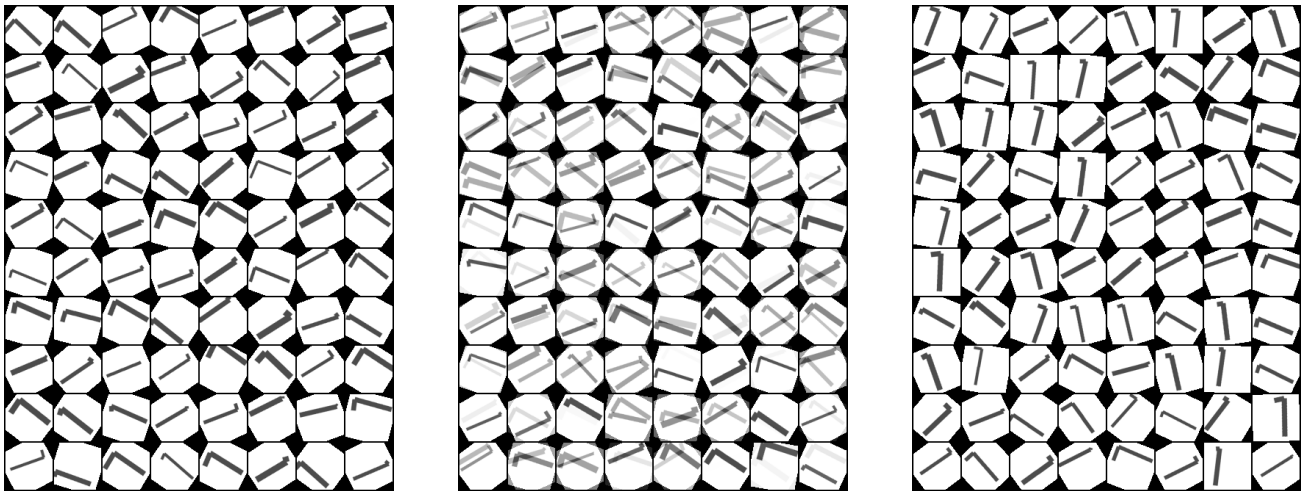

Figure 4: Synthetic task where the underlying factors are known exactly. Training images (left), images from input mixup (center), and images from mixing in the ground truth factor space (right).

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

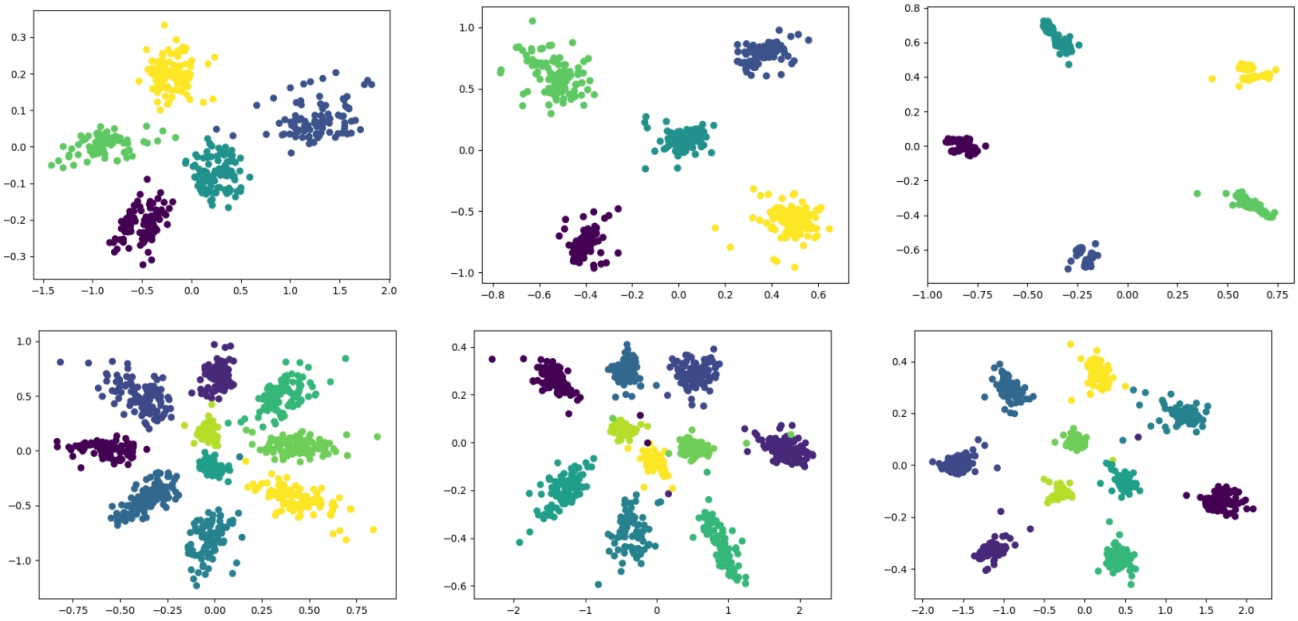

Figure 5: Representations from a classifier on MNIST (top is trained on digits 0-4, bottom is trained on all digits) with a 2D bottleneck representation in the middle layer. No Mixup Baseline (left), Input Mixup (center), *Manifold Mixup* (right).

C.1. Hyperparameter α

For Input Mixup on CIFAR10 and CIFAR100 datasets, we used the value $\alpha = 1.0$ as recommended in (Zhang et al., 2018). For Input Mixup on SVHN and Tiny-imagenet datasets, we experimented with the α values in the set $\{0.1, 0.2, 0.4, 0.8.1.0, 2.0, 4.0\}$. We obtained best results using $\alpha = 1.0$ and $\alpha = 0.2$ for SVHN and Tiny-imagenet, respectively.

For *Manifold Mixup*, for all datasets, we experimented with the α values in the set $\{0.1, 0.2, 0.4, 0.8.1.0, 2.0, 4.0\}$. We obtained best results with $\alpha = 2.0$ for CIFAR10, CIFAR100 and SVHN and with $\alpha = 0.2$ for Tiny-imagenet.

D. Adversarial Examples

We ran the unbounded projected gradient descent (PGD) (Madry et al., 2018) 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 a evidence that our defense did not improve results primarily as a result of gradient masking.

E. Generative Adversarial Networks

The recent literature has suggested that regularizing the discriminator is beneficial for training GANs (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., 2018). Mixup proposes the following objective[1]:

$$\max_{g} \min_{d} \mathbb{E}_{x_1, x_2, \lambda, z}\, \ell(d(\text{Mix}_\lambda(x_1, x_2)), y(\lambda; x_1, x_2)), \tag{4}$$

where x_1, x_2 can be either real or fake samples, and λ is sampled from a $Uniform(0, \alpha)$. Note that we have used a function $y(\lambda; x_1, x_2)$ to denote the label since there are four possibilities depending on x_1 and x_2:

$$y(\lambda; x_1, x_2) = \begin{cases} \lambda, & \text{if } x_1 \text{ is real and } x_2 \text{ is fake} \\ 1-\lambda, & \text{if } x_1 \text{ is fake and } x_2 \text{ is real} \\ 0, & \text{if both are fake} \\ 1, & \text{if both are real} \end{cases} \tag{5}$$

In practice however, we find that it did not make sense to create mixes between real and real where the label is

[1]The formulation written is based on the official code provided with the paper, rather than the description in the paper. The discrepancy between the two is that the formulation in the paper only considers mixes between real and fake.

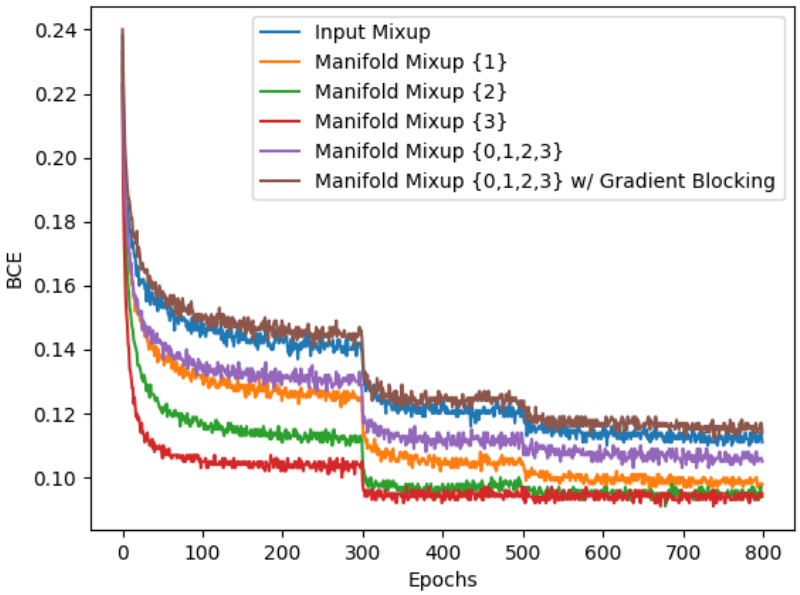

Figure 6: CIFAR-10 train set Binary Cross Entropy Loss (BCE) on Y-axis using PreActResNet18, with respect to training epochs (X-axis). The numbers in {} refer to the resblock after which *Manifold Mixup* is performed. The ordering of the losses is consistent over the course of training: Manifold Mixup with gradient blocked before the mixing layer has the highest training loss, followed by Input Mixup. The lowest training loss is achieved by mixing in the deepest layer, which suggests that having more hidden units can help to prevent underfitting.

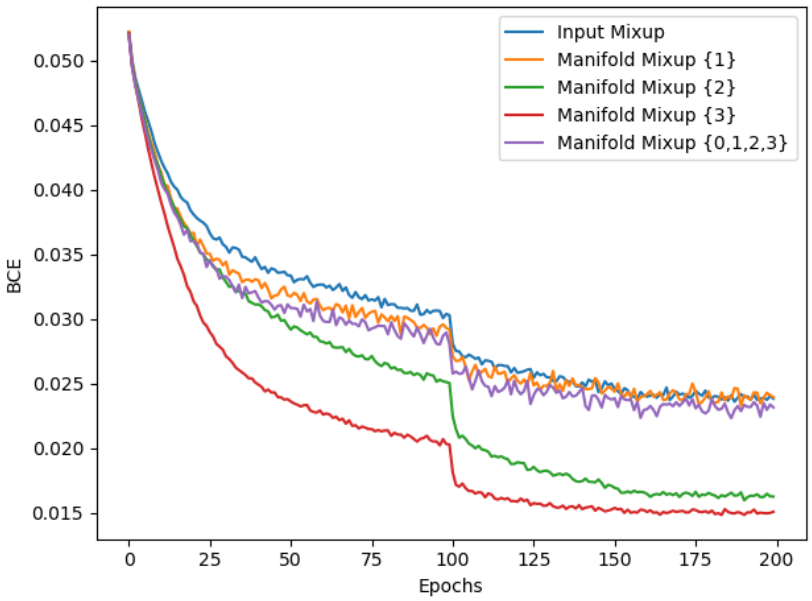

Figure 7: CIFAR-100 train set Binary Cross Entropy Loss (BCE) on Y-axis using PreActResNet50, with respect to training epochs (X-axis). The numbers in {} refer to the resblock after which *Manifold Mixup* is performed. The lowest training loss is achieved by mixing in the deepest layer.

Table 9: 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

set to 1, (as shown in equation 5), since the mixup of two real examples in input space is not a real example. So we only create mixes that are either real-fake, fake-real, or fake-fake. Secondly, instead of using just the equation in 4, we optimize it in addition to the regular minimax GAN equations:

$$\max_g \min_d \mathbb{E}_x \, \ell(d(x), 1) + \mathbb{E}_{g(z)} \, \ell(d(g(z)), 0) + \text{GAN mixup term (Equation 4)} \tag{6}$$

Using similar notation to earlier in the paper, we present the manifold mixup version of our GAN objective in which we mix in the hidden space of the discriminator:

$$\min_d \mathbb{E}_{x_1, x_2, \lambda, z, k} \, \ell(d(x), 1) + \ell(d(g(z), 0) + \ell(f_k(\text{Mix}_\lambda(g_k(x_1), g_k(x_2)), d(\lambda; x_1, x_2)) \tag{7}$$

where $g_k(\cdot)$ is a function denoting the intermediate output of the discriminator at layer k, and $f_k(\cdot)$ the output of the discriminator given input from layer k.

The layer k we choose the sample can be arbitrary combinations of the input layer (i.e., input mixup), or the first or second resblocks of the discriminator, all with equal probability of selection.

We run some experiments evaluating the quality of generated images on CIFAR10, using as a baseline JSGAN with spectral normalization (Miyato et al., 2018) (our configuration is almost identical to theirs). Results are averaged over at least three runs[2]. From these results, the best-performing mixup experiments (both input and *Manifold Mixup*) is with $\alpha = 0.5$, with mixing in all layers (both resblocks and input) achieving an average Inception / FID of 8.04 ± 0.08 / 21.2 ± 0.47, input mixup achieving 8.03 ± 0.08 / 21.4 ± 0.56, for the baseline experiment 7.97 ± 0.07 / 21.9 ± 0.62. This suggests that mixup acts as a useful regularization on the

[2]Inception scores are typically reported with a mean and variance, though this is across multiple splits of samples across a single model. Since we run multiple experiments, we average their respective means and variances.

discriminator, which is even further improved by *Manifold Mixup*. (See Figure 8 for the full set of experimental results.)

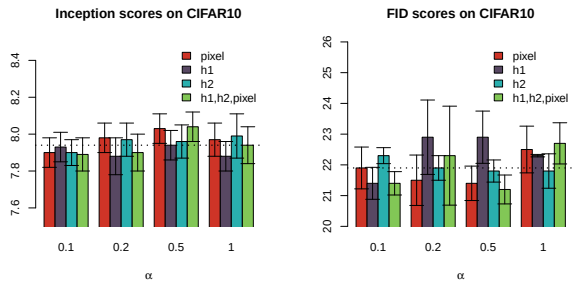

Figure 8: We test out various values of α in conjunction with either: input mixup (`pixel`) (Zhang et al., 2018), mixing in the output of the first resblock (`h1`), mixing in either the output of the first resblock or the output of the second resblock (`h1,2`), and mixing in the input or the output of the first resblock or the output of the second resblock (`1,2,pixel`). The dotted line indicates the baseline Inception / FID score. Higher scores are better for Inception, while lower is better for FID.

F. Intuitive Explanation of how Manifold Mixup avoids Inconsistent Interpolations

An essential motivation behind manifold mixup is that as the network *learns* the hidden states, it does so in a way that encourages them to be a flatter (per-class). Section 3.1 characterized this for hidden states with any number of dimensions and Figure 1 showed how this can occur on the 2D spiral dataset.

Our goal here is to discuss concrete examples to illustrate why this flattening happens, as shown in Figure 3. If we consider any two points, the interpolated point between them is based on a sampled λ and the soft-target for that interpolated point is the targets interpolated with the same λ. So if we consider two points A,B which have the same label, it is apparent that every point on the line between A and B should have that same label with 100% confidence. If we consider two points A,B with different labels, then the point which is halfway between them will be given the soft-label of 50% the label of A and 50% the label of B (and so on for other λ values).

It is clear that for many arrangements of data points, it is possible for a point in the space to be reached through distinct interpolations between different pairs of examples, and reached with different λ values. Because the learned model tries to capture the distribution $p(y|h)$, it can only assign a single distribution over the label values to a single particular point (for example it could say that a point is 100% label A, or it could say that a point is 50% label A and 50% label B). Intuitively, these inconsistent soft-labels at interpolated points can be avoided if the states for each class are more concentrated and the representations do not have variability in directions pointing towards other classes. This leads to flattening: a reduction in the number of directions with variability. The theory in Section 3.1 characterizes exactly what this concentration needs to be: that the representations for each class need to lie on a subspace of dimension equal to "number of hidden dimensions" - "number of classes" + 1.

G. Spectral Analysis of Learned Representations

When we refer to *flattening*, we mean that the class-specific representations have reduced variability in some directions. Our analysis in this section makes this more concrete.

We trained an MNIST classifier with a hidden state bottleneck in the middle with 12 units (intentionally selected to be just slightly greater than the number of classes). We then took the representation for each class and computed a singular value decomposition (Figure 9 and Figure 10) and we also computed an SVD over all of the representations together (Figure 12). Our architecture contained three hidden layers with 1024 units and LeakyReLU activation, followed by a bottleneck representation layer (with either 12 or 30 hidden units), followed by an additional four hidden layers each with 1024 units and LeakyReLU activation. When we performed *Manifold Mixup* for our analysis, we only performed mixing in the bottleneck layer, and used a beta distribution with an alpha of 2.0. Additionally we performed another experiment (Figure 11 where we placed the bottleneck representation layer with 30 units immediately following the first hidden layer with 1024 units and LeakyReLU activation.

We found that *Manifold Mixup* had a striking effect on the singular values, with most of the singular values becoming much smaller. Effectively, this means that the representations for each class have variance in fewer directions. While our theory in Section 3.1 showed that this flattening must force each classes representations onto a lower-dimensional subspace (and hence an upper bound on the number of singular values) but this explores how this occurs empirically and does not require the number of hidden dimensions to be so small that it can be manually visualized. In our experiments we tried using 12 hidden units in the bottleneck Figure 9 as well as 30 hidden units Figure 10 in the bottleneck.

Our results from this experiment are unequivocal: *Manifold Mixup* dramatically reduces the size of the smaller singular values for each classes representations. This indicates a flattening of the class-specific representations. At the same time, the singular values over all the representations are not changed in a clear way (Figure 12), which suggests that this

flattening occurs in directions which are distinct from the directions occupied by representations from other classes, which is the same intuition behind our theory. Moreover, Figure 11 shows that when the mixing is performed earlier in the network, there is still a flattening effect, though it is weaker than in the later layers, and again Input Mixup has an inconsistent effect.

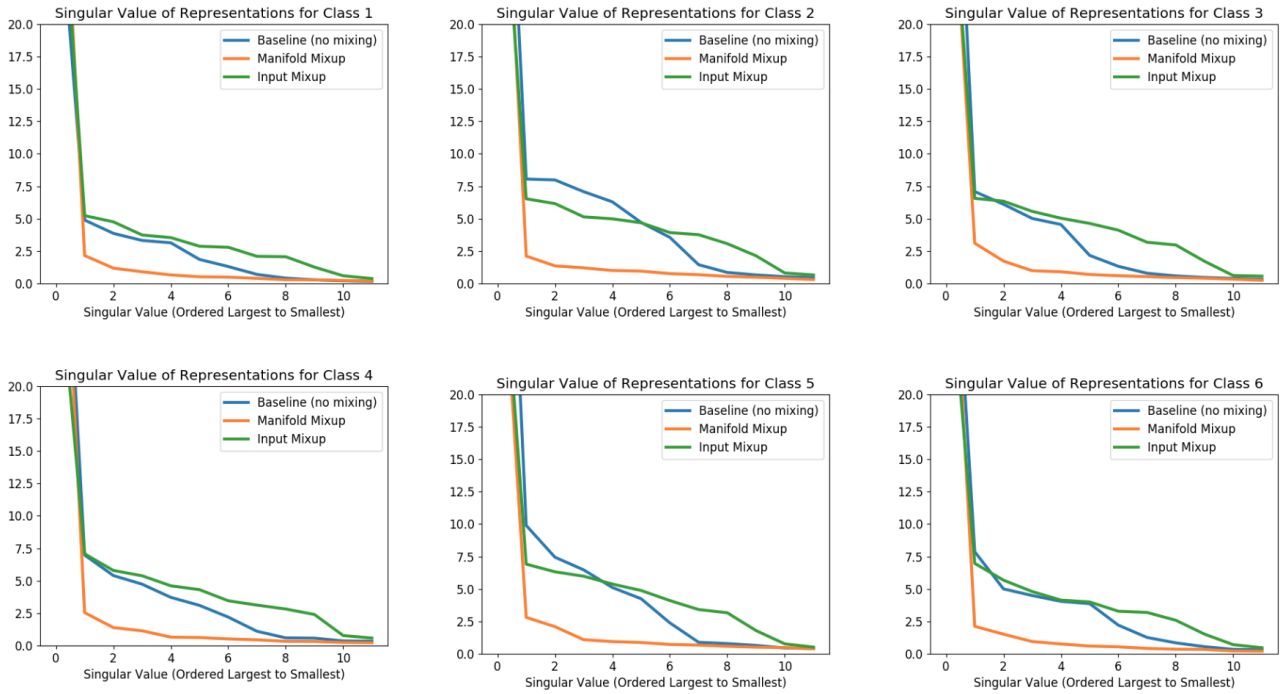

Figure 9: SVD on the class-specific representations in a bottleneck layer with 12 units following 3 hidden layers. For the first singular value, the value (averaged across the plots) is 50.08 for the baseline, 37.17 for Input Mixup, and 43.44 for *Manifold Mixup* (these are the values at x=0 which are cutoff). We can see that the class-specific SVD leads to singular values which are dramatically more concentrated when using *Manifold Mixup* with Input Mixup not having a consistent effect.

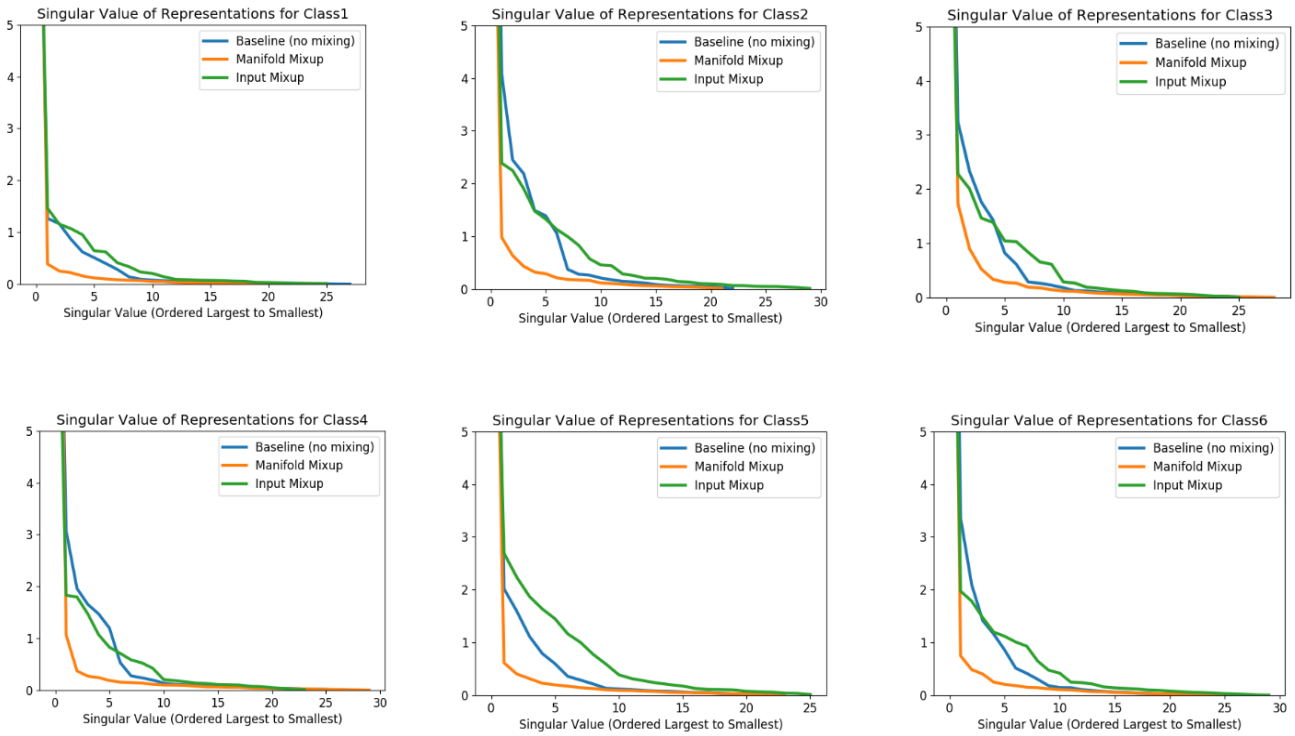

Figure 10: SVD on the class-specific representations in a bottleneck layer with 30 units following 3 hidden layers. For the first singular value, the value (averaged across the plots) is 14.68 for the baseline, 12.49 for Input Mixup, and 14.43 for *Manifold Mixup* (these are the values at x=0 which are cutoff).

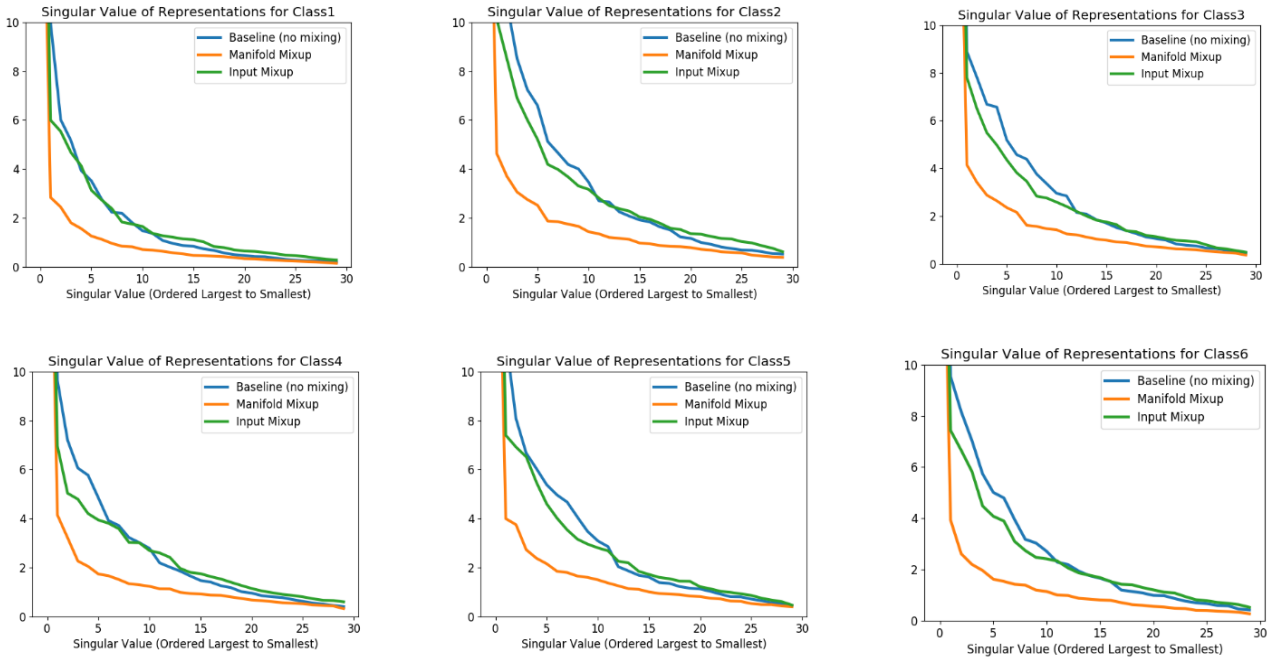

Figure 11: SVD on the class-specific representations in a bottleneck layer with 30 units following a single hidden layer. For the first singular value, the value (averaged across the plots) is 33.64 for the baseline, 27.60 for Input Mixup, and 24.60 for *Manifold Mixup* (these are the values at x=0 which are cutoff). We see that with the bottleneck layer placed earlier, the reduction in the singular values from *Manifold Mixup* is smaller but still clearly visible. This makes sense, as it is not possible for this early layer to be perfectly discriminative.

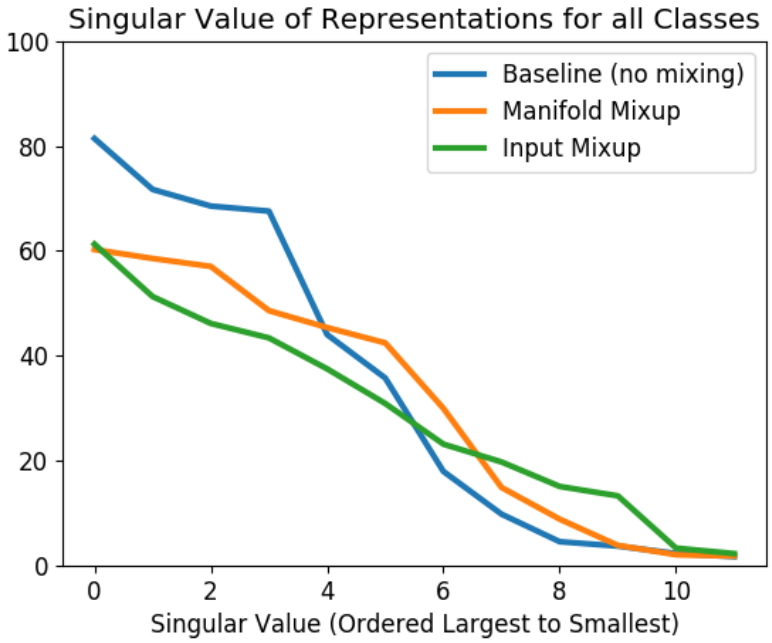

Figure 12: When we run SVD on all of the classes together (in the setup with 12 units in the bottleneck layer following 3 hidden layers), we see no clear difference in the singular values for the Baseline, Input Mixup, and *Manifold Mixup* models. Thus we can see that the flattening effect of manifold mixup is entirely class-specific, and does not appear overall, which is consistent with what our theory has predicted. More intuitively, this means that the directions which are being flattened are those directions which point towards the representations of different classes.