A. Soft Nearest Neighbor Loss on Toy distribution

This Figure complements Figure 1. It adds a second mode to each class of the distribution, showing that minimizing entanglement through the soft nearest neighbor loss preserves the two modes in each class.

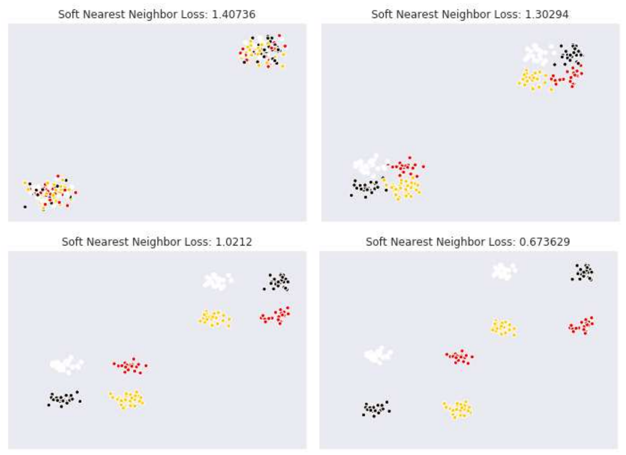

Figure 13. Data is generated for each class by sampling from two Gaussians. As entanglement is minimized using gradient descent on the (x, y) coordinates of the points, each class does not collapse into a single point; instead, both initial modes are preserved.

B. Comparing the Soft Nearest Neighbor Loss with the Triplet Loss

Other approaches have previously explored the use of implicit nearest neighbor search as a regularization term in the loss to handle noisy data (Azadi et al., 2015). Our soft nearest neighbor loss is perhaps most similar to the triplet loss (Hoffer & Ailon, 2015), in that both measure the relative distance between points from the same class and points from different classes. The triplet loss is calculated by taking the maximum of 0 and the difference between (a) the distance between an anchor point and a positive point (in the same class) and (b) the distance between the anchor point and a negative point (in a different class) for every anchor point in a batch. Equation 4 presents the triplet loss where x_i^a denotes the anchor point, x_i^a a positive sample, x_i^n a negative one and α the margin term:

$$L = \sum_{i}^{N} \left(||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2 + \alpha \right) \quad (4)$$

Minimizing the triplet loss should have a similar effect on learned representations as minimizing entanglement (by minimizing the soft nearest neighbor loss) as both are imposing constraints on the relative distance between points within a class and points in different classes. However, a notable difference is that the triplet loss is calculated by sampling positive and negative points to estimate the separation of classes whereas the soft nearest neighbor loss uses all of the points in a batch to measure the separation.

In Figure 2, we compare minimizing and maximizing these two similar losses by visualizing the results of minimizing and maximizing a random set of 2 dimensional points labeled in four classes. We see that both losses have similar effects when the loss is minimized: the classes are separated by a larger margin. However, when the loss is maximized, the end results are not identical: the triplet loss chooses a representation that densely projects the data around a circle whereas the soft nearest neighbor loss spreads out data throughout the representation space.

We provide an additional point of comparison: the impact of both losses on the calibration of DkNN credibility estimates. We train MNIST models with cross-entropy and a regularizer maximizing either the triplet loss or the soft nearest neighbor loss at each layer, as done in Section 5.1. We report the accuracy of DkNN predictions with respect to their credibility in Figure 14. We did not find improved DKNN calibration for networks trained with the triplet loss term—unlike models maximizing entanglement through the soft nearest neighbor term.

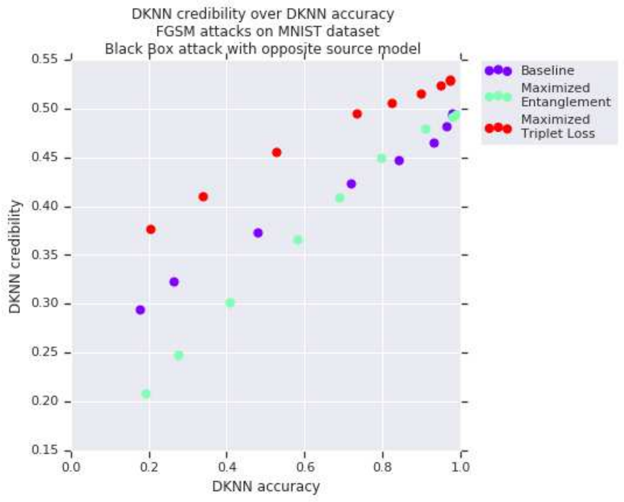

Figure 14. DKNN credibility over accuracy for white box FGSM attacks with varying epsilons, plotted for MNIST. Maximizing the triplet loss seems to have the opposite effect of maximizing the soft nearest neighbor loss.

C. Additional Entanglement Measurements

We report here entanglement measurements made with the soft nearest neighbor loss on MNIST and CIFAR10 models. They complement results presented in Section 3.1, which demonstrated the use of the soft nearest neighbor loss as an

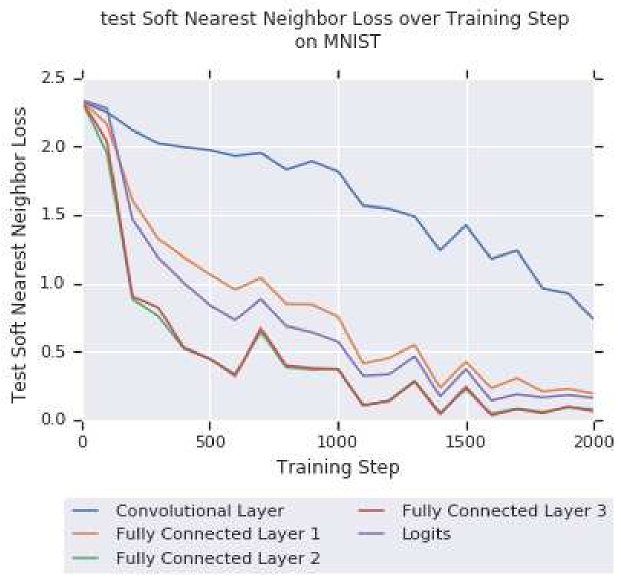

Figure 15. Soft nearest neighbor loss value per layer of a neural network on MNIST. The loss decreases during training despite the model being optimized to minimize cross-entropy only.

analytical tool to follow the evolution of similarity structures during learning in models trained to minimize cross-entropy.

MNIST. We trained a neural network with one convolutional layer and three fully-connected layers on MNIST and measured the Soft Nearest Neighbor Loss of each training batch at each layer during training. Note in Figure 15 how the loss value decreases throughout training, unlike results presented in Section 3. This is most likely because MNIST is easier to separate in the input domain than other datasets considered in our work.

CIFAR10. We repeat the experiment presented in Section 3.1 but now looking at all residual blocks instead of only the last one. In Figure 16, we report the average soft nearest neighbor loss of the layers contained in each residual block, across all of the training data throughout learning. Results are consistent with Section 3.1. Entanglement is fairly constant or increases as training progresses in the first three blocks; suggesting a large amount of feature co-adaptation across classes in the corresponding layers. Instead, the final block's entanglement monotonically decreases as it extracts discriminative features to classify the input. When measuring Soft Nearest Neighbor Loss within a resnet with large hidden layers, we use cosine distance $(1 - cos(\boldsymbol{x}, \boldsymbol{y}))$ instead of euclidean distance to ensure stable calculations.

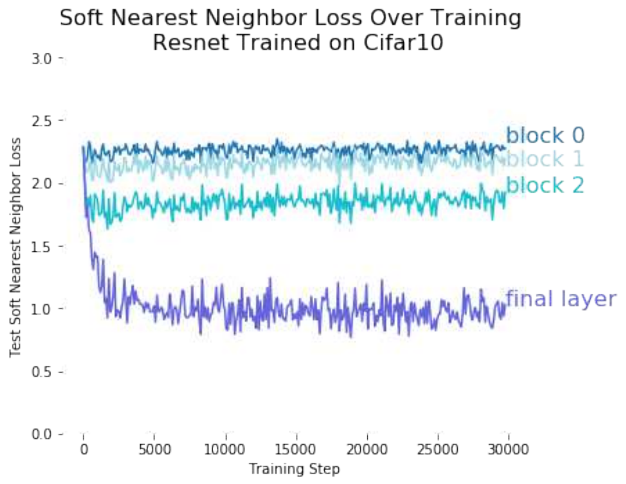

Figure 16. Entanglement loss averaged within each residual block of a ResNet on CIFAR-10. Entanglement remains high throughout learning for the lower blocks, as they extract features that help discriminate classes, and only decreases in the final block.

D. Soft Nearest Neighbor Loss as an Analytical Tool for Generative Models

In Section 3, we showed how the soft nearest neighbor loss allows us to monitor the entanglement of synthetic data with real training data when learning a generative model on CIFAR10. Here, Figure 17 is the analog of Figure 5 for the MNIST dataset: it plots the entanglement between synthetic and real data, as measured by the soft nearest neighbor loss, on three variants of GANs.

Note that the similar values of the soft nearest neighbor loss for MNIST and CIFAR-10 are a coincidence. We can calculate the entanglement a perfect generative model would achieve by measuring the SNNL between batches of data from the same distribution: e.g., the test and training sets. These values are very close (0.698 for MNIST and 0.697 for CIFAR-10) because of the data pre-processing we applied to both datasets: we normalized all dimensions to have pixel values in [0,1]. If we repeat the computation with unnormalized test and training data, that is with integer pixel values between 0 and 255, the optimal SNNL value for MNIST is 11.5 and 4.95 for CIFAR-10.

E. Does Entanglement conflict with Robustness?

We reproduce the adversarial training procedure from Madry et al. (2017), where adversarial examples are generated with projected gradient descent (that is with multiple gradient steps and random restarts). The training objective *only* minimizes cross-entropy over these adversarial examples. Once the model is trained, we measure the entanglement of its

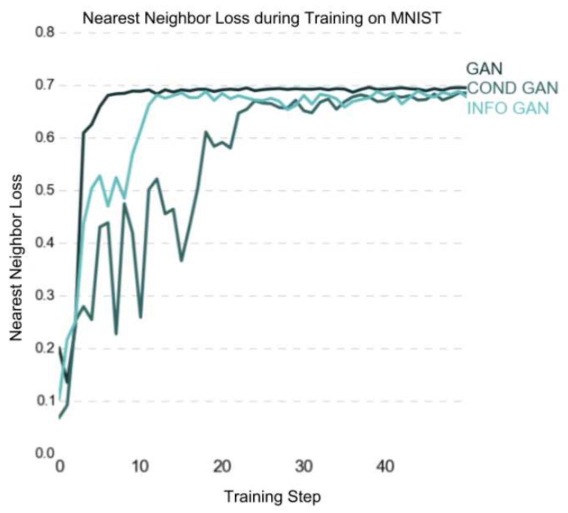

Figure 17. Entanglement of synthetic and real data, as measured by the soft nearest neighbor loss, on three different types of GAN architectures trained on MNIST.

Layer	Baseline model	PGD model
Conv1 (after pool)	1.39	2.21
Conv2 (after pool)	0.75	1.97
Fully Connected layer	1.75	0.46
Logits	0.13	0.21

Table 2. Entanglement loss measured on models trained to minimize cross-entropy on the original training data (baseline model) or adversarial examples (PGD model). Measurements were made at temperature $T = 100$ on a batch of 128 MNIST test points.

hidden layers using the soft nearest neighbor loss. The same architecture, also trained to minimize cross-entropy but on *non-adversarial* data, serves as a baseline to interpret these entanglement measurements. As reported in Table 2, we find that the adversarially trained model's convolutional layers are more entangled than the baseline model's, despite not being explicitly constrained to maximize entanglement during training. This further supports our hypothesis that increased entanglement of representation spaces is beneficial to the similarity structure of internal representations and can support better (here, worst-case) generalization.

F. DkNN Uncertainty Calibration

We include here reports of the DkNN uncertainty calibration on entangled MNIST (Figure 18) and FashionMNIST (Figure 20) models. The experiment performed is the one described in Section 5.1, where the plot visualizes DkNN credibility as a function of DkNN prediction accuracy.

G. Out-of-Distribution Test Inputs

This experiment complements results on SVHN and CIFAR10 in Section 5.3, where we showed that maximizing entanglement leads to representations that better separate test data from data sampled from a different distribution. We repeat the experiment on MNIST and notMNIST.

We train a network on MNIST and see what its behavior is like on notMNIST, a data set made up of MNIST-sized typeface characters between letters A and J. Test examples from the notMNIST dataset should thus be projected very differently by a model trained on MNIST, when compared to examples from the MNIST test set. This is indeed what we observe in Figure 21, which uses t-SNE to visualize how the logits project MNIST and notMNIST test inputs when a model is trained with cross-entropy only or with the soft nearest neighbor loss to maximize entanglement. We observe that the vanilla model makes confident predictions in the MNIST classes for the notMNIST inputs (because they are projected close to one another), whereas the entangled model separates all of the notMNIST data in a distinct cluster and preserves the MNIST clusters.

H. Intuition for Improved DkNN Calibration

In Figure 22, we visualize the activations of a hidden layer on real and adversarial test data. In the non-entangled model trained with cross-entropy, the adversarial data is projected close to the real test data. Instead, on the entangled model, the representation separates better the real and adversarial data. This in turn, results in a better estimate of the number of training neighbors that support the prediction made. As a consequence, the DkNN is able to provide more calibrated estimates of uncertainty on entangled representations.

I. Soft Nearest Neighbor GANs

In Section 3, we found that the entanglement loss can effectively replace the discriminator in a GAN setup on MNIST. However, we were unable to scale the same setup to train a CIFAR10 model. We hypothesized that this is due to the fact that the ℓ_2 distance does not characterize CIFAR10's input domain as well as it does for MNIST. Hence, we run an additional experiment restoring the discriminator but modifying the typical losses used to train GANs: we constrain the generator to entangle synthetic and real data in a 10 dimensional space using the soft nearest neighbor loss while the discriminator is constrained to disentangle the synthetic and real data. While this is simply a proof-of-concept on MNIST, results summarized in Figure 23 demonstrate that this approach deserves further investigation and may scale to larger datasets given the discriminator's ability to learn how to compare points compared to a direct application of the ℓ_2 distance in the pixel space.

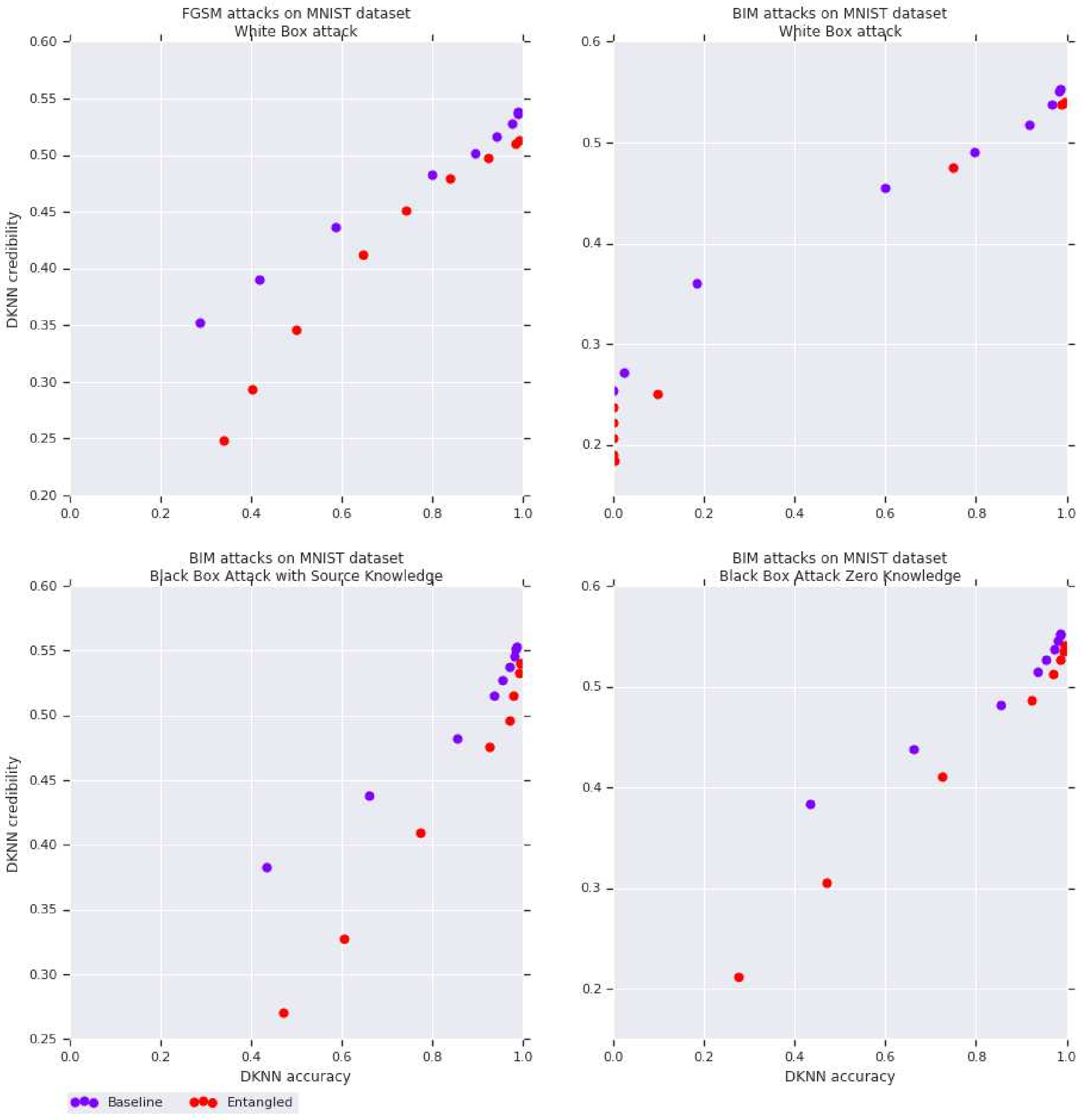

Figure 18. DkNN credibility (prediction support from the training data) as a function of prediction accuracy on MNIST. These plots were created with the same methodology as Figure 9.

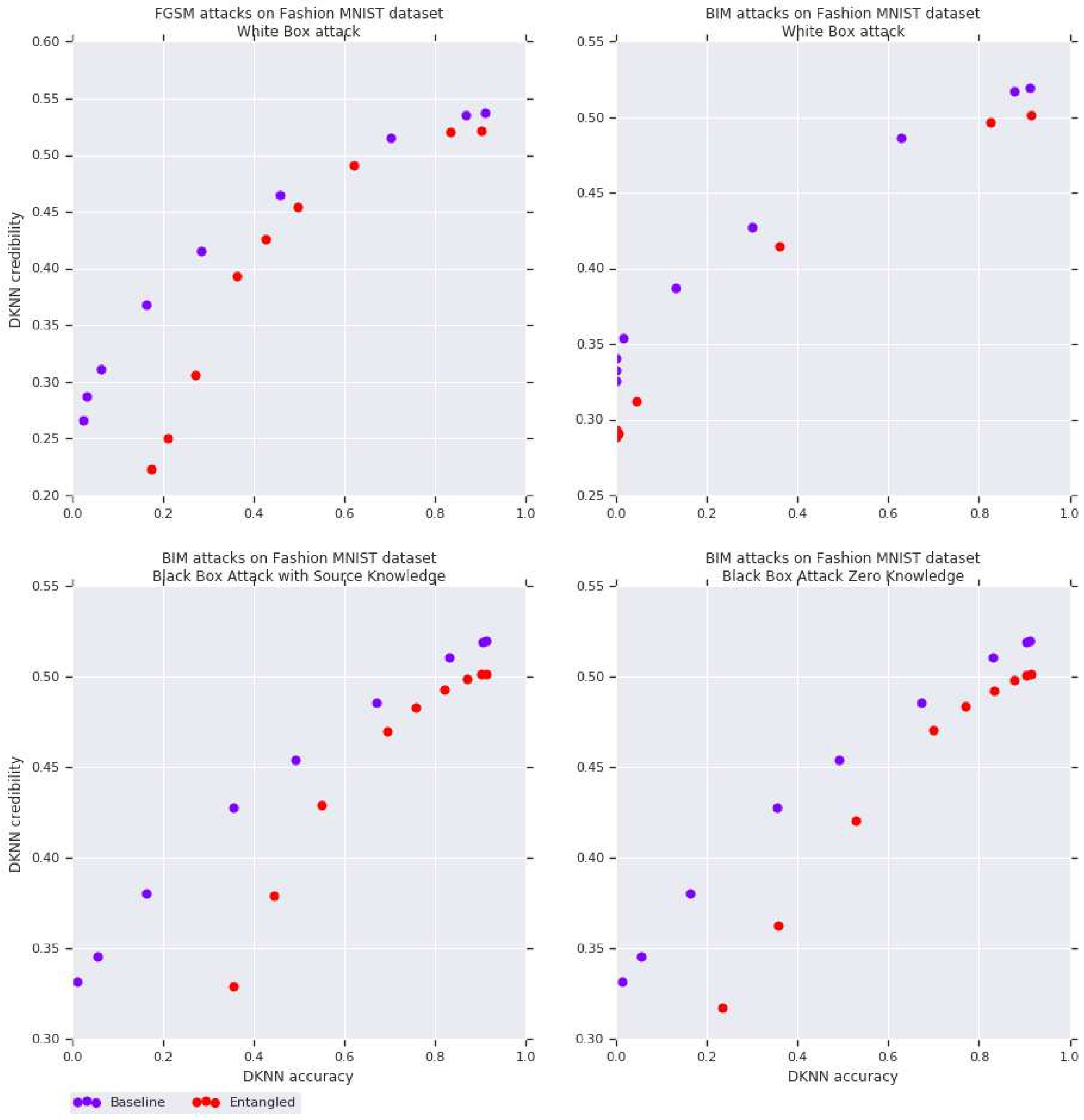

Figure 19. DkNN credibility (prediction support from the training data) as a function of prediction accuracy on FashionMNIST. These plots were created with the same methodology as Figure 9.

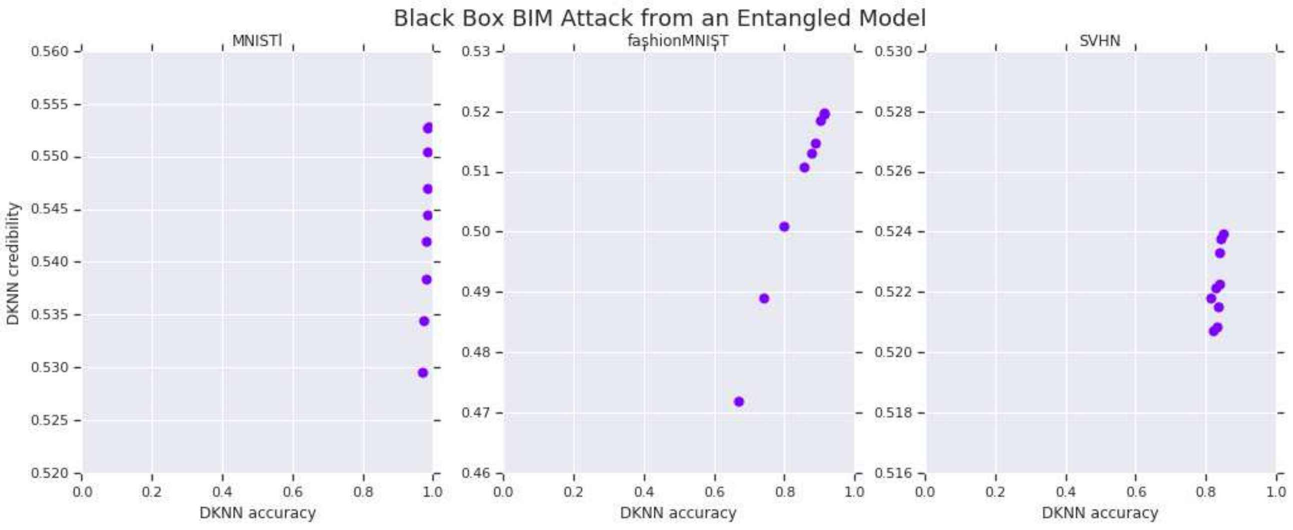

Figure 20. DkNN credibility as a function of prediction accuracy on all three datasets, where the underlying model is trained with cross-entropy only. Each point corresponds to a set of adversarial examples computed on an entangled model and are transferred to the DkNN with a model trained on cross-entropy only. These plots were created with the same methodology as Figure 9. As explained in Section 5.2, entangled models make poor source models for a black box attack based on transferability: the accuracy of the cross-entropy baseline remains high on all three datasets; the attack is noticeably less effective than in previous settings considered above.

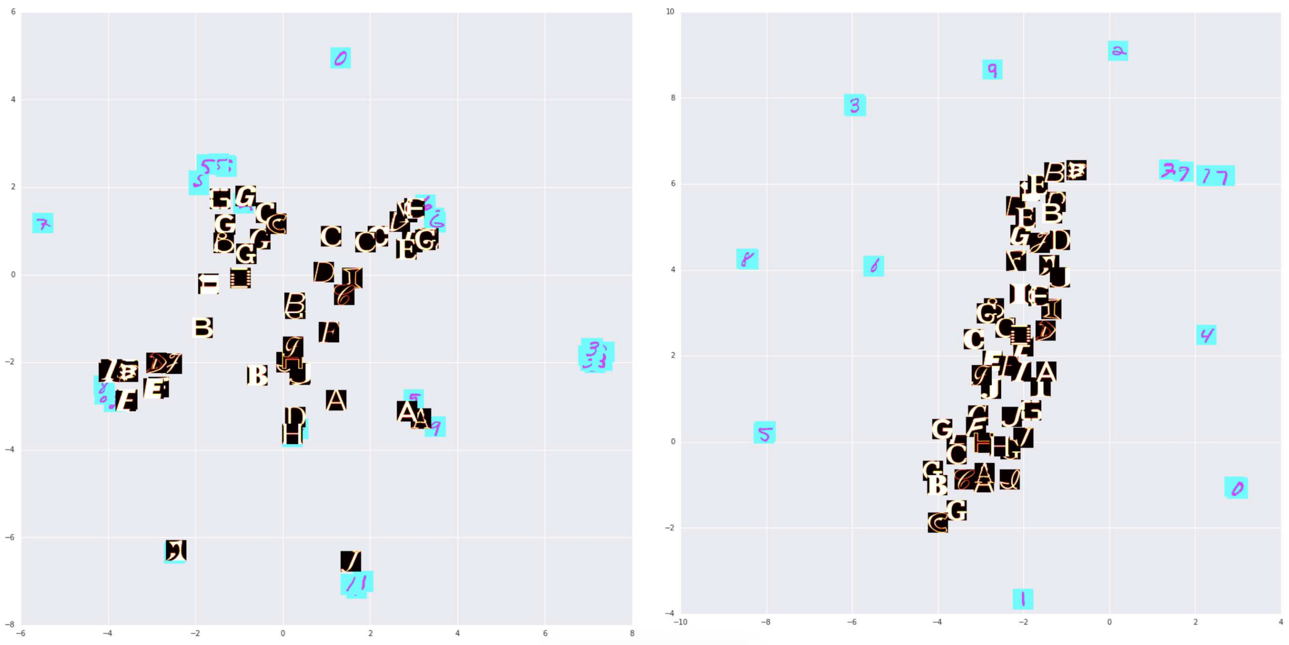

Figure 21. t-SNE visualization of representations of in-distribution (MNIST—blue) and out-of-distribution (notMNIST—dark) test data learned by a vanilla (left) and entangled (right) model.

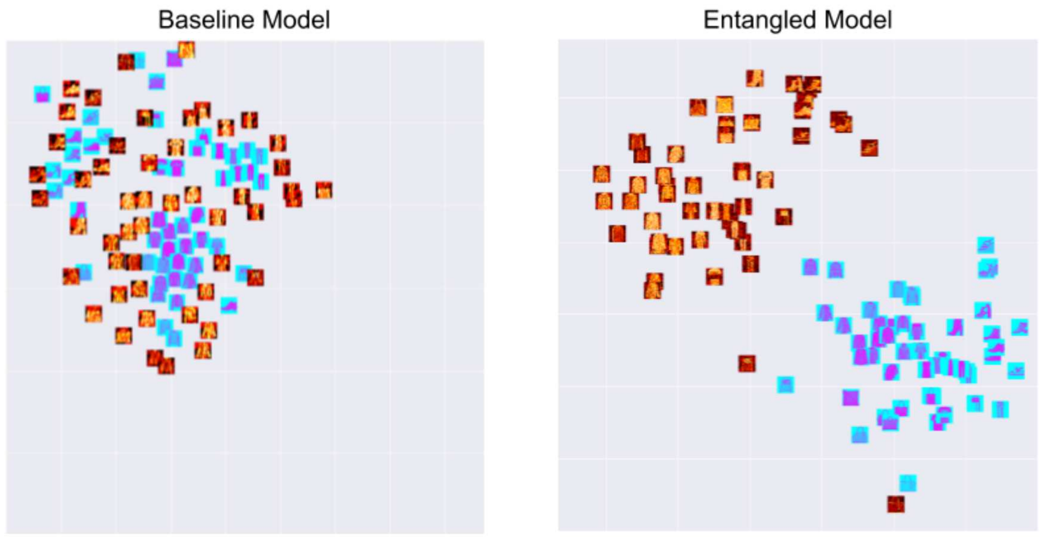

Figure 22. t-SNE visualization of the activations from the first hidden layer of a network trained on FashionMNIST. Real data points are plotted in blue whereas adversarial data points are visualized in red. In the vanilla (non-entangled) model, the representations of adversarial data and real data occupy a similar part of activation space. Instead, in the entangled model, the adversarial data is projected into a separate area. This provide some intuition for why entangled models have better calibrated DkNN uncertainty estimates: it is easier to evaluate support in a particular model prediction through a nearest neighbor search in the training data given a test point.

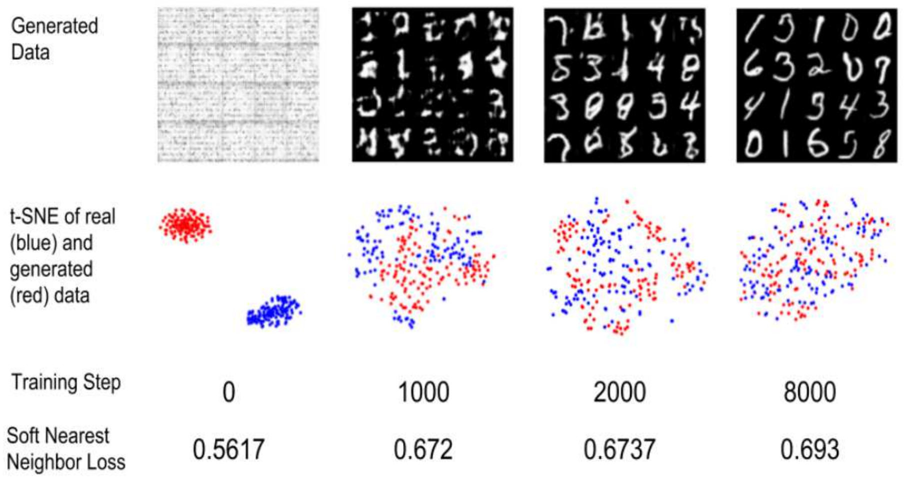

Figure 23. Training progression of a MNIST GAN in which the discriminator minimizes the Soft Nearest Neighbor Loss between real and synthetic data in a learned 10 dimensional space, and the generator maximizes it. We replaced the output layer of the discriminator with a 10 dimensional vector. This proof-of-concept demonstrates that the soft nearest neighbor loss can be used in a learned space as well as the pixel space, which is explored more thoroughly in the main text.

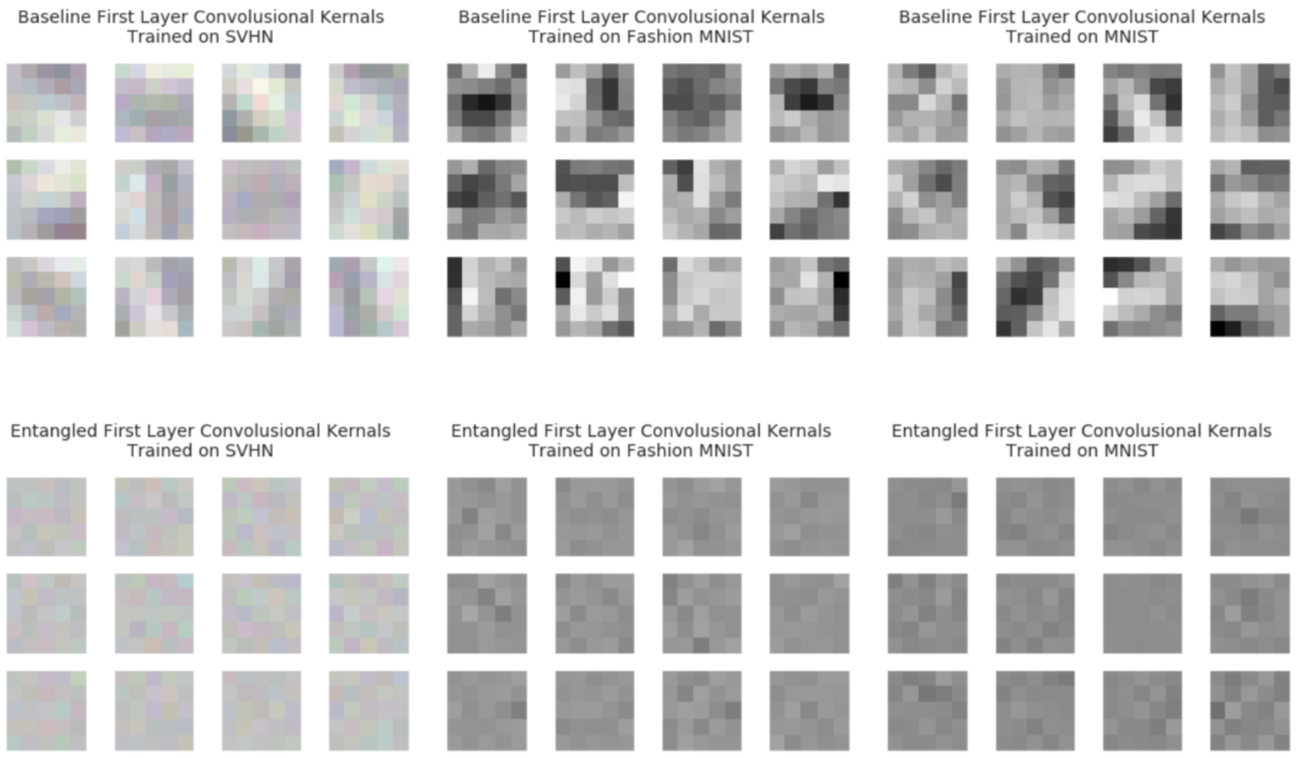
Baseline First Layer Convolusional Kernals
Trained on SVHN
Baseline First Layer Convolusional Kernals
Trained on Fashion MNIST
Baseline First Layer Convolusional Kernals
Trained on MNIST
Entangled First Layer Convolusional Kernals
Trained on SVHN
Entangled First Layer Convolusional Kernals
Trained on Fashion MNIST
Entangled First Layer Convolusional Kernals
Trained on MNIST

Figure 24. The kernels of the first convolutional layers of entangled models look quite a bit different than standard models. They look considerably noisier and less coherent. This may be due to that fact that SNNL was maximized in the first layer as well as all other layers, so the first level features will project the data into class independent clusters.