Max Yuhas
Deep Learning – Final Project
Professor Krishnaswamy
TA: Matthew Amodio

Latent Space Representations for More Complex CIFAR-10 Images

Abstract

The MNIST handwritten digit dataset has been used as the baseline data set for many deep-learning applications. In assignment three, we used a flat, fully connected variational autoencoder to visualize the latent space representation of this data. Here, I use several convolutional variational autoencoders to visualize the latent representations and reconstructions for the CIFAR-10 dataset, a more complex ten-class image set of different objects, to compare to the MNIST data. I built three different networks to train: a simple convolutional variational autoencoder, a high-dimension convolutional variational autoencoder, and a deep convolutional variational autoencoder. My results show that these networks, when trained on CIFAR-10, are able to learn latent distributions that are slightly distinct by class, but are not able to accurately reconstruct images. They perform much worse on CIFAR-10 than on the MNIST data, because the colored images and image subjects are much more complex. However, there is evidence to suggest that continued training with the high-dimensional network could create fairly accurate test image reconstructions.

Introduction

In assignment three, we were tasked with creating a simple, flat, fully connected variational autoencoder to visualize the two-dimensional latent space for the MNIST dataset of handwritten digits. This simple ten-class dataset is known to have simple characteristics and perform extremely well with many machine-learning and deep-learning tasks. However, the CIFAR-10 dataset is a much more complex dataset of colored RGB images of ten classes –

airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks – all with different backgrounds and additional features. Both of these datasets are included as ready to use TensorFlow Datasets and are not extremely large (MNIST 60,000 train/10,000 test 28x28x1 images and CIFAR-10 50,000 train/10,000 test 32x32x3 images), allowing me to perform my analysis on my individual computer.[1] These datasets are both images with ten unique classes, but have main differences in their subjects (digits vs. objects) and numbers of input channels (one vs. three). My goal is to create a convolutional variational autoencoder to visualize both latent spaces and observe the reconstructive capability on example images from each dataset. I want to compare the latent space distribution for the MNIST dataset with the CIFAR-10 to see if the variational autoencoder is still able to learn distinct representations for each class in the more complex image set.

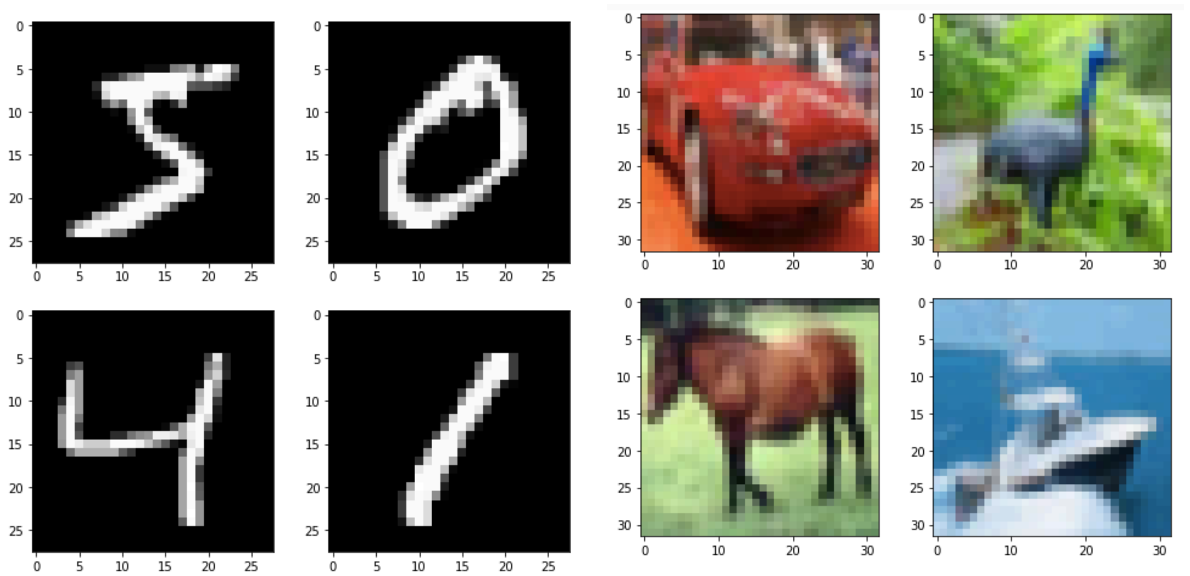

Figure 1: Example training images for MNIST (left) and CIFAR-10 (right).

[1] I ran into trouble using the Zoo computers, as my code required a module that I could not install on the computers. So I needed to do much of the computation on my own laptop.

Background Information

Traditional autoencoders are used to encode data to lower dimensions and then decode the data back to a reconstructed version. Their performance is based on the reconstructed images error compared to the original input image. Encoding the input images in lower dimensions requires the network to learn particular features of the data, for example a noses or eyes in a data set of faces. However, the vanilla autoencoder only encodes images to a single point in the latent space. This prevents the decoder from decoding any point from the latent space that has not previously been encoded, meaning we cannot properly visualize or sample from this representational space. Variational autoencoders instead embed the inputs as vectors of expectations and standard deviations for normal distributions. The decoder then takes a random sample from these distributions as an input to reconstruct the original image. This change allows us to visualize the latent space manifold as a set of probability distributions and to reconstruct images from artificially generated latent space encodings. To ensure that the distributions are similar and not too separable, otherwise making it difficult to sample from the latent space distributions, an additional loss is included, the Kulback-Leibler divergence loss:

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

Minimizing this loss between probability distributions in addition to the pixel-wise Mean Squared Error reconstruction loss ensures that the latent distributions for each class are similar.

Another useful aspect of autoencoders is that the encoding and decoding networks can have any structure – flat multi-perceptron networks, recurrent neural networks, convolutional neural networks, etc. Deep learning models for image data have had great success using convolutional neural networks compared to flat multi-perceptron or recurrent networks, as the convolution filters are good at learning features in images (indeed these filters are often called feature maps). For example, deep convolutional networks have been used to classify the MNIST digits with a 0.23 error rate.[2] A convolutional network contains multiple layers of convolution steps and then flattens the image as an input into a fully connected network that creates an output. In the variational autoencoder, this output is the lower dimensioned encoding for the original input image. The decoder for the variational autoencoder is structured in the reverse of the convolutional network described above. The sampled encoding is fed into a flat, fully connected network and then is deconvolved through multiple layers back to the original image shape.

In general, the MNIST dataset is simple enough that a variational autoencoder with flat fully connected networks creates a clear latent space representation and accurate reconstructions. However, the complexity of the CIFAR-10 requires the use of a convolutional network to hopefully improve the reconstructive ability and learn a distinct latent space manifold.

Methods

After reading in the datasets from TensorFlow, I preprocessed the data in two ways. First, I normalized all the pixel values by dividing by 255, to help prevent saturation in any neurons. Second, I reshaped each dataset to be of the shape [number of images, height, width, number of channels] so they could both be easily be fed into a general neural network architecture. Since MNIST has only one channel, it's original format was just [number of images, height, width,].

I built three different neural networks, all different variations on a convolutional variational autoencoder. The general network I built contains two convolutional layers and two

[2] Ciresan, Dan; Meier, Ueli; Schmidhuber, Jürgen (June 2012). *Multi-column deep neural networks for image classification. 2012 IEEE Conference on Computer Vision and Pattern Recognition.* New York, NY: Institute of Electrical and Electronics Engineers (IEEE). pp. 3642-3649.

flat layers in both the encoder and decoder. The first convolutional layer has 32 filters and the second layer has 64 filters, both with 3x3x(number of channels) kernel size, no stride, and no padding. After each convolutional layer, I perform max pooling with a 2x2 kernel to reduce the number of weights in our network. After the second convolution and pooling, the output is flattened and is the input to the first flat layer with 256 neurons. The second fully connected layer has 64 neurons and outputs mean and standard deviation vectors for the two-dimensional latent space. All of the layers, both convolutional and flat, use the ReLU activation function. This network is used to learn the latent space representation in the MNIST dataset as a baseline and then is used again on the CIFAR-10 dataset.

To try and create an improved reconstruction for the CIFAR-10 data, I adjusted this network by increasing the latent space dimension to 16. We are unable to then understand this latent space graphically, because there is no way to easily visualize 16 dimensions. However, by increasing this dimensionality, less data should be lost in the encoding and reconstruction capability should be improved.

Finally, to improve both the reconstruction capability and latent space representation of the original network, I created a third deep convolutional network. I added an additional flat fully connected layer and increased the number of filters and neurons in many of the layers. The first and second convolutional layers now have 32 and 128 filters, still with 3x3x(number of channels) kernel size, no stride, and no padding. Max pooling with a 2x2 kernel is also performed after each convolution. Then, the three flat fully connected layers have 1024, 256, and 32 neurons before outputting to the two-dimensional encoding space. To improve the latent space distributions, I choose to down weight the Kullback-Leibler divergence loss by 0.5. By reducing the weight of this loss, the distributions should not be forced to be as similar, hopefully spreading them out more.

Each network minimized the pixel-wise mean squared error reconstruction loss using the Adam Optmizer:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i-1}^{n}(Y_i - \hat{Y}_i)^2$$

Here n is the number of pixels and Y_i identifies the individual pixel value for each image. The simple and deep networks both included Kullback-Leibler divergence loss, defined in the background section, to help improve the latent space representation visualization.

Training was the same for all of the networks. After some hyperparameter experimentation, I choose to use a mini batch size of 50 random images, a learning rate of 0.0001, and trained for 10 epochs. Note that the training size of MNIST has 10,000 more images then CIFAR-10, so over 10 epochs with 50 mini-batch size it has 2,000 more iterations.

Results

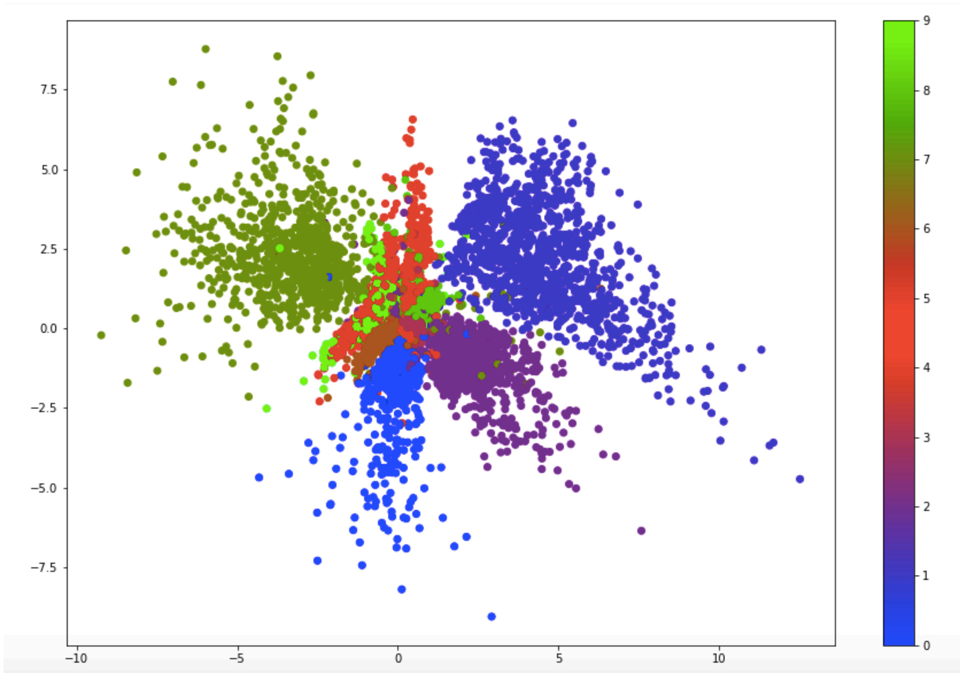

Figure 2: MNIST latent space distribution for test data with the simple convolution VAE.

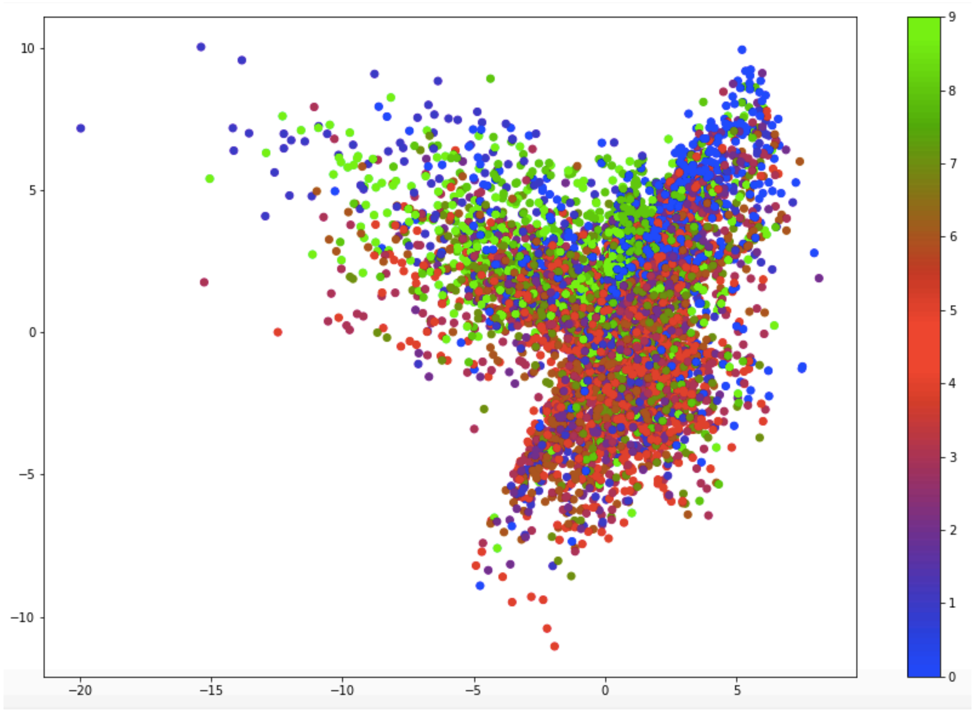

Figure 3: CIFAR-10 latent space distribution for test data with the simple convolutional VAE. The digits 0-9 on the color bar correspond to the ten different classes: airplanes (0), cars (1), birds (2), cats (3), deer (4), dogs (5), frogs (6), horses (7), ships (8), and trucks (9).

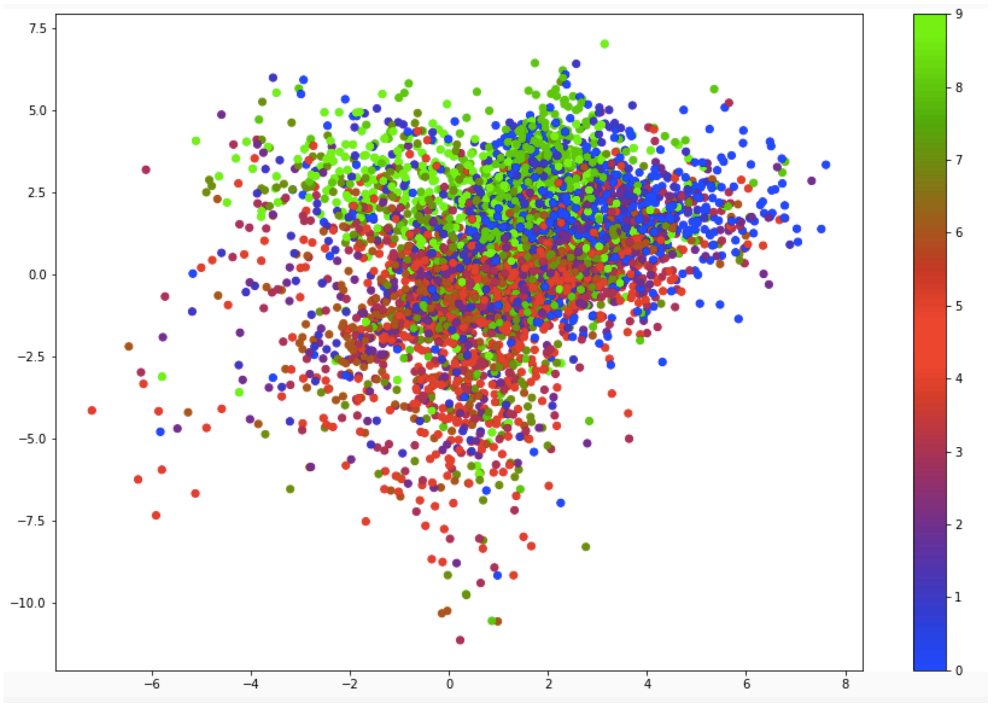

Figure 4: CIFAR-10 latent space distribution for test data with the deep convolutional VAE, with the same color to object correspondence described in figure 3.

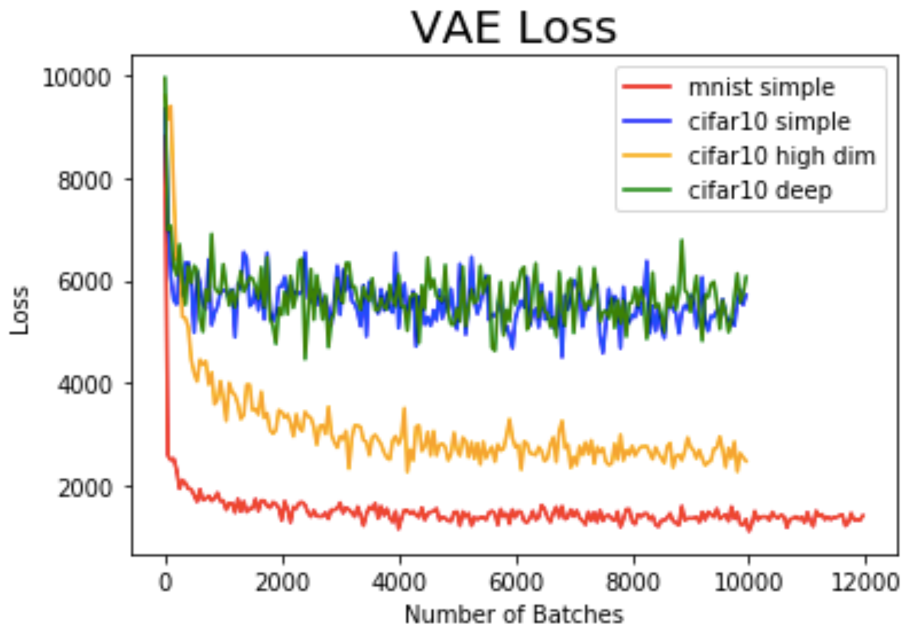

Figure 5: Training loss for each of the four variations: MNIST simple network, CIFAR-10 simple network, CIFAR-10 high dimensional encoding network, CIFAR-10 deep network.

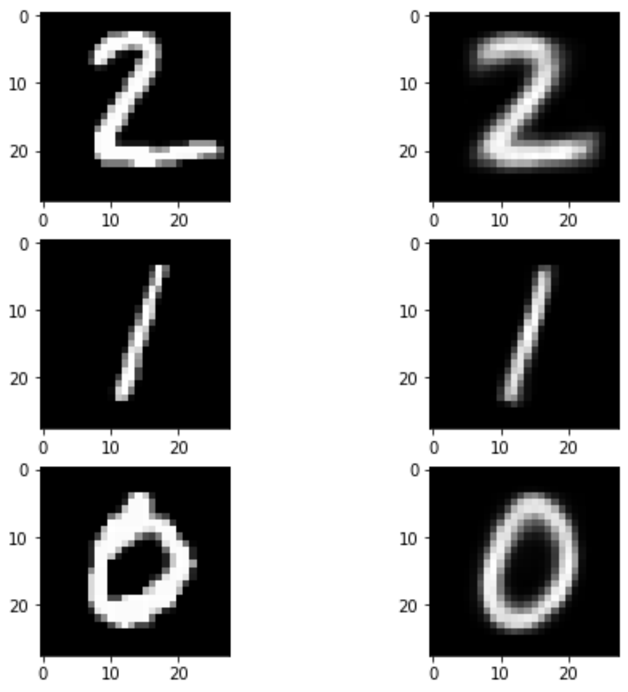

Figure 6: Examples of MNIST test images (left) and their reconstructions with the simple convolutional VAE (right).

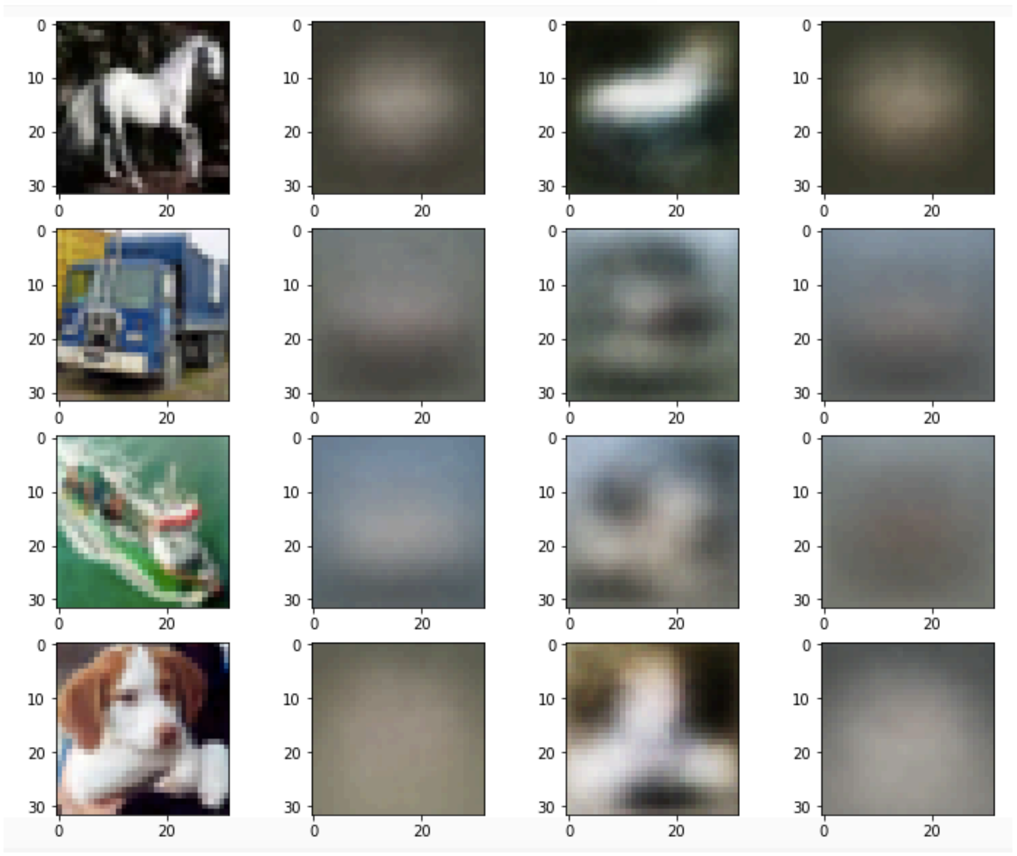

Figure 7: CIFAR-10 example test images (first column) with reconstructions for the simple (second), high dimensional (third), and deep networks (fourth).

Figure 2 shows clearly distinct latent space distributions for most of the digits in the MNIST dataset. As anticipated, 1's, 2's, and 0's all have very different representations as they have distinct shapes. Additionally, 9's and 4's have very similar distributions because these digits share similar features. However, when this same simple network was used to try and learn a latent space manifold for the CIFAR-10 data, the distributions were far less clear. Figure 3 shows non-distinct distributions, we can only see that there is some separation of the vehicles (airplanes, cars, boats, and trucks) from the animals (birds, cats, deer, dogs, frogs, and horses). This makes sense as cars and trucks for example have very similar features (wheels, windows, etc.) and many animals have similar features as well (legs, heads, etc.). However, the images are all very different, compared to MNIST, so the network has more trouble learning clear representations across the entire dataset. The deep network representation in figure 4 has slightly improved separation in the latent distributions, due to the increased network complexity and the decreased weight on the Kullback-Leibler divergence (0.5 weight vs 1.0 weight in the simple network). Again, we see separation between vehicles and animals; particularly we the distributions for the car and truck have distinct distributions from those of the deer and dogs. However, they are not as separable as the latent distributions for the MNIST digits, again because the images are more complex and trying to be encoded down to only two dimensions. The network has trouble learning simple representations of the complex and non-standard images.

We can see the different reconstruction capabilities for each of our networks as well. In figure 6, we see very accurate reconstructions of the MNIST digits with the simple convolutional variational autoencoder. However, in figure 7, we see the limits of reconstructing the more complex data. The second column displays the reconstructions for the CIFAR-10 with the same simple network. The image is extremely blurry with no clear features resembling any of the test images, besides some simple coloring. The third column shows the high dimensional encoding reconstructions. Clearly by increasing the latent space dimensionality, we lose less data and improve the reconstruction. They objects are still not distinguishable, but the color contrast is much more vivid and we can begin to identify some shapes. Much longer training (several hundred epochs) could possibly lead to even more accurate reconstructions. Finally, the fourth column shows the reconstructions with the deeper network. They are slightly better than the simple network, with sharper coloring, but are still very blurry and indistinct. Generally, all of these networks do much worse reconstruction the CIFAR-10 data compared to the MNIST data.

The ability of these networks to learn the MNIST data compared to the CIFAR-10 can be seen in the loss functions in figure 5. I plotted the loss values for each of the four networks by the number of batches performed for training. MNIST clearly has a very quick learning capability and the loss function flattens after roughly 4,000 batch iterations. The two-dimensional encodings for CIFAR-10 (both simple and deep networks) have loss functions that plateau at a much higher value, but have larger oscillation. This makes sense given we have such poor reconstruction, so the reconstruction loss dominates the overall loss function. When we increase the encoding dimensionality and get improved reconstruction, we see the loss function is minimized much more. After 10 epochs it looks that the loss function may still be decreasing, which gives evidence to my earlier idea that training for several hundred epochs could lead to even better reconstructions.