HW4: Latent Variables

Jonah Philion
jonahphilion@college.harvard.edu
github

March 30, 2018

1 Introduction

In this problem set, we attempt to generate reasonable looking MNIST images using two models. With the VAE we model $p(X|z) = \text{Bin}(X|f(z))$ and $p(z|X) = N(z|\mu(X), \Sigma(X))$. The functions μ, Σ, f are parameterized as MLPs. The GAN trains a generator G to generate images into a distribution which is indistinguishable from the distribution of the data D.

2 VAE

Data during training is a collection of handwritten digits $X = \{x_i\}$, where $x_i \in \{0,1\}^{24\times 24}$. The encoder predicts a $\mu \in \mathbb{R}^2$ and $\Sigma \in \mathbb{R}^2$ for every input. We use the re-parameterization trick from the original VAE paper to allow gradients to flow through decoder to encoder.
The encoder used is an MLP with ReLU activation $24 \times 24 \to 50 \to 50 \to 4$ where the first two outputs are interpreted as μ and the second two is the log of the diagonal of the covariance Σ. The decoder is an MLP with ReLU activation $2 \to 50 \to 50 \to 28 \times 28$.
Reconstruction loss is measured with average binary cross entropy between the output of the decoder and the input image x_i. The KL for the case $KL(p||q)$ where q is spherical and p is diagonal is

$$KL = \frac{1}{2}(\Sigma.\text{sum}() + \mu.\text{pow}(2).\text{sum}() - 2 - \Sigma.\log().\text{sum}())$$

The reconstruction loss plus KL loss is equivalent to equation (10) in Kingma. The loss is averaged over the number of images in a batch. The sum of losses $XENT + KL$ is minimized with Adam of 0.001 learning rate over the course of 40 epochs. Both losses are tracked over the course of training on the training set and validation set as shown in Figure 1.
Interpolation is shown in Figure 4. The embedded space is displayed in Figure 2. Examples of the decoder output on a grid of latent variables is shown in Figure 4.

3 GAN

To train our gan we consult Goodfellow's original paper and the Tutorial "How to train your GAN". Small latent dimensions are difficult to train so we use $h = 150$.
The GAN decoder is identical to the decoder used in the VAE except ReLU is replaced with

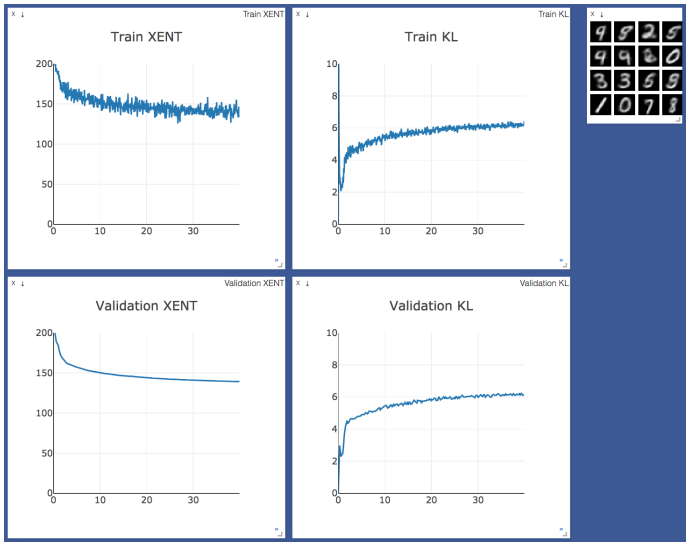

Figure 1: Visualization of training and validation loss for the VAE is handled by visdom. The x-axis is number of epochs through the training set in all cases. Example visualizations of the decoder's output in the top right are on random samples from z which are sampled throughout training.

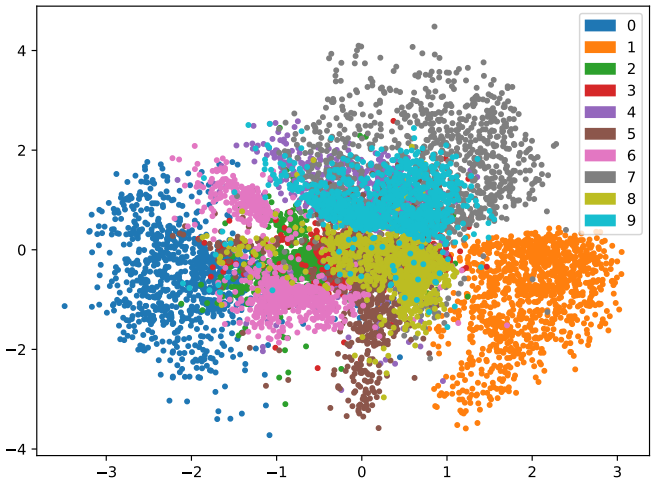

Figure 2: The VAE model learns to separate the different digits without ever seeing a label. After training, on every test sample, we run the encoder and collect all the μ and ground truth labels. The μ are plotted and color coded by their ground truth label. The μ all together are roughly gaussian about the origin as enforced by the KL loss.

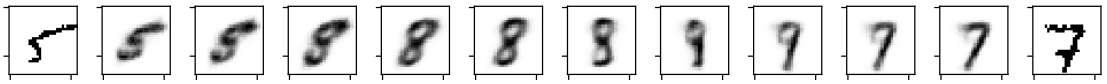

Figure 3: Two images from the test set are displayed on the far left and far right. We run the VAE encoder on both images, then decode 10 interpolations between the latent vectors. To smoothly move from an image of a 5 to an image of a 7, the model transitions to an 8 and then to a 9 and finally a 7. This trajectory is supported by Figure 2.

Figure 4: 15 latent vectors on a grid between -2 and 2 are decoded by the VAE into images. There is a sense in which the latent variable plotted on the y axis encodes the angle of the digit.

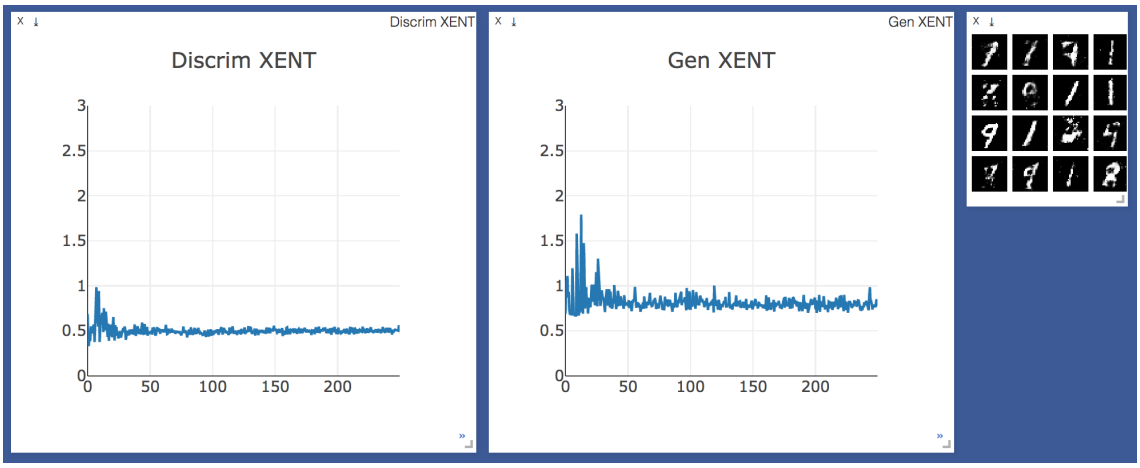

Figure 5: the leftmost image is the cross entropy of the discriminator on 100 data and 100 generated images during training of the GAN. The middle image is the cross entropy of the discriminator on 100 generated images where ground truth is defined as being drawn from the data. The rightmost images are example decoded images which are sampled every epoch.

LeakyReLU and the final layer is tanh instead of linear. The discriminator has identical architecture to the encoder the VAE encoder except the discriminator has final dimension 1.
We train for 250 epochs with an Adam optimizer for both the discriminator and generator of learning rate .001. At each step, the 100 images in a batch are normalized to fall within -1 to 1, we sample 100 latent vectors from a spherical gaussian of dimension h, and we decode these into images. The discriminator is then updated to differentiate between the data images and the decoded images. We next resample 100 latent vectors, apply the decoder, and update the generator to fool the discriminator into classifying the images as drawn from the data.
Noise Unif(0,.2) is added to every ground truth label. The discriminator loss and generator loss are tracked over the course of training and displayed in Figure 5. 36 random example decodings are displayed in Figure 6. A linear interpolation of two random latent vectors is shown in Figure 7.

4 CNN VAE

As an extension, we use a fully convolutional neural network as an autoencoder. The encoder is composed of leaky relus and batchnorms. The decoder is composed of several deconvolutions of either kernel 2, stride 2, and padding 0/1 depending on what worked to get the correct final dimensions. The network is trained in the exact same way as the original autoencoder of fully connected layers.
The same visualizations that were done for the original autoencoder are done for the convolutional one. By eye, the convolutional images look about identical to the original ones. The loss is shown in Figure 8. The labeled scatter plot is shown in 9. Figure 10 shows an example linear interpolation. Figure 11 shows the decoder evaluated on a grid of latent values.

Figure 6: 36 latent vectors are randomly sampled from the GAN space and the output of the GAN decoder on these vectors is plotted in no particular order. The images aren't as smooth as the VAE images but they are also less fuzzy which is a plus.

Figure 7: Two random latent vectors from the GAN space are sampled and the decoding of linearly interpolated latent vectors in between the two are decoded. We can see that the latent space is at least somewhat smooth.

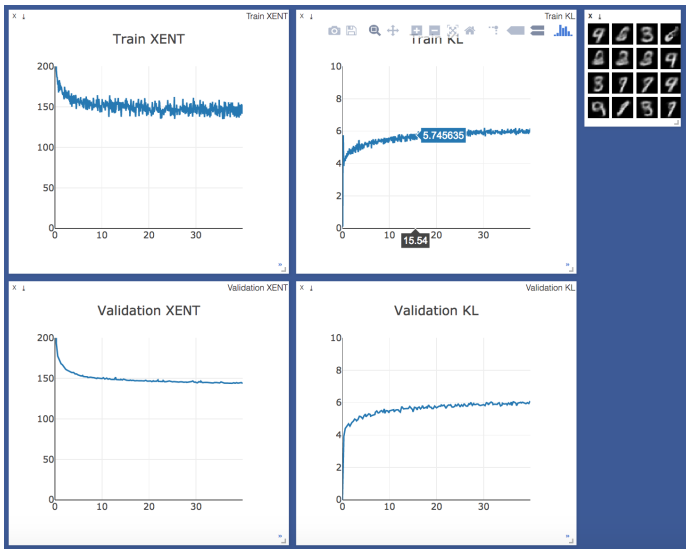

Figure 8: Visualization of training and validation loss for the Convolutional VAE is handled by visdom. The x-axis is number of epochs through the training set in all cases. Example visualizations of the decoder's output in the top right are on random samples from z which are sampled throughout training.

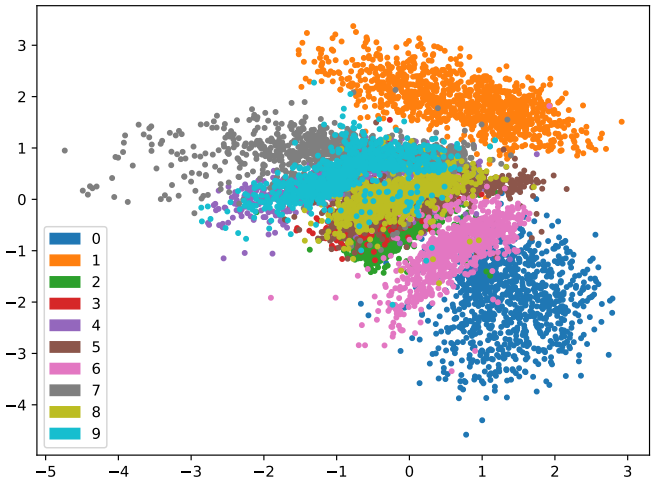

Figure 9: The Convolutional VAE model learns to separate the different digits without ever seeing a label. After training, on every test sample, we run the encoder and collect all the μ and ground truth labels. The μ are plotted and color coded by their ground truth label. The μ all together are roughly gaussian about the origin as enforced by the KL loss.

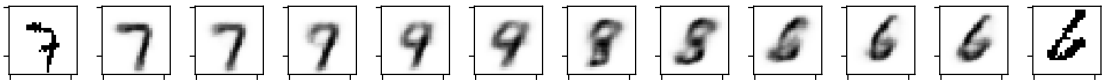

Figure 10: Two images from the test set are displayed on the far left and far right. We run the Convolutional VAE encoder on both images, then decode 10 interpolations between the latent vectors.

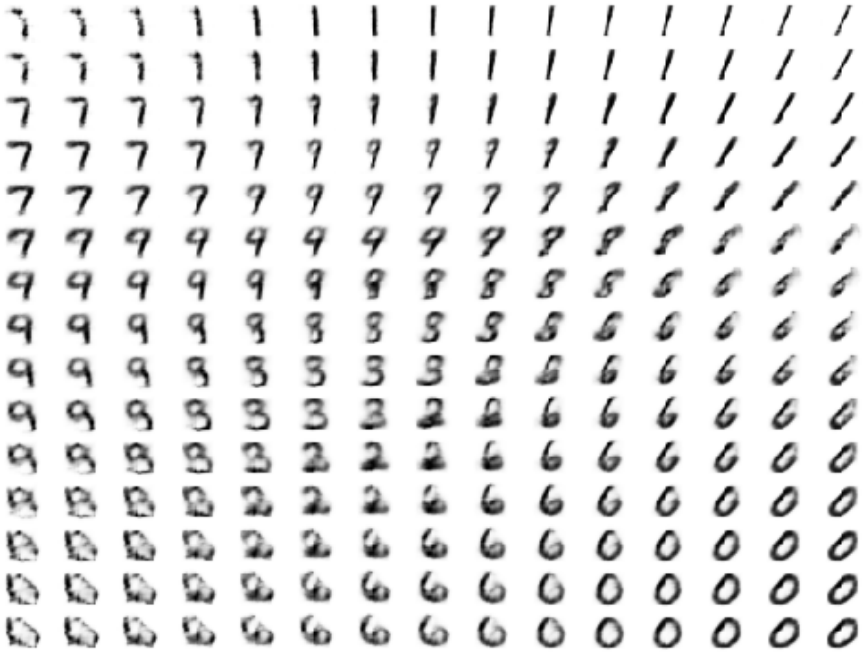

Figure 11: 15 latent vectors from the Convolutional VAE latent space on a grid between -2 and 2 are decoded by the Convolutional VAE into images. This image looks similar to the standard VAE picture.