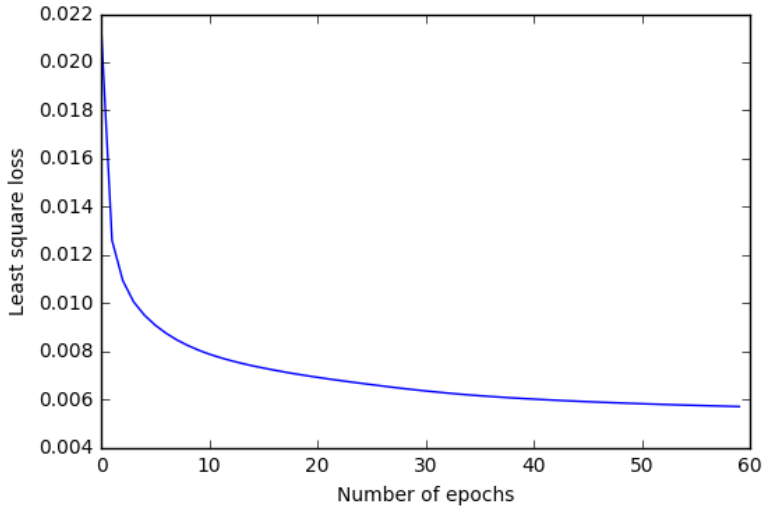
Deep Learning (Homework #4)

Due date: 6/2

1. Autoencoder

An autoencoder is a neural network (NN) that is trained to reconstruct an input vector in output layer of NN. The network may be viewed as two parts: an encoder function $\boldsymbol{h} = f(\boldsymbol{x})$ and a decoder that produces a reconstruction $\boldsymbol{r} = g(\boldsymbol{h})$. In this problem, you should implement a CNN (convolutional neural network) autoencoder on MNIST-M dataset. MNIST-M was artificially created by using each MNIST digit as a binary mask and changing the colors of a background image. The background images are random crops and uniformly sampled from the Berkeley Segmentation Data Set (BSDS500). The given file MNIST_M.npy contains the train set, validation set and test set. You can load data by read_MNIST_M.py file. Please use the least square loss function as your objective function.

(a) Show the reconstruction loss in your report, e.g.



(b) Plot 10 random samples of reconstruction image of test data and original image together, e.g.

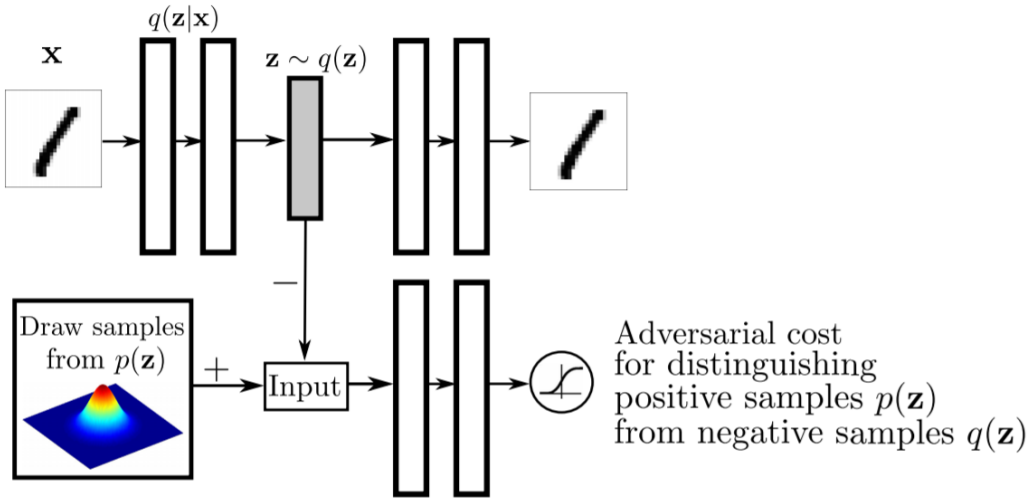
Note: you can use any function in Tensorflow, e.g. tf.nn.con2d for convolution layer and tf.nn.conv2d_transpose for deconvolution.

2. Adversarial Autoencoder

The Generative Adversarial Networks (GANs) establishes a min-max adversarial game between two players – a generative model *G*, and a discriminative model *D*. The discriminator is a neural network that computes the probability that is sampled from data distribution or generative model. In contrast, the generative model is trained to maximally confuse the discriminator into believing that samples it generates come from the data distribution. You can see more details in this paper. The Adversarial Autoencoder (AAE) is an autoencoder that is regularized by matching the aggregated posterior of $q(\mathbf{z})$ on the latent code (hidden unit) to an arbitrary distribution $p(\mathbf{z})$. Let $p_{data}(\mathbf{x})$ be the data distribution and $q(\mathbf{z}|\mathbf{x})$ be an encoding distribution. The aggregated posterior distribution is defined as follows:

$$q(\mathbf{z}) = \int_{\mathbf{x}} q(\mathbf{z}|\mathbf{x}) p_{data}(\mathbf{x}) d\mathbf{x}$$

To implement AAE, an adversarial network is added on top of the hidden code vector of the autoencoder as illustrated.



The training stages of AAE consist of three parts

(a) Train an autoencoder to minimize the reconstruction loss. The autoencoder training objective is defined as

$$\mathcal{L}_{\text{recons}} = \sum_{n=1}^{N} \| \mathbf{x}_n - \hat{\mathbf{x}} \|$$

(b) Train a discriminator to distinguish true samples from fake samples generated by the generator. The discriminator training objective is defined as

$$\mathcal{L}_D = -\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log D(\mathbf{z})] - \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log(1 - D(\text{Enc}(\mathbf{x})))]$$

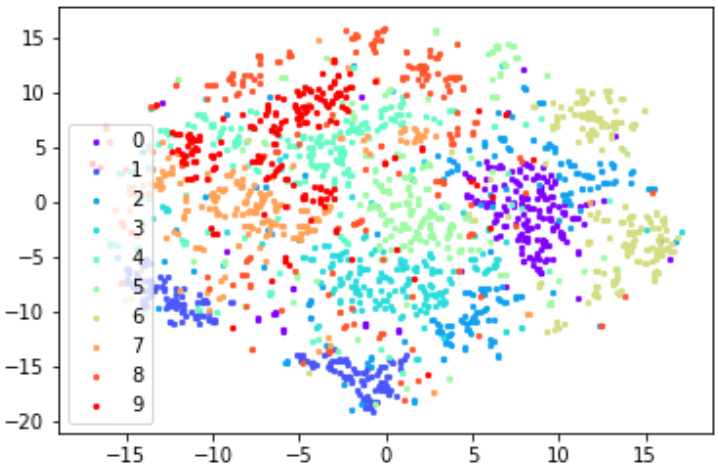
(c) Train an encoder so as to fool the discriminator with its generated samples. The encoder training objective is defined as

$$\mathcal{L}_{\text{Enc}} = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log(1 - D(\text{Enc}(\mathbf{x})))]$$

The details of AAE can be referred to this paper.

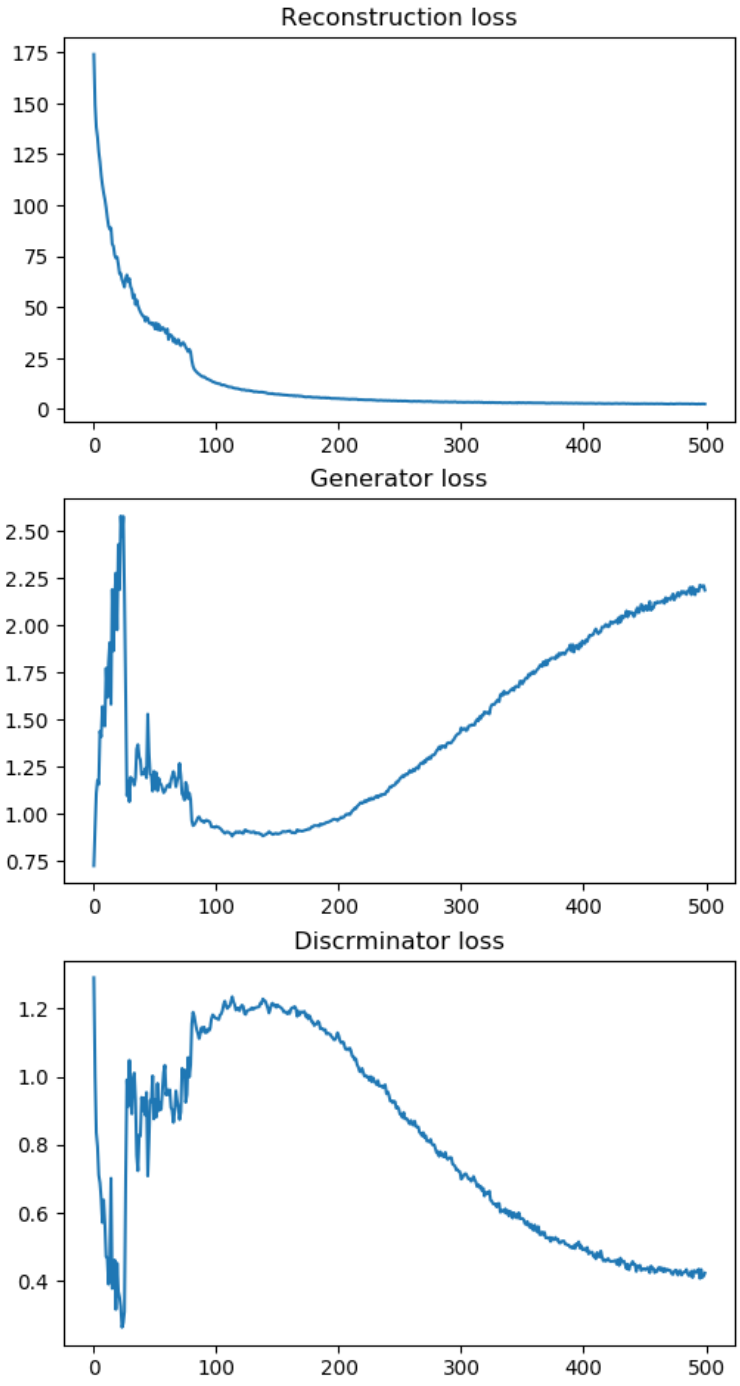
In this problem, you should implement an AAE for dimension reduction by using dataset (data.npy). The prior space is standard normal distribution and its dimension is 100-dimension. Both of autoencoder and discriminator are constructed by multilayer perceptron.

(a) Plot the encoding of training data (Use the *t*-SNE to transform the original feature vector into a two-dimensional space.)



(b) Show the reconstruction loss, adversarial loss of generator and adversarial

loss of discriminator in the training time.



(c) Explore the latent space and reconstruct the data through the decoder. (The output of decoder is 768-dimension, you should reshape the size to 28 x 28)