ECE 685D HW5

1 Problem 1: GAN (35 pts)

In this question, you will implement a GAN for generating MNIST samples.

1.1 Implement a Deep Convolutional GAN (DCGAN) (20pts)

The architecture of the DCGAN, including both the generator and discriminator, is illustrated below:

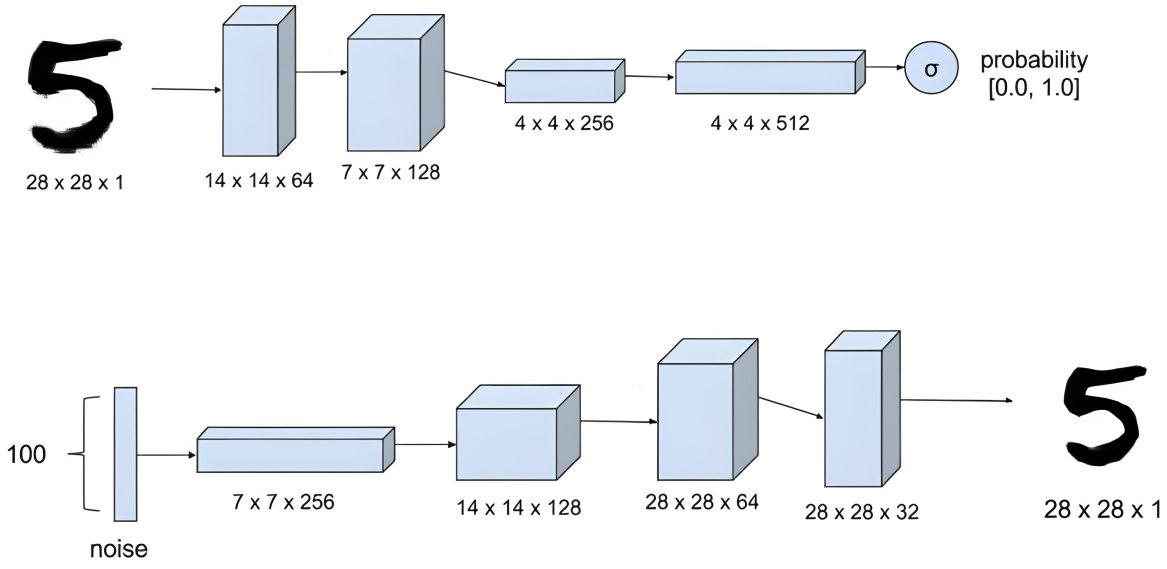

Figure 1: Model Architecture of the Discriminator and Generator

Assume the noise $z \sim N(0, I)$. Use ONLY the TRAIN SPLIT from the MNIST dataset to train your GAN with the min-max loss, defined as follows:

$$\ell = E_x[\log(D(x))] + E_z[\log(1 - D(G(z)))]$$

In this equation, the first term represents is only associated with the discriminator, while the second term affect both the generator and discriminator. The generator will try to minimize the function while the discriminator tries to maximize it.

You are free to choose any activation function and may incorporate Dropout and/or Batch Normalization between layers, as these techniques can theoretically enhance convergence. Additionally, you can normalize your data to ensure pixel intensity values fall within either the range [0, 1] or [-1, 1].

After training the model, utilize the generator to produce 12 new samples.

1.2 GAN as a pre-training framework (15pts)

In this section, you will investigate how GANs can assist in developing a discriminative model. Begin by removing the final linear layer of your discriminator, which is responsible for predicting whether an

image is real or fake. Next, use the remaining components to extract features from **10%** of the training set images. With these extracted features, train a single linear layer to classify the images into categories 0 through 9, based on the latent representations obtained. Then, evaluate the classification accuracy of this model on the test set. Report both the training and testing performance and discuss your results.

2 Problem 2: Gaussian-Bernoulli Restricted Boltzmann Machines (30 pts)

Let the energy function of Gaussian-Bernoulli RBM take the form:

$$E(v,h;\theta) = -\left(\sum_i \sum_j W_{ij} h_j \frac{v_i}{\sigma_i} - \sum_i \frac{(v_i - b_i)^2}{2\sigma_i^2} + \sum_j \alpha_j h_j\right).$$

2.1 Derivations (15pts)

Let Z be the normalization factor, using the definition of $p(\boldsymbol{v}, \boldsymbol{h}) = \exp(-E(\boldsymbol{v}, \boldsymbol{h}))/Z$, derive the mathematical expression of $p(v_i = x|\boldsymbol{h})$ and $p(h_j = 1|\boldsymbol{v})$. You may leave the expression of $p(v_i = x|\boldsymbol{h})$ in an integral form once you cancel all terms involving $\sum_j \alpha_j h_j$.

Hint: you can apply Bayes rule to get $p(v_i, |h) = \frac{p(v_i, h)}{p(h)}, \frac{p(v_i, h)}{p(h)}$, and then simplify your expression.

2.2 GB-RBM on KMNIST (15pts)

Train a Gaussian-Bernoulli RBM over **KMNIST** (with default train-test split) using CD-5 (see the lecture notes for details). Setting lr = 0.001, batch size = 128, weight decay = 0 with Adam optimizer as your hyper-parameter configuration. Then, train three versions of RBM with $M = \{16, 64, 256\}$, where M is the dimension of the hidden weights $\mathbf{W}$, for 25 epochs.

On the test set for RBM with M=16, you should expect an output looking like this:

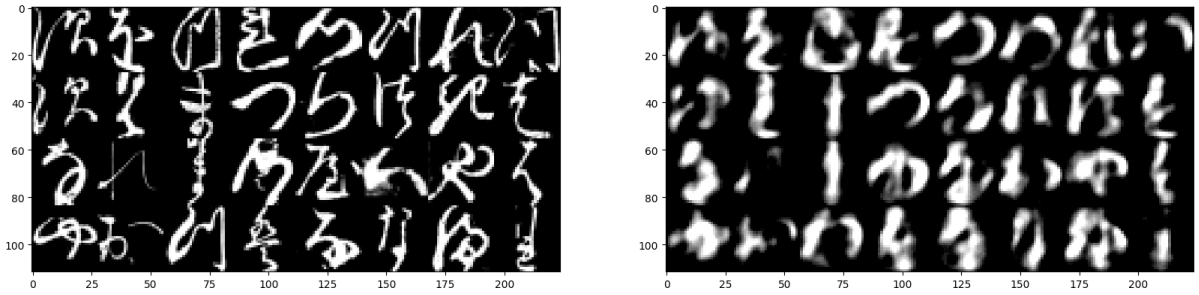

Figure 2: A sample output for M = 16 for reconstructed test set (right), you should expect visibly better results for M=256

After your model training, make a plot as in Fig. 2 for both images in the train set and test set. Then, compute and report the mean squared reconstruction error (MSE) for inferences on both datasets. A template containing the RBM has been provided to you that you can simply fill the missing lines. However, also feel free to make any modification to other parts of the template.

3 Problem 3: Two Variational Autoencoders (45 points)

Recall that for a given data point $\mathbf{x} \in \mathbb{R}^D$, the evidence lower bound (ELBO) is given by:

$$ELBO = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[logp_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})] \tag{1}$$

Here $\mathbf{z} \in \mathbb{R}^M$, $M < D$ denotes the latent variable, $p(\mathbf{z})$ is the prior distribution over $\mathbf{z}$, $q_\phi(\mathbf{z}|\mathbf{x})$ is the variational posterior distribution (i.e., encoder output), and $p_\theta(\mathbf{x}|\mathbf{z})$ is the likelihood function (i.e., decoder output).D_{KL} denotes the Kullback-Liebler divergence and $\mathbb{E}$ is the expectation.

In a standard VAE, the variational posterior is assumed to belong to a class of normal distributions, i.e., $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mu_x, \sigma_x)$, whereas the prior is assumed to be standard normal, that is $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. As a further simplification, the covariance matrix of the variational posterior can be assumed to be diagonal.

In conditional VAE, the generative model is conditioned on a data category $\mathbf{c}$ (or any variable we want to control). Here, we want to learn parameters (θ, ϕ) such that $p_\theta(\mathbf{x}|\mathbf{c})$ generated by the generative model approximates $p(\mathbf{x}|\mathbf{c})$ for each category. The decoder output, $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{c})$ approximates $p_\theta(\mathbf{z}|\mathbf{x}, \mathbf{c})$ for all x, c. You will implement a standard VAE and a C-VAE for **Fashion MNIST** in this problem.

3.1 Vanilla VAE (20 pts)

Train a standard VAE over the Fashion MNIST dataset. Use the standard training and testing split available in PyTorch. A template with both the model architecture and the hyperparameters has been provided to you and you should not change them if they are explicitly defined. Use 20% of the training dataset as a validation set to tune the hyperparameters of your model. Save your model weights for later use.

3.2 C-VAE (10 pts)

Train a class-conditional VAE with the same configuration as in 3.1. Also save your model weights for later use.

3.3 Manifold Comparison (15 pts)

Now you will visualize the latent representation learned by your VAE and c-VAE. You will perform the manifold visualization with the sklearn.manifold.TSNE package. Specifically, you will

1. pass the samples in the testing split through the encoder of the VAE and c-VAE
2. keep the μ obtained from VAE and c-VAE
3. use sklearn.manifold.TSNE to map the μ to 2D manifold
4. plot the 2D manifold for VAE and c-VAE and color the data points by class

Compare the data distribution on the two manifolds and described what you observed. Make a hypothesis on why you observed this phenomenon (but you don't have to prove it).