Assignment 5

Generative Models (VAE and Diffusion Models)

CMPUT 328 - Fall 2023

1 Assignment Description

The main objective in this assignment is to implement and evaluate two of the most popular generative models, namely **Variational Auto-Encoders (VAE)** and **Diffusion Models**. Our goal is to implement each of these models on the FashionMNIST dataset and see how such models can generate new images. However, instead of simply training the models on the whole dataset, we would like to be able to tell the model from which class it should generate samples. Hence, we are going to implement *class-conditional* VAEs and Diffusion Models.

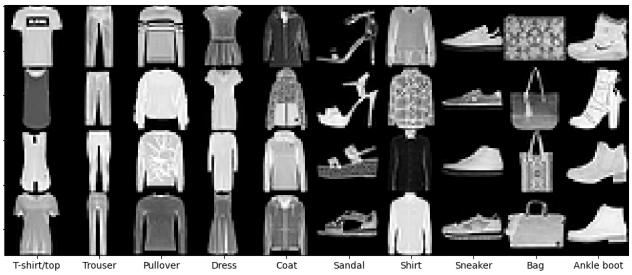

Figure 1: Sample images from the FashionMNIST dataset

Note: Please the watch the video provided for this assignment for better understanding the tasks and objectives.

2 What You Need to Do

For this assignment, 5 files are given to you:

- A5_vae_submission.py
- A5_vae_helper.ipynb
- A5_diffusion_submission.py
- A5_diffusion_helper.ipynb
- classifier.pt

You only need to submit "A5_vae_submission.py", "A5_diffusion_submission.py", and weights of your networks ("vae.pt", "diffusion.pt").

2.1 Task 1: Conditional VAE (40%)

2.1.1 A5_vae_submission.py

In this file there is a skeleton of a VAE class which you are required to complete.

1. For the VAE you need to implement the following components as specified in the code file: Encoder, mu net (for estimating the mean), logvar net (for estimating the log-variance), class embedding module (for properly embedding the labels), and decoder (for reconstructing the samples).
2. The forward function of the VAE class must receive the batch of images and their labels, and return the reconstructed image, estimated mean (output of mu net), and the estimated logvar (output of the logvar net).
3. You need to fill in the "reparameterize" method of the class given mu and logvar vectors (as provided in the code), and implement the reparameterization trick to sample from a Gaussian distribution with mean "mu", and log-variance "logvar".
4. You need to fill in the "kl_loss" method of the class given mu and logvar vectors, and compute the Kullback-Leibler (KL) divergence between the Gaussian distribution with mean "mu" and log-variance "logvar" and the standard Gaussian distribution $\mathcal{N}(0, I)$. Recall that if the the mean and variance of the a Gaussian distribution are μ and σ^2, respectively, the KL divergence with the standard Gaussian can be simply calculated as

$$KL(\mathcal{N}(\mu, \sigma^2) \| \mathcal{N}(0, I)) = \frac{1}{2} \sum_{i=1}^{n} (\sigma_i^2 + \mu_i^2 - 1 - \ln(\sigma_i^2)) \tag{1}$$

5. You need to fill in the "get_loss" method of the class given the input batch of images and their labels. In this method you need to find the estimated mu, estimated logvar, and the reconstructed image, find the KL divergence using mu and logvar and find the reconstruction loss between the input image and the reconstructed image. Usually for the reconstruction loss the Binary Cross-Entropy loss is used.
6. Most importantly, you need to fill in the "generate_sample" method of the class, which receives the number of images to be generated along with their labels, and generates new samples from the VAE. Basically, you need to sample from standard Gaussian noise, combine it with the class embedding and pass it to the networks decoder to generate new images.
7. Please do not rename the VAE class and its methods. You can add as many extra functions/classes as you need in this file. You can change the arguments passed to the "__init__" method of the class based on your needs.
8. Finally, you need to complete the "load_vae_and_generate" function at the bottom of the file, which merely requires you to define your VAE.

2.1.2 A5_vae_helper.ipynb

This file is provided to you so you can train and validate your model more simply. Once you are done with your implementation of the VAE class you can start running the blocks of this file to train your model, save the weights of your model, and generate new samples. You only need to specify some hyperparameters such as batch size, optimizer, learning rate, and epochs, and of course your model.

There is also a brief description of the VAEs at the beginning of this file.

2.2 Task 2: Conditional Diffusion Model (60%)

2.2.1 A5_diffusion_submission.py

In this file there are skeletons of a VarianceScheduler class, NoiseEstimatingNet class, and the DiffusionModel class, which you are required to complete.

1. For the VarianceScheduler class you need to store the statistical variables required for making the images noisy and sampling from the diffusion model, such as β_t, α_t, and $\bar{\alpha}_t$. You also need to complete the "add_noise" method which receives a batch of images and a batch of timesteps and computes the noisy version of the images based on the timesteps.

2. You need to complete the NoiseEstimatingNet class, which is supposed to be a neural network (preferably a UNet) which receives the noisy version of the image, the timestep, and the label of the image, and estimates the amount of noise added to the image. You are encouraged to look at the network architectures you have seen in the notebooks provided to you on eClass resources. Note that you can add extra functions and classes (e.g., for time embedding module) in this file.

3. You need to complete the "DiffusionModel" class. The forward method of the class receives a batch of input images and their labels, randomly adds noise to the images, estimates the noise using NoiseEstimating network, and finally computes the loss between the ground truth noise and the estimated noise. The forward method outputs the loss.

4. Most importantly, you need to fill in the "generate_sample" method of the DiffusionModel class which receives the number of images to be generated along with their labels, and generates new samples using the diffusion model.

5. You need to fill in the "get_loss" method of the class given the input batch of images and their labels. In this method you need to find the estimated mu, estimated logvar, and the reconstructed image, find the KL divergence using mu and logvar and find the reconstruction loss between the input image and the reconstructed image. Usually for the reconstruction loss the Binary Cross-Entropy loss is used.

6. Most importantly, you need to fill in the "generate_sample" method of the class, which receives the number of images to be generated along with their labels, and generates new samples from the VAE. Basically, you need to sample from standard Gaussian noise, combine it with the class embedding and pass it to the networks decoder to generate new images.

7. Please do not rename the VarianceScheduler, NoiseEstimatingNet, and DiffusionModel classes and their methods. You can add as many extra functions/classes as you need in this file.

8. Finally, you need to complete the "load_diffusion_and_generate" function at the bottom of the file, which merely requires you to define your VarianceScheduler and NoiseEstimatingNet.

2.2.2 A5_diffusion_helper.ipynb

This file is provided to you so you can train and validate your model more simply. Once you are done with your implementation of the VarianceScheduler, NoiseEstimatingNet, and DiffusionModel classes you can start running the blocks of this file to train your model, save the weights of your model, and generate new samples. You only need to specify some hyperparameters such as batch size, optimizer, learning rate, and epochs, and of course your model.

There is also a brief description of the Diffusion Models at the beginning of this file, including how to make the noisy images, and how to sample from the diffusion model, which could be helpful.

3 Deliverables

- The correct (working) implementation of the explained modules in the previous section.
- For the diffusion model use a number of diffusion steps less than or equal to 1000 for a roughly fast image generation.
- We verify the quality of the images generated by your models by using a classifier trained over the dataset. This classifier is provided to you in the helper notebooks, and without changing the code you can run the corresponding blocks to load the classifier and apply it to your generated images.
- For the VAE model, a final accuracy of $\geq 65\%$ gets a full mark and an accuracy of $< 55\%$ gets no mark. You mark will linearly vary for any accuracy in between.
- For the Diffusion Model, a final accuracy of $\geq 60\%$ gets a full mark and an accuracy of $< 50\%$ gets no mark. You mark will linearly vary for any accuracy in between.

In the following you can see some sample outputs of a simple VAE and a simple DiffusionModel trained on the FashionMNIST.

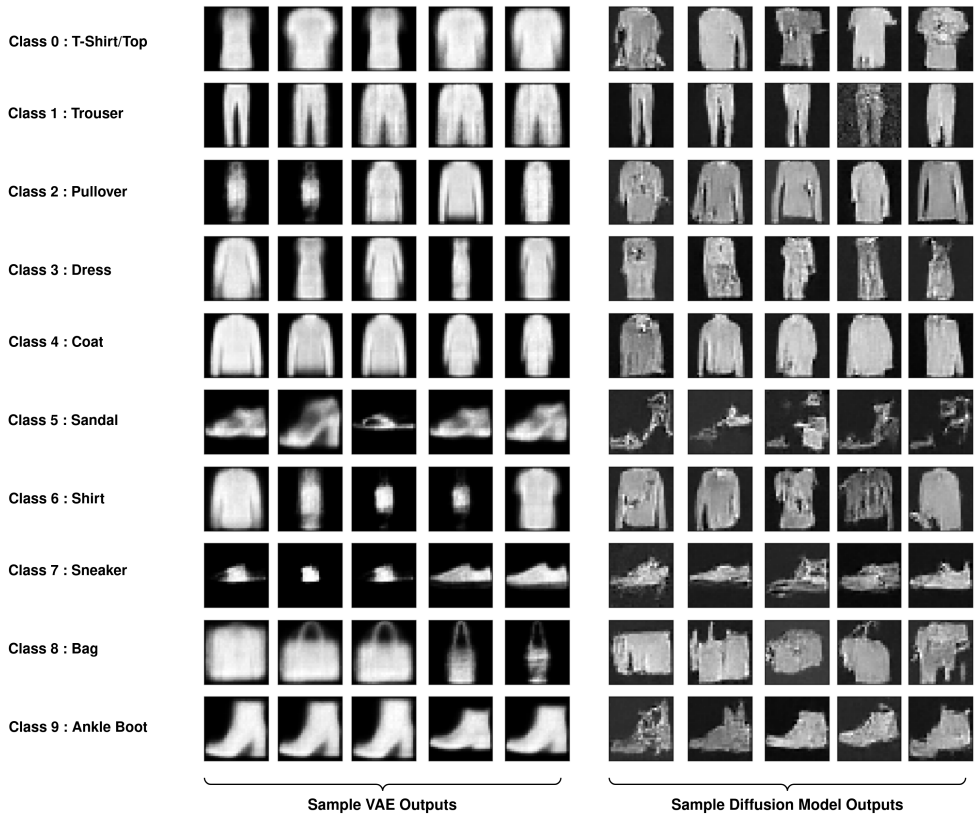

Figure 2: Sample images generated by the VAE and Diffusion Model