Homework assignment 3
Due: Thu., March 24, 2022, 11:59pm

Problem 1. (20 points) This week, we talk about generative models in class. This assignment particularly focuses on variational autoencoder (VAE). In VAE, we constrain the encoder network to produce latent vectors that follow the unit normal distribution. When generating new samples, we simply sample some values from the latent distributions, feed them into the decoder network; the decoder will return completely new data that appear just look the training data our networks haven been trained with.

In this assignment, we implement a simple VAE model. The objective is to generate new digit samples that are look like those in the MNIST data set. Please load the Assignment3_VAE.ipynb to Google Colab (you need to use your CCID to login) and config your GPU environment by "Runtime->change runtime type->GPU". You should be able to train the VAE model successfully (codes in Section 1). Please read the code and answer following questions.

a) (2 points) Is the VAE model has convolutional layers? Please sketch the VAE model.
b) (2 points) After training the model, please run the codes in Section 2. Your digit samples should be clustered and centered about (0,0) in theory. Please attach your 2-D plot of the digit classes in the latent space and explain why the samples should be centered at (0,0) theoretically.
c) (2 point) Please find the cluster region for the VAE model in your 2-D plot in (b), set the values appropriately at the beginning of Section 3, and run the code. Please attach the image displaying a grid of sampled digit images generated by different latent variables. What is the output of (0,0) in your model?
d) (3 points) Please implement the convolutional VAE model whose encoder and decoder models are specified in Page 2 and submit your code here. All convolutional kernels are in size of 3x3 and the model uses the Relu function as the activation function. Please carefully design the stride step and padding in your convolutional layers.
e) (2 points) Please repeat b) and c) for your convolutional VAE.
f) (2 points) The original AVE and your convolutional AVE have similar sizes, i.e. the number of trainable parameters in both model being in the scale of sixty-five thousands. Which model has a better performance? Please explain your observation.
g) (4 points) In conventional VAE, we have no control on the data generation. This could be problematic if we want to generate some specific data. For example, it is difficult to generate "3" without printing out the latent code distribution in (c). One solution to this problem is to add a condition to the VAE. That is, the VAE models latent variables and data, both conditioned to some random variables. Such a model is called conditional VAE (CVAE). Let's use the digit category c as the condition in CVAE, i.e. c is taken as another input to both the encoder and decoder. Specifically, in training, the encoder of the CVAE takes an image and its label c as input and the decoder takes the label c and latent code z as the input. During test, the target label and sampled code z are fed to the decoder together for a new image. The objective function of the CVAE is

$$\mathcal{L}(x^i, c^i, \theta, \phi) = E_z\left[log p_\theta\left(x^i \middle| z, c^i\right)\right] + D_{KL}[q_\emptyset(z|x^i, c^i)||p_\theta\left(z\middle|c^i\right)]$$

Please use the convolutional VAE in (d) as your basis and implement a CVAE with image labels as the condition. Please think about a way to concatenate the label c and the encoder/decoder inputs in your CVAE.

h) (3 points) Please repeat b) and c) for your convolutional CVAE and compare its performance with you convolutional VAE.

Architecture detail of the convolutional VAE in (d)

```
Model: "encoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_15 (InputLayer)        [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 14, 14, 32)        320
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 7, 7, 64)          18496
_________________________________________________________________
flatten_7 (Flatten)          (None, 3136)              0
_________________________________________________________________
dense_14 (Dense)             (None, 14)                43918
_________________________________________________________________
z_mean (Dense)               (None, 2)                 30
_________________________________________________________________
z_log_var (Dense)            (None, 2)                 30
_________________________________________________________________
sampling_7 (Sampling)        (None, 2)                 0

=================================================================
Total params: 62,794
Trainable params: 62,794
Non-trainable params: 0


Model: "decoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 2)]               0
_________________________________________________________________
dense_9 (Dense)              (None, 3136)              9408
_________________________________________________________________
reshape_5 (Reshape)          (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_transpose_10 (Conv2DT (None, 14, 14, 64)        36928
_________________________________________________________________
conv2d_transpose_11 (Conv2DT (None, 28, 28, 32)        18464
_________________________________________________________________
conv2d_transpose_12 (Conv2DT (None, 28, 28, 1)         289
=================================================================
Total params: 65,089
Trainable params: 65,089
Non-trainable params: 0
```