Lab Assignment 3: Variational Autoencoders

Due Date: Sunday November 3, at 11:59 PM

Purpose:

The purpose of this Lab assignment is to:

- To get hands-on experience of applying Deep Neural Networks, specifically variational autoencoders

General Instructions:

Be sure to read the following general instructions carefully:

1. This assignment must be completed individually by all students.
2. Only provide the requested screenshots and make sure to have a complete screenshot, partial screenshots will not earn any marks.
3. You will have to provide a **demonstration video for your solution** and upload the video together with the solution on **eCenntenial** through the assignment link. See the **video recording instructions** at the end of this document.
4. In your 5-minute demonstration video you should explain your solution clearly, going over the main code blocks and the purpose of each module/class/method also demoing the execution of exercises #1. YouTube links and links to google drive or any other media are not acceptable, the actual recording must be submitted.
5. Any submission without an accompanying video will lose 70% of the grade.
6. In your analysis report make sure you provide an introduction and clearly state the facts and findings. Any submission missing Analysis report will lose lost 70%.

Submission:

There are three elements to be submitted for this assignment in one zipped folder (All subject to grading as per rubric for this assignment):

1. For each exercise that require code, please create a project folder and include all project python scripts/modules and screenshot of output, as needed. Name all python scripts your firstname_lab3.py. Name the folder "Exercise#X_firstname", where X is the exercise number and firstname is your first name. (In total 1 folders for this assignment).
2. For all questions that require written or graphic response create one "Word document" and indicate the exercise number and then state your response. Name the document

"Written_response_firstname", where firstname is your firstname. (In total on word or pdf document).

3. All submissions need to be accompanied with a recorded demonstration video not to exceed 5 minutes in length, focus on showing the key code functionalities and run the code.

Create on zipped folder containing all of the above, name it lab2assignment_firstname where firstname is your firstname.

Assignment – exercises:

1. **Exercise #1:** Variational Autoencoders (100 marks)

   Requirements:

   a. Get the data:
      1. Import and load the 'fashion_mnist' dataset from TensorFlow. Using 2 dictionaries store the fashion_mnist datasets into *train_firstname* and *test_firstname*, where firstname is your firstname. The first 60,000 data samples will be stored in *train_firstname* directory with keys '*images*' and '*labels*', which will contain the images and labels for supervised learning. The next 10,000 data samples will be stored in *test_firstname* directory with keys '*images*' and '*labels*', which will contain the images and labels for supervised learning
         For more info checkout:
         https://keras.io/api/datasets/fashion_mnist/#load_data-function
   b. Data Pre-preprocessing
      1. Normalize the pixal values in the dataset to a range between 0-1. Store result back into *unsupervised_firstname['images']* and *supervised_firstname['images']*
      2. Display (print) the shape of the *train_firstname['images'], test_firstname['images'].*
   c. Build Variational Autoencoder with latent dimension size of 2
      1. Implement a customer layer named SampleLayer that extends the tf.keras.layers.layer class. The customer layer will sample the latent space of the encoder for the decoder. For more info checkout:
         https://www.tensorflow.org/tutorials/customization/custom_layers
         i. The call function takes as input the mean and standard deviation as a list. From one of the input use tf.shape to calculate the batch size and dimension of the input. Generate random noise with the

sample dimension as the batch size and dimension of the input from a standard normal distribution using tf.keras.backend.random_normal. Generate samples z using the following formula:

$$z = \mu + \sigma \odot \epsilon$$

where μ and σ are the mean and standard deviation from the input and ϵ is the generated random noise.

2. Use TensorFlow's Model() [For more info, reference: https://www.tensorflow.org/api_docs/python/tf/keras/Model] to build the encoder section of the variational autoencoder with the following architecture:
    i. Input = Size of input image, store the layer as input_img
    ii. Layer 1 = Convolution with 32 kernels with window size 3x3, a 'relu' activation function, and 'same' padding
    iii. Layer 2 = Convolution with 64 kernels with window size 3x3, a 'relu' activation function, 'same' padding, stride of 2x2
    iv. Layer 3 = Convolution with 64 kernels with window size 3x3, a 'relu' activation function, and 'same' padding
    v. Layer 4 = Convolution with 64 kernels with window size 3x3, a 'relu' activation function, and 'same' padding
    vi. Layer 5 = Full connected layer with 32 neurons and 'relu' activation (Note: Input to fully connected layer should be flatten first)
    vii. LatentSpace.A = Full connected layer with 2 neurons, store layer as z_mu*_firstname*
    viii. LatentSpace.B = Full connected layer with 2 neurons, store layer as z_log_sigma*_firstname*
    ix. Output = SampleLayer defined Step C.1, store layer as z*_firstname*
3. Display (print) a summary of the model using summary(). Draw a diagram illustrating the structure of the neural network model, making note of the size of each layer (# of neurons), number of weights in each layer and the unique connection between the latent space layer and the sample output layer.
4. Use TensorFlow's Model() [For more info, reference: https://www.tensorflow.org/api_docs/python/tf/keras/Model] to build the decoder section of the variational autoencoder (store as decoder*_firstname)* with the following architecture:
    i. Input = Size of latent dimension
    ii. Layer 1 = Fully connected layer, the number of neurons should be same as the output shape of Layer 4 in the encoder (i.e. the flatten input dimension for layer 5)

iii. Layer 2 = Use tf.keras.layers.reshape to reshape the tensor as an image. The dimension of the reshape should be the same as Layer 4 in the encoder
iv. Layer 3 = Use tf.keras.layers.Conv2DTranspose to add a transposed convolution layer with 32 kernels with window size 3x3, a 'relu' activation function, 'same' padding, stride of 2x2. For more info, reference: https://keras.io/api/layers/convolution_layers/convolution2d_transpose/
v. Layer 4 = Convolution with 1 kernels with window size 3x3, a sigmoid activation function, and 'same' padding

5. Display (print) a summary of the model using summary(). Draw a diagram illustrating the structure of the neural network model, making note of the size of each layer (# of neurons), number of weights in each layer.
6. Use TensorFlow's Model() [For more info, reference: https://www.tensorflow.org/api_docs/python/tf/keras/Model] to build a variational autoencoders (store model as vae_*firstname*) from the input layer of the encoder in Step C.2.i and output of the decoder(Note: use built model decoder_*firstname from C.4 and* z_*firstname* from Step *C.2 to define output y, which will be the output of the variational autoencoder)*
7. Display (print) a summary of the model using summary(). Draw a diagram illustrating the structure of the neural network model, making note of the size of each layer (# of neurons), number of weights in each layer.

d. Define the KL divergence using the following line, making sure to replace z_mu and z_log_sigma, as defined from Step C.2:

```
kl_loss = -0.5 * tf.reduce_mean(z_mu - tf.square(z_mu) - tf.exp(z_log_sigma) + 1)
```

e. Use model.add_loss() to add the KL loss function defined in Step D. For more info reference: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Layer#add_loss
f. Compile the model with 'adam' optimizer, and 'mean_square_error' loss function
g. Use TensorFlow's fit() and the *train_firstname*['images' dataset to train the VAE with 10 epochs and batch size of 256.

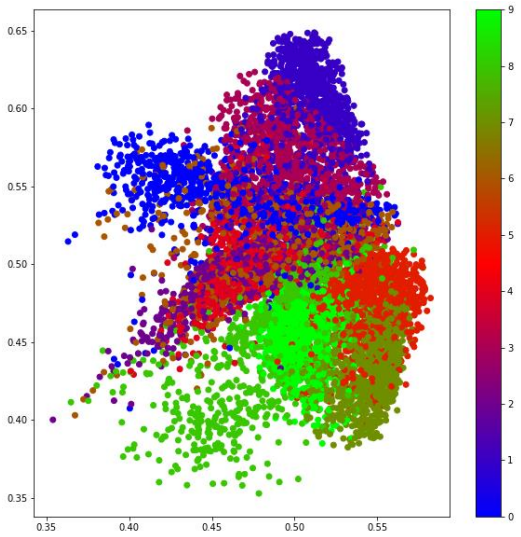
Note: Training a generative model is computationally expensive and can take several minutes to a couple hours to complete training. Therefore, during development you should train the model using a very limited subset of the data. Once you are confident there is no errors in your code, you can run the code and train the model using the full dataset.

h. Review sample code below and generate 10x10 samples from the VAE model using the decoder.

```
import tensorflow_probability as tfp


n = 4
figure_size = 28
norm = tfp.distributions.Normal(0, 1)
grid_x = norm.quantile(np.linspace(0.05, 0.95, n))
grid_y = norm.quantile(np.linspace(0.05, 0.95, n))

figure = np.zeros((figure_size*n, figure_size*n))

for i, yi in enumerate(grid_x):
    for j, xi in enumerate(grid_y):
        z_sample = np.array([[xi, yi]])
        z_sample = np.tile(z_sample, batch_size).reshape(batch_size, 2)

        x_decoded = decoder.predict(z_sample, batch_size=batch_size)
        img = x_decoded[0].reshape(figure_size, figure_size)
        figure[i * figure_size: (i + 1) * figure_size,
               j * figure_size: (j + 1) * figure_size] = img

plt.figure(figsize=(20, 20))
plt.imshow(figure)
plt.show()
```

i. Display (plot) the latent space of z_mu of the test dataset

1. Using TensorFlow's Model() and layers from the encoder (see Step C.2) build a model to generate the latent space for z_mu.
2. Use the model to predict the encoded latent space of the test dataset
3. Use matplotlib.pyplot.scatter to plot the latent space. For more info, reference: https://matplotlib.org/3.5.1/api/_as_gen/matplotlib.pyplot.scatter.html. Your plot should look something like the following.



------ End of Exercises ------

Rubric

Evaluation criteria	**Not acceptable**	**Below Average**	**Average**	**Competent**	**Excellent**
	0% - 24%	**25%-49%**	**50-69%**	**70%-83%**	**84%-100%**
Functionality	Missing all functionalities required	Some requirements are implemented.	Majority of requirements are implemented but some are malfunctioning.	Majority of requirements implemented.	All requirements are implemented Correctly.
Classes	Classes have been created incorrectly or completely missing.	Classes have been defined but have errors. Instances are incorrectly used.	Classes have been defined correctly but instances are used incorrectly or not created at all.	Classes have been defined correctly but some instances are used incorrectly.	Classes are correctly defined and makes use of its own functions which are called somewhere else in the code. Instances have been created and used correctly.
Documentation	No comments explaining code changes.	Minor comments are implemented.	Some code changes are correctly commented.	Majority of code changes are correctly commented.	All code changes are correctly commented.
Design	No adherence to object design principles.	Minor adherence to object design principles.	Some object oriented and modulus design principles are adhered to.	Majority of Object oriented and modulus design principles are adhered to.	Object oriented and modulus design principles are adhered to.
Testing & Evaluation	No evidence of testing and evaluation of the requirements.	Minor evaluation and testing efforts.	Some of the requirements have been tested & evaluated.	Majority of requirements are tested & evaluated.	Realistic evaluation and testing, comparing the solution to the requirements.

Demonstration Video	Very weak no mention of the code changes. Execution of code not demonstrated.	Some parts of the code changes presented. Execution of code partially demonstrated.	All code changes presented but without explanation why. Code demonstrated.	All code changes presented with explanation, exceeding time limit. Code demonstrated.	A comprehensive view of all code changes presented with explanation, within time limit. Code demonstrated.

Demonstration Video Recording

Please record a short video (max 4-5 minutes) to explain/demonstrate your assignment solution. You may use the Windows 10 Game bar to do the recording:

1. Press the Windows key + G at the same time to open the Game Bar dialog.

2. Check the "Yes, this is a game" checkbox to load the Game Bar.

3. Click on the Start Recording button (or Win + Alt + R) to begin capturing the video.

4. Stop the recording by clicking on the red recording bar that will be on the top right of the program window.

(If it disappears on you, press Win + G again to bring the Game Bar back.)

You'll find your recorded video (MP4 file), under the Videos folder in a subfolder called Captures.

Submit the video together with your solution and written response.