CSc 740 Spring 2022
Assignment 3

Due Date: 11:59 pm, May 17, 2022

Read and Follow Assignment Instructions Carefully

1. This is an individual assignment. All work submitted must be your own.

2. First read the entire assignment description to get the big picture; create your own notes about the control flow, expected functionality of the various methods and why you are being asked to implement specific items. To repeat: **First, read the entire assignment completely and think about how the different parts are connected.**

3. You are allowed to use the Scikit-Learn library and standard machine learning and python libraries. You are **Not allowed** to **use any AutoML solutions or packages**.

4. Deliverables: the code and answers should be written in a Jupyter notebook named **'<lastname>_<firstname>_assignment3.ipynb'**. The notebook should also include short write-ups using markdown (3-5 sentences) summarizing results.

5. Make sure you copy each question with the question number as a Markdown Cell in your Jupyter notebook and have the code response right below it. Points will be deducted if it is difficult to locate the question and response.

6. Make sure you comment your code. Points will be deducted if code logic is not apparent.

7. The written sections will be graded on correctness and preciseness while code will be graded on structure, implementation and correctness.

About the assignment: This assignment is intended to build the following skills:

1. Concepts on Dimensionality Reduction
2. Principal Component Analysis (PCA)
3. Autoencoder Neural Networks Model stacking
4. Option / Extra Credit: Neural Network from Scratch

Dimensionality Reduction: This assignment focuses on dimensionality reduction and understanding how PCA and Autoencoders reduce data dimensionality while preserving much of the information in the original data.

Please use the following training and testing data for the assignment

```
import tensorflow as tf
```

```
# Load and prepare the MNIST dataset.
mnist = tf.keras.datasets.mnist
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

Note that you will not need y_train or y_test as we are just interested in reducing the dimensionality of X.

Part A: PCA (80 pts)

1. Perform Exploratory Data Analysis (EDA) on X_train and discuss the data and what you observe prior to beginning modeling (**visualize the images**) **[10 pts]**
2. Normalize the image data so the pixel values are between 0 and 1. **[10 pts]**
3. Use PCA to reduce the 784 dimensions of the data to 32 dimensions using X_train **[10 pts]**
4. Transform X_train, discuss the original variance in X_train and how much variance is explained by the 32 components. Plot the variance explained as a function of the number of components used and explain why the shape of the plot is what it is (use what we know about PCA) **[20 pts]**
5. Using the transform fit on X_train, transform X_test and discuss the original variance in X_test and how much variance is explained on X_test by the 32 components. **[20 pts]**
6. Compare results from #4 and #5 **[10 pts]**

Hint: See Workbook 11 from class.

Part B: AutoEncoder (70 pts)

1. Start with data after step #2 from Section A
2. Build an autoencoder (either at least 1 hidden layer or using CNN) that will reduce the 784 dimensions of the data to 32 dimensions **[20 pts]**
3. Train an autoencoder on X_train. Discuss the original variance in X_train and how much variance is explained by the 32 dimensions. Hint: Use “model.predict(X_train)” to get predictions **[20 pts]**
4. Using the model trained on X_train, transform X_test and discuss the original variance in X_test and how much variance is explained on X_test by the 32 dimensions of the autoencoder. Hint: Use “model.predict(X_test)” to get predictions. **[20 pts]**
5. Compare results from #3 and #4 and why it is important to test out of sample **[10 pts]**

Hint: You can just modify code from here
https://blog.keras.io/building-autoencoders-in-keras.html

Part C: Explain Results (50 pts)

Discuss which approach explained more variance of the mnist data and why this might be the case (not just based on number of parameters). Also, discuss how many parameters each model/transform uses and which uses more parameters. Lastly, discuss when you may want to use these approaches.

Part D: Extra Credit (100 pts)

Build your own auto-encoder from scratch using numpy instead of tensorflow/keras and use it following the same process you did in Part 2. **Recommendation**: do not use CNN as that will be much more complicated - just use a regular feed-forward NN. You will build

- A general Layer class which will be inherited by classes you write for
  - "Dense" fully-connected layer, $f(X)=W \cdot X+b$
  - "ReLU" layer
- Loss function - binary_crossentropy or mean squared error (up to you)
- Backprop algorithm using SGD with backpropageted gradients