Spring 2024: CSCI 6521 Programming Assignment #2

DUE: Wednesday, April 03, 2024 (**Softcopy** @10 PM via Moodle).

Total Marks = 100

Instructions

- ❑ All work must be your own (besides the instructor-provided codes and hints to be used). You are NOT to work in teams on this assignment.

- ❑ Format: Your solutions must be done using a jupyter notebook page. Submit a single (compressed) file (via Canvas). Name it as PA2_<Your_name_ID_>.

Description

Using jupyter notebook, you need to perform the following tasks:

(1) [**Point 7**] Load the Fashion-MNIST dataset from local drive (see the exercise of Chapter 3). Load the data into the variables: X_train_full (60k), y_train_full (60k), X_valid (10k), and y_valid (10k) in the appropriate order. Split the training X_train_full (60k) data into X_train1 (59000) and [X_train2 (1000), y_train2(1000)].

(2) [**Point 8**] Display some sample images dataset from the training dataset in a 6 × 8 grid.

(3) [**Point 20**] **Model**#1: Train a deep (using Convolutional Layers) **denoising** autoencoder using X_train1 and X_valid. Train it for at least 20 Epochs.

(4) [**Point 25**] Extract the trained encoder part of the **Model**#1 trained in Step#3 and add two consequative dense layers using activation="selu" and activation="softmax" activation functions – the first Dense layer will have 30 nodes [**Hints**: add Flatten(), then add Dense(30), …].

To extract the trained encoder, you can clone it, example: conv_encoder_clone = keras.models.clone_model(conv_encoder), assume the name of the encoder was "conv_encoder". Hints: For this pretraing to build **Model**#2, freeze the weight by using: conv_encoder_clone.trainable = False.

Thus, build **Model**#2 here. Compile **Model**#2 using parameters: loss="sparse_categorical_crossentropy", optimizer=keras.optimizers.SGD(lr=0.02), metrics=["accuracy"].

(5) [**Point 15**] Train **Model**#2 using training dataset: [X_train2 (1000), y_train2(1000)] and validation dataset [X_valid, y_valid] for 20 Epoch. Return the training in "history" variable and plot the training graph using the "history" and DataFrame [**Hints**: Line#1: import pandas as pd, Line #2: pd.DataFrame(history.history).plot(), Line#3: plt.show()].

(6) [**Point 20**] Rebuild **Model**#2 as **Model**#3 without including the trained weight of the encoder of **Model**#1 – and repeat the step as done in #5.

(7) [**Point 5**] Comment on the (validation) accuracies of **Model**#2 versus **Model**#3.

Submission: Submit the jupypter notebook page with all the outputs expressed (as well as a pdf file of the notebook page as an optional backup).

---- x ----