Problem Statement: Human Activity Recognition Time Series Classification (13 marks)

Human Activity Recognition (HAR) using smartphones dataset. Classifying the type of movement amongst six categories:

1. Walking
2. Walking Upstairs
3. Walking Downstairs
4. Sitting
5. Standing
6. Laying

Compared to a classical approach, using Long Short-Term Memory cells (LSTMs) requires no or almost no feature engineering. Data can be fed directly into the neural network who acts like a black box, modeling the problem correctly.
Watch Video to get an understanding how data is being made:
https://www.youtube.com/watch?v=XOEN9W05_4A&feature=youtu.be&ab_channel=JorgeLuisReyesOrtiz

Details about the input data
We will be using an LSTM and CNN + LSTM model on the data to learn (as a cellphone attached on the waist) to recognise the type of activity that the user is doing. The dataset's description goes like this:
The sensor signals (accelerometer and gyroscope) were pre-processed by applying noise filters and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings (features) per window). The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into body acceleration and gravity. The gravitational force is assumed to have only low frequency components, therefore a filter with 0.3 Hz cutoff frequency was used.
There are three main signal types in the raw data: total acceleration, body acceleration, and body gyroscope. Each has 3 axises of data. This means that there are a total of nine variables for each time step.

Prepare a python notebook (recommended- use Google Colab) to develop a **Long Short-Term Memory Recurrent Neural Network** and **a one-dimensional Convolutional Neural Network LSTM, or CNN-LSTM, model**. Read the instructions carefully.

1. **Import Libraries/Dataset (0.5 marks)**
   a. Import required libraries (recommended- use keras library).
   b. Check the GPU available (recommended- use free GPU provided by Google Colab).

2. **Data Pre-processing (0.5+0.5+0.5+1=2.5 marks)**
   You need to define several functions which can be used to load the dataset and bring it in trainable format.

   a. Load file
      i. The signals are stored in the Inertial Signals directory under the train and test subdirectories. The input data is in CSV format where columns are separated by whitespace. Write a function to load the file as a NumPy array.
   b. Load group
      i. Write a function to load all data for a given group (train or test) into a single three-dimensional NumPy array, where the dimensions of the array are [samples, time steps, features]. (You can use the dstack() NumPy function to stack each of the loaded 3D arrays into a single 3D array where the variables are separated on the third dimension (features).)
      ii. features = ['total_acc_x', 'total_acc_y', 'total_acc_z', 'body_acc_x', 'body_acc_y', 'body_acc_z', 'body_gyro_x', 'body_gyro_y', 'body_gyro_z'].
   c. Load dataset group
      i. Write a function that loads all input signal data and the output data for a single group using the consistent naming conventions between the directories.
   d. Load dataset
      i. Write a function that returns the train and test X and y elements ready for fitting and evaluating the defined models.
      ii. Print the shapes of train and test data. (number of samples x number of timestamps x number of features which should be : (number of samples x 128 x 9)

3. **Data Visualization (1 + 1 = 2 marks)**
    a. Plot line graphs for the samples from TWO classes - 'Walking Upstairs' and 'Walking Downstairs' for all 9 features of the dataset and assign class labels to the graph as the title. (use matplotlib/seaborn/any other library).

Model 1: Develop an LSTM Network Model (3.5 marks)

1. **Model Building (0.3*5 = 1.5 mark)**
    a. Sequential Model layers- Use AT LEAST 1 LSTM layers with 120 units with appropriate input for each.
    b. Add one layer of dropout at the appropriate position and give reasons.
    c. Use AT LEAST 2 dense layers (one with 120 units and one for output) with the appropriate input and output for each.
    d. Choose the appropriate activation function for all the layers.
    e. Print the model summary.
2. **Model Compilation (0.5 mark)**
    a. Compile the model with the appropriate loss function.
    b. Use adam optimizer. Give reasons for the choice of learning rate value.
    c. Use accuracy as a metric.
3. **Model Training (0.5 + 0.5 = 1 mark)**
    a. Train the model for an appropriate number of epochs (print the train and validation accuracy/loss for each epoch). Use the batch size of 64.
    b. Plot the loss and accuracy history graphs. Print the total time taken for training.
4. **Model Evaluation (0.25 + 0.25 = 0.5 mark)**
    a. Print the final test/validation loss and accuracy.
    b. Print confusion matrix.

Model 2: Develop a CNN-LSTM Network Model (4.5 marks)
Reference: https://machinelearningmastery.com/cnn-long-short-term-memory-networks/

1. **Model Building (0.5*5 = 2.5 mark)**
    a. Sequential Model layers- Use AT LEAST 2 CNN - LSTM layers with 64 Conv filters and 120 LSTM units with
    b. appropriate input for each.
    c. Add MaxPooling1D layer
    d. Use AT LEAST 2 dense layers (one with 120 units and one for output) with the appropriate input and output for each.
    e. Choose the appropriate activation function for all the layers.
    f. Print the model summary.
2. **Model Compilation (0.5 mark)**
    a. Compile the model with the appropriate loss function.
    b. Use SGD optimizer. Give reasons for the choice of learning rate value.
    c. Use accuracy as metric.
3. **Model Training (0.5 + 0.5 = 1 mark)**
    a. Train the model for an appropriate number of epochs (print the train and validation accuracy/loss for each epoch). Use the batch size of 64.
    b. Plot the loss and accuracy history graphs. Print the total time taken for training.
4. **Model Evaluation (0.25 + 0.25 = 0.5 mark)**
    a. Print the final test/validation loss and accuracy.
    b. Print confusion matrix

Evaluation process-
1. Task Response and Task Completion- All the models should be logically sound and have decent accuracy (models with random guessing, frozen and incorrect accuracy, exploding gradients etc. will lead to deduction of marks. Please do a sanity check of your model and results before submission). There are a lot of subparts, so answer each completely and correctly, as no partial marks will be awarded for partially correct subparts.
2. Implementation- The model layers, parameters, hyperparameters, evaluation metrics etc. should be properly implemented.