Summary: In this assignment, you will use an existing Keras example file as template (starting point) for your assignment and you will gain experience on using Keras. As the dataset, you will use both **MNIST** and **Fashion-MNIST** datasets individually (both are available under *keras.datasets*) in this assignment for classification. You can read more about the available datasets in Keras at: https://keras.io/datasets/ If you like to read more about those two datasets, check: MNIST on Wiki and Fashion-MNIST paper . Both datasets are commonly used for designing classification algorithms.

Instructions:

- You will **submit individual python scripts (.py files)** for each step below.
- In addition to that, you will also **submit a single report in PDF format** (save it as PA4_Report.pdf). Your report will include all the figures and/or the output values of your individual python scripts.
- If you use the same code section in multiple sections, then copy and paste that code in each file individually.
- ***Each file should be able to run by itself and be able to produce the asked results.*** **Error producing codes may not receive credit.**
- At the end of this assignment, you should **compress all your *.py* files and your report file** (.pdf) in a **<YourName>_PA4.zip** file (where you will replace <YourName> with your own particular full name).
- Keep all the settings and parameters the same in your code as in the template file unless specified otherwise below.

Additional Computation: Note that while this assignment might take some time to compute, it is within the reasonable limits (and it should not take long).

- You may also consider using Google colab to run some of these experiments remotely without installing anything on your local computer, if you need additional computer. Colab is available at: https://colab.research.google.com/notebooks/welcome.ipynb

- In that case, make sure that you use Google's colab with the GPU (or TPU) support.

- To change the default hardware accelerator setting to GPU from the menu, click on: "Runtime" > "Change Runtime Type" and then click on "Hardware accelerator" to choose GPU on colab. Then click on "save" to use colab with GPU support.
- After that, you can copy and paste your code in a cell and then you can run that cell ☺

(Programming) Assignment 4:

1) First, download and run the example python file: https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py This particular model, uses a simple ConvNet model (let's call this model as: the **basic ConvNet model**) and trains it on the MNIST dataset for classification. You will use this file as template in this assignment.

2) Your first task**: run the template code** and simply write down (report) the accuracy value you obtained on the test data after 12 epochs in your report file. (5 points). You do not need to submit a code for this step.

3) **Your next task is understanding the used network model in the example. Write down the network model** (the architecture) from the code **in your report** (5 points).

   Here is an example for describing a simple model in the text form: Conv ===> Pooling ===> FC ===> softmax. Make sure you also list the number of filters, filter dims, number of units and activation functions for each layer (where applicable). As an alternative, you can **draw** a figure illustrating the network architecture as well, if you prefer drawing figures. You do not need to submit a code for this step.

4) Now, let's have a look at how changing epoch value affects the network's performance. Change the ***epoch*** (iteration) value **to 30 in the code** and **plot both training and test accuracies vs. epoch**. There are many ways to plot that in python. If you don't want to write your own python function for that task, consider using Tensorboard (hint: check *keras.callbacks.TensorBoard*()). Include your plot in your report. (2 points)

   save and submit your code as *PA4_Part4.py* (10 points)

5) **Plot both the training & test (validation) losses vs. epoch over 30 epochs** (3 points). Note that this particular python code uses the test data as the validation data.
   a. save and submit your code as *PA4_Part5.py* (10 points)

6) Now, let's focus on changing (or editing) the existing network architecture in that existing code. Current template file uses a simple ConvNet model and we will replace that with the LeNet5 model that we studied in the class. **Replace the simple ConvNet model with the LeNet5 model**. Refer to the lecture slides for the full LeNet5 model and its details. Note that there are also many available Keras implementations available online for LeNet5. You can use any of them as a starting point as long as they match the model that we discussed and studied in the class. Train the new network (the LeNet5 model) on the **MNIST** dataset and **plot the accuracy (both training and test) vs. epoch** over 30 epochs. Show your plot in your report (3 points). **Note:** change only the model in the code. Do not change the other sections in the code: such as loss functions, optimization parameters, etc.

save and submit your code as *PA4_Part6.py* (10 points)

7) You now have a working LeNet5 model that you trained on the MNIST dataset and now it can recognize the digits. Now, let's train the LeNet5 model on another dataset. **Train your LeNet5 architecture on the Fashion-MNIST dataset and plot the accuracy (both training and test) vs. epoch over 30 epochs**. Now you have a LeNet5 model trained on fashion-MNIST dataset. Now that we have that already available, let's see if a simpler model would do any better. Train the simple ConvNet model (the original model in the template file) on the Fashion-MNIST as well and then **compare** the result of LeNet5 model to the basic ConvNet model you used above. Report your plots and the average training time per epoch for each model in your pdf file (3 points). LeNet5 is a deeper model compared to the simple ConvNet model. Did using a deeper net help in this caseto obtain a higher accuracy? (2 points) Note: in this step, you focus on changing only the data related section from your previous code (*PA4_Par6.py)*.

Save and submit your code as *PA4_Part7.py* (10 points)

8) The original particular template that you downloaded from GitHub uses "Adadelta" as the default optimization algorithm. Let's see how Stochastic Gradient Descent would do in this model instead of Adadelta. **Change the optimizer to stochastic gradient descent (SGD) as shown below in your current LeNet5 implementation. Plot the accuracy vs. epoch over 30 epochs for both training and testing data.** Here, you can use either MNIST or Fashion MNIST dataset. Also plot the loss (or cost ☺) vs. epoch over 30 epochs. You can use the template given below for setting the optimizer to SGD. (Optional: You are welcome to use additional parameters for SGD, if you prefer.) include your plot in your report file (2 points).

```
sgd = keras.optimizers.SGD(lr=0.001)
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=sgd,
              metrics=['accuracy'])
```

save and submit your code as *PA4_Part8.py* (10 points)

9) We know that learning rate is also an important hyper-parameter. Let's study that here. **Use 5 random (but meaningful) learning rates and train your LeNet5 model at each of those 5 random learning rates separately on the MNIST dataset.** Plot your accuracy vs. epoch results over 30 epochs for each of those learning rate values in a for-loop (over the 5 different learning rates). Which learning rate yields the best result? Include your chosen 5 learning rate values and their corresponding plots in your report (2 points)

save and submit your code as *PA4_Part9.py* (10 points)

10) By looking at all your results that you obtained in this assignment, **comment** on which optimization algorithm (AdaDelta or SGD) worked better in your case in your report (3 points). What did you observe by changing the learning rate to any of those 5 values in the previous step: Was all of those 5 learning rates useful, is using a smaller or a higher value better for the learning rate in SGD? (5 points) Did you see any tradeoff between the learning rate and convergence? (5 points)
(In some plots, you should see that if you included more iterations/epochs, you would get even better accuracies while in others you may notice that you have already reached to a saturated point where including more iterations does not help). You do not need to submit your code for this step. Answer the questions in report.

Step2) ----- / 5 points
Step3) ----- / 5 points
Step4) ----- / 12 points
Step5) ----- / 13 points
Step6) ----- / 13 points
Step7) ----- / 15 points
Step8) ----- / 12 points
Step9) ----- / 12 points
Step10) ----- / 13 points

Total: ________ / 100 points