TensorFlow/Keras Assignment

<u>Problem Specification</u>

The objective of this assessment is to build a range of machine learning models using TensorFlow and Keras.

- Part A of the assignment is specifically focused on building TensorFlow models using the <u>low level API</u>. **[40 Marks]**

- Part B of the assignment looks at the application of Keras applied to an image classification problem. **[40 Marks]**

- Part C is a research question focused on specific algorithmic technique for deep learning networks. **[20 Marks]**

Please upload your final submission (as a single zip file) to Canvas before **22:00 on Sunday April 14th**. (Please note if you submit within 7 days of this deadline (on or before Sunday April 21st) a late penalty will not be applied).

Your submitted zip file should contain your python files (one notebook for part A and B) and a report. Please note that all code should be fully commented.

<u>Please do not include the original image data files in your uploaded zip file</u>.

PART A - TensorFlow and the Low Level API. [40 Marks]

The Fashion-MNIST is an image dataset consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image (784 pixel values in total), associated with a label from 10 different classes.

The following are the set of classes in this classification problem (the associated integer class label is listed in brackets).

- T-shirt/top (0)
- Trouser (1)
- Pullover (2)
- Dress (3)
- Coat (4)
- Sandal (5)
- Shirt (6)
- Sneaker (7)
- Bag (8)
- Ankle boot (9)

You can load the Fashion MNIST dataset as follows:

```
import tensorflow as tf

 fashion_mnist = tf.keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

(i) The initial task is to build a binary two layer classifier using the TensorFlow low level API (in graph mode).

Preprocessing Instructions

Your code should extract all data related to two classes from the training and test data (Remember irrespective of the two classes you extract you should map the associated labels to 0 and 1).

For example, assume you want to train your binary classifier to differentiate between images of a dress and a bag. The integer class label for a dress is 3 and the integer class label for a bag is 8. You should extract all data related to just these two classes from the training and test data. Therefore, your class labels for your train and test data will only contain the integer values 3 and 8. Likewise your feature data (train and test) will only contain those rows which were originally associated with the classes 3 or 8. The final pre-processing step is to map the chosen label values to 0 and 1. In other words in this example I would need to map all labels encoded as 3 to 0 and all labels encoded as 8 to 1. The reason for this is that the Sigmoid unit is a binary classifier that can only outputs values between 0 and 1. (Please note there is no need to one-hot-encode these labels).

Model Architecture

Use TensorFlow to build a binary classifier which has the following basic architecture:

a. Layer 1: 100 neurons (ReLu activation function)
b. Layer 2: 1 neuron (Sigmoid activation function)
c. Learning Rate: 0.01 with Gradient Descent

The following methods may be useful:

- tf.nn.sigmoid_cross_entropy_with_logits(logits=A2, labels=y). This will take in the pre-activation values (logits) of a neuron, pipe them through the Sigmoid activation function. It will then compare the resulting predicted values with the true labels y and return the cross entropy error for every single training instance. [See Week 9 lecture note for an example of its usage].
- tf.sigmoid(A2). This will take in the pre-activation values (logits) of the Sigmoid neuron for all training instances (A2). It will pass each value through the Sigmoid function, which will output a predicted value for each training example (between 0 and 1) [Again see Week 9 lecture notes for an example of usage].

(15 marks)

(ii) The second task we will address is a full multi-class classification problem (all 10 classes for Fashion MNIST). Using TensorFlow's low level API (in graph mode) build a multi-layer neural network for tackling this problem. Your network should consist of the following architecture:

a. Layer 1: 300 neurons (ReLu activation functions).
b. Layer 2: 100 neurons (ReLu activation function)
c. Layer 3: Softmax Layer
d. Learning rate: 0.01 for this problem with Gradient Descent.

As the class labels are integer encoded you should convert them to one hot encoded (see neural network example from Week 9 lecture notes).

In your report document the level of accuracy you were able to obtain after 40 epochs with the above model and the time duration for this training process. Include a graph showing the validation and training accuracy after each epoch. **(15 marks)**

(iii) You will have noticed that the training process for the network created in part (ii) is quite slow (even for this relatively small dataset run on a GPU or TPU). Adjust your code to implement mini-batch gradient descent.

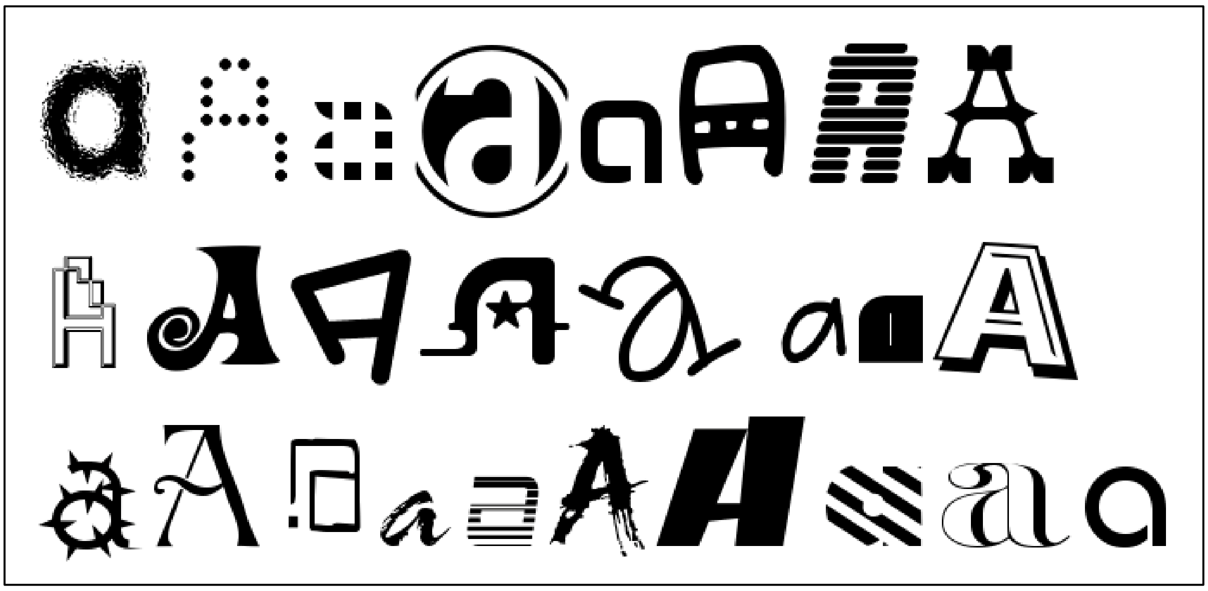
In your report contrast the performance of the mini-batch gradient descent with that achieved by the model in part 1. Did the selection of a specific batch size have any impact? **(10 marks)**

PART B - Keras – High Level API [40 Marks]

The objective of Part B is to explore the problem of image classification is using a dataset called the notMNIST dataset. The dataset contains images of letters from A – J inclusive. A sample of some of the images are depicted in the image below.

We will be working with a modified version of the original dataset. Some pre-processing work has been completed on the dataset and it has also been normalized.

The following is a selection of images from the dataset.



Obtaining the data

In the Canvas folder you will find a .zip file called **data.zip**. This zip file contains a HDF5 file called data.h5. In turn this file contains the training and test data. In total there are 200,000 training images and 17,000 test images. Each image consists of 28*28 pixels, therefore there are 784 features. See **Appendix A** at the end of the assignment for instructions on how to:

1. Upload data.zip to Datalab
2. Extract data.h5 from the zip fie
3. Access the training and test data from the data.h5 file

(**Appendix B** provides the instruction for those using Colab)

(i) The initial task for part 2 is to use Keras to build a SoftMax classifier. This will serve as a benchmark for the work in (ii) and (iii) below. You should be able to achieve an accuracy of approximately 85% on the test data. Specify a batch size of 256 when building your model. **(5 marks)**

(ii) Using Keras build a two layer fully-connected neural network with a single layer of ReLU activation neurons connected to a Softmax layer. Is there an improvement in accuracy over the logistic regression model?

Examine the application of a deeper neural network to your problem (evaluate with 3, 4, 5 layers). What combination gives you the best performance on the test set?

A comprehensive investigation of the network architecture if beyond the scope of this assignment. However, you should compare and contrast the performance of at least three different network configurations such as the following:

- L1 200 Neurons L2 Softmax
- L1 400 Neurons L2 200 Neurons L3 Softmax
- L1 600 Neurons L2 400 Neurons L3 200 Neurons L4 Softmax

For each configuration produce a visualisation of the loss and accuracy for the validation and training data. Please enclose these visualisations and your observations and comparative analysis in your report. **(20 marks)**

(iii) Regularization can be an effective technique to address overfitting in deep neural network. Apply dropout, L1 and L2 regularization to the two deepest networks created in part (ii). In your report describe the impact of regularization techniques. Include visualisations of the training process when dropout is applied and any observations of the impact of Dropout on these networks. **(15 marks)**

Part C: Research [20 Marks]

Deep learning has benefitted from a range of algorithmic advances over the last 10 years. While we have covered a number of these during the course there are many others can have a very significant impact on the training or performance of a model. Select one of the following two algorithmic techniques. Research the technique and provide a description (2 page max, not including images) of the problem that the addresses, how the techniques operates and overcomes this potential problem.

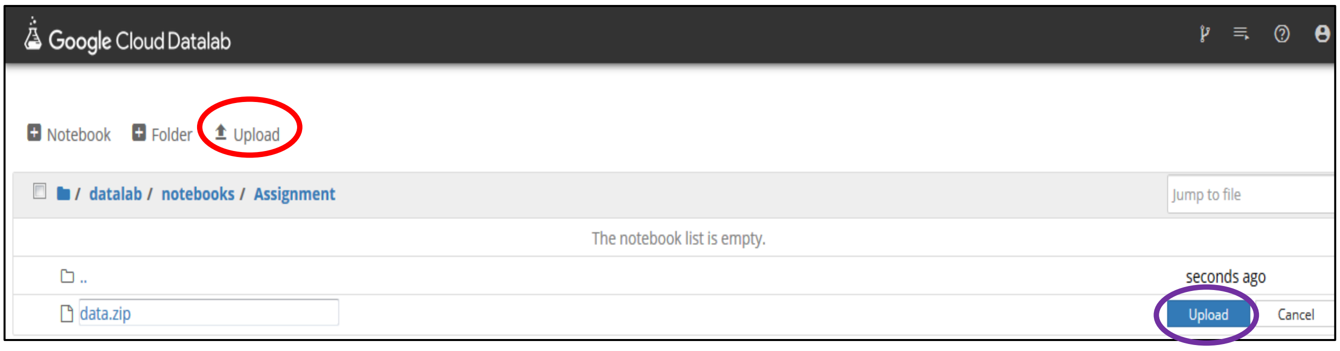
- Batch Normalization
- Adam Optimization Algorithm

To grade well in this section you should demonstrate a clear understanding in your own words of the selected technique. Please reference any sources you use.

(20 marks)

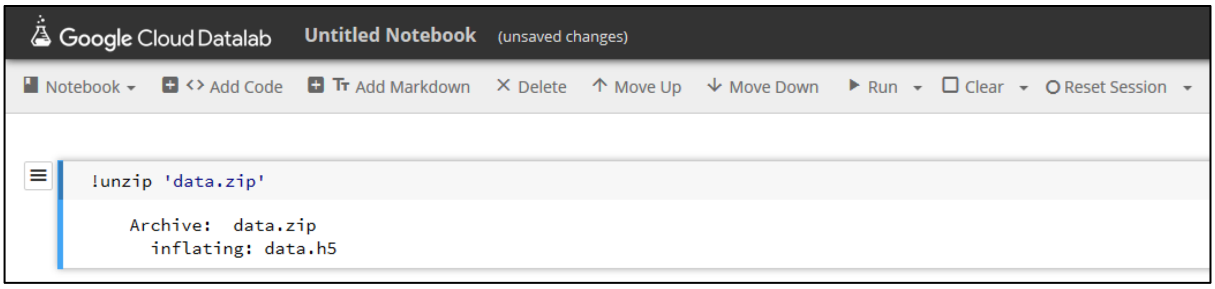
Appendix A: Accessing Training Data for Part B (DataLab).

To upload the dataset to Datalab please click on the upload button highlighted in red below. Then another upload button will appear (highlighted in purple below). Click this upload button. Depending on your connection speed this upload may take some time.



Once you have downloaded the zip file you will then need to extract its contents. To do this in Datalab you just have to open a notebook and type the following command into a cell (the notebook must be in the same directory as the data.zip file). You should now see the data.h5 file in the same directory. This file is 1.36 GBs in size.

```
!unzip 'data.zip'
```



```
!unzip 'data.zip'

    Archive:  data.zip
      inflating: data.h5
```

At the stage you should have a file called data.h5. This file is stored in a hdf5 format, which makes it allows us to read it very efficiently and quickly from disk. Now you should use the code called template.py to read the training and test data into NumPy arrays within your code. This code available on Canvas and reproduced below.

```
import numpy as np

import h5py



def loadData():

    with h5py.File('data.h5','r') as hf:

        print('List of arrays in this file: \n', hf.keys())

        allTrain = hf.get('trainData')

        allTest = hf.get('testData')

        npTrain = np.array(allTrain)

        npTest = np.array(allTest)



        print('Shape of the array dataset_1: \n', npTrain.shape)

        print('Shape of the array dataset_2: \n', npTest.shape)

      return npTrain[:,:-1], npTrain[:, -1], npTest[:,:-1], npTest[:, -1]



trainX, trainY, testX, testY = loadData()
```

Appendix B: Accessing Training Data for Part B (Colab).

While there are a number of ways of importing data into Colab, the following steps describe how to mount your Google Drive from Colab.

1. Copy the compressed data file (data.zip) to a folder in your Google Drive (wait for the upload to complete). For the purposes of this example I have placed data.zip in a folder in Google Drive called Colab Notebooks.

2. This step involved mounting your Google Drive from Colab. In a Colab notebook execute the following code. This will ask you to enter follow a link and enter an authorisation code.

```
from google.colab import drive

drive.mount('/content/gdrive')
```

3. Once complete you should see the message "Mounted at /content/gdrive"

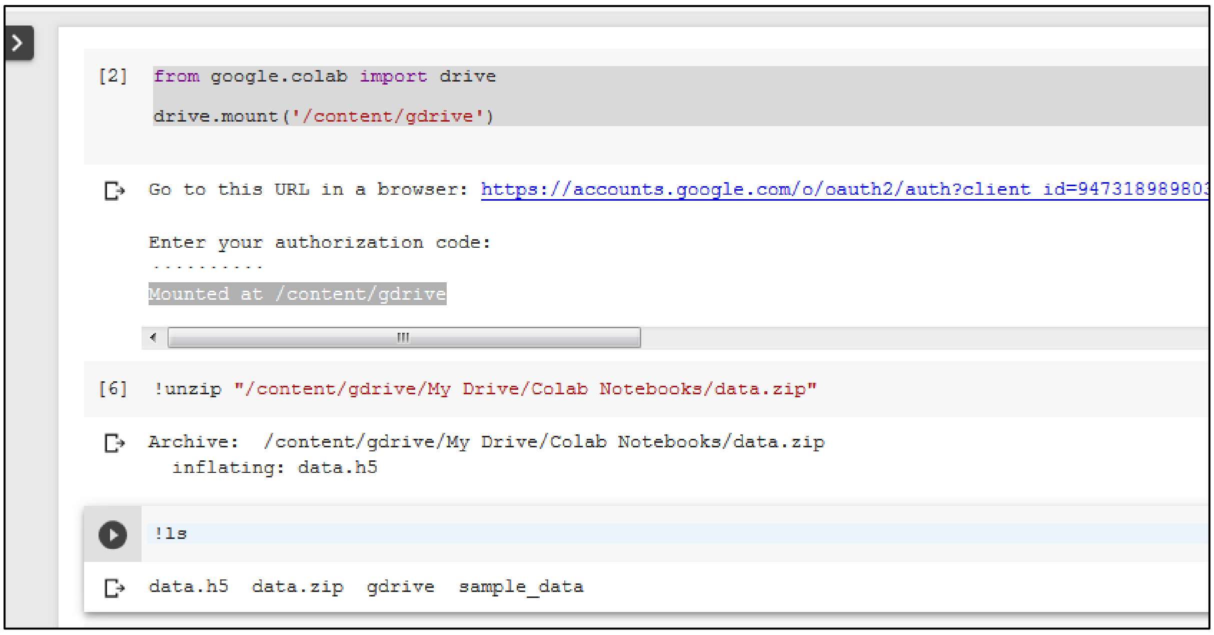
4. Next we decompress the file from Colab by entering the following in a Colab notebook. Remember my zip file in stored in the folder Colab Notebooks.

```
!unzip "/content/gdrive/My Drive/Colab Notebooks/data.zip"
```

5. This should extract the file **data.h5** into your current working directory. To confirm run the following in a Colab cell.

```
!ls
```

You should see the data.h5 file in the output. You can now use the code included at the end of Appendix A above to open data.h5 and extract the training and test data. The following is a screenshot from Colab showing the steps involved.



```
[2] from google.colab import drive

    drive.mount('/content/gdrive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803

Enter your authorization code:
··········
Mounted at /content/gdrive
```

```
[6] !unzip "/content/gdrive/My Drive/Colab Notebooks/data.zip"
```

```
Archive:  /content/gdrive/My Drive/Colab Notebooks/data.zip
  inflating: data.h5
```

```
!ls
```

```
data.h5  data.zip  gdrive  sample_data
```