

TensorFlow/Keras Assignment



Problem Specification

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

- Part A of the assignment is specifically focused on building models using TensorFlow's low level API and directly using automatic differentiation. **[50 Marks]**
- Part B of the assignment looks at the application of Keras to an image classification problem. **[30 Marks]**
- Part C is a research question focuses on specific algorithmic technique for deep learning networks. **[20 Marks]**

Guidelines and Submission Instructions

- Please upload your final submission (as a single zip file) to Canvas before 23:00 on **Sunday April 5th**.
- Your submitted zip file should contain your python files (notebooks for part A and B) and a report.
- Please **do not** include the dataset files in your uploaded zip file.
- It is your responsibility to make sure you upload the correct files.
- Please make sure you fully comment your code. You should clearly explain the operation of important lines of code.
- Please note that marks are awarded for code that is efficient, well structured and with minimum duplication.
- Late submissions will be penalized.
 - If you submit the assignment after the deadline but within 7 days, **no late penalty will be applied**. Even though no penalty is applied in this case I strongly encourage you to aim for the original submission date of April 5th.
 - If you submit the assignment more than 7 days after the deadline but within 14 days, a **20% penalty** will be deducted.
 - A **grade of 0%** will be given to any assignment submitted more than 14 days after the assignment deadline.
- Please clearly reference any sources you use in your code or report.
- Please note that analysis tools will use to identify plagiarism.

PART A - TensorFlow and the Low Level API. [50 Marks]

The Fashion-MNIST dataset is a basic 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)

The code for loading and pre-processing the Fashion MNIST dataset is shown below. You can also obtain the code [here](#). In this code the class labels are one hot encoded. For example, the final shape of the matrix that contains the training class labels is (10, 60000).

You will notice that this code will print out the shape of the training and test data for both features and labels as follows:

```
Shape of training features (60000, 784)
Shape of test features (10000, 784)
Shape of training labels (10, 60000)
Shape of testing labels (10, 10000)
```

Depending on how you implement your solution for Question 1 you may want to change the shape of the above matrices (and this allowable).

```

import tensorflow as tf
from keras.utils import np_utils

fashion_mnist = tf.keras.datasets.fashion_mnist

# load the training and test data
(tr_x, tr_y), (te_x, te_y) = fashion_mnist.load_data()

# reshape the feature data
tr_x = tr_x.reshape(tr_x.shape[0], 784)
te_x = te_x.reshape(te_x.shape[0], 784)

# normalize feature data
tr_x = tr_x / 255.0
te_x = te_x / 255.0

print( "Shape of training features ", tr_x.shape)
print( "Shape of test features ", te_x.shape)

# one hot encode the training labels and get the transpose
tr_y = np_utils.to_categorical(tr_y,10)
tr_y = tr_y.T
print ( "Shape of training labels ", tr_y.shape)

# one hot encode the test labels and get the transpose
te_y = np_utils.to_categorical(te_y,10)
te_y = te_y.T
print ( "Shape of testing labels ", te_y.shape)

```

(i) **[Notebook should be called Question1_1]**

Your initial objective is to build a SoftMax classifier for this problem (in other words your model will only contain one Softmax layer). More specifically you need to use TensorFlow 2.1 to build a single SoftMax layer containing 10 neurons for this problem.

This should be implemented as a vectorized solution (the use of iterative for loops should be minimized). Your code should push the training feature matrix (containing all training data) through the Softmax layer (using matrix multiplication, etc) and calculate the output probabilities for each class for all training instances. It should then calculate the total cross entropy loss. Both forward pass and the cross entropy calculations should be recorded using a gradient tape. You should then use the gradient tape to calculate

the partial derivatives of the loss with respect to all trainable variables. You should use an Adam optimizer to update the trainable parameters.

Your code should use Tensorflow functions wherever possible and should only use lower level functions (such as `tf.matmul`, `tf.reduce_sum`, etc). You will lose marks if you use higher level functions for performing tasks such as calculating the cross entropy loss or for calculating the Softmax activations or output etc. If you are unsure if a specific function is allowable please post the question to the discussion forum.

Please make sure your code includes the following three functions:

1. A function called **forward_pass**. This function should use matrix multiplication along with other low level functions to push the entire feature matrix through the Softmax layer and should return the 10 class probabilities for all feature instances (as one collective matrix).
2. A function called **cross_entropy**. The objective of this function is to take in the matrix of probabilities for all feature instances (the output of `forward_pass` above) along with the one-hot-encoded true class labels and calculate the total cross entropy loss.
3. A function called **calculate_accuracy**. This function should take in the matrix of probabilities for all feature instances (the output of `forward_pass`) along with the one-hot-encoded true class labels and calculate the current accuracy of the model.

You are expected to research appropriate Tensorflow methods to help you complete the functions above. Please include each of the three functions above in your final report document. You should include a paragraph under each function clearly describing how you implemented your solution. It is very important that the written description should show that you clearly understand the operation of the code you implemented.

Again, it is important to emphasize that only low level Tensorflow functions are allowable above.

Your report document should also contain your evaluation of the Softmax model when run on the Fashion MNIST data. Please use an Adam optimizer with Gradient Tape to facilitate the update of all trainable parameters. Support your evaluation with graphs where possible.

[Recommended maximum length of report for Q1(i) is three pages].

Note: You will need to successfully complete Q1(i) in order to move on to Question (ii) and (iii). If you are unable to complete parts of the three functions above using low level code then it is allowable to resort to use of high level Tensorflow function (you will lose marks for the use of these high level functions but it will allow you to progress to Question (ii) and (iii)).

(25 marks)

- (ii) The second task will introduce additional layers of ReLu based neurons before the SoftMax layer in order to build a more complex neural network model.

Using TensorFlow's low level API with autodiff build a multi-layer neural network for tackling this problem. You should build on the code you have implemented in Q1(i). More specifically you should implement and evaluate the following two alternative architectures:

Network architecture A [**Notebook should be called Question1_2_1**] :

- a. Layer 1: 300 neurons (ReLu activation functions).
- b. Layer 2: Softmax Layer (from Q1 (i))

Network architecture B [**Notebook should be called Question1_2_2**]:

- a. Layer 1: 300 neurons (ReLu activation functions).
- b. Layer 2: 100 neurons (ReLu activation function)
- c. Layer 3: Softmax Layer (from Q1 (i))

In your report include your **forward_pass** function for both of the above network architectures.

Clearly document the evaluation of both of the above network architectures. Are these architectures sensitive to the number of neurons in each layers? Is there evidence of these networks overfitting? Please include graphs to support your statements.

Again it is important to note that the code above should be vectorized and should only use low level functions (same as part (i))

(15 marks)

- (iii) [**Notebook should be called Question1_2_2**]

L1 and L2 are commonly used regularization techniques. Implement and evaluate both types of regularization for network architecture B (see below):

- a. Layer 1: 300 neurons (ReLu activation functions).
- b. Layer 2: 100 neurons (ReLu activation function)
- c. Layer 3: Softmax Layer (from Q1 (i))

In your report include the version of the forward_pass function where you have implemented L1. Separately include the version of forward_pass where you have implemented L2. You should also include an analysis of the impact of both L1 and L2 on the model. Support your analysis with graphs.

(10 marks)

PART B - Keras – High Level API [30 Marks]

[Please use just one notebook for this part and call it PartB]

The objective of Part B is to explore the problem of image classification when using a dataset a dataset that 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 assignment 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 Colab
2. Extract data.h5 from the zip file
3. Access the training and test data from the data.h5 file

Please note the process is similar if you are using the Deep Learning VM on Google Cloud.

- (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 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.

Next 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 SoftMax model?

Examine the application of a deeper neural network to your problem (evaluate with 3, 4, 5 layers).

A comprehensive investigation of the network architecture is 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

Please include your comparative analysis in your report and support your observations and findings with graphs. **(15 marks)**

- (ii) Regularization can be an effective technique to address overfitting in deep neural network. Apply dropout regularization to the two deepest networks created in part (ii). In your report describe the impact of dropout. 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 that 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 it addresses, how the technique 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 (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. Please note this process is also described in the Canvas unit called “Guide to Using Google Cloud Deep Learning VM”. See “Guide to Deploying and Running Google Deep Learning VM” for more detail on how to upload data and mount your Google Drive from Colab. The process is summarized below. You can find the full code for accessing the data [here](#).

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 involves 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')
```

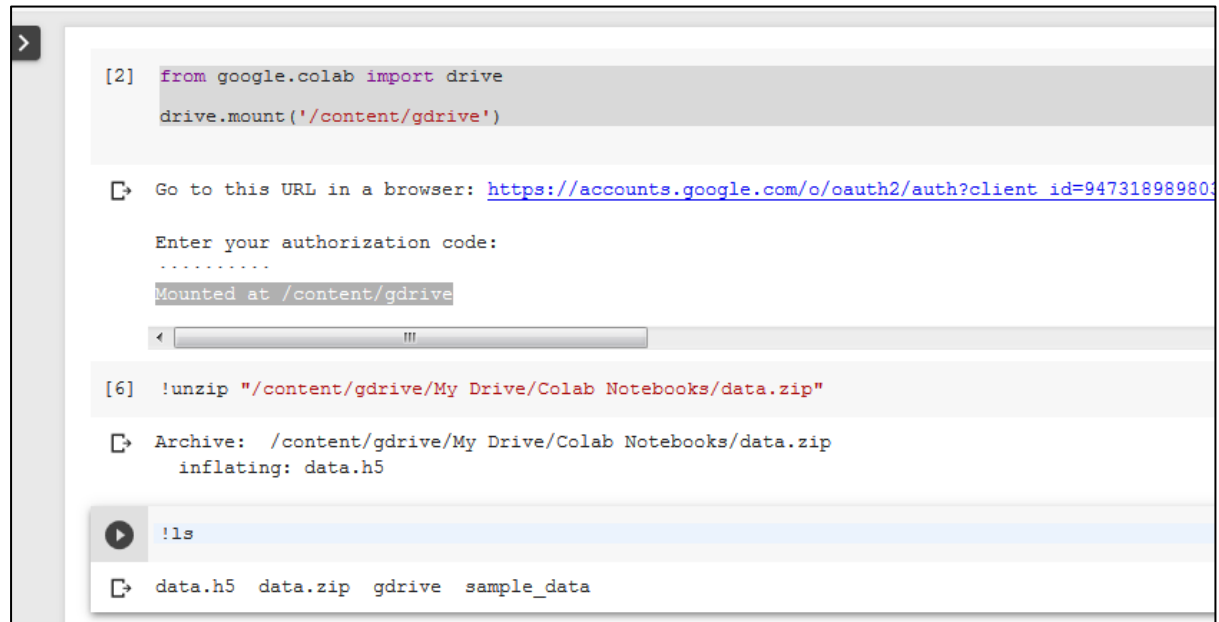
3. Once complete you should see the message “Mounted at /content/gdrive”
4. Next we decompress the file from Colab by entering the following in a Colab notebook. Remember my zip file is stored in the folder Colab Notebooks.

```
!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 (see screen shot below).



```
[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

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 is available on Canvas and reproduced below. Again you just access all this code in the following [Colab Notebook](#).

```
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()
```