Cesar Lopez
Individual Report
Machine Learning II

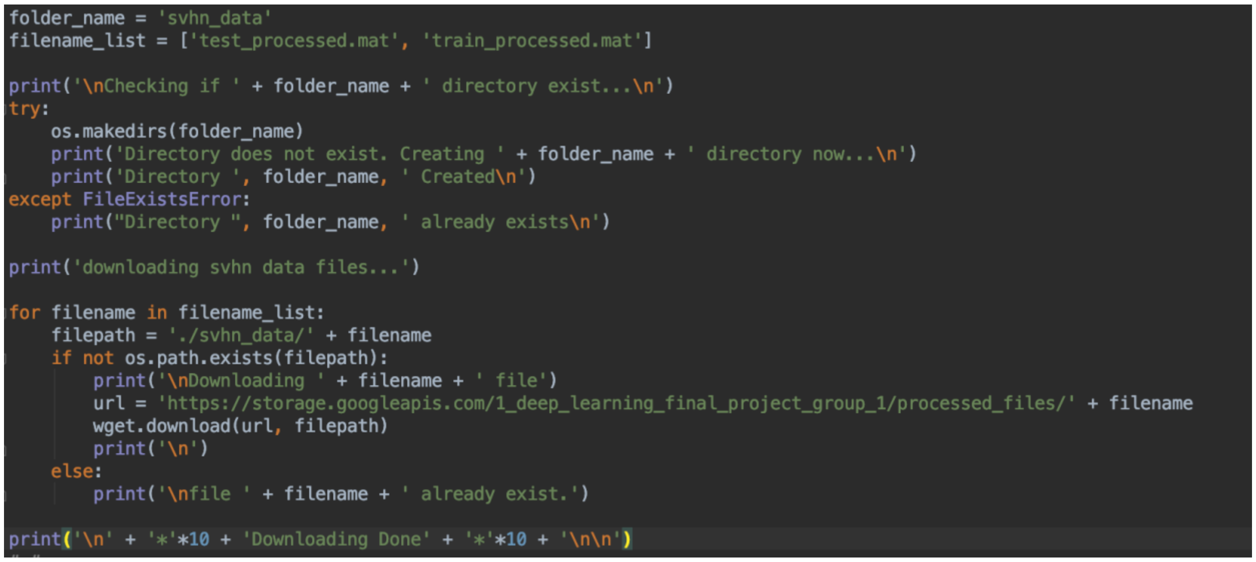
Cesar Lopez Individual Report

I decided to work on Tensorflow and Keras. The reason behind this was because we never got to do homework on with these frameworks and they are important to know in the field.

First thing I dealt with was the data preprocessing. I am familiar with Matlab, so I originally wanted to change the image format to jpeg, then start using python. However, I learned about the SciPy python library. This enabled me to get the data I needed from the Matlab files using python.

I looked around on what to use to download the Matlab files from the website. I wanted to use request, but wget (python version) seemed easier to use. You can find the web page where I got it from by clicking here. I implemented my style of coding to use.

I also wrote code to create a directory for the data. It will check if it exists first. If it does not, it will create one. I used a similar method with checking if the Matlab files existed to prevent multiple downloads when running the code. A screenshot of this code is below. My team implemented this code to efficiently get our data.



```
folder_name = 'svhn_data'
filename_list = ['test_processed.mat', 'train_processed.mat']

print('\nChecking if ' + folder_name + ' directory exist...\n')
try:
    os.makedirs(folder_name)
    print('Directory does not exist. Creating ' + folder_name + ' directory now...\n')
    print('Directory ', folder_name, ' Created\n')
except FileExistsError:
    print("Directory ", folder_name, ' already exists\n')

print('downloading svhn data files...')

for filename in filename_list:
    filepath = './svhn_data/' + filename
    if not os.path.exists(filepath):
        print('\nDownloading ' + filename + ' file')
        url = 'https://storage.googleapis.com/1_deep_learning_final_project_group_1/processed_files/' + filename
        wget.download(url, filepath)
        print('\n')
    else:
        print('\nfile ' + filename + ' already exist.')

print('\n' + '*'*10 + 'Downloading Done' + '*'*10 + '\n\n')
```

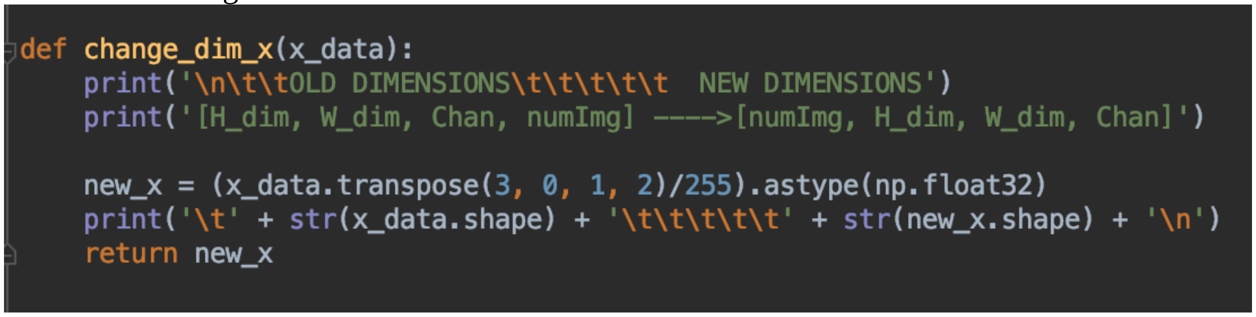
I thought of how long my code file would look, therefore, I created a class that does all this. I put the class in python package I created in PyCharm.

While setting up my code, I needed to get the image data in the correct shape. TensorFlow and Keras like to have their channels in the last shape position.

Image Data shape → [Number of Images, Height Pixel, Width Pixel, Channel]

I wrote a function that does it that for me. Below is a screenshot of the function.

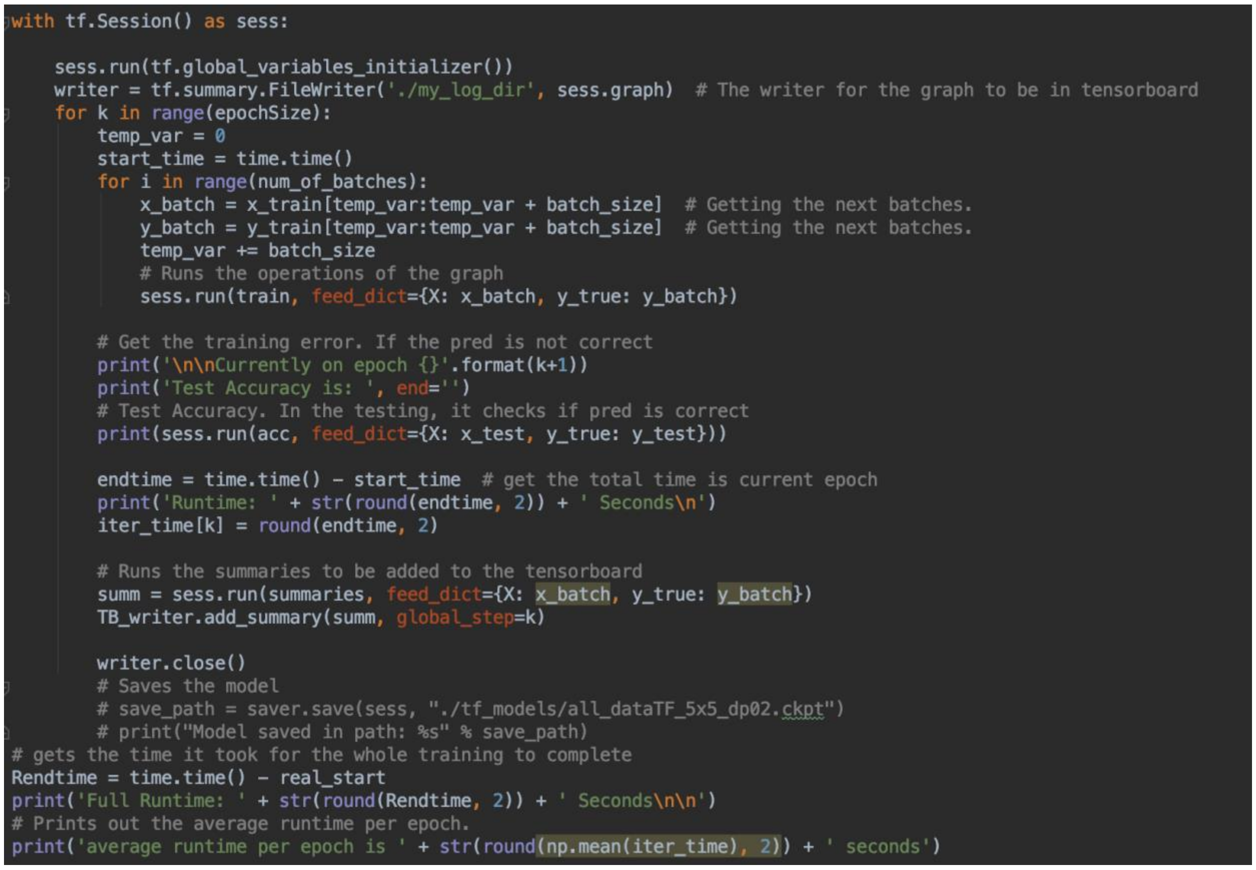
Cesar Lopez
Individual Report
Machine Learning II



```python
def change_dim_x(x_data):
    print('\n\t\tOLD DIMENSIONS\t\t\t\t\t  NEW DIMENSIONS')
    print('[H_dim, W_dim, Chan, numImg] ---->[numImg, H_dim, W_dim, Chan]')

    new_x = (x_data.transpose(3, 0, 1, 2)/255).astype(np.float32)
    print('\t' + str(x_data.shape) + '\t\t\t\t\t' + str(new_x.shape) + '\n')
    return new_x
```

TensorFlow

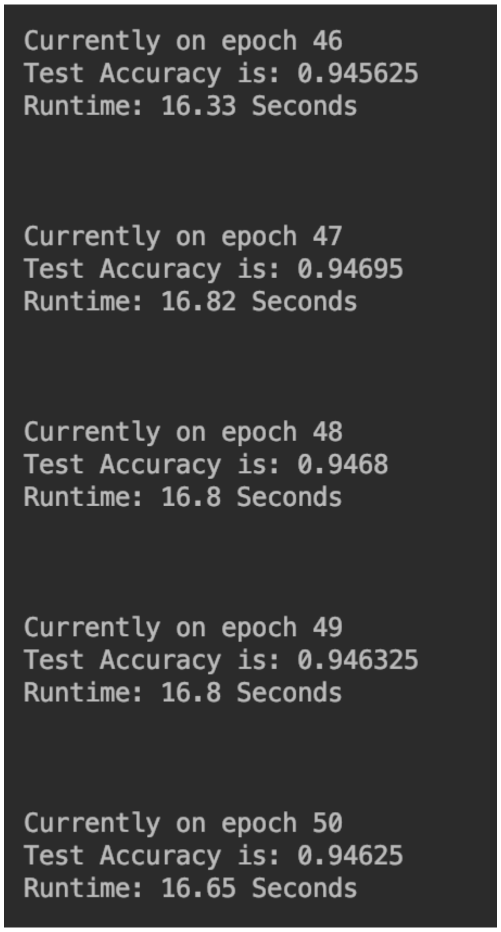
Below is my code where I ran my graph session in Tensorflow:



```python
with tf.Session() as sess:

    sess.run(tf.global_variables_initializer())
    writer = tf.summary.FileWriter('./my_log_dir', sess.graph)  # The writer for the graph to be in tensorboard
    for k in range(epochSize):
        temp_var = 0
        start_time = time.time()
        for i in range(num_of_batches):
            x_batch = x_train[temp_var:temp_var + batch_size]  # Getting the next batches.
            y_batch = y_train[temp_var:temp_var + batch_size]  # Getting the next batches.
            temp_var += batch_size
            # Runs the operations of the graph
            sess.run(train, feed_dict={X: x_batch, y_true: y_batch})

        # Get the training error. If the pred is not correct
        print('\n\nCurrently on epoch {}'.format(k+1))
        print('Test Accuracy is: ', end='')
        # Test Accuracy. In the testing, it checks if pred is correct
        print(sess.run(acc, feed_dict={X: x_test, y_true: y_test}))

        endtime = time.time() - start_time  # get the total time is current epoch
        print('Runtime: ' + str(round(endtime, 2)) + ' Seconds\n')
        iter_time[k] = round(endtime, 2)

        # Runs the summaries to be added to the tensorboard
        summ = sess.run(summaries, feed_dict={X: x_batch, y_true: y_batch})
        TB_writer.add_summary(summ, global_step=k)

        writer.close()
        # Saves the model
        # save_path = saver.save(sess, "./tf_models/all_dataTF_5x5_dp02.ckpt")
        # print("Model saved in path: %s" % save_path)
# gets the time it took for the whole training to complete
Rendtime = time.time() - real_start
print('Full Runtime: ' + str(round(Rendtime, 2)) + ' Seconds\n\n')
# Prints out the average runtime per epoch.
print('average runtime per epoch is ' + str(round(np.mean(iter_time), 2)) + ' seconds')
```

The part of the output from running the code above is below:

Cesar Lopez
Individual Report
Machine Learning II



```
Currently on epoch 46
Test Accuracy is: 0.945625
Runtime: 16.33 Seconds


Currently on epoch 47
Test Accuracy is: 0.94695
Runtime: 16.82 Seconds


Currently on epoch 48
Test Accuracy is: 0.9468
Runtime: 16.8 Seconds


Currently on epoch 49
Test Accuracy is: 0.946325
Runtime: 16.8 Seconds


Currently on epoch 50
Test Accuracy is: 0.94625
Runtime: 16.65 Seconds
```

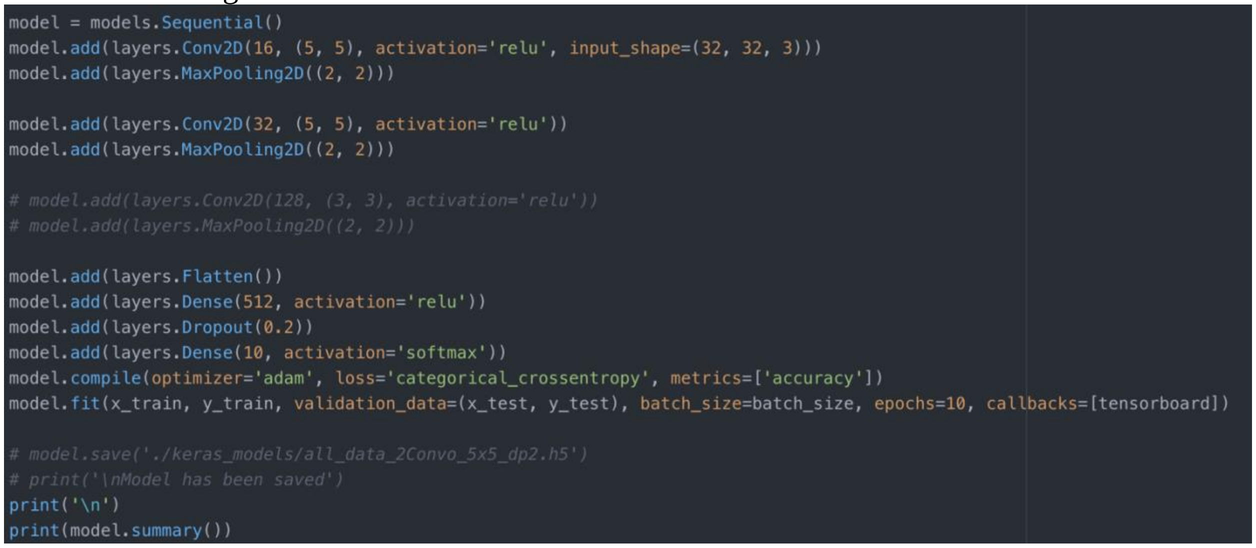
This framework is difficult to debug in because the graphs do not run until the session in initiated. One would have to use tensorboard to look at the graph and its parameter to see what is wrong with the code. My results using Tensorflow was about 95% after 10 epochs.

Much time was spent on setting up the architecture taking the shapes into account. Another issue I had was the bias. The project would have been better if I learned more tensorboard features. Image embedded is a great one that I wanted to use, but it had some input that I do not understand yet. This feature could illustrate how our model groups each label together.

Keras

Keras was very easy to do. I used the same class I to get the data I created for in my Tensorflow since they both require the same shape. The whole model was written with the little bit of code shown below.

Cesar Lopez
Individual Report
Machine Learning II



```
model = models.Sequential()
model.add(layers.Conv2D(16, (5, 5), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(32, (5, 5), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

# model.add(layers.Conv2D(128, (3, 3), activation='relu'))
# model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(10, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, validation_data=(x_test, y_test), batch_size=batch_size, epochs=10, callbacks=[tensorboard])

# model.save('./keras_models/all_data_2Convo_5x5_dp2.h5')
# print('\nModel has been saved')
print('\n')
print(model.summary())
```

Sources I used:

https://stackabuse.com/download-files-with-python/
https://docs.scipy.org/doc/scipy/reference/tutorial/io.html
https://www.tensorflow.org/tutorials/