Thomas Martin, Tran Huynh, Sahil Sutaria
CS 663
12-2-19

Project 3 LSTM Documentation: Part 1 - Training

Link to YouTube Video: https://www.youtube.com/watch?v=cOiZSM-d-HY&feature=youtu.be

Link to GitHub Repository: https://github.com/tmartin293/CS663_Crosswalk_Detection

Installation Instructions:

- All Juypter Notebooks were tested using a Python 3.7.1 kernel and TensorFlow 2.0
- Please pip install or conda install all required packages prior to testing the Jupyter Notebook or the Python script: TensorFlow 2.0, sklearn, and numpy

Screenshots:

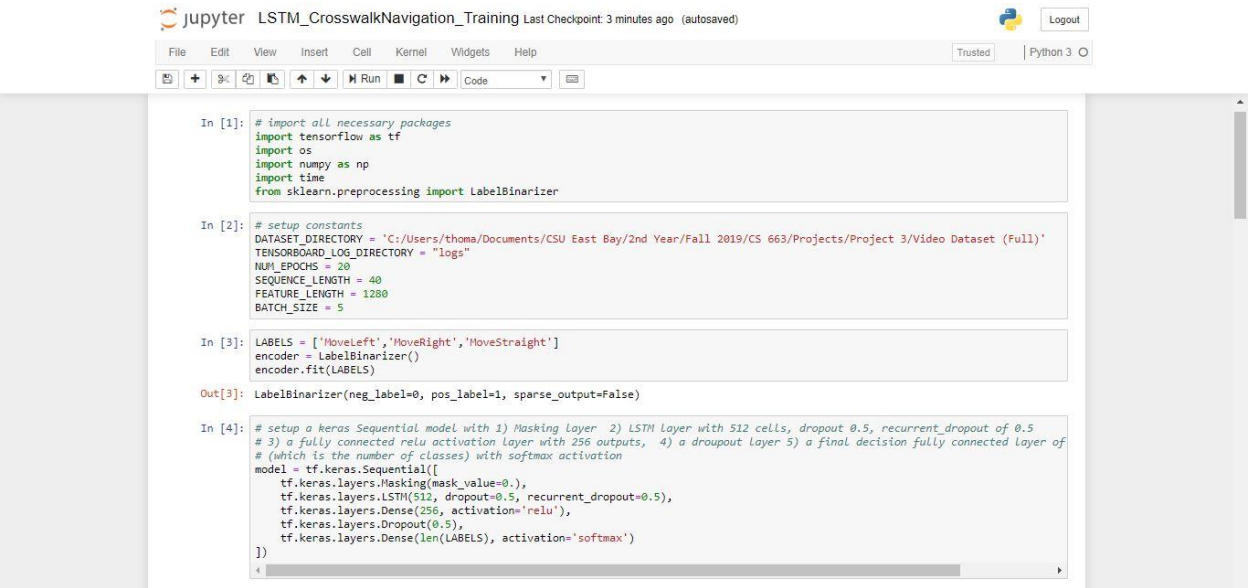


Figure 1: Cells 1-4 of the Jupyter Notebook

jupyter LSTM_CrosswalkNavigation_Training Last Checkpoint: 4 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [5]:
```
# compile the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy', 'top_k_categorical_accuracy'])
```

In [6]:
```
# get all of the filenames and paths for the training and validation datasets
training_file = os.path.join(DATASET_DIRECTORY,'TrainList.txt')
validation_file = os.path.join(DATASET_DIRECTORY,'ValidateList.txt')

with open(training_file) as f:
    training_list = [row.strip() for row in list(f)]

with open(validation_file) as f:
    validation_list = [row.strip() for row in list(f)]
```

In [7]:
```
# function to create a generator that will yield a numpy array of video
# features and the encoded class label
def make_generator(file_list):
    def generator():
        np.random.shuffle(file_list)
        for path in file_list:
            class_label = os.path.basename(os.path.dirname(path))
            features = np.load(path)

            padded_sequence = np.zeros((SEQUENCE_LENGTH, FEATURE_LENGTH))
            padded_sequence[0:len(features)] = np.array(features)

            transformed_label = encoder.transform([class_label])
            yield padded_sequence, transformed_label[0]
    return generator
```

Figure 2: Cells 5-7 of the Jupyter Notebook

jupyter LSTM_CrosswalkNavigation_Training Last Checkpoint: 4 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

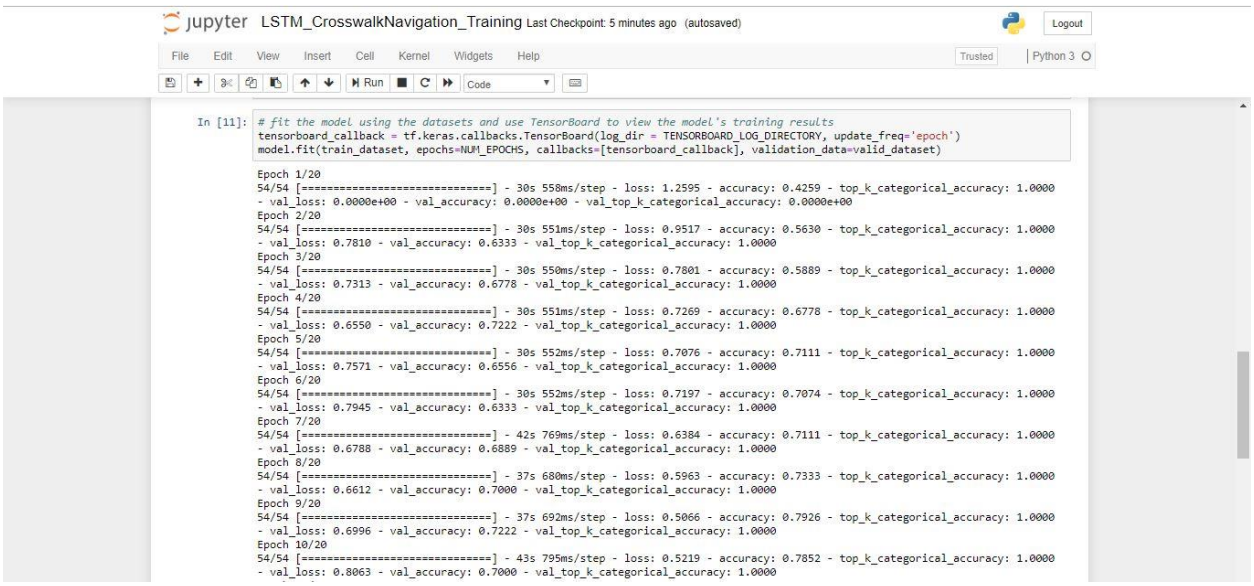
Run Code

In [8]:
```
# setup the training and validation datasets
train_dataset = tf.data.Dataset.from_generator(make_generator(training_list),
                 output_types=(tf.float32, tf.int16),
                 output_shapes=((SEQUENCE_LENGTH, FEATURE_LENGTH), (len(LABELS))))
train_dataset = train_dataset.batch(BATCH_SIZE,drop_remainder=True).prefetch(tf.data.experimental.AUTOTUNE)

valid_dataset = tf.data.Dataset.from_generator(make_generator(validation_list),
                 output_types=(tf.float32, tf.int16),
                 output_shapes=((SEQUENCE_LENGTH, FEATURE_LENGTH), (len(LABELS))))
valid_dataset = valid_dataset.batch(BATCH_SIZE,drop_remainder=True).prefetch(tf.data.experimental.AUTOTUNE)
```

In [10]:
```
# create the directory for the Tensorboard logging files
if not os.path.exists(TENSORBOARD_LOG_DIRECTORY):
    os.mkdir(TENSORBOARD_LOG_DIRECTORY)
```

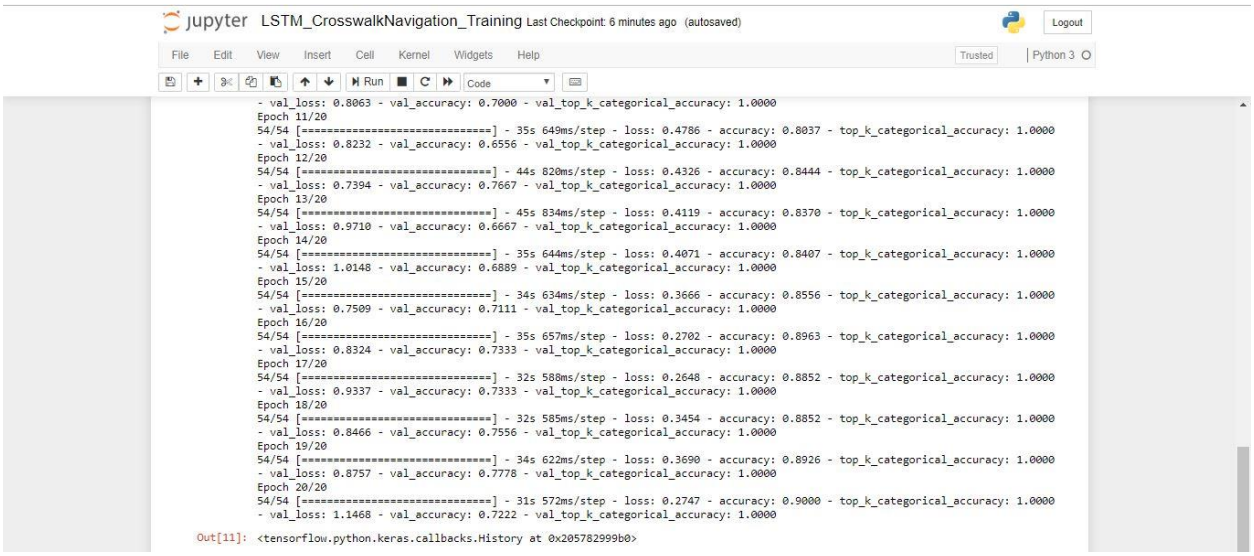
Figure 3: Cells 8 and 9 of the Jupyter Notebook



```
In [11]: # fit the model using the datasets and use TensorBoard to view the model's training results
         tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir = TENSORBOARD_LOG_DIRECTORY, update_freq='epoch')
         model.fit(train_dataset, epochs=NUM_EPOCHS, callbacks=[tensorboard_callback], validation_data=valid_dataset)

         Epoch 1/20
         54/54 [==============================] - 30s 558ms/step - loss: 1.2595 - accuracy: 0.4259 - top_k_categorical_accuracy: 1.0000
         - val_loss: 0.0000e+00 - val_accuracy: 0.0000e+00 - val_top_k_categorical_accuracy: 0.0000e+00
         Epoch 2/20
         54/54 [==============================] - 30s 551ms/step - loss: 0.9517 - accuracy: 0.5630 - top_k_categorical_accuracy: 1.0000
         - val_loss: 0.7810 - val_accuracy: 0.6333 - val_top_k_categorical_accuracy: 1.0000
         Epoch 3/20
         54/54 [==============================] - 30s 550ms/step - loss: 0.7801 - accuracy: 0.5889 - top_k_categorical_accuracy: 1.0000
         - val_loss: 0.7313 - val_accuracy: 0.6778 - val_top_k_categorical_accuracy: 1.0000
         Epoch 4/20
         54/54 [==============================] - 30s 551ms/step - loss: 0.7269 - accuracy: 0.6778 - top_k_categorical_accuracy: 1.0000
         - val_loss: 0.6550 - val_accuracy: 0.7222 - val_top_k_categorical_accuracy: 1.0000
         Epoch 5/20
         54/54 [==============================] - 30s 552ms/step - loss: 0.7076 - accuracy: 0.7111 - top_k_categorical_accuracy: 1.0000
         - val_loss: 0.7571 - val_accuracy: 0.6556 - val_top_k_categorical_accuracy: 1.0000
         Epoch 6/20
         54/54 [==============================] - 30s 552ms/step - loss: 0.7197 - accuracy: 0.7074 - top_k_categorical_accuracy: 1.0000
         - val_loss: 0.7945 - val_accuracy: 0.6333 - val_top_k_categorical_accuracy: 1.0000
         Epoch 7/20
         54/54 [==============================] - 42s 769ms/step - loss: 0.6384 - accuracy: 0.7111 - top_k_categorical_accuracy: 1.0000
         - val_loss: 0.6788 - val_accuracy: 0.6889 - val_top_k_categorical_accuracy: 1.0000
         Epoch 8/20
         54/54 [==============================] - 37s 680ms/step - loss: 0.5963 - accuracy: 0.7333 - top_k_categorical_accuracy: 1.0000
         - val_loss: 0.6612 - val_accuracy: 0.7000 - val_top_k_categorical_accuracy: 1.0000
         Epoch 9/20
         54/54 [==============================] - 37s 692ms/step - loss: 0.5066 - accuracy: 0.7926 - top_k_categorical_accuracy: 1.0000
         - val_loss: 0.6996 - val_accuracy: 0.7222 - val_top_k_categorical_accuracy: 1.0000
         Epoch 10/20
         54/54 [==============================] - 43s 795ms/step - loss: 0.5219 - accuracy: 0.7852 - top_k_categorical_accuracy: 1.0000
         - val_loss: 0.8063 - val_accuracy: 0.7000 - val_top_k_categorical_accuracy: 1.0000
```

Figure 4: Cells 10 of the Jupyter Notebook Showing the LSTM Model Being Fit to the Dataset



```
         - val_loss: 0.8063 - val_accuracy: 0.7000 - val_top_k_categorical_accuracy: 1.0000
         Epoch 11/20
         54/54 [==============================] - 35s 649ms/step - loss: 0.4786 - accuracy: 0.8037 - top_k_categorical_accuracy: 1.0000
         - val_loss: 0.8232 - val_accuracy: 0.6556 - val_top_k_categorical_accuracy: 1.0000
         Epoch 12/20
         54/54 [==============================] - 44s 820ms/step - loss: 0.4326 - accuracy: 0.8444 - top_k_categorical_accuracy: 1.0000
         - val_loss: 0.7394 - val_accuracy: 0.7667 - val_top_k_categorical_accuracy: 1.0000
         Epoch 13/20
         54/54 [==============================] - 45s 834ms/step - loss: 0.4119 - accuracy: 0.8370 - top_k_categorical_accuracy: 1.0000
         - val_loss: 0.9710 - val_accuracy: 0.6667 - val_top_k_categorical_accuracy: 1.0000
         Epoch 14/20
         54/54 [==============================] - 35s 644ms/step - loss: 0.4071 - accuracy: 0.8407 - top_k_categorical_accuracy: 1.0000
         - val_loss: 1.0148 - val_accuracy: 0.6889 - val_top_k_categorical_accuracy: 1.0000
         Epoch 15/20
         54/54 [==============================] - 34s 634ms/step - loss: 0.3666 - accuracy: 0.8556 - top_k_categorical_accuracy: 1.0000
         - val_loss: 0.7509 - val_accuracy: 0.7111 - val_top_k_categorical_accuracy: 1.0000
         Epoch 16/20
         54/54 [==============================] - 35s 657ms/step - loss: 0.2702 - accuracy: 0.8963 - top_k_categorical_accuracy: 1.0000
         - val_loss: 0.8324 - val_accuracy: 0.7333 - val_top_k_categorical_accuracy: 1.0000
         Epoch 17/20
         54/54 [==============================] - 32s 588ms/step - loss: 0.2648 - accuracy: 0.8852 - top_k_categorical_accuracy: 1.0000
         - val_loss: 0.9337 - val_accuracy: 0.7333 - val_top_k_categorical_accuracy: 1.0000
         Epoch 18/20
         54/54 [==============================] - 32s 585ms/step - loss: 0.3454 - accuracy: 0.8852 - top_k_categorical_accuracy: 1.0000
         - val_loss: 0.8466 - val_accuracy: 0.7556 - val_top_k_categorical_accuracy: 1.0000
         Epoch 19/20
         54/54 [==============================] - 34s 622ms/step - loss: 0.3690 - accuracy: 0.8926 - top_k_categorical_accuracy: 1.0000
         - val_loss: 0.8757 - val_accuracy: 0.7778 - val_top_k_categorical_accuracy: 1.0000
         Epoch 20/20
         54/54 [==============================] - 31s 572ms/step - loss: 0.2747 - accuracy: 0.9000 - top_k_categorical_accuracy: 1.0000
         - val_loss: 1.1468 - val_accuracy: 0.7222 - val_top_k_categorical_accuracy: 1.0000

Out[11]: <tensorflow.python.keras.callbacks.History at 0x205782999b0>
```

Figure 5: Cell 10 of the Jupyter Notebook Showing the LSTM Model During Fitting

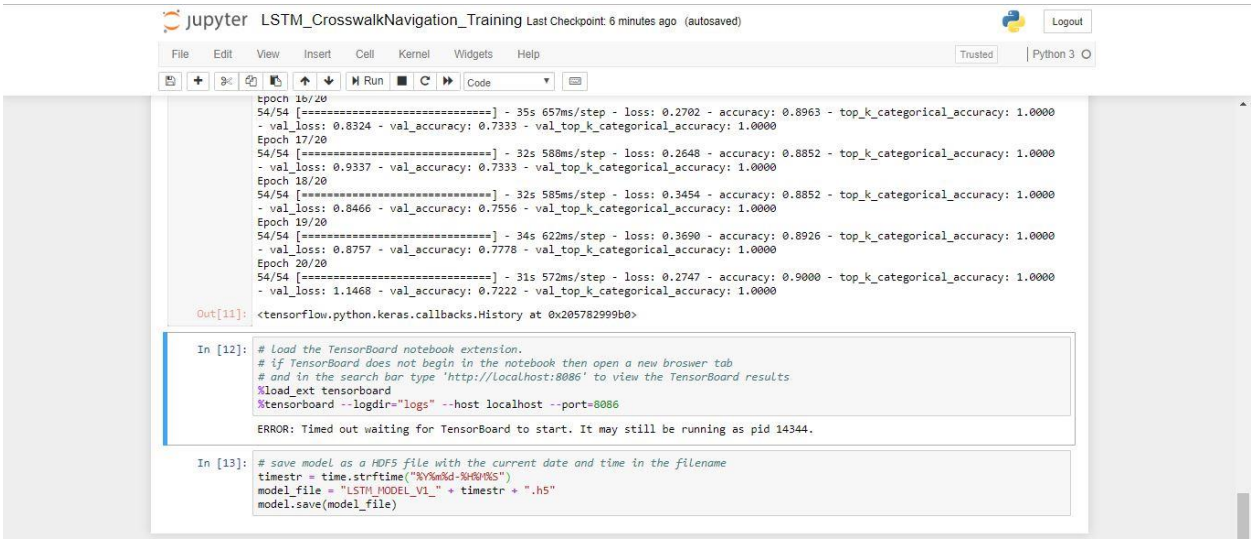


Figure 6: Cells 11 and 12 of the Jupyter Notebook Showing the Model Being Saved and TensorBoard Being Called

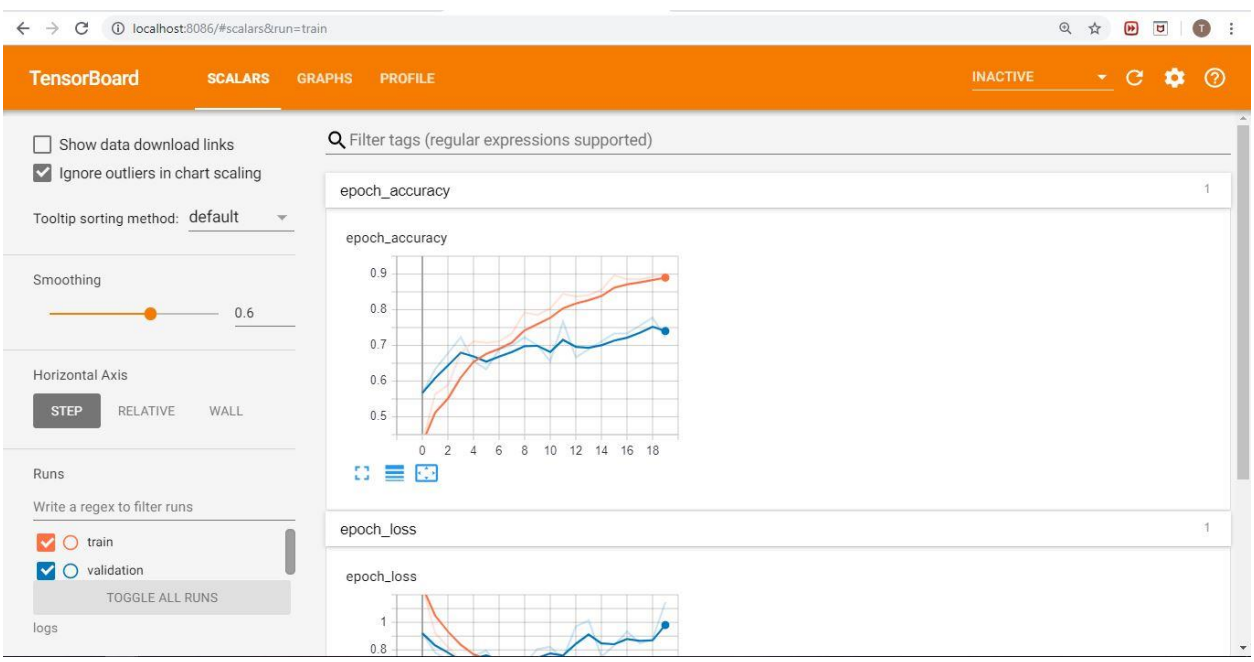


Figure 7: TensorBoard Epoch Accuracy Results for the LSTM Model with 20 Epochs

Figure 8: TensorBoard Epoch Loss Results for the LSTM Model with 20 Epochs

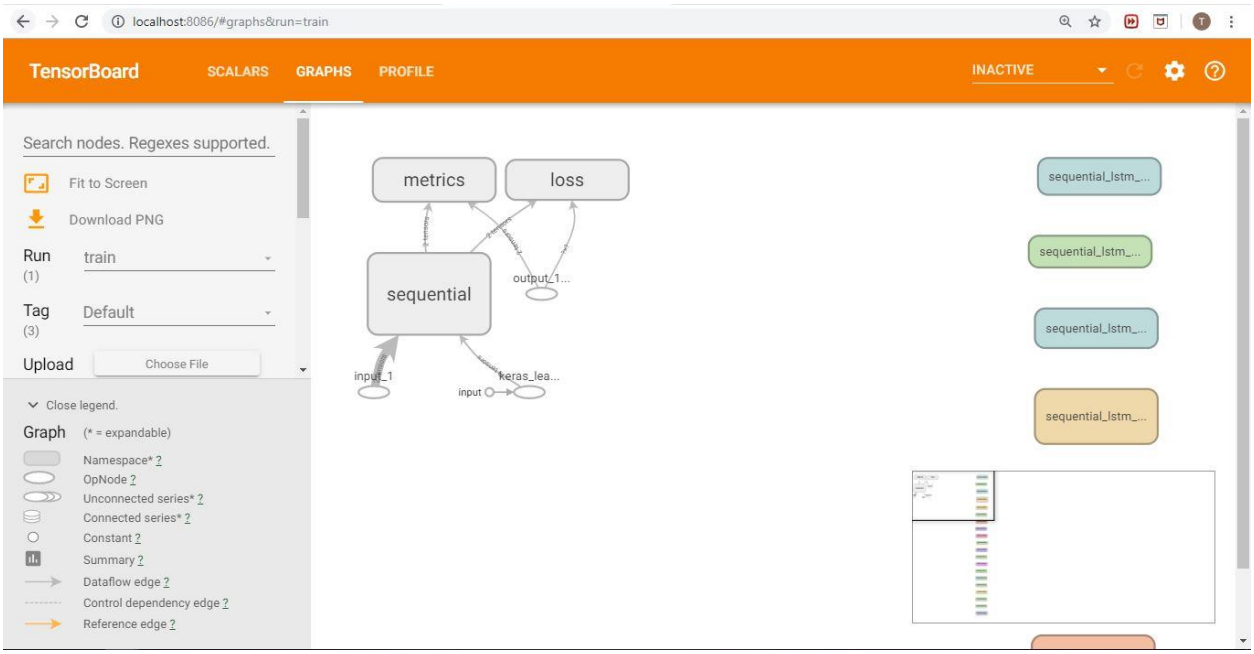


Figure 9: TensorBoard Model Graph for the LSTM Model

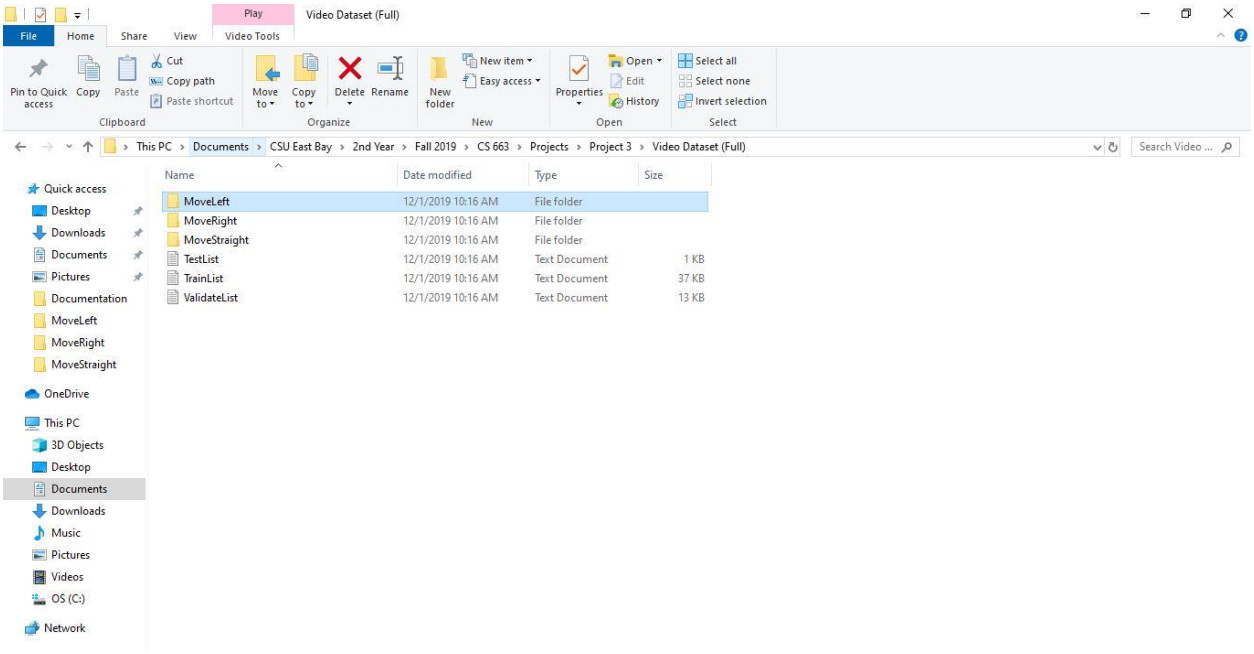


Figure 10: Directory Structure of Video Dataset After Running the Feature Extraction Notebook

LSTM Model Training Analysis:

The TensorFlow LSTM model used for this project was trained using a 75/25 train-validate dataset split after removing two video files for each class for testing the model's accuracy. The LSTM model was fitted with 20 epochs and was able to achieve a maximum epoch accuracy score of 90% with the training data and 74% with the validation data.