

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

Project 3 LSTM Documentation: Part 2 - Feature Extraction

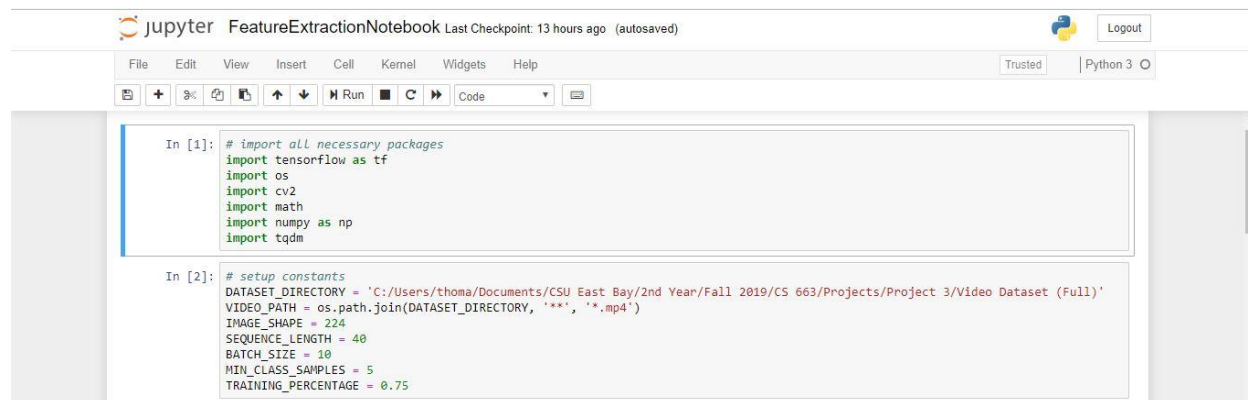
Link to YouTube Video: <https://www.youtube.com/watch?v=mHMkWBMC-dE&feature=youtu.be>

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

Installation Instructions:

- All Jupyter 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, cv2 (OpenCV), tqdm, and numpy

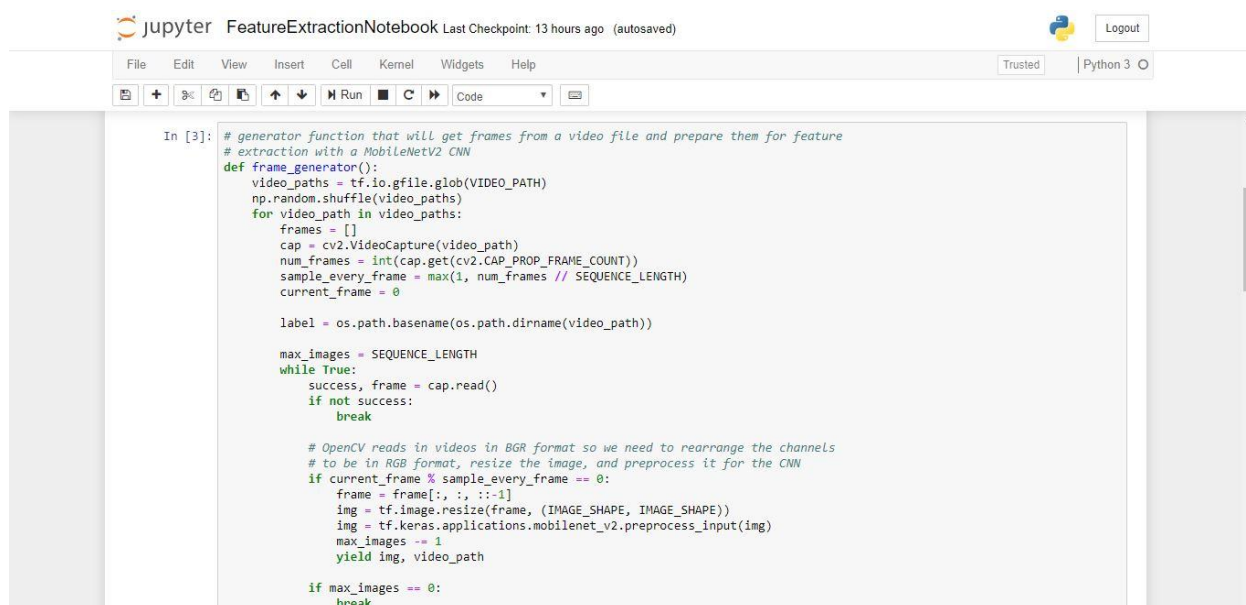
Screenshots:



```
In [1]: # import all necessary packages
import tensorflow as tf
import os
import cv2
import math
import numpy as np
import tqdm

In [2]: # setup constants
DATASET_DIRECTORY = 'C:/Users/thoma/Documents/CSU East Bay/2nd Year/Fall 2019/CS 663/Projects/Project 3/Video Dataset (Full)'
VIDEO_PATH = os.path.join(DATASET_DIRECTORY, '**', '*.mp4')
IMAGE_SHAPE = 224
SEQUENCE_LENGTH = 40
BATCH_SIZE = 10
MIN_CLASS_SAMPLES = 5
TRAINING_PERCENTAGE = 0.75
```

Figure 1: Cells 1 and 2 of the Jupyter Notebook



The screenshot shows a Jupyter Notebook interface with the title 'FeatureExtractionNotebook'. The top bar indicates 'Last Checkpoint: 13 hours ago (autosaved)' and includes a 'Logout' button. The notebook has a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations and execution. The code in Cell 3 is a generator function named 'frame_generator' that processes video frames for feature extraction. It uses OpenCV for video capture and TensorFlow/Keras for image preprocessing. The function yields preprocessed images and their corresponding video paths.

```
In [3]: # generator function that will get frames from a video file and prepare them for feature
# extraction with a MobileNetV2 CNN
def frame_generator():
    video_paths = tf.io.gfile.glob(VIDEO_PATH)
    np.random.shuffle(video_paths)
    for video_path in video_paths:
        frames = []
        cap = cv2.VideoCapture(video_path)
        num_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
        sample_every_frame = max(1, num_frames // SEQUENCE_LENGTH)
        current_frame = 0

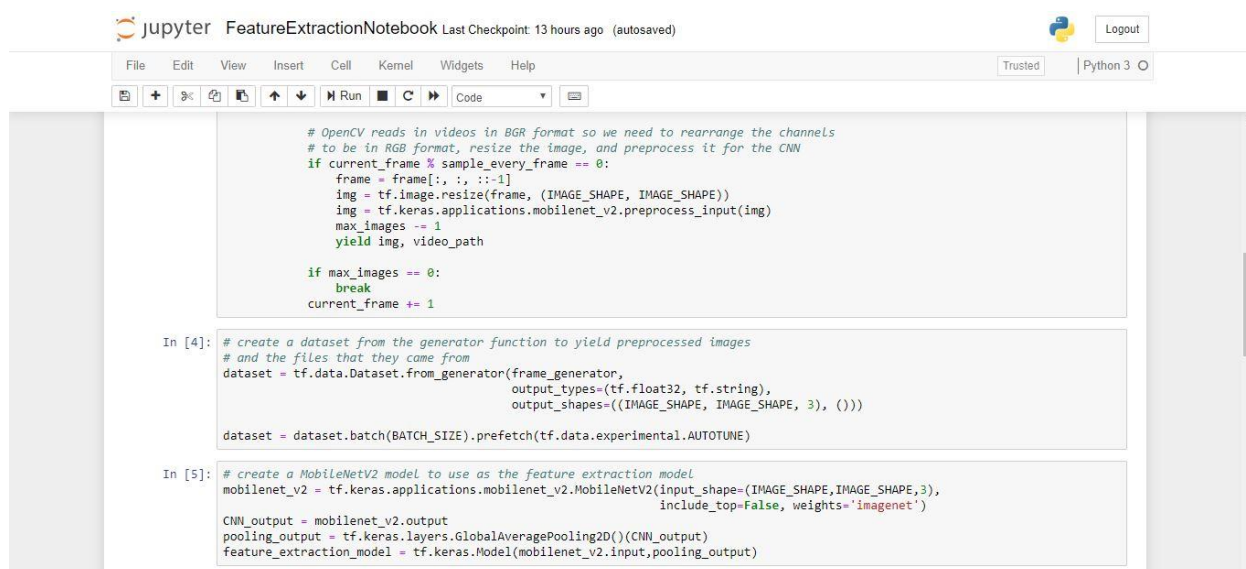
        label = os.path.basename(os.path.dirname(video_path))

        max_images = SEQUENCE_LENGTH
        while True:
            success, frame = cap.read()
            if not success:
                break

            # OpenCV reads in videos in BGR format so we need to rearrange the channels
            # to be in RGB format, resize the image, and preprocess it for the CNN
            if current_frame % sample_every_frame == 0:
                frame = frame[:, :, ::-1]
                img = tf.image.resize(frame, (IMAGE_SHAPE, IMAGE_SHAPE))
                img = tf.keras.applications.mobilenet_v2.preprocess_input(img)
                max_images -= 1
                yield img, video_path

            if max_images == 0:
                break
```

Figure 2: Cell 3 of the Jupyter Notebook



The screenshot shows the continuation of the Jupyter Notebook. Cell 4 creates a dataset from the generator function, and Cell 5 creates a MobileNetV2 model for feature extraction. The code in Cell 4 uses 'tf.data.Dataset.from_generator' to create a dataset from the 'frame_generator' function, specifying output types and shapes. The code in Cell 5 creates a 'mobilenet_v2' model using 'tf.keras.applications.mobilenet_v2.MobileNetV2', and then extracts features by passing the input through the model's layers and pooling.

```
In [4]: # OpenCV reads in videos in BGR format so we need to rearrange the channels
# to be in RGB format, resize the image, and preprocess it for the CNN
if current_frame % sample_every_frame == 0:
    frame = frame[:, :, ::-1]
    img = tf.image.resize(frame, (IMAGE_SHAPE, IMAGE_SHAPE))
    img = tf.keras.applications.mobilenet_v2.preprocess_input(img)
    max_images -= 1
    yield img, video_path

if max_images == 0:
    break
current_frame += 1

In [4]: # create a dataset from the generator function to yield preprocessed images
# and the files that they came from
dataset = tf.data.Dataset.from_generator(frame_generator,
                                         output_types=(tf.float32, tf.string),
                                         output_shapes=((IMAGE_SHAPE, IMAGE_SHAPE, 3), ()))

dataset = dataset.batch(BATCH_SIZE).prefetch(tf.data.experimental.AUTOTUNE)

In [5]: # create a MobileNetV2 model to use as the feature extraction model
mobilenet_v2 = tf.keras.applications.mobilenet_v2.MobileNetV2(input_shape=(IMAGE_SHAPE, IMAGE_SHAPE, 3),
                                                                include_top=False, weights='imagenet')

CNN_output = mobilenet_v2.output
pooling_output = tf.keras.layers.GlobalAveragePooling2D()(CNN_output)
feature_extraction_model = tf.keras.Model(mobilenet_v2.input, pooling_output)
```

Figure 3: Cells 3-5 of the Jupyter Notebook

```

In [6]: current_path = None
all_features = []

# go through the dataset and use the MobileNetV2 model to
# extract the features (1x1280) for each desired frame of
# the videos
for img, batch_paths in tqdm.tqdm(dataset):
    # extract the features
    batch_features = feature_extraction_model(img)

    # reshape the tensor
    batch_features = tf.reshape(batch_features, (batch_features.shape[0], -1))

    for features, path in zip(batch_features.numpy(), batch_paths.numpy()):
        # save the features corresponding to a video file when a new video file is reached
        if path != current_path and current_path is not None:
            output_path = current_path.decode().replace('.mp4', '.npz')
            np.save(output_path, all_features)
            all_features = []

            current_path = path
            all_features.append(features)

# save the final file after exiting the for loop
output_path = current_path.decode().replace('.mp4', '.npz')
np.save(output_path, all_features)
all_features = []

1479it [21:06, 1.12it/s]

```

Figure 4: Cell 6 of the Jupyter Notebook

```

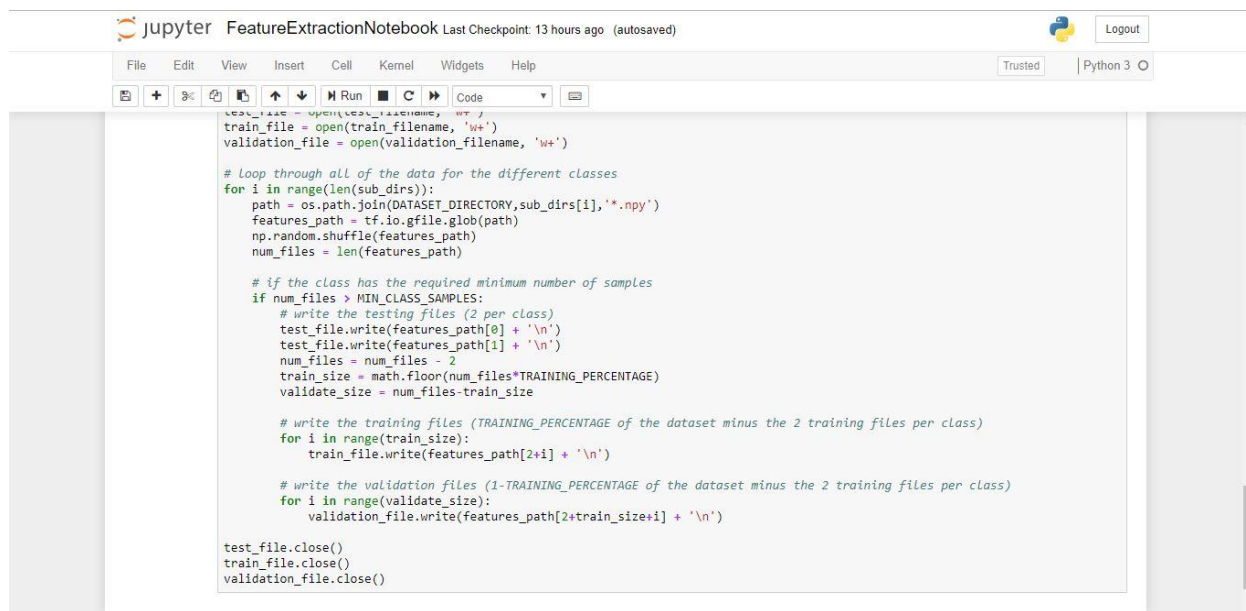
In [7]: # get all of the subdirectories (1 per class) in the dataset directory
sub_dirs = []
for root, dirs, files in os.walk(DATASET_DIRECTORY):
    sub_dirs.append(dirs)
sub_dirs = sub_dirs[0]

In [8]: # create .txt files containing all of the video feature files that will be
# used for model training, validation, and testing
test_filename = os.path.join(DATASET_DIRECTORY, 'TestList.txt')
train_filename = os.path.join(DATASET_DIRECTORY, 'TrainList.txt')
validation_filename = os.path.join(DATASET_DIRECTORY, 'Validatelist.txt')
test_file = open(test_filename, 'w+')
train_file = open(train_filename, 'w+')
validation_file = open(validation_filename, 'w+')

# Loop through all of the data for the different classes
for i in range(len(sub_dirs)):
    path = os.path.join(DATASET_DIRECTORY, sub_dirs[i], '*.npz')
    features_path = tf.io.gfile.glob(path)
    np.random.shuffle(features_path)
    num_files = len(features_path)

```

Figure 5: Cells 7 and 8 of the Jupyter Notebook



```

test_file = open(test_filename, 'w')
train_file = open(train_filename, 'w+')
validation_file = open(validation_filename, 'w+')

# Loop through all of the data for the different classes
for i in range(len(sub_dirs)):
    path = os.path.join(DATASET_DIRECTORY, sub_dirs[i], '*.npy')
    features_path = tf.io.gfile.glob(path)
    np.random.shuffle(features_path)
    num_files = len(features_path)

    # if the class has the required minimum number of samples
    if num_files > MIN_CLASS_SAMPLES:
        # write the testing files (2 per class)
        test_file.write(features_path[0] + '\n')
        test_file.write(features_path[1] + '\n')
        num_files = num_files - 2
        train_size = math.floor(num_files*TRAINING_PERCENTAGE)
        validate_size = num_files - train_size

        # write the training files (TRAINING_PERCENTAGE of the dataset minus the 2 training files per class)
        for i in range(train_size):
            train_file.write(features_path[2+i] + '\n')

        # write the validation files (1-TRAINING_PERCENTAGE of the dataset minus the 2 training files per class)
        for i in range(validate_size):
            validation_file.write(features_path[2+train_size+i] + '\n')

test_file.close()
train_file.close()
validation_file.close()

```

Figure 6: Cell 8 of the Jupyter Notebook

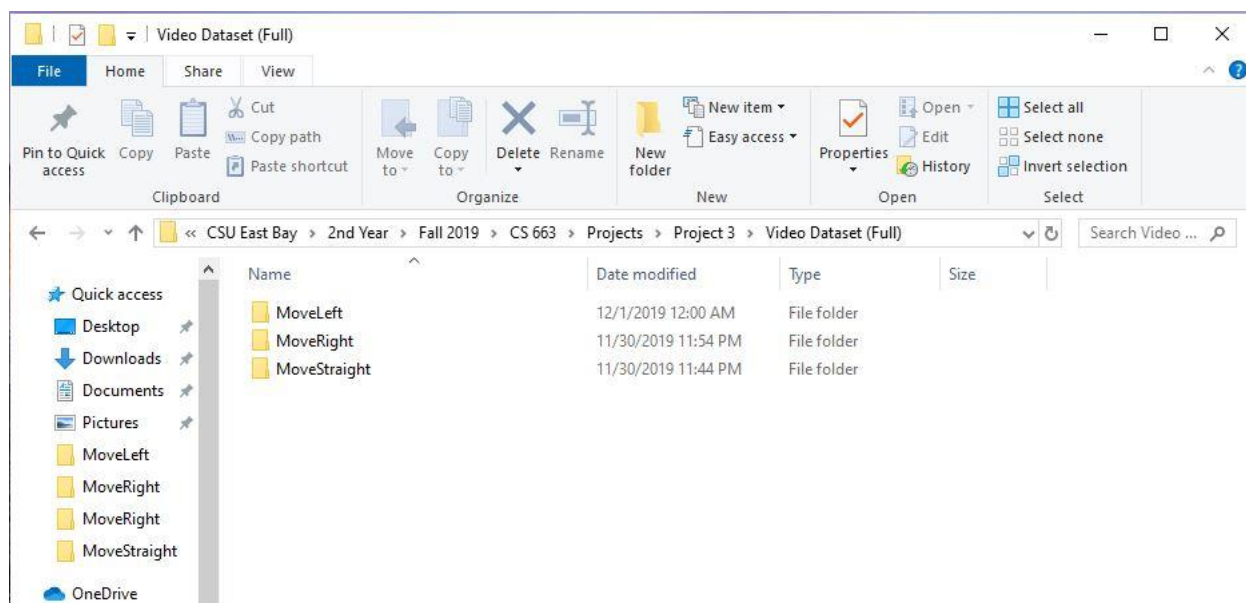


Figure 7: Directory Containing the Three Sub-Directories for the Three Classes (MoveLeft, MoveRight, and MoveStraight)

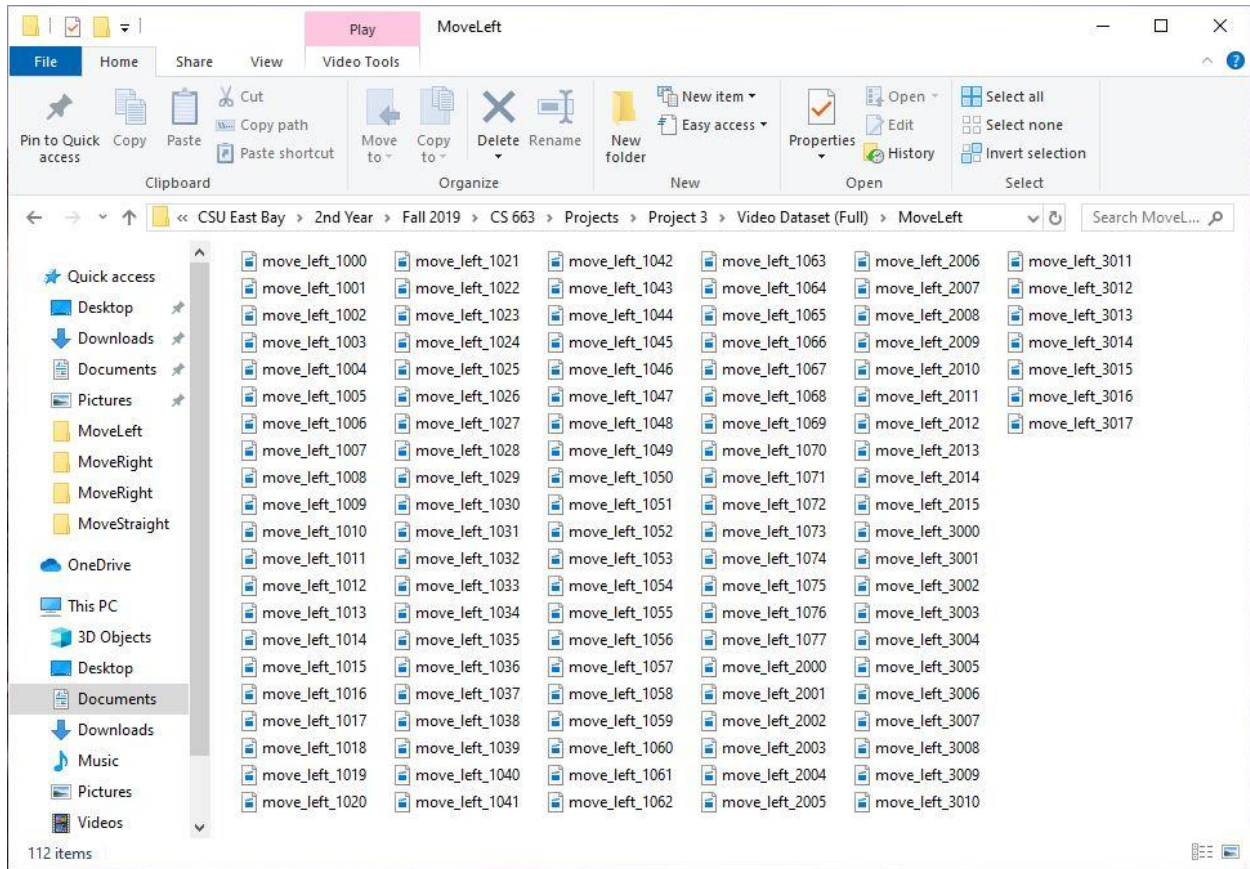


Figure 8: MoveLeft Directory Containing 112 MoveLeft Videos

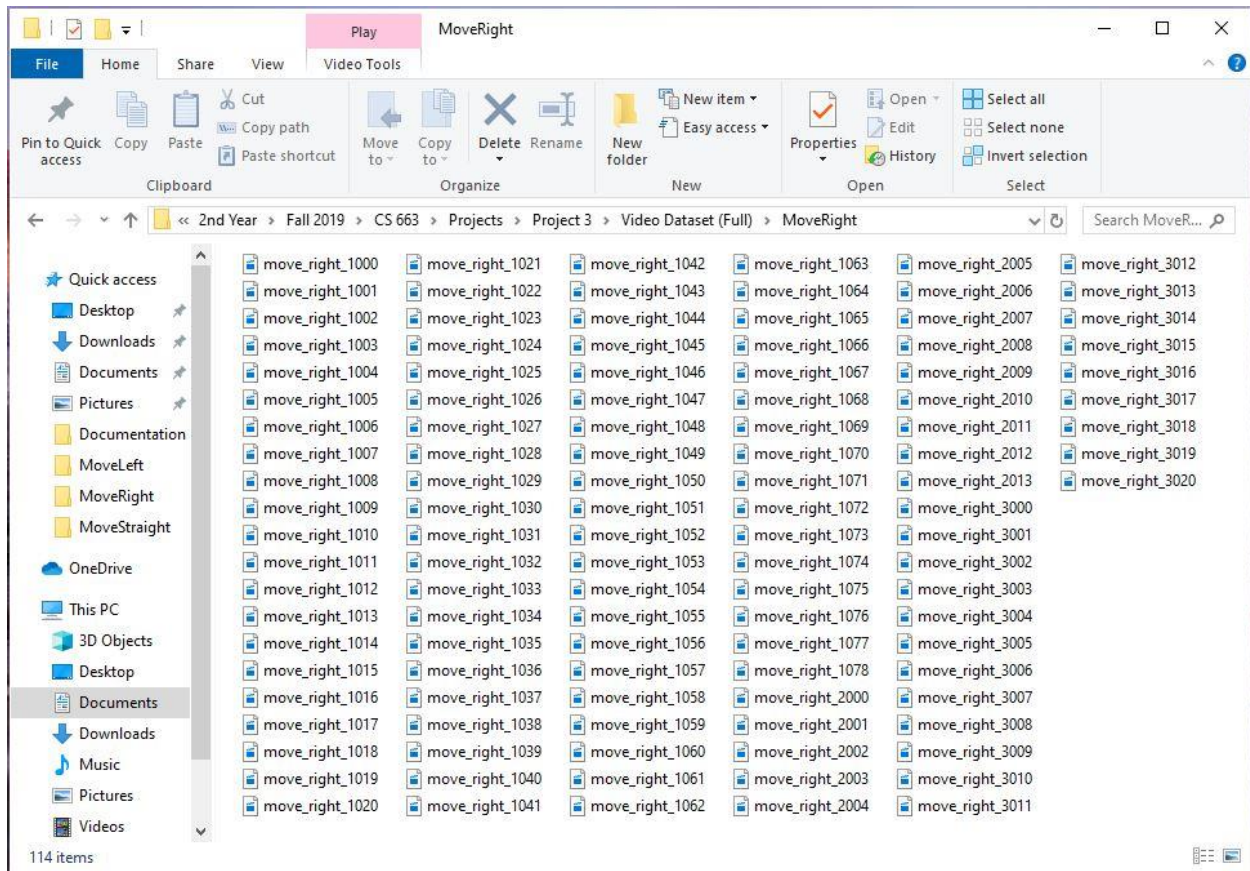


Figure 9: MoveRight Directory Containing 114 MoveRight Videos

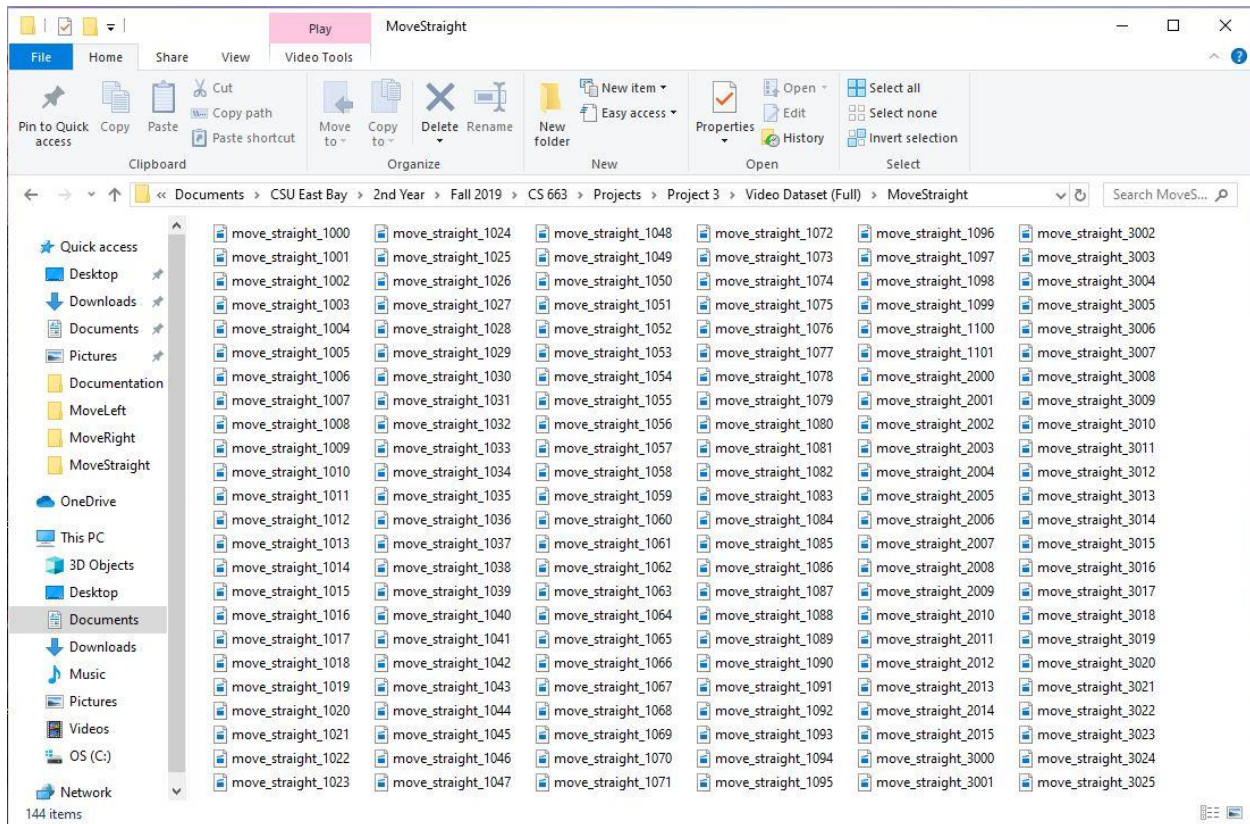


Figure 10: MoveStraight Directory Containing 144 MoveStraight Videos

Feature Extraction Analysis:

In order to perform the feature extraction a MobileNetV2 CNN model with its softmax layer removed was utilized. This CNN was setup to take images of size $224 \times 224 \times 3$ as inputs and output a feature vector of size 1×1280 . The video dataset for this project consists of 112 MoveLeft videos, 114 MoveRight videos, and 144 MoveStraight videos. Each of the videos in the dataset ranges in duration from 2 to 10 seconds.