In [ ]:

```
!pip install wandb
import wandb
wandb.init()
```

```
Collecting wandb
  Downloading https://files.pythonhosted.org/packages/6c/48/b199e2b3b341ac842108c5db
4956091dd75d961cfa77aceb033e99cac20f/wandb-0.10.31-py2.py3-none-any.whl (1.8MB)
     |████████████████████████████████| 1.8MB 14.5MB/s
Collecting sentry-sdk>=0.4.0
  Downloading https://files.pythonhosted.org/packages/1c/4a/a54b254f67d8f4052338d54e
be90126f200693440a93ef76d254d581e3ec/sentry_sdk-1.1.0-py2.py3-none-any.whl (131kB)
     |████████████████████████████████| 133kB 61.6MB/s
Requirement already satisfied: protobuf>=3.12.0 in /usr/local/lib/python3.7/dist-pac
kages (from wandb) (3.12.4)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.7/dist-packages (fro
m wandb) (3.13)
Requirement already satisfied: psutil>=5.0.0 in /usr/local/lib/python3.7/dist-packag
es (from wandb) (5.4.8)
Collecting shortuuid>=0.5.0
  Downloading https://files.pythonhosted.org/packages/25/a6/2ecc1daa6a304e7f1b216f08
96b26156b78e7c38e1211e9b798b4716c53d/shortuuid-1.0.1-py3-none-any.whl
Requirement already satisfied: Click>=7.0 in /usr/local/lib/python3.7/dist-packages
(from wandb) (7.1.2)
Requirement already satisfied: requests<3,>=2.0.0 in /usr/local/lib/python3.7/dist-p
ackages (from wandb) (2.23.0)
Collecting docker-pycreds>=0.4.0
  Downloading https://files.pythonhosted.org/packages/f5/e8/f6bd1eee09314e7e6dee49cb
e2c5e22314ccdb38db16c9fc72d2fa80d054/docker_pycreds-0.4.0-py2.py3-none-any.whl
Collecting pathtools
  Downloading https://files.pythonhosted.org/packages/e7/7f/470d6fcdf23f9f3518f6b0b7
6be9df16dcc8630ad409947f8be2eb0ed13a/pathtools-0.1.2.tar.gz
Collecting GitPython>=1.0.0
  Downloading https://files.pythonhosted.org/packages/27/da/6f6224fdfc47dab57881fe20
c0d1bc3122be290198ba0bf26a953a045d92/GitPython-3.1.17-py3-none-any.whl (166kB)
     |████████████████████████████████| 174kB 56.4MB/s
Collecting subprocess32>=3.5.3
  Downloading https://files.pythonhosted.org/packages/32/c8/564be4d12629b912ea431f1a
50eb8b3b9d00f1a0b1ceff17f266be190007/subprocess32-3.5.4.tar.gz (97kB)
     |████████████████████████████████| 102kB 14.5MB/s
Collecting configparser>=3.8.1
  Downloading https://files.pythonhosted.org/packages/fd/01/ff260a18caaf4457eb028c96
eeb405c4a230ca06c8ec9c1379f813caa52e/configparser-5.0.2-py3-none-any.whl
Requirement already satisfied: six>=1.13.0 in /usr/local/lib/python3.7/dist-packages
(from wandb) (1.15.0)
Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/lib/python3.7/di
st-packages (from wandb) (2.8.1)
Requirement already satisfied: promise<3,>=2.0 in /usr/local/lib/python3.7/dist-pack
ages (from wandb) (2.3)
Requirement already satisfied: urllib3>=1.10.0 in /usr/local/lib/python3.7/dist-pack
ages (from sentry-sdk>=0.4.0->wandb) (1.24.3)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (fr
om sentry-sdk>=0.4.0->wandb) (2020.12.5)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages
(from protobuf>=3.12.0->wandb) (56.1.0)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-pa
ckages (from requests<3,>=2.0.0->wandb) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-package
s (from requests<3,>=2.0.0->wandb) (2.10)
```

In [ ]:

```
import os          #Import required modules
import cv2
import math
import random
import numpy as np
import datetime as dt
import tensorflow as tf

import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import *
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.utils import plot_model
```

In [ ]:

```
seed_constant = 23   #Initialize random number generator
np.random.seed(seed_constant)
random.seed(seed_constant)
tf.random.set_seed(seed_constant)
```

In [ ]:

```
from google.colab import drive
drive.mount('/gdrive')
%cd /gdrive
```

```
Mounted at /gdrive
/gdrive
```

In [ ]:

```
%cd 'My Drive/'

%cd 'Action Recognition'
```

```
/gdrive/My Drive
/gdrive/My Drive/Action Recognition
```

In [ ]:

```
%ls
```

```
hmdb51/
hmdb512/
hmdb51_org.rar
hmdb51rars/
KTH/
Model___Date_Time_2021_05_11__18_40_06___Loss_0.46230462193489075___Accuracy_0.99260
41960716248.h5
Model___Date_Time_2021_05_11__19_57_16___Loss_0.003302493831142783___Accuracy_0.9992
708563804626.h5
Model___Date_Time_2021_05_11__23_01_06___Loss_0.001863537821918726___Accuracy_0.9994
791746139526.h5
Model___Date_Time_2021_05_11__23_34_47___Loss_0.005838708486407995___Accuracy_0.9990
624785423279.h5
Model___Date_Time_2021_05_14__18_33_00___Loss_0.005261874292045832___Accuracy_0.9986
458420753479.h5
Model___Date_Time_2021_05_14__19_02_48___Loss_0.0034059989266097546___Accuracy_0.999
1666674613953.h5
```

In [ ]:

```
#Constants

datasetName= 'hmdb51'              #Chosen Dataset
```

In…

```
image_height, image_width = 64, 64        #Set pixel values
images_per_class = 8000             #Set number of frames from each video clas
dataset_directory = "hmdb51"            #Set dataset name
classes_list = ["pullup", "punch", "dive", "fencing", "ride_bike", "golf"]   #Cho
model_output_size = len(classes_list)     #test
```

In…

```
def frames_extraction(video_path):  #helper function to extract frames from vide
    frames_list = []                #empty list for frames
    video_reader = cv2.VideoCapture(video_path) #Read frames from video
    while True:     #Iterate through frames
        success, frame = video_reader.read() #Whilst frames are available
        if not success:
            break
        resized_frame = cv2.resize(frame, (image_height, image_width))    #Resize
        normalized_frame = resized_frame / 255                            #Normal
        frames_list.append(normalized_frame)                              #Add to
    video_reader.release()                                                #Close
    return frames_list                                                     #Retur
```

I…

```
def create_dataset():    #Create dataset function
    temp_features = [] #empty list to hold each videos frames
    features = []  #final list of frames will be in this list
    labels = []  #Final list of labels will be in this list

    for class_index, class_name in enumerate(classes_list):  #Iterate through cho
        print(f'Extracting Data of Class: {class_name}')
        files_list = os.listdir(os.path.join(dataset_directory, class_name)) #Got
        for file_name in files_list:
            video_file_path = os.path.join(dataset_directory, class_name, file_nai
            frames = frames_extraction(video_file_path)   #Extract frames for curi
            temp_features.extend(frames)      #Add to temp frames list

        features.extend(random.sample(temp_features, images_per_class))       #Cho
        labels.extend([class_index] * images_per_class)                         #
        temp_features.clear()

    features = np.asarray(features)   #Convert both to numpy array
    labels = np.array(labels)
    return features, labels
```

In [ ]:

```
features, labels = create_dataset()   #Fetch data
```

```
Extracting Data of Class: pullup
Extracting Data of Class: punch
Extracting Data of Class: dive
Extracting Data of Class: fencing
Extracting Data of Class: ride_bike
Extracting Data of Class: golf
```

In [ ]:

```
(48000, 64, 64, 3)
(48000,)
```

In ...

```
one_hot_encoded_labels = to_categorical(labels) #convert labels into one-hot-enc
```

I...

```
features_train, features_test, labels_train, labels_test = train_test_split(featur
```

I...

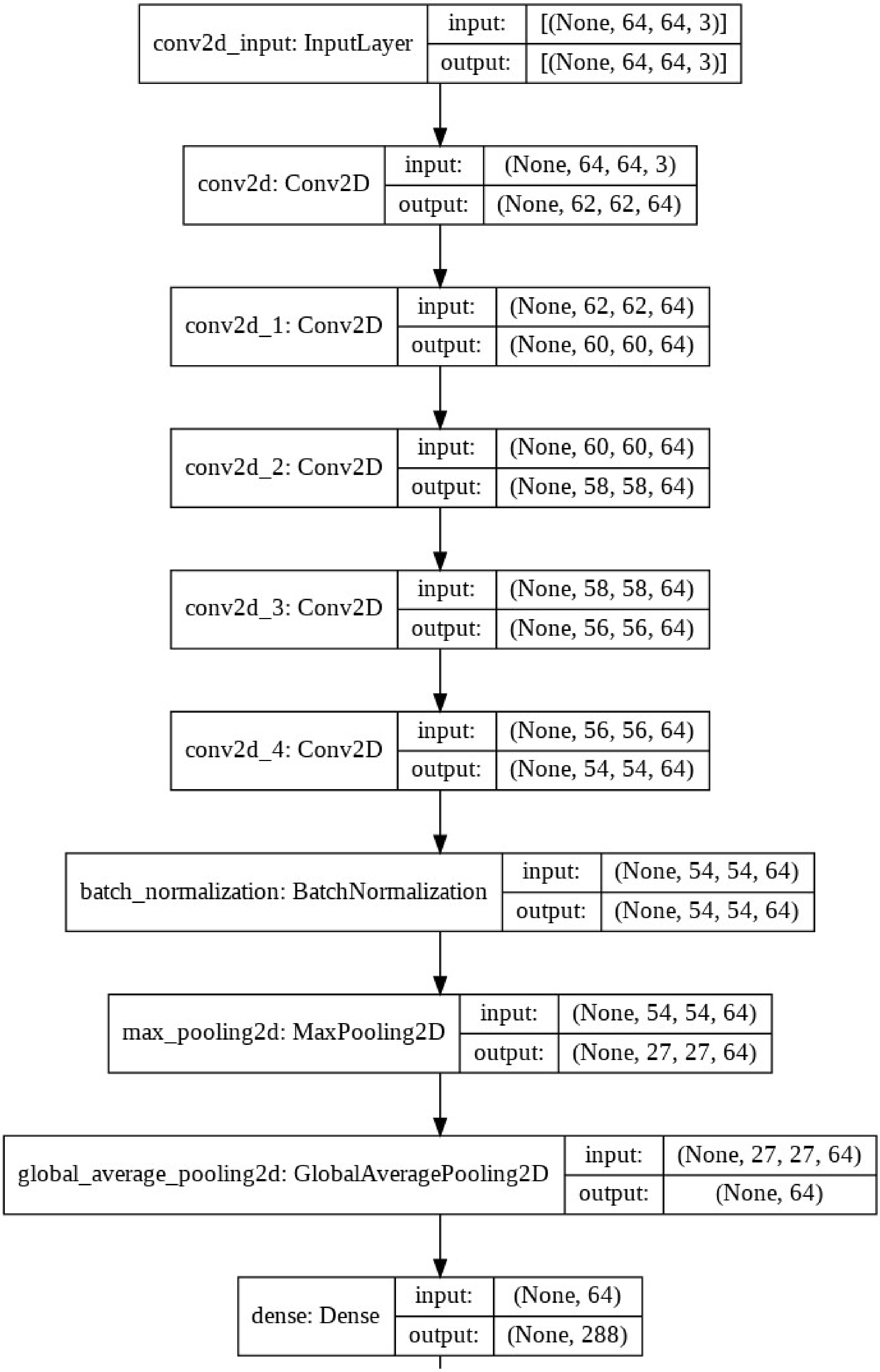
```
def create_model():  #Create NN
    model = Sequential() #Keras sequential model

    model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu', inpu
    model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu'))
    model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu'))
    model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu'))
    model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size = (2, 2)))
    model.add(GlobalAveragePooling2D())
    model.add(Dense(288, activation = 'relu'))
    model.add(Dense(288, activation = 'relu'))
    model.add(BatchNormalization())
    model.add(Dense(model_output_size, activation = 'softmax'))

    model.summary()  #Show model summary

    return model


model = create_model()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 62, 62, 64)        1792
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 60, 60, 64)        36928
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 58, 58, 64)        36928
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 56, 56, 64)        36928
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 54, 54, 64)        36928
_________________________________________________________________
batch_normalization (BatchNo (None, 54, 54, 64)        256
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 27, 27, 64)        0
_________________________________________________________________
global_average_pooling2d (Gl (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 288)               18720
_________________________________________________________________
dense_1 (Dense)              (None, 288)               83232
_________________________________________________________________
batch_normalization_1 (Batch (None, 288)               1152
_________________________________________________________________
```

In ...

```
plot_model(model,show_shapes = True, show_layer_names = True)  #Plot model diagr
```

Out...


conv2d_input: InputLayer
input: [(None, 64, 64, 3)]
output: [(None, 64, 64, 3)]
conv2d: Conv2D
input: (None, 64, 64, 3)
output: (None, 62, 62, 64)
conv2d_1: Conv2D
input: (None, 62, 62, 64)
output: (None, 60, 60, 64)
conv2d_2: Conv2D
input: (None, 60, 60, 64)
output: (None, 58, 58, 64)
conv2d_3: Conv2D
input: (None, 58, 58, 64)
output: (None, 56, 56, 64)
conv2d_4: Conv2D
input: (None, 56, 56, 64)
output: (None, 54, 54, 64)
batch_normalization: BatchNormalization
input: (None, 54, 54, 64)
output: (None, 54, 54, 64)
max_pooling2d: MaxPooling2D
input: (None, 54, 54, 64)
output: (None, 27, 27, 64)
global_average_pooling2d: GlobalAveragePooling2D
input: (None, 27, 27, 64)
output: (None, 64)
dense: Dense
input: (None, 64)
output: (None, 288)

I…

```
from keras import optimizers
import keras
optimizer = keras.optimizers.Adam(lr=0.0001)

early_stopping_callback = EarlyStopping(monitor = 'val_loss', patience = 10, mode


model.compile(loss = 'categorical_crossentropy', optimizer = optimizer, metrics =

model_training_history = model.fit(x = features_train, y = labels_train, epochs =
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/optimizer_v2/optimize
r_v2.py:375: UserWarning: The `lr` argument is deprecated, use `learning_rate` inste
ad.
  "The `lr` argument is deprecated, use `learning_rate` instead.")
Epoch 1/40
1920/1920 [==============================] - 34s 10ms/step - loss: 0.5631 - accurac
y: 0.8086 - val_loss: 0.2843 - val_accuracy: 0.9128
Epoch 2/40
1920/1920 [==============================] - 19s 10ms/step - loss: 0.2154 - accurac
y: 0.9322 - val_loss: 0.2088 - val_accuracy: 0.9211
Epoch 3/40
1920/1920 [==============================] - 19s 10ms/step - loss: 0.1217 - accurac
y: 0.9627 - val_loss: 0.0928 - val_accuracy: 0.9737
Epoch 4/40
1920/1920 [==============================] - 19s 10ms/step - loss: 0.0906 - accurac
y: 0.9722 - val_loss: 0.1249 - val_accuracy: 0.9583
Epoch 5/40
1920/1920 [==============================] - 19s 10ms/step - loss: 0.0682 - accurac
y: 0.9801 - val_loss: 0.1521 - val_accuracy: 0.9424
Epoch 6/40
1920/1920 [==============================] - 19s 10ms/step - loss: 0.0531 - accurac
y: 0.9838 - val_loss: 0.0269 - val_accuracy: 0.9924
Epoch 7/40
1920/1920 [==============================] - 19s 10ms/step - loss: 0.0420 - accurac
y: 0.9877 - val_loss: 0.0351 - val_accuracy: 0.9904
Epoch 8/40
1920/1920 [==============================] - 19s 10ms/step - loss: 0.0408 - accurac
y: 0.9876 - val_loss: 0.0758 - val_accuracy: 0.9771
Epoch 9/40
1920/1920 [==============================] - 19s 10ms/step - loss: 0.0311 - accurac
y: 0.9906 - val_loss: 0.1535 - val_accuracy: 0.9516
Epoch 10/40
1920/1920 [==============================] - 19s 10ms/step - loss: 0.0333 - accurac
y: 0.9898 - val_loss: 0.1692 - val_accuracy: 0.9430
Epoch 11/40
1920/1920 [==============================] - 19s 10ms/step - loss: 0.0289 - accurac
y: 0.9909 - val_loss: 0.0062 - val_accuracy: 0.9984
Epoch 12/40
1920/1920 [==============================] - 19s 10ms/step - loss: 0.0232 - accurac
y: 0.9931 - val_loss: 0.0125 - val_accuracy: 0.9956
Epoch 13/40
1920/1920 [==============================] - 19s 10ms/step - loss: 0.0272 - accurac
y: 0.9916 - val_loss: 0.0845 - val_accuracy: 0.9728
Epoch 14/40
1920/1920 [==============================] - 19s 10ms/step - loss: 0.0174 - accurac
y: 0.9950 - val_loss: 0.0140 - val_accuracy: 0.9964
Epoch 15/40
```

In [ ]:

```
model_evaluation_history = model.evaluate(features_test, labels_test)

from sklearn.metrics import classification_report

y_pred = model.predict(features_test, batch_size=4, verbose=1)
y_pred_bool = np.argmax(y_pred, axis=1)

l_test=np.argmax(labels_test, axis=1)

print(classification_report(l_test, y_pred_bool))
```

```
300/300 [==============================] - 2s 5ms/step - loss: 0.0035 - accuracy: 0.
9986
2400/2400 [==============================] - 4s 1ms/step
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1604
           1       1.00      1.00      1.00      1615
           2       1.00      1.00      1.00      1593
           3       1.00      1.00      1.00      1585
           4       1.00      1.00      1.00      1602
           5       1.00      1.00      1.00      1601

    accuracy                           1.00      9600
   macro avg       1.00      1.00      1.00      9600
weighted avg       1.00      1.00      1.00      9600
```

In [ ]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(l_test, y_pred_bool)

print (cm)
```

```
[[1604    0    0    0    0    0]
 [   0 1609    1    2    3    0]
 [   0    1 1589    1    2    0]
 [   0    1    2 1582    0    0]
 [   0    0    0    0 1602    0]
 [   0    0    0    0    0 1601]]
```

I...

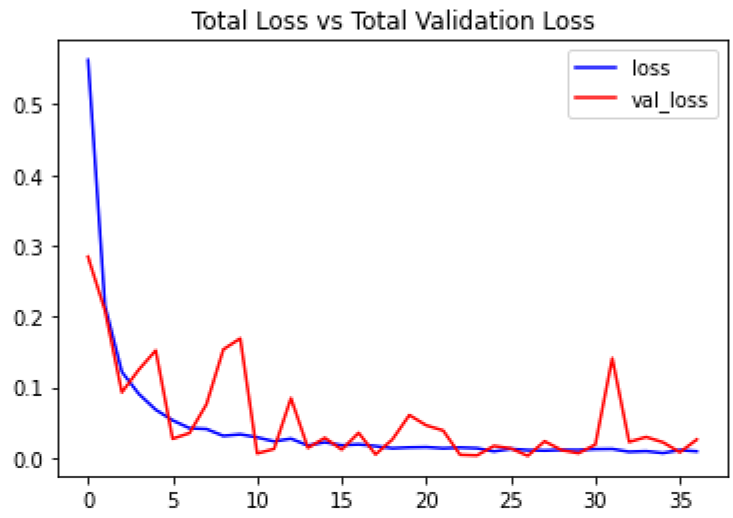
```
# Creating a useful name for our model, incase you're saving multiple models (OPT
date_time_format = '%Y_%m_%d__%H_%M_%S'
current_date_time_dt = dt.datetime.now()
current_date_time_string = dt.datetime.strftime(current_date_time_dt, date_time_f
model_evaluation_loss, model_evaluation_accuracy = model_evaluation_history
model_name = f'Model___Date_Time_{current_date_time_string}___Loss_{model_evaluat

# Saving your Model
model.save(model_name)
```

In [ ]:

```
def plot_metric(metric_name_1, metric_name_2, plot_name):
  # Get Metric values using metric names as identifiers
  metric_value_1 = model_training_history.history[metric_name_1]
  metric_value_2 = model_training_history.history[metric_name_2]

  # Constructing a range object which will be used as time
```

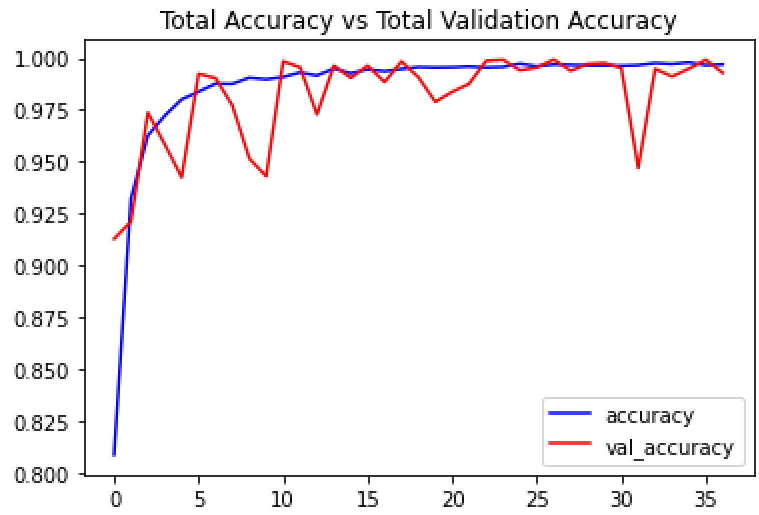
In [ ]:

```
plot_metric('loss', 'val_loss', 'Total Loss vs Total Validation Loss')
```



In ...

```
plot_metric('accuracy', 'val_accuracy', 'Total Accuracy vs Total Validation Accu
```



In [ ]:

```
from collections import Counter

def get_first_mode(a):
    c = Counter(a)
    mode_count = max(c.values())
    mode = {key for key, count in c.items() if count == mode_count}
    first_mode = next(x for x in a if x in mode)
    return first_mode
```

In [ ]:

```
def frames_extraction2(video_path):
    frames_list = []

    vidObj = cv2.VideoCapture(video_path)




    # Used as counter variable
```

In ...

```
#Evaluating a different dataset

from tqdm import tqdm
from statistics import mode


predict = []
actual = []
dataset_directory2="UCF50"

# Declaring Empty Lists to store the features and labels values.
temp_features = []
features = []
labels = []

cc=0

# Iterating through all the classes mentioned in the classes list
for class_index, class_name in enumerate(classes_list):
    print(f'Extracting Data of Class: {class_name}')

    # Getting the list of video files present in the specific class name directc
    files_list = os.listdir(os.path.join(dataset_directory2, class_name))

    # Iterating through all the files present in the files list
    for file_name in files_list:

        # Construct the complete video path
        video_file_path = os.path.join(dataset_directory2, class_name, file_name

        # Calling the frame_extraction method for every video file path
        frames = frames_extraction2(video_file_path)

        temppred=[]

        for i in frames:
          temppred.append(model.predict_classes(np.expand_dims(i, axis = 0))[0])

        print (temppred)
        print ("mode", get_first_mode(temppred), cc)
        cc+=1
        predict.append(get_first_mode(temppred))
        actual.append(class_index)
```

```
Extracting Data of Class: pullup
Defected frame
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/sequential.py:
455: UserWarning: `model.predict_classes()` is deprecated and will be removed after
2021-01-01. Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your m
odel does multi-class classification   (e.g. if it uses a `softmax` last-layer activ
ation).* `(model.predict(x) > 0.5).astype("int32")`,   if your model does binary cla
ssification   (e.g. if it uses a `sigmoid` last-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
[1, 1, 1, 1, 1, 1, 1]
mode 1 0
Defected frame
```

In [ ]:

```
print(classification_report(actual, predict))
```

```
              precision    recall  f1-score   support

           0       0.89      0.39      0.54       120
           1       0.44      0.34      0.39       160
           2       0.51      0.75      0.60       153
           3       0.30      0.22      0.25       111
           4       0.43      0.71      0.54       145
           5       0.87      0.68      0.77       142

    accuracy                           0.53       831
   macro avg       0.57      0.52      0.52       831
weighted avg       0.57      0.53      0.52       831
```

In [ ]:

```
print(confusion_matrix(actual, predict))
```

```
[[ 47  32   9   8  15   9]
 [  4  55  38  13  50   0]
 [  2   6 114  13  16   2]
 [  0  21  25  24  41   0]
 [  0   5  16  18 103   3]
 [  0   5  22   4  14  97]]
```