Project part_1 Dataset preparation

Data set preview

```
# import the realted library

import os
import cv2
#import pafy
import math
import random
import numpy as np
import datetime as dt
import tensorflow as tf
from collections import deque
import matplotlib.pyplot as plt

from moviepy.editor import *
%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


from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# import the data file
total_files_names = os.listdir('drive/My Drive/UCF50')


total_files_names[:10]  # show the first 10 files names
```

```
['Skijet',
 'Swing',
 'SoccerJuggling',
 'TrampolineJumping',
 'WalkingWithDog',
 'TaiChi',
 'ThrowDiscus',
 'TennisSwing',
 'YoYo',
 'VolleyballSpiking']
```

```
selected_activity = total_files_names[10]  # select particular activity file
selected_activity  # show file name
```

```
'RockClimbingIndoor'
```

```
selected_video_list = os.listdir(f'drive/My Drive/UCF50/{selected_activity}')
selected_video_list[:5]  # show the first 5 videos
```

```
['v_RockClimbingIndoor_g01_c01.avi',
 'v_RockClimbingIndoor_g01_c02.avi',
 'v_RockClimbingIndoor_g01_c03.avi',
 'v_RockClimbingIndoor_g01_c04.avi',
 'v_RockClimbingIndoor_g01_c05.avi']
```

```
selected_video = selected_video_list[1] # pick one video in the particular file


video_reader = cv2.VideoCapture(f'drive/My Drive/UCF50/{selected_activity}/{selected_video}')  # use cvs to read the video file


success, begin_frame = video_reader.read()  # capture video first frame
video_reader.isOpened()  # test if the video is opened successfully
```

```
video_reader.isOpened()  # test if the video is opened successfully
```

```
True
```

```
frame_rgb = cv2.cvtColor(begin_frame, cv2.COLOR_BGR2RGB)  # change the image format to RGB
```

```
frame_rgb.shape  # show the image shape in the selected video
```

```
(240, 320, 3)
```

```
cv2.putText(frame_rgb, selected_activity, (20,50), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (66, 135, 245), 2)
plt.imshow(frame_rgb)
```

```
<matplotlib.image.AxesImage at 0x7b92eba9ece0>
```

Data set preprocess

We need to extract several images from the video. Here, I picked 20 sequences in each video and each image has the internal for total frames/20.

```
# select 7 classes we will use in the model
data_classes = ['JumpRope', 'Kayaking', 'Lunges', 'Diving', 'PlayingGuitar', 'PlayingPiano', 'PlayingViolin']
```

```
sequence_length = 20 # choose 20 images for each video
```

```
raw_data = 'drive/My Drive/UCF50'
```

```
def frames_extraction(video_path):
    '''
    This function will extract the required frames from a video after resizing and normalizing them.
    Args:
        video_path: The path of the video in the disk, whose frames are to be extracted.
    Returns:
        frames_list: A list containing the resized and normalized frames of the video.
    '''

    # Declare a list to store video frames.
    frames_list = []

    # Read the Video File using the VideoCapture object.
    video_reader = cv2.VideoCapture(video_path)

    # Get the total number of frames in the video.
    video_frames_count = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))

    # Calculate the the interval after which frames will be added to the list.
    skip_frames_window = max(int(video_frames_count/sequence_length), 1)

    # Iterate through the Video Frames.
    for frame_counter in range(sequence_length):

        # Set the current frame position of the video.
        video_reader.set(cv2.CAP_PROP_POS_FRAMES, frame_counter * skip_frames_window)

        # Reading the frame from the video.
        success, frame = video_reader.read()

        # Check if Video frame is not successfully read then break the loop
        if not success:
            break

        # Resize the Frame
        resized_frame = cv2.resize(frame, (100, 100))  #image height and width both for 100

        # Normalize the frame
        normalized_frame = resized_frame / 255

        # Append the normalized frame into the frames list
        frames_list.append(normalized_frame)

    # Release the VideoCapture object.
    video_reader.release()

    # Return the frames list.
    return frames_list
```

```
def create_dataset():
    '''
    This function will extract the data of the selected classes and create the required dataset.
    Returns:
        features:          A list containing the extracted frames of the videos.
        labels:            A list containing the indexes of the classes associated with the videos.
        video_files_paths: A list containing the paths of the videos in the disk.
    '''

    # Declared Empty Lists to store the features, labels and video file path values.
    features = []
    labels = []
    video_files_paths = []

    # Iterating through all the classes mentioned in the classes list
    for class_index, class_name in enumerate(data_classes):

        # Display the name of the class whose data is being extracted.
        print(f'Dataset label: {class_name}')

        # Get the list of video files present in the specific class name directory
```

```
# Create the dataset
features, labels, video_files_paths = create_dataset()
```

```
 Dataset label: JumpRope
```

```
one_hot_encoded_labels = to_categorical(labels)
```

```
#split the train and test data
features_train, features_test, labels_train, labels_test = train_test_split(features, one_hot_encoded_labels, test_size = 0.2, shuffle = Tru

            # Check if the extracted frames belong to the required number of the specified class
```

```
len(features), len(one_hot_encoded_labels), len(features_train), len(features_test)  # check the number of dataset
```

```
 (964, 964, 771, 193)
```

```
            # Append the data to their repective lists.
```

```
features.shape, labels.shape
```

```
 ((964, 20, 100, 100, 3), (964,))
```

```
    # Converting the list to numpy arrays
```

```
np.savez('drive/My Drive/projet_human_activity.npz', features, labels)
```

```
    labels = np.array(labels)
```

LRCN

November 26, 2023

[1]:
```
# import the realted library

import os
import cv2
#import pafy
import math
import random
import numpy as np
import datetime as dt
import tensorflow as tf
from collections import deque
import matplotlib.pyplot as plt

from moviepy.editor import *
%matplotlib inline

from sklearn.model_selection import train_test_split

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import *
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.utils import plot_model
```

[2]:
```
from tensorflow.keras.applications import InceptionResNetV2
```

[3]:
```
dataset = np.load('project_human_activity.npz')
```

[4]:
```
features = dataset['arr_0']
labels = dataset['arr_1']
```

[5]:
```
np.unique(labels)
```

[5]:
```
array([0, 1, 2, 3, 4, 5, 6])
```

[6]:
```
one_hot_encoded_labels = to_categorical(labels)
```

[7]:
```
#split the train and test data
features_train, features_test, labels_train, labels_test =␣
↪train_test_split(features, one_hot_encoded_labels, test_size = 0.2, shuffle␣
↪= True, random_state = 123)
```

[8]:
```
sequence_length = 20
```

[9]:
```
# select 7 classes we will use in the model
data_classes = ['JumpRope', 'Kayaking', 'Lunges', 'Diving', 'PlayingGuitar',␣
↪'PlayingPiano', 'PlayingViolin']
```

[10]:
```
inceptionresnet = InceptionResNetV2(
    include_top=False,
    weights="imagenet",
    input_shape=(100,100,3)
)
```

[11]:
```
features_train.shape
```

[11]:
```
(771, 20, 100, 100, 3)
```

[12]:
```
# only train the last layer
for layer in inceptionresnet.layers[:-4]:
    layer.trainable = False
```

[13]:
```
model = Sequential()

model.add(TimeDistributed(inceptionresnet, input_shape=(20, 100, 100, 3)))

model.add(TimeDistributed(Flatten()))

model.add(LSTM(32))
model.add(Dropout(0.2))
model.add(Dense(32, activation='relu'))
model.add(Dense(len(data_classes), activation='softmax'))
```

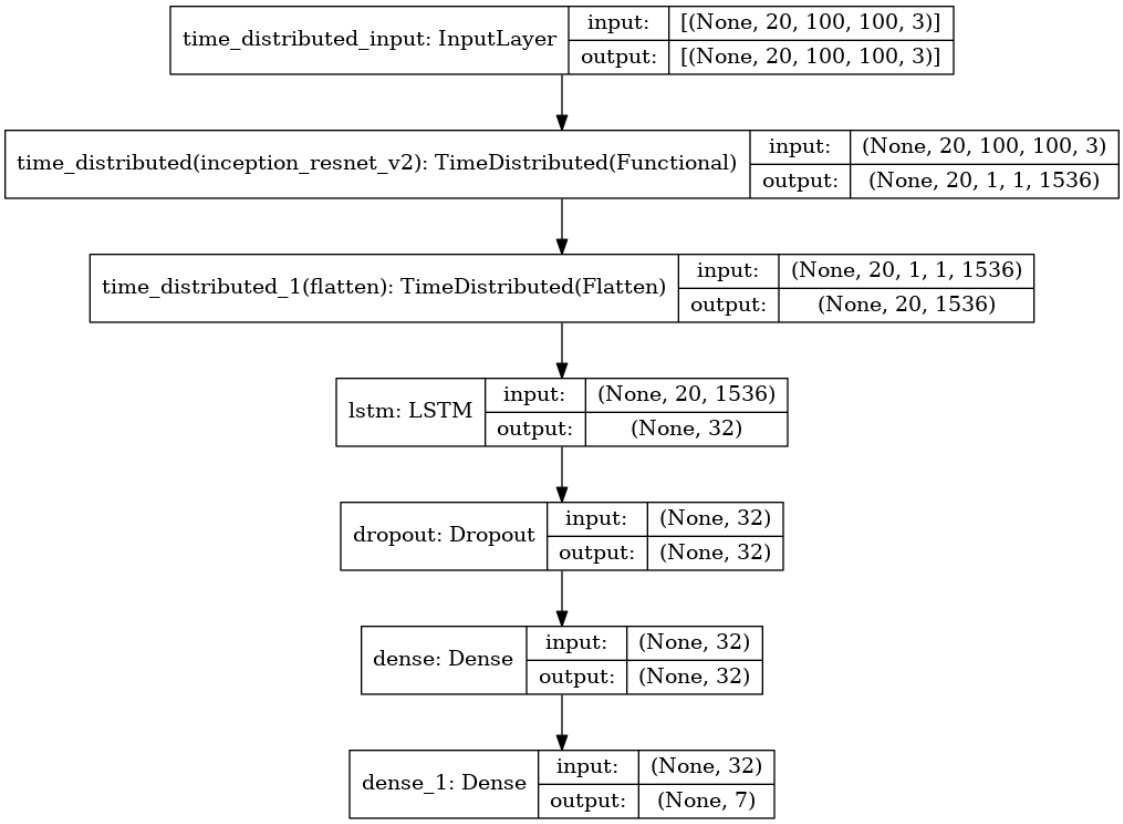
[14]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
time_distributed (TimeDistri (None, 20, 1, 1, 1536)    54336736
_________________________________________________________________
time_distributed_1 (TimeDist (None, 20, 1536)          0
```

```
_________________________________________________________________
lstm (LSTM)                  (None, 32)                200832
_________________________________________________________________
dropout (Dropout)            (None, 32)                0
_________________________________________________________________
dense (Dense)                (None, 32)                1056
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 231
=================================================================
Total params: 54,538,855
Trainable params: 3,398,535
Non-trainable params: 51,140,320
_________________________________________________________________
```

[15]:
```
plot_model(model, show_shapes=True)
```

[15]:



[16]:
```
# Create callback.
callbacks = EarlyStopping(monitor = 'val_loss', patience = 10, mode = 'min',␣
 ↪restore_best_weights = True)

# Compile the model
```

```
model.compile(loss = 'categorical_crossentropy', optimizer = 'rmsprop', metrics␣
↪= ["accuracy"])

# Start training the model.
history = model.fit(x = features_train, y = labels_train, epochs = 60,␣
↪batch_size = 6,
                                                shuffle = True, validation_split =␣
↪0.2, callbacks = callbacks)
```

```
Epoch 1/60
103/103 [==============================] - 33s 183ms/step - loss: 1.2521 -
accuracy: 0.5877 - val_loss: 0.7359 - val_accuracy: 0.7677
Epoch 2/60
103/103 [==============================] - 14s 134ms/step - loss: 0.6619 -
accuracy: 0.8101 - val_loss: 0.4015 - val_accuracy: 0.8903
Epoch 3/60
103/103 [==============================] - 14s 132ms/step - loss: 0.4880 -
accuracy: 0.8620 - val_loss: 0.2108 - val_accuracy: 0.9613
Epoch 4/60
103/103 [==============================] - 14s 133ms/step - loss: 0.3039 -
accuracy: 0.9107 - val_loss: 0.1300 - val_accuracy: 0.9677
Epoch 5/60
103/103 [==============================] - 13s 131ms/step - loss: 0.2329 -
accuracy: 0.9351 - val_loss: 0.1111 - val_accuracy: 0.9677
Epoch 6/60
103/103 [==============================] - 14s 135ms/step - loss: 0.1824 -
accuracy: 0.9545 - val_loss: 0.0766 - val_accuracy: 0.9806
Epoch 7/60
103/103 [==============================] - 14s 132ms/step - loss: 0.1612 -
accuracy: 0.9659 - val_loss: 0.1116 - val_accuracy: 0.9742
Epoch 8/60
103/103 [==============================] - 14s 133ms/step - loss: 0.1765 -
accuracy: 0.9432 - val_loss: 0.0473 - val_accuracy: 0.9935
Epoch 9/60
103/103 [==============================] - 14s 135ms/step - loss: 0.1190 -
accuracy: 0.9627 - val_loss: 0.0635 - val_accuracy: 0.9871
Epoch 10/60
103/103 [==============================] - 13s 131ms/step - loss: 0.1074 -
accuracy: 0.9675 - val_loss: 0.0543 - val_accuracy: 0.9871
Epoch 11/60
103/103 [==============================] - 14s 132ms/step - loss: 0.0765 -
accuracy: 0.9740 - val_loss: 0.0255 - val_accuracy: 0.9935
Epoch 12/60
103/103 [==============================] - 13s 131ms/step - loss: 0.0472 -
accuracy: 0.9919 - val_loss: 0.0597 - val_accuracy: 0.9806
Epoch 13/60
103/103 [==============================] - 14s 132ms/step - loss: 0.0476 -
```

```
accuracy: 0.9838 - val_loss: 0.0625 - val_accuracy: 0.9806
Epoch 14/60
103/103 [==============================] - 14s 133ms/step - loss: 0.0708 -
accuracy: 0.9789 - val_loss: 0.0504 - val_accuracy: 0.9806
Epoch 15/60
103/103 [==============================] - 14s 133ms/step - loss: 0.0617 -
accuracy: 0.9821 - val_loss: 0.0554 - val_accuracy: 0.9935
Epoch 16/60
103/103 [==============================] - 14s 132ms/step - loss: 0.0530 -
accuracy: 0.9838 - val_loss: 0.0425 - val_accuracy: 0.9806
Epoch 17/60
103/103 [==============================] - 14s 132ms/step - loss: 0.0602 -
accuracy: 0.9805 - val_loss: 0.0949 - val_accuracy: 0.9806
Epoch 18/60
103/103 [==============================] - 14s 131ms/step - loss: 0.0449 -
accuracy: 0.9903 - val_loss: 0.0748 - val_accuracy: 0.9871
Epoch 19/60
103/103 [==============================] - 13s 129ms/step - loss: 0.0428 -
accuracy: 0.9838 - val_loss: 0.0844 - val_accuracy: 0.9871
Epoch 20/60
103/103 [==============================] - 14s 135ms/step - loss: 0.0766 -
accuracy: 0.9756 - val_loss: 0.0691 - val_accuracy: 0.9871
Epoch 21/60
103/103 [==============================] - 14s 134ms/step - loss: 0.0190 -
accuracy: 0.9935 - val_loss: 0.0699 - val_accuracy: 0.9871
```

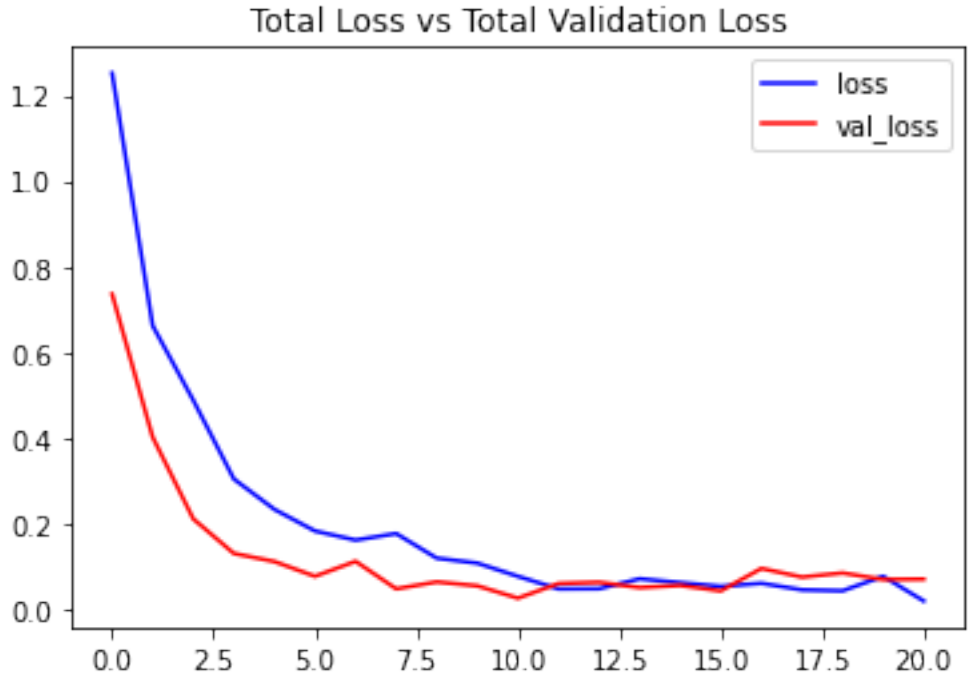
[17]:
```
# Evaluate the trained model.
model_prediction = model.evaluate(features_test, labels_test)
```

```
7/7 [==============================] - 3s 194ms/step - loss: 0.1043 - accuracy:
0.9689
```

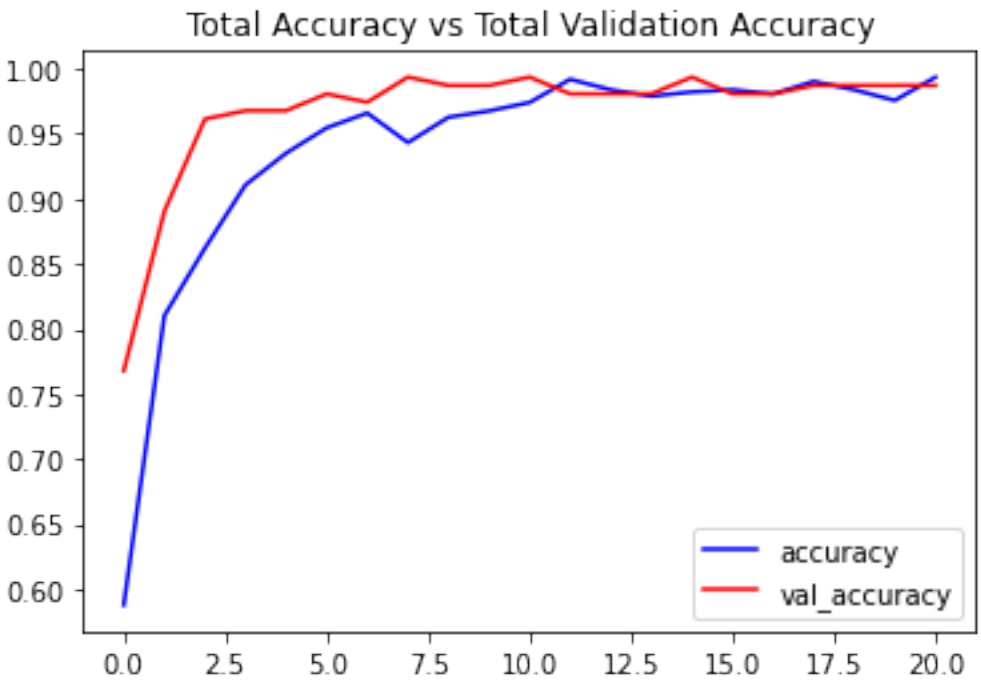
[18]:
```
def plot_metric(history, metric_name_1, metric_name_2, plot_name):
    '''
    This function will plot the metrics passed to it in a graph.
    Args:
        model_training_history: A history object containing a record of␣
↪training and validation
                                loss values and metrics values at successive␣
↪epochs
        metric_name_1:          The name of the first metric that needs to be␣
↪plotted in the graph.
        metric_name_2:          The name of the second metric that needs to be␣
↪plotted in the graph.
        plot_name:              The title of the graph.
    '''
```

```
    # Get metric values using metric names as identifiers.
    metric_value_1 = history.history[metric_name_1]
    metric_value_2 = history.history[metric_name_2]

    # Construct a range object which will be used as x-axis (horizontal plane)␣
↪of the graph.
    epochs = range(len(metric_value_1))

    # Plot the Graph.
    plt.plot(epochs, metric_value_1, 'blue', label = metric_name_1)
    plt.plot(epochs, metric_value_2, 'red', label = metric_name_2)

    # Add title to the plot.
    plt.title(str(plot_name))

    # Add legend to the plot.
    plt.legend()
```

[19]:
```
# Visualize the training and validation loss metrices.
plot_metric(history, 'loss', 'val_loss', 'Total Loss vs Total Validation Loss')
```



[20]:
```
# Visualize the training and validation accuracy metrices.
plot_metric(history, 'accuracy', 'val_accuracy', 'Total Accuracy vs Total␣
↪Validation Accuracy')
```


Total Accuracy vs Total Validation Accuracy
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
accuracy
val_accuracy

3D_cnn

November 26, 2023

[1]:
```
# import the realted library

import os
import cv2
#import pafy
import math
import random
import numpy as np
import datetime as dt
import tensorflow as tf
from collections import deque
import matplotlib.pyplot as plt

from moviepy.editor import *
%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.applications.vgg16 import VGG16
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.utils import plot_model
```

[2]:
```
from tensorflow.keras.initializers import Constant
```

[3]:
```
from tensorflow.keras.layers import BatchNormalization
```

[4]:
```
dataset = np.load('project_human_activity.npz')
```

[5]:
```
features = dataset['arr_0']
labels = dataset['arr_1']
```

[6]:
```
np.unique(labels)
```

[6]:
```
array([0, 1, 2, 3, 4, 5, 6])
```

[7]:
```
one_hot_encoded_labels = to_categorical(labels)
```

[8]:
```
#split the train and test data
features_train, features_test, labels_train, labels_test =␣
↪train_test_split(features, one_hot_encoded_labels, test_size = 0.2, shuffle␣
↪= True, random_state = 123)
```

[9]:
```
len(features_train)
```

[9]:
```
771
```

[10]:
```
features_train[0][0].shape  # check the image shape
```

[10]:
```
(100, 100, 3)
```

0.1 Build the model

[11]:
```
sequence_length = 20
```

[12]:
```
# select 7 classes we will use in the model
data_classes = ['JumpRope', 'Kayaking', 'Lunges', 'Diving', 'PlayingGuitar',␣
↪'PlayingPiano', 'PlayingViolin']
```

[13]:
```
model = Sequential()
model.add(Conv3D(8,(3,3,3), activation='relu', input_shape=(20,100,100,3)))
model.add(Conv3D(16,(3,3,3), activation='relu'))
model.add(MaxPooling3D((2,2,2)))

model.add(Conv3D(32,(3,3,3), activation='relu'))
model.add(Conv3D(64,(2,2,2), activation='relu'))
model.add(MaxPooling3D((2,2,2)))
model.add(Dropout(0.3))
model.add(Flatten())

model.add(Dense(128,'relu'))
model.add(Dropout(0.3))
model.add(Dense(64,'relu'))
model.add(Dropout(0.3))
model.add(Dense(len(data_classes),'softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv3d (Conv3D)              (None, 18, 98, 98, 8)     656
_________________________________________________________________
```

```
conv3d_1 (Conv3D)            (None, 16, 96, 96, 16)    3472
_________________________________________________________________
max_pooling3d (MaxPooling3D) (None, 8, 48, 48, 16)     0
_________________________________________________________________
conv3d_2 (Conv3D)            (None, 6, 46, 46, 32)     13856
_________________________________________________________________
conv3d_3 (Conv3D)            (None, 5, 45, 45, 64)     16448
_________________________________________________________________
max_pooling3d_1 (MaxPooling3 (None, 2, 22, 22, 64)     0
_________________________________________________________________
dropout (Dropout)            (None, 2, 22, 22, 64)     0
_________________________________________________________________
flatten (Flatten)            (None, 61952)             0
_________________________________________________________________
dense (Dense)                (None, 128)               7929984
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 455
=================================================================
Total params: 7,973,127
Trainable params: 7,973,127
Non-trainable params: 0
_________________________________________________________________
```

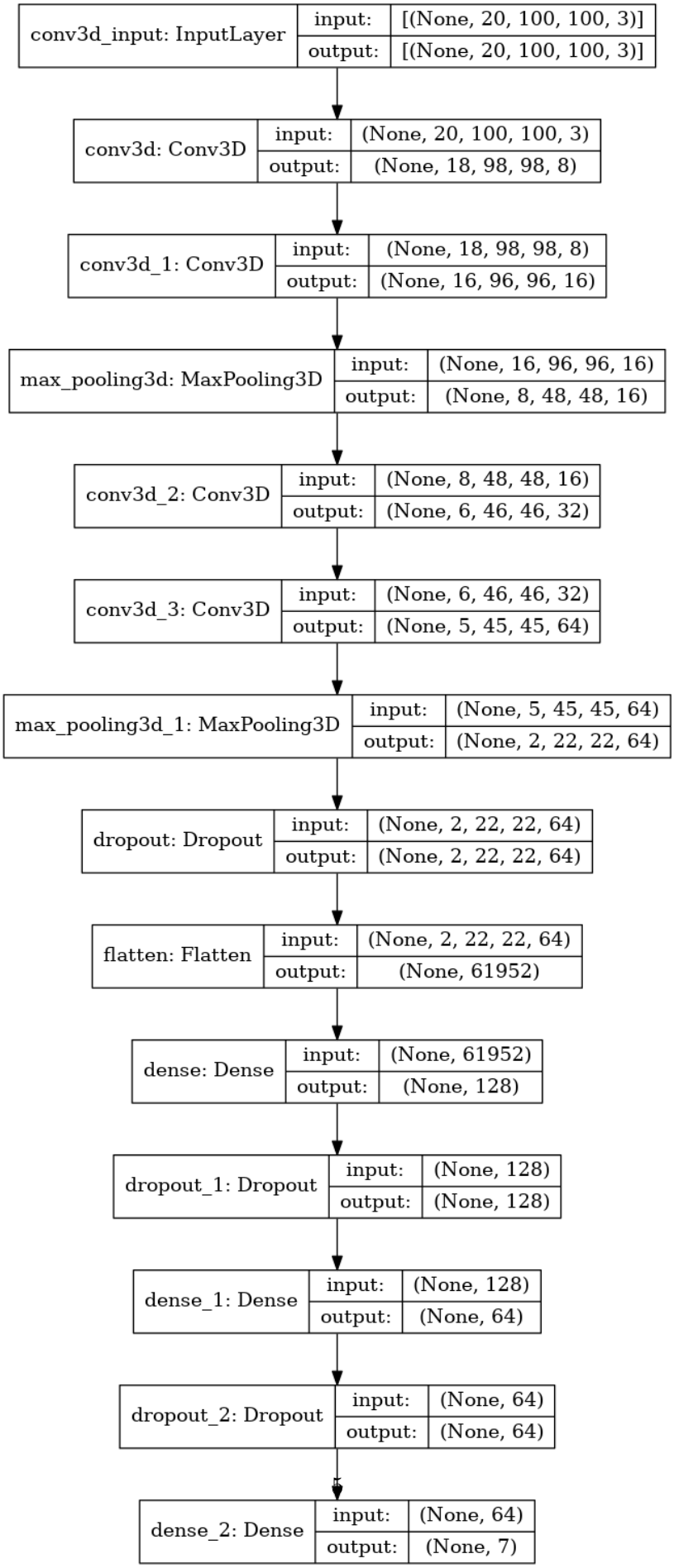
[14]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv3d (Conv3D)              (None, 18, 98, 98, 8)     656
_________________________________________________________________
conv3d_1 (Conv3D)            (None, 16, 96, 96, 16)    3472
_________________________________________________________________
max_pooling3d (MaxPooling3D) (None, 8, 48, 48, 16)     0
_________________________________________________________________
conv3d_2 (Conv3D)            (None, 6, 46, 46, 32)     13856
_________________________________________________________________
conv3d_3 (Conv3D)            (None, 5, 45, 45, 64)     16448
_________________________________________________________________
max_pooling3d_1 (MaxPooling3 (None, 2, 22, 22, 64)     0
_________________________________________________________________
dropout (Dropout)            (None, 2, 22, 22, 64)     0
```

```
_________________________________________________________________
flatten (Flatten)            (None, 61952)             0
_________________________________________________________________
dense (Dense)                (None, 128)               7929984
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 455
=================================================================
Total params: 7,973,127
Trainable params: 7,973,127
Non-trainable params: 0
_________________________________________________________________
```

[15]:
```
plot_model(model, show_shapes=True)
```

[15]:


conv3d_input: InputLayer
input: [(None, 20, 100, 100, 3)]
output: [(None, 20, 100, 100, 3)]
conv3d: Conv3D
input: (None, 20, 100, 100, 3)
output: (None, 18, 98, 98, 8)
conv3d_1: Conv3D
input: (None, 18, 98, 98, 8)
output: (None, 16, 96, 96, 16)
max_pooling3d: MaxPooling3D
input: (None, 16, 96, 96, 16)
output: (None, 8, 48, 48, 16)
conv3d_2: Conv3D
input: (None, 8, 48, 48, 16)
output: (None, 6, 46, 46, 32)
conv3d_3: Conv3D
input: (None, 6, 46, 46, 32)
output: (None, 5, 45, 45, 64)
max_pooling3d_1: MaxPooling3D
input: (None, 5, 45, 45, 64)
output: (None, 2, 22, 22, 64)
dropout: Dropout
input: (None, 2, 22, 22, 64)
output: (None, 2, 22, 22, 64)
flatten: Flatten
input: (None, 2, 22, 22, 64)
output: (None, 61952)
dense: Dense
input: (None, 61952)
output: (None, 128)
dropout_1: Dropout
input: (None, 128)
output: (None, 128)
dense_1: Dense
input: (None, 128)
output: (None, 64)
dropout_2: Dropout
input: (None, 64)
output: (None, 64)
dense_2: Dense
input: (None, 64)
output: (None, 7)

0.1.1 Train the model

[16]:
```
# Create callback.
callbacks = EarlyStopping(monitor = 'val_loss', patience = 5, mode = 'min',␣
↪restore_best_weights = True)

# Compile the model
model.compile(loss = 'categorical_crossentropy', optimizer = 'rmsprop', metrics␣
↪= ["accuracy"])

# Start training the model.
history = model.fit(x = features_train, y = labels_train, epochs = 60,␣
↪batch_size = 6,
                                                shuffle = True, validation_split =␣
↪0.2, callbacks = callbacks)
```

```
Epoch 1/60
103/103 [==============================] - 7s 42ms/step - loss: 1.9160 -
accuracy: 0.2208 - val_loss: 1.6584 - val_accuracy: 0.3355
Epoch 2/60
103/103 [==============================] - 3s 31ms/step - loss: 1.5203 -
accuracy: 0.4399 - val_loss: 0.9229 - val_accuracy: 0.6129
Epoch 3/60
103/103 [==============================] - 3s 31ms/step - loss: 1.1899 -
accuracy: 0.5942 - val_loss: 0.6479 - val_accuracy: 0.7742
Epoch 4/60
103/103 [==============================] - 3s 31ms/step - loss: 0.8639 -
accuracy: 0.6964 - val_loss: 0.5134 - val_accuracy: 0.8258
Epoch 5/60
103/103 [==============================] - 3s 31ms/step - loss: 0.6479 -
accuracy: 0.7727 - val_loss: 0.6527 - val_accuracy: 0.7613
Epoch 6/60
103/103 [==============================] - 3s 31ms/step - loss: 0.5386 -
accuracy: 0.8263 - val_loss: 1.7398 - val_accuracy: 0.6710
Epoch 7/60
103/103 [==============================] - 3s 31ms/step - loss: 0.5221 -
accuracy: 0.8231 - val_loss: 0.4450 - val_accuracy: 0.8387
Epoch 8/60
103/103 [==============================] - 3s 31ms/step - loss: 0.4886 -
accuracy: 0.8588 - val_loss: 0.4952 - val_accuracy: 0.8323
Epoch 9/60
103/103 [==============================] - 3s 31ms/step - loss: 0.4278 -
accuracy: 0.8685 - val_loss: 0.2592 - val_accuracy: 0.9290
Epoch 10/60
103/103 [==============================] - 3s 31ms/step - loss: 0.3734 -
```

```
accuracy: 0.8864 - val_loss: 0.3695 - val_accuracy: 0.8774
Epoch 11/60
103/103 [==============================] - 3s 31ms/step - loss: 0.2969 -
accuracy: 0.9107 - val_loss: 1.4638 - val_accuracy: 0.7613
Epoch 12/60
103/103 [==============================] - 3s 31ms/step - loss: 0.2749 -
accuracy: 0.9286 - val_loss: 0.2843 - val_accuracy: 0.9032
Epoch 13/60
103/103 [==============================] - 3s 31ms/step - loss: 0.2863 -
accuracy: 0.9286 - val_loss: 0.3582 - val_accuracy: 0.9161
Epoch 14/60
103/103 [==============================] - 3s 31ms/step - loss: 0.2180 -
accuracy: 0.9269 - val_loss: 0.8652 - val_accuracy: 0.8516
```

0.2 Evaluate on test set

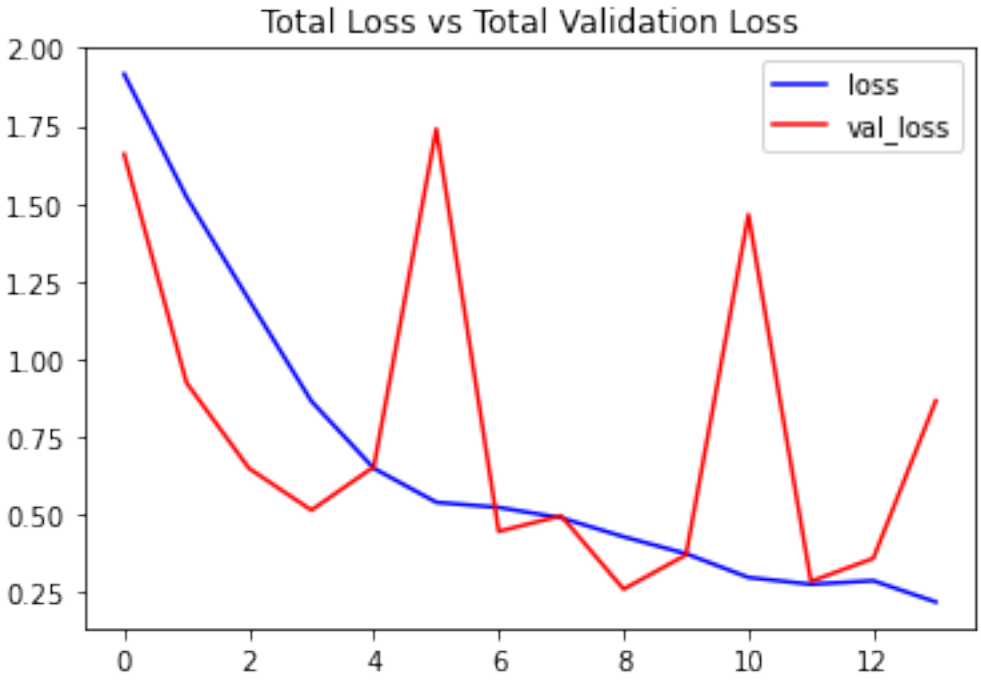
[17]:
```
# Evaluate the trained model.
model_prediction = model.evaluate(features_test, labels_test)
```

```
7/7 [==============================] - 0s 45ms/step - loss: 0.4929 - accuracy:
0.8601
```

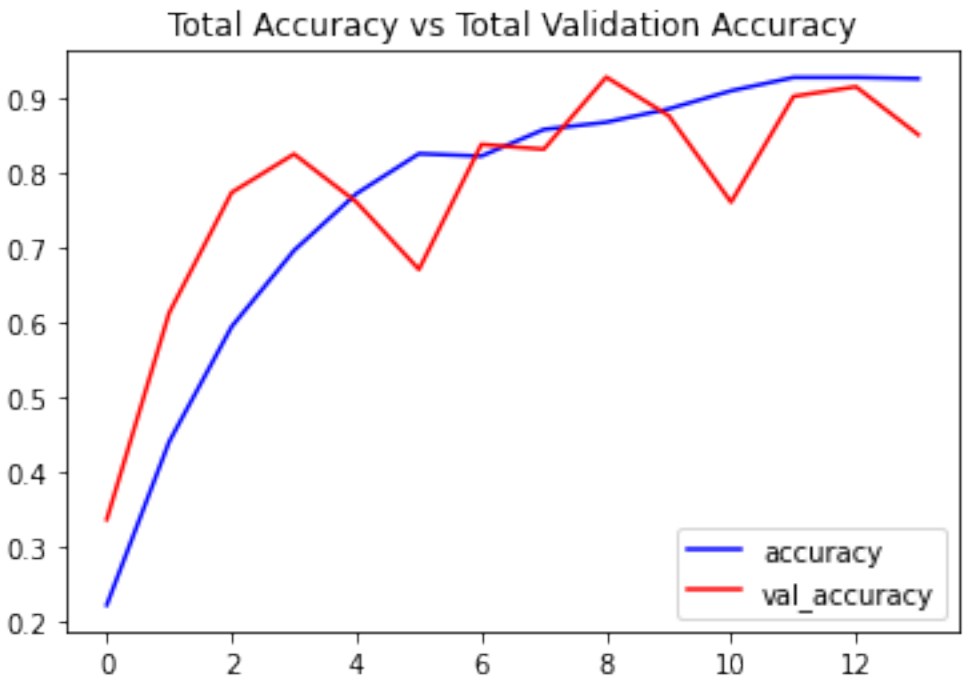
[18]:
```
def plot_metric(history, metric_name_1, metric_name_2, plot_name):
    '''
    This function will plot the metrics passed to it in a graph.
    Args:
        model_training_history: A history object containing a record of␣
↪training and validation
                                loss values and metrics values at successive␣
↪epochs
        metric_name_1:          The name of the first metric that needs to be␣
↪plotted in the graph.
        metric_name_2:          The name of the second metric that needs to be␣
↪plotted in the graph.
        plot_name:              The title of the graph.
    '''

    # Get metric values using metric names as identifiers.
    metric_value_1 = history.history[metric_name_1]
    metric_value_2 = history.history[metric_name_2]

    # Construct a range object which will be used as x-axis (horizontal plane)␣
↪of the graph.
    epochs = range(len(metric_value_1))

    # Plot the Graph.
    plt.plot(epochs, metric_value_1, 'blue', label = metric_name_1)
```

```
    plt.plot(epochs, metric_value_2, 'red', label = metric_name_2)

    # Add title to the plot.
    plt.title(str(plot_name))

    # Add legend to the plot.
    plt.legend()
```

[19]:
```
# Visualize the training and validation loss metrices.
plot_metric(history, 'loss', 'val_loss', 'Total Loss vs Total Validation Loss')
```



[20]:
```
# Visualize the training and validation accuracy metrices.
plot_metric(history, 'accuracy', 'val_accuracy', 'Total Accuracy vs Total␣
↪Validation Accuracy')
```


Total Accuracy vs Total Validation Accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
2
4
6
8
10
12
accuracy
val_accuracy

convlstm

November 26, 2023

[1]:
```
# import the realted library

import os
import cv2
#import pafy
import math
import random
import numpy as np
import datetime as dt
import tensorflow as tf
from collections import deque
import matplotlib.pyplot as plt

from moviepy.editor import *
%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.applications.vgg16 import VGG16
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.utils import plot_model
from tensorflow.keras.layers import BatchNormalization
```

[2]:
```
dataset = np.load('project_human_activity.npz')
```

[3]:
```
features = dataset['arr_0']
labels = dataset['arr_1']
```

[4]:
```
np.unique(labels)
```

[4]:
```
array([0, 1, 2, 3, 4, 5, 6])
```

[5]:
```
one_hot_encoded_labels = to_categorical(labels)
```

[6]:
```
#split the train and test data
features_train, features_test, labels_train, labels_test =␣
↪train_test_split(features, one_hot_encoded_labels, test_size = 0.2, shuffle␣
↪= True, random_state = 123)
```

[7]:
```
len(features_train)
```

[7]: 771

[8]:
```
features_train[0][0].shape  # check the image shape
```

[8]: (100, 100, 3)

0.1 Build the model

[9]:
```
sequence_length = 20
```

[10]:
```
# select 7 classes we will use in the model
data_classes = ['JumpRope', 'Kayaking', 'Lunges', 'Diving', 'PlayingGuitar',␣
↪'PlayingPiano', 'PlayingViolin']
```

[11]:
```
model = Sequential()

model.add(ConvLSTM2D(filters = 8, padding = "same",  kernel_size = (3, 3),
                 return_sequences = True, data_format = "channels_last",␣
↪input_shape = (sequence_length,100,100,3)))
model.add(MaxPooling3D(pool_size=(1, 2, 2), padding='same',␣
↪data_format='channels_last'))
model.add(ConvLSTM2D(filters = 10, padding = "same",  kernel_size = (3, 3),
                 return_sequences = True, data_format = "channels_last"))
model.add(MaxPooling3D(pool_size=(1, 2, 2), padding='same',␣
↪data_format='channels_last'))
model.add(ConvLSTM2D(filters = 16, padding = "same",  kernel_size = (3, 3),
                 return_sequences = True, data_format = "channels_last"))
model.add(MaxPooling3D(pool_size=(1, 2, 2), padding='same',␣
↪data_format='channels_last'))
model.add(Flatten())
model.add(Dense(32, activation="relu"))
model.add(Dropout(0.3))
model.add(Dense(7, activation = "softmax"))
```

[12]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
```

```
conv_lst_m2d (ConvLSTM2D)    (None, 20, 100, 100, 8)   3200
_________________________________________________________________
max_pooling3d (MaxPooling3D) (None, 20, 50, 50, 8)     0
_________________________________________________________________
conv_lst_m2d_1 (ConvLSTM2D)  (None, 20, 50, 50, 10)    6520
_________________________________________________________________
max_pooling3d_1 (MaxPooling3 (None, 20, 25, 25, 10)    0
_________________________________________________________________
conv_lst_m2d_2 (ConvLSTM2D)  (None, 20, 25, 25, 16)    15040
_________________________________________________________________
max_pooling3d_2 (MaxPooling3 (None, 20, 13, 13, 16)    0
_________________________________________________________________
flatten (Flatten)            (None, 54080)             0
_________________________________________________________________
dense (Dense)                (None, 32)                1730592
_________________________________________________________________
dropout (Dropout)            (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 231
=================================================================
Total params: 1,755,583
Trainable params: 1,755,583
Non-trainable params: 0
_________________________________________________________________
```

0.1.1 Train the model

[13]:
```
# Create callback.
callbacks = EarlyStopping(monitor = 'val_loss', patience = 10, mode = 'min',␣
 ↪restore_best_weights = True)

# Compile the model
model.compile(loss = 'categorical_crossentropy', optimizer = 'rmsprop', metrics␣
 ↪= ["accuracy"])

# Start training the model.
history = model.fit(x = features_train, y = labels_train, epochs = 60,␣
 ↪batch_size = 6,
                                                shuffle = True, validation_split =␣
 ↪0.2, callbacks = callbacks)
```

```
Epoch 1/60
103/103 [==============================] - 39s 315ms/step - loss: 1.6217 -
accuracy: 0.3977 - val_loss: 1.1122 - val_accuracy: 0.6258
Epoch 2/60
103/103 [==============================] - 31s 303ms/step - loss: 0.9157 -
accuracy: 0.6786 - val_loss: 0.7271 - val_accuracy: 0.7548
Epoch 3/60
```

```
103/103 [==============================] - 31s 300ms/step - loss: 0.5785 -
accuracy: 0.7922 - val_loss: 0.4874 - val_accuracy: 0.8194
Epoch 4/60
103/103 [==============================] - 31s 300ms/step - loss: 0.3623 -
accuracy: 0.8815 - val_loss: 0.9518 - val_accuracy: 0.7613
Epoch 5/60
103/103 [==============================] - 31s 303ms/step - loss: 0.2660 -
accuracy: 0.8977 - val_loss: 0.6108 - val_accuracy: 0.8581
Epoch 6/60
103/103 [==============================] - 31s 304ms/step - loss: 0.2127 -
accuracy: 0.9286 - val_loss: 0.5443 - val_accuracy: 0.8323
Epoch 7/60
103/103 [==============================] - 31s 301ms/step - loss: 0.1336 -
accuracy: 0.9578 - val_loss: 0.5815 - val_accuracy: 0.8516
Epoch 8/60
103/103 [==============================] - 31s 301ms/step - loss: 0.1340 -
accuracy: 0.9643 - val_loss: 0.6679 - val_accuracy: 0.8839
Epoch 9/60
103/103 [==============================] - 31s 303ms/step - loss: 0.1479 -
accuracy: 0.9627 - val_loss: 0.2943 - val_accuracy: 0.9032
Epoch 10/60
103/103 [==============================] - 31s 304ms/step - loss: 0.0868 -
accuracy: 0.9789 - val_loss: 0.6690 - val_accuracy: 0.8516
Epoch 11/60
103/103 [==============================] - 31s 302ms/step - loss: 0.1250 -
accuracy: 0.9643 - val_loss: 0.6367 - val_accuracy: 0.8645
Epoch 12/60
103/103 [==============================] - 31s 302ms/step - loss: 0.1167 -
accuracy: 0.9692 - val_loss: 0.6709 - val_accuracy: 0.8839
Epoch 13/60
103/103 [==============================] - 31s 305ms/step - loss: 0.1008 -
accuracy: 0.9789 - val_loss: 0.6717 - val_accuracy: 0.8645
Epoch 14/60
103/103 [==============================] - 31s 305ms/step - loss: 0.0685 -
accuracy: 0.9773 - val_loss: 0.5090 - val_accuracy: 0.8968
Epoch 15/60
103/103 [==============================] - 31s 303ms/step - loss: 0.0885 -
accuracy: 0.9724 - val_loss: 0.6785 - val_accuracy: 0.8774
Epoch 16/60
103/103 [==============================] - 31s 303ms/step - loss: 0.1010 -
accuracy: 0.9756 - val_loss: 0.7216 - val_accuracy: 0.8774
Epoch 17/60
103/103 [==============================] - 31s 303ms/step - loss: 0.0710 -
accuracy: 0.9838 - val_loss: 0.5565 - val_accuracy: 0.8839
Epoch 18/60
103/103 [==============================] - 31s 304ms/step - loss: 0.0783 -
accuracy: 0.9789 - val_loss: 0.5113 - val_accuracy: 0.8774
Epoch 19/60
```

```
103/103 [==============================] - 31s 304ms/step - loss: 0.0628 -
accuracy: 0.9854 - val_loss: 0.7047 - val_accuracy: 0.8968
```

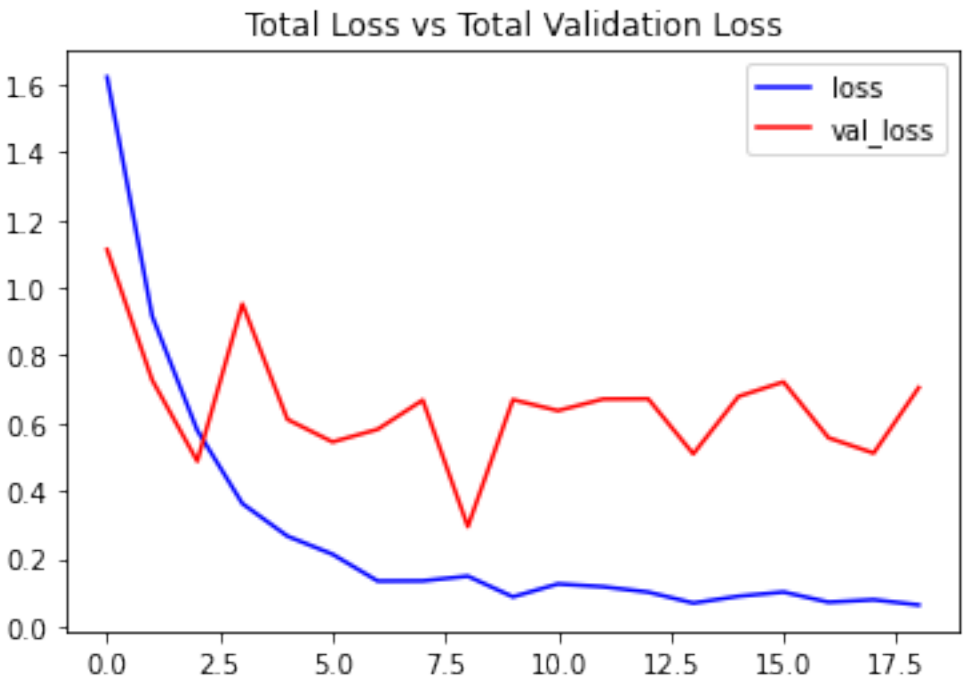
0.2 Evaluate on test set

[14]:
```
# Evaluate the trained model.
model_prediction = model.evaluate(features_test, labels_test)
```

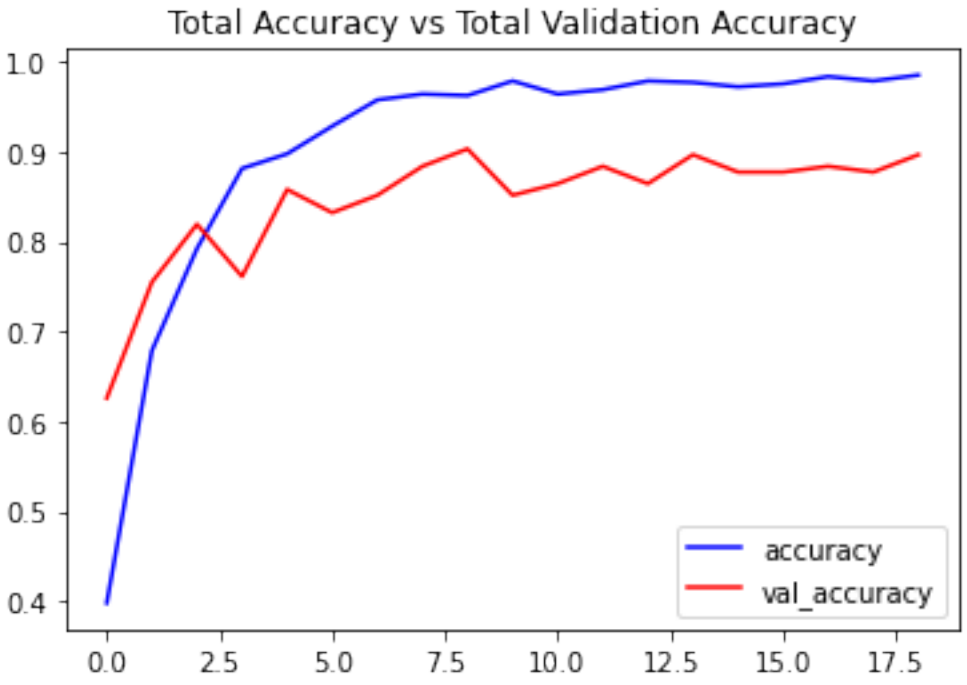
```
7/7 [==============================] - 1s 121ms/step - loss: 0.4652 - accuracy:
0.8964
```

[15]:
```
def plot_metric(history, metric_name_1, metric_name_2, plot_name):
    '''
    This function will plot the metrics passed to it in a graph.
    Args:
        model_training_history: A history object containing a record of␣
↪training and validation
                                loss values and metrics values at successive␣
↪epochs
        metric_name_1:          The name of the first metric that needs to be␣
↪plotted in the graph.
        metric_name_2:          The name of the second metric that needs to be␣
↪plotted in the graph.
        plot_name:              The title of the graph.
    '''

    # Get metric values using metric names as identifiers.
    metric_value_1 = history.history[metric_name_1]
    metric_value_2 = history.history[metric_name_2]

    # Construct a range object which will be used as x-axis (horizontal plane)␣
↪of the graph.
    epochs = range(len(metric_value_1))

    # Plot the Graph.
    plt.plot(epochs, metric_value_1, 'blue', label = metric_name_1)
    plt.plot(epochs, metric_value_2, 'red', label = metric_name_2)

    # Add title to the plot.
    plt.title(str(plot_name))

    # Add legend to the plot.
    plt.legend()
```

[16]:
```
# Visualize the training and validation loss metrices.
plot_metric(history, 'loss', 'val_loss', 'Total Loss vs Total Validation Loss')
```



[17]:
```
# Visualize the training and validation accuracy metrices.
plot_metric(history, 'accuracy', 'val_accuracy', 'Total Accuracy vs Total␣
↪Validation Accuracy')
```



CNN-Transformer

November 26, 2023

[1]:
```
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense,Dropout,GlobalMaxPool1D
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical,plot_model

import pandas as pd
import numpy as np
import imageio
from numpy import random
import cv2
import os
%matplotlib inline
from tensorflow.keras.callbacks import EarlyStopping
```

[2]:
```
dataset = np.load('project_human_activity.npz')
```

[3]:
```
features = dataset['arr_0']
labels = dataset['arr_1']
```

[4]:
```
np.unique(labels)
```

[4]:
```
array([0, 1, 2, 3, 4, 5, 6])
```

[5]:
```
features.shape
```

[5]:
```
(964, 20, 100, 100, 3)
```

[6]:
```
features[0].shape
```

[6]:
```
(20, 100, 100, 3)
```

[7]:
```
features[0][0].shape
```

[7]:
```
(100, 100, 3)
```

[8]:
```
type(features)
```

[8]:
```
numpy.ndarray
```

[9]:
```
'''pretrained_base = DenseNet121(include_top=False,
                                 weights='imagenet',
                                 pooling='avg',
                                 input_shape=(100,100,3))'''
```

[9]:
```
"pretrained_base = DenseNet121(include_top=False,\n
weights='imagenet',\n                                    pooling='avg',\n
input_shape=(100,100,3))"
```

[10]:
```
'''def get_features(dataset):
    final_features = []
    for each_video in dataset:
        frame_array = []
        processed_frame = preprocess_input(frame)
        features_pretrained = pretrained_base.predict(processed_frame)
        final_features.append(features_pretrained)
    return np.concatenate(final_features)'''
```

[10]:
```
'def get_features(dataset):\n    final_features = []\n    for each_video in
dataset:\n        frame_array = []\n        processed_frame =
preprocess_input(frame)\n        features_pretrained =
pretrained_base.predict(processed_frame)\n
final_features.append(features_pretrained)\n    return
np.concatenate(final_features)'
```

[11]:
```
#np.savez('features_extracted_transformer.npz', features_extracted)
```

[12]:
```
new_features_dataset = np.load('features_extracted_transformer.npz')
```

[13]:
```
trans_features = new_features_dataset['arr_0']
```

0.1 From here

[14]:
```
trans_features.shape
```

[14]:
```
(19280, 1024)
```

[15]:
```
len(trans_features)
```

[15]:
```
19280
```

[16]:
```
split_feature = []
step = 20
```

```
for i in range(1, 964+1):
    each_video = []
    for item in trans_features[(i-1)*step:i*step]:
        each_video.append(item)
    split_feature.append(each_video)
```

[17]:
```
final_feature = np.array(split_feature)
```

[18]:
```
final_feature.shape
```

[18]: (964, 20, 1024)

[19]:
```
one_hot_encoded_labels = to_categorical(labels)
```

[20]:
```
#split the train and test data
features_train, features_test, labels_train, labels_test =␣
↪train_test_split(final_feature, one_hot_encoded_labels, test_size = 0.2,␣
↪shuffle = True, random_state = 123)
```

[21]:
```
len(features_train), len(labels_train)
```

[21]: (771, 771)

0.2 Build the model

[22]:
```
sequence_length = 20
```

[23]:
```
# select 7 classes we will use in the model
data_classes = ['JumpRope', 'Kayaking', 'Lunges', 'Diving', 'PlayingGuitar',␣
↪'PlayingPiano', 'PlayingViolin']
```

[24]:
```
class PositionalEmbedding(layers.Layer):
    def __init__(self,sequence_length,output_dim,**kwargs):
        super().__init__(**kwargs)
        self.position_embedding=layers.Embedding(input_dim=sequence_length,
                                                 output_dim=output_dim)

        self.sequence_length=sequence_length
        self.output_dim=output_dim

    def call(self,inputs):
        length=tf.shape(inputs)[1]
        positions=tf.range(start=0,limit=length,delta=1)
        embedded_positions=self.position_embedding(positions)
        return inputs +embedded_positions

    def compute_mask(self,inputs,mask=None):
```

```
        mask=tf.reduce_any(tf.cast(inputs,'bool'),axis=-1)
        return mask
```

[25]:
```
class TransformerEncoder(layers.Layer):
    def __init__(self, embed_dim, dense_dim, num_heads, **kwargs):
        super().__init__(**kwargs)
        self.embed_dim = embed_dim
        self.dense_dim = dense_dim
        self.num_heads = num_heads
        self.attention = layers.MultiHeadAttention(
            num_heads=num_heads, key_dim=embed_dim, dropout=0.4
        )
        self.dense_proj = tf.keras.Sequential(
            [layers.Dense(dense_dim, activation=tf.nn.gelu), layers.
↪Dense(embed_dim),]
        )
        self.layernorm_1 = layers.LayerNormalization()
        self.layernorm_2 = layers.LayerNormalization()

    def call(self, inputs, mask=None):
        if mask is not None:
            mask = mask[:, tf.newaxis, :]

        attention_output = self.attention(inputs, inputs, attention_mask=mask)
        proj_input = self.layernorm_1(inputs + attention_output)
        proj_output = self.dense_proj(proj_input)
        return self.layernorm_2(proj_input + proj_output)
```

[26]:
```
def get_compiled_model():
    sequence_length=20
    embed_dim=1024
    dense_dim=4
    num_heads=1
    classes=len(data_classes)
    lr=0.001
    inputs=tf.keras.Input(shape=(sequence_length,embed_dim))
    ␣
↪x=PositionalEmbedding(sequence_length,embed_dim,name='Frame_positional_embedding')(inputs)
    ␣
↪x=TransformerEncoder(embed_dim,dense_dim,num_heads,name='TransformerEncoder')(x)
    x=GlobalMaxPool1D()(x)
    x=Dropout(0.5)(x)

    outputs=Dense(len(data_classes),activation='softmax')(x)
    model=tf.keras.Model(inputs,outputs,name='transformer')
    model.compile(optimizer=Adam(learning_rate=lr),␣
↪loss='categorical_crossentropy', metrics=['accuracy'])
```

```
    return model
```

[27]:
```
model=get_compiled_model()
```

[28]:
```
model.summary()
```

```
Model: "transformer"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 20, 1024)]        0
_________________________________________________________________
Frame_positional_embedding ( (None, 20, 1024)          20480
_________________________________________________________________
TransformerEncoder (Transfor (None, 20, 1024)          4211716
_________________________________________________________________
global_max_pooling1d (Global (None, 1024)              0
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 7175
=================================================================
Total params: 4,239,371
Trainable params: 4,239,371
Non-trainable params: 0
_________________________________________________________________
```

0.2.1 Train the model

[29]:
```
saved_filespath='saved_model'
checkpoint=tf.keras.callbacks.
 ↪ModelCheckpoint(saved_filespath,saved_best_only=True,saved_weights_only=True,verbose=1)
history = model.fit(x=features_train,y=labels_train,validation_split=0.
 ↪2,verbose=1,epochs=50,shuffle=True,
             callbacks=[checkpoint])
```

```
Epoch 1/50
20/20 [==============================] - 2s 23ms/step - loss: 3.9316 - accuracy:
0.1672 - val_loss: 2.6217 - val_accuracy: 0.1032

Epoch 00001: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 2/50
 1/20 [>…] - ETA: 0s - loss: 2.7737 - accuracy:
0.0938

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 2.5644 - accuracy:
0.1818 - val_loss: 2.1220 - val_accuracy: 0.1097

Epoch 00002: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 3/50
 8/20 [===========>…] - ETA: 0s - loss: 2.6257 - accuracy:
0.1875

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 2.4281 - accuracy:
0.2143 - val_loss: 1.9849 - val_accuracy: 0.2387

Epoch 00003: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets
```

```
INFO:tensorflow:Assets written to: saved_model/assets

Epoch 4/50
16/20 [=======================>…] - ETA: 0s - loss: 2.1072 - accuracy:
0.2676

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 2.0591 - accuracy:
0.2825 - val_loss: 2.1374 - val_accuracy: 0.2903

Epoch 00004: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 5/50
 8/20 [===========>…] - ETA: 0s - loss: 1.9048 - accuracy:
0.3477

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 1.7559 - accuracy:
0.3782 - val_loss: 1.4987 - val_accuracy: 0.4581

Epoch 00005: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets
```

```
Epoch 6/50
 1/20 [>…] - ETA: 0s - loss: 1.5819 - accuracy:
0.5000

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 1.5963 - accuracy:
0.4594 - val_loss: 1.3383 - val_accuracy: 0.4258

Epoch 00006: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 7/50
 1/20 [>…] - ETA: 0s - loss: 1.3227 - accuracy:
0.3750

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 1.4720 - accuracy:
0.4675 - val_loss: 1.7457 - val_accuracy: 0.4065

Epoch 00007: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 8/50
 9/20 [============>…] - ETA: 0s - loss: 1.7185 - accuracy:
```

```
0.4757

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 8ms/step - loss: 1.5352 - accuracy:
0.4919 - val_loss: 1.8151 - val_accuracy: 0.4065

Epoch 00008: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 9/50
 1/20 [>…] - ETA: 0s - loss: 1.2341 - accuracy:
0.5625

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 1.0804 - accuracy:
0.6039 - val_loss: 1.5140 - val_accuracy: 0.4516

Epoch 00009: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 10/50
16/20 [=======================>…] - ETA: 0s - loss: 1.0655 - accuracy:
0.6113
```

```
/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 1.0314 - accuracy:
0.6282 - val_loss: 1.4092 - val_accuracy: 0.4903

Epoch 00010: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 11/50
15/20 [=====================>…] - ETA: 0s - loss: 0.9756 - accuracy:
0.6667

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.9451 - accuracy:
0.6607 - val_loss: 1.2979 - val_accuracy: 0.5677

Epoch 00011: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 12/50
 8/20 [===========>…] - ETA: 0s - loss: 0.7706 - accuracy:
0.7148

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
```

```
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.7897 - accuracy:
0.7127 - val_loss: 0.8093 - val_accuracy: 0.7226

Epoch 00012: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 13/50
16/20 [======================>...] - ETA: 0s - loss: 0.7774 - accuracy:
0.7246

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.7679 - accuracy:
0.7273 - val_loss: 1.1312 - val_accuracy: 0.6516

Epoch 00013: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 14/50
16/20 [======================>...] - ETA: 0s - loss: 0.8262 - accuracy:
0.6953

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
```

```
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.7950 - accuracy:
0.7013 - val_loss: 0.9920 - val_accuracy: 0.6452

Epoch 00014: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 15/50
 8/20 [===========>…] - ETA: 0s - loss: 0.6431 - accuracy:
0.7734

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.6615 - accuracy:
0.7516 - val_loss: 1.3624 - val_accuracy: 0.5806

Epoch 00015: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 16/50
 8/20 [===========>…] - ETA: 0s - loss: 1.1051 - accuracy:
0.6484

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)
```

```
20/20 [==============================] - 0s 9ms/step - loss: 0.9564 - accuracy:
0.6672 - val_loss: 1.0672 - val_accuracy: 0.6065

Epoch 00016: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 17/50
16/20 [======================>...] - ETA: 0s - loss: 0.6349 - accuracy:
0.7754

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.6189 - accuracy:
0.7808 - val_loss: 0.8683 - val_accuracy: 0.7290

Epoch 00017: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 18/50
 8/20 [===========>...] - ETA: 0s - loss: 0.6331 - accuracy:
0.7656

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.7868 - accuracy:
0.7321 - val_loss: 1.2906 - val_accuracy: 0.6194
```

```
Epoch 00018: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 19/50
 7/20 [=========>…] - ETA: 0s - loss: 0.8833 - accuracy:
0.6696

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.7096 - accuracy:
0.7354 - val_loss: 1.0559 - val_accuracy: 0.6581

Epoch 00019: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 20/50
16/20 [=======================>…] - ETA: 0s - loss: 0.5343 - accuracy:
0.8105

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.5592 - accuracy:
0.8019 - val_loss: 0.8269 - val_accuracy: 0.6516

Epoch 00020: saving model to saved_model
```

```
WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 21/50
16/20 [======================>…] - ETA: 0s - loss: 0.5639 - accuracy:
0.7949

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.5601 - accuracy:
0.7955 - val_loss: 1.1443 - val_accuracy: 0.6581

Epoch 00021: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 22/50
16/20 [======================>…] - ETA: 0s - loss: 0.4833 - accuracy:
0.8320

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.4990 - accuracy:
0.8295 - val_loss: 1.5969 - val_accuracy: 0.6194

Epoch 00022: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
```

```
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 23/50
 8/20 [===========>…] - ETA: 0s - loss: 0.4594 - accuracy:
0.8359

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 8ms/step - loss: 0.4189 - accuracy:
0.8539 - val_loss: 1.0498 - val_accuracy: 0.7226

Epoch 00023: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 24/50
 8/20 [===========>…] - ETA: 0s - loss: 0.3238 - accuracy:
0.8789

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.3833 - accuracy:
0.8588 - val_loss: 1.3999 - val_accuracy: 0.6194

Epoch 00024: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
```

```
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 25/50
 7/20 [=========>…] - ETA: 0s - loss: 0.8247 - accuracy:
0.7321

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 1.0498 - accuracy:
0.6786 - val_loss: 1.4136 - val_accuracy: 0.5935

Epoch 00025: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 26/50
16/20 [=======================>…] - ETA: 0s - loss: 0.6422 - accuracy:
0.7754

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.6212 - accuracy:
0.7808 - val_loss: 1.1167 - val_accuracy: 0.6645

Epoch 00026: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 27/50
 7/20 [=========>…] - ETA: 0s - loss: 0.4751 - accuracy:
0.8304

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.3954 - accuracy:
0.8409 - val_loss: 1.1189 - val_accuracy: 0.6774

Epoch 00027: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 28/50
16/20 [=======================>…] - ETA: 0s - loss: 0.3671 - accuracy:
0.8652

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.3632 - accuracy:
0.8653 - val_loss: 1.0840 - val_accuracy: 0.7290

Epoch 00028: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets
```

```
INFO:tensorflow:Assets written to: saved_model/assets

Epoch 29/50
16/20 [=======================>…] - ETA: 0s - loss: 0.3054 - accuracy:
0.8965

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.3043 - accuracy:
0.8945 - val_loss: 1.0933 - val_accuracy: 0.6774

Epoch 00029: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 30/50
16/20 [=======================>…] - ETA: 0s - loss: 0.4007 - accuracy:
0.8750

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 8ms/step - loss: 0.3860 - accuracy:
0.8750 - val_loss: 1.2045 - val_accuracy: 0.6839

Epoch 00030: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets
```

```
Epoch 31/50
16/20 [=======================>…] - ETA: 0s - loss: 0.5956 - accuracy:
0.8145

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 8ms/step - loss: 0.5690 - accuracy:
0.8198 - val_loss: 0.8338 - val_accuracy: 0.7032

Epoch 00031: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 32/50
16/20 [=======================>…] - ETA: 0s - loss: 0.3954 - accuracy:
0.8574

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.3844 - accuracy:
0.8653 - val_loss: 1.1631 - val_accuracy: 0.6710

Epoch 00032: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 33/50
16/20 [=======================>…] - ETA: 0s - loss: 0.3855 - accuracy:
```

```
0.8535

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.3637 - accuracy:
0.8604 - val_loss: 1.9630 - val_accuracy: 0.5871

Epoch 00033: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 34/50
 1/20 [>…] - ETA: 0s - loss: 0.4616 - accuracy:
0.8125

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.3419 - accuracy:
0.8750 - val_loss: 0.7778 - val_accuracy: 0.7355

Epoch 00034: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 35/50
 8/20 [===========>…] - ETA: 0s - loss: 0.1850 - accuracy:
0.9453
```

```
/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.2003 - accuracy:
0.9318 - val_loss: 1.3137 - val_accuracy: 0.6839

Epoch 00035: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 36/50
16/20 [=======================>…] - ETA: 0s - loss: 0.3711 - accuracy:
0.8809

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.3667 - accuracy:
0.8766 - val_loss: 0.8763 - val_accuracy: 0.7226

Epoch 00036: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 37/50
16/20 [=======================>…] - ETA: 0s - loss: 0.2086 - accuracy:
0.9160

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
```

```
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 8ms/step - loss: 0.2138 - accuracy:
0.9123 - val_loss: 0.7567 - val_accuracy: 0.7742

Epoch 00037: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 38/50
 7/20 [=========>…] - ETA: 0s - loss: 0.1542 - accuracy:
0.9554

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.3202 - accuracy:
0.8912 - val_loss: 0.8727 - val_accuracy: 0.7161

Epoch 00038: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 39/50
 7/20 [=========>…] - ETA: 0s - loss: 0.4510 - accuracy:
0.8482

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
```

```
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.3456 - accuracy:
0.8766 - val_loss: 1.0527 - val_accuracy: 0.7226

Epoch 00039: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 40/50
 8/20 [===========>…] - ETA: 0s - loss: 0.3341 - accuracy:
0.8633

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 8ms/step - loss: 0.3300 - accuracy:
0.8750 - val_loss: 0.9319 - val_accuracy: 0.7548

Epoch 00040: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 41/50
16/20 [=======================>…] - ETA: 0s - loss: 0.1827 - accuracy:
0.9336

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)
```

```
20/20 [==============================] - 0s 9ms/step - loss: 0.1961 - accuracy:
0.9253 - val_loss: 1.5677 - val_accuracy: 0.6839

Epoch 00041: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 42/50
 8/20 [===========>…] - ETA: 0s - loss: 0.4155 - accuracy:
0.8672

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 8ms/step - loss: 0.4646 - accuracy:
0.8458 - val_loss: 1.3079 - val_accuracy: 0.6645

Epoch 00042: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 43/50
17/20 [========================>…] - ETA: 0s - loss: 0.2951 - accuracy:
0.9062

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.2881 - accuracy:
0.9091 - val_loss: 0.6623 - val_accuracy: 0.7871
```

```
Epoch 00043: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 44/50
16/20 [=======================>…] - ETA: 0s - loss: 0.1377 - accuracy:
0.9531

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 8ms/step - loss: 0.1496 - accuracy:
0.9464 - val_loss: 0.9121 - val_accuracy: 0.7548

Epoch 00044: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 45/50
 8/20 [===========>…] - ETA: 0s - loss: 0.1286 - accuracy:
0.9531

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.1434 - accuracy:
0.9529 - val_loss: 1.8474 - val_accuracy: 0.6710

Epoch 00045: saving model to saved_model
```

```
WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 46/50
16/20 [=======================>…] - ETA: 0s - loss: 0.1990 - accuracy:
0.9238

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.2183 - accuracy:
0.9156 - val_loss: 0.8746 - val_accuracy: 0.7742

Epoch 00046: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 47/50
15/20 [=====================>…] - ETA: 0s - loss: 0.1303 - accuracy:
0.9521

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.1443 - accuracy:
0.9513 - val_loss: 0.9211 - val_accuracy: 0.7548

Epoch 00047: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
```

```
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 48/50
16/20 [======================>…] - ETA: 0s - loss: 0.3690 - accuracy:
0.8613

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.3496 - accuracy:
0.8718 - val_loss: 1.4845 - val_accuracy: 0.7032

Epoch 00048: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 49/50
 8/20 [===========>…] - ETA: 0s - loss: 0.2573 - accuracy:
0.8945

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 8ms/step - loss: 0.2833 - accuracy:
0.8815 - val_loss: 1.0459 - val_accuracy: 0.7355

Epoch 00049: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
```

```
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets

Epoch 50/50
 1/20 [>…] - ETA: 0s - loss: 0.0520 - accuracy:
1.0000

/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)

20/20 [==============================] - 0s 9ms/step - loss: 0.1407 - accuracy:
0.9497 - val_loss: 0.8089 - val_accuracy: 0.7548


Epoch 00050: saving model to saved_model

WARNING:absl:Found untraced functions such as embedding_layer_call_fn,
embedding_layer_call_and_return_conditional_losses,
multi_head_attention_layer_call_fn,
multi_head_attention_layer_call_and_return_conditional_losses,
layer_normalization_layer_call_fn while saving (showing 5 of 50). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets
/fs/ess/PGS0333/BA_64061_KSU/jupyter/lib64/python3.6/site-
packages/keras/utils/generic_utils.py:497: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask
layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)
```

[30]:
```
'''# Create callback.
callbacks = EarlyStopping(monitor = 'val_loss', patience = 10, mode = 'min',␣
 ↪restore_best_weights = True)

# Compile the model
#model.compile(loss = 'categorical_crossentropy', optimizer = 'rmsprop',␣
 ↪metrics = ["accuracy"])

# Start training the model.
history = model.fit(x = features_train, y = labels_train, epochs = 60,␣
 ↪batch_size = 6,
                                                shuffle = True, validation_split =␣
 ↪0.2, callbacks = callbacks)'''
```

[30]:
```
'# Create callback.\ncallbacks = EarlyStopping(monitor = \'val_loss\', patience
= 10, mode = \'min\', restore_best_weights = True)\n\n# Compile the
model\n#model.compile(loss = \'categorical_crossentropy\', optimizer =
\'rmsprop\', metrics = ["accuracy"])\n\n# Start training the model.\nhistory =
model.fit(x = features_train, y = labels_train, epochs = 60, batch_size = 6,\n
shuffle = True, validation_split = 0.2, callbacks = callbacks)'
```

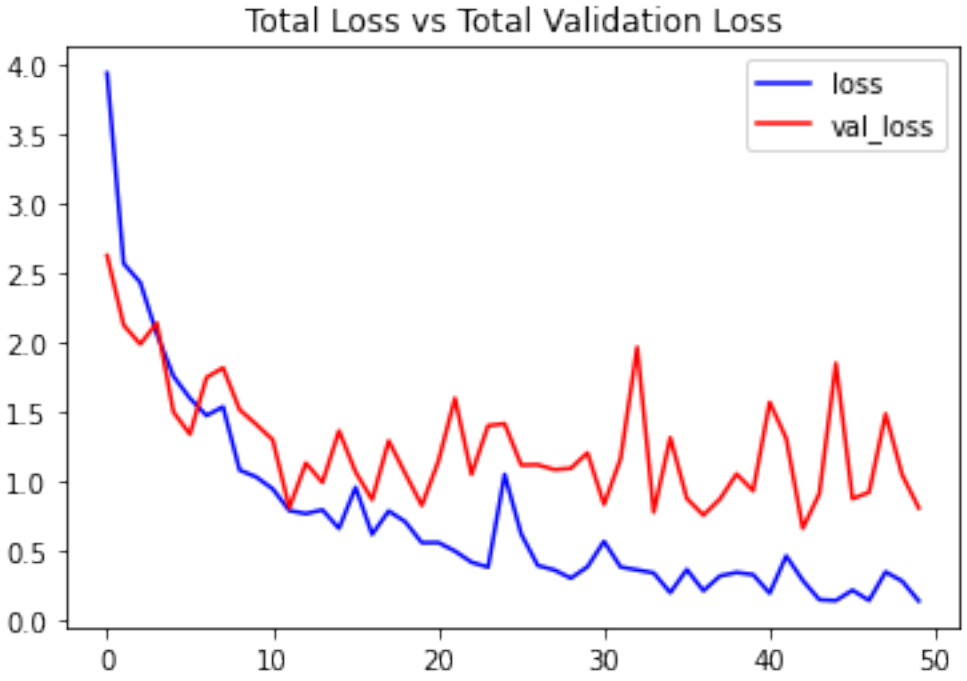
0.3 Evaluate on test set

[31]:
```
# Evaluate the trained model.
model_prediction = model.evaluate(features_test, labels_test, verbose=1)
```

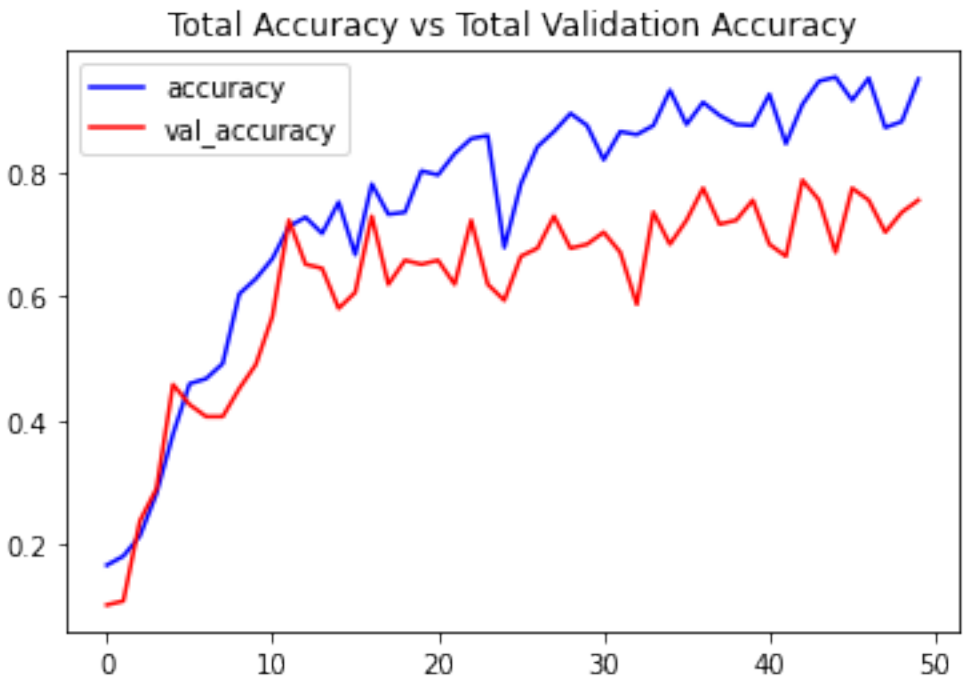
```
7/7 [==============================] - 0s 5ms/step - loss: 0.8870 - accuracy:
0.8031
```

[32]:
```
def plot_metric(history, metric_name_1, metric_name_2, plot_name):
    '''
    This function will plot the metrics passed to it in a graph.
    Args:
        model_training_history: A history object containing a record of␣
↪training and validation
                                loss values and metrics values at successive␣
↪epochs
        metric_name_1:          The name of the first metric that needs to be␣
↪plotted in the graph.
        metric_name_2:          The name of the second metric that needs to be␣
↪plotted in the graph.
        plot_name:              The title of the graph.
    '''

    # Get metric values using metric names as identifiers.
    metric_value_1 = history.history[metric_name_1]
    metric_value_2 = history.history[metric_name_2]

    # Construct a range object which will be used as x-axis (horizontal plane)␣
↪of the graph.
    epochs = range(len(metric_value_1))

    # Plot the Graph.
    plt.plot(epochs, metric_value_1, 'blue', label = metric_name_1)
    plt.plot(epochs, metric_value_2, 'red', label = metric_name_2)

    # Add title to the plot.
    plt.title(str(plot_name))

    # Add legend to the plot.
    plt.legend()
```

[33]:
```
# Visualize the training and validation loss metrices.
plot_metric(history, 'loss', 'val_loss', 'Total Loss vs Total Validation Loss')
```



[34]:
```
# Visualize the training and validation accuracy metrices.
plot_metric(history, 'accuracy', 'val_accuracy', 'Total Accuracy vs Total␣
↪Validation Accuracy')
```


Total Accuracy vs Total Validation Accuracy
accuracy
val_accuracy
0.8
0.6
0.4
0.2
0
10
20
30
40
50

[ ]: