In [1]:

```
import cv2
import numpy as np
import os
from matplotlib import pyplot as plt
import time
import mediapipe as mp
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical

mp_holistic = mp.solutions.holistic # Holistic model
mp_drawing = mp.solutions.drawing_utils # Drawing utilities

def mediapipe_detection(image, model):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # COLOR CONVERSION BGR 2 RGB
    image.flags.writeable = False                  # Image is no longer writeable
    results = model.process(image)                 # Make prediction
    image.flags.writeable = True                   # Image is now writeable
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR) # COLOR COVERSION RGB 2 BGR
    return image, results

def draw_landmarks(image, results):
    mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACEMESH_
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONN
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAN

def draw_styled_landmarks(image, results):
    # Draw face connections
    mp_drawing.draw_landmarks(image, results.face_landmarks,mp_holistic.FACEMESH_T
                             mp_drawing.DrawingSpec(color=(80,110,10), thickness=1
                             mp_drawing.DrawingSpec(color=(80,256,121), thickness=
                             )
    # Draw pose connections
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONN
                             mp_drawing.DrawingSpec(color=(80,22,10), thickness=2,
                             mp_drawing.DrawingSpec(color=(80,44,121), thickness=2
                             )
    # Draw left hand connections
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_
                             mp_drawing.DrawingSpec(color=(121,22,76), thickness=2
                             mp_drawing.DrawingSpec(color=(121,44,250), thickness=
                             )
    # Draw right hand connections
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAN
                             mp_drawing.DrawingSpec(color=(245,117,66), thickness=
                             mp_drawing.DrawingSpec(color=(245,66,230), thickness=
                             )

def extract_keypoints(results):
    pose = np.array([[res.x, res.y, res.z, res.visibility] for res in results.pose_
    face = np.array([[res.x, res.y, res.z] for res in results.face_landmarks.landm
    lh = np.array([[res.x, res.y, res.z] for res in results.left_hand_landmarks.la
    rh = np.array([[res.x, res.y, res.z] for res in results.right_hand_landmarks.l
    return np.concatenate([pose, face, lh, rh])
```

In [2]:

```
actions = np.array(['absent','day','Good morning','Green','hearing','How are you',
```

In [3]:

```
###################### Preprocess Data and Create Labels and Features ############
```

In [4]:

```
label_map = {label:num for num, label in enumerate(actions)}
```

In [5]:

```
label_map
```

Out[5]:

```
{'absent': 0,
 'day': 1,
 'Good morning': 2,
 'Green': 3,
 'hearing': 4,
 'How are you': 5,
 "I don't understand": 6,
 'maths': 7,
 'Maximum': 8,
 'sign': 9,
 'Take a photo': 10,
 'Talk': 11,
 'Thank you very much': 12,
 'time': 13,
 'up': 14}
```

In [6]:

```
DATA_PATH = os.path.join('NUMPY_DATA')

no_sequences = 50

# Videos are going to be 20 frames in length
sequence_length = 20
```

In [7]:

```
sequences, labels = [], []
for action in actions:
    for sequence in np.array(os.listdir(os.path.join(DATA_PATH, action))).astype(i
        window = []
        for frame_num in range(sequence_length):
            res = np.load(os.path.join(DATA_PATH, action, str(sequence), "{}.npy".
            window.append(res)
        sequences.append(window)
        labels.append(label_map[action])
```

In [8]:

```
np.array(sequences).shape
```

Out[8]:

```
(750, 20, 1662)
```

In [9]:

```
np.array(labels).shape
```

Out[9]:

```
(750,)
```

In [10]:

```
X = np.array(sequences)
```

In [11]:

```
X
```

```
Out[11]: array([[[ 5.02994180e-01,  3.24758410e-01, -3.82223904e-01, ...,
                   4.44711417e-01,  6.70127392e-01, -1.18734110e-02],
                 [ 5.12243450e-01,  3.21661860e-01, -4.94859517e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                 [ 5.14524460e-01,  3.16344827e-01, -5.16501069e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                 ...,
                 [ 5.17148495e-01,  3.28944236e-01, -3.31753612e-01, ...,
                   3.53993684e-01,  6.10053003e-01, -2.60807239e-02],
                 [ 5.16740263e-01,  3.29562753e-01, -3.55724633e-01, ...,
                   4.03432429e-01,  6.18105054e-01, -2.94835102e-02],
                 [ 5.16407371e-01,  3.30419987e-01, -3.75552952e-01, ...,
                   4.37740892e-01,  6.43600643e-01, -3.07861948e-03]],

                [[ 5.16196370e-01,  3.30445796e-01, -3.85003179e-01, ...,
                   4.42580223e-01,  6.51686370e-01, -8.67787935e-03],
                 [ 5.19390702e-01,  3.17165315e-01, -6.93943381e-01, ...,
                   4.72232133e-01,  8.29340398e-01,  5.12128288e-04],
                 [ 5.19550204e-01,  3.13520133e-01, -7.41623223e-01, ...,
                   4.74059463e-01,  8.50041866e-01,  2.58041220e-03],
                 ...,
                 [ 5.10387540e-01,  3.28033745e-01, -4.32974488e-01, ...,
                   4.10637677e-01,  6.28341675e-01, -2.16108616e-02],
                 [ 5.10064483e-01,  3.29009354e-01, -5.60820043e-01, ...,
                   4.13479686e-01,  6.17245793e-01, -1.16762882e-02],
                 [ 5.10051370e-01,  3.30393463e-01, -6.22609854e-01, ...,
                   4.12890226e-01,  6.22765303e-01, -1.30460905e-02]],

                [[ 5.97317278e-01,  5.26257098e-01, -6.65985286e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                 [ 6.07867599e-01,  4.46427494e-01, -6.63554966e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                 [ 6.17361486e-01,  4.29450989e-01, -6.51696801e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                 ...,
                 [ 6.11113667e-01,  4.49662954e-01, -7.46027887e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                 [ 6.11111820e-01,  4.49741364e-01, -8.30863655e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                 [ 6.10262990e-01,  4.48247552e-01, -8.51483166e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00]],

                ...,

                [[ 4.76174116e-01,  4.11153764e-01, -6.29921615e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                 [ 4.76191103e-01,  4.10693824e-01, -5.13342798e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                 [ 4.76704210e-01,  4.10682410e-01, -5.39849758e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                 ...,
                 [ 4.78713036e-01,  4.16913122e-01, -6.32822871e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                 [ 4.78356689e-01,  4.15582031e-01, -6.21686161e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                 [ 4.78214502e-01,  4.15212065e-01, -5.89577556e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00]],

                [[ 4.76952732e-01,  4.14357007e-01, -6.08422875e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                 [ 4.76989895e-01,  4.07903403e-01, -5.48847318e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                 [ 4.77120221e-01,  4.05184090e-01, -5.17263830e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                 ...,
                 [ 4.74425852e-01,  4.08896923e-01, -5.89305222e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                 [ 4.74369198e-01,  4.08926010e-01, -5.90117157e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                 [ 4.74402040e-01,  4.08274204e-01, -5.88497818e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00]],

                [[ 4.74526167e-01,  4.08284038e-01, -5.94386816e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                 [ 7.09984422e-01,  5.47799408e-01, -6.49519920e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                 [ 6.57025456e-01,  5.77999473e-01, -6.56331480e-01, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                 ...,
                 [ 6.44817054e-01,  5.63012004e-01, -5.04395187e-01, ...,
                   3.29716057e-01,  4.53725755e-01, -1.80265326e-02],
                 [ 6.44450068e-01,  5.63778162e-01, -5.10316193e-01, ...,
                   3.14921021e-01,  4.84795511e-01, -2.09916979e-02],
                 [ 6.43666387e-01,  5.64001441e-01, -4.96832192e-01, ...,
                   3.10511321e-01,  5.12432456e-01, -2.33853981e-02]]])
```

In [12]:

```
X.shape
```

Out[12]:

```
(750, 20, 1662)
```

In [13]:

```
y = to_categorical(labels).astype(int)
```

In [14]:

```
y
```

Out[14]:

```
array([[1, 0, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 1],
       [0, 0, 0, ..., 0, 0, 1],
       [0, 0, 0, ..., 0, 0, 1]])
```

In [15]:

```
print(y)
```

```
[[1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 1]
 [0 0 0 ... 0 0 1]
 [0 0 0 ... 0 0 1]]
```

In [47]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

In [48]:

```
X_train.shape
```

Out[48]:

```
(600, 20, 1662)
```

In [49]:

```
y_test.shape
```

Out[49]:

```
(150, 15)
```

In [50]:

```
X_test.shape
```

Out[50]:

```
(150, 20, 1662)
```

In [51]:

```
y_train.shape
```

Out[51]:

```
(600, 15)
```

In [52]:

```
actions.shape[0]
```

Out[52]:

```
15
```

In [71]:

```
##############Build and Train LSTM Neural Network###############

from tensorflow.keras.models import Sequential,load_model
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.callbacks import TensorBoard

log_dir = os.path.join('Logs_sc')
tb_callback = TensorBoard(log_dir=log_dir)
```

In [72]:

```
model = Sequential()
model.add(LSTM(128, return_sequences=True, activation='relu', input_shape=(20,1662
model.add(LSTM(128, return_sequences=True, activation='relu'))
model.add(LSTM(128, return_sequences=True, activation='relu'))
model.add(LSTM(64, return_sequences=False, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(actions.shape[0], activation='softmax'))
```

In [73]:

```
model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['categor
```

In [90]:

```
model.fit(X_train, y_train, epochs=50,callbacks=[tb_callback])
```

```
Epoch 1/50
19/19 [==============================] - 9s 176ms/step - loss: 0.0200 - categorica
l_accuracy: 0.9967
Epoch 2/50
19/19 [==============================] - 4s 180ms/step - loss: 0.0567 - categorica
l_accuracy: 0.9850
Epoch 3/50
19/19 [==============================] - 4s 181ms/step - loss: 0.0614 - categorica
l_accuracy: 0.9767
Epoch 4/50
19/19 [==============================] - 3s 177ms/step - loss: 0.0729 - categorica
l_accuracy: 0.9800
Epoch 5/50
19/19 [==============================] - 3s 173ms/step - loss: 0.2159 - categorica
l_accuracy: 0.9233
Epoch 6/50
19/19 [==============================] - 3s 128ms/step - loss: 0.1200 - categorica
l_accuracy: 0.9583
Epoch 7/50
19/19 [==============================] - 3s 129ms/step - loss: 0.0937 - categorica
l_accuracy: 0.9667
Epoch 8/50
19/19 [==============================] - 4s 183ms/step - loss: 0.0287 - categorica
l_accuracy: 0.9967
Epoch 9/50
19/19 [==============================] - 3s 172ms/step - loss: 0.0173 - categorica
l_accuracy: 0.9967
Epoch 10/50
19/19 [==============================] - 3s 127ms/step - loss: 0.0101 - categorica
l_accuracy: 1.0000
Epoch 11/50
19/19 [==============================] - 4s 183ms/step - loss: 0.0048 - categorica
l_accuracy: 1.0000
Epoch 12/50
19/19 [==============================] - 3s 174ms/step - loss: 0.0034 - categorica
l_accuracy: 1.0000
Epoch 13/50
19/19 [==============================] - 3s 172ms/step - loss: 0.0023 - categorica
l_accuracy: 1.0000
Epoch 14/50
19/19 [==============================] - 3s 174ms/step - loss: 0.0018 - categorica
l_accuracy: 1.0000
Epoch 15/50
19/19 [==============================] - 3s 172ms/step - loss: 0.0015 - categorica
l_accuracy: 1.0000
Epoch 16/50
19/19 [==============================] - 3s 178ms/step - loss: 0.0013 - categorica
l_accuracy: 1.0000
Epoch 17/50
19/19 [==============================] - 4s 183ms/step - loss: 0.0012 - categorica
l_accuracy: 1.0000
Epoch 18/50
19/19 [==============================] - 3s 177ms/step - loss: 0.0011 - categorica
l_accuracy: 1.0000
Epoch 19/50
19/19 [==============================] - 3s 174ms/step - loss: 9.9355e-04 - catego
rical_accuracy: 1.0000
Epoch 20/50
19/19 [==============================] - 3s 175ms/step - loss: 9.4926e-04 - catego
rical_accuracy: 1.0000
Epoch 21/50
19/19 [==============================] - 3s 137ms/step - loss: 8.3725e-04 - catego
rical_accuracy: 1.0000
Epoch 22/50
```

```
19/19 [==============================] - 3s 127ms/step - loss: 7.6459e-04 - catego
rical_accuracy: 1.0000
Epoch 23/50
19/19 [==============================] - 3s 166ms/step - loss: 7.0898e-04 - catego
rical_accuracy: 1.0000
Epoch 24/50
19/19 [==============================] - 3s 179ms/step - loss: 6.6418e-04 - catego
rical_accuracy: 1.0000
Epoch 25/50
19/19 [==============================] - 3s 177ms/step - loss: 6.2548e-04 - catego
rical_accuracy: 1.0000
Epoch 26/50
19/19 [==============================] - 3s 128ms/step - loss: 5.8340e-04 - catego
rical_accuracy: 1.0000
Epoch 27/50
19/19 [==============================] - 3s 178ms/step - loss: 5.5098e-04 - catego
rical_accuracy: 1.0000
Epoch 28/50
19/19 [==============================] - 3s 176ms/step - loss: 5.1714e-04 - catego
rical_accuracy: 1.0000
Epoch 29/50
19/19 [==============================] - 3s 174ms/step - loss: 4.9164e-04 - catego
rical_accuracy: 1.0000
Epoch 30/50
19/19 [==============================] - 3s 126ms/step - loss: 4.6385e-04 - catego
rical_accuracy: 1.0000
Epoch 31/50
19/19 [==============================] - 4s 186ms/step - loss: 4.4104e-04 - catego
rical_accuracy: 1.0000
Epoch 32/50
19/19 [==============================] - 3s 177ms/step - loss: 4.1870e-04 - catego
rical_accuracy: 1.0000
Epoch 33/50
19/19 [==============================] - 4s 185ms/step - loss: 4.0059e-04 - catego
rical_accuracy: 1.0000
Epoch 34/50
19/19 [==============================] - 4s 183ms/step - loss: 3.8008e-04 - catego
rical_accuracy: 1.0000
Epoch 35/50
19/19 [==============================] - 3s 130ms/step - loss: 3.6493e-04 - catego
rical_accuracy: 1.0000
Epoch 36/50
19/19 [==============================] - 4s 207ms/step - loss: 3.4653e-04 - catego
rical_accuracy: 1.0000
Epoch 37/50
19/19 [==============================] - 3s 175ms/step - loss: 3.3258e-04 - catego
rical_accuracy: 1.0000
Epoch 38/50
19/19 [==============================] - 3s 175ms/step - loss: 3.1834e-04 - catego
rical_accuracy: 1.0000
Epoch 39/50
19/19 [==============================] - 3s 175ms/step - loss: 3.0561e-04 - catego
rical_accuracy: 1.0000
Epoch 40/50
19/19 [==============================] - 3s 126ms/step - loss: 2.9336e-04 - catego
rical_accuracy: 1.0000
Epoch 41/50
19/19 [==============================] - 4s 186ms/step - loss: 2.8154e-04 - catego
rical_accuracy: 1.0000
Epoch 42/50
19/19 [==============================] - 4s 184ms/step - loss: 2.6891e-04 - catego
rical_accuracy: 1.0000
Epoch 43/50
19/19 [==============================] - 4s 185ms/step - loss: 2.6252e-04 - catego
```

```
rical_accuracy: 1.0000
Epoch 44/50
19/19 [==============================] - 3s 177ms/step - loss: 2.5041e-04 - catego
rical_accuracy: 1.0000
Epoch 45/50
19/19 [==============================] - 3s 128ms/step - loss: 2.4308e-04 - catego
rical_accuracy: 1.0000
Epoch 46/50
19/19 [==============================] - 4s 181ms/step - loss: 2.3263e-04 - catego
rical_accuracy: 1.0000
Epoch 47/50
19/19 [==============================] - 3s 126ms/step - loss: 2.2458e-04 - catego
rical_accuracy: 1.0000
Epoch 48/50
19/19 [==============================] - 4s 188ms/step - loss: 2.1694e-04 - catego
rical_accuracy: 1.0000
Epoch 49/50
19/19 [==============================] - 4s 183ms/step - loss: 2.1097e-04 - catego
rical_accuracy: 1.0000
Epoch 50/50
19/19 [==============================] - 4s 182ms/step - loss: 2.0415e-04 - catego
rical_accuracy: 1.0000
```

Out[90]:

```
<keras.callbacks.History at 0x20468d7d100>
```

In [91]:

```python
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_7 (LSTM)               (None, 20, 128)           916992

 lstm_8 (LSTM)               (None, 20, 128)           131584

 lstm_9 (LSTM)               (None, 20, 128)           131584

 lstm_10 (LSTM)              (None, 64)                49408

 dense_8 (Dense)             (None, 64)                4160

 dense_9 (Dense)             (None, 64)                4160

 dense_10 (Dense)            (None, 64)                4160

 dense_11 (Dense)            (None, 15)                975

=================================================================
Total params: 1,243,023
Trainable params: 1,243,023
Non-trainable params: 0
_________________________________________________________________
```

In [92]:

```python
res = model.predict(X_test)
predict = model.predict(X_test)
```

```
5/5 [==============================] - 1s 79ms/step
5/5 [==============================] - 1s 79ms/step
```

In [97]:

```python
actions[np.argmax(res[53])]
```

Out[97]:

```
'How are you'
```

In [98]:

```python
actions[np.argmax(y_test[53])]
```

Out[98]:

```
'How are you'
```

In [99]:

```
model.save('sc_model_15(OT).h5')
```

In [100…

```
model=load_model('sc_model_15(OT).h5')
```

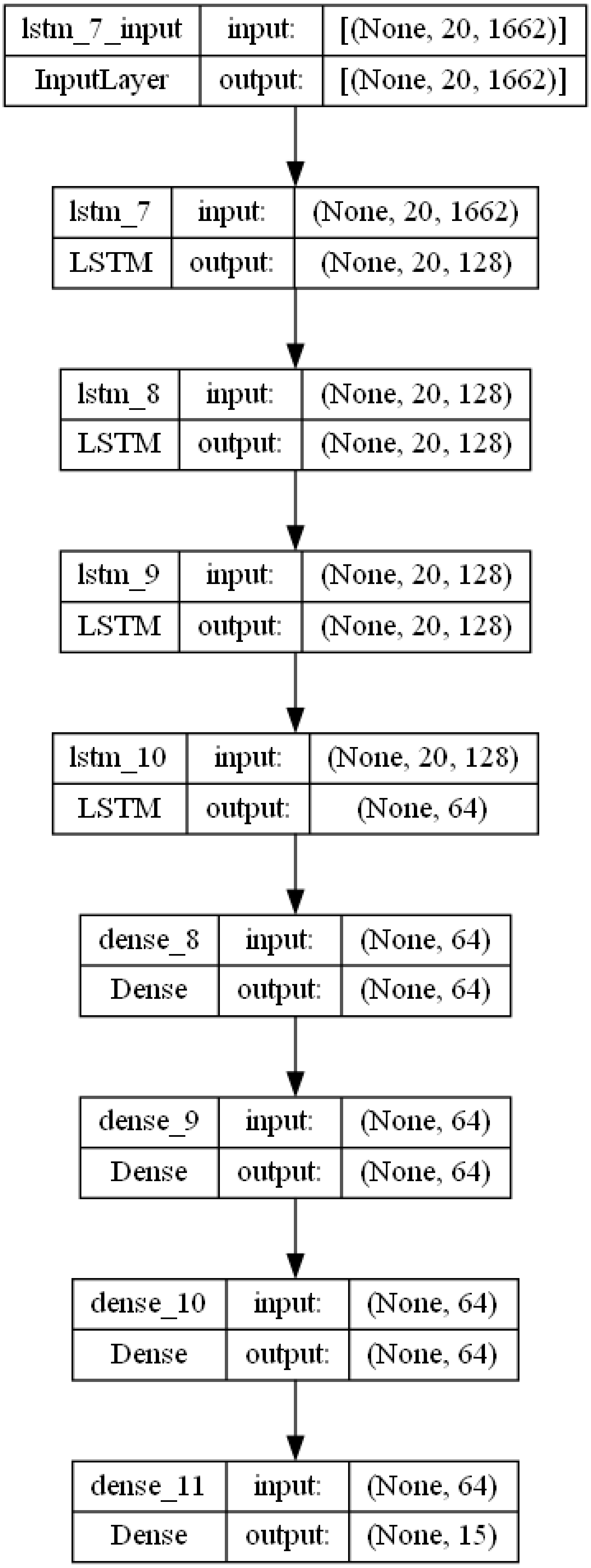
In [101…

```
from sklearn.metrics import multilabel_confusion_matrix, accuracy_score
```

In [102…

```
from keras.utils.vis_utils import plot_model
plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=Tru
```

Out[102]:



lstm_7_input | input: | [(None, 20, 1662)]
InputLayer | output: | [(None, 20, 1662)]

lstm_7 | input: | (None, 20, 1662)
LSTM | output: | (None, 20, 128)

lstm_8 | input: | (None, 20, 128)
LSTM | output: | (None, 20, 128)

lstm_9 | input: | (None, 20, 128)
LSTM | output: | (None, 20, 128)

lstm_10 | input: | (None, 20, 128)
LSTM | output: | (None, 64)

dense_8 | input: | (None, 64)
Dense | output: | (None, 64)

dense_9 | input: | (None, 64)
Dense | output: | (None, 64)

dense_10 | input: | (None, 64)
Dense | output: | (None, 64)

dense_11 | input: | (None, 64)
Dense | output: | (None, 15)

In [34]:

```
!pip install pydot
```

```
Requirement already satisfied: pydot in c:\users\dell\anaconda3\lib\site-packages
(1.4.2)
Requirement already satisfied: graphviz in c:\users\dell\anaconda3\lib\site-packag
es (0.20.1)
Requirement already satisfied: pyparsing>=2.1.4 in c:\users\dell\anaconda3\lib\sit
e-packages (from pydot) (3.0.4)
```

In [103…

```
yhat = model.predict(X_test)
```

```
5/5 [==============================] - 1s 91ms/step
```

In [104…

```
ytrue = np.argmax(y_test, axis=1).tolist()
yhat = np.argmax(yhat, axis=1).tolist()
```

In [105…

```
multilabel_confusion_matrix(ytrue, yhat)
```

Out[105]:

```
array([[[140,   0],
        [  3,   7]],

       [[140,   0],
        [  1,   9]],

       [[143,   2],
        [  0,   5]],

       [[138,   0],
        [  2,  10]],

       [[136,   1],
        [  0,  13]],

       [[137,   1],
        [  2,  10]],

       [[138,   0],
        [  1,  11]],

       [[144,   0],
        [  2,   4]],

       [[143,   0],
        [  1,   6]],

       [[130,   4],
        [  1,  15]],

       [[141,   0],
        [  1,   8]],

       [[140,   2],
        [  0,   8]],

       [[137,   2],
        [  1,  10]],

       [[139,   3],
        [  3,   5]],

       [[136,   3],
        [  0,  11]]], dtype=int64)
```

In [106…

```
accuracy_score(ytrue, yhat)
```

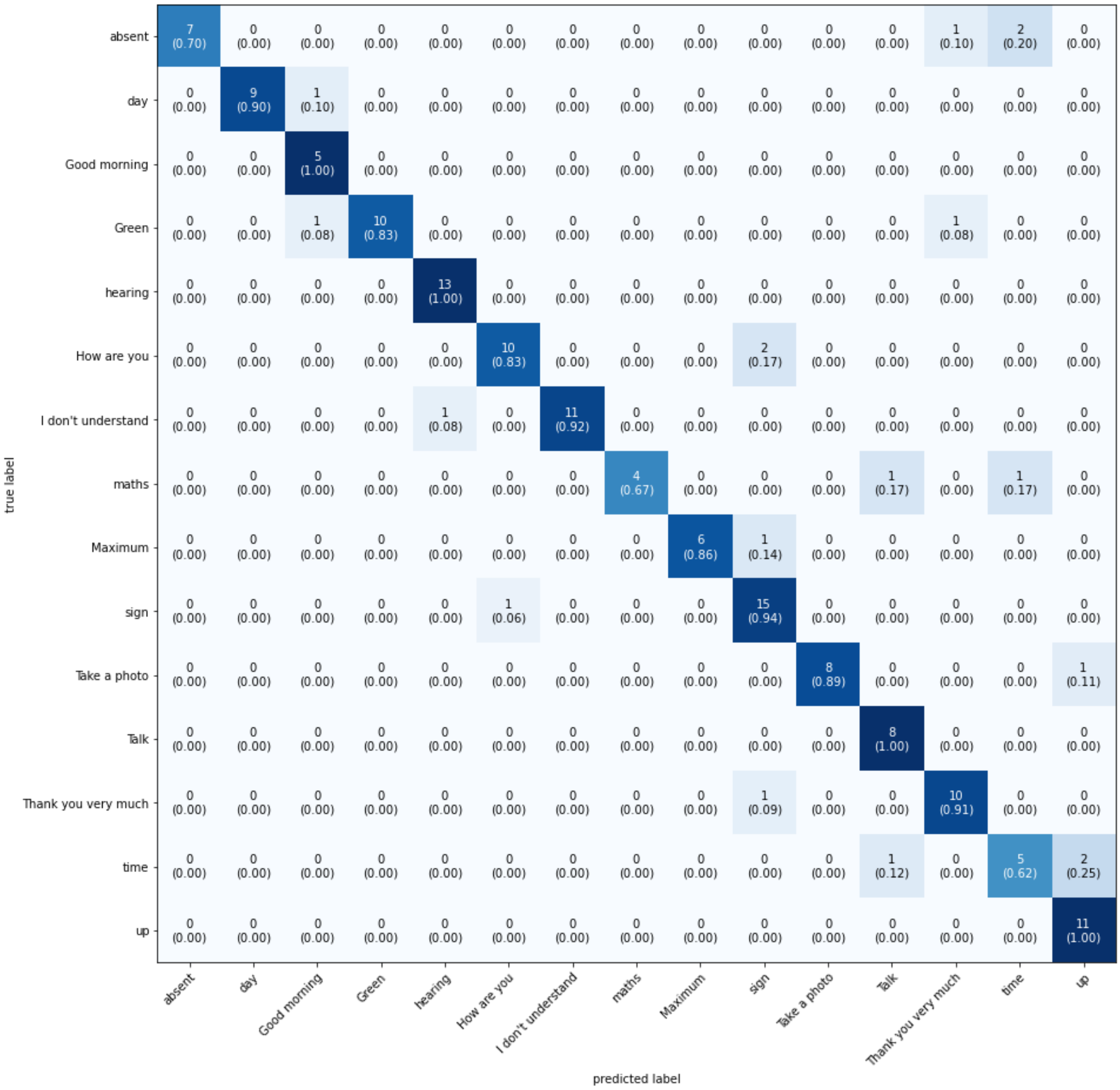
```
Out[106]: 0.88
```

```
In [107…  from mlxtend.plotting import plot_confusion_matrix
```

```
In [108…  from sklearn.metrics import confusion_matrix

          mat = confusion_matrix(ytrue,yhat)
          plot_confusion_matrix(conf_mat=mat, class_names=label_map,show_normed=True , figsi
```

```
Out[108]: (<Figure size 1080x1080 with 1 Axes>,
           <AxesSubplot:xlabel='predicted label', ylabel='true label'>)
```



	absent	day	Good morning	Green	hearing	How are you	I don't understand	maths	Maximum	sign	Take a photo	Talk	Thank you very much	time	up
absent	7 (0.70)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	1 (0.10)	2 (0.20)	0 (0.00)
day	0 (0.00)	9 (0.90)	1 (0.10)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)
Good morning	0 (0.00)	0 (0.00)	5 (1.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)
Green	0 (0.00)	0 (0.00)	1 (0.08)	10 (0.83)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	1 (0.08)	0 (0.00)	0 (0.00)
hearing	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	13 (1.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)
How are you	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	10 (0.83)	0 (0.00)	0 (0.00)	0 (0.00)	2 (0.17)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)
I don't understand	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	1 (0.08)	0 (0.00)	11 (0.92)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)
maths	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	4 (0.67)	0 (0.00)	0 (0.00)	0 (0.00)	1 (0.17)	0 (0.00)	1 (0.17)	0 (0.00)
Maximum	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	6 (0.86)	1 (0.14)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)
sign	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	1 (0.06)	0 (0.00)	0 (0.00)	0 (0.00)	15 (0.94)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)
Take a photo	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	8 (0.89)	0 (0.00)	0 (0.00)	0 (0.00)	1 (0.11)
Talk	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	8 (1.00)	0 (0.00)	0 (0.00)	0 (0.00)
Thank you very much	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	1 (0.09)	0 (0.00)	0 (0.00)	10 (0.91)	0 (0.00)	0 (0.00)
time	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	1 (0.12)	0 (0.00)	5 (0.62)	2 (0.25)
up	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	11 (1.00)

```
In [ ]:
```