mayankgupta9968@gmail.com_23

February 22, 2020

[1]:
```
# Importing Libraries
import pandas as pd
import numpy as np
```

[2]:
```
# Activities are the class labels
# It is a 6 class classification
ACTIVITIES = {
    0: 'WALKING',
    1: 'WALKING_UPSTAIRS',
    2: 'WALKING_DOWNSTAIRS',
    3: 'SITTING',
    4: 'STANDING',
    5: 'LAYING',
}

# Utility function to print the confusion matrix
def confusion_matrix(Y_true, Y_pred):
    Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_true, axis=1)])
    Y_pred = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_pred, axis=1)])

    return pd.crosstab(Y_true, Y_pred, rownames=['True'], colnames=['Pred'])
```

0.0.1 Data

[3]:
```
# Data directory
DATADIR = 'UCI_HAR_Dataset'
```

[4]:
```
# Raw data signals
# Signals are from Accelerometer and Gyroscope
# The signals are in x,y,z directions
# Sensor signals are filtered to have only body acceleration
# excluding the acceleration due to gravity
# Triaxial acceleration from the accelerometer is total acceleration
SIGNALS = [
    "body_acc_x",
    "body_acc_y",
    "body_acc_z",
```

```
    "body_gyro_x",
    "body_gyro_y",
    "body_gyro_z",
    "total_acc_x",
    "total_acc_y",
    "total_acc_z"
]
```

[5]:
```
# Utility function to read the data from csv file
def _read_csv(filename):
    return pd.read_csv(filename, delim_whitespace=True, header=None)

# Utility function to load the load
def load_signals(subset):
    signals_data = []

    for signal in SIGNALS:
        filename = f'UCI_HAR_Dataset/{subset}/Inertial Signals/
 ↪{signal}_{subset}.txt'
        signals_data.append(
            _read_csv(filename).as_matrix()
        )

    # Transpose is used to change the dimensionality of the output,
    # aggregating the signals by combination of sample/timestep.
    # Resultant shape is (7352 train/2947 test samples, 128 timesteps, 9␣
 ↪signals)
    return np.transpose(signals_data, (1, 2, 0))
```

[6]:
```
def load_y(subset):
    """
    The objective that we are trying to predict is a integer, from 1 to 6,
    that represents a human activity. We return a binary representation of
    every sample objective as a 6 bits vector using One Hot Encoding
    (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.
 ↪html)
    """
    filename = f'UCI_HAR_Dataset/{subset}/y_{subset}.txt'
    y = _read_csv(filename)[0]

    return pd.get_dummies(y).as_matrix()
```

[7]:
```
def load_data():
    """
    Obtain the dataset from multiple files.
    Returns: X_train, X_test, y_train, y_test
    """
    X_train, X_test = load_signals('train'), load_signals('test')
```

```
    y_train, y_test = load_y('train'), load_y('test')

    return X_train, X_test, y_train, y_test
```

[8]:
```
# Importing tensorflow
np.random.seed(42)
import tensorflow as tf
tf.set_random_seed(42)
```

[9]:
```
# Configuring a session
session_conf = tf.ConfigProto(
    intra_op_parallelism_threads=1,
    inter_op_parallelism_threads=1
)
```

[10]:
```
# Import Keras
from keras import backend as K
sess = tf.Session(graph=tf.get_default_graph(), config=session_conf)
K.set_session(sess)
```

```
Using TensorFlow backend.
```

[11]:
```
# Importing libraries
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers.core import Dense, Dropout
from keras.layers.normalization import BatchNormalization
```

[12]:
```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 32
```

[13]:
```
# Utility function to count the number of classes
def _count_classes(y):
    return len(set([tuple(category) for category in y]))
```

[14]:
```
# Loading the train and test data
X_train, X_test, Y_train, Y_test = load_data()
```

```
/Users/mayankgupta/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:12: FutureWarning: Method .as_matrix will be
removed in a future version. Use .values instead.
  if sys.path[0] == '':
/Users/mayankgupta/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:11: FutureWarning: Method .as_matrix will be
removed in a future version. Use .values instead.
  # This is added back by InteractiveShellApp.init_path()
```

[15]:
```
timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = _count_classes(Y_train)

print(timesteps)
print(input_dim)
print(len(X_train))
```

```
128
9
7352
```

- Defining the Architecture of LSTM

[16]:
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 32)                5376
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 198
=================================================================
Total params: 5,574
Trainable params: 5,574
Non-trainable params: 0
_________________________________________________________________
```

[17]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[18]:
```
# Training the model
model.fit(X_train,
          Y_train,
```

```
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
WARNING:tensorflow:From /Users/mayankgupta/anaconda3/lib/python3.7/site-
packages/tensorflow/python/ops/math_grad.py:1250:
add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /Users/mayankgupta/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:422: The name tf.global_variables
is deprecated. Please use tf.compat.v1.global_variables instead.

Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 17s 2ms/step - loss: 1.3008 -
accuracy: 0.4645 - val_loss: 1.0891 - val_accuracy: 0.5565
Epoch 2/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.9056 -
accuracy: 0.6171 - val_loss: 0.8277 - val_accuracy: 0.5942
Epoch 3/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.7434 -
accuracy: 0.6549 - val_loss: 0.7569 - val_accuracy: 0.6230
Epoch 4/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.6725 -
accuracy: 0.6800 - val_loss: 0.6941 - val_accuracy: 0.6651
Epoch 5/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.6236 -
accuracy: 0.7116 - val_loss: 0.6568 - val_accuracy: 0.7326
Epoch 6/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.5866 -
accuracy: 0.7333 - val_loss: 0.7696 - val_accuracy: 0.6763
Epoch 7/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.5055 -
accuracy: 0.7748 - val_loss: 0.6162 - val_accuracy: 0.7272
Epoch 8/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.4769 -
accuracy: 0.7791 - val_loss: 0.5323 - val_accuracy: 0.7465
Epoch 9/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.4243 -
accuracy: 0.7979 - val_loss: 0.6893 - val_accuracy: 0.7167
Epoch 10/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.4033 -
accuracy: 0.8096 - val_loss: 0.5631 - val_accuracy: 0.7326
Epoch 11/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.3785 -
```

```
accuracy: 0.8391 - val_loss: 0.5226 - val_accuracy: 0.7937
Epoch 12/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.3327 -
accuracy: 0.8791 - val_loss: 0.5721 - val_accuracy: 0.8694
Epoch 13/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.2857 -
accuracy: 0.9132 - val_loss: 0.4536 - val_accuracy: 0.8812
Epoch 14/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.2537 -
accuracy: 0.9207 - val_loss: 0.5414 - val_accuracy: 0.8748
Epoch 15/30
7352/7352 [==============================] - 14s 2ms/step - loss: 0.2421 -
accuracy: 0.9293 - val_loss: 0.4205 - val_accuracy: 0.8968
Epoch 16/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.2077 -
accuracy: 0.9331 - val_loss: 0.4868 - val_accuracy: 0.8785
Epoch 17/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.1950 -
accuracy: 0.9384 - val_loss: 0.5625 - val_accuracy: 0.8833
Epoch 18/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.1844 -
accuracy: 0.9419 - val_loss: 0.6079 - val_accuracy: 0.8738
Epoch 19/30
7352/7352 [==============================] - 14s 2ms/step - loss: 0.1846 -
accuracy: 0.9414 - val_loss: 0.4497 - val_accuracy: 0.8999
Epoch 20/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.1701 -
accuracy: 0.9459 - val_loss: 0.5215 - val_accuracy: 0.8795
Epoch 21/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1723 -
accuracy: 0.9450 - val_loss: 0.4698 - val_accuracy: 0.8887
Epoch 22/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1813 -
accuracy: 0.9448 - val_loss: 0.4783 - val_accuracy: 0.8795
Epoch 23/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.1579 -
accuracy: 0.9465 - val_loss: 0.4126 - val_accuracy: 0.8968
Epoch 24/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.1754 -
accuracy: 0.9448 - val_loss: 0.3679 - val_accuracy: 0.9111
Epoch 25/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.1827 -
accuracy: 0.9425 - val_loss: 0.9810 - val_accuracy: 0.8426
Epoch 26/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1565 -
accuracy: 0.9489 - val_loss: 0.3639 - val_accuracy: 0.9043
Epoch 27/30
7352/7352 [==============================] - 14s 2ms/step - loss: 0.1508 -
```

```
accuracy: 0.9484 - val_loss: 0.3858 - val_accuracy: 0.8996
Epoch 28/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.1461 -
accuracy: 0.9470 - val_loss: 0.4374 - val_accuracy: 0.9013
Epoch 29/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.1558 -
accuracy: 0.9449 - val_loss: 0.3596 - val_accuracy: 0.9043
Epoch 30/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.1712 -
accuracy: 0.9433 - val_loss: 0.4166 - val_accuracy: 0.8955
```

[18]:
```
<keras.callbacks.callbacks.History at 0x121401518>
```

[19]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 510        0        27        0                   0
SITTING                  2      381       105        1                   1
STANDING                 0       86       446        0                   0
WALKING                  0        0         0      454                  15
WALKING_DOWNSTAIRS       0        0         0        0                 419
WALKING_UPSTAIRS         0        7         0        4                  31

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            1
STANDING                           0
WALKING                           27
WALKING_DOWNSTAIRS                 1
WALKING_UPSTAIRS                 429
```

[20]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 1s 299us/step
```

[21]:
```
score
```

[21]:
```
[0.41655886096877154, 0.8954869508743286]
```

- With a simple 2 layer architecture we got approx. 89.54% accuracy and a loss of 0.4165
- We can further imporve the performace with Hyperparameter tuning

1. Assignment

1.1 Update LSTM Layer

[23]:
```
# update LSTM layers
n_hidden = 64
```

- Defining the Architecture of LSTM

[24]:
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 64)                18944
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 390
=================================================================
Total params: 19,334
Trainable params: 19,334
Non-trainable params: 0
_________________________________________________________________
```

[25]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[26]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 24s 3ms/step - loss: 1.2560 -
accuracy: 0.4470 - val_loss: 1.1326 - val_accuracy: 0.4825
Epoch 2/30
```

```
7352/7352 [==============================] - 26s 4ms/step - loss: 0.9495 -
accuracy: 0.6081 - val_loss: 0.8467 - val_accuracy: 0.6912
Epoch 3/30
7352/7352 [==============================] - 27s 4ms/step - loss: 0.8132 -
accuracy: 0.6473 - val_loss: 0.8576 - val_accuracy: 0.6322
Epoch 4/30
7352/7352 [==============================] - 21s 3ms/step - loss: 0.7342 -
accuracy: 0.6933 - val_loss: 0.7157 - val_accuracy: 0.7458
Epoch 5/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.5382 -
accuracy: 0.7784 - val_loss: 0.5688 - val_accuracy: 0.7984
Epoch 6/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.3811 -
accuracy: 0.8579 - val_loss: 0.5755 - val_accuracy: 0.8480
Epoch 7/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.3327 -
accuracy: 0.8900 - val_loss: 0.5255 - val_accuracy: 0.8487
Epoch 8/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2582 -
accuracy: 0.9129 - val_loss: 0.4217 - val_accuracy: 0.8873
Epoch 9/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2411 -
accuracy: 0.9207 - val_loss: 0.3309 - val_accuracy: 0.9046
Epoch 10/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1987 -
accuracy: 0.9325 - val_loss: 0.4274 - val_accuracy: 0.8711
Epoch 11/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.1965 -
accuracy: 0.9334 - val_loss: 0.3487 - val_accuracy: 0.8731
Epoch 12/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1784 -
accuracy: 0.9357 - val_loss: 0.3973 - val_accuracy: 0.8951
Epoch 13/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.1763 -
accuracy: 0.9391 - val_loss: 0.3849 - val_accuracy: 0.8979
Epoch 14/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1544 -
accuracy: 0.9431 - val_loss: 0.4401 - val_accuracy: 0.9033
Epoch 15/30
7352/7352 [==============================] - 27s 4ms/step - loss: 0.1778 -
accuracy: 0.9421 - val_loss: 0.2845 - val_accuracy: 0.9131
Epoch 16/30
7352/7352 [==============================] - 26s 3ms/step - loss: 0.1749 -
accuracy: 0.9392 - val_loss: 0.3084 - val_accuracy: 0.9087
Epoch 17/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.1582 -
accuracy: 0.9436 - val_loss: 0.3808 - val_accuracy: 0.9108
Epoch 18/30
```

```
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1480 -
accuracy: 0.9487 - val_loss: 0.3502 - val_accuracy: 0.9138
Epoch 19/30
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1504 -
accuracy: 0.9484 - val_loss: 0.4116 - val_accuracy: 0.9036
Epoch 20/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.1431 -
accuracy: 0.9494 - val_loss: 0.2973 - val_accuracy: 0.9050
Epoch 21/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1542 -
accuracy: 0.9474 - val_loss: 0.3730 - val_accuracy: 0.9145
Epoch 22/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.1414 -
accuracy: 0.9502 - val_loss: 0.3996 - val_accuracy: 0.9158
Epoch 23/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1364 -
accuracy: 0.9497 - val_loss: 0.3808 - val_accuracy: 0.9097
Epoch 24/30
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1403 -
accuracy: 0.9512 - val_loss: 0.3501 - val_accuracy: 0.9036
Epoch 25/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.1381 -
accuracy: 0.9478 - val_loss: 0.4506 - val_accuracy: 0.9074
Epoch 26/30
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1473 -
accuracy: 0.9491 - val_loss: 0.4966 - val_accuracy: 0.9006
Epoch 27/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.1539 -
accuracy: 0.9499 - val_loss: 0.3719 - val_accuracy: 0.9060
Epoch 28/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1417 -
accuracy: 0.9508 - val_loss: 0.3556 - val_accuracy: 0.9057
Epoch 29/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1339 -
accuracy: 0.9512 - val_loss: 0.4193 - val_accuracy: 0.8928
Epoch 30/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.1781 -
accuracy: 0.9480 - val_loss: 0.3534 - val_accuracy: 0.9148
```

[26]:
```
<keras.callbacks.callbacks.History at 0x62f5f3518>
```

[27]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 509        0        27        0                   0
SITTING                  0      365       126        0                   0
```

```
STANDING                0       47       485        0                   0
WALKING                 0        0         0      459                  25
WALKING_DOWNSTAIRS      0        0         0        0                 415
WALKING_UPSTAIRS        0        0         0        4                   4

Pred                WALKING_UPSTAIRS
True
LAYING                             1
SITTING                            0
STANDING                           0
WALKING                           12
WALKING_DOWNSTAIRS                 5
WALKING_UPSTAIRS                 463
```

[28]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 1s 411us/step
```

[30]:
```
score
```

[30]:
```
[0.35375094639873494, 0.9148286581039429]
```

- Updating LSTM layer from 32 to 64 increase the performance from 89.54% to 91.48% and reduce loss from 0.4165 to 0.3537

1.2 Update Dropout Rate
1.2.1 Use 64 LSTM Layer and Dropout rate 0.7

[31]:
```
# update LSTM layers
n_hidden = 64
```

- Defining the Architecture of LSTM

[32]:
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.7))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
WARNING:tensorflow:Large dropout rate: 0.7 (>0.5). In TensorFlow 2.x, dropout()
uses dropout rate instead of keep_prob. Please ensure that this is intended.
Model: "sequential_3"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 64)                18944
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 390
=================================================================
Total params: 19,334
Trainable params: 19,334
Non-trainable params: 0
_________________________________________________________________
```

[33]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[34]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 21s 3ms/step - loss: 1.2668 -
accuracy: 0.4523 - val_loss: 1.0892 - val_accuracy: 0.5199
Epoch 2/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.9252 -
accuracy: 0.5966 - val_loss: 0.8212 - val_accuracy: 0.6244
Epoch 3/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.7997 -
accuracy: 0.6314 - val_loss: 0.8336 - val_accuracy: 0.6179
Epoch 4/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.7036 -
accuracy: 0.6722 - val_loss: 0.7824 - val_accuracy: 0.6216
Epoch 5/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.6839 -
accuracy: 0.7152 - val_loss: 0.6875 - val_accuracy: 0.7248
Epoch 6/30
7352/7352 [==============================] - 21s 3ms/step - loss: 0.5654 -
accuracy: 0.8130 - val_loss: 0.5180 - val_accuracy: 0.8273
Epoch 7/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.4122 -
accuracy: 0.8833 - val_loss: 0.6306 - val_accuracy: 0.8344
Epoch 8/30
```

```
7352/7352 [==============================] - 17s 2ms/step - loss: 0.4030 -
accuracy: 0.8911 - val_loss: 1.1381 - val_accuracy: 0.7255
Epoch 9/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.3243 -
accuracy: 0.9074 - val_loss: 0.3398 - val_accuracy: 0.8979
Epoch 10/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.2563 -
accuracy: 0.9255 - val_loss: 0.3548 - val_accuracy: 0.8914
Epoch 11/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.3041 -
accuracy: 0.9191 - val_loss: 0.3749 - val_accuracy: 0.8867
Epoch 12/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.2504 -
accuracy: 0.9257 - val_loss: 0.4726 - val_accuracy: 0.8761
Epoch 13/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.2604 -
accuracy: 0.9285 - val_loss: 0.5156 - val_accuracy: 0.8663
Epoch 14/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.2206 -
accuracy: 0.9331 - val_loss: 0.5957 - val_accuracy: 0.8612
Epoch 15/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.2023 -
accuracy: 0.9324 - val_loss: 0.3542 - val_accuracy: 0.8877
Epoch 16/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.1955 -
accuracy: 0.9385 - val_loss: 0.4524 - val_accuracy: 0.8836
Epoch 17/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.2068 -
accuracy: 0.9372 - val_loss: 0.3065 - val_accuracy: 0.9016
Epoch 18/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.2338 -
accuracy: 0.9358 - val_loss: 0.5638 - val_accuracy: 0.8860
Epoch 19/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.2166 -
accuracy: 0.9362 - val_loss: 0.4288 - val_accuracy: 0.8863
Epoch 20/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.1724 -
accuracy: 0.9431 - val_loss: 0.3221 - val_accuracy: 0.8985
Epoch 21/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.1826 -
accuracy: 0.9438 - val_loss: 0.3623 - val_accuracy: 0.9043
Epoch 22/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.1771 -
accuracy: 0.9430 - val_loss: 0.5044 - val_accuracy: 0.8931
Epoch 23/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.1904 -
accuracy: 0.9426 - val_loss: 0.3908 - val_accuracy: 0.8951
Epoch 24/30
```

```
7352/7352 [==============================] - 16s 2ms/step - loss: 0.1789 -
accuracy: 0.9426 - val_loss: 0.4822 - val_accuracy: 0.9118
Epoch 25/30
7352/7352 [==============================] - 16s 2ms/step - loss: nan -
accuracy: 0.8251 - val_loss: nan - val_accuracy: 0.1683
Epoch 26/30
7352/7352 [==============================] - 16s 2ms/step - loss: nan -
accuracy: 0.1668 - val_loss: nan - val_accuracy: 0.1683
Epoch 27/30
7352/7352 [==============================] - 16s 2ms/step - loss: nan -
accuracy: 0.1668 - val_loss: nan - val_accuracy: 0.1683
Epoch 28/30
7352/7352 [==============================] - 16s 2ms/step - loss: nan -
accuracy: 0.1668 - val_loss: nan - val_accuracy: 0.1683
Epoch 29/30
7352/7352 [==============================] - 16s 2ms/step - loss: nan -
accuracy: 0.1668 - val_loss: nan - val_accuracy: 0.1683
Epoch 30/30
7352/7352 [==============================] - 16s 2ms/step - loss: nan -
accuracy: 0.1668 - val_loss: nan - val_accuracy: 0.1683
```

[34]:
```
<keras.callbacks.callbacks.History at 0x6300a6438>
```

[35]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                WALKING
True
LAYING                  537
SITTING                 491
STANDING                532
WALKING                 496
WALKING_DOWNSTAIRS      420
WALKING_UPSTAIRS        471
```

[36]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 1s 286us/step
```

[37]:
```
score
```

[37]:
```
[nan, 0.16830675303936005]
```

- With dropout 0.7 performance reduced drastically

1.2.2 Use 32 LSTM Layer and Dropout rate 0.7

[38]:
```
# update LSTM layers
n_hidden = 32
```

- Defining the Architecture of LSTM

[39]:
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.7))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
WARNING:tensorflow:Large dropout rate: 0.7 (>0.5). In TensorFlow 2.x, dropout()
uses dropout rate instead of keep_prob. Please ensure that this is intended.
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_4 (LSTM)                (None, 32)                5376
_________________________________________________________________
dropout_4 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 198
=================================================================
Total params: 5,574
Trainable params: 5,574
Non-trainable params: 0
_________________________________________________________________
```

[40]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[41]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 15s 2ms/step - loss: 1.4131 -
```

```
accuracy: 0.4053 - val_loss: 1.2057 - val_accuracy: 0.5199
Epoch 2/30
7352/7352 [==============================] - 16s 2ms/step - loss: 1.1514 -
accuracy: 0.5004 - val_loss: 1.0854 - val_accuracy: 0.5212
Epoch 3/30
7352/7352 [==============================] - 14s 2ms/step - loss: 1.0306 -
accuracy: 0.5483 - val_loss: 0.9328 - val_accuracy: 0.6186
Epoch 4/30
7352/7352 [==============================] - 14s 2ms/step - loss: 0.8413 -
accuracy: 0.6328 - val_loss: 0.8098 - val_accuracy: 0.6403
Epoch 5/30
7352/7352 [==============================] - 14s 2ms/step - loss: 0.7508 -
accuracy: 0.6718 - val_loss: 0.7237 - val_accuracy: 0.6481
Epoch 6/30
7352/7352 [==============================] - 14s 2ms/step - loss: 0.7021 -
accuracy: 0.7036 - val_loss: 0.7130 - val_accuracy: 0.6977
Epoch 7/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.6250 -
accuracy: 0.7408 - val_loss: 0.6587 - val_accuracy: 0.7218
Epoch 8/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.6306 -
accuracy: 0.7467 - val_loss: 0.9023 - val_accuracy: 0.6749
Epoch 9/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.5728 -
accuracy: 0.7552 - val_loss: 0.6066 - val_accuracy: 0.7370
Epoch 10/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.5665 -
accuracy: 0.7582 - val_loss: 0.6308 - val_accuracy: 0.7122
Epoch 11/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.5529 -
accuracy: 0.7632 - val_loss: 0.8747 - val_accuracy: 0.7129
Epoch 12/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.5008 -
accuracy: 0.7757 - val_loss: 0.5392 - val_accuracy: 0.7513
Epoch 13/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.5068 -
accuracy: 0.7767 - val_loss: 0.5349 - val_accuracy: 0.7448
Epoch 14/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.4965 -
accuracy: 0.7792 - val_loss: 0.7161 - val_accuracy: 0.7313
Epoch 15/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.6753 -
accuracy: 0.7588 - val_loss: 0.5898 - val_accuracy: 0.7448
Epoch 16/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.5031 -
accuracy: 0.7777 - val_loss: 0.7619 - val_accuracy: 0.7245
Epoch 17/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.4484 -
```

```
accuracy: 0.7890 - val_loss: 0.6495 - val_accuracy: 0.7336
Epoch 18/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.4806 -
accuracy: 0.7837 - val_loss: 0.5810 - val_accuracy: 0.7496
Epoch 19/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.4920 -
accuracy: 0.7875 - val_loss: 0.5989 - val_accuracy: 0.7855
Epoch 20/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.4532 -
accuracy: 0.8157 - val_loss: 0.5946 - val_accuracy: 0.8049
Epoch 21/30
7352/7352 [==============================] - 13s 2ms/step - loss: 0.4626 -
accuracy: 0.8255 - val_loss: 0.4647 - val_accuracy: 0.8317
Epoch 22/30
7352/7352 [==============================] - 13s 2ms/step - loss: 0.4284 -
accuracy: 0.8477 - val_loss: 0.4966 - val_accuracy: 0.8616
Epoch 23/30
7352/7352 [==============================] - 13s 2ms/step - loss: 0.4184 -
accuracy: 0.8561 - val_loss: 0.4365 - val_accuracy: 0.8636
Epoch 24/30
7352/7352 [==============================] - 13s 2ms/step - loss: 0.3954 -
accuracy: 0.8677 - val_loss: 0.4070 - val_accuracy: 0.8778
Epoch 25/30
7352/7352 [==============================] - 13s 2ms/step - loss: 0.3787 -
accuracy: 0.8750 - val_loss: 0.4029 - val_accuracy: 0.8761
Epoch 26/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.4372 -
accuracy: 0.8668 - val_loss: 0.4681 - val_accuracy: 0.8619
Epoch 27/30
7352/7352 [==============================] - 14s 2ms/step - loss: 0.4192 -
accuracy: 0.8783 - val_loss: 0.4248 - val_accuracy: 0.8833
Epoch 28/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.4191 -
accuracy: 0.8732 - val_loss: 1.1744 - val_accuracy: 0.7923
Epoch 29/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.3634 -
accuracy: 0.8832 - val_loss: 0.4041 - val_accuracy: 0.8907
Epoch 30/30
7352/7352 [==============================] - 14s 2ms/step - loss: 0.3713 -
accuracy: 0.8783 - val_loss: 0.5968 - val_accuracy: 0.8381
```

[41]:
```
<keras.callbacks.callbacks.History at 0x6309d69e8>
```

[42]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
```

```
LAYING                510        0         0        0                 0
SITTING                 0      280       206        1                 1
STANDING                0       34       489        4                 2
WALKING                 0        0         0      360                33
WALKING_DOWNSTAIRS      0        0         0        3               416
WALKING_UPSTAIRS        1        3         0       28                24

Pred                WALKING_UPSTAIRS
True
LAYING                            27
SITTING                            3
STANDING                           3
WALKING                          103
WALKING_DOWNSTAIRS                 1
WALKING_UPSTAIRS                 415
```

[43]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 1s 243us/step
```

[44]:
```
score
```

[44]:
```
[0.596762544016313, 0.8381404876708984]
```

- Better than 64 layer with same dropout 0.7

1.2.3 Use 64 LSTM Layer and Dropout rate 0.6

[45]:
```
# update LSTM layers
n_hidden = 64
```

- Defining the Architecture of LSTM

[46]:
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.6))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
WARNING:tensorflow:Large dropout rate: 0.6 (>0.5). In TensorFlow 2.x, dropout()
uses dropout rate instead of keep_prob. Please ensure that this is intended.
Model: "sequential_5"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_5 (LSTM)                (None, 64)                18944
_________________________________________________________________
dropout_5 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_5 (Dense)              (None, 6)                 390
=================================================================
Total params: 19,334
Trainable params: 19,334
Non-trainable params: 0
_________________________________________________________________
```

[47]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[48]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 19s 3ms/step - loss: 1.2606 -
accuracy: 0.4646 - val_loss: 1.0564 - val_accuracy: 0.5443
Epoch 2/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.8792 -
accuracy: 0.6002 - val_loss: 0.8393 - val_accuracy: 0.6179
Epoch 3/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.7371 -
accuracy: 0.6447 - val_loss: 0.7495 - val_accuracy: 0.6183
Epoch 4/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.6515 -
accuracy: 0.7088 - val_loss: 0.6700 - val_accuracy: 0.7048
Epoch 5/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.4806 -
accuracy: 0.8392 - val_loss: 0.5685 - val_accuracy: 0.8073
Epoch 6/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.2974 -
accuracy: 0.9057 - val_loss: 0.4664 - val_accuracy: 0.8660
Epoch 7/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.2478 -
accuracy: 0.9236 - val_loss: 0.4457 - val_accuracy: 0.8738
Epoch 8/30
```

```
7352/7352 [==============================] - 18s 2ms/step - loss: 0.2243 -
accuracy: 0.9282 - val_loss: 0.2996 - val_accuracy: 0.8863
Epoch 9/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.2131 -
accuracy: 0.9249 - val_loss: 0.5138 - val_accuracy: 0.8792
Epoch 10/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.2012 -
accuracy: 0.9329 - val_loss: 0.2673 - val_accuracy: 0.8979
Epoch 11/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1872 -
accuracy: 0.9353 - val_loss: 0.2656 - val_accuracy: 0.9101
Epoch 12/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1965 -
accuracy: 0.9366 - val_loss: 0.3882 - val_accuracy: 0.9080
Epoch 13/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1710 -
accuracy: 0.9404 - val_loss: 0.2497 - val_accuracy: 0.9091
Epoch 14/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1621 -
accuracy: 0.9408 - val_loss: 0.3737 - val_accuracy: 0.8921
Epoch 15/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1517 -
accuracy: 0.9456 - val_loss: 0.2366 - val_accuracy: 0.9165
Epoch 16/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1870 -
accuracy: 0.9423 - val_loss: 0.2972 - val_accuracy: 0.9036
Epoch 17/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1612 -
accuracy: 0.9418 - val_loss: 0.4051 - val_accuracy: 0.8890
Epoch 18/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1619 -
accuracy: 0.9434 - val_loss: 0.3921 - val_accuracy: 0.8941
Epoch 19/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1394 -
accuracy: 0.9493 - val_loss: 0.5242 - val_accuracy: 0.8985
Epoch 20/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.1573 -
accuracy: 0.9467 - val_loss: 0.3529 - val_accuracy: 0.9097
Epoch 21/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1615 -
accuracy: 0.9499 - val_loss: 0.7128 - val_accuracy: 0.8789
Epoch 22/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1429 -
accuracy: 0.9518 - val_loss: 0.3295 - val_accuracy: 0.9223
Epoch 23/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1548 -
accuracy: 0.9479 - val_loss: 0.5361 - val_accuracy: 0.9063
Epoch 24/30
```

```
7352/7352 [==============================] - 20s 3ms/step - loss: 0.1543 -
accuracy: 0.9467 - val_loss: 0.2969 - val_accuracy: 0.9016
Epoch 25/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1431 -
accuracy: 0.9471 - val_loss: 0.5244 - val_accuracy: 0.9050
Epoch 26/30
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1457 -
accuracy: 0.9499 - val_loss: 0.3353 - val_accuracy: 0.9186
Epoch 27/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1532 -
accuracy: 0.9510 - val_loss: 0.5023 - val_accuracy: 0.9074
Epoch 28/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.1357 -
accuracy: 0.9514 - val_loss: 0.2958 - val_accuracy: 0.9152
Epoch 29/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.1368 -
accuracy: 0.9490 - val_loss: 0.3224 - val_accuracy: 0.9189
Epoch 30/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.1569 -
accuracy: 0.9467 - val_loss: 0.4957 - val_accuracy: 0.9060
```

[48]:
```
<keras.callbacks.callbacks.History at 0x631129d68>
```

[49]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 517        0        19        0                   0
SITTING                  0      416        74        0                   0
STANDING                 0       94       438        0                   0
WALKING                  0        0         0      495                   0
WALKING_DOWNSTAIRS       0        0         0       15                 377
WALKING_UPSTAIRS         0        0         0       41                   3

Pred                WALKING_UPSTAIRS
True
LAYING                             1
SITTING                            1
STANDING                           0
WALKING                            1
WALKING_DOWNSTAIRS                28
WALKING_UPSTAIRS                 427
```

[50]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 1s 283us/step
```

[51]:
```
score
```

[51]:
```
[0.4956902541945778, 0.9060060977935791]
```

- Better than 0.7 dropout

1.2.4 Use 32 LSTM Layer and Dropout rate 0.6

[52]:
```
# update LSTM layers
n_hidden = 32
```

- Defining the Architecture of LSTM

[53]:
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.6))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
WARNING:tensorflow:Large dropout rate: 0.6 (>0.5). In TensorFlow 2.x, dropout()
uses dropout rate instead of keep_prob. Please ensure that this is intended.
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_6 (LSTM)                (None, 32)                5376
_________________________________________________________________
dropout_6 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_6 (Dense)              (None, 6)                 198
=================================================================
Total params: 5,574
Trainable params: 5,574
Non-trainable params: 0
_________________________________________________________________
```

[54]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[55]:
```
# Training the model
model.fit(X_train,
          Y_train,
```

```
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 17s 2ms/step - loss: 1.3738 -
accuracy: 0.3864 - val_loss: 1.3047 - val_accuracy: 0.4381
Epoch 2/30
7352/7352 [==============================] - 15s 2ms/step - loss: 1.0837 -
accuracy: 0.5292 - val_loss: 1.0199 - val_accuracy: 0.5565
Epoch 3/30
7352/7352 [==============================] - 14s 2ms/step - loss: 0.8871 -
accuracy: 0.6268 - val_loss: 0.8298 - val_accuracy: 0.6121
Epoch 4/30
7352/7352 [==============================] - 14s 2ms/step - loss: 0.7479 -
accuracy: 0.6549 - val_loss: 0.7888 - val_accuracy: 0.6149
Epoch 5/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.7072 -
accuracy: 0.6639 - val_loss: 0.8324 - val_accuracy: 0.6020
Epoch 6/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.6625 -
accuracy: 0.6814 - val_loss: 0.8694 - val_accuracy: 0.5979
Epoch 7/30
7352/7352 [==============================] - 14s 2ms/step - loss: 0.6415 -
accuracy: 0.7081 - val_loss: 0.7401 - val_accuracy: 0.6614
Epoch 8/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.5871 -
accuracy: 0.7678 - val_loss: 0.6918 - val_accuracy: 0.7754
Epoch 9/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.4870 -
accuracy: 0.8341 - val_loss: 0.7611 - val_accuracy: 0.7540
Epoch 10/30
7352/7352 [==============================] - 14s 2ms/step - loss: 0.4380 -
accuracy: 0.8558 - val_loss: 0.6525 - val_accuracy: 0.8022
Epoch 11/30
7352/7352 [==============================] - 14s 2ms/step - loss: 0.4693 -
accuracy: 0.8546 - val_loss: 0.5199 - val_accuracy: 0.8636
Epoch 12/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.3959 -
accuracy: 0.8878 - val_loss: 0.6564 - val_accuracy: 0.8290
Epoch 13/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.3703 -
accuracy: 0.8953 - val_loss: 0.5741 - val_accuracy: 0.8324
Epoch 14/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.3400 -
accuracy: 0.9029 - val_loss: 0.5634 - val_accuracy: 0.8497
```

```
Epoch 15/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.3523 -
accuracy: 0.8976 - val_loss: 0.6537 - val_accuracy: 0.8449
Epoch 16/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.3631 -
accuracy: 0.8946 - val_loss: 0.4173 - val_accuracy: 0.8789
Epoch 17/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.3099 -
accuracy: 0.9123 - val_loss: 0.6522 - val_accuracy: 0.8490
Epoch 18/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.3181 -
accuracy: 0.9066 - val_loss: 0.4997 - val_accuracy: 0.8748
Epoch 19/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.2624 -
accuracy: 0.9219 - val_loss: 0.5491 - val_accuracy: 0.8700
Epoch 20/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.2779 -
accuracy: 0.9229 - val_loss: 0.5618 - val_accuracy: 0.8850
Epoch 21/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.2645 -
accuracy: 0.9272 - val_loss: 0.4384 - val_accuracy: 0.8941
Epoch 22/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.3439 -
accuracy: 0.9105 - val_loss: 0.4841 - val_accuracy: 0.8789
Epoch 23/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.2935 -
accuracy: 0.9139 - val_loss: 0.5743 - val_accuracy: 0.8870
Epoch 24/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.2599 -
accuracy: 0.9293 - val_loss: 0.4404 - val_accuracy: 0.8945
Epoch 25/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.2976 -
accuracy: 0.9149 - val_loss: 0.4465 - val_accuracy: 0.8836
Epoch 26/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.2633 -
accuracy: 0.9237 - val_loss: 0.4992 - val_accuracy: 0.8924
Epoch 27/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.2649 -
accuracy: 0.9261 - val_loss: 0.5707 - val_accuracy: 0.8711
Epoch 28/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.2830 -
accuracy: 0.9229 - val_loss: 0.4471 - val_accuracy: 0.8887
Epoch 29/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.2589 -
accuracy: 0.9263 - val_loss: 0.4967 - val_accuracy: 0.8843
Epoch 30/30
7352/7352 [==============================] - 15s 2ms/step - loss: 0.2628 -
accuracy: 0.9279 - val_loss: 0.5602 - val_accuracy: 0.8812
```

[55]:
```
<keras.callbacks.callbacks.History at 0x633723b70>
```

[56]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 510        0        24        0                   0
SITTING                  3      434        29        7                   0
STANDING                 0      135       379        1                   0
WALKING                  0        1         0      409                  20
WALKING_DOWNSTAIRS       0        0         0        7                 399
WALKING_UPSTAIRS         0        0         0        5                   0

Pred                WALKING_UPSTAIRS
True
LAYING                             3
SITTING                           18
STANDING                          17
WALKING                           66
WALKING_DOWNSTAIRS                14
WALKING_UPSTAIRS                 466
```

[57]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 1s 256us/step
```

[58]:
```
score
```

[58]:
```
[0.5602035771178105, 0.8812351822853088]
```

- It's perform worse than 64 layer with dropout 0.6

1.3 Use 2 LSTM Layers with Larger Dropout
1.3.1 LSTM Layer (64,32) with 2 layer of dropout 0.7

[95]:
```
# update LSTM layers
n_hidden_1 = 64
n_hidden_2 = 32
```

- Initialize the LSTM Architecture

[96]:
```
# https://stackoverflow.com/questions/51763983/
 ↪error-when-checking-target-expected-dense-1-to-have-3-dimensions-but-got-array
# https://github.com/keras-team/keras/issues/7403
# 1. You need to set return_sequences=True from first LSTM
# 2. You need to set return_sequences=True from second LSTM
# Initiliazing the sequential model
```

```
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden_1, input_shape=(timesteps, input_dim),␣
 ↪return_sequences=True))
# Adding a dropout layer
model.add(Dropout(0.7))
# Configuring the parameters
model.add(LSTM(n_hidden_2 , return_sequences=False))
# Adding a dropout layer
model.add(Dropout(0.7))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_28"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_46 (LSTM)               (None, 128, 64)           18944
_________________________________________________________________
dropout_27 (Dropout)         (None, 128, 64)           0
_________________________________________________________________
lstm_47 (LSTM)               (None, 32)                12416
_________________________________________________________________
dropout_28 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_12 (Dense)             (None, 6)                 198
=================================================================
Total params: 31,558
Trainable params: 31,558
Non-trainable params: 0
_________________________________________________________________
```

[97]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[98]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
```

```
7352/7352 [==============================] - 85s 12ms/step - loss: 1.2737 -
accuracy: 0.4884 - val_loss: 0.9074 - val_accuracy: 0.5938
Epoch 2/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.8894 -
accuracy: 0.6167 - val_loss: 0.7144 - val_accuracy: 0.6206
Epoch 3/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.7967 -
accuracy: 0.6294 - val_loss: 0.7580 - val_accuracy: 0.6237
Epoch 4/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.7435 -
accuracy: 0.6506 - val_loss: 0.7531 - val_accuracy: 0.6216
Epoch 5/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.7146 -
accuracy: 0.6518 - val_loss: 0.7442 - val_accuracy: 0.6315
Epoch 6/30
7352/7352 [==============================] - 84s 11ms/step - loss: 0.7182 -
accuracy: 0.6578 - val_loss: 0.7171 - val_accuracy: 0.6328
Epoch 7/30
7352/7352 [==============================] - 86s 12ms/step - loss: 0.7103 -
accuracy: 0.6644 - val_loss: 0.8312 - val_accuracy: 0.6189
Epoch 8/30
7352/7352 [==============================] - 83s 11ms/step - loss: 0.6082 -
accuracy: 0.7271 - val_loss: 0.5815 - val_accuracy: 0.7472
Epoch 9/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.5212 -
accuracy: 0.7779 - val_loss: 0.6235 - val_accuracy: 0.7482
Epoch 10/30
7352/7352 [==============================] - 84s 11ms/step - loss: 0.4809 -
accuracy: 0.7831 - val_loss: 0.4557 - val_accuracy: 0.7560
Epoch 11/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.4502 -
accuracy: 0.7949 - val_loss: 0.5220 - val_accuracy: 0.7486
Epoch 12/30
7352/7352 [==============================] - 84s 11ms/step - loss: 0.4417 -
accuracy: 0.8214 - val_loss: 0.4750 - val_accuracy: 0.8195
Epoch 13/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.3882 -
accuracy: 0.8807 - val_loss: 0.6525 - val_accuracy: 0.8300
Epoch 14/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.3461 -
accuracy: 0.9051 - val_loss: 0.3229 - val_accuracy: 0.8856
Epoch 15/30
7352/7352 [==============================] - 93s 13ms/step - loss: 0.2932 -
accuracy: 0.9212 - val_loss: 0.3832 - val_accuracy: 0.8873
Epoch 16/30
7352/7352 [==============================] - 86s 12ms/step - loss: 0.2834 -
accuracy: 0.9229 - val_loss: 0.4894 - val_accuracy: 0.8565
Epoch 17/30
```

```
7352/7352 [==============================] - 87s 12ms/step - loss: 0.2440 -
accuracy: 0.9328 - val_loss: 0.3598 - val_accuracy: 0.9040
Epoch 18/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.2287 -
accuracy: 0.9357 - val_loss: 0.2449 - val_accuracy: 0.9070
Epoch 19/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.2432 -
accuracy: 0.9314 - val_loss: 0.2899 - val_accuracy: 0.9199
Epoch 20/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.2307 -
accuracy: 0.9343 - val_loss: 0.3551 - val_accuracy: 0.8948
Epoch 21/30
7352/7352 [==============================] - 117s 16ms/step - loss: 0.2071 -
accuracy: 0.9377 - val_loss: 0.3508 - val_accuracy: 0.9145
Epoch 22/30
7352/7352 [==============================] - 106s 14ms/step - loss: 0.2145 -
accuracy: 0.9353 - val_loss: 0.7906 - val_accuracy: 0.8666
Epoch 23/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.2291 -
accuracy: 0.9347 - val_loss: 0.3717 - val_accuracy: 0.9080
Epoch 24/30
7352/7352 [==============================] - 97s 13ms/step - loss: 0.1955 -
accuracy: 0.9402 - val_loss: 0.3518 - val_accuracy: 0.9182
Epoch 25/30
7352/7352 [==============================] - 77s 10ms/step - loss: 0.2144 -
accuracy: 0.9388 - val_loss: 0.4186 - val_accuracy: 0.9155
Epoch 26/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.2106 -
accuracy: 0.9389 - val_loss: 0.2988 - val_accuracy: 0.9165
Epoch 27/30
7352/7352 [==============================] - 79s 11ms/step - loss: 0.1955 -
accuracy: 0.9395 - val_loss: 0.3549 - val_accuracy: 0.9046
Epoch 28/30
7352/7352 [==============================] - 75s 10ms/step - loss: nan -
accuracy: 0.3413 - val_loss: nan - val_accuracy: 0.1683
Epoch 29/30
7352/7352 [==============================] - 76s 10ms/step - loss: nan -
accuracy: 0.1668 - val_loss: nan - val_accuracy: 0.1683
Epoch 30/30
7352/7352 [==============================] - 78s 11ms/step - loss: nan -
accuracy: 0.1668 - val_loss: nan - val_accuracy: 0.1683
```

[98]:
```
<keras.callbacks.callbacks.History at 0x63c6b10b8>
```

[99]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                WALKING
```

```
True
LAYING                  537
SITTING                 491
STANDING                532
WALKING                 496
WALKING_DOWNSTAIRS      420
WALKING_UPSTAIRS        471
```

[100]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 4s 1ms/step
```

[101]:
```
score
```

[101]:
```
[nan, 0.16830675303936005]
```

- It's performance is worst

1.3.2 LSTM Layer (64,32) with 2 layer dropout 0.6

[102]:
```
# update LSTM layers
n_hidden_1 = 64
n_hidden_2 = 32
```

- Initialize the LSTM Architecture

[103]:
```
# https://stackoverflow.com/questions/51763983/
 ↪error-when-checking-target-expected-dense-1-to-have-3-dimensions-but-got-array
# https://github.com/keras-team/keras/issues/7403
# 1. You need to set return_sequences=True from first LSTM
# 2. You need to set return_sequences=True from second LSTM
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden_1, input_shape=(timesteps, input_dim),␣
 ↪return_sequences=True))
# Adding a dropout layer
model.add(Dropout(0.6))
# Configuring the parameters
model.add(LSTM(n_hidden_2 , return_sequences=False))
# Adding a dropout layer
model.add(Dropout(0.6))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_29"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_48 (LSTM)               (None, 128, 64)           18944
_________________________________________________________________
dropout_29 (Dropout)         (None, 128, 64)           0
_________________________________________________________________
lstm_49 (LSTM)               (None, 32)                12416
_________________________________________________________________
dropout_30 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_13 (Dense)             (None, 6)                 198
=================================================================
Total params: 31,558
Trainable params: 31,558
Non-trainable params: 0
_________________________________________________________________
```

[104]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[105]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 78s 11ms/step - loss: 1.2568 -
accuracy: 0.4778 - val_loss: 1.2977 - val_accuracy: 0.4462
Epoch 2/30
7352/7352 [==============================] - 77s 10ms/step - loss: 0.8864 -
accuracy: 0.6138 - val_loss: 0.8237 - val_accuracy: 0.6179
Epoch 3/30
7352/7352 [==============================] - 79s 11ms/step - loss: 0.7344 -
accuracy: 0.6719 - val_loss: 0.7058 - val_accuracy: 0.6804
Epoch 4/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.6797 -
accuracy: 0.7223 - val_loss: 0.6778 - val_accuracy: 0.7411
Epoch 5/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.5348 -
accuracy: 0.8092 - val_loss: 0.4354 - val_accuracy: 0.8537
Epoch 6/30
7352/7352 [==============================] - 79s 11ms/step - loss: 0.3872 -
accuracy: 0.8938 - val_loss: 0.4438 - val_accuracy: 0.8799
```

```
Epoch 7/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.3565 -
accuracy: 0.9079 - val_loss: 0.4188 - val_accuracy: 0.8823
Epoch 8/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.2932 -
accuracy: 0.9203 - val_loss: 0.4461 - val_accuracy: 0.8850
Epoch 9/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.2624 -
accuracy: 0.9271 - val_loss: 0.4734 - val_accuracy: 0.8894
Epoch 10/30
7352/7352 [==============================] - 77s 11ms/step - loss: 0.2479 -
accuracy: 0.9272 - val_loss: 0.3673 - val_accuracy: 0.9023
Epoch 11/30
7352/7352 [==============================] - 77s 10ms/step - loss: 0.2425 -
accuracy: 0.9328 - val_loss: 0.4653 - val_accuracy: 0.8839
Epoch 12/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.2103 -
accuracy: 0.9389 - val_loss: 0.3446 - val_accuracy: 0.9030
Epoch 13/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.1988 -
accuracy: 0.9399 - val_loss: 0.3912 - val_accuracy: 0.9148
Epoch 14/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.1747 -
accuracy: 0.9414 - val_loss: 0.5197 - val_accuracy: 0.8921
Epoch 15/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.1786 -
accuracy: 0.9455 - val_loss: 0.6204 - val_accuracy: 0.8945
Epoch 16/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.1747 -
accuracy: 0.9426 - val_loss: 0.3647 - val_accuracy: 0.9087
Epoch 17/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.1764 -
accuracy: 0.9414 - val_loss: 0.7391 - val_accuracy: 0.8799
Epoch 18/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.1691 -
accuracy: 0.9440 - val_loss: 0.3439 - val_accuracy: 0.9053
Epoch 19/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.1668 -
accuracy: 0.9487 - val_loss: 0.4053 - val_accuracy: 0.9040
Epoch 20/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1707 -
accuracy: 0.9430 - val_loss: 0.5533 - val_accuracy: 0.8904
Epoch 21/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1765 -
accuracy: 0.9457 - val_loss: 0.4594 - val_accuracy: 0.9077
Epoch 22/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.2275 -
accuracy: 0.9396 - val_loss: 0.4258 - val_accuracy: 0.9169
```

```
Epoch 23/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.1920 -
accuracy: 0.9434 - val_loss: 0.4475 - val_accuracy: 0.9077
Epoch 24/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1931 -
accuracy: 0.9445 - val_loss: 0.6805 - val_accuracy: 0.8799
Epoch 25/30
7352/7352 [==============================] - 77s 11ms/step - loss: 0.1535 -
accuracy: 0.9461 - val_loss: 0.4987 - val_accuracy: 0.9006
Epoch 26/30
7352/7352 [==============================] - 75s 10ms/step - loss: nan -
accuracy: 0.6254 - val_loss: nan - val_accuracy: 0.1683
Epoch 27/30
7352/7352 [==============================] - 77s 10ms/step - loss: nan -
accuracy: 0.1668 - val_loss: nan - val_accuracy: 0.1683
Epoch 28/30
7352/7352 [==============================] - 75s 10ms/step - loss: nan -
accuracy: 0.1668 - val_loss: nan - val_accuracy: 0.1683
Epoch 29/30
7352/7352 [==============================] - 79s 11ms/step - loss: nan -
accuracy: 0.1668 - val_loss: nan - val_accuracy: 0.1683
Epoch 30/30
7352/7352 [==============================] - 77s 10ms/step - loss: nan -
accuracy: 0.1668 - val_loss: nan - val_accuracy: 0.1683
```

[105]:
```
<keras.callbacks.callbacks.History at 0x63d0cecf8>
```

[106]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                WALKING
True
LAYING                  537
SITTING                 491
STANDING                532
WALKING                 496
WALKING_DOWNSTAIRS      420
WALKING_UPSTAIRS        471
```

[107]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 4s 1ms/step
```

[108]:
```
score
```

[108]:
```
[nan, 0.16830675303936005]
```

- It's performance is worst

1.3.3 LSTM Layer (64,32) with 2 layer dropout 0.5

[110]:
```
# update LSTM layers
n_hidden_1 = 64
n_hidden_2 = 32
```

[111]:
```
# https://stackoverflow.com/questions/51763983/
 ↪error-when-checking-target-expected-dense-1-to-have-3-dimensions-but-got-array
# https://github.com/keras-team/keras/issues/7403
# 1. You need to set return_sequences=True from first LSTM
# 2. You need to set return_sequences=True from second LSTM
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden_1, input_shape=(timesteps, input_dim),␣
 ↪return_sequences=True))
# Adding a dropout layer
model.add(Dropout(0.5))
# Configuring the parameters
model.add(LSTM(n_hidden_2 , return_sequences=False))
# Adding a dropout layer
model.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_30"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_50 (LSTM)               (None, 128, 64)           18944
_________________________________________________________________
dropout_31 (Dropout)         (None, 128, 64)           0
_________________________________________________________________
lstm_51 (LSTM)               (None, 32)                12416
_________________________________________________________________
dropout_32 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_14 (Dense)             (None, 6)                 198
=================================================================
Total params: 31,558
Trainable params: 31,558
Non-trainable params: 0
_________________________________________________________________
```

[112]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[113]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 96s 13ms/step - loss: 1.0873 -
accuracy: 0.5563 - val_loss: 0.7762 - val_accuracy: 0.7038
Epoch 2/30
7352/7352 [==============================] - 95s 13ms/step - loss: 0.6862 -
accuracy: 0.7337 - val_loss: 0.5855 - val_accuracy: 0.7299
Epoch 3/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.5282 -
accuracy: 0.7763 - val_loss: 0.5286 - val_accuracy: 0.7615
Epoch 4/30
7352/7352 [==============================] - 93s 13ms/step - loss: 0.4156 -
accuracy: 0.8392 - val_loss: 0.4071 - val_accuracy: 0.8819
Epoch 5/30
7352/7352 [==============================] - 81s 11ms/step - loss: 0.2848 -
accuracy: 0.9139 - val_loss: 0.4556 - val_accuracy: 0.8755
Epoch 6/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.2346 -
accuracy: 0.9327 - val_loss: 0.5155 - val_accuracy: 0.8731
Epoch 7/30
7352/7352 [==============================] - 83s 11ms/step - loss: 0.2368 -
accuracy: 0.9323 - val_loss: 0.4285 - val_accuracy: 0.8772
Epoch 8/30
7352/7352 [==============================] - 84s 11ms/step - loss: 0.2111 -
accuracy: 0.9358 - val_loss: 0.3157 - val_accuracy: 0.9057
Epoch 9/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1866 -
accuracy: 0.9404 - val_loss: 0.4269 - val_accuracy: 0.9019
Epoch 10/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1835 -
accuracy: 0.9389 - val_loss: 0.2840 - val_accuracy: 0.9108
Epoch 11/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1625 -
accuracy: 0.9470 - val_loss: 0.3311 - val_accuracy: 0.9077
Epoch 12/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1528 -
accuracy: 0.9460 - val_loss: 0.4299 - val_accuracy: 0.9145
Epoch 13/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1541 -
accuracy: 0.9434 - val_loss: 0.3474 - val_accuracy: 0.9199
```

```
Epoch 14/30
7352/7352 [==============================] - 83s 11ms/step - loss: 0.1506 -
accuracy: 0.9464 - val_loss: 0.4714 - val_accuracy: 0.8968
Epoch 15/30
7352/7352 [==============================] - 83s 11ms/step - loss: 0.1506 -
accuracy: 0.9482 - val_loss: 0.4906 - val_accuracy: 0.9104
Epoch 16/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.1481 -
accuracy: 0.9486 - val_loss: 0.5460 - val_accuracy: 0.9023
Epoch 17/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.1604 -
accuracy: 0.9490 - val_loss: 0.4505 - val_accuracy: 0.9057
Epoch 18/30
7352/7352 [==============================] - 77s 10ms/step - loss: 0.1389 -
accuracy: 0.9510 - val_loss: 0.5048 - val_accuracy: 0.9036
Epoch 19/30
7352/7352 [==============================] - 77s 11ms/step - loss: 0.1574 -
accuracy: 0.9479 - val_loss: 0.6674 - val_accuracy: 0.8880
Epoch 20/30
7352/7352 [==============================] - 77s 10ms/step - loss: 0.1591 -
accuracy: 0.9509 - val_loss: 0.5377 - val_accuracy: 0.9118
Epoch 21/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.1516 -
accuracy: 0.9474 - val_loss: 0.3849 - val_accuracy: 0.9030
Epoch 22/30
7352/7352 [==============================] - 79s 11ms/step - loss: 0.1376 -
accuracy: 0.9524 - val_loss: 0.9215 - val_accuracy: 0.8877
Epoch 23/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.1471 -
accuracy: 0.9508 - val_loss: 0.5519 - val_accuracy: 0.9074
Epoch 24/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.1971 -
accuracy: 0.9373 - val_loss: 0.6929 - val_accuracy: 0.8823
Epoch 25/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.1533 -
accuracy: 0.9482 - val_loss: 0.6994 - val_accuracy: 0.8867
Epoch 26/30
7352/7352 [==============================] - 123s 17ms/step - loss: 0.1517 -
accuracy: 0.9510 - val_loss: 0.4023 - val_accuracy: 0.8948
Epoch 27/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.1306 -
accuracy: 0.9523 - val_loss: 0.4267 - val_accuracy: 0.9036
Epoch 28/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.1561 -
accuracy: 0.9476 - val_loss: 0.5926 - val_accuracy: 0.9114
Epoch 29/30
7352/7352 [==============================] - 84s 11ms/step - loss: 0.1539 -
accuracy: 0.9452 - val_loss: 0.5725 - val_accuracy: 0.9030
```

```
Epoch 30/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.1292 -
accuracy: 0.9533 - val_loss: 0.6457 - val_accuracy: 0.9002
```

[113]:
```
<keras.callbacks.callbacks.History at 0x63f20dd30>
```

[114]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 510        0        27        0                   0
SITTING                  0      397        91        0                   0
STANDING                 0       92       440        0                   0
WALKING                  0        1         1      458                  28
WALKING_DOWNSTAIRS       0        0         0        1                 410
WALKING_UPSTAIRS         1        0         6       16                  10

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            3
STANDING                           0
WALKING                            8
WALKING_DOWNSTAIRS                 9
WALKING_UPSTAIRS                 438
```

[115]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 4s 1ms/step
```

[116]:
```
score
```

[116]:
```
[0.6456601167650766, 0.900237500667572]
```

- It's performance is better than 0.7 and 0.6 but loss is considerably high

1.3.4 LSTM Layer (64,32) with 1 layer of dropout 0.7

[117]:
```
# update LSTM layers
n_hidden_1 = 64
n_hidden_2 = 32
```

[118]:
```
# https://stackoverflow.com/questions/51763983/
 ↪error-when-checking-target-expected-dense-1-to-have-3-dimensions-but-got-array
# https://github.com/keras-team/keras/issues/7403
# 1. You need to set return_sequences=True from first LSTM
# 2. You need to set return_sequences=True from second LSTM
# Initiliazing the sequential model
```

```
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden_1, input_shape=(timesteps, input_dim),␣
 ↪return_sequences=True))
# Adding a dropout layer
# model.add(Dropout(0.5))
# Configuring the parameters
model.add(LSTM(n_hidden_2 , return_sequences=False))
# Adding a dropout layer
model.add(Dropout(0.7))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_31"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_52 (LSTM)               (None, 128, 64)           18944
_________________________________________________________________
lstm_53 (LSTM)               (None, 32)                12416
_________________________________________________________________
dropout_33 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_15 (Dense)             (None, 6)                 198
=================================================================
Total params: 31,558
Trainable params: 31,558
Non-trainable params: 0
_________________________________________________________________
```

[119]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[120]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 97s 13ms/step - loss: 1.2252 -
accuracy: 0.4883 - val_loss: 0.9514 - val_accuracy: 0.5755
```

```
Epoch 2/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.8989 -
accuracy: 0.6023 - val_loss: 0.7508 - val_accuracy: 0.6498
Epoch 3/30
7352/7352 [==============================] - 81s 11ms/step - loss: 0.7873 -
accuracy: 0.6359 - val_loss: 0.7832 - val_accuracy: 0.6159
Epoch 4/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.7230 -
accuracy: 0.6726 - val_loss: 0.8069 - val_accuracy: 0.5959
Epoch 5/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.6513 -
accuracy: 0.6876 - val_loss: 0.7456 - val_accuracy: 0.6227
Epoch 6/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.5906 -
accuracy: 0.7155 - val_loss: 0.6538 - val_accuracy: 0.7469
Epoch 7/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.5490 -
accuracy: 0.7606 - val_loss: 0.5527 - val_accuracy: 0.7784
Epoch 8/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.5179 -
accuracy: 0.7835 - val_loss: 0.5049 - val_accuracy: 0.7720
Epoch 9/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.4791 -
accuracy: 0.8198 - val_loss: 0.5555 - val_accuracy: 0.8599
Epoch 10/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.4130 -
accuracy: 0.8700 - val_loss: 0.4166 - val_accuracy: 0.8751
Epoch 11/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.3465 -
accuracy: 0.9037 - val_loss: 0.3210 - val_accuracy: 0.8965
Epoch 12/30
7352/7352 [==============================] - 77s 11ms/step - loss: 0.2978 -
accuracy: 0.9174 - val_loss: 0.3080 - val_accuracy: 0.9067
Epoch 13/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.2804 -
accuracy: 0.9202 - val_loss: 0.3070 - val_accuracy: 0.9145
Epoch 14/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.2638 -
accuracy: 0.9248 - val_loss: 0.3665 - val_accuracy: 0.9057
Epoch 15/30
7352/7352 [==============================] - 77s 10ms/step - loss: 0.2558 -
accuracy: 0.9266 - val_loss: 0.3212 - val_accuracy: 0.9125
Epoch 16/30
7352/7352 [==============================] - 87s 12ms/step - loss: 0.2393 -
accuracy: 0.9369 - val_loss: 0.3172 - val_accuracy: 0.9148
Epoch 17/30
7352/7352 [==============================] - 93s 13ms/step - loss: 0.2349 -
accuracy: 0.9370 - val_loss: 0.3730 - val_accuracy: 0.9135
```

```
Epoch 18/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.2188 -
accuracy: 0.9378 - val_loss: 0.3914 - val_accuracy: 0.9030
Epoch 19/30
7352/7352 [==============================] - 92s 13ms/step - loss: 0.2022 -
accuracy: 0.9384 - val_loss: 0.3416 - val_accuracy: 0.9274
Epoch 20/30
7352/7352 [==============================] - 81s 11ms/step - loss: 0.1937 -
accuracy: 0.9403 - val_loss: 0.3226 - val_accuracy: 0.9199
Epoch 21/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.2018 -
accuracy: 0.9391 - val_loss: 0.3630 - val_accuracy: 0.9135
Epoch 22/30
7352/7352 [==============================] - 80s 11ms/step - loss: nan -
accuracy: 0.5140 - val_loss: nan - val_accuracy: 0.1683
Epoch 23/30
7352/7352 [==============================] - 86s 12ms/step - loss: nan -
accuracy: 0.1668 - val_loss: nan - val_accuracy: 0.1683
Epoch 24/30
7352/7352 [==============================] - 85s 12ms/step - loss: nan -
accuracy: 0.1668 - val_loss: nan - val_accuracy: 0.1683
Epoch 25/30
7352/7352 [==============================] - 110s 15ms/step - loss: nan -
accuracy: 0.1668 - val_loss: nan - val_accuracy: 0.1683
Epoch 26/30
7352/7352 [==============================] - 90s 12ms/step - loss: nan -
accuracy: 0.1668 - val_loss: nan - val_accuracy: 0.1683
Epoch 27/30
7352/7352 [==============================] - 87s 12ms/step - loss: nan -
accuracy: 0.1668 - val_loss: nan - val_accuracy: 0.1683
Epoch 28/30
7352/7352 [==============================] - 87s 12ms/step - loss: nan -
accuracy: 0.1668 - val_loss: nan - val_accuracy: 0.1683
Epoch 29/30
7352/7352 [==============================] - 120s 16ms/step - loss: nan -
accuracy: 0.1668 - val_loss: nan - val_accuracy: 0.1683
Epoch 30/30
7352/7352 [==============================] - 122s 17ms/step - loss: nan -
accuracy: 0.1668 - val_loss: nan - val_accuracy: 0.1683
```

[120]:
```
<keras.callbacks.callbacks.History at 0x640211cc0>
```

[121]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                WALKING
True
LAYING                  537
```

```
SITTING                 491
STANDING                532
WALKING                 496
WALKING_DOWNSTAIRS      420
WALKING_UPSTAIRS        471
```

[122]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 6s 2ms/step
```

[123]:
```
score
```

[123]:
```
[nan, 0.16830675303936005]
```

- It's performance is worst

1.3.5 LSTM Layer (64,32) with 1 layer of dropout 0.6

[124]:
```
# update LSTM layers
n_hidden_1 = 64
n_hidden_2 = 32
```

[125]:
```
# https://stackoverflow.com/questions/51763983/
 ↪error-when-checking-target-expected-dense-1-to-have-3-dimensions-but-got-array
# https://github.com/keras-team/keras/issues/7403
# 1. You need to set return_sequences=True from first LSTM
# 2. You need to set return_sequences=True from second LSTM
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden_1, input_shape=(timesteps, input_dim),␣
 ↪return_sequences=True))
# Adding a dropout layer
# model.add(Dropout(0.5))
# Configuring the parameters
model.add(LSTM(n_hidden_2 , return_sequences=False))
# Adding a dropout layer
model.add(Dropout(0.6))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_32"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_54 (LSTM)               (None, 128, 64)           18944
_________________________________________________________________
```

```
lstm_55 (LSTM)               (None, 32)                12416
_________________________________________________________________
dropout_34 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_16 (Dense)             (None, 6)                 198
=================================================================
Total params: 31,558
Trainable params: 31,558
Non-trainable params: 0
_________________________________________________________________
```

[126]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[127]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 118s 16ms/step - loss: 1.2553 -
accuracy: 0.4732 - val_loss: 0.9856 - val_accuracy: 0.6434
Epoch 2/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.9545 -
accuracy: 0.6049 - val_loss: 0.8514 - val_accuracy: 0.6244
Epoch 3/30
7352/7352 [==============================] - 136s 18ms/step - loss: 0.7894 -
accuracy: 0.6738 - val_loss: 0.6679 - val_accuracy: 0.7408
Epoch 4/30
7352/7352 [==============================] - 119s 16ms/step - loss: 0.6387 -
accuracy: 0.7881 - val_loss: 0.5631 - val_accuracy: 0.7842
Epoch 5/30
7352/7352 [==============================] - 93s 13ms/step - loss: 0.4644 -
accuracy: 0.8754 - val_loss: 0.4742 - val_accuracy: 0.8548
Epoch 6/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.3441 -
accuracy: 0.9072 - val_loss: 0.3645 - val_accuracy: 0.8972
Epoch 7/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.3192 -
accuracy: 0.9155 - val_loss: 0.4117 - val_accuracy: 0.8904
Epoch 8/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.3052 -
accuracy: 0.9158 - val_loss: 0.4517 - val_accuracy: 0.8768
```

```
Epoch 9/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.2712 -
accuracy: 0.9221 - val_loss: 0.3708 - val_accuracy: 0.9002
Epoch 10/30
7352/7352 [==============================] - 77s 10ms/step - loss: 0.2295 -
accuracy: 0.9372 - val_loss: 0.3945 - val_accuracy: 0.9108
Epoch 11/30
7352/7352 [==============================] - 77s 11ms/step - loss: 0.2267 -
accuracy: 0.9385 - val_loss: 0.5272 - val_accuracy: 0.8935
Epoch 12/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.2112 -
accuracy: 0.9363 - val_loss: 0.4001 - val_accuracy: 0.9060
Epoch 13/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.2116 -
accuracy: 0.9414 - val_loss: 0.4747 - val_accuracy: 0.9023
Epoch 14/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.2028 -
accuracy: 0.9385 - val_loss: 0.4646 - val_accuracy: 0.9070
Epoch 15/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.1896 -
accuracy: 0.9414 - val_loss: 0.4164 - val_accuracy: 0.9033
Epoch 16/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1874 -
accuracy: 0.9423 - val_loss: 0.2969 - val_accuracy: 0.9179
Epoch 17/30
7352/7352 [==============================] - 81s 11ms/step - loss: 0.1810 -
accuracy: 0.9399 - val_loss: 0.2758 - val_accuracy: 0.9182
Epoch 18/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1782 -
accuracy: 0.9425 - val_loss: 0.5199 - val_accuracy: 0.8941
Epoch 19/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1690 -
accuracy: 0.9476 - val_loss: 0.3118 - val_accuracy: 0.9213
Epoch 20/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.1558 -
accuracy: 0.9433 - val_loss: 0.4242 - val_accuracy: 0.9155
Epoch 21/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.1561 -
accuracy: 0.9505 - val_loss: 0.4087 - val_accuracy: 0.9026
Epoch 22/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1585 -
accuracy: 0.9467 - val_loss: 0.4679 - val_accuracy: 0.9019
Epoch 23/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.1696 -
accuracy: 0.9452 - val_loss: 0.4305 - val_accuracy: 0.8999
Epoch 24/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1605 -
accuracy: 0.9479 - val_loss: 0.4813 - val_accuracy: 0.9019
```

```
Epoch 25/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1505 -
accuracy: 0.9480 - val_loss: 0.5688 - val_accuracy: 0.9016
Epoch 26/30
7352/7352 [==============================] - 77s 11ms/step - loss: 0.1430 -
accuracy: 0.9498 - val_loss: 0.5675 - val_accuracy: 0.8951
Epoch 27/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.1477 -
accuracy: 0.9498 - val_loss: 0.6166 - val_accuracy: 0.8989
Epoch 28/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.1605 -
accuracy: 0.9445 - val_loss: 0.3368 - val_accuracy: 0.9169
Epoch 29/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1704 -
accuracy: 0.9456 - val_loss: 0.3571 - val_accuracy: 0.9175
Epoch 30/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1643 -
accuracy: 0.9449 - val_loss: 0.5376 - val_accuracy: 0.9053
```

[127]:
```
<keras.callbacks.callbacks.History at 0x63edc1a58>
```

[128]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 512        0        25        0                   0
SITTING                  2      400        87        0                   0
STANDING                 0       95       436        1                   0
WALKING                  0        0         0      450                   3
WALKING_DOWNSTAIRS       0        0         0        2                 414
WALKING_UPSTAIRS         1        0         0       11                   3

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            2
STANDING                           0
WALKING                           43
WALKING_DOWNSTAIRS                 4
WALKING_UPSTAIRS                 456
```

[129]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 4s 1ms/step
```

[130]:
```
score
```

[130]:
```
[0.5375813010956876, 0.9053274393081665]
```

- It's performance is better but loss is higher

1.3.6 LSTM Layer (64,32) with 1 layer of dropout 0.5

[131]:
```
# update LSTM layers
n_hidden_1 = 64
n_hidden_2 = 32
```

[132]:
```
# https://stackoverflow.com/questions/51763983/
 ↪error-when-checking-target-expected-dense-1-to-have-3-dimensions-but-got-array
# https://github.com/keras-team/keras/issues/7403
# 1. You need to set return_sequences=True from first LSTM
# 2. You need to set return_sequences=True from second LSTM
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden_1, input_shape=(timesteps, input_dim),␣
 ↪return_sequences=True))
# Adding a dropout layer
# model.add(Dropout(0.5))
# Configuring the parameters
model.add(LSTM(n_hidden_2 , return_sequences=False))
# Adding a dropout layer
model.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_33"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_56 (LSTM)               (None, 128, 64)           18944
_________________________________________________________________
lstm_57 (LSTM)               (None, 32)                12416
_________________________________________________________________
dropout_35 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_17 (Dense)             (None, 6)                 198
=================================================================
Total params: 31,558
Trainable params: 31,558
Non-trainable params: 0
_________________________________________________________________
```

[133]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[134]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 80s 11ms/step - loss: 1.1323 -
accuracy: 0.5397 - val_loss: 0.8477 - val_accuracy: 0.6787
Epoch 2/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.7404 -
accuracy: 0.7203 - val_loss: 0.5979 - val_accuracy: 0.7699
Epoch 3/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.5092 -
accuracy: 0.8400 - val_loss: 1.2509 - val_accuracy: 0.6763
Epoch 4/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.3954 -
accuracy: 0.8917 - val_loss: 0.4537 - val_accuracy: 0.8300
Epoch 5/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.2696 -
accuracy: 0.9217 - val_loss: 0.3626 - val_accuracy: 0.8887
Epoch 6/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.2443 -
accuracy: 0.9300 - val_loss: 0.3439 - val_accuracy: 0.8975
Epoch 7/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.2122 -
accuracy: 0.9350 - val_loss: 0.4547 - val_accuracy: 0.8996
Epoch 8/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.2518 -
accuracy: 0.9225 - val_loss: 0.3921 - val_accuracy: 0.8982
Epoch 9/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.1827 -
accuracy: 0.9440 - val_loss: 0.3997 - val_accuracy: 0.9019
Epoch 10/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1704 -
accuracy: 0.9460 - val_loss: 0.4512 - val_accuracy: 0.8782
Epoch 11/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.1747 -
accuracy: 0.9406 - val_loss: 0.3169 - val_accuracy: 0.9080
Epoch 12/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1636 -
```

```
accuracy: 0.9459 - val_loss: 0.2829 - val_accuracy: 0.9162
Epoch 13/30
7352/7352 [==============================] - 77s 10ms/step - loss: 0.1427 -
accuracy: 0.9514 - val_loss: 0.3661 - val_accuracy: 0.9013
Epoch 14/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.1546 -
accuracy: 0.9475 - val_loss: 0.3250 - val_accuracy: 0.9128
Epoch 15/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1498 -
accuracy: 0.9518 - val_loss: 0.4040 - val_accuracy: 0.9046
Epoch 16/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1360 -
accuracy: 0.9525 - val_loss: 0.3080 - val_accuracy: 0.9087
Epoch 17/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.1641 -
accuracy: 0.9431 - val_loss: 0.3339 - val_accuracy: 0.9189
Epoch 18/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.1443 -
accuracy: 0.9497 - val_loss: 0.3439 - val_accuracy: 0.9084
Epoch 19/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.1435 -
accuracy: 0.9494 - val_loss: 0.2949 - val_accuracy: 0.9104
Epoch 20/30
7352/7352 [==============================] - 77s 10ms/step - loss: 0.1318 -
accuracy: 0.9543 - val_loss: 0.4108 - val_accuracy: 0.9074
Epoch 21/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.1448 -
accuracy: 0.9509 - val_loss: 0.3502 - val_accuracy: 0.9135
Epoch 22/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.1454 -
accuracy: 0.9495 - val_loss: 0.3810 - val_accuracy: 0.9101
Epoch 23/30
7352/7352 [==============================] - 77s 10ms/step - loss: 0.1220 -
accuracy: 0.9547 - val_loss: 0.4707 - val_accuracy: 0.9040
Epoch 24/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.1434 -
accuracy: 0.9487 - val_loss: 0.4171 - val_accuracy: 0.9087
Epoch 25/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1398 -
accuracy: 0.9506 - val_loss: 0.3381 - val_accuracy: 0.9179
Epoch 26/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.1377 -
accuracy: 0.9504 - val_loss: 0.3495 - val_accuracy: 0.9114
Epoch 27/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1352 -
accuracy: 0.9512 - val_loss: 0.3892 - val_accuracy: 0.9175
Epoch 28/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.1536 -
```

```
accuracy: 0.9480 - val_loss: 0.5848 - val_accuracy: 0.8985
Epoch 29/30
7352/7352 [==============================] - 77s 11ms/step - loss: 0.1492 -
accuracy: 0.9535 - val_loss: 0.4154 - val_accuracy: 0.9135
Epoch 30/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1280 -
accuracy: 0.9525 - val_loss: 0.3805 - val_accuracy: 0.9141
```

[134]:
```
<keras.callbacks.callbacks.History at 0x64262e940>
```

[135]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 510        0        25        0                   0
SITTING                  3      416        68        0                   2
STANDING                 0       96       434        1                   0
WALKING                  0        0         2      473                  20
WALKING_DOWNSTAIRS       0        0         0        4                 413
WALKING_UPSTAIRS         0        1         6       16                   0

Pred                WALKING_UPSTAIRS
True
LAYING                             2
SITTING                            2
STANDING                           1
WALKING                            1
WALKING_DOWNSTAIRS                 3
WALKING_UPSTAIRS                 448
```

[136]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 4s 1ms/step
```

[137]:
```
score
```

[137]:
```
[0.3804781971405892, 0.9141499996185303]
```

- It's perform much better compared to 0.6 and 0.7

1.3.7 LSTM Layer (64,16) with dropout 0.5

[141]:
```
# update LSTM layers
n_hidden_1 = 64
n_hidden_2 = 16
```

- Initialize the LSTM Architecture

[142]:
```
# https://stackoverflow.com/questions/51763983/
 ↪error-when-checking-target-expected-dense-1-to-have-3-dimensions-but-got-array
# https://github.com/keras-team/keras/issues/7403
# 1. You need to set return_sequences=True from first LSTM
# 2. You need to set return_sequences=True from second LSTM
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden_1, input_shape=(timesteps, input_dim),␣
 ↪return_sequences=True))
# Adding a dropout layer
# model.add(Dropout(0.6))
# Configuring the parameters
model.add(LSTM(n_hidden_2 , return_sequences=False))
# Adding a dropout layer
model.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_35"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_60 (LSTM)               (None, 128, 64)           18944
_________________________________________________________________
lstm_61 (LSTM)               (None, 16)                5184
_________________________________________________________________
dropout_37 (Dropout)         (None, 16)                0
_________________________________________________________________
dense_19 (Dense)             (None, 6)                 102
=================================================================
Total params: 24,230
Trainable params: 24,230
Non-trainable params: 0
_________________________________________________________________
```

[143]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[144]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 87s 12ms/step - loss: 1.2980 -
accuracy: 0.5016 - val_loss: 1.0061 - val_accuracy: 0.5796
Epoch 2/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.9582 -
accuracy: 0.6060 - val_loss: 0.8328 - val_accuracy: 0.6495
Epoch 3/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.8071 -
accuracy: 0.7008 - val_loss: 0.6999 - val_accuracy: 0.7710
Epoch 4/30
7352/7352 [==============================] - 79s 11ms/step - loss: 0.6449 -
accuracy: 0.7953 - val_loss: 0.5793 - val_accuracy: 0.8297
Epoch 5/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.5014 -
accuracy: 0.8687 - val_loss: 0.5518 - val_accuracy: 0.8422
Epoch 6/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.4958 -
accuracy: 0.8734 - val_loss: 0.3736 - val_accuracy: 0.8860
Epoch 7/30
7352/7352 [==============================] - 81s 11ms/step - loss: 0.3628 -
accuracy: 0.9060 - val_loss: 0.4399 - val_accuracy: 0.8860
Epoch 8/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.3272 -
accuracy: 0.9119 - val_loss: 0.3274 - val_accuracy: 0.9030
Epoch 9/30
7352/7352 [==============================] - 79s 11ms/step - loss: 0.2879 -
accuracy: 0.9218 - val_loss: 0.3329 - val_accuracy: 0.9009
Epoch 10/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.2658 -
accuracy: 0.9252 - val_loss: 0.6884 - val_accuracy: 0.8347
Epoch 11/30
7352/7352 [==============================] - 77s 10ms/step - loss: 0.2757 -
accuracy: 0.9193 - val_loss: 0.4459 - val_accuracy: 0.8955
Epoch 12/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.2671 -
accuracy: 0.9245 - val_loss: 0.3746 - val_accuracy: 0.9070
Epoch 13/30
7352/7352 [==============================] - 92s 12ms/step - loss: 0.2660 -
accuracy: 0.9229 - val_loss: 0.3872 - val_accuracy: 0.9036
Epoch 14/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.2746 -
accuracy: 0.9197 - val_loss: 0.3580 - val_accuracy: 0.9091
Epoch 15/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.2464 -
accuracy: 0.9225 - val_loss: 0.4930 - val_accuracy: 0.8975
Epoch 16/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.2147 -
```

```
accuracy: 0.9320 - val_loss: 0.4268 - val_accuracy: 0.9063
Epoch 17/30
7352/7352 [==============================] - 84s 11ms/step - loss: 0.2211 -
accuracy: 0.9260 - val_loss: 0.4903 - val_accuracy: 0.8999
Epoch 18/30
7352/7352 [==============================] - 93s 13ms/step - loss: 0.2274 -
accuracy: 0.9261 - val_loss: 0.4916 - val_accuracy: 0.9063
Epoch 19/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.2101 -
accuracy: 0.9335 - val_loss: 0.5254 - val_accuracy: 0.9013
Epoch 20/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.2397 -
accuracy: 0.9218 - val_loss: 0.4428 - val_accuracy: 0.9186
Epoch 21/30
7352/7352 [==============================] - 81s 11ms/step - loss: 0.2112 -
accuracy: 0.9241 - val_loss: 0.4918 - val_accuracy: 0.9023
Epoch 22/30
7352/7352 [==============================] - 86s 12ms/step - loss: 0.2058 -
accuracy: 0.9283 - val_loss: 0.4324 - val_accuracy: 0.9175
Epoch 23/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.2068 -
accuracy: 0.9316 - val_loss: 0.5082 - val_accuracy: 0.8951
Epoch 24/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.2010 -
accuracy: 0.9325 - val_loss: 0.4982 - val_accuracy: 0.9026
Epoch 25/30
7352/7352 [==============================] - 92s 13ms/step - loss: 0.2106 -
accuracy: 0.9316 - val_loss: 0.6370 - val_accuracy: 0.8921
Epoch 26/30
7352/7352 [==============================] - 83s 11ms/step - loss: 0.2256 -
accuracy: 0.9276 - val_loss: 0.5423 - val_accuracy: 0.9023
Epoch 27/30
7352/7352 [==============================] - 84s 11ms/step - loss: 0.2045 -
accuracy: 0.9340 - val_loss: 0.5803 - val_accuracy: 0.9141
Epoch 28/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.2197 -
accuracy: 0.9323 - val_loss: 0.5890 - val_accuracy: 0.9050
Epoch 29/30
7352/7352 [==============================] - 95s 13ms/step - loss: 0.1964 -
accuracy: 0.9357 - val_loss: 0.5434 - val_accuracy: 0.9091
Epoch 30/30
7352/7352 [==============================] - 93s 13ms/step - loss: 0.1971 -
accuracy: 0.9308 - val_loss: 0.5415 - val_accuracy: 0.9057
```

[144]:
```
<keras.callbacks.callbacks.History at 0x644516ef0>
```

[145]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 510        0         3        0                   0
SITTING                  5      425        57        2                   0
STANDING                 0      101       430        0                   0
WALKING                  0        0         0      444                  34
WALKING_DOWNSTAIRS       0        0         0        5                 401
WALKING_UPSTAIRS         0        0         0        0                  12

Pred                WALKING_UPSTAIRS
True
LAYING                            24
SITTING                            2
STANDING                           1
WALKING                           18
WALKING_DOWNSTAIRS                14
WALKING_UPSTAIRS                 459
```

[146]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 4s 1ms/step
```

[147]:
```
score
```

[147]:
```
[0.5414807511955393, 0.9056667685508728]
```

- It's performance is good but loss is high

1.3.8 LSTM Layer (32,16) with dropout 0.5

[148]:
```
# update LSTM layers
n_hidden_1 = 32
n_hidden_2 = 16
```

- Initialize the LSTM Architecture

[149]:
```
# https://stackoverflow.com/questions/51763983/
 ↪error-when-checking-target-expected-dense-1-to-have-3-dimensions-but-got-array
# https://github.com/keras-team/keras/issues/7403
# 1. You need to set return_sequences=True from first LSTM
# 2. You need to set return_sequences=True from second LSTM
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden_1, input_shape=(timesteps, input_dim),␣
 ↪return_sequences=True))
# Adding a dropout layer
# model.add(Dropout(0.6))
```

```
# Configuring the parameters
model.add(LSTM(n_hidden_2 , return_sequences=False))
# Adding a dropout layer
model.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_36"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_62 (LSTM)               (None, 128, 32)           5376
_________________________________________________________________
lstm_63 (LSTM)               (None, 16)                3136
_________________________________________________________________
dropout_38 (Dropout)         (None, 16)                0
_________________________________________________________________
dense_20 (Dense)             (None, 6)                 102
=================================================================
Total params: 8,614
Trainable params: 8,614
Non-trainable params: 0
_________________________________________________________________
```

[150]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[151]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 108s 15ms/step - loss: 1.3059 -
accuracy: 0.4993 - val_loss: 1.0121 - val_accuracy: 0.5724
Epoch 2/30
7352/7352 [==============================] - 113s 15ms/step - loss: 0.9606 -
accuracy: 0.5797 - val_loss: 0.7977 - val_accuracy: 0.6081
Epoch 3/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.8199 -
accuracy: 0.6274 - val_loss: 0.7922 - val_accuracy: 0.5938
```

```
Epoch 4/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.7544 -
accuracy: 0.6387 - val_loss: 0.7379 - val_accuracy: 0.6179
Epoch 5/30
7352/7352 [==============================] - 111s 15ms/step - loss: 0.7201 -
accuracy: 0.6538 - val_loss: 0.7295 - val_accuracy: 0.6315
Epoch 6/30
7352/7352 [==============================] - 93s 13ms/step - loss: 0.7087 -
accuracy: 0.6568 - val_loss: 0.7252 - val_accuracy: 0.6247
Epoch 7/30
7352/7352 [==============================] - 92s 13ms/step - loss: 0.6802 -
accuracy: 0.6673 - val_loss: 0.7158 - val_accuracy: 0.6172
Epoch 8/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.6422 -
accuracy: 0.6827 - val_loss: 0.6055 - val_accuracy: 0.6335
Epoch 9/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.6010 -
accuracy: 0.6967 - val_loss: 0.5963 - val_accuracy: 0.6345
Epoch 10/30
7352/7352 [==============================] - 84s 11ms/step - loss: 0.5590 -
accuracy: 0.7346 - val_loss: 0.5663 - val_accuracy: 0.7455
Epoch 11/30
7352/7352 [==============================] - 86s 12ms/step - loss: 0.5296 -
accuracy: 0.7776 - val_loss: 0.5300 - val_accuracy: 0.7645
Epoch 12/30
7352/7352 [==============================] - 84s 11ms/step - loss: 0.4499 -
accuracy: 0.8449 - val_loss: 0.4595 - val_accuracy: 0.8690
Epoch 13/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.3973 -
accuracy: 0.8825 - val_loss: 0.3912 - val_accuracy: 0.8748
Epoch 14/30
7352/7352 [==============================] - 69s 9ms/step - loss: 0.3332 -
accuracy: 0.8976 - val_loss: 0.3851 - val_accuracy: 0.8928
Epoch 15/30
7352/7352 [==============================] - 69s 9ms/step - loss: 0.3229 -
accuracy: 0.9115 - val_loss: 0.4235 - val_accuracy: 0.8772
Epoch 16/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.2927 -
accuracy: 0.9166 - val_loss: 0.3071 - val_accuracy: 0.9013
Epoch 17/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.2774 -
accuracy: 0.9210 - val_loss: 0.4124 - val_accuracy: 0.9019
Epoch 18/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.2632 -
accuracy: 0.9267 - val_loss: 0.3328 - val_accuracy: 0.9053
Epoch 19/30
7352/7352 [==============================] - 86s 12ms/step - loss: 0.2567 -
accuracy: 0.9286 - val_loss: 0.3363 - val_accuracy: 0.8921
```

```
Epoch 20/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.2283 -
accuracy: 0.9346 - val_loss: 0.3214 - val_accuracy: 0.9077
Epoch 21/30
7352/7352 [==============================] - 70s 9ms/step - loss: 0.2282 -
accuracy: 0.9357 - val_loss: 0.4113 - val_accuracy: 0.8965
Epoch 22/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.2258 -
accuracy: 0.9332 - val_loss: 0.3970 - val_accuracy: 0.8965
Epoch 23/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.2020 -
accuracy: 0.9400 - val_loss: 0.3651 - val_accuracy: 0.9077
Epoch 24/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.2156 -
accuracy: 0.9372 - val_loss: 0.2834 - val_accuracy: 0.9199
Epoch 25/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.2100 -
accuracy: 0.9369 - val_loss: 0.2916 - val_accuracy: 0.9145
Epoch 26/30
7352/7352 [==============================] - 70s 9ms/step - loss: 0.2071 -
accuracy: 0.9369 - val_loss: 0.3984 - val_accuracy: 0.9053
Epoch 27/30
7352/7352 [==============================] - 70s 10ms/step - loss: 0.2109 -
accuracy: 0.9388 - val_loss: 0.3134 - val_accuracy: 0.9172
Epoch 28/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.2009 -
accuracy: 0.9374 - val_loss: 0.3572 - val_accuracy: 0.9199
Epoch 29/30
7352/7352 [==============================] - 81s 11ms/step - loss: 0.2002 -
accuracy: 0.9377 - val_loss: 0.3658 - val_accuracy: 0.9128
Epoch 30/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.2017 -
accuracy: 0.9395 - val_loss: 0.3589 - val_accuracy: 0.9226
```

[151]:
```
<keras.callbacks.callbacks.History at 0x6456d7a20>
```

[152]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 536        0         1        0                   0
SITTING                  1      414        71        2                   1
STANDING                 0       80       445        7                   0
WALKING                  0        0         0      458                   9
WALKING_DOWNSTAIRS       0        6         0        1                 403
WALKING_UPSTAIRS         0        1         0        4                   3
```

```
Pred                 WALKING_UPSTAIRS
True
LAYING                              0
SITTING                             2
STANDING                            0
WALKING                            29
WALKING_DOWNSTAIRS                 10
WALKING_UPSTAIRS                  463
```

[153]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 3s 1ms/step
```

[154]:
```
score
```

[154]:
```
[0.3589311393391746, 0.922633171081543]
```

- It's performance is best till now, I got 92.26% accuracy and loss 0.3589

1.3.9 LSTM Layer (32,8) with dropout 0.5

[155]:
```
# update LSTM layers
n_hidden_1 = 32
n_hidden_2 = 8
```

- Initialize the LSTM Architecture

[156]:
```
# https://stackoverflow.com/questions/51763983/
 ↪error-when-checking-target-expected-dense-1-to-have-3-dimensions-but-got-array
# https://github.com/keras-team/keras/issues/7403
# 1. You need to set return_sequences=True from first LSTM
# 2. You need to set return_sequences=True from second LSTM
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden_1, input_shape=(timesteps, input_dim),␣
 ↪return_sequences=True))
# Adding a dropout layer
# model.add(Dropout(0.6))
# Configuring the parameters
model.add(LSTM(n_hidden_2 , return_sequences=False))
# Adding a dropout layer
model.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_37"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_64 (LSTM)               (None, 128, 32)           5376
_________________________________________________________________
lstm_65 (LSTM)               (None, 8)                 1312
_________________________________________________________________
dropout_39 (Dropout)         (None, 8)                 0
_________________________________________________________________
dense_21 (Dense)             (None, 6)                 54
=================================================================
Total params: 6,742
Trainable params: 6,742
Non-trainable params: 0
_________________________________________________________________
```

[157]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[158]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 72s 10ms/step - loss: 1.4856 -
accuracy: 0.4123 - val_loss: 1.3371 - val_accuracy: 0.4608
Epoch 2/30
7352/7352 [==============================] - 71s 10ms/step - loss: 1.2285 -
accuracy: 0.5116 - val_loss: 1.0945 - val_accuracy: 0.5270
Epoch 3/30
7352/7352 [==============================] - 73s 10ms/step - loss: 1.0665 -
accuracy: 0.5305 - val_loss: 0.9436 - val_accuracy: 0.5541
Epoch 4/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.9660 -
accuracy: 0.5257 - val_loss: 0.9440 - val_accuracy: 0.4880
Epoch 5/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.9060 -
accuracy: 0.5547 - val_loss: 0.8240 - val_accuracy: 0.5684
Epoch 6/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.8786 -
accuracy: 0.5654 - val_loss: 0.7971 - val_accuracy: 0.6529
```

```
Epoch 7/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.8483 -
accuracy: 0.5914 - val_loss: 0.8113 - val_accuracy: 0.6586
Epoch 8/30
7352/7352 [==============================] - 68s 9ms/step - loss: 0.8286 -
accuracy: 0.6151 - val_loss: 0.8513 - val_accuracy: 0.6071
Epoch 9/30
7352/7352 [==============================] - 62s 8ms/step - loss: 0.8480 -
accuracy: 0.6223 - val_loss: 0.7308 - val_accuracy: 0.6508
Epoch 10/30
7352/7352 [==============================] - 62s 8ms/step - loss: 0.7699 -
accuracy: 0.6659 - val_loss: 0.6510 - val_accuracy: 0.7122
Epoch 11/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.7407 -
accuracy: 0.6878 - val_loss: 0.6161 - val_accuracy: 0.7282
Epoch 12/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.7204 -
accuracy: 0.7065 - val_loss: 0.6398 - val_accuracy: 0.7489
Epoch 13/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.6857 -
accuracy: 0.7171 - val_loss: 0.5320 - val_accuracy: 0.7818
Epoch 14/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.6436 -
accuracy: 0.7297 - val_loss: 0.5111 - val_accuracy: 0.7374
Epoch 15/30
7352/7352 [==============================] - 61s 8ms/step - loss: 0.6151 -
accuracy: 0.7432 - val_loss: 0.4787 - val_accuracy: 0.7296
Epoch 16/30
7352/7352 [==============================] - 62s 8ms/step - loss: 0.6339 -
accuracy: 0.7444 - val_loss: 0.4256 - val_accuracy: 0.7642
Epoch 17/30
7352/7352 [==============================] - 62s 8ms/step - loss: 0.5582 -
accuracy: 0.7655 - val_loss: 0.4290 - val_accuracy: 0.7560
Epoch 18/30
7352/7352 [==============================] - 61s 8ms/step - loss: 0.5327 -
accuracy: 0.7703 - val_loss: 0.4810 - val_accuracy: 0.7669
Epoch 19/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.5647 -
accuracy: 0.7677 - val_loss: 0.4285 - val_accuracy: 0.7516
Epoch 20/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.5090 -
accuracy: 0.7817 - val_loss: 0.3965 - val_accuracy: 0.7401
Epoch 21/30
7352/7352 [==============================] - 69s 9ms/step - loss: 0.5086 -
accuracy: 0.7835 - val_loss: 0.4745 - val_accuracy: 0.7418
Epoch 22/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.4800 -
accuracy: 0.7829 - val_loss: 0.4516 - val_accuracy: 0.7282
```

```
Epoch 23/30
7352/7352 [==============================] - 69s 9ms/step - loss: 0.4852 -
accuracy: 0.7777 - val_loss: 0.4051 - val_accuracy: 0.7533
Epoch 24/30
7352/7352 [==============================] - 69s 9ms/step - loss: 0.4614 -
accuracy: 0.7845 - val_loss: 0.3709 - val_accuracy: 0.7540
Epoch 25/30
7352/7352 [==============================] - 70s 10ms/step - loss: 0.4826 -
accuracy: 0.7795 - val_loss: 1.3639 - val_accuracy: 0.6037
Epoch 26/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.5301 -
accuracy: 0.7686 - val_loss: 0.8841 - val_accuracy: 0.6529
Epoch 27/30
7352/7352 [==============================] - 69s 9ms/step - loss: 0.4808 -
accuracy: 0.7758 - val_loss: 0.3784 - val_accuracy: 0.7472
Epoch 28/30
7352/7352 [==============================] - 68s 9ms/step - loss: 0.4476 -
accuracy: 0.7924 - val_loss: 0.4192 - val_accuracy: 0.7520
Epoch 29/30
7352/7352 [==============================] - 69s 9ms/step - loss: 0.4208 -
accuracy: 0.8067 - val_loss: 0.3983 - val_accuracy: 0.7496
Epoch 30/30
7352/7352 [==============================] - 69s 9ms/step - loss: 0.4232 -
accuracy: 0.7926 - val_loss: 0.3762 - val_accuracy: 0.7540
```

[158]:
```
<keras.callbacks.callbacks.History at 0x64672ce10>
```

[159]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 534        0         0        0                   0
SITTING                  0      420        61        0                   0
STANDING                 0      122       405        0                   0
WALKING                  0        0         0        0                   0
WALKING_DOWNSTAIRS       0        0         0        4                 397
WALKING_UPSTAIRS         0        5         0        0                   0

Pred                WALKING_UPSTAIRS
True
LAYING                             3
SITTING                           10
STANDING                           5
WALKING                          496
WALKING_DOWNSTAIRS                19
WALKING_UPSTAIRS                 466
```

[160]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 3s 953us/step
```

[161]:
```
score
```

[161]:
```
[0.3761567989555303, 0.7539871335029602]
```

- Performance degraded for this combination of LSTM

1.4.0 Assignment feedback to improve accuracy > 94%
1.4.1 Increase dropout on the best LSTM combination

[17]:
```
# update LSTM layers
n_hidden_1 = 32
n_hidden_2 = 16
```

- Initialize the LSTM Architecture

[18]:
```
# https://stackoverflow.com/questions/51763983/
 ↪error-when-checking-target-expected-dense-1-to-have-3-dimensions-but-got-array
# https://github.com/keras-team/keras/issues/7403
# 1. You need to set return_sequences=True from first LSTM
# 2. You need to set return_sequences=True from second LSTM
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden_1, input_shape=(timesteps, input_dim),␣
 ↪return_sequences=True))
# Adding a dropout layer
# model.add(Dropout(0.6))
# Configuring the parameters
model.add(LSTM(n_hidden_2 , return_sequences=False))
# Adding a dropout layer
model.add(Dropout(0.6))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
WARNING:tensorflow:Large dropout rate: 0.6 (>0.5). In TensorFlow 2.x, dropout()
uses dropout rate instead of keep_prob. Please ensure that this is intended.
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 128, 32)           5376
_________________________________________________________________
```

```
lstm_2 (LSTM)                (None, 16)                3136
_________________________________________________________________
dropout_1 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 102
=================================================================
Total params: 8,614
Trainable params: 8,614
Non-trainable params: 0
_________________________________________________________________
```

[19]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[20]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
WARNING:tensorflow:From /Users/mayankgupta/anaconda3/lib/python3.7/site-
packages/tensorflow/python/ops/math_grad.py:1250:
add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /Users/mayankgupta/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:422: The name tf.global_variables
is deprecated. Please use tf.compat.v1.global_variables instead.

Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 28s 4ms/step - loss: 1.3771 -
accuracy: 0.4679 - val_loss: 1.2075 - val_accuracy: 0.5063
Epoch 2/30
7352/7352 [==============================] - 29s 4ms/step - loss: 1.0661 -
accuracy: 0.5615 - val_loss: 0.9215 - val_accuracy: 0.6546
Epoch 3/30
7352/7352 [==============================] - 28s 4ms/step - loss: 0.9246 -
accuracy: 0.6035 - val_loss: 0.7882 - val_accuracy: 0.6474
Epoch 4/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.8082 -
accuracy: 0.6484 - val_loss: 0.9662 - val_accuracy: 0.6223
Epoch 5/30
7352/7352 [==============================] - 28s 4ms/step - loss: 0.7176 -
```

```
accuracy: 0.7084 - val_loss: 0.7233 - val_accuracy: 0.6702
Epoch 6/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.6610 -
accuracy: 0.7329 - val_loss: 0.6340 - val_accuracy: 0.7268
Epoch 7/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.5957 -
accuracy: 0.7606 - val_loss: 0.4884 - val_accuracy: 0.7458
Epoch 8/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.5444 -
accuracy: 0.7705 - val_loss: 0.4884 - val_accuracy: 0.7547
Epoch 9/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.5301 -
accuracy: 0.7749 - val_loss: 0.5527 - val_accuracy: 0.7448
Epoch 10/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.4985 -
accuracy: 0.7761 - val_loss: 0.5010 - val_accuracy: 0.7489
Epoch 11/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.4811 -
accuracy: 0.7811 - val_loss: 0.5013 - val_accuracy: 0.7706
Epoch 12/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.4723 -
accuracy: 0.7758 - val_loss: 0.5427 - val_accuracy: 0.7384
Epoch 13/30
7352/7352 [==============================] - 28s 4ms/step - loss: 0.4699 -
accuracy: 0.7801 - val_loss: 0.5415 - val_accuracy: 0.7638
Epoch 14/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.4475 -
accuracy: 0.7879 - val_loss: 0.5366 - val_accuracy: 0.7564
Epoch 15/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.4817 -
accuracy: 0.7795 - val_loss: 0.9002 - val_accuracy: 0.6841
Epoch 16/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.4505 -
accuracy: 0.7894 - val_loss: 0.4372 - val_accuracy: 0.7798
Epoch 17/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.4303 -
accuracy: 0.7878 - val_loss: 0.4430 - val_accuracy: 0.7845
Epoch 18/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.4273 -
accuracy: 0.7950 - val_loss: 0.4931 - val_accuracy: 0.7672
Epoch 19/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.4387 -
accuracy: 0.7996 - val_loss: 0.4052 - val_accuracy: 0.7927
Epoch 20/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.4401 -
accuracy: 0.8062 - val_loss: 0.4650 - val_accuracy: 0.7774
Epoch 21/30
7352/7352 [==============================] - 28s 4ms/step - loss: 0.4293 -
```

```
accuracy: 0.8244 - val_loss: 0.5555 - val_accuracy: 0.7587
Epoch 22/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.3975 -
accuracy: 0.8384 - val_loss: 0.4290 - val_accuracy: 0.7665
Epoch 23/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.3760 -
accuracy: 0.8474 - val_loss: 0.4030 - val_accuracy: 0.8836
Epoch 24/30
7352/7352 [==============================] - 28s 4ms/step - loss: 0.3758 -
accuracy: 0.8453 - val_loss: 0.4144 - val_accuracy: 0.9057
Epoch 25/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.3429 -
accuracy: 0.8659 - val_loss: 0.4352 - val_accuracy: 0.8948
Epoch 26/30
7352/7352 [==============================] - 27s 4ms/step - loss: 0.3329 -
accuracy: 0.8802 - val_loss: 0.4077 - val_accuracy: 0.8914
Epoch 27/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.3310 -
accuracy: 0.8977 - val_loss: 0.3974 - val_accuracy: 0.8948
Epoch 28/30
7352/7352 [==============================] - 26s 3ms/step - loss: 0.3205 -
accuracy: 0.9048 - val_loss: 0.3721 - val_accuracy: 0.8897
Epoch 29/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.2805 -
accuracy: 0.9168 - val_loss: 0.4258 - val_accuracy: 0.8833
Epoch 30/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.2897 -
accuracy: 0.9117 - val_loss: 0.3616 - val_accuracy: 0.9101
```

[20]:
```
<keras.callbacks.callbacks.History at 0x629ebef60>
```

[21]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 510        0         2        0                   0
SITTING                  0      404        87        0                   0
STANDING                 0       81       450        1                   0
WALKING                  0        0         0      450                  44
WALKING_DOWNSTAIRS       0        0         0        0                 414
WALKING_UPSTAIRS         1        0        10        0                   6

Pred                WALKING_UPSTAIRS
True
LAYING                            25
SITTING                            0
STANDING                           0
```

```
WALKING                          2
WALKING_DOWNSTAIRS               6
WALKING_UPSTAIRS               454
```

[22]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 1s 402us/step
```

[23]:
```
score
```

[23]:
```
[0.36157595883878874, 0.9100780487060547]
```

1.4.2 Decrease dropout on the best LSTM combination

[24]:
```
# update LSTM layers
n_hidden_1 = 32
n_hidden_2 = 16
```

- Initialize the LSTM architecture

[25]:
```
# https://stackoverflow.com/questions/51763983/
 ↪error-when-checking-target-expected-dense-1-to-have-3-dimensions-but-got-array
# https://github.com/keras-team/keras/issues/7403
# 1. You need to set return_sequences=True from first LSTM
# 2. You need to set return_sequences=True from second LSTM
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden_1, input_shape=(timesteps, input_dim),␣
 ↪return_sequences=True))
# Adding a dropout layer
# model.add(Dropout(0.6))
# Configuring the parameters
model.add(LSTM(n_hidden_2 , return_sequences=False))
# Adding a dropout layer
model.add(Dropout(0.4))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 128, 32)           5376
_________________________________________________________________
lstm_4 (LSTM)                (None, 16)                3136
_________________________________________________________________
```

```
dropout_2 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 102
=================================================================
Total params: 8,614
Trainable params: 8,614
Non-trainable params: 0
_________________________________________________________________
```

[26]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[27]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 30s 4ms/step - loss: 1.2955 -
accuracy: 0.5092 - val_loss: 0.9788 - val_accuracy: 0.5667
Epoch 2/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.9335 -
accuracy: 0.6260 - val_loss: 0.8553 - val_accuracy: 0.5850
Epoch 3/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.7509 -
accuracy: 0.6903 - val_loss: 0.6851 - val_accuracy: 0.6875
Epoch 4/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.6576 -
accuracy: 0.7371 - val_loss: 0.6529 - val_accuracy: 0.7153
Epoch 5/30
7352/7352 [==============================] - 28s 4ms/step - loss: 0.5653 -
accuracy: 0.7776 - val_loss: 0.5315 - val_accuracy: 0.7598
Epoch 6/30
7352/7352 [==============================] - 28s 4ms/step - loss: 0.4963 -
accuracy: 0.7875 - val_loss: 0.5693 - val_accuracy: 0.7397
Epoch 7/30
7352/7352 [==============================] - 28s 4ms/step - loss: 0.4593 -
accuracy: 0.8021 - val_loss: 0.8562 - val_accuracy: 0.6997
Epoch 8/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.4336 -
accuracy: 0.8022 - val_loss: 0.4901 - val_accuracy: 0.7733
Epoch 9/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.4291 -
```

```
accuracy: 0.8105 - val_loss: 0.4846 - val_accuracy: 0.8344
Epoch 10/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.3871 -
accuracy: 0.8746 - val_loss: 0.4090 - val_accuracy: 0.8626
Epoch 11/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.3232 -
accuracy: 0.9104 - val_loss: 0.3683 - val_accuracy: 0.8884
Epoch 12/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.2601 -
accuracy: 0.9312 - val_loss: 0.4261 - val_accuracy: 0.8789
Epoch 13/30
7352/7352 [==============================] - 27s 4ms/step - loss: 0.2371 -
accuracy: 0.9312 - val_loss: 0.3696 - val_accuracy: 0.8955
Epoch 14/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2219 -
accuracy: 0.9354 - val_loss: 0.3854 - val_accuracy: 0.8897
Epoch 15/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.2124 -
accuracy: 0.9388 - val_loss: 0.3734 - val_accuracy: 0.8890
Epoch 16/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1963 -
accuracy: 0.9403 - val_loss: 0.4284 - val_accuracy: 0.8887
Epoch 17/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1849 -
accuracy: 0.9396 - val_loss: 0.3498 - val_accuracy: 0.9016
Epoch 18/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1981 -
accuracy: 0.9415 - val_loss: 0.4388 - val_accuracy: 0.8924
Epoch 19/30
7352/7352 [==============================] - 27s 4ms/step - loss: 0.1978 -
accuracy: 0.9408 - val_loss: 0.4307 - val_accuracy: 0.8911
Epoch 20/30
7352/7352 [==============================] - 27s 4ms/step - loss: 0.1884 -
accuracy: 0.9377 - val_loss: 0.4191 - val_accuracy: 0.8948
Epoch 21/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1669 -
accuracy: 0.9444 - val_loss: 0.5406 - val_accuracy: 0.8901
Epoch 22/30
7352/7352 [==============================] - 27s 4ms/step - loss: 0.1731 -
accuracy: 0.9442 - val_loss: 0.4263 - val_accuracy: 0.8921
Epoch 23/30
7352/7352 [==============================] - 27s 4ms/step - loss: 0.1697 -
accuracy: 0.9459 - val_loss: 0.4441 - val_accuracy: 0.8904
Epoch 24/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.1615 -
accuracy: 0.9474 - val_loss: 0.4375 - val_accuracy: 0.8955
Epoch 25/30
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1651 -
```

```
accuracy: 0.9448 - val_loss: 0.6645 - val_accuracy: 0.8826
Epoch 26/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1563 -
accuracy: 0.9475 - val_loss: 0.6143 - val_accuracy: 0.8890
Epoch 27/30
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1582 -
accuracy: 0.9489 - val_loss: 0.5653 - val_accuracy: 0.8958
Epoch 28/30
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1520 -
accuracy: 0.9465 - val_loss: 0.3868 - val_accuracy: 0.9111
Epoch 29/30
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1610 -
accuracy: 0.9465 - val_loss: 0.4674 - val_accuracy: 0.9033
Epoch 30/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.1460 -
accuracy: 0.9489 - val_loss: 0.3913 - val_accuracy: 0.9067
```

[27]:
```
<keras.callbacks.callbacks.History at 0x62b400e10>
```

[28]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 537        0         0        0                   0
SITTING                  4      387        78        0                   4
STANDING                 0       98       432        0                   0
WALKING                  0        0         0      448                  10
WALKING_DOWNSTAIRS       0        0         0        0                 413
WALKING_UPSTAIRS         0        0         0        7                   9

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                           18
STANDING                           2
WALKING                           38
WALKING_DOWNSTAIRS                 7
WALKING_UPSTAIRS                 455
```

[29]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 1s 397us/step
```

[30]:
```
score
```

[30]:
```
[0.39125852214684165, 0.9066847562789917]
```

1.4.3 Increase layer size to 128 and lesser dropout

[37]:
```
# update LSTM layers
n_hidden_1 = 128
# n_hidden_2 = 16
```

- Initialize LSTM

[38]:
```
# https://stackoverflow.com/questions/51763983/
 ↪error-when-checking-target-expected-dense-1-to-have-3-dimensions-but-got-array
# https://github.com/keras-team/keras/issues/7403
# 1. You need to set return_sequences=True from first LSTM
# 2. You need to set return_sequences=True from second LSTM
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden_1, input_shape=(timesteps, input_dim)))#,␣
 ↪return_sequences=True
# Adding BatchNormalization
model.add(BatchNormalization())
# Adding a dropout layer
# model.add(Dropout(0.6))
# Configuring the parameters
# model.add(LSTM(n_hidden_2 , return_sequences=False))
# Adding a dropout layer
model.add(Dropout(0.25))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_8 (LSTM)                (None, 128)               70656
_________________________________________________________________
batch_normalization_3 (Batch (None, 128)               512
_________________________________________________________________
dropout_5 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 6)                 774
=================================================================
Total params: 71,942
Trainable params: 71,686
Non-trainable params: 256
_________________________________________________________________
```

[39]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
```

```
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[40]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.9291 -
accuracy: 0.5952 - val_loss: 0.7638 - val_accuracy: 0.6359
Epoch 2/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.6998 -
accuracy: 0.6602 - val_loss: 0.6965 - val_accuracy: 0.6481
Epoch 3/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.5196 -
accuracy: 0.7916 - val_loss: 0.3441 - val_accuracy: 0.8907
Epoch 4/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2674 -
accuracy: 0.9101 - val_loss: 0.3433 - val_accuracy: 0.8989
Epoch 5/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1985 -
accuracy: 0.9293 - val_loss: 0.3900 - val_accuracy: 0.8772
Epoch 6/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1990 -
accuracy: 0.9279 - val_loss: 0.3218 - val_accuracy: 0.9121
Epoch 7/30
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1790 -
accuracy: 0.9362 - val_loss: 0.2864 - val_accuracy: 0.9108
Epoch 8/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1543 -
accuracy: 0.9412 - val_loss: 0.2592 - val_accuracy: 0.9199
Epoch 9/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.1592 -
accuracy: 0.9404 - val_loss: 0.3458 - val_accuracy: 0.9158
Epoch 10/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1445 -
accuracy: 0.9436 - val_loss: 0.2678 - val_accuracy: 0.9199
Epoch 11/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1397 -
accuracy: 0.9395 - val_loss: 0.2976 - val_accuracy: 0.9155
Epoch 12/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.1523 -
accuracy: 0.9392 - val_loss: 0.2769 - val_accuracy: 0.9070
Epoch 13/30
```

```
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1375 -
accuracy: 0.9415 - val_loss: 0.3131 - val_accuracy: 0.9087
Epoch 14/30
7352/7352 [==============================] - 26s 3ms/step - loss: 0.1459 -
accuracy: 0.9433 - val_loss: 0.3071 - val_accuracy: 0.9152
Epoch 15/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1451 -
accuracy: 0.9455 - val_loss: 0.3450 - val_accuracy: 0.9169
Epoch 16/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1497 -
accuracy: 0.9460 - val_loss: 0.3297 - val_accuracy: 0.9158
Epoch 17/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1425 -
accuracy: 0.9489 - val_loss: 0.3909 - val_accuracy: 0.9192
Epoch 18/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1529 -
accuracy: 0.9465 - val_loss: 0.3178 - val_accuracy: 0.9220
Epoch 19/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1487 -
accuracy: 0.9456 - val_loss: 0.3939 - val_accuracy: 0.9104
Epoch 20/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1389 -
accuracy: 0.9483 - val_loss: 0.3776 - val_accuracy: 0.9179
Epoch 21/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1451 -
accuracy: 0.9449 - val_loss: 0.3028 - val_accuracy: 0.9233
Epoch 22/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1386 -
accuracy: 0.9465 - val_loss: 0.3515 - val_accuracy: 0.9247
Epoch 23/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.2044 -
accuracy: 0.9280 - val_loss: 0.3435 - val_accuracy: 0.9226
Epoch 24/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1235 -
accuracy: 0.9521 - val_loss: 0.3666 - val_accuracy: 0.9230
Epoch 25/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1276 -
accuracy: 0.9504 - val_loss: 0.3031 - val_accuracy: 0.9182
Epoch 26/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1243 -
accuracy: 0.9520 - val_loss: 0.3607 - val_accuracy: 0.9257
Epoch 27/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1223 -
accuracy: 0.9502 - val_loss: 0.3756 - val_accuracy: 0.9155
Epoch 28/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1273 -
accuracy: 0.9525 - val_loss: 0.3839 - val_accuracy: 0.9186
Epoch 29/30
```

```
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1620 -
accuracy: 0.9433 - val_loss: 0.2694 - val_accuracy: 0.9294
Epoch 30/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1218 -
accuracy: 0.9510 - val_loss: 0.3121 - val_accuracy: 0.9257
```

[40]:
```
<keras.callbacks.callbacks.History at 0x62db041d0>
```

[41]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 537        0         0        0                   0
SITTING                  0      418        71        0                   0
STANDING                 0      101       431        0                   0
WALKING                  0        0         0      470                  24
WALKING_DOWNSTAIRS       0        0         0        1                 417
WALKING_UPSTAIRS         0        1         0       13                   2

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            2
STANDING                           0
WALKING                            2
WALKING_DOWNSTAIRS                 2
WALKING_UPSTAIRS                 455
```

[42]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 2s 542us/step
```

[43]:
```
score
```

[43]:
```
[0.31214076887265046, 0.9256871342658997]
```

1.4.4 Implement Divide and Conquer Technique

[22]:
```
# Citation: https://github.com/UdiBhaskar/
 ↪Human-Activity-Recognition--Using-Deep-NN
```

In the dataset, Y_labels are represented as numbers from 1 to 6 as their identifiers.
WALKING as 1
WALKING_UPSTAIRS as 2
WALKING_DOWNSTAIRS as 3
SITTING as 4
STANDING as 5
LAYING as 6

- in Data exploration section we observed that we can divide the data into dynamic and static type so divided walking, walking_upstairs and walking_downstairs into category 0 i.e Dynamic and sitting, standing and laying into category 1 i.e. static. - Will use 2 more classifiers separately for classifying classes of dynamic and static activities. so that model can learn differnt features for static and dynamic activities

referred below paper

Divide and Conquer-Based 1D CNN Human Activity Recognition Using Test Data Sharpening (https://www.mdpi.com/1424-8220/18/4/1055/pdf)

[23]:
```
# 20200212 Mayank Gupta I completly refactored the code provided in the above␣
 ↪github repo,
# I didn't follow the oops based approach
```

[24]:
```
from keras.layers import Flatten
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
```

[25]:
```
import pickle

def dump_file(filename, mode, data):
    '''
    Save model on the disk
    '''
    pickle.dump(data, open(filename, mode))
```

[26]:
```
from sklearn.preprocessing import StandardScaler

def load_y_static_dynamic(subset):
        """
        The objective that we are trying to predict is a integer, from 1 to 6,
        that represents a human activity. We return a binary representation of
        every sample objective as a 6 bits vector using One Hot Encoding
        (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.
 ↪get_dummies.html)
        """
        filename = f'UCI_HAR_Dataset/{subset}/y_{subset}.txt'
        y = _read_csv(filename)[0]
        y[y<=3] = 0
        y[y>3] = 1
        return pd.get_dummies(y).as_matrix()

def load_data_static_dynamic():
    '''
    Load train, test data and scale the data as well
    '''
    X_train_2c, X_val_2c = load_signals('train'), load_signals('test')
    Y_train_2c, Y_val_2c = load_y_static_dynamic('train'),␣
 ↪load_y_static_dynamic('test')
```

```
    # fit and transform data
    Scale = fit(X_train_2c)
    dump_file('Scale_2class.p','wb', Scale)
    X_train_2c = transform(X_train_2c, Scale)
    X_val_2c = transform(X_val_2c, Scale)

    return X_train_2c, Y_train_2c, X_val_2c, Y_val_2c

def transform(X, scale):
    '''
    Transform the data
    '''
    temp_X1 = X.reshape((X.shape[0] * X.shape[1], X.shape[2]))
    temp_X1 = scale.transform(temp_X1)
    return temp_X1.reshape(X.shape)

def fit(X):
    '''
    Fit data for scaling
    '''
    # remove overlaping
    remove = int(X.shape[1] / 2)
    temp_X = X[:, -remove:, :]
    # flatten data
    temp_X = temp_X.reshape((temp_X.shape[0] * temp_X.shape[1], temp_X.
 ↪shape[2]))
    scale = StandardScaler()
    scale.fit(temp_X)
    return scale
```

[27]:
```
X_train_2c, Y_train_2c, X_val_2c,  Y_val_2c = load_data_static_dynamic()
```

```
/Users/mayankgupta/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:12: FutureWarning: Method .as_matrix will be
removed in a future version. Use .values instead.
  if sys.path[0] == '':
/Users/mayankgupta/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:14: FutureWarning: Method .as_matrix will be
removed in a future version. Use .values instead.
```

[28]:
```
print(Y_train_2c.shape)
print(Y_val_2c.shape)
```

```
(7352, 2)
(2947, 2)
```

1.4.4.1 Model for classifying data into Static and Dynamic activities

[29]:
```
# Clear previous session
K.clear_session()
# Add seed
np.random.seed(42)
tf.set_random_seed(42)
# Start Session
sess = tf.Session(graph=tf.get_default_graph())
K.set_session(sess)
# Create model
model = Sequential()
model.add(Conv1D(filters=32, kernel_size=3,␣
 ↪activation='relu',kernel_initializer='he_uniform',input_shape=(128,9)))
model.add(Conv1D(filters=32, kernel_size=3,␣
 ↪activation='relu',kernel_initializer='he_uniform'))
model.add(Dropout(0.6))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(50, activation='relu'))
model.add(Dense(2, activation='softmax'))
model.summary()
```

```
WARNING:tensorflow:Large dropout rate: 0.6 (>0.5). In TensorFlow 2.x, dropout()
uses dropout rate instead of keep_prob. Please ensure that this is intended.
WARNING:tensorflow:From /Users/mayankgupta/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:4070: The name tf.nn.max_pool is
deprecated. Please use tf.nn.max_pool2d instead.

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 126, 32)           896
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 124, 32)           3104
_________________________________________________________________
dropout_1 (Dropout)          (None, 124, 32)           0
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 62, 32)            0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1984)              0
_________________________________________________________________
dense_1 (Dense)              (None, 50)                99250
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 102
=================================================================
Total params: 103,352
Trainable params: 103,352
```

```
Non-trainable params: 0
_________________________________________________________________
```

[30]:
```
import keras
import math
adam = keras.optimizers.Adam(lr=0.001)
```

[31]:
```
model.compile(loss='categorical_crossentropy', optimizer=adam,␣
 ↪metrics=['accuracy'])
model.fit(X_train_2c,Y_train_2c, epochs=20,␣
 ↪batch_size=16,validation_data=(X_val_2c, Y_val_2c), verbose=1)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/20
7352/7352 [==============================] - 3s 448us/step - loss: 0.0371 -
accuracy: 0.9869 - val_loss: 0.0202 - val_accuracy: 0.9949
Epoch 2/20
7352/7352 [==============================] - 3s 432us/step - loss: 0.0053 -
accuracy: 0.9989 - val_loss: 0.0404 - val_accuracy: 0.9881
Epoch 3/20
7352/7352 [==============================] - 3s 429us/step - loss: 0.0013 -
accuracy: 0.9995 - val_loss: 0.0480 - val_accuracy: 0.9834
Epoch 4/20
7352/7352 [==============================] - 3s 433us/step - loss: 8.4357e-06 -
accuracy: 1.0000 - val_loss: 0.0514 - val_accuracy: 0.9834
Epoch 5/20
7352/7352 [==============================] - 3s 430us/step - loss: 0.0024 -
accuracy: 0.9990 - val_loss: 0.1564 - val_accuracy: 0.9851
Epoch 6/20
7352/7352 [==============================] - 3s 432us/step - loss: 1.7301e-04 -
accuracy: 1.0000 - val_loss: 0.1063 - val_accuracy: 0.9871
Epoch 7/20
7352/7352 [==============================] - 3s 435us/step - loss: 1.0457e-04 -
accuracy: 1.0000 - val_loss: 0.1719 - val_accuracy: 0.9810
Epoch 8/20
7352/7352 [==============================] - 3s 423us/step - loss: 1.5191e-05 -
accuracy: 1.0000 - val_loss: 0.1096 - val_accuracy: 0.9871
Epoch 9/20
7352/7352 [==============================] - 3s 430us/step - loss: 3.3170e-06 -
accuracy: 1.0000 - val_loss: 0.1095 - val_accuracy: 0.9871
Epoch 10/20
7352/7352 [==============================] - 3s 422us/step - loss: 0.0014 -
accuracy: 0.9996 - val_loss: 0.0903 - val_accuracy: 0.9891
Epoch 11/20
7352/7352 [==============================] - 3s 426us/step - loss: 0.0064 -
accuracy: 0.9988 - val_loss: 0.0193 - val_accuracy: 0.9963
Epoch 12/20
7352/7352 [==============================] - 3s 431us/step - loss: 2.6799e-05 -
```

```
accuracy: 1.0000 - val_loss: 0.0117 - val_accuracy: 0.9976
Epoch 13/20
7352/7352 [==============================] - 3s 426us/step - loss: 0.0015 -
accuracy: 0.9997 - val_loss: 0.0103 - val_accuracy: 0.9976
Epoch 14/20
7352/7352 [==============================] - 3s 430us/step - loss: 9.5532e-06 -
accuracy: 1.0000 - val_loss: 0.0098 - val_accuracy: 0.9980
Epoch 15/20
7352/7352 [==============================] - 3s 438us/step - loss: 1.2184e-06 -
accuracy: 1.0000 - val_loss: 0.0099 - val_accuracy: 0.9980
Epoch 16/20
7352/7352 [==============================] - 4s 522us/step - loss: 2.4589e-05 -
accuracy: 1.0000 - val_loss: 0.0077 - val_accuracy: 0.9983
Epoch 17/20
7352/7352 [==============================] - 3s 455us/step - loss: 7.1136e-06 -
accuracy: 1.0000 - val_loss: 0.0138 - val_accuracy: 0.9983
Epoch 18/20
7352/7352 [==============================] - 3s 434us/step - loss: 9.7108e-07 -
accuracy: 1.0000 - val_loss: 0.0126 - val_accuracy: 0.9983
Epoch 19/20
7352/7352 [==============================] - 3s 424us/step - loss: 2.4397e-07 -
accuracy: 1.0000 - val_loss: 0.0123 - val_accuracy: 0.9983
Epoch 20/20
7352/7352 [==============================] - 3s 423us/step - loss: 1.7281e-07 -
accuracy: 1.0000 - val_loss: 0.0121 - val_accuracy: 0.9983
```

[31]:
```
<keras.callbacks.callbacks.History at 0x633c85be0>
```

[32]:
```
_,acc_val = model.evaluate(X_val_2c,Y_val_2c,verbose=0)
_,acc_train = model.evaluate(X_train_2c,Y_train_2c,verbose=0)
print('Train_accuracy',acc_train,'test_accuracy',acc_val)
```

```
Train_accuracy 1.0 test_accuracy 0.9983033537864685
```

[33]:
```
##saving model
model.save('final_model_2class.h5')
```

- Classification of Static and Dynamic Activities is Perfect, We got 99.83% accuracy on test data

1.4.4.2 Model for Classifying Static Activities

[34]:
```
def load_y_static(subset):
        """
        The objective that we are trying to predict is a integer, from 1 to 6,
        that represents a human activity. We return a binary representation of
        every sample objective as a 6 bits vector using One Hot Encoding
```

```
        (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.
↪get_dummies.html)
        """
        filename = f'UCI_HAR_Dataset/{subset}/y_{subset}.txt'
        y = _read_csv(filename)[0]
        y_subset = y>3
        y = y[y_subset]
        return pd.get_dummies(y).as_matrix(),y_subset

def load_data_static():
    '''
    Load train, test data and scale the data as well
    '''
    Y_train_s, y_train_sub = load_y_static('train')
    Y_val_s, y_test_sub = load_y_static('test')

    X_train_s, X_val_s = load_signals('train'), load_signals('test')
    X_train_s = X_train_s[y_train_sub]
    X_val_s = X_val_s[y_test_sub]

    # fit and transform data
    Scale = None
    Scale = fit(X_train_s)
    dump_file('Scale_static.p','wb', Scale)
    X_train_s = transform(X_train_s, Scale)
    X_val_s = transform(X_val_s, Scale)

    return X_train_s, Y_train_s, X_val_s, Y_val_s
```

[35]:
```
X_train_s, Y_train_s, X_val_s,  Y_val_s = load_data_static()
```

```
/Users/mayankgupta/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:12: FutureWarning: Method .as_matrix will be
removed in a future version. Use .values instead.
  if sys.path[0] == '':
```

[36]:
```
print('X Shape of train data',X_train_s.shape, 'Y shape', Y_train_s.shape)
print('X Shape of val data',X_val_s.shape,'Y shape',Y_val_s.shape)
```

```
X Shape of train data (4067, 128, 9) Y shape (4067, 3)
X Shape of val data (1560, 128, 9) Y shape (1560, 3)
```

- Model to distinguish Static Activities

[37]:
```
# Clear session
K.clear_session()
# Random seed
```

```
np.random.seed(42)
tf.set_random_seed(42)
# Start session
sess = tf.Session(graph=tf.get_default_graph())
K.set_session(sess)
# Define the model
model = Sequential()
model.add(Conv1D(filters=64, kernel_size=7,␣
 ↪activation='relu',kernel_initializer='he_uniform',input_shape=(128,9)))
model.add(Conv1D(filters=32, kernel_size=3,␣
 ↪activation='relu',kernel_initializer='he_uniform'))
model.add(Dropout(0.6))
model.add(MaxPooling1D(pool_size=3))
model.add(Flatten())
model.add(Dense(30, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.summary()
```

```
WARNING:tensorflow:Large dropout rate: 0.6 (>0.5). In TensorFlow 2.x, dropout()
uses dropout rate instead of keep_prob. Please ensure that this is intended.
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 122, 64)           4096
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 120, 32)           6176
_________________________________________________________________
dropout_1 (Dropout)          (None, 120, 32)           0
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 40, 32)            0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1280)              0
_________________________________________________________________
dense_1 (Dense)              (None, 30)                38430
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 93
=================================================================
Total params: 48,795
Trainable params: 48,795
Non-trainable params: 0
_________________________________________________________________
```

[38]:
```
import math
adam = keras.optimizers.Adam(lr=0.004)
model.compile(loss='categorical_crossentropy', optimizer=adam,␣
 ↪metrics=['accuracy'])
```

```
model.fit(X_train_s,Y_train_s, epochs=20,␣
 ↪batch_size=32,validation_data=(X_val_s, Y_val_s), verbose=1)
# K.clear_session()
```

```
Train on 4067 samples, validate on 1560 samples
Epoch 1/20
4067/4067 [==============================] - 1s 356us/step - loss: 0.3909 -
accuracy: 0.8645 - val_loss: 0.3511 - val_accuracy: 0.8756
Epoch 2/20
4067/4067 [==============================] - 1s 313us/step - loss: 0.2796 -
accuracy: 0.9073 - val_loss: 0.3104 - val_accuracy: 0.8821
Epoch 3/20
4067/4067 [==============================] - 1s 318us/step - loss: 0.1880 -
accuracy: 0.9230 - val_loss: 0.3504 - val_accuracy: 0.8737
Epoch 4/20
4067/4067 [==============================] - 1s 306us/step - loss: 0.1893 -
accuracy: 0.9270 - val_loss: 0.4072 - val_accuracy: 0.8795
Epoch 5/20
4067/4067 [==============================] - 1s 314us/step - loss: 0.1932 -
accuracy: 0.9272 - val_loss: 0.3510 - val_accuracy: 0.8942
Epoch 6/20
4067/4067 [==============================] - 1s 312us/step - loss: 0.1570 -
accuracy: 0.9400 - val_loss: 0.2834 - val_accuracy: 0.9205
Epoch 7/20
4067/4067 [==============================] - 1s 321us/step - loss: 0.1536 -
accuracy: 0.9417 - val_loss: 0.3581 - val_accuracy: 0.8923
Epoch 8/20
4067/4067 [==============================] - 1s 318us/step - loss: 0.1305 -
accuracy: 0.9486 - val_loss: 0.3250 - val_accuracy: 0.9135
Epoch 9/20
4067/4067 [==============================] - 1s 309us/step - loss: 0.1279 -
accuracy: 0.9508 - val_loss: 0.3133 - val_accuracy: 0.9224
Epoch 10/20
4067/4067 [==============================] - 1s 314us/step - loss: 0.1187 -
accuracy: 0.9543 - val_loss: 0.3198 - val_accuracy: 0.8846
Epoch 11/20
4067/4067 [==============================] - 1s 309us/step - loss: 0.1281 -
accuracy: 0.9511 - val_loss: 0.2427 - val_accuracy: 0.9038
Epoch 12/20
4067/4067 [==============================] - 1s 306us/step - loss: 0.2054 -
accuracy: 0.9430 - val_loss: 0.2403 - val_accuracy: 0.9071
Epoch 13/20
4067/4067 [==============================] - 1s 313us/step - loss: 0.1181 -
accuracy: 0.9498 - val_loss: 0.1966 - val_accuracy: 0.9397
Epoch 14/20
4067/4067 [==============================] - 1s 314us/step - loss: 0.1308 -
accuracy: 0.9543 - val_loss: 0.1835 - val_accuracy: 0.9462
```

```
Epoch 15/20
4067/4067 [==============================] - 1s 317us/step - loss: 0.1231 -
accuracy: 0.9609 - val_loss: 0.2575 - val_accuracy: 0.9288
Epoch 16/20
4067/4067 [==============================] - 1s 310us/step - loss: 0.1269 -
accuracy: 0.9471 - val_loss: 0.2157 - val_accuracy: 0.9378
Epoch 17/20
4067/4067 [==============================] - 1s 312us/step - loss: 0.1078 -
accuracy: 0.9570 - val_loss: 0.1923 - val_accuracy: 0.9391
Epoch 18/20
4067/4067 [==============================] - 1s 309us/step - loss: 0.1097 -
accuracy: 0.9567 - val_loss: 0.2281 - val_accuracy: 0.9276
Epoch 19/20
4067/4067 [==============================] - 1s 315us/step - loss: 0.1035 -
accuracy: 0.9570 - val_loss: 0.2457 - val_accuracy: 0.9160
Epoch 20/20
4067/4067 [==============================] - 1s 304us/step - loss: 0.0795 -
accuracy: 0.9680 - val_loss: 0.2568 - val_accuracy: 0.9231
```

[38]:
```
<keras.callbacks.callbacks.History at 0x6342bbdd8>
```

[39]:
```
_,acc_val = model.evaluate(X_val_s, Y_val_s,verbose=0)
_,acc_train = model.evaluate(X_train_s,Y_train_s,verbose=0)
print('Train_accuracy',acc_train,'test_accuracy',acc_val)
```

```
Train_accuracy 0.9773789048194885 test_accuracy 0.9230769276618958
```

[40]:
```
##saving model
model.save('final_model_static.h5')
```

[41]:
```
# clear tf session
K.clear_session()
```

- Simple model gives us approx. 95% accuracy for classifying Static Activities

1.4.4.3 Model for Classifying Dynamic Activities

[42]:
```
def load_y_dynamic(subset):
    """
    The objective that we are trying to predict is a integer, from 1 to 6,
    that represents a human activity. We return a binary representation of
    every sample objective as a 6 bits vector using One Hot Encoding
    (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.
 ↪html)
    """
    filename = f'UCI_HAR_Dataset/{subset}/y_{subset}.txt'
    y = _read_csv(filename)[0]
    y_subset = y<=3
    y = y[y_subset]
```

```
    return pd.get_dummies(y).as_matrix(),y_subset

def load_data_dynamic():
    '''
    Load train, test data and scale the data as well
    '''
    Y_train_d, y_train_sub = load_y_dynamic('train')
    Y_val_d, y_test_sub = load_y_dynamic('test')

    X_train_d, X_val_d = load_signals('train'), load_signals('test')
    X_train_d = X_train_d[y_train_sub]
    X_val_d = X_val_d[y_test_sub]

    # fit and transform data
    Scale = None
    Scale = fit(X_train_d)
    dump_file('Scale_dynamic.p','wb', Scale)
    X_train_d = transform(X_train_d, Scale)
    X_val_d = transform(X_val_d, Scale)

    return X_train_d, Y_train_d, X_val_d, Y_val_d
```

[43]:
```
X_train_d, Y_train_d, X_val_d,  Y_val_d = load_data_dynamic()
```

```
/Users/mayankgupta/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:12: FutureWarning: Method .as_matrix will be
removed in a future version. Use .values instead.
  if sys.path[0] == '':
```

[44]:
```
print('Train X shape',X_train_d.shape,'Test X shape',X_val_d.shape)
print('Train Y shape',Y_train_d.shape,'Test Y shape',Y_val_d.shape)
```

```
Train X shape (3285, 128, 9) Test X shape (1387, 128, 9)
Train Y shape (3285, 3) Test Y shape (1387, 3)
```

- Model to distinguish Dynamic Activities

[45]:
```
# Clear session
K.clear_session();
# Set random seed
np.random.seed(42)
tf.set_random_seed(42)
# Start session
sess = tf.Session(graph=tf.get_default_graph())
K.set_session(sess)
# define model
model = Sequential()
```

```
model.add(Conv1D(filters=64, kernel_size=7,␣
 ↪activation='relu',kernel_initializer='he_uniform',input_shape=(128,9)))
model.add(Conv1D(filters=32, kernel_size=3,␣
 ↪activation='relu',kernel_initializer='he_uniform'))
model.add(Dropout(0.6))
model.add(MaxPooling1D(pool_size=3))
model.add(Flatten())
model.add(Dense(30, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.summary()
```

```
WARNING:tensorflow:Large dropout rate: 0.6 (>0.5). In TensorFlow 2.x, dropout()
uses dropout rate instead of keep_prob. Please ensure that this is intended.
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 122, 64)           4096
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 120, 32)           6176
_________________________________________________________________
dropout_1 (Dropout)          (None, 120, 32)           0
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 40, 32)            0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1280)              0
_________________________________________________________________
dense_1 (Dense)              (None, 30)                38430
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 93
=================================================================
Total params: 48,795
Trainable params: 48,795
Non-trainable params: 0
_________________________________________________________________
```

[46]:
```
import math
adam = keras.optimizers.Adam(lr=0.004)
model.compile(loss='categorical_crossentropy', optimizer=adam,␣
 ↪metrics=['accuracy'])
model.fit(X_train_d,Y_train_d, epochs=20,␣
 ↪batch_size=32,validation_data=(X_val_d, Y_val_d), verbose=1)
# K.clear_session()
```

```
Train on 3285 samples, validate on 1387 samples
Epoch 1/20
3285/3285 [==============================] - 1s 417us/step - loss: 0.4893 -
```

```
accuracy: 0.7988 - val_loss: 0.2099 - val_accuracy: 0.9279
Epoch 2/20
3285/3285 [==============================] - 1s 350us/step - loss: 0.0319 -
accuracy: 0.9903 - val_loss: 0.1145 - val_accuracy: 0.9517
Epoch 3/20
3285/3285 [==============================] - 1s 351us/step - loss: 0.0036 -
accuracy: 0.9994 - val_loss: 0.1048 - val_accuracy: 0.9690
Epoch 4/20
3285/3285 [==============================] - 1s 341us/step - loss: 0.0122 -
accuracy: 0.9951 - val_loss: 0.1344 - val_accuracy: 0.9495
Epoch 5/20
3285/3285 [==============================] - 1s 339us/step - loss: 0.0189 -
accuracy: 0.9942 - val_loss: 0.0857 - val_accuracy: 0.9683
Epoch 6/20
3285/3285 [==============================] - 1s 336us/step - loss: 0.0310 -
accuracy: 0.9924 - val_loss: 0.1000 - val_accuracy: 0.9726
Epoch 7/20
3285/3285 [==============================] - 1s 346us/step - loss: 0.0090 -
accuracy: 0.9967 - val_loss: 0.0783 - val_accuracy: 0.9697
Epoch 8/20
3285/3285 [==============================] - 1s 409us/step - loss: 3.8390e-04 -
accuracy: 1.0000 - val_loss: 0.0342 - val_accuracy: 0.9906
Epoch 9/20
3285/3285 [==============================] - 1s 426us/step - loss: 2.2239e-04 -
accuracy: 1.0000 - val_loss: 0.0444 - val_accuracy: 0.9856
Epoch 10/20
3285/3285 [==============================] - 1s 420us/step - loss: 2.4868e-04 -
accuracy: 1.0000 - val_loss: 0.0421 - val_accuracy: 0.9856
Epoch 11/20
3285/3285 [==============================] - 1s 414us/step - loss: 0.0395 -
accuracy: 0.9912 - val_loss: 0.2583 - val_accuracy: 0.9308
Epoch 12/20
3285/3285 [==============================] - 2s 460us/step - loss: 0.0376 -
accuracy: 0.9900 - val_loss: 0.1201 - val_accuracy: 0.9704
Epoch 13/20
3285/3285 [==============================] - 1s 446us/step - loss: 0.0151 -
accuracy: 0.9960 - val_loss: 0.0589 - val_accuracy: 0.9791
Epoch 14/20
3285/3285 [==============================] - 1s 374us/step - loss: 0.0115 -
accuracy: 0.9967 - val_loss: 0.0636 - val_accuracy: 0.9776
Epoch 15/20
3285/3285 [==============================] - 1s 354us/step - loss: 0.0121 -
accuracy: 0.9963 - val_loss: 0.0385 - val_accuracy: 0.9841
Epoch 16/20
3285/3285 [==============================] - 1s 342us/step - loss: 0.0182 -
accuracy: 0.9957 - val_loss: 0.0859 - val_accuracy: 0.9740
Epoch 17/20
3285/3285 [==============================] - 1s 341us/step - loss: 0.0087 -
```

```
accuracy: 0.9979 - val_loss: 0.0534 - val_accuracy: 0.9834
Epoch 18/20
3285/3285 [==============================] - 1s 340us/step - loss: 0.0020 -
accuracy: 0.9994 - val_loss: 0.1136 - val_accuracy: 0.9690
Epoch 19/20
3285/3285 [==============================] - 1s 341us/step - loss: 3.3887e-04 -
accuracy: 1.0000 - val_loss: 0.0776 - val_accuracy: 0.9791
Epoch 20/20
3285/3285 [==============================] - 1s 334us/step - loss: 1.4385e-04 -
accuracy: 1.0000 - val_loss: 0.1050 - val_accuracy: 0.9726
```

[46]:
```
<keras.callbacks.callbacks.History at 0x635778668>
```

[47]:
```
_,acc_val = model.evaluate(X_val_d, Y_val_d,verbose=0)
_,acc_train = model.evaluate(X_train_d,Y_train_d,verbose=0)
print('Train_accuracy',acc_train,'test_accuracy',acc_val)
```

```
Train_accuracy 1.0 test_accuracy 0.9726027250289917
```

[48]:
```
##saving model
model.save('final_model_dynamic.h5')
```

[49]:
```
# clear tf session
K.clear_session()
```

1.4.4.4 Load and Split whole data

[50]:
```
def load_y_whole_data(subset):
    """
    The objective that we are trying to predict is a integer, from 1 to 6,
    that represents a human activity. We return a binary representation of
    every sample objective as a 6 bits vector using One Hot Encoding
    (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.
 ↪html)
    """
    filename = f'UCI_HAR_Dataset/{subset}/y_{subset}.txt'
    y = _read_csv(filename)[0]
    return y

def load_whole_data():
    '''
    Load and split whole data
    '''
    X_train, X_val = load_signals('train'), load_signals('test')
    Y_train, Y_val = load_y_whole_data('train'), load_y_whole_data('test')

    return X_train, Y_train, X_val, Y_val
```

[51]:
```
X_train, Y_train, X_val, Y_val = load_whole_data()
```

```
/Users/mayankgupta/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:12: FutureWarning: Method .as_matrix will be
removed in a future version. Use .values instead.
  if sys.path[0] == '':
```

[52]:
```
print('shape of train X',X_train.shape, 'shape of train Y',Y_train.shape)
print('shape of test X', X_val.shape, 'shape of test Y', Y_val.shape)
```

```
shape of train X (7352, 128, 9) shape of train Y (7352,)
shape of test X (2947, 128, 9) shape of test Y (2947,)
```

1.4.4.5 Final prediction pipeline

[53]:
```
##loading keras models and picle files for scaling data
from keras.models import load_model
import pickle
model_2class = load_model('final_model_2class.h5')
model_dynamic = load_model('final_model_dynamic.h5')
model_static = load_model('final_model_static.h5')
scale_2class = pickle.load(open('Scale_2class.p','rb'))
scale_static = pickle.load(open('Scale_static.p','rb'))
scale_dynamic = pickle.load(open('Scale_dynamic.p','rb'))
```

```
WARNING:tensorflow:Large dropout rate: 0.6 (>0.5). In TensorFlow 2.x, dropout()
uses dropout rate instead of keep_prob. Please ensure that this is intended.
WARNING:tensorflow:Large dropout rate: 0.6 (>0.5). In TensorFlow 2.x, dropout()
uses dropout rate instead of keep_prob. Please ensure that this is intended.
```

[54]:
```
##scaling the data
def transform_data(X,scale):
    X_temp = X.reshape((X.shape[0] * X.shape[1], X.shape[2]))
    X_temp = scale.transform(X_temp)
    return X_temp.reshape(X.shape)
```

[55]:
```
#predicting output activity
def predict_activity(X):
    ##predicting whether dynamic or static
    predict_2class = model_2class.predict(transform_data(X,scale_2class))
    Y_pred_2class =  np.argmax(predict_2class, axis=1)
    #static data filter
    X_static = X[Y_pred_2class==1]
    #dynamic data filter
    X_dynamic = X[Y_pred_2class==0]
    #predicting static activities
    predict_static = model_static.predict(transform_data(X_static,scale_static))
    predict_static = np.argmax(predict_static,axis=1)
    #adding 4 because need to get final prediction lable as output
    predict_static = predict_static + 4
```

```
    #predicting dynamic activites
    predict_dynamic = model_dynamic.
↪predict(transform_data(X_dynamic,scale_dynamic))
    predict_dynamic = np.argmax(predict_dynamic,axis=1)
    #adding 1 because need to get final prediction lable as output
    predict_dynamic = predict_dynamic + 1
    ##appending final output to one list in the same sequence of input data
    i,j = 0,0
    final_pred = []
    for mask in Y_pred_2class:
        if mask == 1:
            final_pred.append(predict_static[i])
            i = i + 1
        else:
            final_pred.append(predict_dynamic[j])
            j = j + 1
    return final_pred
```

[56]:
```
##predicting
final_pred_val = predict_activity(X_val)
final_pred_train = predict_activity(X_train)
```

[57]:
```
##accuracy of train and test
from sklearn.metrics import accuracy_score
print('Accuracy of train data',accuracy_score(Y_train,final_pred_train))
print('Accuracy of validation data',accuracy_score(Y_val,final_pred_val))
```

```
Accuracy of train data 0.9874863982589771
Accuracy of validation data 0.9460468272819816
```

[58]:
```
import itertools
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)
```

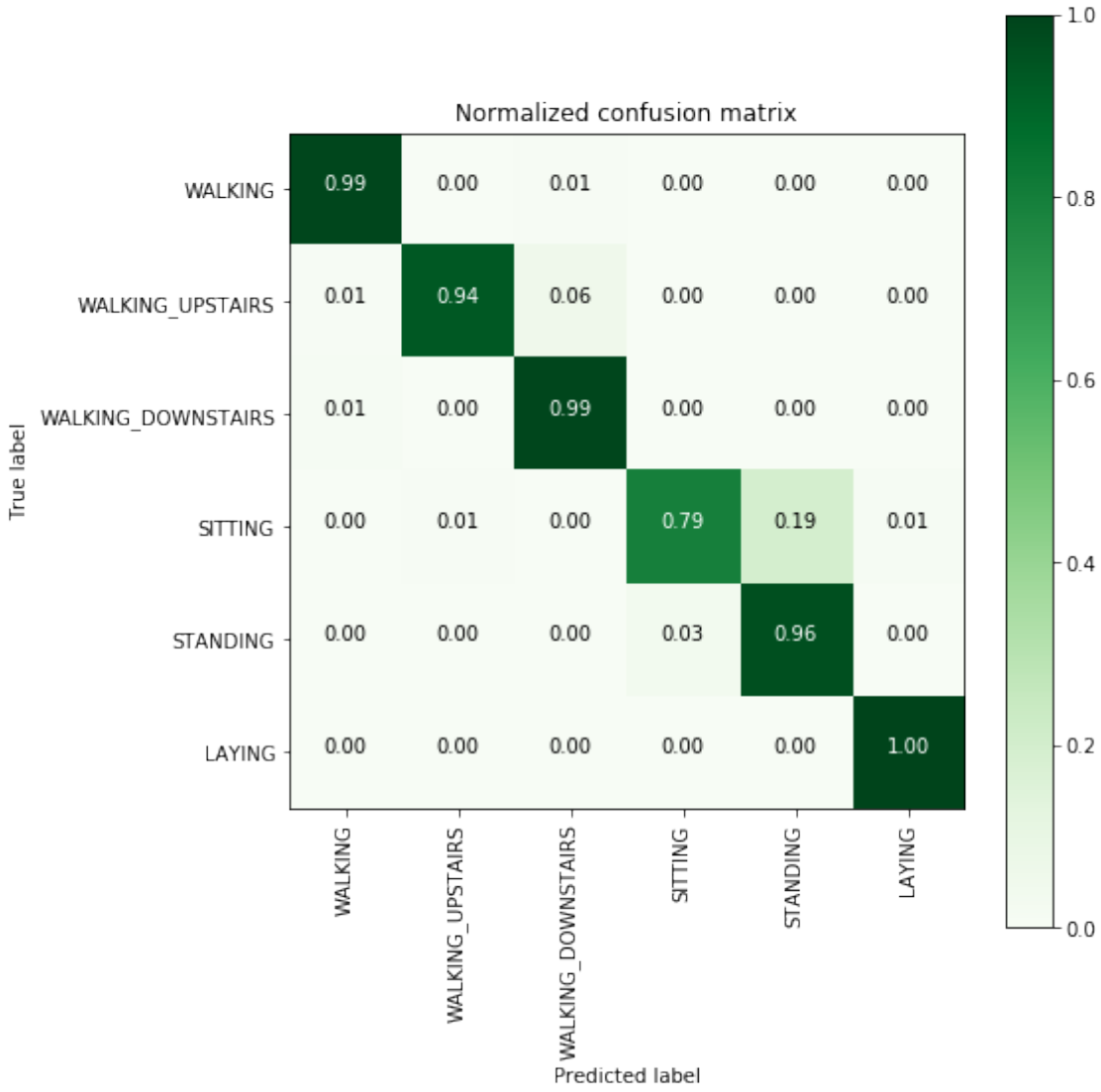
```
    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

[59]:
```
#confusion metric
cm = confusion_matrix(Y_val, final_pred_val,labels=range(1,7))
cm
```

[59]:
```
array([[492,   1,   3,   0,   0,   0],
       [  3, 441,  27,   0,   0,   0],
       [  4,   0, 416,   0,   0,   0],
       [  0,   3,   0, 390,  94,   4],
       [  0,   1,   1,  18, 512,   0],
       [  0,   0,   0,   0,   0, 537]])
```

[60]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(8,8))
labels=['WALKING','WALKING_UPSTAIRS','WALKING_DOWNSTAIRS','SITTING','STANDING','LAYING']
plot_confusion_matrix(cm, classes=labels,
                      normalize=True, title='Normalized confusion matrix', cmap␣
 ↪= plt.cm.Greens)
plt.show()
```



1.5 Now work with RAW features only, in your HAR. And output of the lstms should be sent to conv layers and then apply some dense layers and finally a soft max layer.

[62]:
```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 32
```

[23]:
```
timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = _count_classes(Y_train)

print(timesteps)
print(input_dim)
print(len(X_train))
```

```
128
9
7352
```

[27]:
```
from keras.layers import Flatten
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
```

[38]:
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
# 1st use LSTM and then output of LSTM to  CONV layers
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim),␣
 ↪return_sequences=True))
model.add(Conv1D(filters=32, kernel_size=3,␣
 ↪activation='relu',kernel_initializer='he_uniform'))
model.add(Conv1D(filters=32, kernel_size=3,␣
 ↪activation='relu',kernel_initializer='he_uniform'))
# Adding a dropout layer
model.add(Dropout(0.6))
# Add maxpooling and flatten layer
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
# Adding a dense output layer with sigmoid activation
# Output of  conv layers  to dense layer, output of dense layer to softmax␣
 ↪layer
model.add(Dense(50, activation='relu'))
model.add(Dense(6, activation='softmax'))
model.summary()
```

```
WARNING:tensorflow:Large dropout rate: 0.6 (>0.5). In TensorFlow 2.x, dropout()
uses dropout rate instead of keep_prob. Please ensure that this is intended.
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_9 (LSTM)                (None, 128, 32)           5376
_________________________________________________________________
conv1d_8 (Conv1D)            (None, 126, 32)           3104
_________________________________________________________________
conv1d_9 (Conv1D)            (None, 124, 32)           3104
_________________________________________________________________
dropout_5 (Dropout)          (None, 124, 32)           0
_________________________________________________________________
max_pooling1d_4 (MaxPooling1 (None, 62, 32)            0
_________________________________________________________________
flatten_4 (Flatten)          (None, 1984)              0
_________________________________________________________________
```

```
dense_8 (Dense)              (None, 50)                99250
_________________________________________________________________
dense_9 (Dense)              (None, 6)                 306
=================================================================
Total params: 111,140
Trainable params: 111,140
Non-trainable params: 0
_________________________________________________________________
```

[39]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[40]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.3692 -
accuracy: 0.8501 - val_loss: 0.3845 - val_accuracy: 0.8931
Epoch 2/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1585 -
accuracy: 0.9357 - val_loss: 0.5315 - val_accuracy: 0.8826
Epoch 3/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1303 -
accuracy: 0.9471 - val_loss: 0.6576 - val_accuracy: 0.9013
Epoch 4/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1301 -
accuracy: 0.9486 - val_loss: 0.6384 - val_accuracy: 0.9077
Epoch 5/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1116 -
accuracy: 0.9563 - val_loss: 0.5395 - val_accuracy: 0.9026
Epoch 6/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.1127 -
accuracy: 0.9535 - val_loss: 0.8545 - val_accuracy: 0.8999
Epoch 7/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1003 -
accuracy: 0.9587 - val_loss: 0.6561 - val_accuracy: 0.8968
Epoch 8/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1031 -
accuracy: 0.9600 - val_loss: 0.7699 - val_accuracy: 0.8918
Epoch 9/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1153 -
```

```
accuracy: 0.9589 - val_loss: 0.6547 - val_accuracy: 0.9084
Epoch 10/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0914 -
accuracy: 0.9612 - val_loss: 0.6956 - val_accuracy: 0.9046
Epoch 11/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1038 -
accuracy: 0.9596 - val_loss: 0.6759 - val_accuracy: 0.9057
Epoch 12/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0976 -
accuracy: 0.9627 - val_loss: 0.6510 - val_accuracy: 0.8938
Epoch 13/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0955 -
accuracy: 0.9619 - val_loss: 0.8151 - val_accuracy: 0.9026
Epoch 14/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0965 -
accuracy: 0.9649 - val_loss: 0.5930 - val_accuracy: 0.9009
Epoch 15/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0946 -
accuracy: 0.9644 - val_loss: 0.6105 - val_accuracy: 0.9009
Epoch 16/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0953 -
accuracy: 0.9637 - val_loss: 0.6110 - val_accuracy: 0.8928
Epoch 17/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0847 -
accuracy: 0.9645 - val_loss: 0.6800 - val_accuracy: 0.8877
Epoch 18/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.0900 -
accuracy: 0.9674 - val_loss: 0.5505 - val_accuracy: 0.8901
Epoch 19/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0926 -
accuracy: 0.9693 - val_loss: 0.6652 - val_accuracy: 0.9043
Epoch 20/30
7352/7352 [==============================] - 18s 3ms/step - loss: 0.1022 -
accuracy: 0.9709 - val_loss: 1.0711 - val_accuracy: 0.9026
Epoch 21/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.0894 -
accuracy: 0.9671 - val_loss: 0.7108 - val_accuracy: 0.8931
Epoch 22/30
7352/7352 [==============================] - 18s 3ms/step - loss: 0.0818 -
accuracy: 0.9716 - val_loss: 0.6325 - val_accuracy: 0.9050
Epoch 23/30
7352/7352 [==============================] - 18s 3ms/step - loss: 0.0937 -
accuracy: 0.9697 - val_loss: 0.4771 - val_accuracy: 0.9189
Epoch 24/30
7352/7352 [==============================] - 21s 3ms/step - loss: 0.0961 -
accuracy: 0.9724 - val_loss: 0.4298 - val_accuracy: 0.9223
Epoch 25/30
7352/7352 [==============================] - 26s 3ms/step - loss: 0.0824 -
```

```
accuracy: 0.9733 - val_loss: 0.7755 - val_accuracy: 0.9019
Epoch 26/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.0916 -
accuracy: 0.9718 - val_loss: 0.6355 - val_accuracy: 0.9206
Epoch 27/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.0896 -
accuracy: 0.9725 - val_loss: 0.4728 - val_accuracy: 0.9111
Epoch 28/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0821 -
accuracy: 0.9748 - val_loss: 0.6026 - val_accuracy: 0.9158
Epoch 29/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0677 -
accuracy: 0.9744 - val_loss: 0.7449 - val_accuracy: 0.9172
Epoch 30/30
7352/7352 [==============================] - 18s 3ms/step - loss: 0.0782 -
accuracy: 0.9748 - val_loss: 0.5481 - val_accuracy: 0.9182
```

[40]:
```
<keras.callbacks.callbacks.History at 0x63c5395c0>
```

[41]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 537        0         0        0                   0
SITTING                  5      372       114        0                   0
STANDING                 0       60       469        3                   0
WALKING                  0        0         1      489                   0
WALKING_DOWNSTAIRS       0        0         0        1                 389
WALKING_UPSTAIRS         0        0         0       19                   2

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            0
STANDING                           0
WALKING                            6
WALKING_DOWNSTAIRS                30
WALKING_UPSTAIRS                 450
```

[42]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 1s 290us/step
```

[43]:
```
score
```

[43]:
```
[0.5481175343919985, 0.9182218909263611]
```

Decrease dropout and tried other variations

[69]:
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
# 1st use LSTM and then output of LSTM to  CONV layers
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim),␣
 ↪return_sequences=True))
model.add(Conv1D(filters=32, kernel_size=3,␣
 ↪activation='relu',kernel_initializer='he_uniform'))
model.add(Conv1D(filters=32, kernel_size=3,␣
 ↪activation='relu',kernel_initializer='he_uniform'))
# Adding a dropout layer
model.add(Dropout(0.5))
# Add maxpooling and flatten layer
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
# Adding a dense output layer with sigmoid activation
# Output of  conv layers  to dense layer, output of dense layer to softmax␣
 ↪layer
model.add(Dense(50, activation='relu'))
model.add(Dense(6, activation='softmax'))
model.summary()
```

```
Model: "sequential_15"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_15 (LSTM)               (None, 128, 32)           5376
_________________________________________________________________
conv1d_24 (Conv1D)           (None, 126, 32)           3104
_________________________________________________________________
conv1d_25 (Conv1D)           (None, 124, 32)           3104
_________________________________________________________________
dropout_13 (Dropout)         (None, 124, 32)           0
_________________________________________________________________
max_pooling1d_10 (MaxPooling (None, 62, 32)            0
_________________________________________________________________
flatten_10 (Flatten)         (None, 1984)              0
_________________________________________________________________
dense_20 (Dense)             (None, 50)                99250
_________________________________________________________________
dense_21 (Dense)             (None, 6)                 306
=================================================================
Total params: 111,140
Trainable params: 111,140
Non-trainable params: 0
_________________________________________________________________
```

[70]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[71]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.4346 -
accuracy: 0.8316 - val_loss: 0.4471 - val_accuracy: 0.8867
Epoch 2/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1599 -
accuracy: 0.9363 - val_loss: 0.3654 - val_accuracy: 0.9019
Epoch 3/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1304 -
accuracy: 0.9483 - val_loss: 0.4729 - val_accuracy: 0.9036
Epoch 4/30
7352/7352 [==============================] - 18s 3ms/step - loss: 0.1197 -
accuracy: 0.9501 - val_loss: 0.4634 - val_accuracy: 0.9155
Epoch 5/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1152 -
accuracy: 0.9513 - val_loss: 0.5189 - val_accuracy: 0.9298
Epoch 6/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1043 -
accuracy: 0.9558 - val_loss: 0.5348 - val_accuracy: 0.9291
Epoch 7/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1103 -
accuracy: 0.9591 - val_loss: 0.6338 - val_accuracy: 0.9063
Epoch 8/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0963 -
accuracy: 0.9585 - val_loss: 0.6357 - val_accuracy: 0.9213
Epoch 9/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1077 -
accuracy: 0.9574 - val_loss: 0.8853 - val_accuracy: 0.8992
Epoch 10/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1017 -
accuracy: 0.9601 - val_loss: 0.6541 - val_accuracy: 0.9077
Epoch 11/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0981 -
accuracy: 0.9604 - val_loss: 0.6410 - val_accuracy: 0.9046
Epoch 12/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0994 -
```

```
accuracy: 0.9633 - val_loss: 0.6200 - val_accuracy: 0.9121
Epoch 13/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1157 -
accuracy: 0.9641 - val_loss: 0.8546 - val_accuracy: 0.9131
Epoch 14/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.0970 -
accuracy: 0.9621 - val_loss: 0.7290 - val_accuracy: 0.9033
Epoch 15/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0999 -
accuracy: 0.9671 - val_loss: 0.9121 - val_accuracy: 0.9006
Epoch 16/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0863 -
accuracy: 0.9648 - val_loss: 1.0077 - val_accuracy: 0.8897
Epoch 17/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1005 -
accuracy: 0.9640 - val_loss: 1.0764 - val_accuracy: 0.8924
Epoch 18/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0948 -
accuracy: 0.9649 - val_loss: 1.0773 - val_accuracy: 0.9030
Epoch 19/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0837 -
accuracy: 0.9664 - val_loss: 0.9049 - val_accuracy: 0.9131
Epoch 20/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1002 -
accuracy: 0.9665 - val_loss: 0.7059 - val_accuracy: 0.9277
Epoch 21/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0901 -
accuracy: 0.9693 - val_loss: 0.7491 - val_accuracy: 0.9104
Epoch 22/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0795 -
accuracy: 0.9697 - val_loss: 0.8270 - val_accuracy: 0.9111
Epoch 23/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0827 -
accuracy: 0.9701 - val_loss: 1.0694 - val_accuracy: 0.9108
Epoch 24/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0727 -
accuracy: 0.9712 - val_loss: 0.8130 - val_accuracy: 0.9264
Epoch 25/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0918 -
accuracy: 0.9725 - val_loss: 0.9463 - val_accuracy: 0.9158
Epoch 26/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0777 -
accuracy: 0.9721 - val_loss: 1.1119 - val_accuracy: 0.9125
Epoch 27/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0759 -
accuracy: 0.9706 - val_loss: 1.3036 - val_accuracy: 0.9019
Epoch 28/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0755 -
```

```
accuracy: 0.9744 - val_loss: 0.9760 - val_accuracy: 0.9250
Epoch 29/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1036 -
accuracy: 0.9739 - val_loss: 1.0356 - val_accuracy: 0.9087
Epoch 30/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0705 -
accuracy: 0.9757 - val_loss: 1.0847 - val_accuracy: 0.9206
```

[71]:
```
<keras.callbacks.callbacks.History at 0x6458fcb70>
```

[72]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 1s 414us/step
```

[73]:
```
score
```

[73]:
```
[1.08474476597534, 0.9205971956253052]
```

LSTM with Conv1D slightly increase performance.
Add two layer dropout

[63]:
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
# 1st use LSTM and then output of LSTM to  CONV layers
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim),␣
 ↪return_sequences=True))
model.add(Conv1D(filters=32, kernel_size=3,␣
 ↪activation='relu',kernel_initializer='he_uniform'))
# Adding a dropout layer
model.add(Dropout(0.5))
model.add(Conv1D(filters=32, kernel_size=3,␣
 ↪activation='relu',kernel_initializer='he_uniform'))
# Adding a dropout layer
model.add(Dropout(0.5))
# Add maxpooling and flatten layer
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
# Adding a dense output layer with sigmoid activation
# Output of  conv layers  to dense layer, output of dense layer to softmax␣
 ↪layer
model.add(Dense(50, activation='relu'))
model.add(Dense(6, activation='softmax'))
model.summary()
```

```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
```

```
lstm_14 (LSTM)               (None, 128, 32)           5376
_________________________________________________________________
conv1d_22 (Conv1D)           (None, 126, 32)           3104
_________________________________________________________________
dropout_11 (Dropout)         (None, 126, 32)           0
_________________________________________________________________
conv1d_23 (Conv1D)           (None, 124, 32)           3104
_________________________________________________________________
dropout_12 (Dropout)         (None, 124, 32)           0
_________________________________________________________________
max_pooling1d_9 (MaxPooling1 (None, 62, 32)            0
_________________________________________________________________
flatten_9 (Flatten)          (None, 1984)              0
_________________________________________________________________
dense_18 (Dense)             (None, 50)                99250
_________________________________________________________________
dense_19 (Dense)             (None, 6)                 306
=================================================================
Total params: 111,140
Trainable params: 111,140
Non-trainable params: 0
_________________________________________________________________
```

[64]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[65]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.5253 -
accuracy: 0.7854 - val_loss: 0.5624 - val_accuracy: 0.8208
Epoch 2/30
7352/7352 [==============================] - 18s 3ms/step - loss: 0.1975 -
accuracy: 0.9232 - val_loss: 0.5013 - val_accuracy: 0.8914
Epoch 3/30
7352/7352 [==============================] - 16s 2ms/step - loss: 0.1451 -
accuracy: 0.9423 - val_loss: 0.5495 - val_accuracy: 0.8677
Epoch 4/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1338 -
accuracy: 0.9472 - val_loss: 0.3797 - val_accuracy: 0.9030
```

```
Epoch 5/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1304 -
accuracy: 0.9478 - val_loss: 0.4006 - val_accuracy: 0.8877
Epoch 6/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1229 -
accuracy: 0.9484 - val_loss: 0.3259 - val_accuracy: 0.9108
Epoch 7/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1191 -
accuracy: 0.9528 - val_loss: 0.3226 - val_accuracy: 0.9223
Epoch 8/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1149 -
accuracy: 0.9529 - val_loss: 0.3726 - val_accuracy: 0.9043
Epoch 9/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1154 -
accuracy: 0.9543 - val_loss: 0.3421 - val_accuracy: 0.9128
Epoch 10/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1055 -
accuracy: 0.9585 - val_loss: 0.5777 - val_accuracy: 0.9108
Epoch 11/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1067 -
accuracy: 0.9581 - val_loss: 0.9684 - val_accuracy: 0.8965
Epoch 12/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1140 -
accuracy: 0.9581 - val_loss: 0.5287 - val_accuracy: 0.8958
Epoch 13/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1095 -
accuracy: 0.9604 - val_loss: 0.4963 - val_accuracy: 0.9040
Epoch 14/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1032 -
accuracy: 0.9595 - val_loss: 0.5505 - val_accuracy: 0.9125
Epoch 15/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1031 -
accuracy: 0.9607 - val_loss: 0.6846 - val_accuracy: 0.9155
Epoch 16/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1036 -
accuracy: 0.9607 - val_loss: 0.7777 - val_accuracy: 0.9067
Epoch 17/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0914 -
accuracy: 0.9635 - val_loss: 1.1068 - val_accuracy: 0.9009
Epoch 18/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1009 -
accuracy: 0.9611 - val_loss: 1.8117 - val_accuracy: 0.9050
Epoch 19/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1219 -
accuracy: 0.9630 - val_loss: 0.9199 - val_accuracy: 0.9043
Epoch 20/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0938 -
accuracy: 0.9641 - val_loss: 1.2955 - val_accuracy: 0.8904
```

```
Epoch 21/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.0924 -
accuracy: 0.9676 - val_loss: 1.3183 - val_accuracy: 0.8721
Epoch 22/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1005 -
accuracy: 0.9660 - val_loss: 1.2037 - val_accuracy: 0.8850
Epoch 23/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.0913 -
accuracy: 0.9656 - val_loss: 1.1553 - val_accuracy: 0.8904
Epoch 24/30
7352/7352 [==============================] - 18s 3ms/step - loss: 0.0860 -
accuracy: 0.9661 - val_loss: 1.2973 - val_accuracy: 0.9101
Epoch 25/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1043 -
accuracy: 0.9660 - val_loss: 1.2217 - val_accuracy: 0.8948
Epoch 26/30
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1022 -
accuracy: 0.9679 - val_loss: 1.5939 - val_accuracy: 0.8914
Epoch 27/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.0987 -
accuracy: 0.9690 - val_loss: 1.0499 - val_accuracy: 0.8904
Epoch 28/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.0944 -
accuracy: 0.9669 - val_loss: 0.8462 - val_accuracy: 0.8901
Epoch 29/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.0910 -
accuracy: 0.9689 - val_loss: 0.7629 - val_accuracy: 0.8935
Epoch 30/30
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0978 -
accuracy: 0.9660 - val_loss: 0.7633 - val_accuracy: 0.8955
```

[65]:
```
<keras.callbacks.callbacks.History at 0x64453d9e8>
```

[66]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 510        0         0        0                   0
SITTING                  6      387        95        0                   0
STANDING                 0       77       452        0                   1
WALKING                  0        0         1      436                  30
WALKING_DOWNSTAIRS       0        0         0        0                 418
WALKING_UPSTAIRS         0        0         0        8                  27

Pred                WALKING_UPSTAIRS
True
LAYING                            27
```

```
SITTING                          3
STANDING                         2
WALKING                         29
WALKING_DOWNSTAIRS               2
WALKING_UPSTAIRS               436
```

[67]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 2s 742us/step
```

[68]:
```
score
```

[68]:
```
[0.7633403087432127, 0.8954869508743286]
```

LSTM with Conv1D and 2 layer dropout doesn't increase performance much.

2. Conclusion

2.1 Models Output

[75]:
```
from prettytable import PrettyTable

table = PrettyTable()
table.field_names = ['LSTM Layer', 'Dropout', 'Loss', 'Accuracy']

table.add_row([32, 0.5, 0.41655886096877154, 0.8954869508743286])
table.add_row([64, 0.5, 0.35375094639873494, 0.9148286581039429])
table.add_row([64, 0.7, 'nan', 0.16830675303936005])
table.add_row([32, 0.7, 0.596762544016313, 0.8381404876708984])
table.add_row([64, 0.6, 0.4956902541945778, 0.9060060977935791])
table.add_row([32, 0.6, 0.5602035771178105, 0.8812351822853088])
table.add_row(['64+32', '2*0.7', 'nan', 0.16830675303936005])
table.add_row(['64+32', '2*0.6', 'nan', 0.16830675303936005])
table.add_row(['64+32', '2*0.5', 0.6456601167650766, 0.900237500667572])
table.add_row(['64+32', 0.7, 'nan', 0.16830675303936005])
table.add_row(['64+32', 0.6, 0.5375813010956876, 0.9053274393081665])
table.add_row(['64+32', 0.5, 0.3804781971405892, 0.9141499996185303])
table.add_row(['64+16', 0.5, 0.5414807511955393, 0.9056667685508728])
table.add_row(['32+16', 0.5, 0.3589311393391746, 0.922633171081543])
table.add_row(['32+8', 0.5, 0.3761567989555303, 0.7539871335029602])
table.add_row(['32+8', 0.5, 0.3761567989555303, 0.7539871335029602])
table.add_row(['32+8', 0.5, 0.3761567989555303, 0.7539871335029602])
table.add_row(['32+16', 0.6, 0.36157595883878874, 0.9100780487060547])
table.add_row(['32+16', 0.4, 0.39125852214684165, 0.9066847562789917])
table.add_row(['128', 0.25, 0.31214076887265046, 0.9256871342658997])
table.add_row(['Divide and Conquer', 'N/A', '-', 0.9460468272819816])

print(table)
```

LSTM Layer	Dropout	Loss	Accuracy
32	0.5	0.41655886096877154	0.8954869508743286
64	0.5	0.35375094639873494	0.9148286581039429
64	0.7	nan	0.16830675303936005
32	0.7	0.596762544016313	0.8381404876708984
64	0.6	0.4956902541945778	0.9060060977935791
32	0.6	0.5602035771178105	0.8812351822853088
64+32	2*0.7	nan	0.16830675303936005
64+32	2*0.6	nan	0.16830675303936005
64+32	2*0.5	0.6456601167650766	0.900237500667572
64+32	0.7	nan	0.16830675303936005
64+32	0.6	0.5375813010956876	0.9053274393081665
64+32	0.5	0.3804781971405892	0.9141499996185303
64+16	0.5	0.5414807511955393	0.9056667685508728
32+16	0.5	0.3589311393391746	0.922633171081543
32+8	0.5	0.3761567989555303	0.7539871335029602
32+8	0.5	0.3761567989555303	0.7539871335029602
32+8	0.5	0.3761567989555303	0.7539871335029602
32+16	0.6	0.36157595883878874	0.9100780487060547
32+16	0.4	0.39125852214684165	0.9066847562789917
128	0.25	0.31214076887265046	0.9256871342658997
Divide and Conquer	N/A	-	0.9460468272819816

2.2 Steps I followed

- Did EDA on dataset
- Run classical machine learning models on 561 handcrafted features
- Run different LSTMs on raw time series data
- Tried 2 LSTMs with larger dropout, increase layer of LSTM from 32 to 64
- Above table contains different permutation and combination of LSTM that i tried
- Implement Divide and Conquer technique for classifying Static and Dynamic Activities and Merge Models for Activities Prediction

2.3 Divide and Conquer

- First classify static and dynamic activities, if label > 3 then static else dynamic
- Classify static activities (Sitting, Standing and Laying)
- Classify dynamic activities (Walking, Walking Upstairs and Walking Downstairs)
- Classify whole data on above three models

2.4 Best Model

- Divide and Conquer gives me 94.60% accuracy

[ ]: