```
# Importing Libraries
```

```
import pandas as pd
import numpy as np
```

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947

Enter your authorization code:
..........
Mounted at /content/drive
```

```
import joblib
X_train = joblib.load('/content/drive/My Drive/Colab Notebooks/Assignment 21/X_train.pkl')
X_test = joblib.load('/content/drive/My Drive/Colab Notebooks/Assignment 21/X_test.pkl')
Y_train = joblib.load('/content/drive/My Drive/Colab Notebooks/Assignment 21/Y_train.pkl')
Y_test  = joblib.load('/content/drive/My Drive/Colab Notebooks/Assignment 21/Y_test.pkl')
```

```
# 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'])
```

```
import matplotlib.pyplot as plt
import numpy as np
import time
# https://gist.github.com/greydanus/f6eee59eaf1d90fcb3b534a25362cea4
# https://stackoverflow.com/a/14434334
# this function is used to update the plots for each epoch and error
def plt_dynamic(x, vy, ty, ax, colors=['b']):
    ax.plot(x, vy, 'b', label="Validation Loss")
    ax.plot(x, ty, 'r', label="Train Loss")
    plt.legend()
    plt.grid()
    fig.canvas.draw()
```

Data

```
# 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"
]
```

```
# Importing tensorflow
np.random.seed(42)
import tensorflow as tf
tf.set_random_seed(42)
```

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

```
# 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.
```

```
# Importing libraries
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers.core import Dense, Dropout
```

```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 32
```

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

```
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

- Defining the Architecture of LSTM

```
# 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()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/fram
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob
_________________________________________________________________
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
_________________________________________________________________
```

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

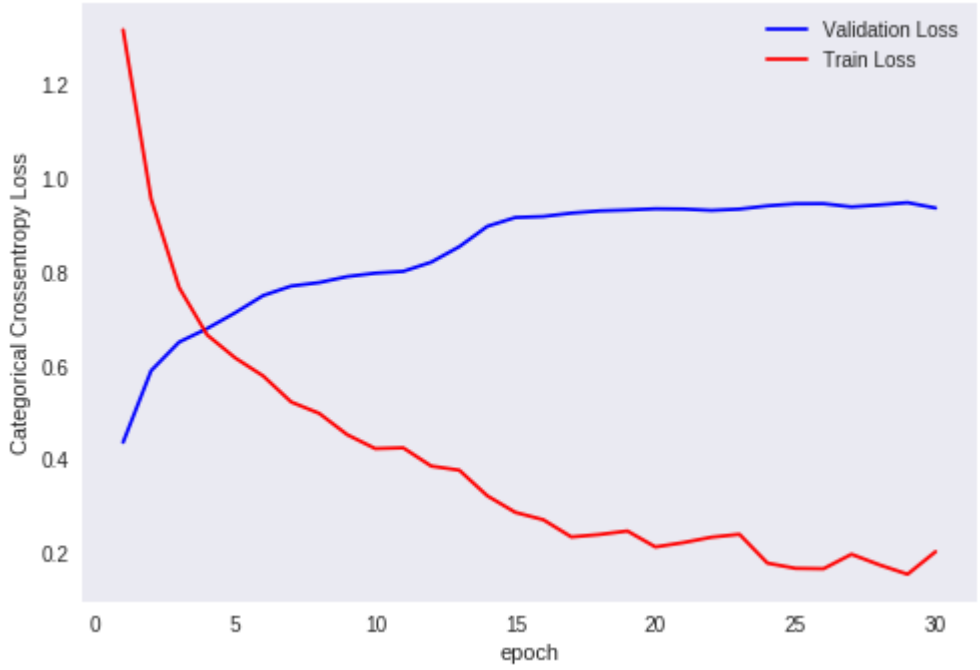
```
# Training the model
history = model.fit(X_train,
                    Y_train,
                    batch_size=batch_size,
                    validation_data=(X_test, Y_test),
                    epochs=epochs)
```

```
Epoch 2/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.9571 - acc: 0.589
Epoch 3/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.7675 - acc: 0.651
Epoch 4/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.6663 - acc: 0.680
Epoch 5/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.6172 - acc: 0.713
Epoch 6/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.5785 - acc: 0.750
Epoch 7/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.5228 - acc: 0.770
Epoch 8/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.4985 - acc: 0.778
Epoch 9/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.4529 - acc: 0.791
Epoch 10/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.4232 - acc: 0.798
Epoch 11/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.4247 - acc: 0.802
Epoch 12/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.3857 - acc: 0.822
Epoch 13/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.3770 - acc: 0.855
Epoch 14/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.3221 - acc: 0.898
Epoch 15/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2868 - acc: 0.917
Epoch 16/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2709 - acc: 0.919
Epoch 17/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2342 - acc: 0.926
Epoch 18/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2394 - acc: 0.931
Epoch 19/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2469 - acc: 0.933
Epoch 20/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.2128 - acc: 0.936
Epoch 21/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2221 - acc: 0.935
Epoch 22/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2336 - acc: 0.932
Epoch 23/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2399 - acc: 0.935
Epoch 24/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1782 - acc: 0.942
Epoch 25/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1670 - acc: 0.946
```

```
# Final evaluation of the model
score = model.evaluate(X_test, Y_test)
score_1_2 = score[0]
score_2_2 = score[1]
print('Test score:', score_1_2)
print('Test accuracy:', score_2_2)

print(history.history.keys())

fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
```

```
# list of epoch numbers
x = list(range(1,epochs+1))

vy = history.history['acc']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
2947/2947 [==============================] - 1s 370us/step
Test score: 0.3421370585869036
Test accuracy: 0.9063454360366474
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```




```
# 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      375       113        0                   0
STANDING                 0       54       474        4                   0
WALKING                  0        8        10      467                   2
WALKING_DOWNSTAIRS       0        0         0        0                 408
WALKING_UPSTAIRS         0       12         7        5                  10

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            1
STANDING                           0
WALKING                            9
WALKING_DOWNSTAIRS                12
WALKING_UPSTAIRS                 437
```

- With a simple 2 layer architecture we got 90.63% accuracy and a loss of 0.34
- We can further imporve the performace with Hyperparameter tuning

Hyperparameter tunning: case 1

```
# Initializing parameters
epochs = 20
batch_size = 24
n_hidden = 48


# 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(0.75))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_3 (LSTM)                (None, 48)                11136     
_________________________________________________________________
dropout_3 (Dropout)          (None, 48)                0         
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 294       
=================================================================
Total params: 11,430
Trainable params: 11,430
Non-trainable params: 0
_________________________________________________________________
```

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


# Training the model
history = 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 1/20
7352/7352 [==============================] - 22s 3ms/step - loss: 1.3759 - acc: 0.419
Epoch 2/20
7352/7352 [==============================] - 21s 3ms/step - loss: 1.0504 - acc: 0.542
Epoch 3/20
7352/7352 [==============================] - 21s 3ms/step - loss: 0.9842 - acc: 0.553
Epoch 4/20
7352/7352 [==============================] - 21s 3ms/step - loss: 0.8210 - acc: 0.617
Epoch 5/20
7352/7352 [==============================] - 21s 3ms/step - loss: 0.7619 - acc: 0.636
Epoch 6/20
7352/7352 [==============================] - 21s 3ms/step - loss: 0.7796 - acc: 0.638
Epoch 7/20
7352/7352 [==============================] - 21s 3ms/step - loss: 0.7433 - acc: 0.644
Epoch 8/20
7352/7352 [==============================] - 21s 3ms/step - loss: 0.7222 - acc: 0.648
Epoch 9/20
7352/7352 [==============================] - 21s 3ms/step - loss: 0.7110 - acc: 0.651
Epoch 10/20
7352/7352 [==============================] - 21s 3ms/step - loss: 0.7635 - acc: 0.641
Epoch 11/20
7352/7352 [==============================] - 21s 3ms/step - loss: 0.7975 - acc: 0.624
Epoch 12/20
7352/7352 [==============================] - 21s 3ms/step - loss: 0.7415 - acc: 0.647
Epoch 13/20
7352/7352 [==============================] - 21s 3ms/step - loss: 0.7192 - acc: 0.656
Epoch 14/20
7352/7352 [==============================] - 22s 3ms/step - loss: 0.6957 - acc: 0.659
Epoch 15/20
```

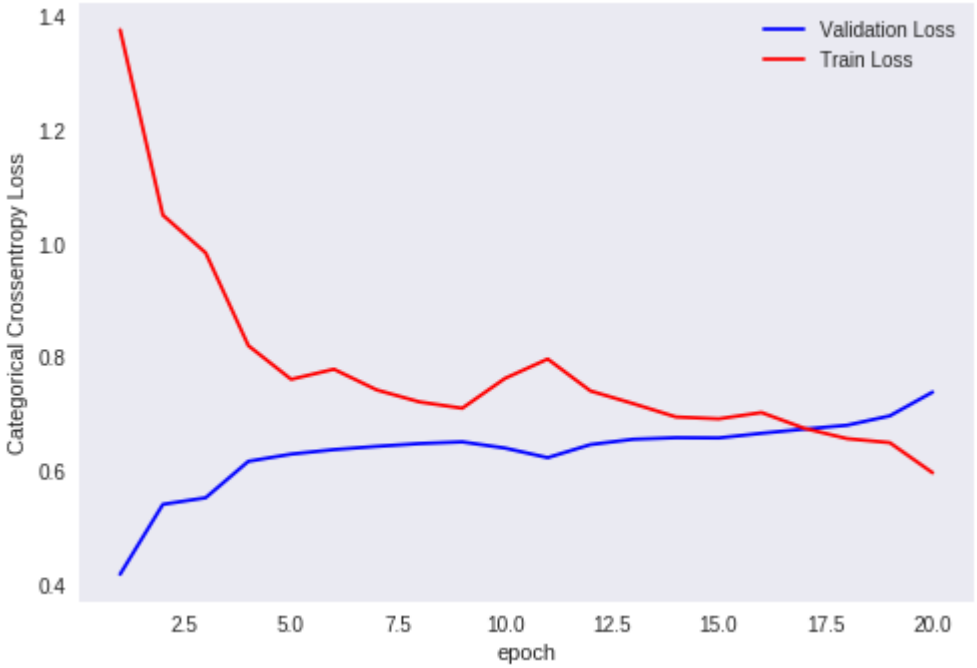
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 509        1        27        0                   0
SITTING                  0      417        64        3                   0
STANDING                 0      127       396        1                   0
WALKING                  0        2         1      425                   0
WALKING_DOWNSTAIRS       0        0         0      371                  18
WALKING_UPSTAIRS         0        1         0       95                   2

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            7
STANDING                           8
WALKING                           68
WALKING_DOWNSTAIRS                31
WALKING_UPSTAIRS                 373
```

```
# Final evaluation of the model
score = model.evaluate(X_test, Y_test)
score_1_2 = score[0]
score_2_2 = score[1]
```

```
print('\nTest score:', score_1_2)
print('Test accuracy:', score_2_2)

print(history.history.keys())

fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

# list of epoch numbers
x = list(range(1,epochs+1))

vy = history.history['acc']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
2947/2947 [==============================] - 1s 406us/step

Test score: 0.6012172610892415
Test accuracy: 0.7254835426059059
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```



Hyperparameter tunning: case 2

```
# Initializing parameters
epochs = 25
batch_size = 32
n_hidden = 64


# 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(0.25))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_4 (LSTM)                (None, 64)                18944     
_________________________________________________________________
dropout_4 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 390       
=================================================================
Total params: 19,334
Trainable params: 19,334
Non-trainable params: 0
_________________________________________________________________
```

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


# Training the model
history = 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 1/25
7352/7352 [==============================] - 21s 3ms/step - loss: 1.2441 - acc: 0.449
Epoch 2/25
7352/7352 [==============================] - 20s 3ms/step - loss: 0.9742 - acc: 0.582
Epoch 3/25
7352/7352 [==============================] - 20s 3ms/step - loss: 0.7984 - acc: 0.625
Epoch 4/25
7352/7352 [==============================] - 20s 3ms/step - loss: 0.7413 - acc: 0.632
Epoch 5/25
7352/7352 [==============================] - 20s 3ms/step - loss: 0.6666 - acc: 0.699
Epoch 6/25
7352/7352 [==============================] - 21s 3ms/step - loss: 0.5595 - acc: 0.790
Epoch 7/25
7352/7352 [==============================] - 20s 3ms/step - loss: 0.4618 - acc: 0.835
Epoch 8/25
7352/7352 [==============================] - 20s 3ms/step - loss: 0.4009 - acc: 0.866
Epoch 9/25
7352/7352 [==============================] - 20s 3ms/step - loss: 0.3360 - acc: 0.893
Epoch 10/25
7352/7352 [==============================] - 20s 3ms/step - loss: 0.2749 - acc: 0.916
```

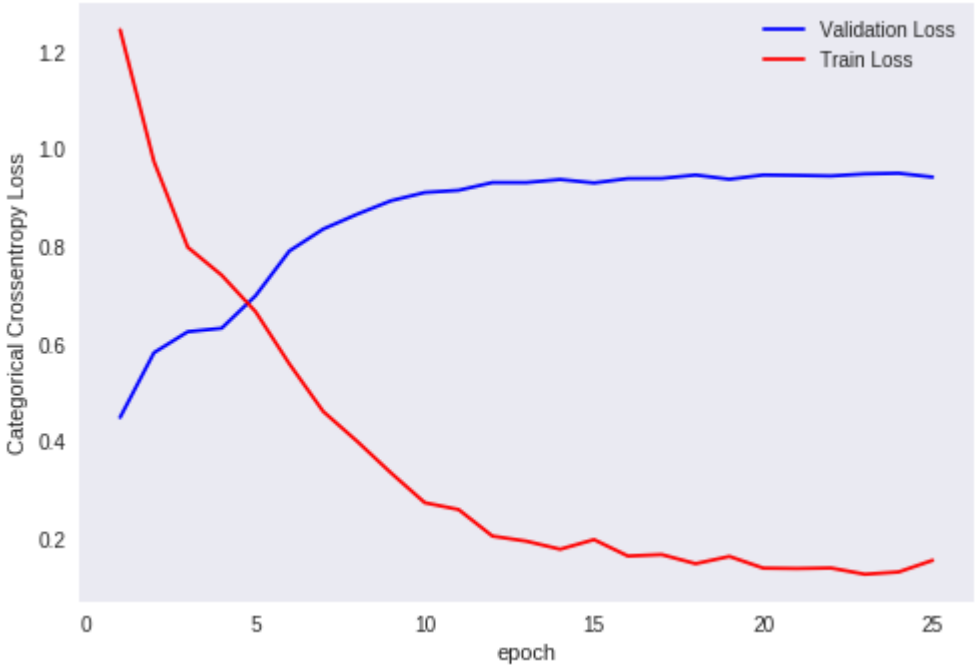
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 511        0         0        0                   0
SITTING                  0      413        72        1                   1
STANDING                 0      106       422        2                   1
WALKING                  0        0         0      464                  25
WALKING_DOWNSTAIRS       0        0         0        0                 415
WALKING_UPSTAIRS         0        0         0       18                  11

Pred                WALKING_UPSTAIRS
True
LAYING                            26
SITTING                            4
STANDING                           1
WALKING                            7
WALKING_DOWNSTAIRS                 5
WALKING_UPSTAIRS                 442
```

```
# Final evaluation of the model
score = model.evaluate(X_test, Y_test)
score_1_2 = score[0]
score_2_2 = score[1]
print('\nTest score:', score_1_2)
print('Test accuracy:', score_2_2)

print(history.history.keys())

fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

# list of epoch numbers
x = list(range(1,epochs+1))
```

```
vy = history.history['acc']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
2947/2947 [==============================] - 1s 468us/step

Test score: 0.3409559749895642
Test accuracy: 0.9049881235154394
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```



Hyperparameter tunning: case 3

```
# Initializing parameters
epochs = 25
batch_size = 24
n_hidden = 48


# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim),return_sequences=True))
# Adding a dropout layer
model.add(Dropout(0.25))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))

# Configuring the parameters
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
# 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()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_7 (LSTM)                (None, 128, 48)           11136
_________________________________________________________________
dropout_7 (Dropout)          (None, 128, 48)           0
_________________________________________________________________
dense_7 (Dense)              (None, 128, 6)            294
_________________________________________________________________
lstm_8 (LSTM)                (None, 48)                10560
_________________________________________________________________
dropout_8 (Dropout)          (None, 48)                0
_________________________________________________________________
dense_8 (Dense)              (None, 6)                 294
=================================================================
Total params: 22,284
Trainable params: 22,284
```

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


# Training the model
history = 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 1/25
7352/7352 [==============================] - 47s 6ms/step - loss: 1.0725 - acc: 0.525
Epoch 2/25
7352/7352 [==============================] - 44s 6ms/step - loss: 0.8095 - acc: 0.635
Epoch 3/25
7352/7352 [==============================] - 44s 6ms/step - loss: 0.6834 - acc: 0.705
Epoch 4/25
7352/7352 [==============================] - 45s 6ms/step - loss: 0.5835 - acc: 0.743
Epoch 5/25
7352/7352 [==============================] - 45s 6ms/step - loss: 0.5066 - acc: 0.760
Epoch 6/25
7352/7352 [==============================] - 45s 6ms/step - loss: 0.4805 - acc: 0.772
Epoch 7/25
7352/7352 [==============================] - 45s 6ms/step - loss: 0.4430 - acc: 0.788
Epoch 8/25
7352/7352 [==============================] - 44s 6ms/step - loss: 0.4395 - acc: 0.787
Epoch 9/25
7352/7352 [==============================] - 44s 6ms/step - loss: 0.4020 - acc: 0.802
Epoch 10/25
7352/7352 [==============================] - 45s 6ms/step - loss: 0.3743 - acc: 0.837
Epoch 11/25
7352/7352 [==============================] - 45s 6ms/step - loss: 0.2883 - acc: 0.903
Epoch 12/25
```

```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

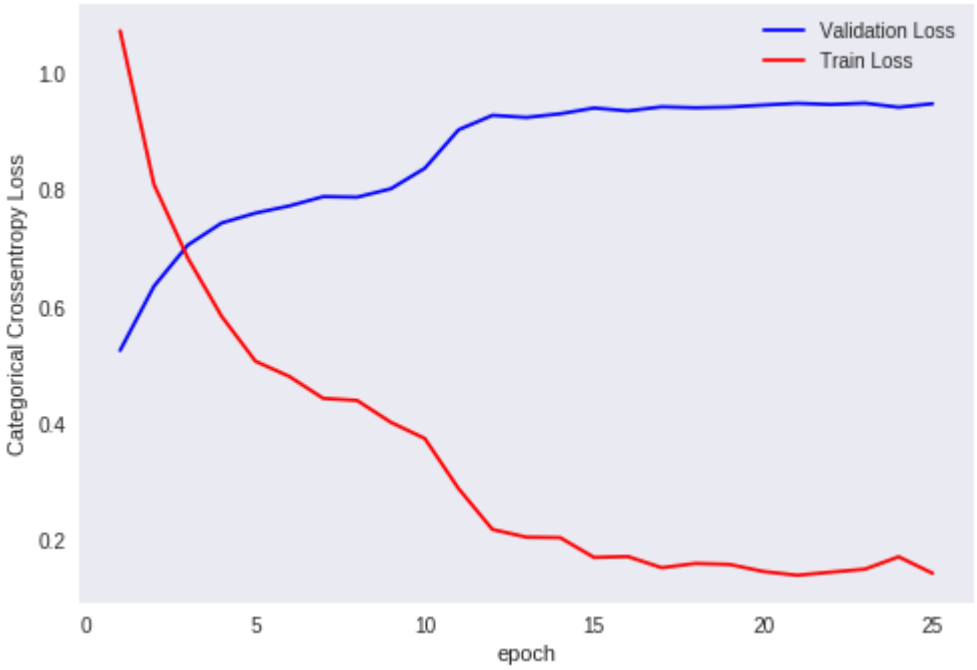
```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 510        0         0        0                  26
SITTING                  0      384        83        1                   0
STANDING                 0       60       457       15                   0
WALKING                  0        0         1      433                  56
WALKING_DOWNSTAIRS       0        0         0        0                 417
WALKING_UPSTAIRS         0        0         0        5                  23

Pred                WALKING_UPSTAIRS
True
LAYING                             1
SITTING                           23
STANDING                           0
WALKING                            6
WALKING_DOWNSTAIRS                 3
WALKING_UPSTAIRS                 443
```

```
Epoch 24/25
```

```
# Final evaluation of the model
score = model.evaluate(X_test, Y_test)
score_1_2 = score[0]
score_2_2 = score[1]
print('\nTest score:', score_1_2)
print('Test accuracy:', score_2_2)

print(history.history.keys())

fig,ax = plt.subplots(1,1)
```

```
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

# list of epoch numbers
x = list(range(1,epochs+1))

vy = history.history['acc']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
2947/2947 [==============================] - 2s 808us/step

Test score: 0.5399093672506643
Test accuracy: 0.8971835765184933
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```



Hyperparameter tunning: case 4

```
# Initializing parameters
epochs = 25
batch_size = 32
n_hidden = 64


# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim),return_sequences=True))
# Adding a dropout layer
model.add(Dropout(0.25))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))

# Configuring the parameters
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
# 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()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_9 (LSTM)                (None, 128, 64)           18944     
_________________________________________________________________
dropout_9 (Dropout)          (None, 128, 64)           0         
_________________________________________________________________
dense_9 (Dense)              (None, 128, 6)            390       
_________________________________________________________________
lstm_10 (LSTM)               (None, 64)                18176     
_________________________________________________________________
dropout_10 (Dropout)         (None, 64)                0         
_________________________________________________________________
dense_10 (Dense)             (None, 6)                 390       
=================================================================
Total params: 37,900
Trainable params: 37,900
Non-trainable params: 0
_________________________________________________________________
```

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


# Training the model
history = 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 1/25
7352/7352 [==============================] - 46s 6ms/step - loss: 1.3817 - acc: 0.397
Epoch 2/25
7352/7352 [==============================] - 42s 6ms/step - loss: 1.2321 - acc: 0.455
Epoch 3/25
7352/7352 [==============================] - 42s 6ms/step - loss: 1.1934 - acc: 0.467
Epoch 4/25
7352/7352 [==============================] - 42s 6ms/step - loss: 1.1453 - acc: 0.469
Epoch 5/25
7352/7352 [==============================] - 42s 6ms/step - loss: 1.0521 - acc: 0.540
Epoch 6/25
7352/7352 [==============================] - 42s 6ms/step - loss: 0.8402 - acc: 0.641
Epoch 7/25
7352/7352 [==============================] - 43s 6ms/step - loss: 0.6232 - acc: 0.727
Epoch 8/25
7352/7352 [==============================] - 43s 6ms/step - loss: 0.5545 - acc: 0.754
Epoch 9/25
7352/7352 [==============================] - 42s 6ms/step - loss: 0.5375 - acc: 0.758
Epoch 10/25
7352/7352 [==============================] - 43s 6ms/step - loss: 0.4578 - acc: 0.778
Epoch 11/25
7352/7352 [==============================] - 42s 6ms/step - loss: 0.4498 - acc: 0.775
Epoch 12/25
7352/7352 [==============================] - 42s 6ms/step - loss: 0.4430 - acc: 0.783
Epoch 13/25
7352/7352 [==============================] - 42s 6ms/step - loss: 0.3869 - acc: 0.823
Epoch 14/25
7352/7352 [==============================] - 43s 6ms/step - loss: 0.3470 - acc: 0.876
```

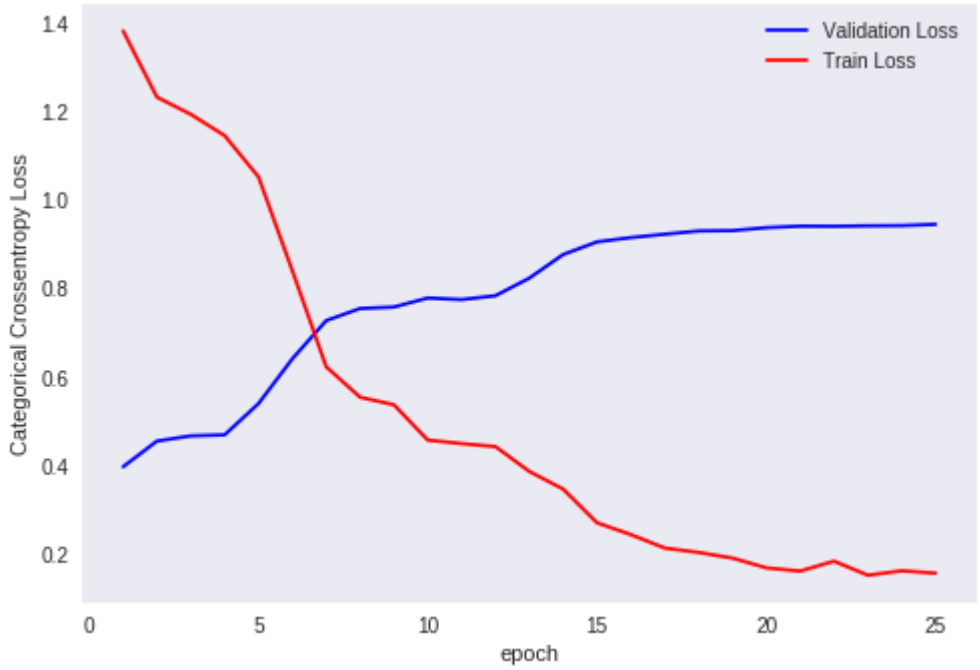
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 536        1         0        0                   0
SITTING                  6      387        84        0                   0
STANDING                 0       58       474        0                   0
WALKING                  0        2        22      438                  34
WALKING_DOWNSTAIRS       0        0         0        5                 415
WALKING_UPSTAIRS         0        5         3       22                   8

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                           14
STANDING                           0
WALKING                            0
WALKING_DOWNSTAIRS                 0
WALKING_UPSTAIRS                 433
```

```
# Final evaluation of the model
score = model.evaluate(X_test, Y_test)
score_1_2 = score[0]
score_2_2 = score[1]
print('\nTest score:', score_1_2)
```

```
print('Test accuracy:', score_2_2)

print(history.history.keys())

fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

# list of epoch numbers
x = list(range(1,epochs+1))

vy = history.history['acc']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
2947/2947 [==============================] - 3s 999us/step

Test score: 0.3675638035252511
Test accuracy: 0.9104173736002714
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```



Hyperparameter tunning: case 5

```
# Initializing parameters
epochs = 25
batch_size = 32
n_hidden = 64


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

# 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()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_11 (LSTM)               (None, 128, 64)           18944     
_________________________________________________________________
dropout_11 (Dropout)         (None, 128, 64)           0         
_________________________________________________________________
dense_11 (Dense)             (None, 128, 6)            390       
_________________________________________________________________
lstm_12 (LSTM)               (None, 64)                18176     
_________________________________________________________________
dropout_12 (Dropout)         (None, 64)                0         
_________________________________________________________________
dense_12 (Dense)             (None, 6)                 390       
=================================================================
Total params: 37,900
Trainable params: 37,900
Non-trainable params: 0
_________________________________________________________________
```

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


# Training the model
history = model.fit(X_train,
                    Y_train,
                    batch_size=batch_size,
                    validation_data=(X_test, Y_test),
                    epochs=epochs)
```

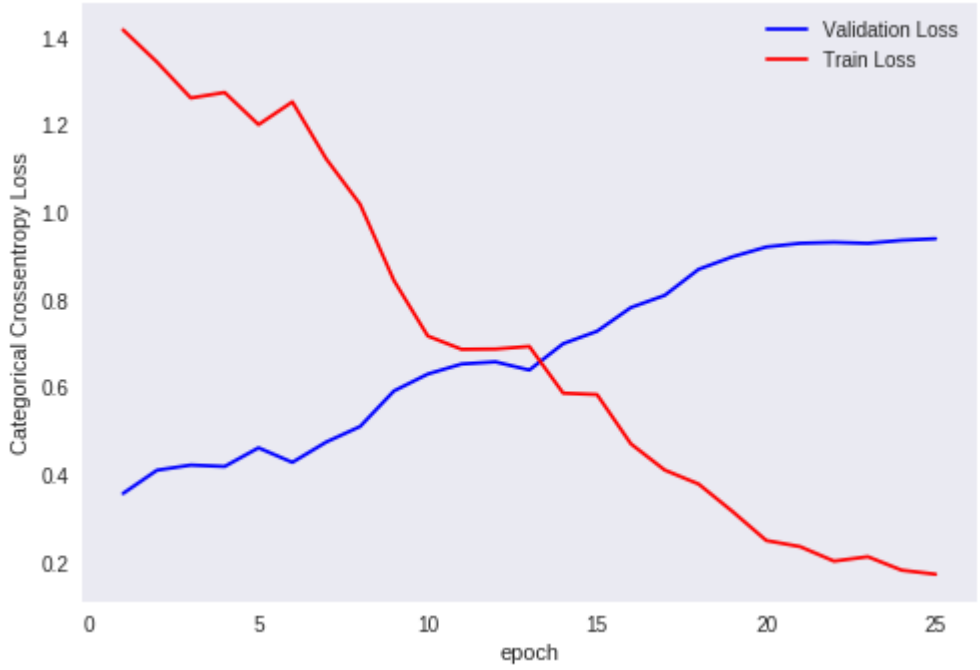
```
Train on 7352 samples, validate on 2947 samples
Epoch 1/25
7352/7352 [==============================] - 46s 6ms/step - loss: 1.4182 - acc: 0.357
Epoch 2/25
7352/7352 [==============================] - 43s 6ms/step - loss: 1.3446 - acc: 0.410
Epoch 3/25
7352/7352 [==============================] - 42s 6ms/step - loss: 1.2623 - acc: 0.422
Epoch 4/25
7352/7352 [==============================] - 42s 6ms/step - loss: 1.2750 - acc: 0.419
Epoch 5/25
7352/7352 [==============================] - 42s 6ms/step - loss: 1.2014 - acc: 0.461
Epoch 6/25
7352/7352 [==============================] - 42s 6ms/step - loss: 1.2535 - acc: 0.428
Epoch 7/25
7352/7352 [==============================] - 42s 6ms/step - loss: 1.1230 - acc: 0.475
Epoch 8/25
7352/7352 [==============================] - 42s 6ms/step - loss: 1.0193 - acc: 0.510
Epoch 9/25
7352/7352 [==============================] - 42s 6ms/step - loss: 0.8447 - acc: 0.591
Epoch 10/25
7352/7352 [==============================] - 42s 6ms/step - loss: 0.7178 - acc: 0.630
Epoch 11/25
7352/7352 [==============================] - 43s 6ms/step - loss: 0.6873 - acc: 0.653
Epoch 12/25
7352/7352 [==============================] - 43s 6ms/step - loss: 0.6880 - acc: 0.658
Epoch 13/25
7352/7352 [==============================] - 43s 6ms/step - loss: 0.6940 - acc: 0.639
Epoch 14/25
7352/7352 [==============================] - 43s 6ms/step - loss: 0.5870 - acc: 0.700
Epoch 15/25
7352/7352 [==============================] - 43s 6ms/step - loss: 0.5838 - acc: 0.728
Epoch 16/25
7352/7352 [==============================] - 43s 6ms/step - loss: 0.4710 - acc: 0.782
```

```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 523        0         0        0                  14
SITTING                  2      398        68        1                   2
STANDING                 0       99       433        0                   0
WALKING                  0        0         0      445                  38
WALKING_DOWNSTAIRS       0        0         0        1                 418
WALKING_UPSTAIRS         2        2         0       21                  29

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                           20
STANDING                           0
WALKING                           13
WALKING_DOWNSTAIRS                 1
WALKING_UPSTAIRS                 417
```

```
# Final evaluation of the model
score = model.evaluate(X_test, Y_test)
score_1_2 = score[0]
score_2_2 = score[1]
print('\nTest score:', score_1_2)
print('Test accuracy:', score_2_2)

print(history.history.keys())

fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

# list of epoch numbers
x = list(range(1,epochs+1))

vy = history.history['acc']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```

```
2947/2947 [==============================] - 3s 982us/step

Test score: 0.45591508246931634
Test accuracy: 0.8937902952154734
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```




```
score_1 = 0.3421370585869036
score_2 = 0.9063454360366474

score_3 = 0.6012172610892415
score_4 = 0.7254835426059059

score_5 = 0.3409559749895642
score_6 = 0.9049881235154394

score_7 = 0.5399093672506643
score_8 = 0.8971835765184933

score_9 =  0.3675638035252511
score_10 = 0.9104173736002714

score_11 = 0.45591508246931634
score_12 = 0.8937902952154734
```

```
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Models/Paramters","Test Score", "Test accuracy"]

x.add_row(["Base Model: ",score_1, score_2])
x.add_row(["Hyperparameter tunning - Case 1 : ", score_3, score_4])
x.add_row(["Hyperparameter tunning - Case 2 : ",score_5, score_6])
x.add_row(["Hyperparameter tunning - Case 3 : ", score_7, score_8])
x.add_row(["Hyperparameter tunning - Case 4 : ",score_9, score_10])
x.add_row(["Hyperparameter tunning - Case 5 : ", score_11, score_12])


print(x)
```

```
+-----------------------------------+---------------------+--------------------+
|          Models/Paramters         |      Test Score     |   Test accuracy    |
+-----------------------------------+---------------------+--------------------+
|            Base Model:            |  0.3421370585869036 | 0.9063454360366474 |
| Hyperparameter tunning - Case 1 : |  0.6012172610892415 | 0.7254835426059059 |
| Hyperparameter tunning - Case 2 : |  0.3409559749895642 | 0.9049881235154394 |
| Hyperparameter tunning - Case 3 : |  0.5399093672506643 | 0.8971835765184933 |
| Hyperparameter tunning - Case 4 : |  0.3675638035252511 | 0.9104173736002714 |
| Hyperparameter tunning - Case 5 : | 0.45591508246931634 | 0.8937902952154734 |
+-----------------------------------+---------------------+--------------------+
```

In our base model, we have taken only one hidden layer, one dropout layer and one dense layer, and we got an accuracy of 90% with a loss of 0.34. We have taken accuracy as measure because the data is not unbalanced. So, now our aim is to improve accuracy by trying different architechture of NN.

Here, I've tried a combination of a different number of batch size, number of hidden layer, percentage of drop out and adding another layer of LSTM. Above table summarizes the results. Finally, I was ableto improve accuracy to 91% in case of 2-layer LSTM model with 25% of dropout in both layer with 32 neurons in hidden layers and batch size of 32.

Double-click (or enter) to edit