Human Activity Recognition

- Millions of People on Earth have a Smartphone available with them, and in addition to the cameras, smartphones also have the presence of many sensors in them. **These Sensors are extremely useful especially when we are playing Games on our Smartphones.**
- Two such Sensors on our Smartphones are Accelerometer and a Gyroscope : ***An Accelerometer measures the Acceleration whereas a Gyroscope measures the Angular Velocity.***
- Now given that we have a Smartphone and we are able to collect the Accelerometer as well as the Gyroscope Data, the question is : **Can we somehow use this data to predict the activity that the person is performing?**
- The bunch of activities that a person is performing includes Walking, Walking Upstairs, Walking Downstairs, Sitting Down, Standing Up, Laying Down etc. :- A Total of 6 Activities.

Q. Why is this Useful in the First Place?

- There are very popular Smart watches like Fitbit, Apple Watch etc., which also internally contain an accelerometer and a gyroscope. This data is actually used to determine the activity that you are carrying out and to track your activities throughout the day.
- The advantage is that after all of this data is collected by the Smart watches, the devices predict the amount of calories that have been burnt by the user throughout the course of the day. Similarly these devices can also measure the number of hours that the user has slept, because if you are laying down, it is most likely that you are sleeping.
- ***However, it is to be noted that most of the Modern Smartphones have an additional sensor called a Heart-Rate Sensor, which is used in addition to accelerometers and gyroscopes to determine if you are actually sleeping or not.*** *The earlier versions of these devices did not have Heart Rate Monitors because these sensors were expensive at that period of time. Also, we cannot have Heart Rate Monitors on our Smartphones.*

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Mathematical Understanding of the Problem :

- Most of these Accelerometers that we have are Tri-Axial Accelerometers which means that the acceleration is measured across the X-axis, Y-axis and Z-axis. Similarly the Gyroscopes measure the Angular Velocity across the X-axis, Y-axis and Z-axis.
- In the Accelerometer, we have the data with the X-axis acceleration measured over time. Similarly we have the acceleration on Y-axis measured over time and acceleration on Z-axis measured over time. Similarly for the Gyroscope, we have Angular Velocity measurements across X-axis, Y-axis and Z-axis. **We basically have a total of 6 Time-Series Data.**
- This 6 Time-Series Data becomes our Input, using which we want to predict our Output. The output that we want is the Activity : one of the 6 classes. **At the outset, it obviously looks like a Multi-class Classification Problem.**
- Some of the Metrics that we use in case of a Multi-class Classification Problem is the Overall Accuracy, as well as the Confusion Matrices, and the Multi-class log loss.
- The Confusion Matrices are important because they give information about the type of confusions that are taking place. Eg : Our Classification Model could get confused between 'Sitting' and 'Standing'.

==========

- This dataset is collected from 30 people (referred as subjects in this dataset) in a laboratory type setting, performing different activities with a smartphone to their waists. The data is recorded with the help of sensors (accelerometer and Gyroscope) in that smartphone.
- This experiment was video recorded to label the data manually. That was the only purpose of the same. This video data has not been provided to us.

How was this Data Recorded?

- The Source for the data has been provided in the "References" Section in the IPYNB (https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones), which also gives a brief summary on the data, that it is a Multivariate Time Series Dataset with 10299 instances and 561 attributes.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

- By using the sensors(Gyroscope and accelerometer) in a smartphone, they have captured '3-axial linear acceleration'(*tAcc-XYZ*) from accelerometer and '3-axial angular velocity' (*tGyro-XYZ*) from Gyroscope with several variations.

> prefix 't' in those metrics denotes time.
>
> suffix 'XYZ' represents 3-axial signals in X , Y, and Z directions.

How was this Data Processed?

Feature Names

- Assume that you have a data that depicts the acceleration along the x-axis being plotted against time. ***These sensor signals are preprocessed by applying noise filters and then sampled in fixed-width windows(sliding windows) of 2.56 seconds each with 50% overlap. ie., each window has 128 readings. This means that each of the vectors that we obtain is of dimensionality equal to 128.***
- From Each window, a feature vector was obtained by calculating variables from the time and frequency domain. The vector thus obtained is of a fixed length. Now we can sample this and obtain values at a fixed interval.
- If we were to solve this problem in the real world, we would have worked with experts in signal processing who would have helped us with these filtering operations.
- Now we have already obtained our first vector. ***The second vector is not obtained by directly going to the second window.*** Now what we do is that we take the second window that overlaps this first window by exactly 50%.
- This means that instead of taking a window from time (2.56s to 2 * 2.56s), we are taking a window from (1.28s to 1.28 + 2.56). **This entire technique is called Overlappped Sampling, wherein the second window is overlapping the first window by 50%.** Now for this window obtained, we apply some signal processing filters, carry out some sampling, and convert it into a numerical vector of a fixed size.

This entire technique that we discussed above is not carried out not just for Acc(x) ie. acceleration along the x-axis, but for all the time series datasets that we have.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

==========

- In the original dataset we have accelerations along the x, y and z axes and the angular velocities along the x, y and z axes. *All of these are Time Series Datasets. We have a total of 6 time series datasets.*
- The people who created this dataset are domain experts, who know in depth about Human Activity Recognition, about Accelerometers, Gyroscopes etc. Also, they have a deep understanding of how the time series datasets look like.
- Therefore, in addition to providing us with the 6 Time Series Datasets, these domain experts have designed/engineered features for us. **Some of these engineered features could be very complex because they involve a lot of Signal Processing stuff.**
- These features are as shown below. Eg : 'tBodyAccJerk-XYZ_'. The Jerk over here refers to the rate of change of acceleration, whereas the definition of acceleration is rate of change of velocity.

For all of these manually engineered features, all of the features that are 'starting with t' are Time Based Features and all the features that are 'starting with f' are after applying the FFT.

> In our dataset, each datapoint represents a window with different readings

- The accelertion signal was saperated into Body and Gravity acceleration signals(***tBodyAcc-XYZ*** and ***tGravityAcc-XYZ***) using some low pass filter with corner frequecy of 0.3Hz.

1. After that, the body linear acceleration and angular velocity were derived in time to obtian *jerk signals* (***tBodyAccJerk-XYZ*** and ***tBodyGyroJerk-XYZ***).
2. The magnitude of these 3-dimensional signals were calculated using the Euclidian norm. This magnitudes are represented as features with names like *tBodyAccMag*, *tGravityAccMag*, *tBodyAccJerkMag*, *tBodyGyroMag* and *tBodyGyroJerkMag*.
3. Finally, We've got frequency domain signals from some of the available signals by applying a FFT (Fast Fourier Transform). These signals obtained were labeled with ***prefix 'f'*** just like

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

original signals with ***prefix 't'***. These signals are labeled as ***fBodyAcc-XYZ***, ***fBodyGyroMag*** etc.,.

4. These are the signals that we got so far.

    - tBodyAcc-XYZ
    - tGravityAcc-XYZ
    - tBodyAccJerk-XYZ
    - tBodyGyro-XYZ
    - tBodyGyroJerk-XYZ
    - tBodyAccMag - The Mag over here refers to the Magnitude of the wave.
    - tGravityAccMag
    - tBodyAccJerkMag
    - tBodyGyroMag
    - tBodyGyroJerkMag
    - fBodyAcc-XYZ
    - fBodyAccJerk-XYZ
    - fBodyGyro-XYZ
    - fBodyAccMag
    - fBodyAccJerkMag
    - fBodyGyroMag
    - fBodyGyroJerkMag
5. Our Raw Data is encoded as a 128 dimensional vector for every window that we have, and on top of this vector we have computed a lot of features. We can esitmate some set of variables from the above signals. ie., **We will estimate the following properties on each and every signal that we recoreded so far.**

    - ***mean()***: Mean value
    - ***std()***: Standard deviation
    - ***mad()***: Median absolute deviation
    - ***max()***: Largest value in array
    - ***min()***: Smallest value in array
    - ***sma()***: Signal magnitude area
    - ***energy()***: Energy measure. Sum of the squares divided by the number of values.
    - ***iqr()***: Interquartile range

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

- ***entropy()***: Signal entropy
- ***arCoeff()***: Autorregresion coefficients with Burg order equal to 4
- ***correlation()***: correlation coefficient between two signals
- ***maxInds()***: index of the frequency component with largest magnitude
- ***meanFreq()***: Weighted average of the frequency components to obtain a mean frequency
- ***skewness()***: skewness of the frequency domain signal
- ***kurtosis()***: kurtosis of the frequency domain signal
- ***bandsEnergy()***: Energy of a frequency interval within the 64 bins of the FFT of each window.
- ***angle()***: Angle between to vectors.

6. We can obtain some other vectors by taking the average of signals in a single window sample. These are used on the angle() variable' `

    - gravityMean
    - tBodyAccMean
    - tBodyAccJerkMean
    - tBodyGyroMean
    - tBodyGyroJerkMean

Y_Labels(Encoded)

- 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**

Train and test data were saperated

- The readings from ***70%*** of the volunteers (21 people) were taken as ***trianing data*** and remaining ***30%*** subjects (9 people) recordings were taken for ***test data***.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Data

- All the data is present in 'UCI_HAR_dataset/' folder in present working directory.
  - Feature names are present in 'UCI_HAR_dataset/features.txt'
  - ***Train Data***
    - 'UCI_HAR_dataset/train/X_train.txt'
    - 'UCI_HAR_dataset/train/subject_train.txt'
    - 'UCI_HAR_dataset/train/y_train.txt'
  - ***Test Data***
    - 'UCI_HAR_dataset/test/X_test.txt'
    - 'UCI_HAR_dataset/test/subject_test.txt'
    - 'UCI_HAR_dataset/test/y_test.txt'
- The 'Readme' text file over here tells us how this data was generated and gives information about what each of the files mean, and a complete list of all the 561 Expert Generated Features.
- What we have in X_train over here are Expert Engineered Features for each window, whereas y_train has the corresponding labels : {1,2,3,4,5,6}, corresponding to each of our activities.
- We also have a folder called 'Inertial Signals' that has the raw time series data broken down into windows.

Data Size :

> 27 MB

This case study is a tradeoff between Expert Engineered Features and the raw time series, which is broken down into windows with overlaps. Therefore we will build models on the Expert Engineered Features as well as the Raw Time series data that we have. In the latter case we will be building Deep Learning Models, especially RNN Models because we know that RNNs can handle Time Series Data, and any sequence data in general very well.

1. Quick overview of the dataset :

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

- Accelerometer and Gyroscope readings are taken from 30 volunteers(referred as subjects) while performing the following 6 Activities.

    1. Walking
    2. Walking Upstairs
    3. Walking Downstairs
    4. Standing
    5. Sitting
    6. Lying.

- Readings are divided into a window of 2.56 seconds with 50% overlapping.
- Accelerometer readings are divided into gravity acceleration and body acceleration readings, which has x,y and z components each.
- Gyroscope readings are the measure of angular velocities which has x,y and z components.
- Jerk signals are calculated for BodyAcceleration readings.
- Fourier Transforms are made on the above time readings to obtain frequency readings.
- Now, on all the base signal readings., mean, max, mad, sma, arcoefficient, engerybands,entropy etc., are calculated for each window.
- We get a feature vector of 561 features and these features are given in the dataset.
- Each window of readings is a datapoint of 561 features.

1.1 Problem Framework

- 30 subjects(volunteers) data is randomly split to 70%(21) Train and 30%(7) Test data.
- Each datapoint corresponds one of the 6 Activities.

1.2 Problem Statement

- Given a new datapoint we have to predict the Activity.

    We will basically start with these Expert Generated Features, carry out some EDA, build some models using these expert generated features, after which we will go to

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

the raw 128 dimensional time series data, and then apply Deep Learning Models on top of them.

In [1]:
```
import numpy as np
import pandas as pd

# get the features from the file features.txt
features = list()
with open('HAR/UCI_HAR_Dataset/features.txt') as f:
    features = [line.split()[1] for line in f.readlines()]
print('No of Features: {}'.format(len(features)))
```

```
No of Features: 561
```

1.3 Obtain the Train Data

In [2]:
```
# get the data from txt files to pandas dataffame
X_train = pd.read_csv('HAR/UCI_HAR_dataset/train/X_train.txt', delim_wh
itespace=True, header=None)
train_final = X_train
train_final.columns = features

# add subject column to the dataframe
train_final['subject'] = pd.read_csv('HAR/UCI_HAR_dataset/train/subject
_train.txt', header=None, squeeze=True)

y_train = pd.read_csv('HAR/UCI_HAR_dataset/train/y_train.txt', names=[
'Activity'], squeeze=True)
y_train_labels = y_train.map({1: 'WALKING', 2:'WALKING_UPSTAIRS',3:'WAL
KING_DOWNSTAIRS',\
                        4:'SITTING', 5:'STANDING',6:'LAYING'})

# put all columns in a single dataframe
train_final['Activity'] = y_train
train_final['ActivityName'] = y_train_labels
train_final.sample()
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Out[2]:

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc mad()-
3072	0.278744	-0.021144	-0.101674	-0.995358	-0.97887	-0.975187	-0.996012	-0.97725

1 rows × 564 columns

1.4 Obtain the Test Data

In [3]:

```
# get the data from txt files to pandas dataffame
X_test = pd.read_csv('HAR/UCI_HAR_dataset/test/X_test.txt', delim_white
space=True, header=None)
test_final = X_test
test_final.columns = features

# add subject column to the dataframe
test_final['subject'] = pd.read_csv('HAR/UCI_HAR_dataset/test/subject_t
est.txt', header=None, squeeze=True)

# get y labels from the txt file
y_test = pd.read_csv('HAR/UCI_HAR_dataset/test/y_test.txt', names=['Act
ivity'], squeeze=True)
y_test_labels = y_test.map({1: 'WALKING', 2:'WALKING_UPSTAIRS',3:'WALKI
NG_DOWNSTAIRS',\
                        4:'SITTING', 5:'STANDING',6:'LAYING'})


# put all columns in a single dataframe
test_final['Activity'] = y_test
test_final['ActivityName'] = y_test_labels
test_final.sample()
```

Out[3]:

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y
694	0.247733	-0.037917	-0.140631	-0.208779	-0.162206	-0.04182	-0.250009	-0.19158

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

1 rows × 564 columns

In [4]:
```
print(train_final.shape)
print(test_final.shape)
```

```
(7352, 564)
(2947, 564)
```

2. Data Cleaning

2.1 Check for Duplicates

In [5]:
```
print('No of duplicates in train: {}'.format(sum(train_final.duplicated
()))
print('No of duplicates in test : {}'.format(sum(test_final.duplicated
())))
```

```
No of duplicates in train: 0
No of duplicates in test : 0
```

Observations

- There are no duplicates present in both the Train Data as well as the Test Data.

2.2 Checking for NaN/null values

In [6]:
```
print('We have {} NaN/Null values in train'.format(train_final.isnull()
.values.sum()))
print('We have {} NaN/Null values in test'.format(test_final.isnull().v
alues.sum()))
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
We have 0 NaN/Null values in train
We have 0 NaN/Null values in test
```

Observations

- There are no Null values present in both the Train Data as well as the Test Data.

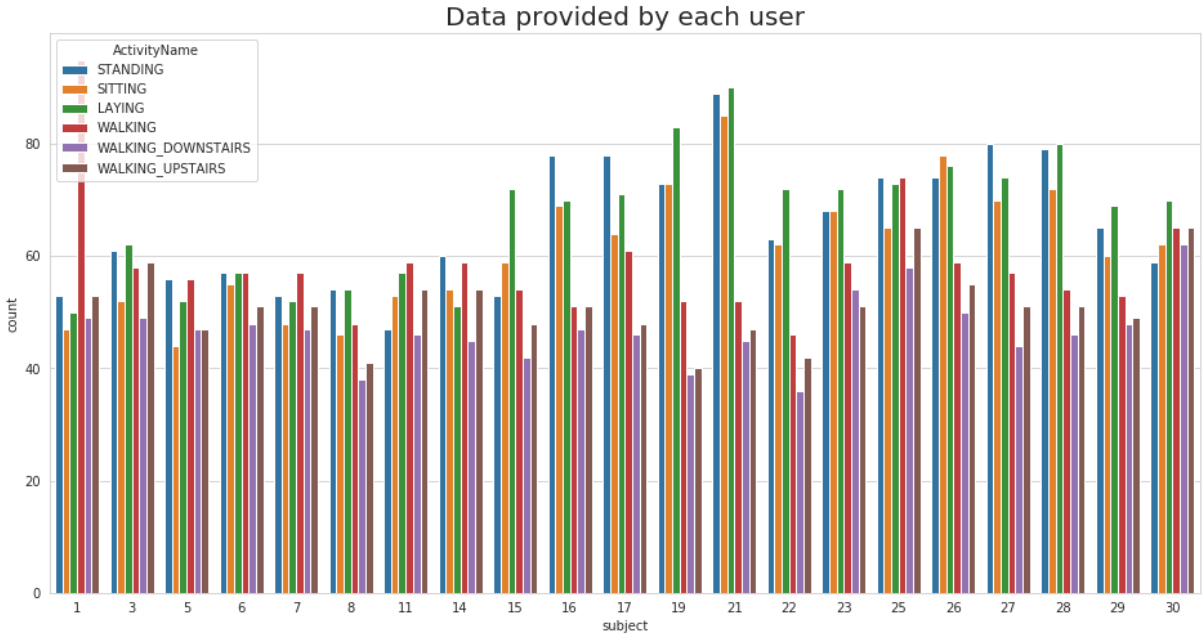
2.3 Check for Data Imbalance

- This is a very important thing to be checked for since we have a 6-class Classification Task.
- Since we have a total of 30 subjects whose data we are collecting, we check the number of datapoints that we have for each person. Eg : For Person 1, we check the number of datapoints that we have for 'Standing'.

In [7]:
```
import warnings
warnings.filterwarnings('ignore')

import matplotlib.pyplot as plt
import seaborn as sns

sns.set_style('whitegrid')
plt.rcParams['font.family'] = 'Dejavu Sans'
```

In [8]:
```
plt.figure(figsize=(16,8))
plt.title('Data provided by each user', fontsize=20)
sns.countplot(x='subject',hue='ActivityName', data = train_final)
plt.show()
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



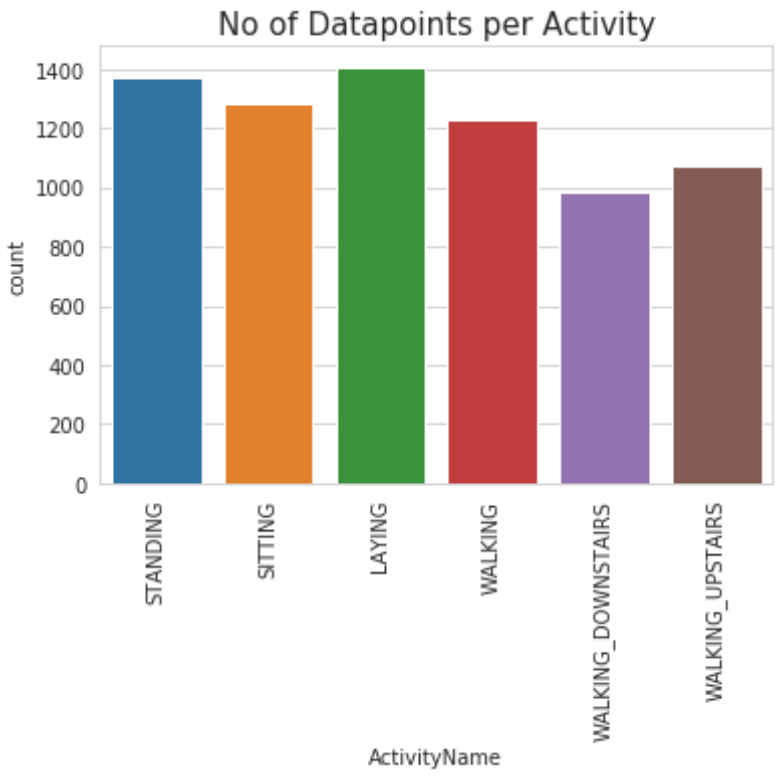
Observations

- We can see that for most of the people we have a decent amount of data. **There is not a Huge Skew per user.** There is some difference but it is not a huge discrepancy among different users.

Now we check the Number of Datapoints that we have per class label :

In [9]:
```
#We are checking the number of datapoints present per class label
plt.title('No of Datapoints per Activity', fontsize=15)
sns.countplot(train_final.ActivityName)
plt.xticks(rotation=90)
plt.show()
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



Observations

- We can notice that across all of our class labels, we have the least amount of data for 'Walking Downstairs', but the number of datapoints is not significantly lower than 'Laying'.
- This shows that there is some class imbalance but this imbalance is not severe that we have to be worried about.

2.4 Changing Feature Names

In [10]:
```
columns = train_final.columns
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
# Removing '()' from the feature/column names
columns = columns.str.replace('[()]','')
columns = columns.str.replace('[-]', '')
columns = columns.str.replace('[,]','')

train_final.columns = columns
test_final.columns = columns

test_final.columns
```

Out[10]:
```
Index(['tBodyAccmeanX', 'tBodyAccmeanY', 'tBodyAccmeanZ', 'tBodyAccstd
X',
       'tBodyAccstdY', 'tBodyAccstdZ', 'tBodyAccmadX', 'tBodyAccmadY',
       'tBodyAccmadZ', 'tBodyAccmaxX',
       ...
       'angletBodyAccMeangravity', 'angletBodyAccJerkMeangravityMean',
       'angletBodyGyroMeangravityMean', 'angletBodyGyroJerkMeangravityM
ean',
       'angleXgravityMean', 'angleYgravityMean', 'angleZgravityMean',
       'subject', 'Activity', 'ActivityName'],
      dtype='object', length=564)
```

2.5 Save this Dataframe in csv Files

In [11]:
```
train_final.to_csv('HAR/UCI_HAR_Dataset/csv_files/train_final.csv', ind
ex=False)
test_final.to_csv('HAR/UCI_HAR_Dataset/csv_files/test_final.csv', index
=False)

#These csv files thus generated will consist of only the Expert Enginee
red Features.
```

3. Exploratory Data Analysis

"***Without domain knowledge EDA has no meaning, without EDA a problem has no soul.***"

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

3.1 Featuring Engineering from Domain Knowledge

We can typically break up our 6 Activities into 2 Broad Categories : Static Activities & Dynamic Activities.

- **Static and Dynamic Activities**
  - In Static Activities (Sitting, Standing & Lying Down), Motion Information will not be very useful.
  - In the Dynamic Activities (Walking, Walking_Upstairs & Walking_Downstairs) Motion info will be significant.

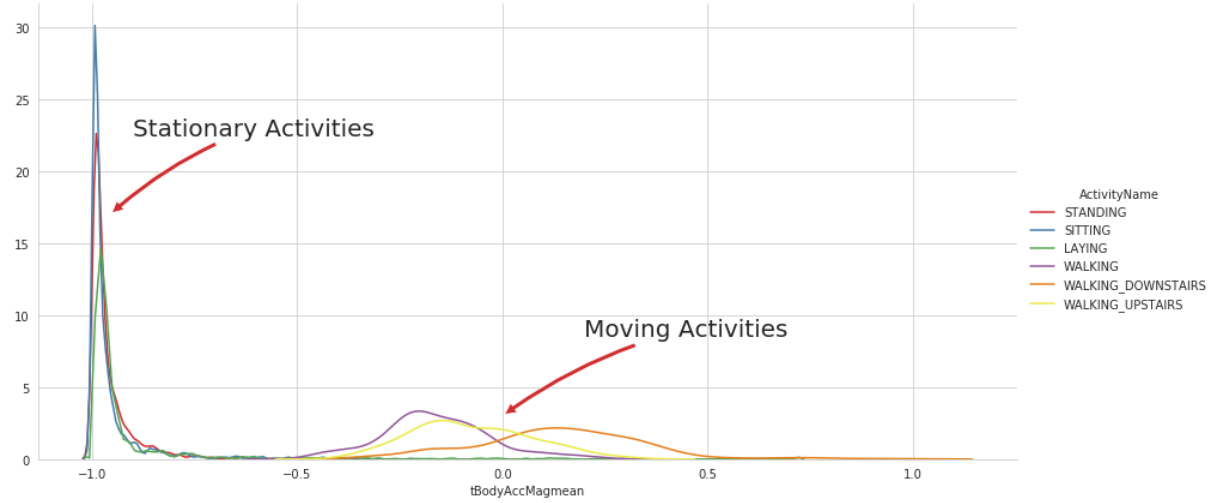
3.2 Stationary and Moving activities are Completely different

- The first feature that we are considering is tBodyAccMagMean, and we can clearly see from the graph obtained below that there is a distinction between the Static Activities and Dynamic Activities.
- **This feature alone can clearly separate our Static Activities from the Dynamic Activities.**

In [12]:
```
# Here we are considering the tBodyAccMagmean.

sns.set_palette("Set1", desat=0.80)
facetgrid = sns.FacetGrid(train_final, hue='ActivityName', size=6,aspec
t=2)
facetgrid.map(sns.distplot,'tBodyAccMagmean', hist=False)\
    .add_legend()
plt.annotate("Stationary Activities", xy=(-0.956,17), xytext=(-0.9, 23
), size=20,\
            va='center', ha='left',\
            arrowprops=dict(arrowstyle="simple",connectionstyle="arc3,r
ad=0.1"))
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

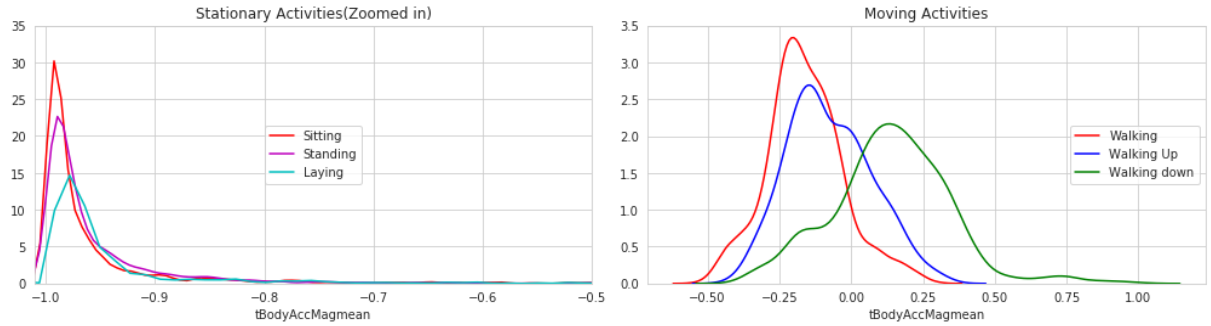
```
plt.annotate("Moving Activities", xy=(0,3), xytext=(0.2, 9), size=20,\
             va='center', ha='left',\
             arrowprops=dict(arrowstyle="simple",connectionstyle="arc3,r
ad=0.1"))
plt.show()
```



In [13]:
```
# for plotting purposes taking datapoints of each activity to a differe
nt dataframe
df1 = train_final[train_final['Activity']==1]
df2 = train_final[train_final['Activity']==2]
df3 = train_final[train_final['Activity']==3]
df4 = train_final[train_final['Activity']==4]
df5 = train_final[train_final['Activity']==5]
df6 = train_final[train_final['Activity']==6]

plt.figure(figsize=(14,7))
plt.subplot(2,2,1)
plt.title('Stationary Activities(Zoomed in)')
sns.distplot(df4['tBodyAccMagmean'],color = 'r',hist = False, label =
'Sitting')
sns.distplot(df5['tBodyAccMagmean'],color = 'm',hist = False,label = 'S
tanding')
sns.distplot(df6['tBodyAccMagmean'],color = 'c',hist = False, label =
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
'Laying')
plt.axis([-1.01, -0.5, 0, 35])
plt.legend(loc='center')

plt.subplot(2,2,2)
plt.title('Moving Activities')
sns.distplot(df1['tBodyAccMagmean'],color = 'red',hist = False, label =
 'Walking')
sns.distplot(df2['tBodyAccMagmean'],color = 'blue',hist = False,label =
 'Walking Up')
sns.distplot(df3['tBodyAccMagmean'],color = 'green',hist = False, label
 = 'Walking down')
plt.legend(loc='center right')


plt.tight_layout()
plt.show()
```



Observations

- If we zoom in to the plots further for both the stationary as well as the dynamic activities, we are able to see that the Stationary Activities have a tBodyAccMagmean values typically in the range of -1 to -0.8, but the same feature, once we zoom in for the Dynamic Activities ranges in between -0.5 and 1.0.

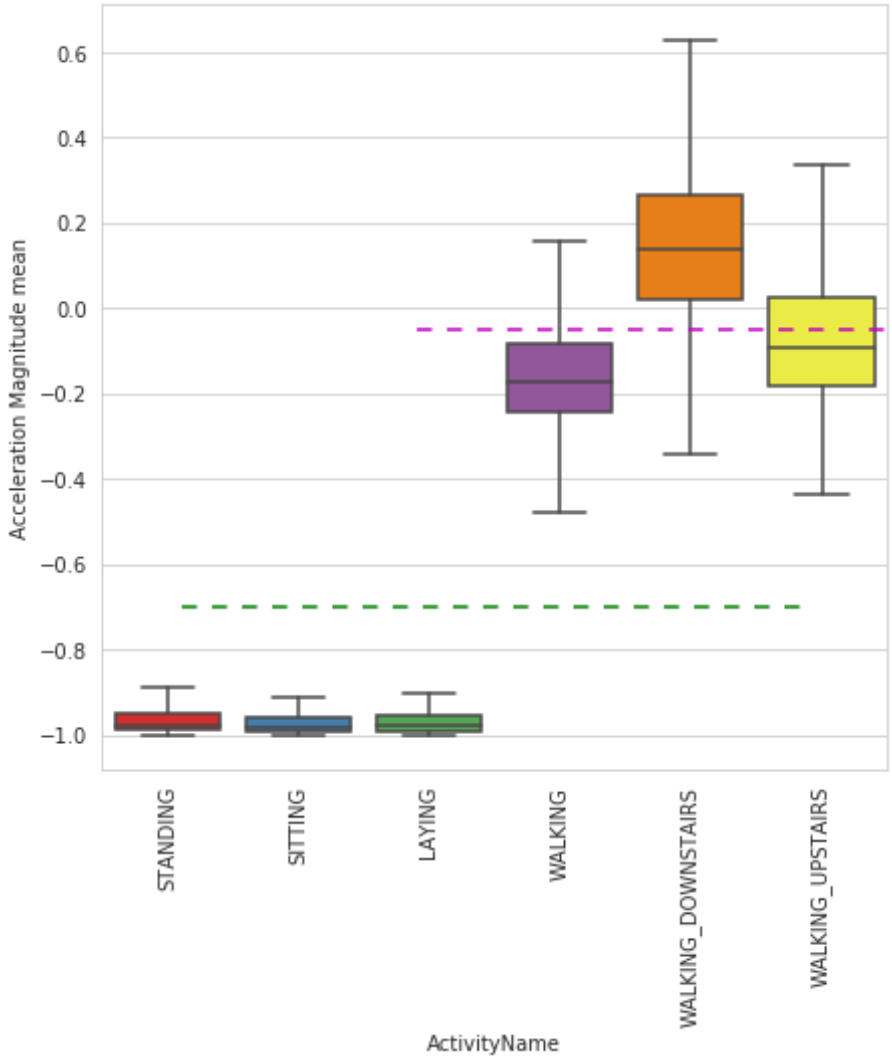
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

3.3 Magnitude of an acceleration can seperate it well

- We construct the Boxplot for each of these activities on the same feature and we are able to notice the clear separation among the activities as shown below.

In [14]:
```
plt.figure(figsize=(7,7))
sns.boxplot(x='ActivityName', y='tBodyAccMagmean',data=train_final, sho
wfliers=False, saturation=1)
plt.ylabel('Acceleration Magnitude mean')
plt.axhline(y=-0.7, xmin=0.1, xmax=0.9,dashes=(5,5), c='g')
plt.axhline(y=-0.05, xmin=0.4, dashes=(5,5), c='m')
plt.xticks(rotation=90)
plt.show()
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



Observations:

- If tBodyAccMagMean is < -0.8, then the Activities are either Standing or Sitting or Laying (Stationary Activities).
- If tBodyAccMagMean is > -0.6, then the Activities are either Walking or Walking_Downstairs or Walking_Upstairs (Dynamic Activities).

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

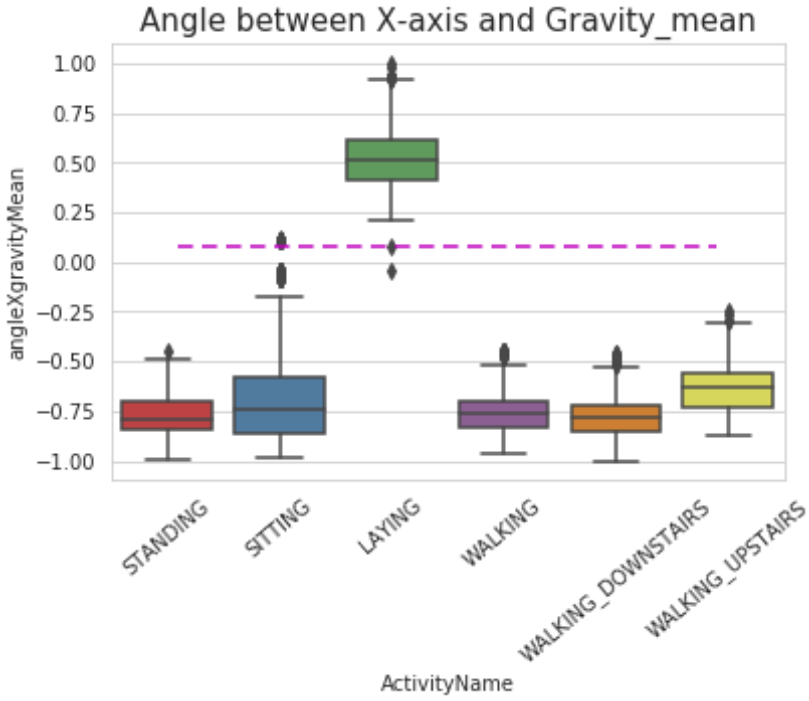
- If tBodyAccMagMean > 0.0, then the Activity is Walking_Downstairs.
- We can classify 75% the Acitivity labels with some errors.

Since the classifications are possible fairly well with the presence of a single feature, it means that these features are well designed by the Experts.

3.4 Position of GravityAccelerationComponents also matters

We are performing Univariate Analysis for 'angleXgravityMean'.

In [15]:
```
sns.boxplot(x='ActivityName', y='angleXgravityMean', data=train_final)
plt.axhline(y=0.08, xmin=0.1, xmax=0.9,c='m',dashes=(5,3))
plt.title('Angle between X-axis and Gravity_mean', fontsize=15)
plt.xticks(rotation = 40)
plt.show()
```



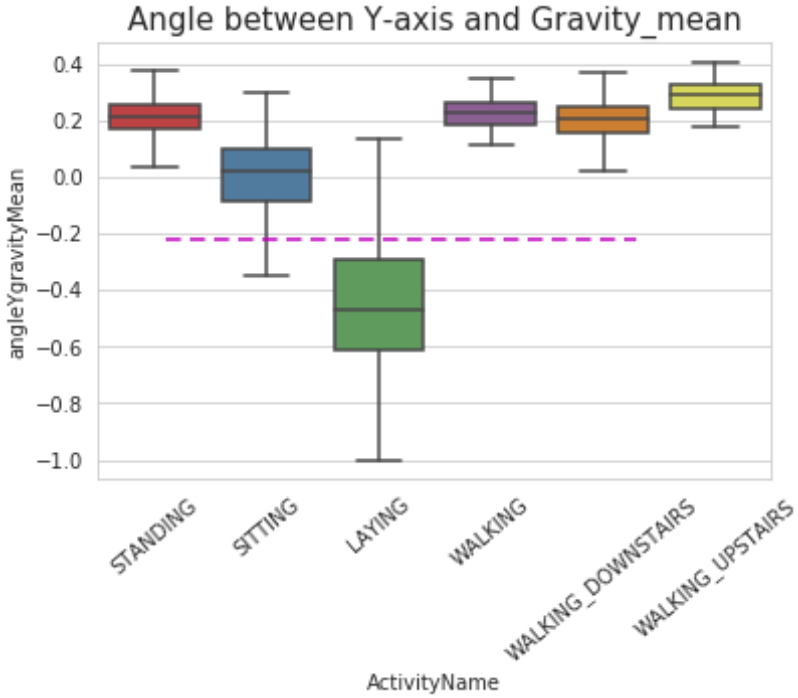
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Observations:

- We can classify all the datapoints belonging to the 'Laying activity' with just a single if else statement and using this feature of 'angleXgravityMean'.
- If angleXgravityMean > 0, then the Activity is Laying.

Univariate Analysis for angleYgravityMean

In [16]:
```
sns.boxplot(x='ActivityName', y='angleYgravityMean', data = train_final
, showfliers=False)
plt.title('Angle between Y-axis and Gravity_mean', fontsize=15)
plt.xticks(rotation = 40)
plt.axhline(y=-0.22, xmin=0.1, xmax=0.8, dashes=(5,3), c='m')
plt.show()
```



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Observations:

- With the help of this feature also, we can notice that we can very well separate 'Laying Down' as an activity from the other activities.

Even with simple Univariate Analysis, we are able to find some features to work very well for our Classification Task.

4. Applying T-SNE on the Data

- Note that the entire data that we have is in 561 dimensions, and in order to visualize data in such a case, we need to carry out dimensionality reduction. This is what we try and achieve with the help of T-SNE.
- Since we have 561 Dimensional Representation of Expert Engineered Features, and as we have seen above in the Univariate Analysis, that some single features themselves are able to partially separate the class labels, we will try and understand what we can achieve by incorporating all the 561 dimensions, the tool to achieve this being T-SNE.
- We are plotting into a 2-Dimensional space.

In [17]:
```
import numpy as np
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
import seaborn as sns
```

In [18]:
```
# performs t-sne with different perplexity values and their repective p
lots..

def perform_tsne(X_data, y_data, perplexities, n_iter=1000, img_name_pr
efix='t-sne'):

    for index,perplexity in enumerate(perplexities):
        # perform t-sne
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        print('\nperforming tsne with perplexity {} and with {} iterati
ons at max'.format(perplexity, n_iter))
        X_reduced = TSNE(verbose=2, perplexity=perplexity).fit_transfor
m(X_data)
        print('Done..')

        # prepare the data for seaborn
        print('Creating plot for this t-sne visualization..')
        df = pd.DataFrame({'x':X_reduced[:,0], 'y':X_reduced[:,1] ,'lab
el':y_data})

        # draw the plot in appropriate place in the grid
        sns.lmplot(data=df, x='x', y='y', hue='label', fit_reg=False, s
ize=8,\
                   palette="Set1",markers=['^','v','s','o', '1','2'])
        plt.title("perplexity : {} and max_iter : {}".format(perplexity
, n_iter))
        img_name = img_name_prefix + '_perp_{}_iter_{}.png'.format(perp
lexity, n_iter)
        print('saving this plot as image in present working director
y...')
        plt.savefig(img_name)
        plt.show()
        print('Done')
```

In [19]:

```
X_pre_tsne = train_final.drop(['subject', 'Activity','ActivityName'], a
xis=1)
y_pre_tsne = train_final['ActivityName']
perform_tsne(X_data = X_pre_tsne,y_data=y_pre_tsne, perplexities =[2,5,
10,20,50])
```

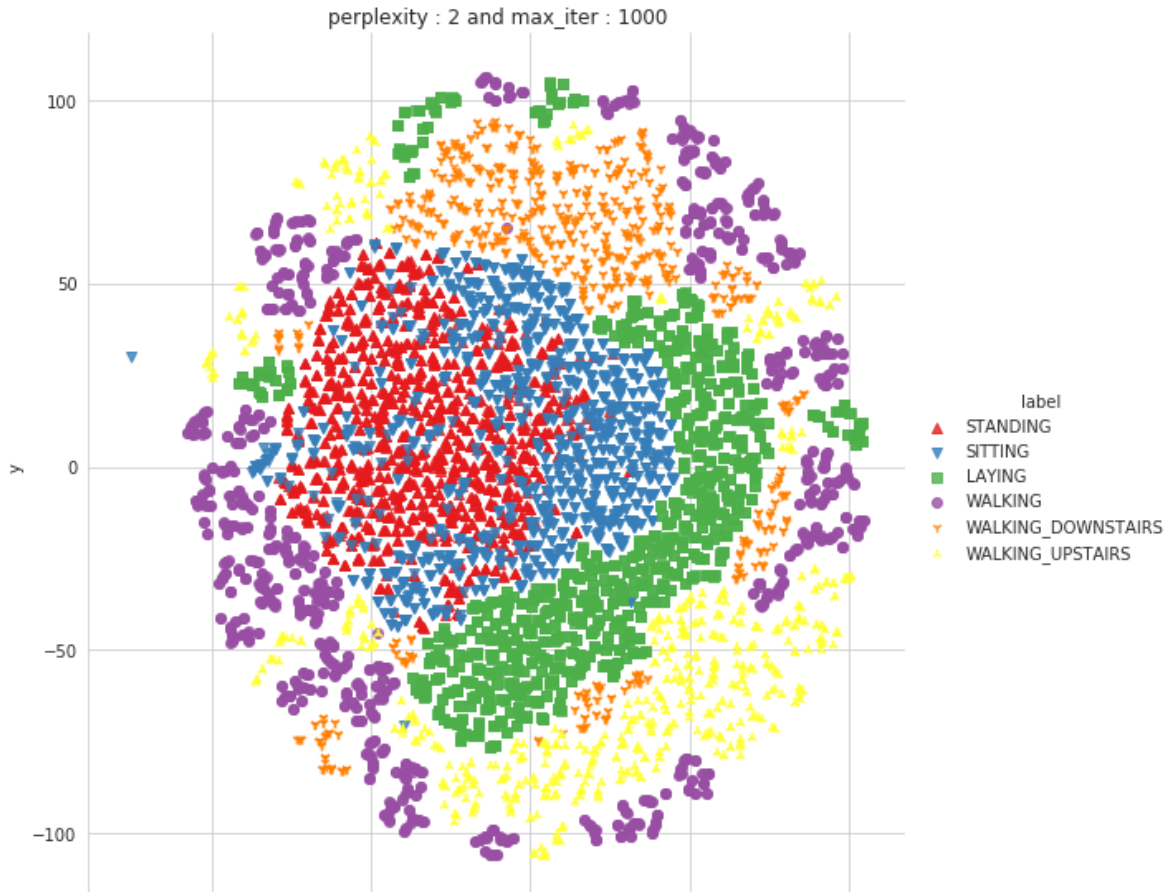
```
performing tsne with perplexity 2 and with 1000 iterations at max
[t-SNE] Computing 7 nearest neighbors...
[t-SNE] Indexed 7352 samples in 0.224s...
[t-SNE] Computed neighbors for 7352 samples in 32.840s...
[t-SNE] Computed conditional probabilities for sample 1000 / 7352
[t-SNE] Computed conditional probabilities for sample 2000 / 7352
[t-SNE] Computed conditional probabilities for sample 3000 / 7352
[t-SNE] Computed conditional probabilities for sample 4000 / 7352
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
[t-SNE] Computed conditional probabilities for sample 5000 / 7352
[t-SNE] Computed conditional probabilities for sample 6000 / 7352
[t-SNE] Computed conditional probabilities for sample 7000 / 7352
[t-SNE] Computed conditional probabilities for sample 7352 / 7352
[t-SNE] Mean sigma: 0.635855
[t-SNE] Computed conditional probabilities in 0.044s
[t-SNE] Iteration 50: error = 124.7860336, gradient norm = 0.0235710
(50 iterations in 9.768s)
[t-SNE] Iteration 100: error = 107.6125717, gradient norm = 0.0273096
(50 iterations in 3.940s)
[t-SNE] Iteration 150: error = 101.4506836, gradient norm = 0.0200841
(50 iterations in 2.864s)
[t-SNE] Iteration 200: error = 98.0484772, gradient norm = 0.0161042
(50 iterations in 2.624s)
[t-SNE] Iteration 250: error = 95.6122589, gradient norm = 0.0145228
(50 iterations in 2.569s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 9
5.612259
[t-SNE] Iteration 300: error = 4.1219406, gradient norm = 0.0015629
(50 iterations in 2.363s)
[t-SNE] Iteration 350: error = 3.2119935, gradient norm = 0.0009958
(50 iterations in 2.179s)
[t-SNE] Iteration 400: error = 2.7841446, gradient norm = 0.0007135
(50 iterations in 2.237s)
[t-SNE] Iteration 450: error = 2.5203629, gradient norm = 0.0005684
(50 iterations in 2.258s)
[t-SNE] Iteration 500: error = 2.3376608, gradient norm = 0.0004804
(50 iterations in 2.281s)
[t-SNE] Iteration 550: error = 2.1999509, gradient norm = 0.0004218
(50 iterations in 2.324s)
[t-SNE] Iteration 600: error = 2.0904710, gradient norm = 0.0003688
(50 iterations in 2.439s)
[t-SNE] Iteration 650: error = 2.0005980, gradient norm = 0.0003324
(50 iterations in 2.393s)
[t-SNE] Iteration 700: error = 1.9250830, gradient norm = 0.0002976
(50 iterations in 2.371s)
[t-SNE] Iteration 750: error = 1.8601131, gradient norm = 0.0002767
(50 iterations in 2.531s)
[t-SNE] Iteration 800: error = 1.8033799, gradient norm = 0.0002605
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
(50 iterations in 2.371s)
[t-SNE] Iteration 850: error = 1.7534183, gradient norm = 0.0002422
(50 iterations in 2.424s)
[t-SNE] Iteration 900: error = 1.7088405, gradient norm = 0.0002261
(50 iterations in 2.475s)
[t-SNE] Iteration 950: error = 1.6687087, gradient norm = 0.0002126
(50 iterations in 2.506s)
[t-SNE] Iteration 1000: error = 1.6325023, gradient norm = 0.0002006
(50 iterations in 2.390s)
[t-SNE] KL divergence after 1000 iterations: 1.632502
Done..
Creating plot for this t-sne visualization..
saving this plot as image in present working directory...
```

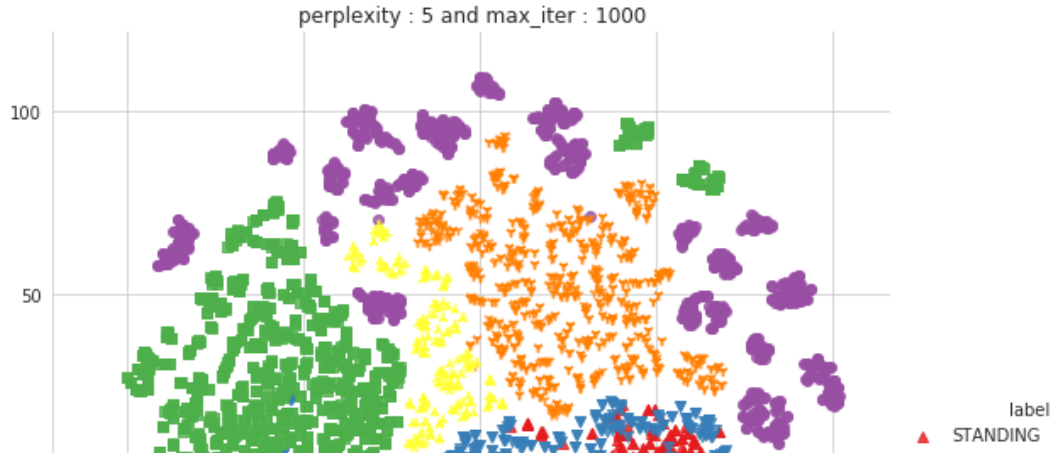


Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

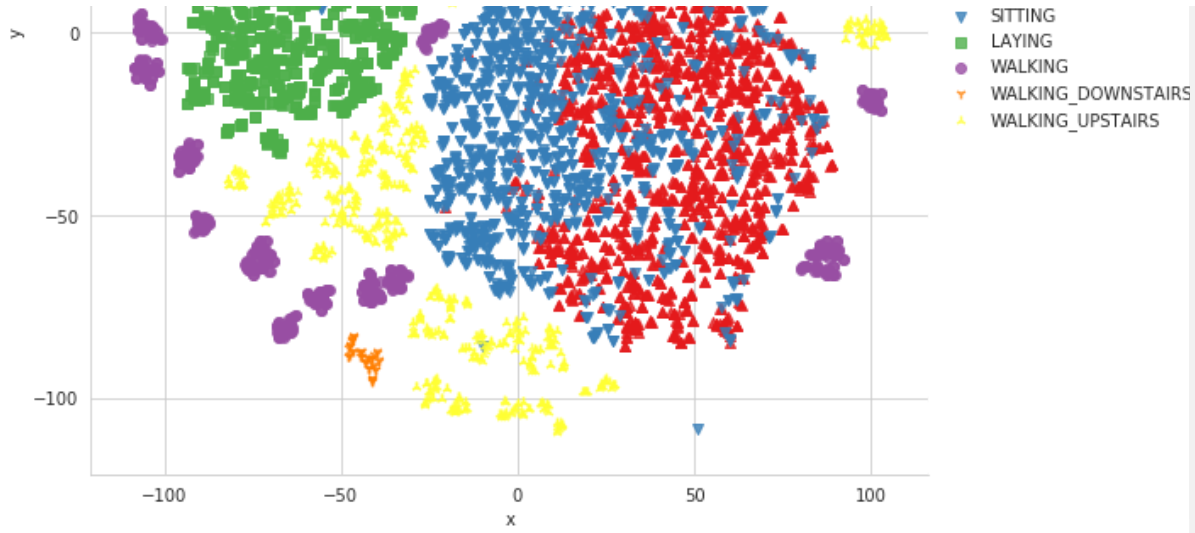
```
Done

performing tsne with perplexity 5 and with 1000 iterations at max
[t-SNE] Computing 16 nearest neighbors...
[t-SNE] Indexed 7352 samples in 0.199s...
[t-SNE] Computed neighbors for 7352 samples in 33.530s...
[t-SNE] Computed conditional probabilities for sample 1000 / 7352
[t-SNE] Computed conditional probabilities for sample 2000 / 7352
[t-SNE] Computed conditional probabilities for sample 3000 / 7352
[t-SNE] Computed conditional probabilities for sample 4000 / 7352
[t-SNE] Computed conditional probabilities for sample 5000 / 7352
[t-SNE] Computed conditional probabilities for sample 6000 / 7352
[t-SNE] Computed conditional probabilities for sample 7000 / 7352
[t-SNE] Computed conditional probabilities for sample 7352 / 7352
[t-SNE] Mean sigma: 0.961265
[t-SNE] Computed conditional probabilities in 0.046s
[t-SNE] Iteration 50: error = 113.9008026, gradient norm = 0.0236872
(50 iterations in 6.468s)
[t-SNE] Iteration 100: error = 97.1602020, gradient norm = 0.0133891
(50 iterations in 2.862s)
[t-SNE] Iteration 150: error = 93.0010147, gradient norm = 0.0139940
(50 iterations in 2.379s)
[t-SNE] Iteration 200: error = 91.0864334, gradient norm = 0.0066107
(50 iterations in 2.339s)
[t-SNE] Iteration 250: error = 89.9553299, gradient norm = 0.0054566
(50 iterations in 2.330s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 8
9.955330
[t-SNE] Iteration 300: error = 3.5711365, gradient norm = 0.0014622
(50 iterations in 2.200s)
[t-SNE] Iteration 350: error = 2.8137655, gradient norm = 0.0007491
(50 iterations in 2.155s)
[t-SNE] Iteration 400: error = 2.4332302, gradient norm = 0.0005217
(50 iterations in 2.199s)
[t-SNE] Iteration 450: error = 2.2151649, gradient norm = 0.0004046
(50 iterations in 2.260s)
[t-SNE] Iteration 500: error = 2.0707734, gradient norm = 0.0003312
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
(50 iterations in 2.231s)
[t-SNE] Iteration 550: error = 1.9655740, gradient norm = 0.0002811
(50 iterations in 2.219s)
[t-SNE] Iteration 600: error = 1.8845799, gradient norm = 0.0002446
(50 iterations in 2.220s)
[t-SNE] Iteration 650: error = 1.8195461, gradient norm = 0.0002194
(50 iterations in 2.268s)
[t-SNE] Iteration 700: error = 1.7658883, gradient norm = 0.0001978
(50 iterations in 2.292s)
[t-SNE] Iteration 750: error = 1.7204863, gradient norm = 0.0001810
(50 iterations in 2.244s)
[t-SNE] Iteration 800: error = 1.6816058, gradient norm = 0.0001642
(50 iterations in 2.236s)
[t-SNE] Iteration 850: error = 1.6475700, gradient norm = 0.0001534
(50 iterations in 2.241s)
[t-SNE] Iteration 900: error = 1.6175040, gradient norm = 0.0001428

(50 iterations in 2.256s)
[t-SNE] Iteration 950: error = 1.5909351, gradient norm = 0.0001344
(50 iterations in 2.260s)
[t-SNE] Iteration 1000: error = 1.5671402, gradient norm = 0.0001263
(50 iterations in 2.262s)
[t-SNE] KL divergence after 1000 iterations: 1.567140
Done..
Creating plot for this t-sne visualization..
saving this plot as image in present working directory...
```



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

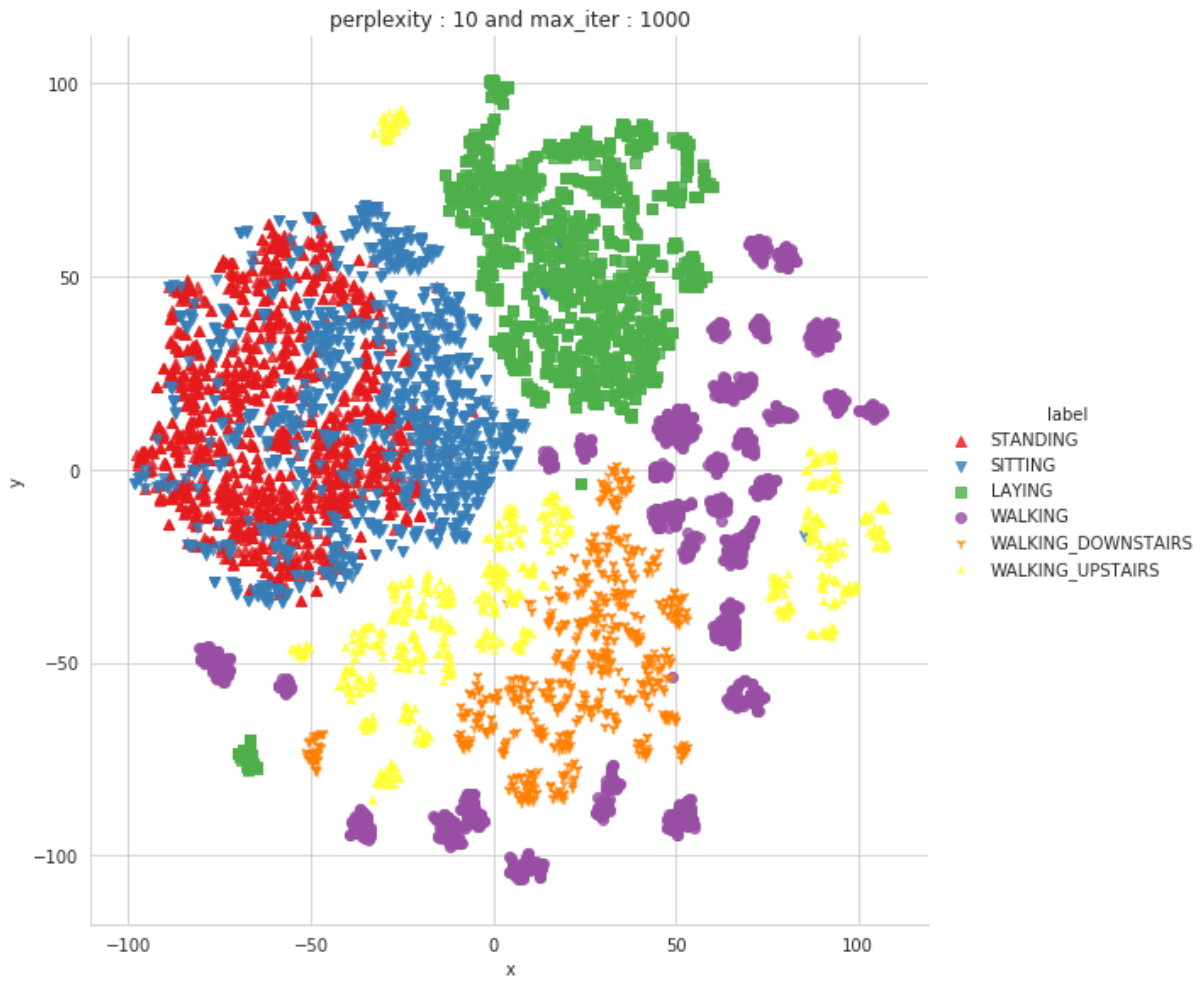



```
Done

performing tsne with perplexity 10 and with 1000 iterations at max
[t-SNE] Computing 31 nearest neighbors...
[t-SNE] Indexed 7352 samples in 0.165s...
[t-SNE] Computed neighbors for 7352 samples in 32.799s...
[t-SNE] Computed conditional probabilities for sample 1000 / 7352
[t-SNE] Computed conditional probabilities for sample 2000 / 7352
[t-SNE] Computed conditional probabilities for sample 3000 / 7352
[t-SNE] Computed conditional probabilities for sample 4000 / 7352
[t-SNE] Computed conditional probabilities for sample 5000 / 7352
[t-SNE] Computed conditional probabilities for sample 6000 / 7352
[t-SNE] Computed conditional probabilities for sample 7000 / 7352
[t-SNE] Computed conditional probabilities for sample 7352 / 7352
[t-SNE] Mean sigma: 1.133828
[t-SNE] Computed conditional probabilities in 0.079s
[t-SNE] Iteration 50: error = 105.4497910, gradient norm = 0.0228234
(50 iterations in 4.020s)
[t-SNE] Iteration 100: error = 90.6357803, gradient norm = 0.0116685
(50 iterations in 2.816s)
[t-SNE] Iteration 150: error = 87.4024963, gradient norm = 0.0057445
(50 iterations in 2.478s)
[t-SNE] Iteration 200: error = 86.1237183, gradient norm = 0.0040316
(50 iterations in 2.376s)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
[t-SNE] Iteration 250: error = 85.4047699, gradient norm = 0.0032668
(50 iterations in 2.407s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 8
5.404770
[t-SNE] Iteration 300: error = 3.1312881, gradient norm = 0.0013802
(50 iterations in 2.390s)
[t-SNE] Iteration 350: error = 2.4905443, gradient norm = 0.0006502
(50 iterations in 2.327s)
[t-SNE] Iteration 400: error = 2.1710591, gradient norm = 0.0004228
(50 iterations in 2.300s)
[t-SNE] Iteration 450: error = 1.9866132, gradient norm = 0.0003137
(50 iterations in 2.315s)
[t-SNE] Iteration 500: error = 1.8680816, gradient norm = 0.0002501
(50 iterations in 2.333s)
[t-SNE] Iteration 550: error = 1.7847890, gradient norm = 0.0002082
(50 iterations in 2.411s)
[t-SNE] Iteration 600: error = 1.7222850, gradient norm = 0.0001797
(50 iterations in 2.380s)
[t-SNE] Iteration 650: error = 1.6733613, gradient norm = 0.0001600
(50 iterations in 2.364s)
[t-SNE] Iteration 700: error = 1.6338769, gradient norm = 0.0001409
(50 iterations in 2.391s)
[t-SNE] Iteration 750: error = 1.6013981, gradient norm = 0.0001310
(50 iterations in 2.389s)
[t-SNE] Iteration 800: error = 1.5740784, gradient norm = 0.0001187
(50 iterations in 2.415s)
[t-SNE] Iteration 850: error = 1.5511621, gradient norm = 0.0001109
(50 iterations in 2.364s)
[t-SNE] Iteration 900: error = 1.5315396, gradient norm = 0.0001043
(50 iterations in 2.376s)
[t-SNE] Iteration 950: error = 1.5146669, gradient norm = 0.0000967
(50 iterations in 2.447s)
[t-SNE] Iteration 1000: error = 1.4994367, gradient norm = 0.0000915
(50 iterations in 2.363s)
[t-SNE] KL divergence after 1000 iterations: 1.499437
Done..
Creating plot for this t-sne visualization..
saving this plot as image in present working directory...
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

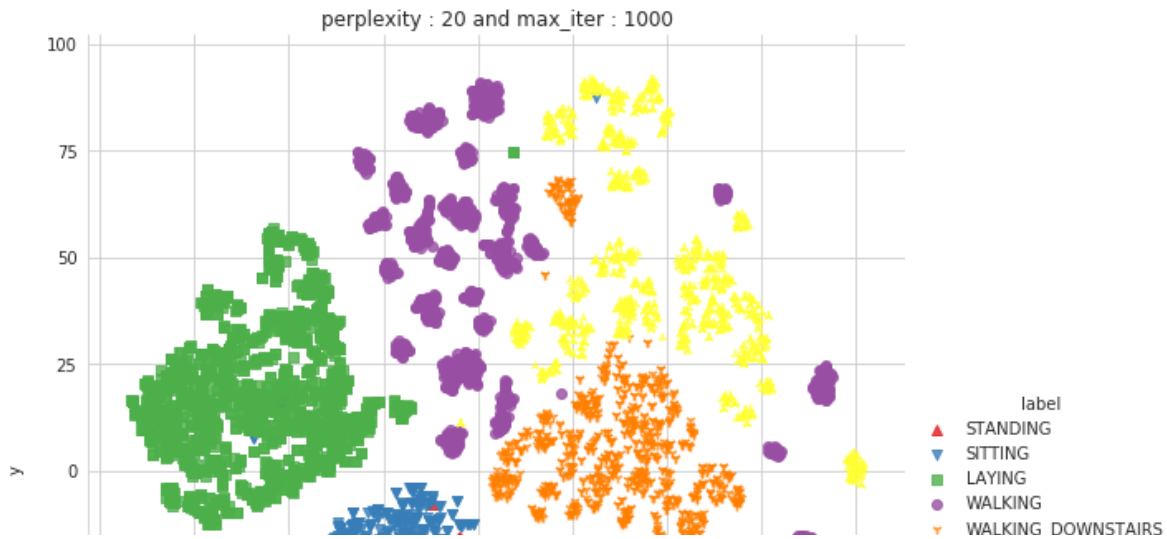


Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

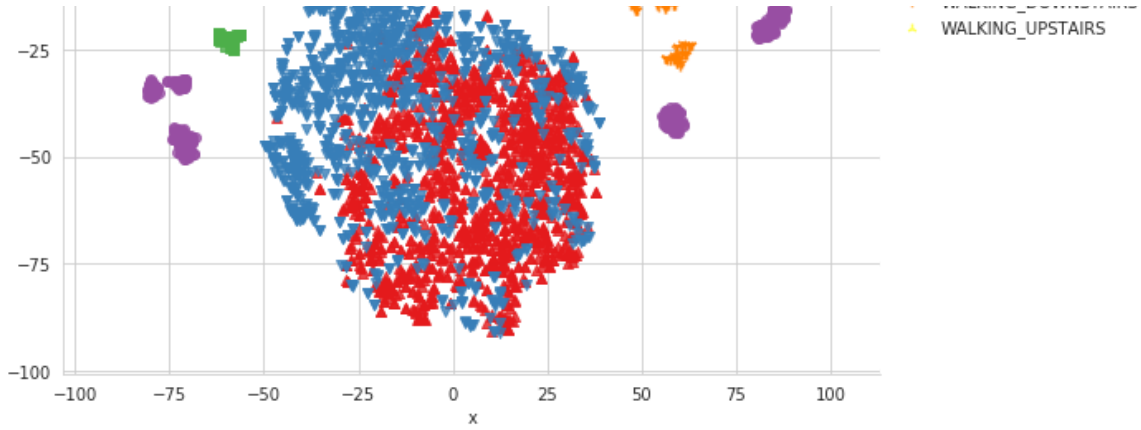
```
Done

performing tsne with perplexity 20 and with 1000 iterations at max
[t-SNE] Computing 61 nearest neighbors...
[t-SNE] Indexed 7352 samples in 0.167s...
[t-SNE] Computed neighbors for 7352 samples in 33.568s...
[t-SNE] Computed conditional probabilities for sample 1000 / 7352
[t-SNE] Computed conditional probabilities for sample 2000 / 7352
[t-SNE] Computed conditional probabilities for sample 3000 / 7352
[t-SNE] Computed conditional probabilities for sample 4000 / 7352
[t-SNE] Computed conditional probabilities for sample 5000 / 7352
[t-SNE] Computed conditional probabilities for sample 6000 / 7352
[t-SNE] Computed conditional probabilities for sample 7000 / 7352
[t-SNE] Computed conditional probabilities for sample 7352 / 7352
[t-SNE] Mean sigma: 1.274335
[t-SNE] Computed conditional probabilities in 0.148s
[t-SNE] Iteration 50: error = 97.1190872, gradient norm = 0.0234644
(50 iterations in 4.900s)
[t-SNE] Iteration 100: error = 84.2793961, gradient norm = 0.0077311
(50 iterations in 3.341s)

[t-SNE] Iteration 150: error = 82.0703659, gradient norm = 0.0038934
(50 iterations in 2.759s)
[t-SNE] Iteration 200: error = 81.2492828, gradient norm = 0.0026572
(50 iterations in 2.775s)
[t-SNE] Iteration 250: error = 80.8303452, gradient norm = 0.0018988
(50 iterations in 2.657s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 8
0.830345
[t-SNE] Iteration 300: error = 2.7032905, gradient norm = 0.0013060
(50 iterations in 2.521s)
[t-SNE] Iteration 350: error = 2.1673834, gradient norm = 0.0005746
(50 iterations in 2.567s)
[t-SNE] Iteration 400: error = 1.9181011, gradient norm = 0.0003465
(50 iterations in 2.564s)
[t-SNE] Iteration 450: error = 1.7719345, gradient norm = 0.0002483
(50 iterations in 2.492s)
[t-SNE] Iteration 500: error = 1.6781938, gradient norm = 0.0001931
(50 iterations in 2.500s)
[t-SNE] Iteration 550: error = 1.6143360, gradient norm = 0.0001580
(50 iterations in 2.554s)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
(50 iterations in 2.554s)
[t-SNE] Iteration 600: error = 1.5679270, gradient norm = 0.0001335
(50 iterations in 2.556s)
[t-SNE] Iteration 650: error = 1.5326270, gradient norm = 0.0001159
(50 iterations in 2.618s)
[t-SNE] Iteration 700: error = 1.5050586, gradient norm = 0.0001054
(50 iterations in 2.663s)
[t-SNE] Iteration 750: error = 1.4830132, gradient norm = 0.0000967
(50 iterations in 2.598s)
[t-SNE] Iteration 800: error = 1.4650533, gradient norm = 0.0000889
(50 iterations in 2.590s)
[t-SNE] Iteration 850: error = 1.4501868, gradient norm = 0.0000850
(50 iterations in 2.550s)
[t-SNE] Iteration 900: error = 1.4379947, gradient norm = 0.0000784
(50 iterations in 2.615s)
[t-SNE] Iteration 950: error = 1.4278527, gradient norm = 0.0000736
(50 iterations in 2.576s)
[t-SNE] Iteration 1000: error = 1.4191968, gradient norm = 0.0000710
(50 iterations in 2.543s)
[t-SNE] KL divergence after 1000 iterations: 1.419197

Done..
Creating plot for this t-sne visualization..
saving this plot as image in present working directory...
```



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

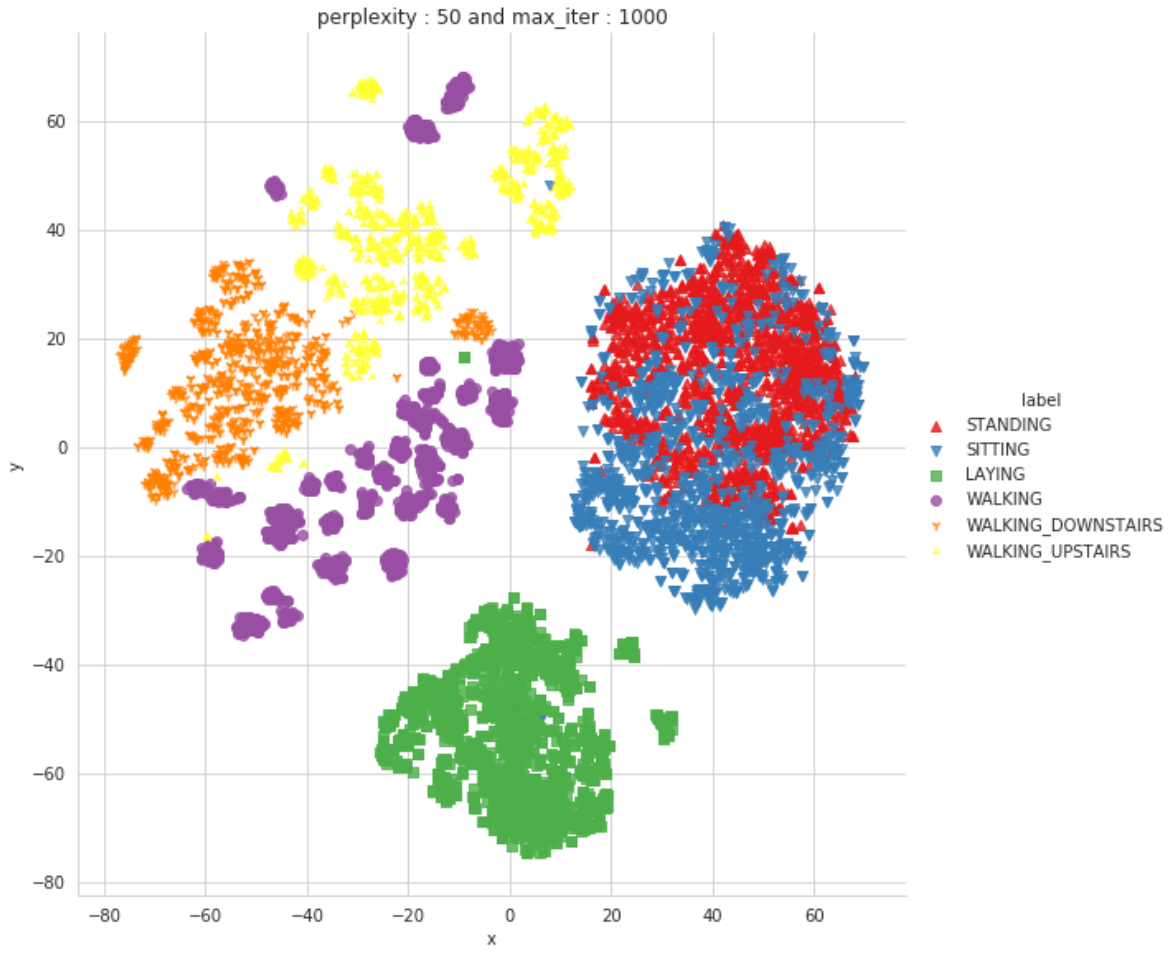



```
Done

performing tsne with perplexity 50 and with 1000 iterations at max
[t-SNE] Computing 151 nearest neighbors...
[t-SNE] Indexed 7352 samples in 0.161s...
[t-SNE] Computed neighbors for 7352 samples in 34.526s...
[t-SNE] Computed conditional probabilities for sample 1000 / 7352
[t-SNE] Computed conditional probabilities for sample 2000 / 7352
[t-SNE] Computed conditional probabilities for sample 3000 / 7352
[t-SNE] Computed conditional probabilities for sample 4000 / 7352
[t-SNE] Computed conditional probabilities for sample 5000 / 7352
[t-SNE] Computed conditional probabilities for sample 6000 / 7352
[t-SNE] Computed conditional probabilities for sample 7000 / 7352
[t-SNE] Computed conditional probabilities for sample 7352 / 7352
[t-SNE] Mean sigma: 1.437672
[t-SNE] Computed conditional probabilities in 0.351s
[t-SNE] Iteration 50: error = 85.8522568, gradient norm = 0.0255863
(50 iterations in 4.830s)
[t-SNE] Iteration 100: error = 75.5067062, gradient norm = 0.0042781
(50 iterations in 3.843s)
[t-SNE] Iteration 150: error = 74.5672226, gradient norm = 0.0021256
(50 iterations in 3.431s)
[t-SNE] Iteration 200: error = 74.2281189, gradient norm = 0.0015189
(50 iterations in 3.514s)
[t-SNE] Iteration 250: error = 74.0547028, gradient norm = 0.0016761
(50 iterations in 3.625s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 7
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
[t-SNE] KL divergence after 250 iterations with early exaggeration: 7
4.054703
[t-SNE] Iteration 300: error = 2.1561179, gradient norm = 0.0011883
(50 iterations in 3.375s)
[t-SNE] Iteration 350: error = 1.7569965, gradient norm = 0.0004851
(50 iterations in 3.123s)
[t-SNE] Iteration 400: error = 1.5868489, gradient norm = 0.0002925
(50 iterations in 3.160s)
[t-SNE] Iteration 450: error = 1.4921901, gradient norm = 0.0001922
(50 iterations in 3.140s)
[t-SNE] Iteration 500: error = 1.4325324, gradient norm = 0.0001420
(50 iterations in 3.117s)
[t-SNE] Iteration 550: error = 1.3913937, gradient norm = 0.0001125
(50 iterations in 3.122s)
[t-SNE] Iteration 600: error = 1.3620505, gradient norm = 0.0000947
(50 iterations in 3.120s)
[t-SNE] Iteration 650: error = 1.3407662, gradient norm = 0.0000835
(50 iterations in 3.130s)
[t-SNE] Iteration 700: error = 1.3251126, gradient norm = 0.0000767
(50 iterations in 3.135s)
[t-SNE] Iteration 750: error = 1.3135396, gradient norm = 0.0000737
(50 iterations in 3.320s)
[t-SNE] Iteration 800: error = 1.3045951, gradient norm = 0.0000681
(50 iterations in 3.186s)
[t-SNE] Iteration 850: error = 1.2976224, gradient norm = 0.0000605
(50 iterations in 3.137s)
[t-SNE] Iteration 900: error = 1.2917911, gradient norm = 0.0000593
(50 iterations in 3.144s)
[t-SNE] Iteration 950: error = 1.2867483, gradient norm = 0.0000588
(50 iterations in 3.134s)
[t-SNE] Iteration 1000: error = 1.2828871, gradient norm = 0.0000553
(50 iterations in 3.154s)
[t-SNE] KL divergence after 1000 iterations: 1.282887
Done..
Creating plot for this t-sne visualization..
saving this plot as image in present working directory...
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



Done

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Observations:

- As we can see above, the best separated plot that we obtain is for Perplexity= 2 and when we have run the same for 1000 iterations.
- One of the first observations that we can make from this plot is that most of the points are well clustered together. **There is not too much of overlap among our datapoints except in the case of 'Sitting' & 'Standing'.** *There is a region where the blue and red points do not overlap, whereas there is also a considerable region where the overlap is present.*
- **We are able to infer from this T-SNE plot that we should be able to separate all the classes very well. But we may face some challenges to separate 'Sitting' and 'Standing', because they overlap as shown.**

Over here, we are using the 561 Expert Engineered Features and we will apply classical Machine Learning Models such as Logistic Regression, SVM etc.

In [20]:
```
# get X_train and y_train from csv files
X_train = train_final.drop(['subject', 'Activity', 'ActivityName'], axi
s=1)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
y_train = train_final.ActivityName

# get X_test and y_test from test csv file
X_test = test_final.drop(['subject', 'Activity', 'ActivityName'], axis=
1)
y_test = test_final.ActivityName

print('X_train and y_train : ({},{})'.format(X_train.shape, y_train.sha
pe))
print('X_test  and y_test  : ({},{})'.format(X_test.shape, y_test.shape
))
```

```
X_train and y_train : ((7352, 561),(7352,))
X_test  and y_test  : ((2947, 561),(2947,))
```

5. Let's Model with our Data

5.1 Labels that are useful in plotting confusion matrix

In [21]:
```
labels=['LAYING', 'SITTING','STANDING','WALKING','WALKING_DOWNSTAIRS',
'WALKING_UPSTAIRS']
```

5.2 Function to plot the confusion matrix

In [22]:
```
import itertools
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
plt.rcParams["font.family"] = 'DejaVu Sans'

#function to plot the confusion matrix
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
                                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')
```

5.3 Generic function to run any model specified

In [23]:
```
from datetime import datetime
def perform_model(model, X_train, y_train, X_test, y_test, class_labels
, cm_normalize=True, \
                 print_cm=True, cm_cmap=plt.cm.Greens):


    # to store results at various phases
    results = dict()

    # time at which model starts training
    train_start_time = datetime.now()
    print('Training the model..')
    model.fit(X_train, y_train)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    print('Done \n \n')
    train_end_time = datetime.now()
    results['training_time'] =  train_end_time - train_start_time
    print('Training Time (HH:MM:SS.ms) - {}\n\n'.format(results['traini
ng_time']))


    # predict test data
    print('Predicting Test Data')
    test_start_time = datetime.now()
    y_pred = model.predict(X_test)
    test_end_time = datetime.now()
    print('Done \n \n')
    results['testing_time'] = test_end_time - test_start_time
    print('Testing Time (HH:MM:SS:ms) - {}\n\n'.format(results['testing
_time']))
    results['predicted'] = y_pred


    # calculate overall accuracty of the model
    accuracy = metrics.accuracy_score(y_true=y_test, y_pred=y_pred)
    # store accuracy in results
    results['accuracy'] = accuracy
    print('---------------------')
    print('|      Accuracy      |')
    print('---------------------')
    print('\n    {}\n\n'.format(accuracy))


    # confusion matrix
    cm = metrics.confusion_matrix(y_test, y_pred)
    results['confusion_matrix'] = cm
    if print_cm:
        print('--------------------')
        print('| Confusion Matrix |')
        print('--------------------')
        print('\n {}'.format(cm))

    # plot confusin matrix
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    plt.figure(figsize=(8,8))
    plt.grid(b=False)
    plot_confusion_matrix(cm, classes=class_labels, normalize=True, tit
le='Normalized confusion matrix', cmap = cm_cmap)
    plt.show()

    # get classification report
    print('-------------------------')
    print('| Classifiction Report |')
    print('-------------------------')
    classification_report = metrics.classification_report(y_test, y_pre
d)
    # store report in results
    results['classification_report'] = classification_report
    print(classification_report)

    # add the trained  model to the results
    results['model'] = model

    return results
```

5.4 Method to print the GridSearch Attributes

In [24]:
```
def print_grid_search_attributes(model):
    # Estimator that gave highest score among all the estimators formed
 in GridSearch
    print('--------------------------')
    print('|      Best Estimator     |')
    print('--------------------------')
    print('\n\t{}\n'.format(model.best_estimator_))


    # parameters that gave best results while performing grid search
    print('--------------------------')
    print('|     Best Parameters     |')
    print('--------------------------')
    print('\tParameters of best estimator : \n\n\t{}\n'.format(model.be
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
st_params_))


    #  number of cross validation splits
    print('---------------------------------')
    print('|   No of Cross Validation sets   |')
    print('---------------------------------')
    print('\n\tTotal Number of Cross Validation sets: {}\n'.format(mode
l.n_splits_))


    # Average cross validated score of the best estimator, from the Gri
d Search
    print('--------------------------')
    print('|        Best Score       |')
    print('--------------------------')
    print('\n\tAverage Cross Validate scores of Best estimator : \n\n\t
{}\n'.format(model.best_score_))
```

6. Applying Machine Learning Models on the Data

6.1 Logistic Regression with Grid Search

In [25]:
```
from sklearn import linear_model
from sklearn import metrics

from sklearn.model_selection import GridSearchCV

# start Grid search
parameters = {'C':[0.01, 0.1, 1, 10, 20, 30], 'penalty':['l2','l1']}
log_reg = linear_model.LogisticRegression()
log_reg_grid = GridSearchCV(log_reg, param_grid=parameters, cv=3, verbo
se=1, n_jobs=-1)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

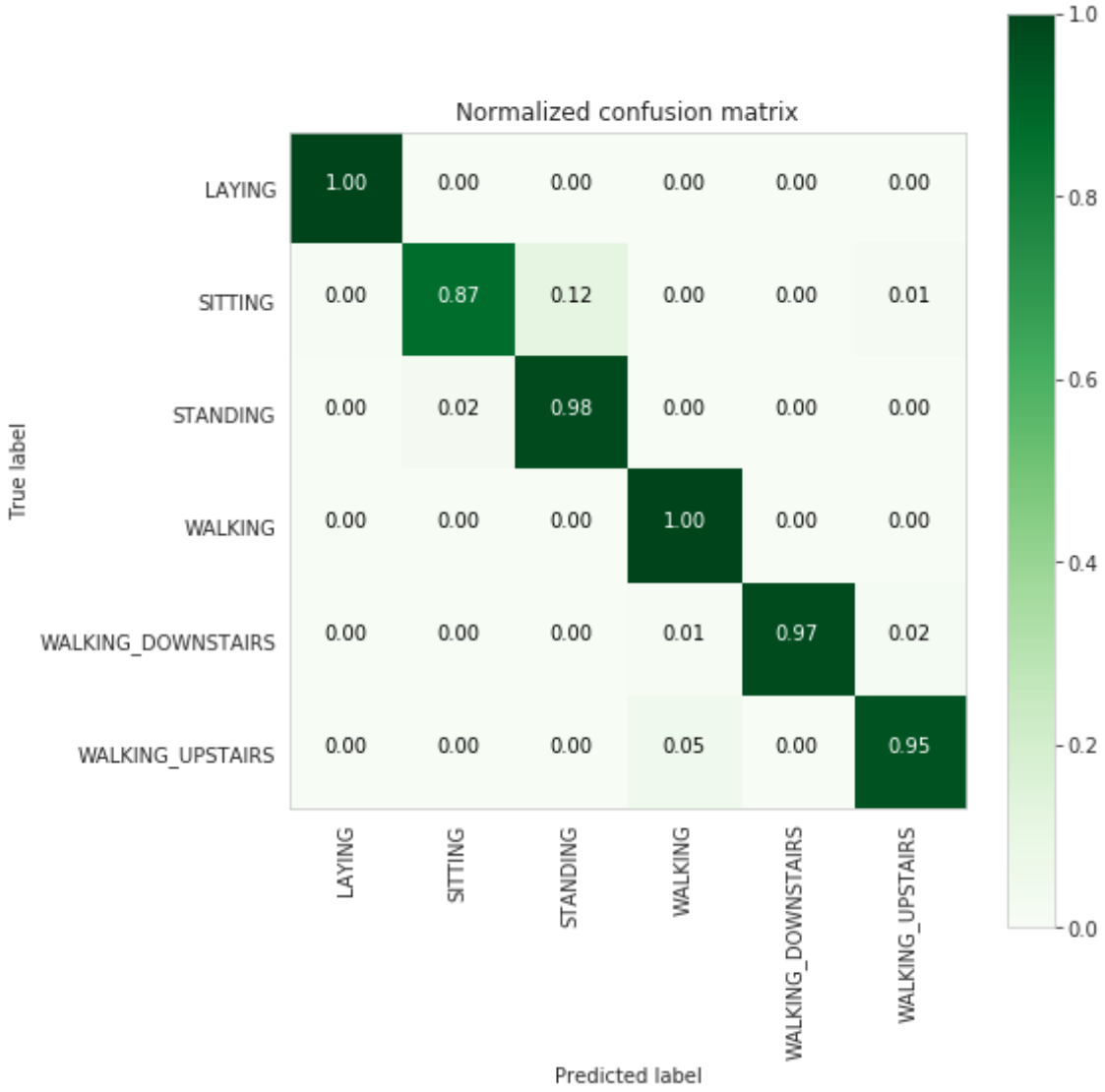
```
log_reg_grid_results =  perform_model(log_reg_grid, X_train, y_train, X
_test, y_test, class_labels=labels)
```

```
Training the model..
Fitting 3 folds for each of 12 candidates, totalling 36 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  36 out of  36 | elapsed:  1.4min finished
```

```
Done


Training Time (HH:MM:SS.ms) - 0:01:32.015959


Predicting Test Data
Done


Testing Time (HH:MM:SS:ms) - 0:00:00.006298


---------------------
|      Accuracy      |
---------------------

    0.9630132337970818


--------------------
| Confusion Matrix |
--------------------

 [[537   0   0   0   0   0]
 [  2 428  57   0   0   4]
 [  0  11 520   1   0   0]
 [  0   0   0 495   1   0]
 [  0   0   0   3 409   8]
 [  0   0   0  22   0 449]]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD




```
-------------------------
| Classifiction Report |
-------------------------
                    precision    recall  f1-score   support
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

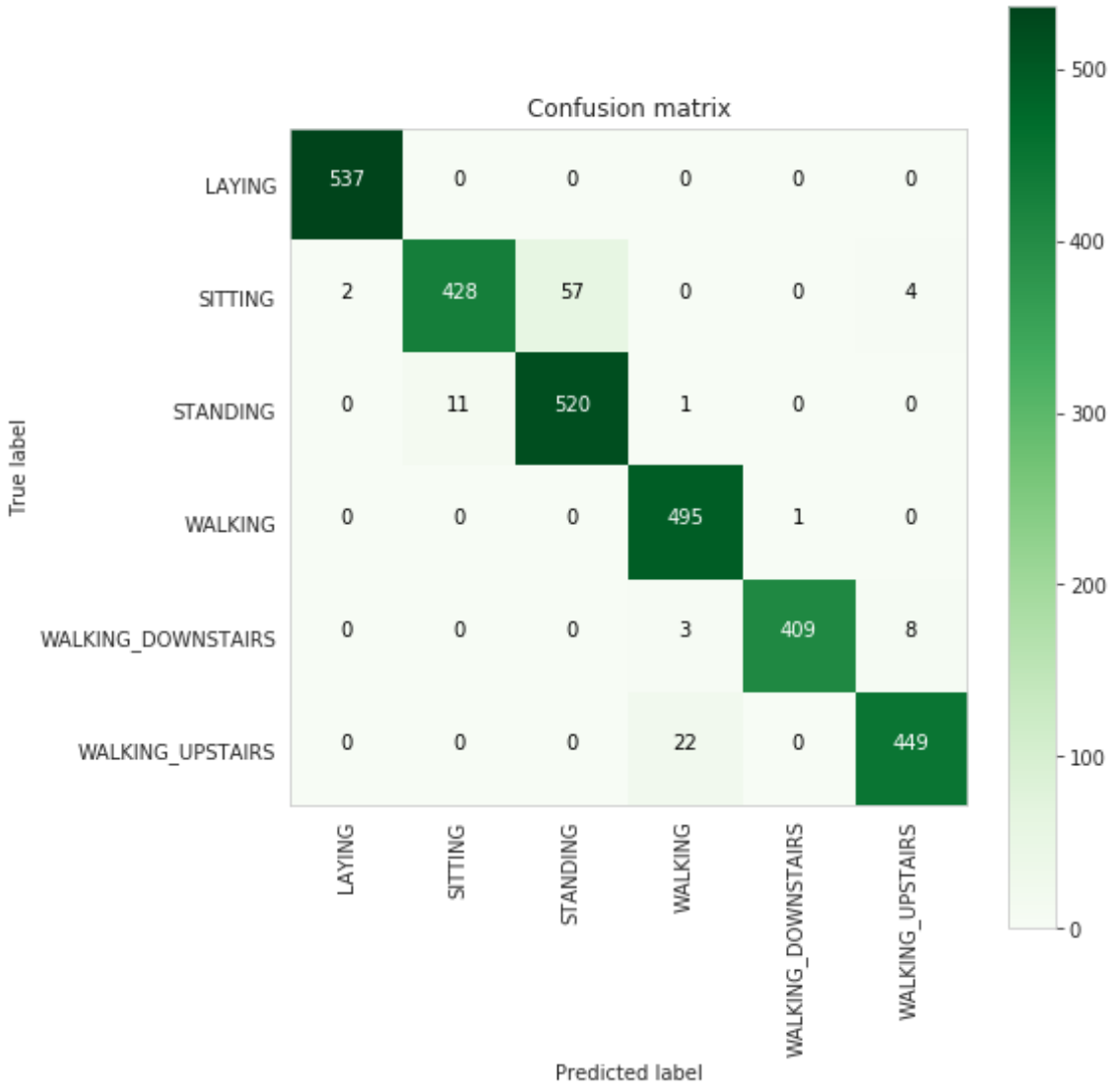
```
                    precision    recall  f1-score   support

            LAYING       1.00      1.00      1.00       537
           SITTING       0.97      0.87      0.92       491
          STANDING       0.90      0.98      0.94       532
           WALKING       0.95      1.00      0.97       496
WALKING_DOWNSTAIRS       1.00      0.97      0.99       420
  WALKING_UPSTAIRS       0.97      0.95      0.96       471

          accuracy                           0.96      2947
         macro avg       0.97      0.96      0.96      2947
      weighted avg       0.96      0.96      0.96      2947
```

Observations:

- In the ideal scenario, all of these diagonal elements in the confusion matrix should be 1, which basically means that over here we are able to detect the activity 'Laying' very well, with 100% Accuracy & 100% Recall : we are not confusing 'Laying' with any other class. **This is obvious because even with simple univariate analysis of some features, we could detect all the 'Laying'.**
- Similarly we can also see that 'Walking' is very easy to detect. Sitting is the only value which is bad : 0.87. This implies that although 87% of the points are classified correctly, about 12% of the datapoints, which actually belong to 'Sitting' class label, are being classified as 'Standing'. Even in the T-SNE plot, because of overlap among our class labels, 'Standing' and 'Sitting' were the 2 classes where we saw a lot of confusion.
- We can see that for 'Sitting' class our Recall is low whereas for the 'Standing' class, the Precision is low.

In [26]:
```
plt.figure(figsize=(8,8))
plt.grid(b=False)
plot_confusion_matrix(log_reg_grid_results['confusion_matrix'], classes
=labels, cmap=plt.cm.Greens, )
plt.show()
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



In [27]:
```
# observe the attributes of the model
print_grid_search_attributes(log_reg_grid_results['model'])
```

```
--------------------------
|      Best Estimator     |
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
--------------------------

        LogisticRegression(C=30, class_weight=None, dual=False, fit_int
ercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbos
e=0,
                   warm_start=False)

--------------------------
|     Best Parameters     |
--------------------------
        Parameters of best estimator :

        {'C': 30, 'penalty': 'l2'}

---------------------------------
|   No of Cross Validation sets   |
--------------------------------

        Total Number of Cross Validation sets: 3

--------------------------
|        Best Score       |
--------------------------

        Average Cross Validate scores of Best estimator :

        0.9461371055495104
```

6.2 Linear SVC with GridSearch

In [28]:
```
from sklearn.svm import LinearSVC

parameters = {'C':[0.125, 0.5, 1, 2, 8, 16]}
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
lr_svc = LinearSVC(tol=0.00005)
lr_svc_grid = GridSearchCV(lr_svc, param_grid=parameters,cv=3, n_jobs=-
1, verbose=1)
lr_svc_grid_results = perform_model(lr_svc_grid, X_train, y_train, X_te
st, y_test, class_labels=labels)
```

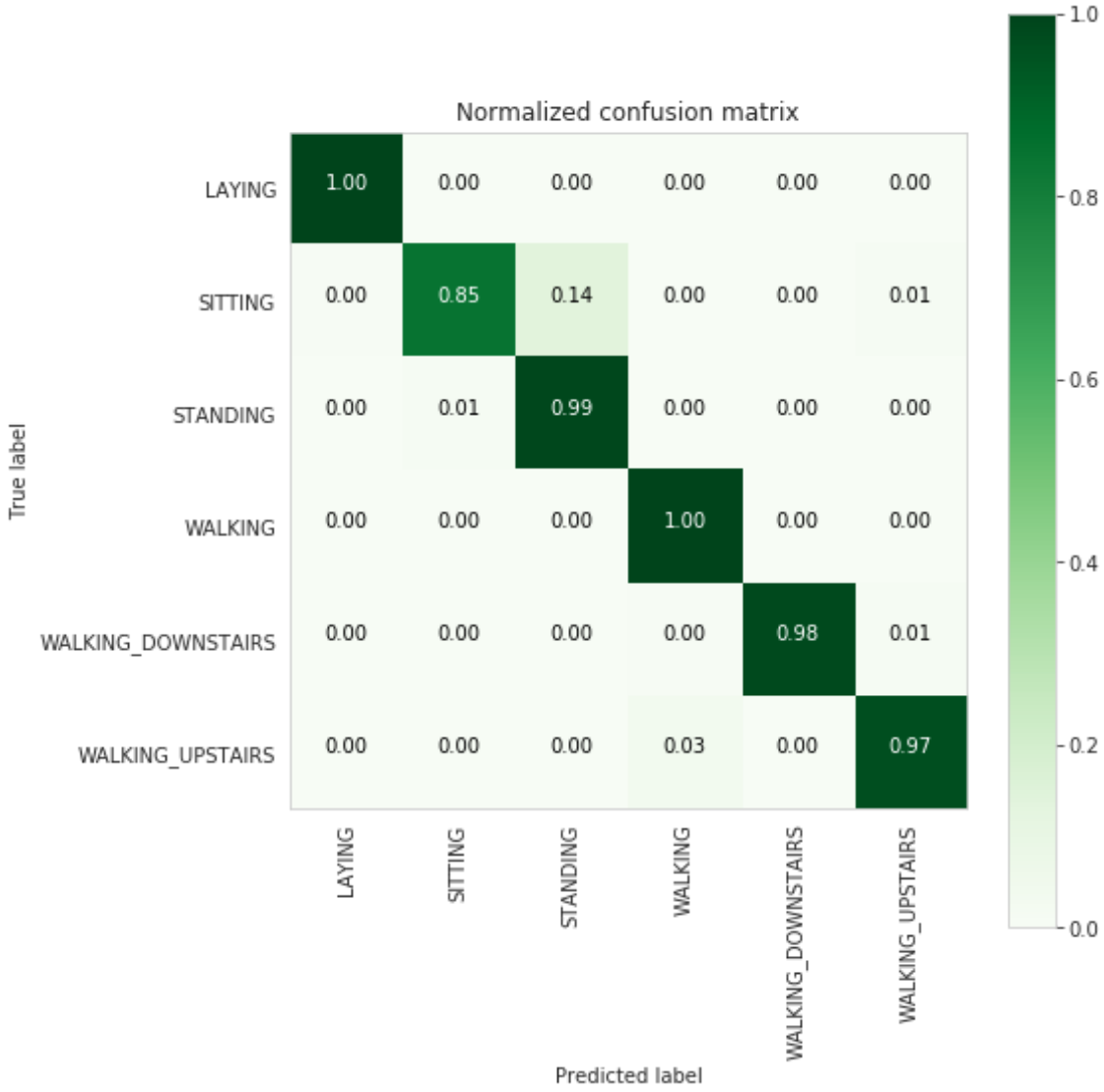
```
Training the model..
Fitting 3 folds for each of 6 candidates, totalling 18 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  18 out of  18 | elapsed:   29.9s finished
```

```
Done


Training Time (HH:MM:SS.ms) - 0:00:34.604442


Predicting Test Data
Done


Testing Time (HH:MM:SS:ms) - 0:00:00.005162


---------------------
|      Accuracy      |
---------------------

    0.9640312181879878


--------------------
| Confusion Matrix |
--------------------

 [[537   0   0   0   0   0]
 [  2 415  69   0   0   5]
 [  0   6 525   1   0   0]
 [  0   0   0 496   0   0]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
[  0    0    0    2 413    5]
[  0    0    0   15    1 455]]
```




```
-------------------------
| Classifiction Report |
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
-------------------------
                    precision    recall  f1-score   support

            LAYING       1.00      1.00      1.00       537
           SITTING       0.99      0.85      0.91       491
          STANDING       0.88      0.99      0.93       532
           WALKING       0.96      1.00      0.98       496
WALKING_DOWNSTAIRS       1.00      0.98      0.99       420
  WALKING_UPSTAIRS       0.98      0.97      0.97       471

          accuracy                           0.96      2947
         macro avg       0.97      0.96      0.96      2947
      weighted avg       0.97      0.96      0.96      2947
```

Observations:

- Here also, for the Confusion Matrix that we obtain, we can see the Confusion between 'Sitting' and 'Standing'.

In [29]:
```
print_grid_search_attributes(lr_svc_grid_results['model'])
```

```
-------------------------
|      Best Estimator     |
-------------------------

        LinearSVC(C=2, class_weight=None, dual=True, fit_intercept=Tru
e,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=5e-0
5,
          verbose=0)

-------------------------
|     Best Parameters     |
-------------------------
        Parameters of best estimator :
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        {'C': 2}

--------------------------------
|   No of Cross Validation sets   |
--------------------------------

        Total Number of Cross Validation sets: 3

--------------------------
|        Best Score       |
--------------------------

        Average Cross Validate scores of Best estimator :

        0.9458650707290533
```

6.3 Kernel SVM with GridSearch

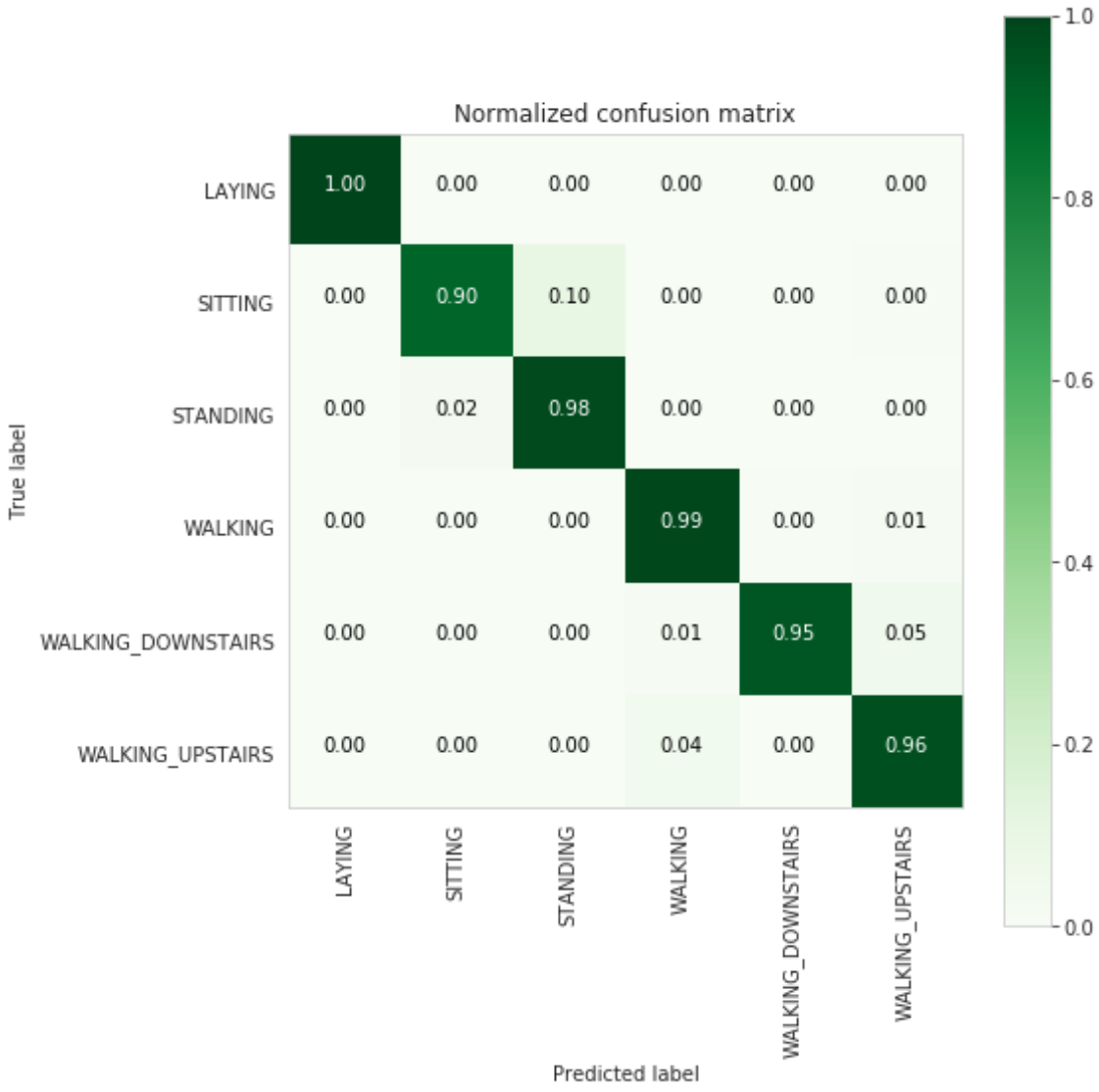
In [30]:
```
from sklearn.svm import SVC

parameters = {'C':[2,8,16],\
              'gamma': [ 0.0078125, 0.125, 2]}
rbf_svm = SVC(kernel='rbf')
rbf_svm_grid = GridSearchCV(rbf_svm,param_grid=parameters,cv=3, n_jobs=
-1)
rbf_svm_grid_results = perform_model(rbf_svm_grid, X_train, y_train, X_
test, y_test, class_labels=labels)
```

```
Training the model..
Done


Training Time (HH:MM:SS.ms) - 0:03:36.819734


Predicting Test Data
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Done


Testing Time (HH:MM:SS:ms) - 0:00:02.163891


---------------------
|      Accuracy      |
---------------------

    0.9626739056667798


--------------------
| Confusion Matrix |
--------------------

 [[537   0   0   0   0   0]
 [  0 441  48   0   0   2]
 [  0  12 520   0   0   0]
 [  0   0   0 489   2   5]
 [  0   0   0   4 397  19]
 [  0   0   0  17   1 453]]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD




```
-------------------------
| Classifiction Report |
-------------------------
                    precision    recall  f1-score   support

             LAYING       1.00      1.00      1.00       537
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
           SITTING       0.97      0.90      0.93       491
          STANDING       0.92      0.98      0.95       532
           WALKING       0.96      0.99      0.97       496
WALKING_DOWNSTAIRS       0.99      0.95      0.97       420
  WALKING_UPSTAIRS       0.95      0.96      0.95       471

          accuracy                           0.96      2947
         macro avg       0.96      0.96      0.96      2947
      weighted avg       0.96      0.96      0.96      2947
```

Observations:

- In the Confusion Matrix that we see in this case, we can see that there is some improvement in the 'Recall' for Sitting activity. It has improved to a value equal to 0.90, because only 10% of the sitting class is being classified as 'Standing', whereas in the previous case, 12% of the datapoints were getting misclassified.
- But at the same time, we can also see that there is some deterioration in the values for the other classes.

6.4 Decision Trees with GridSearchCV

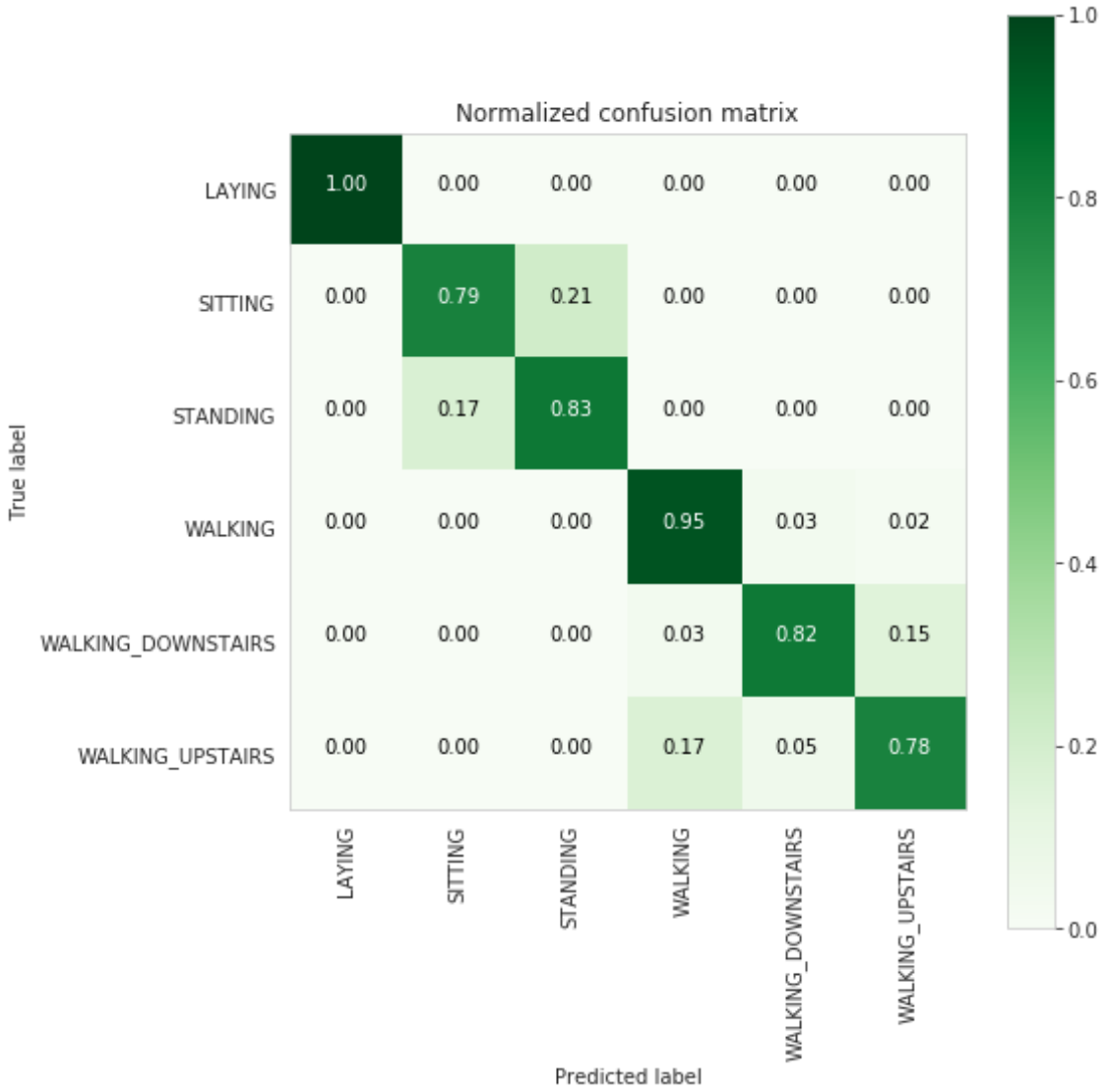
In [31]:
```
from sklearn.tree import DecisionTreeClassifier
parameters = {'max_depth':np.arange(3,10,2)}

dt = DecisionTreeClassifier()
dt_grid = GridSearchCV(dt,param_grid=parameters, cv=3, n_jobs=-1)
dt_grid_results = perform_model(dt_grid, X_train, y_train, X_test, y_te
st, class_labels=labels)
print_grid_search_attributes(dt_grid_results['model'])
```

```
Training the model..
Done
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Training Time (HH:MM:SS.ms) - 0:00:09.612736


Predicting Test Data
Done


Testing Time (HH:MM:SS:ms) - 0:00:00.009232


---------------------
|      Accuracy      |
---------------------

    0.8646080760095012


--------------------
| Confusion Matrix |
--------------------

 [[537   0   0   0   0   0]
 [  0 387 104   0   0   0]
 [  0  93 439   0   0   0]
 [  0   0   0 471  17   8]
 [  0   0   0  14 345  61]
 [  0   0   0  78  24 369]]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD




```
-------------------------
| Classifiction Report |
-------------------------
                    precision    recall  f1-score   support

             LAYING       1.00      1.00      1.00       537
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
           SITTING       0.81      0.79      0.80       491
          STANDING       0.81      0.83      0.82       532
           WALKING       0.84      0.95      0.89       496
WALKING_DOWNSTAIRS       0.89      0.82      0.86       420
  WALKING_UPSTAIRS       0.84      0.78      0.81       471

          accuracy                           0.86      2947
         macro avg       0.86      0.86      0.86      2947
      weighted avg       0.87      0.86      0.86      2947

--------------------------
|      Best Estimator     |
--------------------------

        DecisionTreeClassifier(class_weight=None, criterion='gini', max
_depth=7,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=No
ne,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')

--------------------------
|     Best Parameters     |
--------------------------
        Parameters of best estimator :

        {'max_depth': 7}

---------------------------------
|   No of Cross Validation sets   |
---------------------------------

        Total Number of Cross Validation sets: 3

--------------------------
|        Best Score       |
--------------------------
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Average Cross Validate scores of Best estimator :

0.8414036996735582
```

6.5 Random Forest Classifier with GridSearch

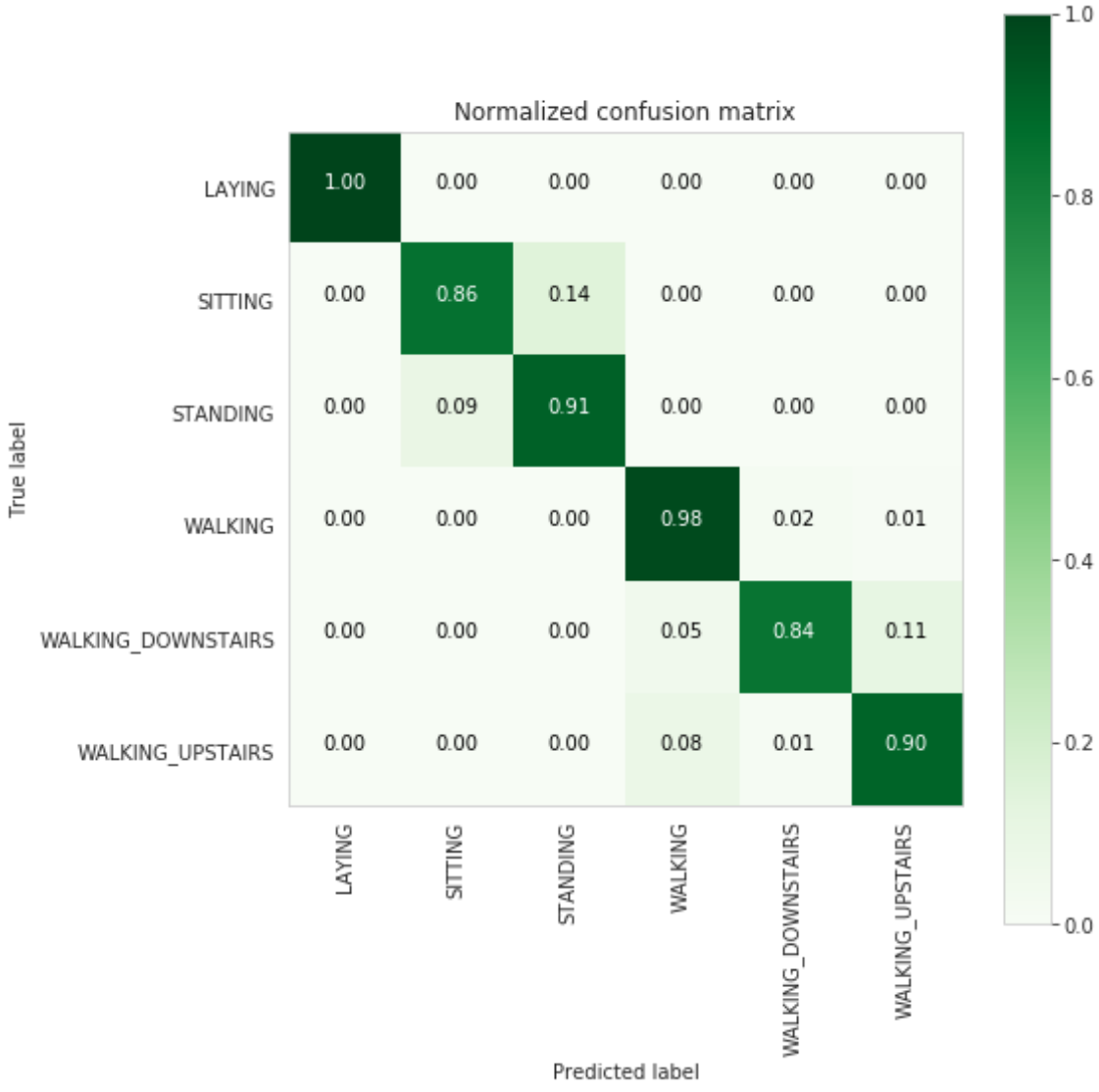
In [32]:
```
from sklearn.ensemble import RandomForestClassifier

params = {'n_estimators': np.arange(10,201,20), 'max_depth':np.arange(3
,15,2)}
rfc = RandomForestClassifier()
rfc_grid = GridSearchCV(rfc, param_grid=params, cv=3, n_jobs=-1)
rfc_grid_results = perform_model(rfc_grid, X_train, y_train, X_test, y_
test, class_labels=labels)
print_grid_search_attributes(rfc_grid_results['model'])
```

```
Training the model..
Done


Training Time (HH:MM:SS.ms) - 0:05:50.618098


Predicting Test Data
Done


Testing Time (HH:MM:SS:ms) - 0:00:00.060744


---------------------
|      Accuracy      |
---------------------

    0.9178825924669155
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
--------------------
| Confusion Matrix |
--------------------

 [[537   0   0   0   0   0]
 [   0 420  71   0   0   0]
 [   0  48 484   0   0   0]
 [   0   0   0 485   8   3]
 [   0   0   0  21 354  45]
 [   0   0   0  40   6 425]]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD




```
-------------------------
| Classifiction Report |
-------------------------
                    precision    recall  f1-score   support
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
              LAYING       1.00      1.00      1.00       537
             SITTING       0.90      0.86      0.88       491
            STANDING       0.87      0.91      0.89       532
             WALKING       0.89      0.98      0.93       496
WALKING_DOWNSTAIRS       0.96      0.84      0.90       420
  WALKING_UPSTAIRS       0.90      0.90      0.90       471

           accuracy                           0.92      2947
          macro avg       0.92      0.91      0.92      2947
       weighted avg       0.92      0.92      0.92      2947

--------------------------
|      Best Estimator     |
--------------------------

        RandomForestClassifier(bootstrap=True, class_weight=None, crite
rion='gini',
                       max_depth=9, max_features='auto', max_leaf_nodes
=None,
                       min_impurity_decrease=0.0, min_impurity_split=No
ne,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=130,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)

--------------------------
|     Best Parameters     |
--------------------------
        Parameters of best estimator :

        {'max_depth': 9, 'n_estimators': 130}

---------------------------------
|   No of Cross Validation sets   |
---------------------------------

        Total Number of Cross Validation sets: 3
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
--------------------------
|        Best Score       |
--------------------------

        Average Cross Validate scores of Best estimator :

        0.9129488574537541
```

6.6 Gradient Boosted Decision Trees With GridSearch

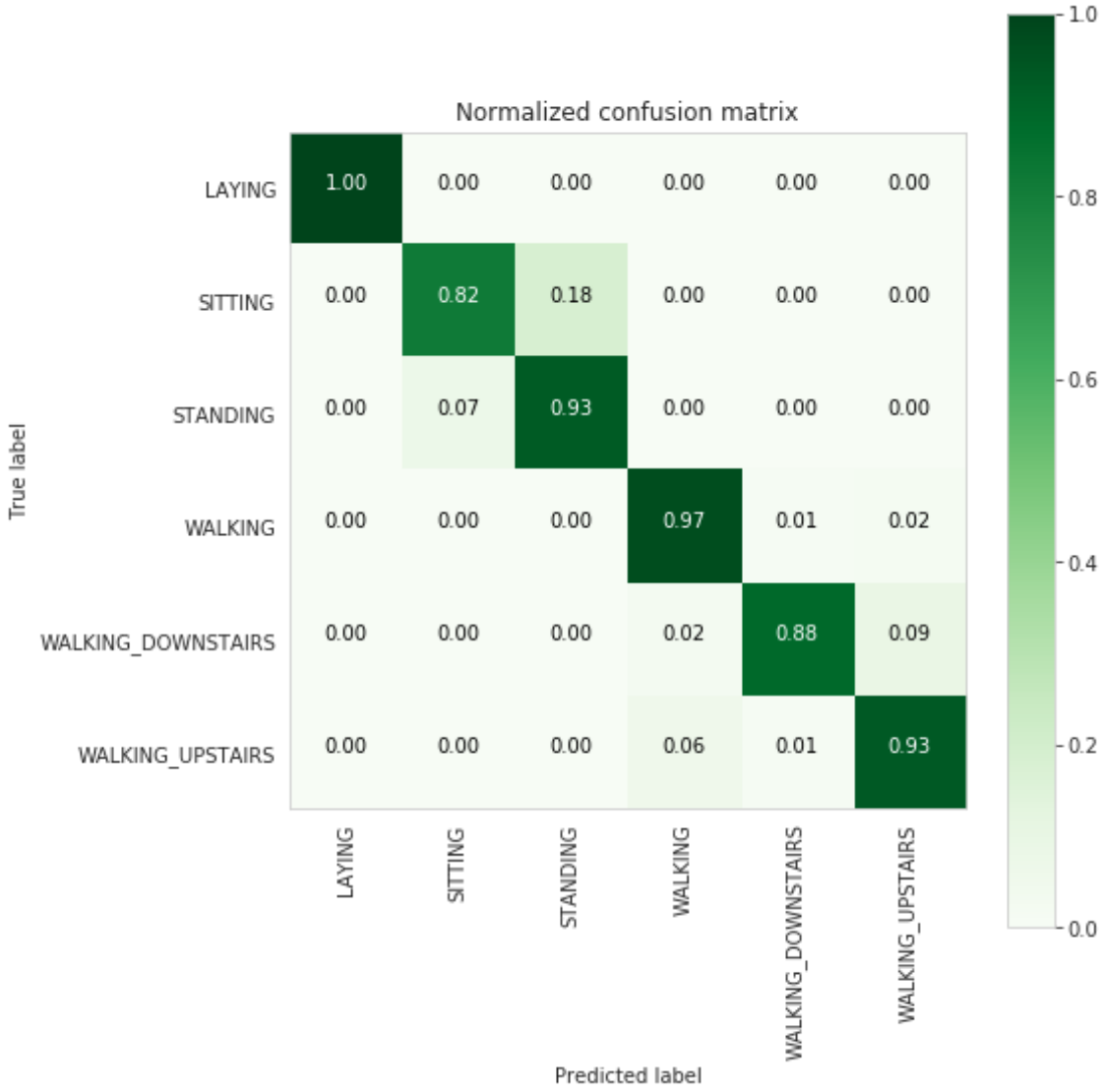
In [33]:
```
from sklearn.ensemble import GradientBoostingClassifier

param_grid = {'max_depth': np.arange(5,8,1), \
              'n_estimators':np.arange(130,170,10)}
gbdt = GradientBoostingClassifier()
gbdt_grid = GridSearchCV(gbdt, param_grid=param_grid, cv=3, n_jobs=-1)
gbdt_grid_results = perform_model(gbdt_grid, X_train, y_train, X_test,
y_test, class_labels=labels)
print_grid_search_attributes(gbdt_grid_results['model'])
```

```
Training the model..
Done


Training Time (HH:MM:SS.ms) - 0:49:08.619890


Predicting Test Data
Done


Testing Time (HH:MM:SS:ms) - 0:00:00.075518
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
---------------------
|      Accuracy      |
---------------------

    0.9236511706820495


--------------------
| Confusion Matrix |
--------------------

 [[537   0   0   0   0   0]
 [  0 401  89   0   0   1]
 [  0  39 493   0   0   0]
 [  0   0   0 481   6   9]
 [  0   0   0  10 371  39]
 [  0   1   0  26   5 439]]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD




```
-------------------------
| Classifiction Report |
-------------------------
                    precision    recall  f1-score   support

             LAYING       1.00      1.00      1.00       537
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
           SITTING       0.91      0.82      0.86       491
          STANDING       0.85      0.93      0.89       532
           WALKING       0.93      0.97      0.95       496
WALKING_DOWNSTAIRS       0.97      0.88      0.93       420
  WALKING_UPSTAIRS       0.90      0.93      0.92       471

          accuracy                           0.92      2947
         macro avg       0.93      0.92      0.92      2947
      weighted avg       0.93      0.92      0.92      2947

--------------------------
|      Best Estimator     |
--------------------------

        GradientBoostingClassifier(criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_dept
h=5,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_spli
t=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=1
30,
                           n_iter_no_change=None, presort='auto',
                           random_state=None, subsample=1.0, tol=0.000
1,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)

--------------------------
|     Best Parameters     |
--------------------------
        Parameters of best estimator :

        {'max_depth': 5, 'n_estimators': 130}

---------------------------------
|   No of Cross Validation sets   |
---------------------------------
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        Total Number of Cross Validation sets: 3

--------------------------
|        Best Score       |
--------------------------

        Average Cross Validate scores of Best estimator :

        0.9060119695321001
```

7. Applying Deep Learning Models on the Raw Data

- Here, instead of 6 Time Series Datasets, we actually have a total of 9 Time Series Datasets, where 6 of these Time Series Datasets are for the Accelerometer Readings whereas 3 of them are for the Gyroscope Readings : This is the Actual/Raw Time Series Data.
- All of these Time Series Datasets start at Time=0, and we break each time series into overlapping windows of 2.56 seconds (overlap of 50%). Each window corresponded to the 561 engineered features in the previous example.
- We take each window, perform some signal processing on top of that and encode the same as a 128-dimensional vector.
- Thus, at any Time t, we have 9 readings ie. we have a vector of 9 values. We are not designing any features over here and are just taking the raw windows :

```
 Each window has 128 time steps. At every Time step, we have
a total of 9 recordings, one corresponding to each of the Ti
me Series.
```

In [24]:
```
from tensorflow.python.keras.wrappers.scikit_learn import KerasClassifi
er
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
from sklearn.model_selection import GridSearchCV
from datetime import datetime
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Dropout, Dens
e, Flatten, BatchNormalization
```

In [25]:
```
# Activities are the class labels
# It is a 6 class classification
ACTIVITIES = {
    1: 'WALKING',
    2: 'WALKING_UPSTAIRS',
    3: 'WALKING_DOWNSTAIRS',
    4: 'SITTING',
    5: 'STANDING',
    6: '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=['Pr
ed'])
```

In [26]:
```
##gives train and validation data
def data():
    """
    Obtain the dataset from multiple files.
    Returns: X_train, X_test, y_train, y_test
    """
    # Data directory
    DATADIR = 'UCI_HAR_Dataset'
    SIGNALS = [
        "body_acc_x",
        "body_acc_y",
        "body_acc_z",
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        "body_gyro_x",
        "body_gyro_y",
        "body_gyro_z",
        "total_acc_x",
        "total_acc_y",
        "total_acc_z"
        ]
    # 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'HAR/UCI_HAR_Dataset/{subset}/Inertial Signals/
{signal}_{subset}.txt'
            signals_data.append( _read_csv(filename).to_numpy())

        # 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 timeste
ps, 9 signals)
        return np.transpose(signals_data, (1, 2, 0))

    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 representa
tion of
        every sample objective as a 6 bits vector using One Hot Encodin
g
        (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.
get_dummies.html)
        """
        filename = f'HAR/UCI_HAR_Dataset/{subset}/y_{subset}.txt'
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        y = _read_csv(filename)[0]
        return pd.get_dummies(y).to_numpy()

    X_train, X_test = load_signals('train'), load_signals('test')
    Y_train, Y_test = load_y('train'), load_y('test')

    return X_train, Y_train, X_test,  Y_test
```

In [27]:

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

# Configuring a session
session_conf = tf.compat.v1.ConfigProto(
    intra_op_parallelism_threads=1,
    inter_op_parallelism_threads=1
)

import keras.backend as K
sess = tf.Session(graph=tf.get_default_graph(), config=session_conf)
K.set_session(sess)

# Importing libraries
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import LSTM

# Initializing parameters
epochs = 30
batch_size = 256
n_hidden = 32
```

```
Using TensorFlow backend.
```

In [28]:

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

def plt_dynamic(x, vy, ty, ax, colors=['b']):
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    ax.plot(x, vy, 'b', label="Test Accuracy")
    ax.plot(x, ty, 'r', label="Train Accuracy")
    plt.legend()
    plt.grid()
    fig.canvas.draw()
```

In [29]:
```
#Function to plot the confusion matrix : Both Normalized and Non-Normal
ized Confusion Matrices

def plot_lstm_confusion_matrix(model,normalize=False,):

    y_true = pd.Series([ACTIVITIES[y+1] for y in np.argmax(Y_test, axis
=1)])
    y_pred = model.predict(X_test)
    y_pred_final = pd.Series([ACTIVITIES[y+1] for y in np.argmax(y_pred
, axis=1)])

    plt.figure(figsize=(8,8))
    plt.grid(b=False)

    if normalize:
        conf_matrix = pd.crosstab(y_true, y_pred_final, rownames=['True
 label'], colnames=['Predicted label'])
        conf_matrix = conf_matrix.astype('float') / conf_matrix.sum(axi
s=1)[:, np.newaxis]
        sns.heatmap(conf_matrix, annot=True, cmap="Greens", fmt=".3f",
xticklabels=labels, yticklabels=labels)
        plt.title('NORMALIZED CONFUSION MATRIX')
    else:
        conf_matrix = pd.crosstab(y_true, y_pred_final, rownames=['True
 label'], colnames=['Predicted label'])
        sns.heatmap(conf_matrix, annot=True, cmap="Blues", fmt=".3f", x
ticklabels=labels, yticklabels=labels)
        plt.title('NON-NORMALIZED CONFUSION MATRIX')

    plt.show()
```

In [30]:
```
# Loading the train and test data
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
X_train, Y_train, X_test, Y_test = data()

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
```

Observations :

- This means that we have 128 Time Steps and at each Time step we have 9 readings corresponding to each of the Time series datasets that we have.
- As we can see above, there are a total of 7352 windows (with overlaps present) : We have a total of 7352 time series where each time series is 128 time steps long and at each time step we have a total of 9 readings, all of which are real values.

Defining the Architecture of LSTM:

- We only have a total of 7352 datapoints and hence we only define a very simple LSTM, because the increasing complexity means that the chances of Overfitting the LSTM model also increases drastically : **It is very easy to overfit an LSTM Model.** *Also, because our number of datapoints is small, we can very easily overfit.*

7.1 Applying LSTM Models

7.1.1 Model 1

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

7.1.1.1 Constructing the Neural Network

In [46]:
```
import tensorflow.python.util.deprecation as deprecation
deprecation._PRINT_DEPRECATION_WARNINGS = False

# Initiliazing the sequential model
model1 = Sequential()
# Configuring the parameters
model1.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model1.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model1.add(Dense(n_classes, activation='softmax'))
model1.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
_________________________________________________________________
```

7.1.1.2 Running the Neural Network on Train & Test Datasets for 30 Epochs

In [49]:
```
#Refer :- https://github.com/tensorflow/tensorflow/issues/27023

import tensorflow.python.util.deprecation as deprecation
deprecation._PRINT_DEPRECATION_WARNINGS = False

#Compiling the Model
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
model1.compile(optimizer='rmsprop', loss='categorical_crossentropy', me
trics=['accuracy'])

#Training the Model
M1_history = model1.fit(X_train, Y_train, batch_size=batch_size, epochs
=epochs, verbose=1, \
                        validation_data=(X_test, Y_test))
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 5s 747us/sample - loss: 0.
4274 - acc: 0.8078 - val_loss: 0.5936 - val_acc: 0.7886
Epoch 2/30
7352/7352 [==============================] - 5s 671us/sample - loss: 0.
4075 - acc: 0.8211 - val_loss: 0.6167 - val_acc: 0.7720
Epoch 3/30
7352/7352 [==============================] - 5s 651us/sample - loss: 0.
3792 - acc: 0.8380 - val_loss: 0.5954 - val_acc: 0.7879
Epoch 4/30
7352/7352 [==============================] - 5s 637us/sample - loss: 0.
3923 - acc: 0.8413 - val_loss: 0.6599 - val_acc: 0.7927
Epoch 5/30
7352/7352 [==============================] - 5s 639us/sample - loss: 0.
4284 - acc: 0.8504 - val_loss: 0.6136 - val_acc: 0.8215
Epoch 6/30
7352/7352 [==============================] - 5s 626us/sample - loss: 0.
4138 - acc: 0.8468 - val_loss: 0.6568 - val_acc: 0.7967
Epoch 7/30
7352/7352 [==============================] - 5s 634us/sample - loss: 0.
3718 - acc: 0.8546 - val_loss: 0.6051 - val_acc: 0.8164
Epoch 8/30
7352/7352 [==============================] - 5s 631us/sample - loss: 0.
3466 - acc: 0.8641 - val_loss: 0.5450 - val_acc: 0.8337
Epoch 9/30
7352/7352 [==============================] - 5s 630us/sample - loss: 0.
3442 - acc: 0.8742 - val_loss: 0.5070 - val_acc: 0.8551
Epoch 10/30
7352/7352 [==============================] - 5s 648us/sample - loss: 0.
3570 - acc: 0.8675 - val_loss: 0.5439 - val_acc: 0.8215
Epoch 11/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
7352/7352 [==============================] - 5s 629us/sample - loss: 0.
3213 - acc: 0.8848 - val_loss: 0.5775 - val_acc: 0.8320
Epoch 12/30
7352/7352 [==============================] - 5s 648us/sample - loss: 0.
3075 - acc: 0.8908 - val_loss: 0.4919 - val_acc: 0.8592
Epoch 13/30
7352/7352 [==============================] - 5s 641us/sample - loss: 0.
3020 - acc: 0.8947 - val_loss: 0.5339 - val_acc: 0.8741
Epoch 14/30
7352/7352 [==============================] - 5s 649us/sample - loss: 0.
2803 - acc: 0.9051 - val_loss: 0.4998 - val_acc: 0.8663
Epoch 15/30
7352/7352 [==============================] - 5s 636us/sample - loss: 0.
2909 - acc: 0.9044 - val_loss: 0.6051 - val_acc: 0.8273
Epoch 16/30
7352/7352 [==============================] - 5s 636us/sample - loss: 0.
2811 - acc: 0.9047 - val_loss: 0.4806 - val_acc: 0.8724
Epoch 17/30
7352/7352 [==============================] - 5s 636us/sample - loss: 0.
2583 - acc: 0.9143 - val_loss: 0.5059 - val_acc: 0.8711
Epoch 18/30
7352/7352 [==============================] - 5s 657us/sample - loss: 0.
2634 - acc: 0.9125 - val_loss: 0.4959 - val_acc: 0.8724
Epoch 19/30
7352/7352 [==============================] - 5s 627us/sample - loss: 0.
2703 - acc: 0.9149 - val_loss: 0.6231 - val_acc: 0.8249
Epoch 20/30
7352/7352 [==============================] - 5s 634us/sample - loss: 0.
2459 - acc: 0.9202 - val_loss: 0.5965 - val_acc: 0.8616
Epoch 21/30
7352/7352 [==============================] - 5s 631us/sample - loss: 0.
2753 - acc: 0.9161 - val_loss: 0.5453 - val_acc: 0.8751
Epoch 22/30
7352/7352 [==============================] - 5s 635us/sample - loss: 0.
2743 - acc: 0.9113 - val_loss: 0.5413 - val_acc: 0.8711
Epoch 23/30
7352/7352 [==============================] - 5s 635us/sample - loss: 0.
2344 - acc: 0.9197 - val_loss: 0.4987 - val_acc: 0.8829
Epoch 24/30
7352/7352 [==============================] - 5s 655us/sample - loss: 0.
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
2227 - acc: 0.9263 - val_loss: 0.5358 - val_acc: 0.8823
Epoch 25/30
7352/7352 [==============================] - 5s 638us/sample - loss: 0.
2151 - acc: 0.9312 - val_loss: 0.5220 - val_acc: 0.8775
Epoch 26/30
7352/7352 [==============================] - 5s 633us/sample - loss: 0.
2423 - acc: 0.9223 - val_loss: 0.5005 - val_acc: 0.8792
Epoch 27/30
7352/7352 [==============================] - 5s 634us/sample - loss: 0.
2224 - acc: 0.9298 - val_loss: 0.5073 - val_acc: 0.8823
Epoch 28/30
7352/7352 [==============================] - 5s 638us/sample - loss: 0.
2108 - acc: 0.9267 - val_loss: 0.5424 - val_acc: 0.8670
Epoch 29/30
7352/7352 [==============================] - 5s 637us/sample - loss: 0.
2630 - acc: 0.9206 - val_loss: 0.5457 - val_acc: 0.8772
Epoch 30/30
7352/7352 [==============================] - 5s 635us/sample - loss: 0.
2300 - acc: 0.9286 - val_loss: 0.5404 - val_acc: 0.8728
```

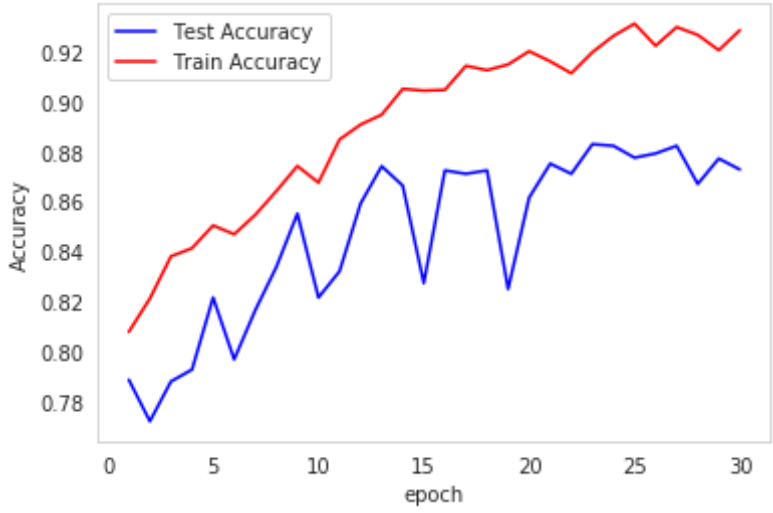
7.1.1.3 Number of Epochs vs Train Loss & Test Loss

In [50]:
```
M1_score = model1.evaluate(X_test, Y_test, verbose=0)
print('Test Log Loss score:', M1_score[0])
print('Test accuracy:', M1_score[1])

fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Accuracy')

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

# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epo
chs=nb_epoch, verbose=1, validation_data=(X_test, Y_test))
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
# we will get val_loss and val_acc only when you pass the parameter val
idation_data
# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
# for each key in history.history we will have a list of length equal t
o number of epochs

vy = M1_history.history['val_acc']
ty = M1_history.history['acc']
plt_dynamic(x, vy, ty, ax)
```

```
Test Log Loss score: 0.5403980870428108
Test accuracy: 0.87275195
```



7.1.1.4 Normalized and Non-Normalized Confusion Matrices

In [58]:
```
#Plot a Non-Normalized confusion matrix
plot_lstm_confusion_matrix(model1, normalize=False)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
#Plot a Non-Normalized confusion matrix
plot_lstm_confusion_matrix(model1, normalize=True)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

7.1.2 Model 2

7.1.2.1 Constructing the Neural Network

In [32]:
```
# Initiliazing the sequential model
import tensorflow.python.util.deprecation as deprecation
deprecation._PRINT_DEPRECATION_WARNINGS = False

model2 = Sequential()

model2.add(Dense(64, activation='relu', input_shape=(timesteps, input_d
im), kernel_initializer='he_uniform'))
model2.add(BatchNormalization())
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
model2.add(Dropout(0.3))

model2.add(Dense(32, activation='relu', kernel_initializer='he_uniform'
))
model2.add(BatchNormalization())
model2.add(Dropout(0.3))

# Configuring the parameters
model2.add(LSTM(32,kernel_initializer='he_uniform', \
                dropout=0.3, return_sequences=True))
model2.add(LSTM(32,kernel_initializer='he_uniform', dropout=0.3))

model2.add(Dense(n_classes, activation='softmax'))
model2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 128, 64)           640
_________________________________________________________________
batch_normalization_2 (Batch (None, 128, 64)           256
_________________________________________________________________
dropout_2 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
dense_4 (Dense)              (None, 128, 32)           2080
_________________________________________________________________
batch_normalization_3 (Batch (None, 128, 32)           128
_________________________________________________________________
dropout_3 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
lstm_2 (LSTM)                (None, 128, 32)           8320
_________________________________________________________________
lstm_3 (LSTM)                (None, 32)                8320
_________________________________________________________________
dense_5 (Dense)              (None, 6)                 198
=================================================================
Total params: 19,942
Trainable params: 19,750
Non-trainable params: 192
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

7.1.2.2 Running the Neural Network on Train & Test Datasets for 30 Epochs

In [34]:
```
#Refer :- https://github.com/tensorflow/tensorflow/issues/27023

import tensorflow.python.util.deprecation as deprecation
deprecation._PRINT_DEPRECATION_WARNINGS = False

#Compiling the Model
model2.compile(optimizer='rmsprop', loss='categorical_crossentropy', me
trics=['accuracy'])

#Training the Model
M2_history = model2.fit(X_train, Y_train, batch_size=batch_size, epochs
=epochs, verbose=1, \
                        validation_data=(X_test, Y_test))
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 20s 3ms/sample - loss: 1.3
700 - acc: 0.4671 - val_loss: 1.3180 - val_acc: 0.4578
Epoch 2/30
7352/7352 [==============================] - 19s 3ms/sample - loss: 0.9
187 - acc: 0.6177 - val_loss: 0.9227 - val_acc: 0.5867
Epoch 3/30
7352/7352 [==============================] - 18s 2ms/sample - loss: 0.7
448 - acc: 0.6557 - val_loss: 1.1521 - val_acc: 0.5629
Epoch 4/30
7352/7352 [==============================] - 18s 2ms/sample - loss: 0.6
695 - acc: 0.6797 - val_loss: 1.2811 - val_acc: 0.5467
Epoch 5/30
7352/7352 [==============================] - 18s 2ms/sample - loss: 0.6
383 - acc: 0.6853 - val_loss: 1.5240 - val_acc: 0.5338
Epoch 6/30
7352/7352 [==============================] - 18s 2ms/sample - loss: 0.6
329 - acc: 0.6950 - val_loss: 1.6953 - val_acc: 0.5070
Epoch 7/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
7352/7352 [==============================] - 18s 2ms/sample - loss: 0.6
042 - acc: 0.7141 - val_loss: 1.9526 - val_acc: 0.5100
Epoch 8/30
7352/7352 [==============================] - 18s 2ms/sample - loss: 0.5
837 - acc: 0.7364 - val_loss: 1.3392 - val_acc: 0.5945
Epoch 9/30
7352/7352 [==============================] - 18s 2ms/sample - loss: 0.5
427 - acc: 0.7677 - val_loss: 1.2236 - val_acc: 0.6159
Epoch 10/30
7352/7352 [==============================] - 18s 2ms/sample - loss: 0.4
907 - acc: 0.7998 - val_loss: 0.9605 - val_acc: 0.6634
Epoch 11/30
7352/7352 [==============================] - 18s 2ms/sample - loss: 0.4
540 - acc: 0.8156 - val_loss: 0.9088 - val_acc: 0.6868
Epoch 12/30
7352/7352 [==============================] - 18s 2ms/sample - loss: 0.4
210 - acc: 0.8346 - val_loss: 0.7335 - val_acc: 0.7570
Epoch 13/30
7352/7352 [==============================] - 18s 2ms/sample - loss: 0.3
951 - acc: 0.8467 - val_loss: 0.7039 - val_acc: 0.7679
Epoch 14/30
7352/7352 [==============================] - 18s 2ms/sample - loss: 0.3
662 - acc: 0.8595 - val_loss: 0.7190 - val_acc: 0.7506
Epoch 15/30
7352/7352 [==============================] - 18s 2ms/sample - loss: 0.3
487 - acc: 0.8648 - val_loss: 0.5354 - val_acc: 0.8249
Epoch 16/30
7352/7352 [==============================] - 18s 2ms/sample - loss: 0.3
074 - acc: 0.8853 - val_loss: 0.5555 - val_acc: 0.8015
Epoch 17/30
7352/7352 [==============================] - 17s 2ms/sample - loss: 0.3
038 - acc: 0.8906 - val_loss: 0.4783 - val_acc: 0.8517
Epoch 18/30
7352/7352 [==============================] - 18s 2ms/sample - loss: 0.2
925 - acc: 0.8919 - val_loss: 0.4570 - val_acc: 0.8646
Epoch 19/30
7352/7352 [==============================] - 17s 2ms/sample - loss: 0.2
801 - acc: 0.8976 - val_loss: 0.4825 - val_acc: 0.8592
Epoch 20/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
7352/7352 [==============================] - 18s 2ms/sample - loss: 0.2
597 - acc: 0.9071 - val_loss: 0.4621 - val_acc: 0.8551
Epoch 21/30
7352/7352 [==============================] - 18s 2ms/sample - loss: 0.2
614 - acc: 0.9040 - val_loss: 0.4175 - val_acc: 0.8575
Epoch 22/30
7352/7352 [==============================] - 17s 2ms/sample - loss: 0.2
345 - acc: 0.9134 - val_loss: 0.3800 - val_acc: 0.8887
Epoch 23/30
7352/7352 [==============================] - 17s 2ms/sample - loss: 0.2
232 - acc: 0.9207 - val_loss: 0.4237 - val_acc: 0.8836
Epoch 24/30
7352/7352 [==============================] - 18s 2ms/sample - loss: 0.2
194 - acc: 0.9214 - val_loss: 0.3799 - val_acc: 0.8921
Epoch 25/30
7352/7352 [==============================] - 17s 2ms/sample - loss: 0.2
095 - acc: 0.9237 - val_loss: 0.3726 - val_acc: 0.8948
Epoch 26/30
7352/7352 [==============================] - 18s 2ms/sample - loss: 0.2
145 - acc: 0.9211 - val_loss: 0.3648 - val_acc: 0.9002
Epoch 27/30
7352/7352 [==============================] - 18s 2ms/sample - loss: 0.2
011 - acc: 0.9229 - val_loss: 0.3385 - val_acc: 0.9091
Epoch 28/30
7352/7352 [==============================] - 19s 3ms/sample - loss: 0.2
020 - acc: 0.9279 - val_loss: 0.3261 - val_acc: 0.9067
Epoch 29/30
7352/7352 [==============================] - 19s 3ms/sample - loss: 0.1
818 - acc: 0.9325 - val_loss: 0.3115 - val_acc: 0.9135
Epoch 30/30
7352/7352 [==============================] - 19s 3ms/sample - loss: 0.1
899 - acc: 0.9306 - val_loss: 0.3258 - val_acc: 0.9074
```

7.1.2.3 Number of Epochs vs Train Loss & Test Loss

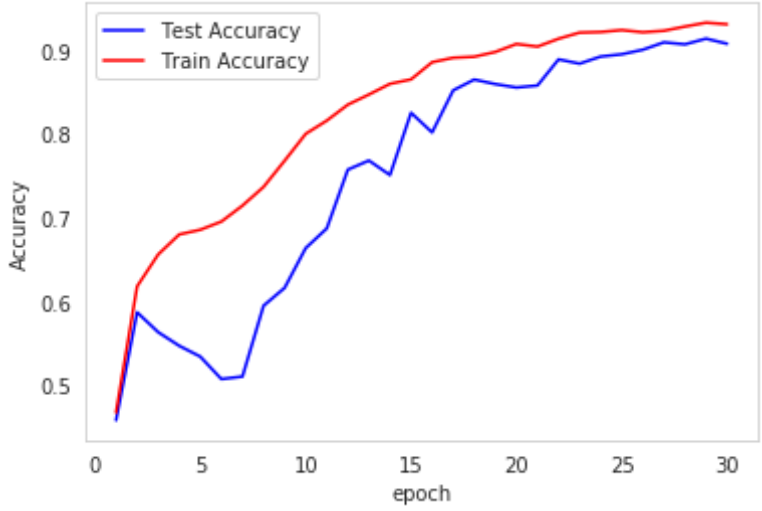
In [35]:
```
M2_score = model2.evaluate(X_test, Y_test, verbose=0)
print('Test Log Loss score:', M2_score[0])
print('Test accuracy:', M2_score[1])
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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

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

# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epo
chs=nb_epoch, verbose=1, validation_data=(X_test, Y_test))

# we will get val_loss and val_acc only when you pass the parameter val
idation_data
# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
# for each key in history.history we will have a list of length equal t
o number of epochs

vy = M2_history.history['val_acc']
ty = M2_history.history['acc']
plt_dynamic(x, vy, ty, ax)
```

```
Test Log Loss score: 0.3257699168741028
Test accuracy: 0.9073634
```

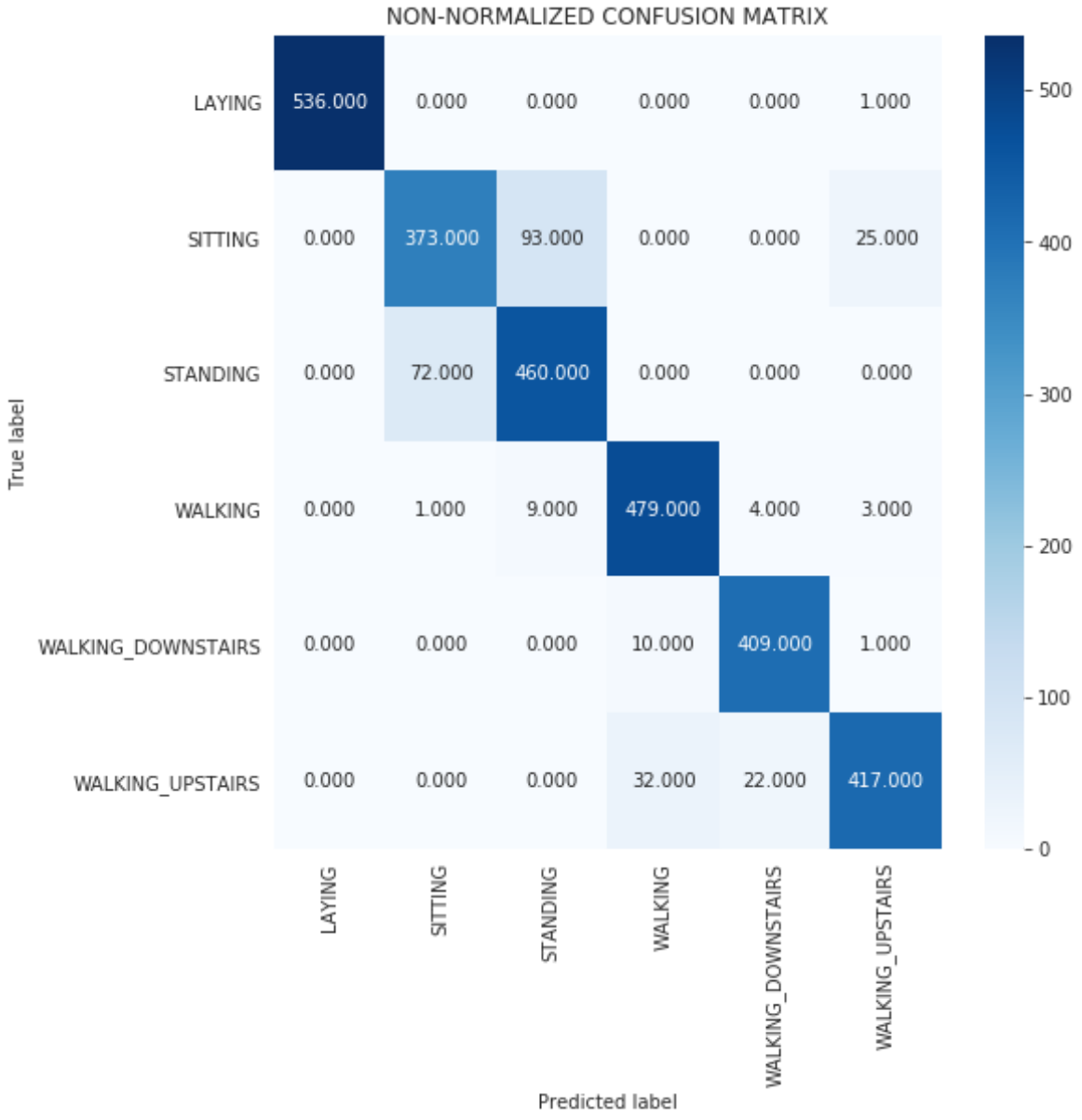
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



7.1.2.4 Normalized and Non-Normalized Confusion Matrices

In [36]:
```
#Plot a Non-Normalized confusion matrix
plot_lstm_confusion_matrix(model2, normalize=False)

#Plot a Non-Normalized confusion matrix
plot_lstm_confusion_matrix(model2, normalize=True)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

7.1.2.5 Hyperparameter Tuning using Hyperas

In [37]:
```
def model2(X_train, Y_train, X_test, Y_test):

    # Importing tensorflow
    from keras.models import Sequential
    from keras.layers import Dense, Dropout, LSTM, BatchNormalization
    from keras.callbacks import TensorBoard
    from keras.callbacks import ModelCheckpoint
    from keras.optimizers import RMSprop
    from keras.optimizers import adam

    np.random.seed(36)
    import tensorflow as tf
    tf.set_random_seed(36)
    # Initiliazing the sequential model
    model2 = Sequential()
    # Configuring the parameters
    model2.add(Dense(64, activation='relu', input_shape=(128, 9), kerne
l_initializer='he_uniform'))
    model2.add(BatchNormalization())
    model2.add(Dropout({{uniform(0.25,0.65)}},name='Dropout_1'))

    model2.add(Dense(32, activation='relu', kernel_initializer='he_unif
orm'))
    model2.add(BatchNormalization())
    model2.add(Dropout({{uniform(0.25,0.85)}},name='Dropout_2'))

    model2.add(LSTM({{choice([32,36,64])}},\
                    return_sequences=True, input_shape=(128, 9),name='L
STM_1'))
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    model2.add(BatchNormalization())
    model2.add(Dropout({{uniform(0.25,0.85)}},name='Dropout_3'))

    model2.add(LSTM({{choice([32,36,64])}}, \
                    input_shape=(128, 9),name='LSTM_2'))
    model2.add(BatchNormalization())
    model2.add(Dropout({{uniform(0.25,0.85)}},name='Dropout_4'))

    # Adding a dense output layer with sigmoid activation
    model2.add(Dense(6, activation='softmax'))

    adam = keras.optimizers.Adam(lr={{uniform(0.0009,0.025)}})
    rmsprop = keras.optimizers.RMSprop(lr={{uniform(0.0009,0.025)}})

    choiceval = {{choice(['adam', 'rmsprop'])}}

    if choiceval == 'adam':
        optim = adam
    else:
        optim = rmsprop

    print(model2.summary())

    model2.compile(loss='categorical_crossentropy', metrics=['accuracy'
],optimizer=optim)

    result = model2.fit(X_train, Y_train,
              batch_size=16,
              nb_epoch=30,
              verbose=2,
              validation_data=(X_test, Y_test))

    score, acc = model2.evaluate(X_test, Y_test, verbose=0)
    print('Test accuracy:', acc)
    print('-------------------------------------------------------------
--------------------------')
    return {'loss': -acc, 'status': STATUS_OK, 'model': model2}
```

In [37]:

```
start = datetime.now()
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
import keras
from keras.regularizers import l2
from tensorflow.python.keras import utils
from hyperas import optim
from hyperopt import Trials, STATUS_OK, tpe

X_train, Y_train, X_test, Y_test = data()
trials = Trials()
best_run, best_model, space = optim.minimize(model=model2,
                                        data=data,
                                        algo=tpe.suggest,
                                        max_evals=15,
                                        trials=trials,notebook_name = 'Hu
man_Activity_Recognition',
                                        return_space = True, eval_space=T
rue)

print(" "*100)
print('Time taken :', datetime.now() - start)
```

```
>>> Imports:
#coding=utf-8

try:
    import numpy as np
except:
    pass

try:
    import pandas as pd
except:
    pass

try:
    import warnings
except:
    pass

try:
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    import matplotlib.pyplot as plt
except:
    pass

try:
    import seaborn as sns
except:
    pass

try:
    import numpy as np
except:
    pass

try:
    from sklearn.manifold import TSNE
except:
    pass

try:
    import matplotlib.pyplot as plt
except:
    pass

try:
    import seaborn as sns
except:
    pass

try:
    import itertools
except:
    pass

try:
    import numpy as np
except:
    pass
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
try:
    import matplotlib.pyplot as plt
except:
    pass

try:
    from sklearn.metrics import confusion_matrix
except:
    pass

try:
    from datetime import datetime
except:
    pass

try:
    from sklearn import linear_model
except:
    pass

try:
    from sklearn import metrics
except:
    pass

try:
    from sklearn.model_selection import GridSearchCV
except:
    pass

try:
    from sklearn.svm import LinearSVC
except:
    pass

try:
    from sklearn.svm import SVC
except:
    pass
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
try:
    from sklearn.tree import DecisionTreeClassifier
except:
    pass

try:
    from sklearn.ensemble import RandomForestClassifier
except:
    pass

try:
    from sklearn.ensemble import GradientBoostingClassifier
except:
    pass

try:
    from tensorflow.python.keras.wrappers.scikit_learn import KerasClas
sifier
except:
    pass

try:
    from sklearn.model_selection import GridSearchCV
except:
    pass

try:
    from datetime import datetime
except:
    pass

try:
    from tensorflow.python.keras.layers.normalization import BatchNorma
lization
except:
    pass

try:
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    from tensorflow.python.keras.layers import Dropout
except:
    pass

try:
    import tensorflow as tf
except:
    pass

try:
    from keras import backend as K
except:
    pass

try:
    from tensorflow.python.keras.models import Sequential
except:
    pass

try:
    from tensorflow.python.keras.layers import LSTM
except:
    pass

try:
    from tensorflow.python.keras.layers.core import Dense, Dropout
except:
    pass

try:
    import tensorflow.python.util.deprecation as deprecation
except:
    pass

try:
    import tensorflow.python.util.deprecation as deprecation
except:
    pass
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
try:
    import tensorflow.python.util.deprecation as deprecation
except:
    pass

try:
    import tensorflow.python.util.deprecation as deprecation
except:
    pass

try:
    import tensorflow.python.util.deprecation as deprecation
except:
    pass

try:
    import tensorflow.python.util.deprecation as deprecation
except:
    pass

try:
    import tensorflow.python.util.deprecation as deprecation
except:
    pass

try:
    import tensorflow.python.util.deprecation as deprecation
except:
    pass

try:
    from keras.models import Sequential
except:
    pass

try:
    from keras.layers import Dense, Dropout, LSTM, BatchNormalization
except:
    pass
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
try:
    from keras.callbacks import TensorBoard
except:
    pass

try:
    from keras.callbacks import ModelCheckpoint
except:
    pass

try:
    from keras.optimizers import RMSprop
except:
    pass

try:
    from keras.optimizers import adam
except:
    pass

try:
    import tensorflow as tf
except:
    pass

try:
    import keras
except:
    pass

try:
    from keras.regularizers import l2
except:
    pass

try:
    from tensorflow.python.keras import utils
except:
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    pass

try:
    from hyperas import optim
except:
    pass

try:
    from hyperopt import Trials, STATUS_OK, tpe
except:
    pass

try:
    import warnings
except:
    pass

try:
    import tensorflow.python.util.deprecation as deprecation
except:
    pass

try:
    from tensorflow.python.keras.optimizers import RMSprop
except:
    pass

>>> Hyperas search space:

def get_space():
    return {
        'Dropout': hp.uniform('Dropout', 0.25,0.65),
        'Dropout_1': hp.uniform('Dropout_1', 0.25,0.85),
        'LSTM': hp.choice('LSTM', [32,36,64]),
        'Dropout_2': hp.uniform('Dropout_2', 0.25,0.85),
        'LSTM_1': hp.choice('LSTM_1', [32,36,64]),
        'Dropout_3': hp.uniform('Dropout_3', 0.25,0.85),
        'lr': hp.uniform('lr', 0.0009,0.025),
        'lr_1': hp.uniform('lr_1', 0.0009,0.025),
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        'choiceval': hp.choice('choiceval', ['adam', 'rmsprop']),
    }

>>> Data
   1:
   2: """
   3: Obtain the dataset from multiple files.
   4: Returns: X_train, X_test, y_train, y_test
   5: """
   6: # Data directory
   7: DATADIR = 'UCI_HAR_Dataset'
   8: SIGNALS = [
   9:     "body_acc_x",
  10:     "body_acc_y",
  11:     "body_acc_z",
  12:     "body_gyro_x",
  13:     "body_gyro_y",
  14:     "body_gyro_z",
  15:     "total_acc_x",
  16:     "total_acc_y",
  17:     "total_acc_z"
  18:     ]
  19: # Utility function to read the data from csv file
  20: def _read_csv(filename):
  21:     return pd.read_csv(filename, delim_whitespace=True, header=No
ne)
  22:
  23: # Utility function to load the load
  24: def load_signals(subset):
  25:     signals_data = []
  26:
  27:     for signal in SIGNALS:
  28:         filename = f'HAR/UCI_HAR_Dataset/{subset}/Inertial Signal
s/{signal}_{subset}.txt'
  29:         signals_data.append( _read_csv(filename).to_numpy())
  30:
  31:     # Transpose is used to change the dimensionality of the outpu
t,
  32:     # aggregating the signals by combination of sample/timestep.
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
  33:     # Resultant shape is (7352 train/2947 test samples, 128 times
teps, 9 signals)
  34:     return np.transpose(signals_data, (1, 2, 0))
  35:
  36: def load_y(subset):
  37:     """
  38:     The objective that we are trying to predict is a integer, fro
m 1 to 6,
  39:     that represents a human activity. We return a binary represen
tation of
  40:     every sample objective as a 6 bits vector using One Hot Encod
ing
  41:     (https://pandas.pydata.org/pandas-docs/stable/generated/panda
s.get_dummies.html)
  42:     """
  43:     filename = f'HAR/UCI_HAR_Dataset/{subset}/y_{subset}.txt'
  44:     y = _read_csv(filename)[0]
  45:     return pd.get_dummies(y).to_numpy()
  46:
  47: X_train, X_test = load_signals('train'), load_signals('test')
  48: Y_train, Y_test = load_y('train'), load_y('test')
  49:
  50:
  51:
  52:
>>> Resulting replaced keras model:

   1: def keras_fmin_fnct(space):
   2:
   3:
   4:     # Importing tensorflow
   5:
   6:     np.random.seed(36)
   7:     tf.set_random_seed(36)
   8:     # Initiliazing the sequential model
   9:     model7 = Sequential()
  10:     # Configuring the parameters
  11:     model7.add(Dense(64, activation='relu', input_shape=(128, 9),
kernel_initializer='he_uniform'))
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    model7.add(BatchNormalization())
    model7.add(Dropout(space['Dropout'],name='Dropout_1'))

    model7.add(Dense(32, activation='relu', kernel_initializer='he_uniform'))
    model7.add(BatchNormalization())
    model7.add(Dropout(space['Dropout_1'],name='Dropout_2'))

    model7.add(LSTM(space['LSTM'],\
                    return_sequences=True, input_shape=(128, 9),name='LSTM_1'))
    model7.add(BatchNormalization())
    model7.add(Dropout(space['Dropout_2'],name='Dropout_3'))

    model7.add(LSTM(space['LSTM_1'], \
                    input_shape=(128, 9),name='LSTM_2'))
    model7.add(BatchNormalization())
    model7.add(Dropout(space['Dropout_3'],name='Dropout_4'))

    # Adding a dense output layer with sigmoid activation
    model7.add(Dense(6, activation='softmax'))

    adam = keras.optimizers.Adam(lr=space['lr'])
    rmsprop = keras.optimizers.RMSprop(lr=space['lr_1'])

    choiceval = space['choiceval']

    if choiceval == 'adam':
        optim = adam
    else:
        optim = rmsprop

    print(model7.summary())

    model7.compile(loss='categorical_crossentropy', metrics=['accuracy'],optimizer=optim)

    result = model7.fit(X_train, Y_train,
               batch_size=16,
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
  48:                nb_epoch=30,
  49:                verbose=2,
  50:                validation_data=(X_test, Y_test))
  51:
  52:     score, acc = model7.evaluate(X_test, Y_test, verbose=0)
  53:     print('Test accuracy:', acc)
  54:     print('-------------------------------------------------------
-------------------------------')
  55:     return {'loss': -acc, 'status': STATUS_OK, 'model': model7}
  56:
  0%|          | 0/15 [00:00<?, ?trial/s, best loss=?]WARNING:tensorflo
w:From /Users/dhruvnarayanan/temp_model.py:335: The name tf.set_random_
seed is deprecated. Please use tf.compat.v1.set_random_seed instead.

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128, 64)           640
_________________________________________________________________
batch_normalization_1 (Batch (None, 128, 64)           256
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
dense_2 (Dense)              (None, 128, 32)           2080
_________________________________________________________________
batch_normalization_2 (Batch (None, 128, 32)           128
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 36)           9936
_________________________________________________________________
batch_normalization_3 (Batch (None, 128, 36)           144
_________________________________________________________________
Dropout_3 (Dropout)          (None, 128, 36)           0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 36)                10512
_________________________________________________________________
batch_normalization_4 (Batch (None, 36)                144
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
Dropout_4 (Dropout)          (None, 36)                0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 222
=================================================================
Total params: 24,062
Trainable params: 23,726
Non-trainable params: 336
_________________________________________________________________
None
  0%|          | 0/15 [00:01<?, ?trial/s, best loss=?]WARNING:tensorflo
w:From /Users/dhruvnarayanan/opt/anaconda3/lib/python3.7/site-packages/
keras/backend/tensorflow_backend.py:422: The name tf.global_variables i
s deprecated. Please use tf.compat.v1.global_variables instead.

Train on 7352 samples, validate on 2947 samples
Epoch 1/30
 - 57s - loss: 0.7385 - accuracy: 0.7140 - val_loss: 0.7518 - val_accur
acy: 0.7631

Epoch 2/30
 - 49s - loss: 0.4198 - accuracy: 0.8678 - val_loss: 0.4173 - val_accur
acy: 0.8829

Epoch 3/30
 - 50s - loss: 0.3673 - accuracy: 0.8874 - val_loss: 0.5282 - val_accur
acy: 0.8029

Epoch 4/30
 - 48s - loss: 0.3600 - accuracy: 0.8915 - val_loss: 0.4637 - val_accur
acy: 0.8517

Epoch 5/30
 - 48s - loss: 0.3549 - accuracy: 0.8904 - val_loss: 0.4823 - val_accur
acy: 0.8758

Epoch 6/30
 - 48s - loss: 0.3300 - accuracy: 0.9008 - val_loss: 0.3273 - val_accur
acy: 0.8775
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 7/30
 - 48s - loss: 0.3417 - accuracy: 0.8950 - val_loss: 0.3711 - val_accur
acy: 0.8921

Epoch 8/30
 - 49s - loss: 0.3370 - accuracy: 0.8896 - val_loss: 0.4524 - val_accur
acy: 0.8537

Epoch 9/30
 - 71s - loss: 0.3432 - accuracy: 0.8900 - val_loss: 0.4460 - val_accur
acy: 0.8609

Epoch 10/30
 - 57s - loss: 0.3625 - accuracy: 0.8810 - val_loss: 0.6277 - val_accur
acy: 0.8205

Epoch 11/30
 - 57s - loss: 0.3614 - accuracy: 0.8874 - val_loss: 0.4718 - val_accur
acy: 0.8738

Epoch 12/30
 - 53s - loss: 0.3426 - accuracy: 0.8909 - val_loss: 0.4717 - val_accur
acy: 0.8310

Epoch 13/30
 - 53s - loss: 0.3885 - accuracy: 0.8757 - val_loss: 0.4591 - val_accur
acy: 0.8670

Epoch 14/30
 - 49s - loss: 0.3482 - accuracy: 0.8862 - val_loss: 0.3799 - val_accur
acy: 0.8802

Epoch 15/30
 - 49s - loss: 0.3359 - accuracy: 0.8962 - val_loss: 0.3469 - val_accur
acy: 0.8768

Epoch 16/30
 - 50s - loss: 0.3319 - accuracy: 0.8950 - val_loss: 0.4545 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.8792

Epoch 17/30
 - 49s - loss: 0.3663 - accuracy: 0.8845 - val_loss: 0.4368 - val_accur
acy: 0.8548

Epoch 18/30
 - 50s - loss: 0.3513 - accuracy: 0.8840 - val_loss: 0.3866 - val_accur
acy: 0.8829

Epoch 19/30
 - 50s - loss: 0.3451 - accuracy: 0.8934 - val_loss: 0.6818 - val_accur
acy: 0.8466

Epoch 20/30
 - 53s - loss: 0.3607 - accuracy: 0.8879 - val_loss: 0.4097 - val_accur
acy: 0.8775

Epoch 21/30
 - 63s - loss: 0.3899 - accuracy: 0.8746 - val_loss: 0.3399 - val_accur
acy: 0.8894

Epoch 22/30
 - 48s - loss: 0.3194 - accuracy: 0.8930 - val_loss: 0.3653 - val_accur
acy: 0.8836

Epoch 23/30
 - 48s - loss: 0.3605 - accuracy: 0.8863 - val_loss: 0.4456 - val_accur
acy: 0.8269

Epoch 24/30
 - 48s - loss: 0.3681 - accuracy: 0.8832 - val_loss: 0.3675 - val_accur
acy: 0.8514

Epoch 25/30
 - 48s - loss: 0.3989 - accuracy: 0.8696 - val_loss: 0.3807 - val_accur
acy: 0.8799

Epoch 26/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 49s - loss: 0.3866 - accuracy: 0.8715 - val_loss: 0.4140 - val_accur
acy: 0.8415

Epoch 27/30
 - 50s - loss: 0.3793 - accuracy: 0.8823 - val_loss: 0.5881 - val_accur
acy: 0.8300

Epoch 28/30
 - 48s - loss: 0.4468 - accuracy: 0.8587 - val_loss: 0.8433 - val_accur
acy: 0.7676

Epoch 29/30
 - 48s - loss: 0.5099 - accuracy: 0.8341 - val_loss: 0.5565 - val_accur
acy: 0.8157

Epoch 30/30
 - 48s - loss: 0.4333 - accuracy: 0.8640 - val_loss: 0.4125 - val_accur
acy: 0.8806

Test accuracy:
0.8805565237998962
----------------------------------------------------------------------
--------------
  7%|▉         | 1/15 [25:47<6:00:58, 1547.02s/trial, best loss: -0.880
5565237998962]WARNING:tensorflow:Large dropout rate: 0.576545 (>0.5). I
n TensorFlow 2.x, dropout() uses dropout rate instead of keep_prob. Ple
ase ensure that this is intended.
WARNING:tensorflow:Large dropout rate: 0.510425 (>0.5). In TensorFlow
2.x, dropout() uses dropout rate instead of keep_prob. Please ensure th
at this is intended.
WARNING:tensorflow:Large dropout rate: 0.668505 (>0.5). In TensorFlow
2.x, dropout() uses dropout rate instead of keep_prob. Please ensure th
at this is intended.
Model: "sequential_2"

_________________________________________________________________

Layer (type)                 Output Shape              Param #
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
=================================================================
dense_4 (Dense)              (None, 128, 64)           640
_________________________________________________________________
batch_normalization_5 (Batch (None, 128, 64)           256
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
dense_5 (Dense)              (None, 128, 32)           2080
_________________________________________________________________
batch_normalization_6 (Batch (None, 128, 32)           128
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 64)           24832
_________________________________________________________________
batch_normalization_7 (Batch (None, 128, 64)           256
_________________________________________________________________
Dropout_3 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 36)                14544
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
batch_normalization_8 (Batch (None, 36)                144
_________________________________________________________________
Dropout_4 (Dropout)          (None, 36)                0
_________________________________________________________________
dense_6 (Dense)              (None, 6)                 222
=================================================================
Total params: 43,102
Trainable params: 42,710
Non-trainable params: 392
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
 - 60s - loss: 1.1793 - accuracy: 0.4709 - val_loss: 1.0624 - val_accur
acy: 0.4710
Epoch 2/30
 - 58s - loss: 1.0327 - accuracy: 0.5053 - val_loss: 0.8797 - val_accur
acy: 0.4944
Epoch 3/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 58s - loss: 0.9923 - accuracy: 0.5211 - val_loss: 0.8452 - val_accur
acy: 0.5741

Epoch 4/30

 - 58s - loss: 1.0432 - accuracy: 0.5022 - val_loss: 1.3252 - val_accur
acy: 0.4381

Epoch 5/30

 - 58s - loss: 1.0519 - accuracy: 0.4933 - val_loss: 1.5491 - val_accur
acy: 0.3258

Epoch 6/30

 - 58s - loss: 1.1053 - accuracy: 0.4871 - val_loss: 1.2856 - val_accur
acy: 0.5148

Epoch 7/30

 - 60s - loss: 1.0655 - accuracy: 0.4984 - val_loss: 1.0388 - val_accur
acy: 0.5008

Epoch 8/30

 - 58s - loss: 1.0635 - accuracy: 0.4992 - val_loss: 1.2665 - val_accur
acy: 0.3838

Epoch 9/30

 - 58s - loss: 1.0326 - accuracy: 0.5095 - val_loss: 1.5517 - val_accur
acy: 0.3163

Epoch 10/30

 - 58s - loss: 1.1577 - accuracy: 0.4702 - val_loss: 1.1143 - val_accur
acy: 0.5168

Epoch 11/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 58s - loss: 1.1439 - accuracy: 0.4845 - val_loss: 1.6053 - val_accur
acy: 0.4357

Epoch 12/30

 - 58s - loss: 1.1336 - accuracy: 0.4925 - val_loss: 1.1677 - val_accur
acy: 0.5439

Epoch 13/30

 - 58s - loss: 1.0777 - accuracy: 0.5076 - val_loss: 1.3248 - val_accur
acy: 0.4639

Epoch 14/30

 - 58s - loss: 1.0517 - accuracy: 0.4917 - val_loss: 1.5123 - val_accur
acy: 0.4476

Epoch 15/30

 - 58s - loss: 1.1926 - accuracy: 0.4539 - val_loss: 1.7160 - val_accur
acy: 0.3770

Epoch 16/30

 - 58s - loss: 1.2068 - accuracy: 0.4699 - val_loss: 1.7897 - val_accur
acy: 0.3770

Epoch 17/30

 - 58s - loss: 1.2482 - accuracy: 0.4521 - val_loss: 1.6399 - val_accur
acy: 0.3634

Epoch 18/30

 - 58s - loss: 1.1574 - accuracy: 0.4727 - val_loss: 1.8559 - val_accur
acy: 0.3841
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 19/30

 - 58s - loss: 1.1111 - accuracy: 0.4747 - val_loss: 1.3026 - val_accur
acy: 0.4822

Epoch 20/30

 - 58s - loss: 1.0582 - accuracy: 0.4883 - val_loss: 1.4885 - val_accur
acy: 0.3716

Epoch 21/30

 - 58s - loss: 1.0903 - accuracy: 0.4835 - val_loss: 1.2741 - val_accur
acy: 0.4737

Epoch 22/30

 - 58s - loss: 1.0872 - accuracy: 0.4929 - val_loss: 1.7199 - val_accur
acy: 0.2759

Epoch 23/30

 - 58s - loss: 1.0899 - accuracy: 0.4902 - val_loss: 1.2226 - val_accur
acy: 0.4028

Epoch 24/30

 - 58s - loss: 1.0825 - accuracy: 0.4933 - val_loss: 1.3425 - val_accur
acy: 0.4089

Epoch 25/30

 - 58s - loss: 1.0771 - accuracy: 0.4867 - val_loss: 1.0289 - val_accur
acy: 0.5249

Epoch 26/30

 - 58s - loss: 1.0474 - accuracy: 0.5018 - val_loss: 1.1171 - val_accur
acy: 0.5636
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 27/30

 - 58s - loss: 1.0676 - accuracy: 0.4980 - val_loss: 1.1817 - val_accur
acy: 0.4812

Epoch 28/30

 - 58s - loss: 1.0525 - accuracy: 0.4927 - val_loss: 1.2376 - val_accur
acy: 0.4564

Epoch 29/30

 - 57s - loss: 1.0754 - accuracy: 0.4869 - val_loss: 1.3008 - val_accur
acy: 0.4686

Epoch 30/30

 - 58s - loss: 1.0526 - accuracy: 0.4899 - val_loss: 1.2783 - val_accur
acy: 0.4289

Test accuracy:

0.4289107620716095

----------------------------------------------------------------------
--------------
 13%|└         | 2/15 [54:52<5:48:04, 1606.52s/trial, best loss: -0.880
5565237998962]WARNING:tensorflow:Large dropout rate: 0.631709 (>0.5). I
n TensorFlow 2.x, dropout() uses dropout rate instead of keep_prob. Ple
ase ensure that this is intended.
WARNING:tensorflow:Large dropout rate: 0.676855 (>0.5). In TensorFlow
2.x, dropout() uses dropout rate instead of keep_prob. Please ensure th
at this is intended.
Model: "sequential_3"

_________________________________________________________________

Layer (type)                 Output Shape              Param #
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
=================================================================
dense_7 (Dense)              (None, 128, 64)           640
_________________________________________________________________
batch_normalization_9 (Batch (None, 128, 64)           256
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
dense_8 (Dense)              (None, 128, 32)           2080
_________________________________________________________________
batch_normalization_10 (Batc (None, 128, 32)           128
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 64)           24832
_________________________________________________________________
batch_normalization_11 (Batc (None, 128, 64)           256
_________________________________________________________________
Dropout_3 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
LSTM_2 (LSTM)                (None, 36)                14544
_________________________________________________________________
batch_normalization_12 (Batc (None, 36)                144
_________________________________________________________________
Dropout_4 (Dropout)          (None, 36)                0
_________________________________________________________________
dense_9 (Dense)              (None, 6)                 222
=================================================================
Total params: 43,102
Trainable params: 42,710
Non-trainable params: 392
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
 - 59s - loss: 0.9400 - accuracy: 0.6034 - val_loss: 0.8558 - val_accur
acy: 0.6888
Epoch 2/30
 - 57s - loss: 0.6835 - accuracy: 0.7378 - val_loss: 0.7280 - val_accur
acy: 0.7625
Epoch 3/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 57s - loss: 0.5875 - accuracy: 0.7889 - val_loss: 2.2126 - val_accur
acy: 0.6600

Epoch 4/30

 - 57s - loss: 0.5281 - accuracy: 0.8228 - val_loss: 0.4505 - val_accur
acy: 0.8283

Epoch 5/30

 - 57s - loss: 0.4851 - accuracy: 0.8496 - val_loss: 0.5555 - val_accur
acy: 0.8358

Epoch 6/30

 - 57s - loss: 0.4710 - accuracy: 0.8534 - val_loss: 0.4240 - val_accur
acy: 0.8707

Epoch 7/30

 - 58s - loss: 0.4295 - accuracy: 0.8712 - val_loss: 0.4314 - val_accur
acy: 0.8337

Epoch 8/30

 - 58s - loss: 0.4224 - accuracy: 0.8713 - val_loss: 0.5381 - val_accur
acy: 0.8263

Epoch 9/30

 - 64s - loss: 0.4590 - accuracy: 0.8545 - val_loss: 0.4370 - val_accur
acy: 0.8381

Epoch 10/30

 - 65s - loss: 0.4716 - accuracy: 0.8511 - val_loss: 0.6027 - val_accur
acy: 0.8127
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 11/30

 - 74s - loss: 0.4873 - accuracy: 0.8439 - val_loss: 0.9106 - val_accur
acy: 0.7520

Epoch 12/30

 - 63s - loss: 0.4387 - accuracy: 0.8607 - val_loss: 0.6680 - val_accur
acy: 0.7777

Epoch 13/30

 - 64s - loss: 0.4484 - accuracy: 0.8672 - val_loss: 0.4272 - val_accur
acy: 0.8700

Epoch 14/30

 - 69s - loss: 0.4475 - accuracy: 0.8678 - val_loss: 0.6234 - val_accur
acy: 0.8161

Epoch 15/30

 - 63s - loss: 0.4867 - accuracy: 0.8536 - val_loss: 0.5604 - val_accur
acy: 0.8256

Epoch 16/30

 - 63s - loss: 0.4584 - accuracy: 0.8614 - val_loss: 0.5262 - val_accur
acy: 0.8293

Epoch 17/30

 - 63s - loss: 0.4658 - accuracy: 0.8653 - val_loss: 0.6148 - val_accur
acy: 0.8056

Epoch 18/30

 - 68s - loss: 0.4683 - accuracy: 0.8531 - val_loss: 0.4434 - val_accur
acy: 0.8487
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 19/30

 - 62s - loss: 0.4645 - accuracy: 0.8562 - val_loss: 0.4765 - val_accur
acy: 0.8514

Epoch 20/30

 - 62s - loss: 0.4511 - accuracy: 0.8649 - val_loss: 0.5058 - val_accur
acy: 0.8337

Epoch 21/30

 - 64s - loss: 0.4500 - accuracy: 0.8621 - val_loss: 0.6295 - val_accur
acy: 0.8069

Epoch 22/30

 - 63s - loss: 0.4926 - accuracy: 0.8433 - val_loss: 0.4764 - val_accur
acy: 0.8402

Epoch 23/30

 - 61s - loss: 0.4633 - accuracy: 0.8644 - val_loss: 0.3585 - val_accur
acy: 0.8894

Epoch 24/30

 - 60s - loss: 0.4874 - accuracy: 0.8539 - val_loss: 0.4697 - val_accur
acy: 0.8554

Epoch 25/30

 - 60s - loss: 0.4702 - accuracy: 0.8555 - val_loss: 0.4646 - val_accur
acy: 0.8476

Epoch 26/30

 - 61s - loss: 0.4693 - accuracy: 0.8541 - val_loss: 0.6048 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.7957

Epoch 27/30

 - 63s - loss: 0.4644 - accuracy: 0.8575 - val_loss: 0.6599 - val_accur
acy: 0.8174

Epoch 28/30

 - 61s - loss: 0.4499 - accuracy: 0.8535 - val_loss: 0.6183 - val_accur
acy: 0.8239

Epoch 29/30

 - 63s - loss: 0.4576 - accuracy: 0.8632 - val_loss: 0.4120 - val_accur
acy: 0.8558

Epoch 30/30

 - 58s - loss: 0.4707 - accuracy: 0.8527 - val_loss: 0.6979 - val_accur
acy: 0.7469

Test accuracy:

0.7468612194061279

---------------------------------------------------------------------
--------------
Model: "sequential_4"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_10 (Dense)             (None, 128, 64)           640

_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
batch_normalization_13 (Batc (None, 128, 64)          256
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 64)          0
_________________________________________________________________
dense_11 (Dense)             (None, 128, 32)          2080
_________________________________________________________________
batch_normalization_14 (Batc (None, 128, 32)          128
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 32)          0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 32)          8320
_________________________________________________________________
batch_normalization_15 (Batc (None, 128, 32)          128
_________________________________________________________________
Dropout_3 (Dropout)          (None, 128, 32)          0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 64)               24832
_________________________________________________________________
batch_normalization_16 (Batc (None, 64)               256
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
Dropout_4 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_12 (Dense)             (None, 6)                 390
=================================================================
Total params: 37,030
Trainable params: 36,646
Non-trainable params: 384
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
 - 55s - loss: 1.6578 - accuracy: 0.5004 - val_loss: 1.2423 - val_accur
acy: 0.5531
Epoch 2/30
 - 53s - loss: 0.8764 - accuracy: 0.6224 - val_loss: 0.9196 - val_accur
acy: 0.6793
Epoch 3/30
 - 53s - loss: 0.7638 - accuracy: 0.6767 - val_loss: 0.6992 - val_accur
acy: 0.7408
Epoch 4/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 57s - loss: 0.6681 - accuracy: 0.7539 - val_loss: 0.5693 - val_accur
acy: 0.8018

Epoch 5/30

 - 60s - loss: 0.5594 - accuracy: 0.8073 - val_loss: 0.4033 - val_accur
acy: 0.8565

Epoch 6/30

 - 57s - loss: 0.4485 - accuracy: 0.8591 - val_loss: 0.3711 - val_accur
acy: 0.8806

Epoch 7/30

 - 57s - loss: 0.3919 - accuracy: 0.8787 - val_loss: 0.3763 - val_accur
acy: 0.8982

Epoch 8/30

 - 57s - loss: 0.4082 - accuracy: 0.8825 - val_loss: 0.3690 - val_accur
acy: 0.9060

Epoch 9/30

 - 59s - loss: 0.3592 - accuracy: 0.8962 - val_loss: 0.2874 - val_accur
acy: 0.9172

Epoch 10/30

 - 58s - loss: 0.3617 - accuracy: 0.9003 - val_loss: 0.3657 - val_accur
acy: 0.9063

Epoch 11/30

 - 57s - loss: 0.3477 - accuracy: 0.9015 - val_loss: 0.3694 - val_accur
acy: 0.9114

Epoch 12/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 65s - loss: 0.3050 - accuracy: 0.9125 - val_loss: 0.7804 - val_accur
acy: 0.8792

Epoch 13/30

 - 62s - loss: 0.3357 - accuracy: 0.9087 - val_loss: 0.4405 - val_accur
acy: 0.9046

Epoch 14/30

 - 58s - loss: 0.3058 - accuracy: 0.9125 - val_loss: 0.3180 - val_accur
acy: 0.9270

Epoch 15/30

 - 58s - loss: 0.2866 - accuracy: 0.9150 - val_loss: 0.4619 - val_accur
acy: 0.9074

Epoch 16/30

 - 59s - loss: 0.3433 - accuracy: 0.9101 - val_loss: 0.2375 - val_accur
acy: 0.9281

Epoch 17/30

 - 65s - loss: 0.2945 - accuracy: 0.9169 - val_loss: 0.2494 - val_accur
acy: 0.9213

Epoch 18/30

 - 65s - loss: 0.2878 - accuracy: 0.9142 - val_loss: 0.2959 - val_accur
acy: 0.9260

Epoch 19/30

 - 64s - loss: 0.2583 - accuracy: 0.9128 - val_loss: 0.4247 - val_accur
acy: 0.9121
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 20/30

 - 65s - loss: 0.3063 - accuracy: 0.9191 - val_loss: 0.5038 - val_accur
acy: 0.9145

Epoch 21/30

 - 64s - loss: 0.2772 - accuracy: 0.9146 - val_loss: 0.4122 - val_accur
acy: 0.9240

Epoch 22/30

 - 64s - loss: 0.2669 - accuracy: 0.9187 - val_loss: 0.5217 - val_accur
acy: 0.9125

Epoch 23/30

 - 63s - loss: 0.2445 - accuracy: 0.9199 - val_loss: 0.3557 - val_accur
acy: 0.9247

Epoch 24/30

 - 56s - loss: 0.2450 - accuracy: 0.9261 - val_loss: 0.3571 - val_accur
acy: 0.9199

Epoch 25/30

 - 55s - loss: 0.2652 - accuracy: 0.9173 - val_loss: 0.2904 - val_accur
acy: 0.9318

Epoch 26/30

 - 55s - loss: 0.3127 - accuracy: 0.9189 - val_loss: 0.3392 - val_accur
acy: 0.9209

Epoch 27/30

 - 55s - loss: 0.2973 - accuracy: 0.9184 - val_loss: 0.2218 - val_accur
acy: 0.9352
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 28/30

 - 55s - loss: 0.2282 - accuracy: 0.9301 - val_loss: 0.2623 - val_accur
acy: 0.9243

Epoch 29/30

 - 55s - loss: 0.2876 - accuracy: 0.9249 - val_loss: 0.3684 - val_accur
acy: 0.9128

Epoch 30/30

 - 56s - loss: 0.2608 - accuracy: 0.9251 - val_loss: 0.3941 - val_accur
acy: 0.9175

Test accuracy:

0.9175432920455933

---------------------------------------------------------------------
--------------
Model: "sequential_5"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_13 (Dense)             (None, 128, 64)           640

_________________________________________________________________

batch_normalization_17 (Batc (None, 128, 64)           256

_________________________________________________________________

Dropout_1 (Dropout)          (None, 128, 64)           0
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
dense_14 (Dense)             (None, 128, 32)           2080
_________________________________________________________________
batch_normalization_18 (Batc (None, 128, 32)           128
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 36)           9936
_________________________________________________________________
batch_normalization_19 (Batc (None, 128, 36)           144
_________________________________________________________________
Dropout_3 (Dropout)          (None, 128, 36)           0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 64)                25856
_________________________________________________________________
batch_normalization_20 (Batc (None, 64)                256
_________________________________________________________________
Dropout_4 (Dropout)          (None, 64)                0
_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
dense_15 (Dense)             (None, 6)                 390

=================================================================

Total params: 39,686

Trainable params: 39,294

Non-trainable params: 392

_________________________________________________________________

None

Train on 7352 samples, validate on 2947 samples

Epoch 1/30

 - 64s - loss: 1.1034 - accuracy: 0.5241 - val_loss: 0.9890 - val_accur
acy: 0.6010

Epoch 2/30

 - 60s - loss: 0.9169 - accuracy: 0.5903 - val_loss: 0.9387 - val_accur
acy: 0.5667

Epoch 3/30

 - 64s - loss: 1.0102 - accuracy: 0.5566 - val_loss: 1.2246 - val_accur
acy: 0.5300

Epoch 4/30

 - 63s - loss: 1.0162 - accuracy: 0.5732 - val_loss: 1.3355 - val_accur
acy: 0.5219

Epoch 5/30

 - 63s - loss: 1.0426 - accuracy: 0.5307 - val_loss: 1.2398 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.5361

Epoch 6/30

 - 62s - loss: 1.0048 - accuracy: 0.5501 - val_loss: 1.9403 - val_accur
acy: 0.4618

Epoch 7/30

 - 59s - loss: 1.1686 - accuracy: 0.4766 - val_loss: 1.4324 - val_accur
acy: 0.3417

Epoch 8/30

 - 62s - loss: 1.3678 - accuracy: 0.3912 - val_loss: 1.3653 - val_accur
acy: 0.4479

Epoch 9/30

 - 62s - loss: 1.3534 - accuracy: 0.3974 - val_loss: 2.3146 - val_accur
acy: 0.2667

Epoch 10/30

 - 65s - loss: 1.6139 - accuracy: 0.3156 - val_loss: 1.9652 - val_accur
acy: 0.3186

Epoch 11/30

 - 59s - loss: 1.7133 - accuracy: 0.2686 - val_loss: 2.3924 - val_accur
acy: 0.3712

Epoch 12/30

 - 62s - loss: 1.5795 - accuracy: 0.3226 - val_loss: 1.9244 - val_accur
acy: 0.3570

Epoch 13/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 56s - loss: 1.7039 - accuracy: 0.2904 - val_loss: 1.5071 - val_accur
acy: 0.3536

Epoch 14/30

 - 61s - loss: 1.6590 - accuracy: 0.3158 - val_loss: 1.7060 - val_accur
acy: 0.3451

Epoch 15/30

 - 60s - loss: 1.6399 - accuracy: 0.3303 - val_loss: 1.7077 - val_accur
acy: 0.2782

Epoch 16/30

 - 60s - loss: 1.6226 - accuracy: 0.3278 - val_loss: 1.5980 - val_accur
acy: 0.3661

Epoch 17/30

 - 64s - loss: 1.6605 - accuracy: 0.3097 - val_loss: 1.4460 - val_accur
acy: 0.4242

Epoch 18/30

 - 63s - loss: 1.6132 - accuracy: 0.3196 - val_loss: 1.5400 - val_accur
acy: 0.3665

Epoch 19/30

 - 60s - loss: 1.6254 - accuracy: 0.3111 - val_loss: 1.5536 - val_accur
acy: 0.3851

Epoch 20/30

 - 60s - loss: 1.6063 - accuracy: 0.3300 - val_loss: 1.7013 - val_accur
acy: 0.3356

Epoch 21/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 60s - loss: 1.6526 - accuracy: 0.2964 - val_loss: 1.6747 - val_accur
acy: 0.2637

Epoch 22/30

 - 60s - loss: 1.6476 - accuracy: 0.3048 - val_loss: 2.4480 - val_accur
acy: 0.2243

Epoch 23/30

 - 60s - loss: 1.6405 - accuracy: 0.3001 - val_loss: 2.7411 - val_accur
acy: 0.2060

Epoch 24/30

 - 60s - loss: 1.6726 - accuracy: 0.2933 - val_loss: 2.0987 - val_accur
acy: 0.1887

Epoch 25/30

 - 60s - loss: 1.7552 - accuracy: 0.2522 - val_loss: 2.0452 - val_accur
acy: 0.1758

Epoch 26/30

 - 60s - loss: 1.7838 - accuracy: 0.2257 - val_loss: 1.7127 - val_accur
acy: 0.2155

Epoch 27/30

 - 60s - loss: 1.8171 - accuracy: 0.2010 - val_loss: 1.7974 - val_accur
acy: 0.2358

Epoch 28/30

 - 60s - loss: 1.8160 - accuracy: 0.1937 - val_loss: 1.8648 - val_accur
acy: 0.1795
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 29/30

 - 60s - loss: 1.7983 - accuracy: 0.2193 - val_loss: 2.0110 - val_accur
acy: 0.1934

Epoch 30/30

 - 61s - loss: 1.7977 - accuracy: 0.2218 - val_loss: 1.9540 - val_accur
acy: 0.0302

Test accuracy:

0.030200203880667686

---------------------------------------------------------------------
--------------
Model: "sequential_6"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_16 (Dense)             (None, 128, 64)           640

_________________________________________________________________

batch_normalization_21 (Batc (None, 128, 64)           256

_________________________________________________________________

Dropout_1 (Dropout)          (None, 128, 64)           0

_________________________________________________________________

dense_17 (Dense)             (None, 128, 32)           2080

_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
batch_normalization_22 (Batc (None, 128, 32)          128
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 32)          0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 36)          9936
_________________________________________________________________
batch_normalization_23 (Batc (None, 128, 36)          144
_________________________________________________________________
Dropout_3 (Dropout)          (None, 128, 36)          0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 32)               8832
_________________________________________________________________
batch_normalization_24 (Batc (None, 32)               128
_________________________________________________________________
Dropout_4 (Dropout)          (None, 32)               0
_________________________________________________________________
dense_18 (Dense)             (None, 6)                198
=================================================================
Total params: 22,342
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Trainable params: 22,014

Non-trainable params: 328

_________________________________________________________________

None

Train on 7352 samples, validate on 2947 samples

Epoch 1/30

 - 57s - loss: 0.8503 - accuracy: 0.6269 - val_loss: 0.9391 - val_accur
acy: 0.5748

Epoch 2/30

 - 53s - loss: 0.6917 - accuracy: 0.7097 - val_loss: 0.7446 - val_accur
acy: 0.6617

Epoch 3/30

 - 53s - loss: 0.6300 - accuracy: 0.7329 - val_loss: 0.6188 - val_accur
acy: 0.7133

Epoch 4/30

 - 53s - loss: 0.5795 - accuracy: 0.7541 - val_loss: 0.7389 - val_accur
acy: 0.6797

Epoch 5/30

 - 57s - loss: 0.6580 - accuracy: 0.7231 - val_loss: 0.5883 - val_accur
acy: 0.7621

Epoch 6/30

 - 49s - loss: 0.5551 - accuracy: 0.7621 - val_loss: 0.6961 - val_accur
acy: 0.6837
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 7/30

 - 48s - loss: 0.6621 - accuracy: 0.7131 - val_loss: 0.7385 - val_accur
acy: 0.6949

Epoch 8/30

 - 48s - loss: 0.7397 - accuracy: 0.6915 - val_loss: 0.9522 - val_accur
acy: 0.6125

Epoch 9/30

 - 48s - loss: 0.6541 - accuracy: 0.7273 - val_loss: 0.5735 - val_accur
acy: 0.7448

Epoch 10/30

 - 48s - loss: 0.5942 - accuracy: 0.7382 - val_loss: 0.5666 - val_accur
acy: 0.7027

Epoch 11/30

 - 48s - loss: 0.5702 - accuracy: 0.7632 - val_loss: 0.5488 - val_accur
acy: 0.7391

Epoch 12/30

 - 48s - loss: 0.5795 - accuracy: 0.7519 - val_loss: 0.5896 - val_accur
acy: 0.6675

Epoch 13/30

 - 48s - loss: 0.5870 - accuracy: 0.7477 - val_loss: 0.7473 - val_accur
acy: 0.6084

Epoch 14/30

 - 48s - loss: 0.5943 - accuracy: 0.7440 - val_loss: 0.6745 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.6858

Epoch 15/30

 - 48s - loss: 0.6357 - accuracy: 0.7305 - val_loss: 0.6367 - val_accur
acy: 0.6885

Epoch 16/30

 - 48s - loss: 0.6081 - accuracy: 0.7314 - val_loss: 0.7134 - val_accur
acy: 0.6858

Epoch 17/30

 - 48s - loss: 0.5884 - accuracy: 0.7480 - val_loss: 0.8367 - val_accur
acy: 0.6166

Epoch 18/30

 - 48s - loss: 0.6986 - accuracy: 0.7073 - val_loss: 0.8487 - val_accur
acy: 0.6885

Epoch 19/30

 - 48s - loss: 0.6271 - accuracy: 0.7335 - val_loss: 1.3227 - val_accur
acy: 0.6166

Epoch 20/30

 - 48s - loss: 0.6905 - accuracy: 0.6956 - val_loss: 0.9970 - val_accur
acy: 0.5816

Epoch 21/30

 - 48s - loss: 0.7400 - accuracy: 0.6800 - val_loss: 1.1698 - val_accur
acy: 0.5728

Epoch 22/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 48s - loss: 0.7958 - accuracy: 0.6644 - val_loss: 1.5467 - val_accur
acy: 0.4299

Epoch 23/30

 - 48s - loss: 0.8619 - accuracy: 0.6299 - val_loss: 1.8112 - val_accur
acy: 0.4035

Epoch 24/30

 - 48s - loss: 0.8395 - accuracy: 0.6291 - val_loss: 1.5424 - val_accur
acy: 0.4706

Epoch 25/30

 - 48s - loss: 0.9364 - accuracy: 0.6005 - val_loss: 2.3090 - val_accur
acy: 0.3621

Epoch 26/30

 - 48s - loss: 0.9568 - accuracy: 0.5744 - val_loss: 2.2365 - val_accur
acy: 0.4072

Epoch 27/30

 - 48s - loss: 0.8289 - accuracy: 0.6274 - val_loss: 1.8732 - val_accur
acy: 0.4530

Epoch 28/30

 - 48s - loss: 0.8076 - accuracy: 0.6372 - val_loss: 1.2199 - val_accur
acy: 0.5830

Epoch 29/30

 - 48s - loss: 0.8015 - accuracy: 0.6408 - val_loss: 2.1239 - val_accur
acy: 0.3465

Epoch 30/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 48s - loss: 0.8423 - accuracy: 0.6202 - val_loss: 1.6042 - val_accur
acy: 0.4313

Test accuracy:

0.4312860667705536

---------------------------------------------------------------------
--------------
Model: "sequential_7"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_19 (Dense)             (None, 128, 64)           640

_________________________________________________________________

batch_normalization_25 (Batc (None, 128, 64)           256

_________________________________________________________________

Dropout_1 (Dropout)          (None, 128, 64)           0

_________________________________________________________________

dense_20 (Dense)             (None, 128, 32)           2080

_________________________________________________________________

batch_normalization_26 (Batc (None, 128, 32)           128

_________________________________________________________________

Dropout_2 (Dropout)          (None, 128, 32)           0
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 64)           24832
_________________________________________________________________
batch_normalization_27 (Batc (None, 128, 64)           256
_________________________________________________________________
Dropout_3 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 64)                33024
_________________________________________________________________
batch_normalization_28 (Batc (None, 64)                256
_________________________________________________________________
Dropout_4 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_21 (Dense)             (None, 6)                 390
=================================================================
Total params: 61,862
Trainable params: 61,414
Non-trainable params: 448
_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
None

Train on 7352 samples, validate on 2947 samples

Epoch 1/30

 - 67s - loss: 0.9220 - accuracy: 0.6032 - val_loss: 0.8697 - val_accur
acy: 0.6386

Epoch 2/30

 - 64s - loss: 0.7599 - accuracy: 0.6766 - val_loss: 1.6473 - val_accur
acy: 0.5935

Epoch 3/30

 - 64s - loss: 0.5394 - accuracy: 0.7786 - val_loss: 0.7199 - val_accur
acy: 0.7845

Epoch 4/30

 - 64s - loss: 0.4812 - accuracy: 0.8205 - val_loss: 0.6065 - val_accur
acy: 0.7852

Epoch 5/30

 - 64s - loss: 0.4227 - accuracy: 0.8473 - val_loss: 2.0495 - val_accur
acy: 0.5480

Epoch 6/30

 - 65s - loss: 0.3904 - accuracy: 0.8591 - val_loss: 0.4915 - val_accur
acy: 0.8436

Epoch 7/30

 - 64s - loss: 0.3574 - accuracy: 0.8716 - val_loss: 0.3233 - val_accur
acy: 0.8775
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 8/30

 - 64s - loss: 0.3014 - accuracy: 0.8909 - val_loss: 0.5397 - val_accur
acy: 0.8317

Epoch 9/30

 - 64s - loss: 0.3277 - accuracy: 0.8885 - val_loss: 0.3561 - val_accur
acy: 0.8884

Epoch 10/30

 - 64s - loss: 0.2761 - accuracy: 0.9030 - val_loss: 0.3039 - val_accur
acy: 0.8907

Epoch 11/30

 - 65s - loss: 0.2501 - accuracy: 0.9113 - val_loss: 0.3018 - val_accur
acy: 0.8996

Epoch 12/30

 - 65s - loss: 0.3621 - accuracy: 0.8765 - val_loss: 0.4025 - val_accur
acy: 0.8639

Epoch 13/30

 - 64s - loss: 0.2995 - accuracy: 0.8953 - val_loss: 0.2924 - val_accur
acy: 0.8877

Epoch 14/30

 - 64s - loss: 0.2558 - accuracy: 0.9087 - val_loss: 0.5005 - val_accur
acy: 0.8544

Epoch 15/30

 - 64s - loss: 0.2574 - accuracy: 0.9097 - val_loss: 0.3656 - val_accur
acy: 0.8887
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 16/30

 - 64s - loss: 0.2503 - accuracy: 0.9121 - val_loss: 0.3064 - val_accur
acy: 0.8887

Epoch 17/30

 - 64s - loss: 0.2554 - accuracy: 0.9060 - val_loss: 0.3816 - val_accur
acy: 0.8687

Epoch 18/30

 - 64s - loss: 0.2530 - accuracy: 0.9076 - val_loss: 0.2830 - val_accur
acy: 0.9016

Epoch 19/30

 - 65s - loss: 0.2562 - accuracy: 0.9076 - val_loss: 0.3981 - val_accur
acy: 0.8738

Epoch 20/30

 - 65s - loss: 0.2352 - accuracy: 0.9116 - val_loss: 0.3408 - val_accur
acy: 0.8941

Epoch 21/30

 - 65s - loss: 0.2678 - accuracy: 0.9112 - val_loss: 0.4491 - val_accur
acy: 0.8649

Epoch 22/30

 - 65s - loss: 0.2522 - accuracy: 0.9044 - val_loss: 0.5324 - val_accur
acy: 0.8364

Epoch 23/30

 - 65s - loss: 0.2298 - accuracy: 0.9098 - val_loss: 0.4630 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.8772

Epoch 24/30

 - 65s - loss: 0.2411 - accuracy: 0.9132 - val_loss: 0.5567 - val_accur
acy: 0.8266

Epoch 25/30

 - 65s - loss: 0.2580 - accuracy: 0.9038 - val_loss: 0.4130 - val_accur
acy: 0.8829

Epoch 26/30

 - 65s - loss: 0.2357 - accuracy: 0.9165 - val_loss: 0.3725 - val_accur
acy: 0.8785

Epoch 27/30

 - 65s - loss: 0.2219 - accuracy: 0.9187 - val_loss: 0.2545 - val_accur
acy: 0.9023

Epoch 28/30

 - 65s - loss: 0.2508 - accuracy: 0.9064 - val_loss: 0.3964 - val_accur
acy: 0.8711

Epoch 29/30

 - 65s - loss: 0.2445 - accuracy: 0.9115 - val_loss: 0.3518 - val_accur
acy: 0.8517

Epoch 30/30

 - 65s - loss: 0.2393 - accuracy: 0.9135 - val_loss: 0.3750 - val_accur
acy: 0.8907

Test accuracy:
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
0.8907363414764404

_________________________________________________________________
--------------
Model: "sequential_8"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_22 (Dense)             (None, 128, 64)           640

_________________________________________________________________
batch_normalization_29 (Batc (None, 128, 64)           256

_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 64)           0

_________________________________________________________________
dense_23 (Dense)             (None, 128, 32)           2080

_________________________________________________________________
batch_normalization_30 (Batc (None, 128, 32)           128

_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 32)           0

_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 32)           8320

_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
batch_normalization_31 (Batc (None, 128, 32)          128
_________________________________________________________________
Dropout_3 (Dropout)          (None, 128, 32)          0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 64)               24832
_________________________________________________________________
batch_normalization_32 (Batc (None, 64)               256
_________________________________________________________________
Dropout_4 (Dropout)          (None, 64)               0
_________________________________________________________________
dense_24 (Dense)             (None, 6)                390
=================================================================
Total params: 37,030
Trainable params: 36,646
Non-trainable params: 384
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 57s - loss: 0.9126 - accuracy: 0.6172 - val_loss: 1.4713 - val_accur
acy: 0.5521

Epoch 2/30

 - 53s - loss: 0.6683 - accuracy: 0.7227 - val_loss: 0.7178 - val_accur
acy: 0.7072

Epoch 3/30

 - 53s - loss: 0.5503 - accuracy: 0.7845 - val_loss: 0.5385 - val_accur
acy: 0.8107

Epoch 4/30

 - 53s - loss: 0.4093 - accuracy: 0.8519 - val_loss: 0.4236 - val_accur
acy: 0.8565

Epoch 5/30

 - 53s - loss: 0.4012 - accuracy: 0.8592 - val_loss: 0.4292 - val_accur
acy: 0.8337

Epoch 6/30

 - 53s - loss: 0.3264 - accuracy: 0.8774 - val_loss: 0.3853 - val_accur
acy: 0.8571

Epoch 7/30

 - 53s - loss: 0.3103 - accuracy: 0.8923 - val_loss: 0.2872 - val_accur
acy: 0.8958

Epoch 8/30

 - 53s - loss: 0.3156 - accuracy: 0.8905 - val_loss: 0.3983 - val_accur
acy: 0.8561

Epoch 9/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 55s - loss: 0.3241 - accuracy: 0.8904 - val_loss: 0.5232 - val_accur
acy: 0.8259

Epoch 10/30

 - 58s - loss: 0.2697 - accuracy: 0.9051 - val_loss: 0.3330 - val_accur
acy: 0.8938

Epoch 11/30

 - 58s - loss: 0.2449 - accuracy: 0.9082 - val_loss: 0.3789 - val_accur
acy: 0.8819

Epoch 12/30

 - 59s - loss: 0.2446 - accuracy: 0.9121 - val_loss: 0.4726 - val_accur
acy: 0.8517

Epoch 13/30

 - 59s - loss: 0.2371 - accuracy: 0.9192 - val_loss: 0.3187 - val_accur
acy: 0.8860

Epoch 14/30

 - 58s - loss: 0.2171 - accuracy: 0.9204 - val_loss: 0.4191 - val_accur
acy: 0.8843

Epoch 15/30

 - 59s - loss: 0.2131 - accuracy: 0.9208 - val_loss: 0.3148 - val_accur
acy: 0.9050

Epoch 16/30

 - 58s - loss: 0.2677 - accuracy: 0.9041 - val_loss: 0.3301 - val_accur
acy: 0.8904
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 17/30

 - 59s - loss: 0.2240 - accuracy: 0.9169 - val_loss: 0.3624 - val_accur
acy: 0.8836

Epoch 18/30

 - 59s - loss: 0.2103 - accuracy: 0.9217 - val_loss: 0.4138 - val_accur
acy: 0.8595

Epoch 19/30

 - 58s - loss: 0.2035 - accuracy: 0.9200 - val_loss: 0.4087 - val_accur
acy: 0.8728

Epoch 20/30

 - 58s - loss: 0.2117 - accuracy: 0.9223 - val_loss: 0.5546 - val_accur
acy: 0.8259

Epoch 21/30

 - 58s - loss: 0.2108 - accuracy: 0.9204 - val_loss: 0.3754 - val_accur
acy: 0.8721

Epoch 22/30

 - 59s - loss: 0.1934 - accuracy: 0.9314 - val_loss: 0.3235 - val_accur
acy: 0.8778

Epoch 23/30

 - 59s - loss: 0.2312 - accuracy: 0.9170 - val_loss: 0.3208 - val_accur
acy: 0.8918

Epoch 24/30

 - 59s - loss: 0.2134 - accuracy: 0.9259 - val_loss: 0.3122 - val_accur
acy: 0.9026
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 25/30

 - 58s - loss: 0.2274 - accuracy: 0.9173 - val_loss: 0.3929 - val_accur
acy: 0.8504

Epoch 26/30

 - 58s - loss: 0.2216 - accuracy: 0.9200 - val_loss: 0.3982 - val_accur
acy: 0.8870

Epoch 27/30

 - 58s - loss: 0.2141 - accuracy: 0.9212 - val_loss: 0.2774 - val_accur
acy: 0.9053

Epoch 28/30

 - 59s - loss: 0.2101 - accuracy: 0.9223 - val_loss: 0.3024 - val_accur
acy: 0.8901

Epoch 29/30

 - 58s - loss: 0.1867 - accuracy: 0.9295 - val_loss: 0.3049 - val_accur
acy: 0.8694

Epoch 30/30

 - 58s - loss: 0.2216 - accuracy: 0.9238 - val_loss: 0.3495 - val_accur
acy: 0.8843

Test accuracy:

0.8842890858650208

---------------------------------------------------------------------
--------------
Model: "sequential_9"
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_25 (Dense)             (None, 128, 64)           640
_________________________________________________________________
batch_normalization_33 (Batc (None, 128, 64)           256
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
dense_26 (Dense)             (None, 128, 32)           2080
_________________________________________________________________
batch_normalization_34 (Batc (None, 128, 32)           128
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 64)           24832
_________________________________________________________________
batch_normalization_35 (Batc (None, 128, 64)           256
_________________________________________________________________
Dropout_3 (Dropout)          (None, 128, 64)           0
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
LSTM_2 (LSTM)                (None, 64)                33024
_________________________________________________________________
batch_normalization_36 (Batc (None, 64)                256
_________________________________________________________________
Dropout_4 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_27 (Dense)             (None, 6)                 390
=================================================================
Total params: 61,862
Trainable params: 61,414
Non-trainable params: 448
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
 - 73s - loss: 0.7395 - accuracy: 0.7012 - val_loss: 0.5584 - val_accur
acy: 0.8073
Epoch 2/30
 - 70s - loss: 0.4346 - accuracy: 0.8467 - val_loss: 0.3770 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.8473

Epoch 3/30

 - 71s - loss: 0.3482 - accuracy: 0.8785 - val_loss: 0.5664 - val_accur
acy: 0.8354

Epoch 4/30

 - 70s - loss: 0.3270 - accuracy: 0.8913 - val_loss: 0.4386 - val_accur
acy: 0.8514

Epoch 5/30

 - 72s - loss: 0.2970 - accuracy: 0.8987 - val_loss: 0.4209 - val_accur
acy: 0.8368

Epoch 6/30

 - 70s - loss: 0.2856 - accuracy: 0.9041 - val_loss: 0.2866 - val_accur
acy: 0.8850

Epoch 7/30

 - 70s - loss: 0.2834 - accuracy: 0.9048 - val_loss: 0.2837 - val_accur
acy: 0.9043

Epoch 8/30

 - 70s - loss: 0.2826 - accuracy: 0.9079 - val_loss: 0.4097 - val_accur
acy: 0.8799

Epoch 9/30

 - 71s - loss: 0.2823 - accuracy: 0.9075 - val_loss: 0.2774 - val_accur
acy: 0.9016

Epoch 10/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 70s - loss: 0.2762 - accuracy: 0.9044 - val_loss: 0.3561 - val_accur
acy: 0.8870

Epoch 11/30

 - 70s - loss: 0.2589 - accuracy: 0.9089 - val_loss: 0.4253 - val_accur
acy: 0.8442

Epoch 12/30

 - 70s - loss: 0.2801 - accuracy: 0.9059 - val_loss: 0.3139 - val_accur
acy: 0.9023

Epoch 13/30

 - 70s - loss: 0.2542 - accuracy: 0.9127 - val_loss: 0.2959 - val_accur
acy: 0.8873

Epoch 14/30

 - 71s - loss: 0.2534 - accuracy: 0.9146 - val_loss: 0.5349 - val_accur
acy: 0.8385

Epoch 15/30

 - 70s - loss: 0.2593 - accuracy: 0.9124 - val_loss: 0.4105 - val_accur
acy: 0.8744

Epoch 16/30

 - 70s - loss: 0.2538 - accuracy: 0.9161 - val_loss: 0.3436 - val_accur
acy: 0.8904

Epoch 17/30

 - 70s - loss: 0.2626 - accuracy: 0.9068 - val_loss: 0.3245 - val_accur
acy: 0.8907

Epoch 18/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 70s - loss: 0.2636 - accuracy: 0.9087 - val_loss: 0.3713 - val_accur
acy: 0.8918

Epoch 19/30

 - 70s - loss: 0.2565 - accuracy: 0.9131 - val_loss: 0.3054 - val_accur
acy: 0.8985

Epoch 20/30

 - 70s - loss: 0.2598 - accuracy: 0.9138 - val_loss: 0.3750 - val_accur
acy: 0.8873

Epoch 21/30

 - 70s - loss: 0.2644 - accuracy: 0.9095 - val_loss: 0.4678 - val_accur
acy: 0.8429

Epoch 22/30

 - 70s - loss: 0.2882 - accuracy: 0.9036 - val_loss: 0.5357 - val_accur
acy: 0.8375

Epoch 23/30

 - 70s - loss: 0.2808 - accuracy: 0.9081 - val_loss: 0.3799 - val_accur
acy: 0.8877

Epoch 24/30

 - 71s - loss: 0.2460 - accuracy: 0.9157 - val_loss: 0.6291 - val_accur
acy: 0.8100

Epoch 25/30

 - 70s - loss: 0.2832 - accuracy: 0.8974 - val_loss: 0.5368 - val_accur
acy: 0.8334
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 26/30

 - 70s - loss: 0.2575 - accuracy: 0.9138 - val_loss: 0.6310 - val_accur
acy: 0.8086

Epoch 27/30

 - 70s - loss: 0.2781 - accuracy: 0.9011 - val_loss: 0.3137 - val_accur
acy: 0.8948

Epoch 28/30

 - 70s - loss: 0.2687 - accuracy: 0.9098 - val_loss: 0.3899 - val_accur
acy: 0.8663

Epoch 29/30

 - 71s - loss: 0.2720 - accuracy: 0.9085 - val_loss: 0.4400 - val_accur
acy: 0.8646

Epoch 30/30

 - 70s - loss: 0.2997 - accuracy: 0.8968 - val_loss: 0.6388 - val_accur
acy: 0.7866

Test accuracy:

0.7865626215934753

---------------------------------------------------------------------
--------------
Model: "sequential_10"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
dense_28 (Dense)             (None, 128, 64)           640
_________________________________________________________________
batch_normalization_37 (Batc (None, 128, 64)           256
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
dense_29 (Dense)             (None, 128, 32)           2080
_________________________________________________________________
batch_normalization_38 (Batc (None, 128, 32)           128
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 64)           24832
_________________________________________________________________
batch_normalization_39 (Batc (None, 128, 64)           256
_________________________________________________________________
Dropout_3 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 32)                12416
_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
batch_normalization_40 (Batc (None, 32)                128
_________________________________________________________________
Dropout_4 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_30 (Dense)             (None, 6)                 198
=================================================================
Total params: 40,934
Trainable params: 40,550
Non-trainable params: 384
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
 - 64s - loss: 1.1048 - accuracy: 0.5204 - val_loss: 0.8194 - val_accur
acy: 0.6043
Epoch 2/30
 - 60s - loss: 0.9223 - accuracy: 0.6061 - val_loss: 0.7940 - val_accur
acy: 0.6854
Epoch 3/30
 - 59s - loss: 0.8207 - accuracy: 0.6579 - val_loss: 0.6677 - val_accur
acy: 0.6770
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 4/30

 - 60s - loss: 0.8223 - accuracy: 0.6726 - val_loss: 0.8223 - val_accur
acy: 0.6661

Epoch 5/30

 - 60s - loss: 0.8389 - accuracy: 0.6697 - val_loss: 0.5823 - val_accur
acy: 0.6994

Epoch 6/30

 - 60s - loss: 0.7970 - accuracy: 0.6842 - val_loss: 0.7829 - val_accur
acy: 0.6128

Epoch 7/30

 - 60s - loss: 0.8324 - accuracy: 0.6593 - val_loss: 0.7943 - val_accur
acy: 0.6111

Epoch 8/30

 - 60s - loss: 0.7626 - accuracy: 0.6838 - val_loss: 0.8160 - val_accur
acy: 0.6532

Epoch 9/30

 - 60s - loss: 0.7465 - accuracy: 0.6974 - val_loss: 0.6602 - val_accur
acy: 0.6783

Epoch 10/30

 - 60s - loss: 0.7755 - accuracy: 0.6926 - val_loss: 0.8071 - val_accur
acy: 0.6563

Epoch 11/30

 - 60s - loss: 0.8028 - accuracy: 0.6817 - val_loss: 1.2169 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.6128

Epoch 12/30

 - 60s - loss: 0.7718 - accuracy: 0.6887 - val_loss: 1.3413 - val_accur
acy: 0.5830

Epoch 13/30

 - 60s - loss: 0.9002 - accuracy: 0.6382 - val_loss: 0.9511 - val_accur
acy: 0.6237

Epoch 14/30

 - 60s - loss: 0.7406 - accuracy: 0.6934 - val_loss: 0.7618 - val_accur
acy: 0.7520

Epoch 15/30

 - 60s - loss: 0.8221 - accuracy: 0.6930 - val_loss: 0.6428 - val_accur
acy: 0.7197

Epoch 16/30

 - 60s - loss: 0.8271 - accuracy: 0.6767 - val_loss: 0.7276 - val_accur
acy: 0.7506

Epoch 17/30

 - 60s - loss: 0.8445 - accuracy: 0.6591 - val_loss: 0.6051 - val_accur
acy: 0.7319

Epoch 18/30

 - 60s - loss: 0.8151 - accuracy: 0.6782 - val_loss: 0.5983 - val_accur
acy: 0.7499

Epoch 19/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 60s - loss: 0.8555 - accuracy: 0.6269 - val_loss: 1.2881 - val_accur
acy: 0.5884

Epoch 20/30

 - 60s - loss: 0.9008 - accuracy: 0.6364 - val_loss: 0.7047 - val_accur
acy: 0.7146

Epoch 21/30

 - 60s - loss: 0.8795 - accuracy: 0.6269 - val_loss: 0.8454 - val_accur
acy: 0.6997

Epoch 22/30

 - 60s - loss: 0.8849 - accuracy: 0.6421 - val_loss: 0.9188 - val_accur
acy: 0.6583

Epoch 23/30

 - 59s - loss: 0.8459 - accuracy: 0.6609 - val_loss: 0.9406 - val_accur
acy: 0.6837

Epoch 24/30

 - 59s - loss: 0.8399 - accuracy: 0.6542 - val_loss: 1.6874 - val_accur
acy: 0.5365

Epoch 25/30

 - 60s - loss: 0.9019 - accuracy: 0.6330 - val_loss: 1.6647 - val_accur
acy: 0.5982

Epoch 26/30

 - 60s - loss: 0.8736 - accuracy: 0.6465 - val_loss: 1.2725 - val_accur
acy: 0.5623

Epoch 27/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 60s - loss: 0.8798 - accuracy: 0.6409 - val_loss: 1.0087 - val_accur
acy: 0.6281

Epoch 28/30

 - 60s - loss: 0.9159 - accuracy: 0.6276 - val_loss: 1.7170 - val_accur
acy: 0.5219

Epoch 29/30

 - 60s - loss: 0.8964 - accuracy: 0.6360 - val_loss: 1.5165 - val_accur
acy: 0.5243

Epoch 30/30

 - 60s - loss: 0.8922 - accuracy: 0.6295 - val_loss: 1.5501 - val_accur
acy: 0.5378

Test accuracy:

0.5378350615501404

---------------------------------------------------------------------
--------------
Model: "sequential_11"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_31 (Dense)             (None, 128, 64)           640

_________________________________________________________________

batch_normalization_41 (Batc (None, 128, 64)           256
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
dense_32 (Dense)             (None, 128, 32)           2080
_________________________________________________________________
batch_normalization_42 (Batc (None, 128, 32)           128
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 32)           8320
_________________________________________________________________
batch_normalization_43 (Batc (None, 128, 32)           128
_________________________________________________________________
Dropout_3 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 64)                24832
_________________________________________________________________
batch_normalization_44 (Batc (None, 64)                256
_________________________________________________________________
Dropout_4 (Dropout)          (None, 64)                0
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
dense_33 (Dense)             (None, 6)                 390
=================================================================
Total params: 37,030
Trainable params: 36,646
Non-trainable params: 384
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
 - 61s - loss: 0.7386 - accuracy: 0.7027 - val_loss: 0.6784 - val_accur
acy: 0.7187
Epoch 2/30
 - 57s - loss: 0.4947 - accuracy: 0.8184 - val_loss: 0.4339 - val_accur
acy: 0.8012
Epoch 3/30
 - 57s - loss: 0.4405 - accuracy: 0.8443 - val_loss: 0.8921 - val_accur
acy: 0.6644
Epoch 4/30
 - 57s - loss: 0.4687 - accuracy: 0.8341 - val_loss: 0.5267 - val_accur
acy: 0.8476
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 5/30

 - 58s - loss: 0.4293 - accuracy: 0.8421 - val_loss: 0.5572 - val_accur
acy: 0.8161

Epoch 6/30

 - 57s - loss: 0.4274 - accuracy: 0.8360 - val_loss: 0.5468 - val_accur
acy: 0.8426

Epoch 7/30

 - 58s - loss: 0.3509 - accuracy: 0.8668 - val_loss: 0.4680 - val_accur
acy: 0.8235

Epoch 8/30

 - 57s - loss: 0.3574 - accuracy: 0.8705 - val_loss: 0.3939 - val_accur
acy: 0.8466

Epoch 9/30

 - 58s - loss: 0.3256 - accuracy: 0.8788 - val_loss: 0.4278 - val_accur
acy: 0.8599

Epoch 10/30

 - 57s - loss: 0.3130 - accuracy: 0.8879 - val_loss: 0.4161 - val_accur
acy: 0.8575

Epoch 11/30

 - 58s - loss: 0.3037 - accuracy: 0.8908 - val_loss: 0.4565 - val_accur
acy: 0.8599

Epoch 12/30

 - 58s - loss: 0.3183 - accuracy: 0.8818 - val_loss: 0.4499 - val_accur
acy: 0.8571
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 13/30

 - 58s - loss: 0.3471 - accuracy: 0.8720 - val_loss: 0.6787 - val_accur
acy: 0.7462

Epoch 14/30

 - 57s - loss: 0.4000 - accuracy: 0.8549 - val_loss: 0.4826 - val_accur
acy: 0.8385

Epoch 15/30

 - 57s - loss: 0.2998 - accuracy: 0.8909 - val_loss: 0.4225 - val_accur
acy: 0.8633

Epoch 16/30

 - 58s - loss: 0.3467 - accuracy: 0.8774 - val_loss: 0.6066 - val_accur
acy: 0.7974

Epoch 17/30

 - 58s - loss: 0.3604 - accuracy: 0.8700 - val_loss: 0.5145 - val_accur
acy: 0.8130

Epoch 18/30

 - 58s - loss: 0.3785 - accuracy: 0.8589 - val_loss: 0.7070 - val_accur
acy: 0.7469

Epoch 19/30

 - 58s - loss: 0.4214 - accuracy: 0.8425 - val_loss: 0.5212 - val_accur
acy: 0.7927

Epoch 20/30

 - 58s - loss: 0.3962 - accuracy: 0.8517 - val_loss: 0.6194 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.7072

Epoch 21/30

 - 57s - loss: 0.3860 - accuracy: 0.8521 - val_loss: 0.7892 - val_accur
acy: 0.7302

Epoch 22/30

 - 58s - loss: 0.4001 - accuracy: 0.8459 - val_loss: 0.5473 - val_accur
acy: 0.7998

Epoch 23/30

 - 58s - loss: 0.3847 - accuracy: 0.8553 - val_loss: 0.5917 - val_accur
acy: 0.7767

Epoch 24/30

 - 58s - loss: 0.3670 - accuracy: 0.8592 - val_loss: 0.5966 - val_accur
acy: 0.7822

Epoch 25/30

 - 57s - loss: 0.3889 - accuracy: 0.8505 - val_loss: 0.4810 - val_accur
acy: 0.8008

Epoch 26/30

 - 58s - loss: 0.3605 - accuracy: 0.8651 - val_loss: 0.3598 - val_accur
acy: 0.8683

Epoch 27/30

 - 57s - loss: 0.3403 - accuracy: 0.8764 - val_loss: 0.4539 - val_accur
acy: 0.8215

Epoch 28/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 57s - loss: 0.3201 - accuracy: 0.8779 - val_loss: 0.4175 - val_accur
acy: 0.8517

Epoch 29/30

 - 58s - loss: 0.3744 - accuracy: 0.8614 - val_loss: 0.4513 - val_accur
acy: 0.8395

Epoch 30/30

 - 58s - loss: 0.4901 - accuracy: 0.8063 - val_loss: 0.5878 - val_accur
acy: 0.7625

Test accuracy:

0.7624703049659729

---------------------------------------------------------------------
--------------
Model: "sequential_12"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_34 (Dense)             (None, 128, 64)           640

_________________________________________________________________

batch_normalization_45 (Batc (None, 128, 64)           256

_________________________________________________________________

Dropout_1 (Dropout)          (None, 128, 64)           0

_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
dense_35 (Dense)             (None, 128, 32)           2080
_________________________________________________________________
batch_normalization_46 (Batc (None, 128, 32)           128
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 64)           24832
_________________________________________________________________
batch_normalization_47 (Batc (None, 128, 64)           256
_________________________________________________________________
Dropout_3 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 36)                14544
_________________________________________________________________
batch_normalization_48 (Batc (None, 36)                144
_________________________________________________________________
Dropout_4 (Dropout)          (None, 36)                0
_________________________________________________________________
dense_36 (Dense)             (None, 6)                 222
=================================================================
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Total params: 43,102

Trainable params: 42,710

Non-trainable params: 392

_________________________________________________________________

None

Train on 7352 samples, validate on 2947 samples

Epoch 1/30

 - 67s - loss: 0.7451 - accuracy: 0.7205 - val_loss: 0.4184 - val_accur
acy: 0.8493

Epoch 2/30

 - 63s - loss: 0.4351 - accuracy: 0.8549 - val_loss: 0.3566 - val_accur
acy: 0.8819

Epoch 3/30

 - 62s - loss: 0.3912 - accuracy: 0.8791 - val_loss: 0.4604 - val_accur
acy: 0.8639

Epoch 4/30

 - 63s - loss: 0.3690 - accuracy: 0.8796 - val_loss: 0.5604 - val_accur
acy: 0.8202

Epoch 5/30

 - 63s - loss: 0.3108 - accuracy: 0.8981 - val_loss: 0.5087 - val_accur
acy: 0.8500

Epoch 6/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 63s - loss: 0.3208 - accuracy: 0.9022 - val_loss: 0.4392 - val_accur
acy: 0.8687

Epoch 7/30

 - 62s - loss: 0.3134 - accuracy: 0.9011 - val_loss: 0.4743 - val_accur
acy: 0.8537

Epoch 8/30

 - 63s - loss: 0.3271 - accuracy: 0.8995 - val_loss: 0.4679 - val_accur
acy: 0.8792

Epoch 9/30

 - 63s - loss: 0.2940 - accuracy: 0.9082 - val_loss: 0.3513 - val_accur
acy: 0.8863

Epoch 10/30

 - 63s - loss: 0.2748 - accuracy: 0.9132 - val_loss: 0.3827 - val_accur
acy: 0.8901

Epoch 11/30

 - 62s - loss: 0.2662 - accuracy: 0.9124 - val_loss: 0.6272 - val_accur
acy: 0.8405

Epoch 12/30

 - 63s - loss: 0.2691 - accuracy: 0.9089 - val_loss: 0.4081 - val_accur
acy: 0.8918

Epoch 13/30

 - 63s - loss: 0.2592 - accuracy: 0.9150 - val_loss: 0.2973 - val_accur
acy: 0.8982
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 14/30

 - 63s - loss: 0.2838 - accuracy: 0.9042 - val_loss: 0.3667 - val_accur
acy: 0.8924

Epoch 15/30

 - 63s - loss: 0.2731 - accuracy: 0.9115 - val_loss: 0.3334 - val_accur
acy: 0.8724

Epoch 16/30

 - 63s - loss: 0.2780 - accuracy: 0.9149 - val_loss: 0.3022 - val_accur
acy: 0.8985

Epoch 17/30

 - 63s - loss: 0.3065 - accuracy: 0.9053 - val_loss: 0.3591 - val_accur
acy: 0.8738

Epoch 18/30

 - 63s - loss: 0.2949 - accuracy: 0.9042 - val_loss: 0.3367 - val_accur
acy: 0.8853

Epoch 19/30

 - 63s - loss: 0.2760 - accuracy: 0.9135 - val_loss: 0.3583 - val_accur
acy: 0.9013

Epoch 20/30

 - 63s - loss: 0.3252 - accuracy: 0.9037 - val_loss: 0.5924 - val_accur
acy: 0.8442

Epoch 21/30

 - 63s - loss: 0.3456 - accuracy: 0.8996 - val_loss: 0.3392 - val_accur
acy: 0.8958
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 22/30

 - 63s - loss: 0.3143 - accuracy: 0.9036 - val_loss: 0.3426 - val_accur
acy: 0.8989

Epoch 23/30

 - 62s - loss: 0.3293 - accuracy: 0.8893 - val_loss: 0.4898 - val_accur
acy: 0.8364

Epoch 24/30

 - 63s - loss: 0.3661 - accuracy: 0.8779 - val_loss: 0.5730 - val_accur
acy: 0.8341

Epoch 25/30

 - 63s - loss: 0.3411 - accuracy: 0.8936 - val_loss: 0.5018 - val_accur
acy: 0.8616

Epoch 26/30

 - 62s - loss: 0.3300 - accuracy: 0.8947 - val_loss: 0.4895 - val_accur
acy: 0.8612

Epoch 27/30

 - 63s - loss: 0.3368 - accuracy: 0.8883 - val_loss: 0.3707 - val_accur
acy: 0.8904

Epoch 28/30

 - 63s - loss: 0.3446 - accuracy: 0.8883 - val_loss: 0.3746 - val_accur
acy: 0.8802

Epoch 29/30

 - 63s - loss: 0.3676 - accuracy: 0.8804 - val_loss: 0.4281 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.8609

Epoch 30/30

 - 63s - loss: 0.3324 - accuracy: 0.8912 - val_loss: 0.5719 - val_accur
acy: 0.8432

Test accuracy:

0.8432304263114929

------------------------------------------------------------------------
--------------
Model: "sequential_13"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_37 (Dense)             (None, 128, 64)           640

_________________________________________________________________

batch_normalization_49 (Batc (None, 128, 64)           256

_________________________________________________________________

Dropout_1 (Dropout)          (None, 128, 64)           0

_________________________________________________________________

dense_38 (Dense)             (None, 128, 32)           2080

_________________________________________________________________

batch_normalization_50 (Batc (None, 128, 32)           128
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 64)           24832
_________________________________________________________________
batch_normalization_51 (Batc (None, 128, 64)           256
_________________________________________________________________
Dropout_3 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 64)                33024
_________________________________________________________________
batch_normalization_52 (Batc (None, 64)                256
_________________________________________________________________
Dropout_4 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_39 (Dense)             (None, 6)                 390
=================================================================
Total params: 61,862
Trainable params: 61,414
Non-trainable params: 448
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
None

Train on 7352 samples, validate on 2947 samples

Epoch 1/30

 - 74s - loss: 0.9963 - accuracy: 0.5764 - val_loss: 0.9131 - val_accur
acy: 0.6356

Epoch 2/30

 - 70s - loss: 0.8056 - accuracy: 0.6602 - val_loss: 1.2461 - val_accur
acy: 0.4510

Epoch 3/30

 - 71s - loss: 0.7999 - accuracy: 0.6439 - val_loss: 0.9980 - val_accur
acy: 0.5616

Epoch 4/30

 - 70s - loss: 0.7786 - accuracy: 0.6665 - val_loss: 0.7209 - val_accur
acy: 0.6861

Epoch 5/30

 - 70s - loss: 0.7200 - accuracy: 0.6995 - val_loss: 0.9675 - val_accur
acy: 0.5609

Epoch 6/30

 - 70s - loss: 0.7129 - accuracy: 0.6944 - val_loss: 0.7634 - val_accur
acy: 0.6461

Epoch 7/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 70s - loss: 0.7838 - accuracy: 0.6659 - val_loss: 0.8656 - val_accur
acy: 0.5653

Epoch 8/30

 - 70s - loss: 0.8164 - accuracy: 0.6302 - val_loss: 0.9462 - val_accur
acy: 0.6162

Epoch 9/30

 - 70s - loss: 0.8644 - accuracy: 0.6209 - val_loss: 1.0076 - val_accur
acy: 0.5120

Epoch 10/30

 - 71s - loss: 0.7904 - accuracy: 0.6401 - val_loss: 0.8300 - val_accur
acy: 0.6240

Epoch 11/30

 - 70s - loss: 0.7319 - accuracy: 0.6730 - val_loss: 0.8655 - val_accur
acy: 0.6101

Epoch 12/30

 - 70s - loss: 0.7552 - accuracy: 0.6741 - val_loss: 0.8467 - val_accur
acy: 0.5663

Epoch 13/30

 - 70s - loss: 0.7519 - accuracy: 0.6715 - val_loss: 0.6851 - val_accur
acy: 0.6892

Epoch 14/30

 - 70s - loss: 0.7833 - accuracy: 0.6688 - val_loss: 0.9484 - val_accur
acy: 0.5619

Epoch 15/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 70s - loss: 0.8131 - accuracy: 0.6556 - val_loss: 0.9215 - val_accur
acy: 0.5674

Epoch 16/30

 - 71s - loss: 0.7668 - accuracy: 0.6812 - val_loss: 0.8983 - val_accur
acy: 0.5243

Epoch 17/30

 - 70s - loss: 0.8104 - accuracy: 0.6455 - val_loss: 1.0380 - val_accur
acy: 0.4717

Epoch 18/30

 - 70s - loss: 0.8133 - accuracy: 0.6408 - val_loss: 1.1397 - val_accur
acy: 0.5137

Epoch 19/30

 - 70s - loss: 0.8182 - accuracy: 0.6394 - val_loss: 0.8518 - val_accur
acy: 0.5429

Epoch 20/30

 - 71s - loss: 0.7996 - accuracy: 0.6419 - val_loss: 1.1206 - val_accur
acy: 0.5056

Epoch 21/30

 - 70s - loss: 0.8506 - accuracy: 0.6304 - val_loss: 2.1719 - val_accur
acy: 0.3990

Epoch 22/30

 - 70s - loss: 0.8157 - accuracy: 0.6540 - val_loss: 1.2223 - val_accur
acy: 0.4781
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 23/30

 - 71s - loss: 0.8608 - accuracy: 0.6325 - val_loss: 1.3324 - val_accur
acy: 0.5456

Epoch 24/30

 - 70s - loss: 0.8959 - accuracy: 0.6162 - val_loss: 0.9533 - val_accur
acy: 0.5460

Epoch 25/30

 - 71s - loss: 0.9450 - accuracy: 0.5845 - val_loss: 1.6401 - val_accur
acy: 0.5137

Epoch 26/30

 - 71s - loss: 0.9697 - accuracy: 0.5830 - val_loss: 1.4275 - val_accur
acy: 0.5372

Epoch 27/30

 - 71s - loss: 0.9900 - accuracy: 0.5551 - val_loss: 1.5571 - val_accur
acy: 0.4913

Epoch 28/30

 - 71s - loss: 1.0081 - accuracy: 0.5468 - val_loss: 1.3802 - val_accur
acy: 0.4978

Epoch 29/30

 - 70s - loss: 0.9980 - accuracy: 0.5374 - val_loss: 1.4664 - val_accur
acy: 0.5616

Epoch 30/30

 - 70s - loss: 1.0440 - accuracy: 0.5248 - val_loss: 1.2778 - val_accur
acy: 0.4754
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Test accuracy:

0.4753987193107605

_________________________________________________________________
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_40 (Dense)             (None, 128, 64)           640
_________________________________________________________________
batch_normalization_53 (Batc (None, 128, 64)           256
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
dense_41 (Dense)             (None, 128, 32)           2080
_________________________________________________________________
batch_normalization_54 (Batc (None, 128, 32)           128
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
LSTM_1 (LSTM)                (None, 128, 64)          24832
_________________________________________________________________
batch_normalization_55 (Batc (None, 128, 64)          256
_________________________________________________________________
Dropout_3 (Dropout)          (None, 128, 64)          0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 64)               33024
_________________________________________________________________
batch_normalization_56 (Batc (None, 64)               256
_________________________________________________________________
Dropout_4 (Dropout)          (None, 64)               0
_________________________________________________________________
dense_42 (Dense)             (None, 6)                390
=================================================================
Total params: 61,862
Trainable params: 61,414
Non-trainable params: 448
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 1/30

 - 74s - loss: 0.9559 - accuracy: 0.5903 - val_loss: 0.8903 - val_accur
acy: 0.6800

Epoch 2/30

 - 71s - loss: 0.7734 - accuracy: 0.6816 - val_loss: 0.8306 - val_accur
acy: 0.7156

Epoch 3/30

 - 71s - loss: 0.7392 - accuracy: 0.6961 - val_loss: 0.9286 - val_accur
acy: 0.6960

Epoch 4/30

 - 71s - loss: 0.7461 - accuracy: 0.6863 - val_loss: 1.6276 - val_accur
acy: 0.5789

Epoch 5/30

 - 71s - loss: 0.8382 - accuracy: 0.6315 - val_loss: 2.7175 - val_accur
acy: 0.3047

Epoch 6/30

 - 71s - loss: 0.7859 - accuracy: 0.6498 - val_loss: 1.2262 - val_accur
acy: 0.4761

Epoch 7/30

 - 71s - loss: 0.8033 - accuracy: 0.6461 - val_loss: 2.2920 - val_accur
acy: 0.3268

Epoch 8/30

 - 71s - loss: 0.7761 - accuracy: 0.6670 - val_loss: 2.5132 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.3458

Epoch 9/30

 - 71s - loss: 0.7984 - accuracy: 0.6483 - val_loss: 1.5674 - val_accur
acy: 0.3790

Epoch 10/30

 - 70s - loss: 0.8166 - accuracy: 0.6347 - val_loss: 2.9600 - val_accur
acy: 0.3519

Epoch 11/30

 - 71s - loss: 0.7990 - accuracy: 0.6523 - val_loss: 4.3568 - val_accur
acy: 0.3101

Epoch 12/30

 - 70s - loss: 0.8133 - accuracy: 0.6492 - val_loss: 4.6617 - val_accur
acy: 0.2036

Epoch 13/30

 - 71s - loss: 0.7955 - accuracy: 0.6581 - val_loss: 5.4934 - val_accur
acy: 0.1917

Epoch 14/30

 - 71s - loss: 0.8528 - accuracy: 0.6368 - val_loss: 4.5168 - val_accur
acy: 0.2335

Epoch 15/30

 - 71s - loss: 0.8892 - accuracy: 0.5997 - val_loss: 3.9821 - val_accur
acy: 0.2776

Epoch 16/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 71s - loss: 0.8595 - accuracy: 0.6200 - val_loss: 4.5127 - val_accur
acy: 0.1856

Epoch 17/30

 - 70s - loss: 0.8834 - accuracy: 0.6151 - val_loss: 6.5314 - val_accur
acy: 0.2643

Epoch 18/30

 - 71s - loss: 0.8647 - accuracy: 0.6216 - val_loss: 2.8226 - val_accur
acy: 0.2399

Epoch 19/30

 - 71s - loss: 0.8603 - accuracy: 0.6234 - val_loss: 3.5535 - val_accur
acy: 0.2945

Epoch 20/30

 - 71s - loss: 0.8775 - accuracy: 0.6111 - val_loss: 3.7150 - val_accur
acy: 0.2365

Epoch 21/30

 - 71s - loss: 0.9202 - accuracy: 0.5857 - val_loss: 6.7342 - val_accur
acy: 0.1815

Epoch 22/30

 - 71s - loss: 0.8878 - accuracy: 0.5890 - val_loss: 5.0362 - val_accur
acy: 0.2050

Epoch 23/30

 - 71s - loss: 0.9175 - accuracy: 0.5989 - val_loss: 3.8274 - val_accur
acy: 0.2165

Epoch 24/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 71s - loss: 0.9045 - accuracy: 0.5862 - val_loss: 3.6244 - val_accur
acy: 0.1629

Epoch 25/30

 - 71s - loss: 0.9018 - accuracy: 0.6075 - val_loss: 4.0093 - val_accur
acy: 0.2603

Epoch 26/30

 - 71s - loss: 0.8811 - accuracy: 0.6122 - val_loss: 3.5012 - val_accur
acy: 0.2148

Epoch 27/30

 - 71s - loss: 0.8780 - accuracy: 0.5940 - val_loss: 4.2661 - val_accur
acy: 0.2572

Epoch 28/30

 - 72s - loss: 0.8473 - accuracy: 0.6178 - val_loss: 3.0702 - val_accur
acy: 0.2647

Epoch 29/30

 - 71s - loss: 0.8366 - accuracy: 0.6264 - val_loss: 2.9916 - val_accur
acy: 0.2382

Epoch 30/30

 - 72s - loss: 0.8366 - accuracy: 0.6204 - val_loss: 2.1270 - val_accur
acy: 0.3237

Test accuracy:

0.3237190246582031

----------------------------------------------------------------------
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
--------------
Model: "sequential_15"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_43 (Dense)             (None, 128, 64)           640
_________________________________________________________________
batch_normalization_57 (Batc (None, 128, 64)           256
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
dense_44 (Dense)             (None, 128, 32)           2080
_________________________________________________________________
batch_normalization_58 (Batc (None, 128, 32)           128
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 36)           9936
_________________________________________________________________
batch_normalization_59 (Batc (None, 128, 36)           144
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
Dropout_3 (Dropout)          (None, 128, 36)           0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 36)                10512
_________________________________________________________________
batch_normalization_60 (Batc (None, 36)                144
_________________________________________________________________
Dropout_4 (Dropout)          (None, 36)                0
_________________________________________________________________
dense_45 (Dense)             (None, 6)                 222
=================================================================
Total params: 24,062
Trainable params: 23,726
Non-trainable params: 336
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
 - 57s - loss: 0.9745 - accuracy: 0.5748 - val_loss: 0.6910 - val_accur
acy: 0.6797
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 2/30

 - 54s - loss: 0.7970 - accuracy: 0.6491 - val_loss: 0.7096 - val_accur
acy: 0.6600

Epoch 3/30

 - 54s - loss: 0.7541 - accuracy: 0.6825 - val_loss: 0.9208 - val_accur
acy: 0.6132

Epoch 4/30

 - 55s - loss: 0.7558 - accuracy: 0.6840 - val_loss: 0.6824 - val_accur
acy: 0.7194

Epoch 5/30

 - 54s - loss: 0.7968 - accuracy: 0.6734 - val_loss: 0.8453 - val_accur
acy: 0.6770

Epoch 6/30

 - 54s - loss: 0.8812 - accuracy: 0.6474 - val_loss: 0.7389 - val_accur
acy: 0.6647

Epoch 7/30

 - 54s - loss: 0.7761 - accuracy: 0.6846 - val_loss: 0.8335 - val_accur
acy: 0.6766

Epoch 8/30

 - 54s - loss: 0.7909 - accuracy: 0.6695 - val_loss: 0.8650 - val_accur
acy: 0.6430

Epoch 9/30

 - 54s - loss: 0.7605 - accuracy: 0.6827 - val_loss: 0.9149 - val_accur
acy: 0.6614
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 10/30

 - 54s - loss: 0.7605 - accuracy: 0.6556 - val_loss: 0.8475 - val_accur
acy: 0.6569

Epoch 11/30

 - 54s - loss: 0.7741 - accuracy: 0.6536 - val_loss: 0.7123 - val_accur
acy: 0.5908

Epoch 12/30

 - 54s - loss: 0.8666 - accuracy: 0.6352 - val_loss: 1.1838 - val_accur
acy: 0.4788

Epoch 13/30

 - 54s - loss: 0.7734 - accuracy: 0.6643 - val_loss: 0.9450 - val_accur
acy: 0.5870

Epoch 14/30

 - 54s - loss: 0.9564 - accuracy: 0.6089 - val_loss: 2.2123 - val_accur
acy: 0.3322

Epoch 15/30

 - 54s - loss: 0.8843 - accuracy: 0.6344 - val_loss: 1.2894 - val_accur
acy: 0.5596

Epoch 16/30

 - 54s - loss: 0.8472 - accuracy: 0.6348 - val_loss: 1.5540 - val_accur
acy: 0.3739

Epoch 17/30

 - 54s - loss: 0.8634 - accuracy: 0.6296 - val_loss: 1.1736 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.4883

Epoch 18/30

 - 55s - loss: 0.8116 - accuracy: 0.6532 - val_loss: 0.9810 - val_accur
acy: 0.5809

Epoch 19/30

 - 54s - loss: 0.8407 - accuracy: 0.6323 - val_loss: 1.0567 - val_accur
acy: 0.5976

Epoch 20/30

 - 54s - loss: 0.8650 - accuracy: 0.6268 - val_loss: 0.9297 - val_accur
acy: 0.5446

Epoch 21/30

 - 55s - loss: 0.8913 - accuracy: 0.6021 - val_loss: 1.3756 - val_accur
acy: 0.4622

Epoch 22/30

 - 54s - loss: 0.8959 - accuracy: 0.5955 - val_loss: 1.6889 - val_accur
acy: 0.3434

Epoch 23/30

 - 54s - loss: 0.8885 - accuracy: 0.6096 - val_loss: 1.5918 - val_accur
acy: 0.4177

Epoch 24/30

 - 54s - loss: 0.9692 - accuracy: 0.5724 - val_loss: 2.4136 - val_accur
acy: 0.3539

Epoch 25/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 54s - loss: 1.0486 - accuracy: 0.5354 - val_loss: 1.7627 - val_accur
acy: 0.3899

Epoch 26/30

 - 54s - loss: 1.1091 - accuracy: 0.5121 - val_loss: 3.6681 - val_accur
acy: 0.2542

Epoch 27/30

 - 54s - loss: 1.0943 - accuracy: 0.5152 - val_loss: 1.3642 - val_accur
acy: 0.3797

Epoch 28/30

 - 54s - loss: 1.1285 - accuracy: 0.4992 - val_loss: 2.2588 - val_accur
acy: 0.2640

Epoch 29/30

 - 54s - loss: 1.3204 - accuracy: 0.4430 - val_loss: 2.8197 - val_accur
acy: 0.1781

Epoch 30/30

 - 54s - loss: 1.3083 - accuracy: 0.4200 - val_loss: 2.2192 - val_accur
acy: 0.2993

Test accuracy:

0.2992874085903168

----------------------------------------------------------------------
--------------
100%|██████████| 15/15 [7:36:39<00:00, 1826.64s/trial, best loss: -0.91
75432920455933]


Time taken : 7:36:48.208962
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [38]:
```
print(best_run)
```

```
{'Dropout': 0.5758321559991686, 'Dropout_1': 0.2572526004449005, 'Dropo
ut_2': 0.8485318844943431, 'Dropout_3': 0.8447208322297787, 'LSTM': 32,
'LSTM_1': 64, 'choiceval': 'rmsprop', 'lr': 0.012575137121941487, 'lr_
1': 0.0015285232549964167}
```

In [41]:
```
M2_tuned_score = best_model.evaluate(X_test, Y_test, verbose=0)
print('Test Log Loss score:', M2_tuned_score[0])
print('Test accuracy:', M2_tuned_score[1])
```

```
Test Log Loss score: 0.39408119095370603
Test accuracy: 0.9175432920455933
```

In [42]:
```
#Plot a Non-Normalized confusion matrix
plot_lstm_confusion_matrix(best_model, normalize=False)

#Plot a Normalized confusion matrix
plot_lstm_confusion_matrix(best_model, normalize=True)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

7.1.3 Model 3

7.1.3.1 Constructing the Neural Network

In [38]:
```
model3 = Sequential()

model3.add(Dense(64, activation='relu', input_shape=(128,9),\
                     kernel_initializer='he_uniform'))
model3.add(BatchNormalization())
model3.add(Dropout(0.3, name='Dropout'))

model3.add(Dense(32, activation='relu', kernel_initializer='he_uniform'
))
model3.add(BatchNormalization())
model3.add(Dropout(0.3, name='Dropout_1'))

# Configuring the parameters
model3.add(LSTM(64,kernel_initializer='he_uniform', \
                return_sequences=True, input_shape=(128,9), name='LSTM_
1'))
model3.add(BatchNormalization())
model3.add(Dropout(0.3, name='Dropout_2'))

model3.add(LSTM(64,kernel_initializer='he_uniform', \
               name='LSTM_2'))
model3.add(BatchNormalization())
model3.add(Dropout(0.3, name='Dropout_3'))

model3.add(Dense(6, activation='softmax'))
model3.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 128, 64)           640
_________________________________________________________________
batch_normalization_4 (Batch (None, 128, 64)           256
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
Dropout (Dropout)            (None, 128, 64)           0
_________________________________________________________________
dense_7 (Dense)              (None, 128, 32)           2080
_________________________________________________________________
batch_normalization_5 (Batch (None, 128, 32)           128
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 64)           24832
_________________________________________________________________
batch_normalization_6 (Batch (None, 128, 64)           256
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 64)                33024
_________________________________________________________________
batch_normalization_7 (Batch (None, 64)                256
_________________________________________________________________
Dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_8 (Dense)              (None, 6)                 390
=================================================================
Total params: 61,862
Trainable params: 61,414
Non-trainable params: 448
_________________________________________________________________
```

7.1.3.2 Running the Neural Network on Train & Test Datasets for 30 Epochs

In [39]:

```
#Refer :- https://github.com/tensorflow/tensorflow/issues/27023

import tensorflow.python.util.deprecation as deprecation
deprecation._PRINT_DEPRECATION_WARNINGS = False

#Compiling the Model
model3.compile(optimizer='adam', loss='categorical_crossentropy', metri
cs=['accuracy'])
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
#Training the Model
M3_history = model3.fit(X_train, Y_train, batch_size=batch_size, epochs
=epochs, verbose=1, \
                        validation_data=(X_test, Y_test))
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 36s 5ms/sample - loss: 1.2
848 - acc: 0.5321 - val_loss: 1.3899 - val_acc: 0.4442
Epoch 2/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.7
567 - acc: 0.6850 - val_loss: 1.1042 - val_acc: 0.5368
Epoch 3/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.6
532 - acc: 0.7074 - val_loss: 0.8776 - val_acc: 0.6057
Epoch 4/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.5
794 - acc: 0.7469 - val_loss: 0.8413 - val_acc: 0.6339
Epoch 5/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.4
965 - acc: 0.7933 - val_loss: 0.7187 - val_acc: 0.6844
Epoch 6/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.3
901 - acc: 0.8443 - val_loss: 0.6953 - val_acc: 0.7316
Epoch 7/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.2
747 - acc: 0.8991 - val_loss: 0.5320 - val_acc: 0.8347
Epoch 8/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.2
302 - acc: 0.9140 - val_loss: 0.4583 - val_acc: 0.8582
Epoch 9/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.1
857 - acc: 0.9336 - val_loss: 0.5111 - val_acc: 0.8483
Epoch 10/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.1
709 - acc: 0.9343 - val_loss: 0.4433 - val_acc: 0.8792
Epoch 11/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.1
554 - acc: 0.9392 - val_loss: 0.4619 - val_acc: 0.8629
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 12/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.1
633 - acc: 0.9400 - val_loss: 0.4094 - val_acc: 0.8785
Epoch 13/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.1
535 - acc: 0.9392 - val_loss: 0.3945 - val_acc: 0.8812
Epoch 14/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.1
345 - acc: 0.9472 - val_loss: 0.3815 - val_acc: 0.8982
Epoch 15/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.1
246 - acc: 0.9518 - val_loss: 0.3634 - val_acc: 0.8989
Epoch 16/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.1
348 - acc: 0.9483 - val_loss: 0.3879 - val_acc: 0.9026
Epoch 17/30
7352/7352 [==============================] - 33s 5ms/sample - loss: 0.1
191 - acc: 0.9494 - val_loss: 0.3704 - val_acc: 0.9060
Epoch 18/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.1
199 - acc: 0.9506 - val_loss: 0.4096 - val_acc: 0.9094
Epoch 19/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.1
274 - acc: 0.9497 - val_loss: 0.4235 - val_acc: 0.8890
Epoch 20/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.1
107 - acc: 0.9531 - val_loss: 0.3511 - val_acc: 0.9131
Epoch 21/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.1
124 - acc: 0.9544 - val_loss: 0.4282 - val_acc: 0.9002
Epoch 22/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.1
056 - acc: 0.9532 - val_loss: 0.3826 - val_acc: 0.9138
Epoch 23/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.1
120 - acc: 0.9517 - val_loss: 0.4183 - val_acc: 0.8985
Epoch 24/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.1
094 - acc: 0.9517 - val_loss: 0.4314 - val_acc: 0.9013
Epoch 25/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.1
074 - acc: 0.9536 - val_loss: 0.3787 - val_acc: 0.9067
Epoch 26/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.1
125 - acc: 0.9543 - val_loss: 0.4353 - val_acc: 0.8867
Epoch 27/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.1
037 - acc: 0.9528 - val_loss: 0.4352 - val_acc: 0.9148
Epoch 28/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.0
997 - acc: 0.9574 - val_loss: 0.4069 - val_acc: 0.9135
Epoch 29/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.0
979 - acc: 0.9562 - val_loss: 0.3821 - val_acc: 0.9128
Epoch 30/30
7352/7352 [==============================] - 33s 4ms/sample - loss: 0.0
963 - acc: 0.9578 - val_loss: 0.3761 - val_acc: 0.9175
```

7.1.3.3 Number of Epochs vs Train Loss & Test Loss

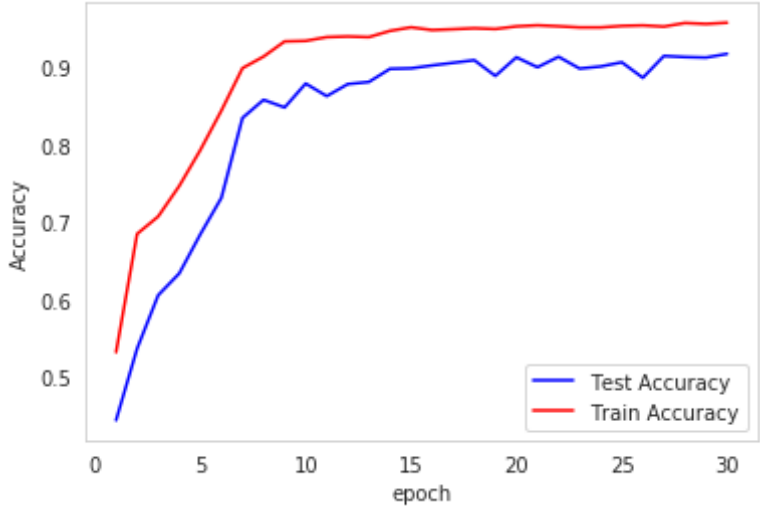
In [40]:

```
M3_score = model3.evaluate(X_test, Y_test, verbose=0)
print('Test Log Loss score:', M3_score[0])
print('Test accuracy:', M3_score[1])

fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Accuracy')

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

# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epo
chs=nb_epoch, verbose=1, validation_data=(X_test, Y_test))

# we will get val_loss and val_acc only when you pass the parameter val
idation_data
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
# for each key in history.history we will have a list of length equal t
o number of epochs

vy = M3_history.history['val_acc']
ty = M3_history.history['acc']
plt_dynamic(x, vy, ty, ax)
```

```
Test Log Loss score: 0.37614556847441283
Test accuracy: 0.9175433
```



7.1.3.4 Normalized and Non-Normalized Confusion Matrices

In [41]:
```
#Plot a Non-Normalized confusion matrix
plot_lstm_confusion_matrix(model3, normalize=False)

#Plot a Normalized confusion matrix
plot_lstm_confusion_matrix(model3, normalize=True)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

7.1.3.5 Hyperparameter Tuning

In [42]:
```
def model3(X_train, Y_train, X_test, Y_test):

    # Importing tensorflow
    from keras.models import Sequential
    from keras.layers import Dense, Dropout, LSTM, BatchNormalization
    from keras.callbacks import TensorBoard
    from keras.callbacks import ModelCheckpoint
    from keras.optimizers import RMSprop
    from keras.optimizers import adam

    model3 = Sequential()

    model3.add(Dense({{choice([64,128,256,512])}}, activation='relu', i
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
nput_shape=(128,9),\
                     kernel_initializer='he_uniform'))
    model3.add(BatchNormalization())
    model3.add(Dropout({{uniform(0.25,0.85)}}, name='Dropout'))

    model3.add(Dense({{choice([32,64,128,256])}}, activation='relu', ke
rnel_initializer='he_uniform'))
    model3.add(BatchNormalization())
    model3.add(Dropout({{uniform(0.25,0.85)}}, name='Dropout_1'))

    # Configuring the parameters
    model3.add(LSTM({{choice([32,64,128])}},kernel_initializer='he_unif
orm', \
                 return_sequences=True, input_shape=(128,9), name='LSTM_
1'))
    model3.add(BatchNormalization())
    model3.add(Dropout({{uniform(0.25,0.85)}}, name='Dropout_2'))

    model3.add(LSTM({{choice([16,32,64,128])}},kernel_initializer='he_u
niform', \
                name='LSTM_2'))
    model3.add(BatchNormalization())
    model3.add(Dropout({{uniform(0.25,0.85)}}, name='Dropout_3'))

    model3.add(Dense(6, activation='softmax'))

    adam = keras.optimizers.Adam(lr={{uniform(0.0009,0.025)}})
    rmsprop = keras.optimizers.RMSprop(lr={{uniform(0.0009,0.025)}})

    choiceval = {{choice(['adam', 'rmsprop'])}}

    if choiceval == 'adam':
        optim = adam
    else:
        optim = rmsprop

    print(model3.summary())

    model3.compile(loss='categorical_crossentropy', metrics=['accuracy'
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
],optimizer=optim)

    result = model3.fit(X_train, Y_train,
              batch_size=256,
              nb_epoch=30,
              verbose=2,
              validation_data=(X_test, Y_test))

    score, acc = model3.evaluate(X_test, Y_test, verbose=0)

    print('Test accuracy:', acc)
    print('-------------------------------------------------------------
-------------------------')
    return {'loss': -acc, 'status': STATUS_OK, 'model': model3}
```

In [42]:

```
start = datetime.now()

import keras
from keras.regularizers import l2
from tensorflow.python.keras import utils
from hyperas import optim
from hyperopt import Trials, STATUS_OK, tpe

X_train, Y_train, X_test, Y_test = data()
trials = Trials()
best_run, best_model, space = optim.minimize(model=model3,
                                        data=data,
                                        algo=tpe.suggest,
                                        max_evals=15,
                                        trials=trials,notebook_name = 'Hu
man_Activity_Recognition',
                                        return_space = True, eval_space=T
rue)

print(" "*100)
print('Time taken :', datetime.now() - start)
```

```
>>> Imports:
#coding=utf-8
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
try:
    import numpy as np
except:
    pass

try:
    import pandas as pd
except:
    pass

try:
    import warnings
except:
    pass

try:
    import matplotlib.pyplot as plt
except:
    pass

try:
    import seaborn as sns
except:
    pass

try:
    import numpy as np
except:
    pass

try:
    from sklearn.manifold import TSNE
except:
    pass

try:
    import matplotlib.pyplot as plt
except:
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    pass

try:
    import seaborn as sns
except:
    pass

try:
    import itertools
except:
    pass

try:
    import numpy as np
except:
    pass

try:
    import matplotlib.pyplot as plt
except:
    pass

try:
    from sklearn.metrics import confusion_matrix
except:
    pass

try:
    from datetime import datetime
except:
    pass

try:
    from sklearn import linear_model
except:
    pass

try:
    from sklearn import metrics
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
except:
    pass

try:
    from sklearn.model_selection import GridSearchCV
except:
    pass

try:
    from sklearn.svm import LinearSVC
except:
    pass

try:
    from sklearn.svm import SVC
except:
    pass

try:
    from sklearn.tree import DecisionTreeClassifier
except:
    pass

try:
    from sklearn.ensemble import RandomForestClassifier
except:
    pass

try:
    from sklearn.ensemble import GradientBoostingClassifier
except:
    pass

try:
    from tensorflow.python.keras.wrappers.scikit_learn import KerasClas
sifier
except:
    pass
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
try:
    from sklearn.model_selection import GridSearchCV
except:
    pass

try:
    from datetime import datetime
except:
    pass

try:
    from tensorflow.python.keras.layers.normalization import BatchNorma
lization
except:
    pass

try:
    from tensorflow.python.keras.layers import Dropout
except:
    pass

try:
    import tensorflow as tf
except:
    pass

try:
    from keras import backend as K
except:
    pass

try:
    from tensorflow.python.keras.models import Sequential
except:
    pass

try:
    from tensorflow.python.keras.layers import LSTM
except:
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    pass

try:
    from tensorflow.python.keras.layers.core import Dense, Dropout
except:
    pass

try:
    import tensorflow.python.util.deprecation as deprecation
except:
    pass

try:
    import tensorflow.python.util.deprecation as deprecation
except:
    pass

try:
    import tensorflow.python.util.deprecation as deprecation
except:
    pass

try:
    import tensorflow.python.util.deprecation as deprecation
except:
    pass

try:
    import tensorflow.python.util.deprecation as deprecation
except:
    pass

try:
    import tensorflow.python.util.deprecation as deprecation
except:
    pass

try:
    import tensorflow.python.util.deprecation as deprecation
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
except:
    pass

try:
    from keras.models import Sequential
except:
    pass

try:
    from keras.layers import Dense, Dropout, LSTM, BatchNormalization
except:
    pass

try:
    from keras.callbacks import TensorBoard
except:
    pass

try:
    from keras.callbacks import ModelCheckpoint
except:
    pass

try:
    from keras.optimizers import RMSprop
except:
    pass

try:
    from keras.optimizers import adam
except:
    pass

try:
    import tensorflow as tf
except:
    pass

try:
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    import keras
except:
    pass

try:
    from keras.regularizers import l2
except:
    pass

try:
    from tensorflow.python.keras import utils
except:
    pass

try:
    from hyperas import optim
except:
    pass

try:
    from hyperopt import Trials, STATUS_OK, tpe
except:
    pass

try:
    import tensorflow.python.util.deprecation as deprecation
except:
    pass

try:
    from keras.models import Sequential
except:
    pass

try:
    from keras.layers import Dense, Dropout, LSTM, BatchNormalization
except:
    pass
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
try:
    from keras.callbacks import TensorBoard
except:
    pass

try:
    from keras.callbacks import ModelCheckpoint
except:
    pass

try:
    from keras.optimizers import RMSprop
except:
    pass

try:
    from keras.optimizers import adam
except:
    pass

try:
    import keras
except:
    pass

try:
    from keras.regularizers import l2
except:
    pass

try:
    from tensorflow.python.keras import utils
except:
    pass

try:
    from hyperas import optim
except:
    pass
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
try:
    from hyperopt import Trials, STATUS_OK, tpe
except:
    pass

>>> Hyperas search space:

def get_space():
    return {
        'Dense': hp.choice('Dense', [64,128,256,512]),
        'Dropout': hp.uniform('Dropout', 0.25,0.85),
        'Dense_1': hp.choice('Dense_1', [32,64,128,256]),
        'Dropout_1': hp.uniform('Dropout_1', 0.25,0.85),
        'LSTM': hp.choice('LSTM', [32,64,128]),
        'Dropout_2': hp.uniform('Dropout_2', 0.25,0.85),
        'LSTM_1': hp.choice('LSTM_1', [16,32,64,128]),
        'Dropout_3': hp.uniform('Dropout_3', 0.25,0.85),
        'lr': hp.uniform('lr', 0.0009,0.025),
        'lr_1': hp.uniform('lr_1', 0.0009,0.025),
        'choiceval': hp.choice('choiceval', ['adam', 'rmsprop']),
    }

>>> Data
   1:
   2: """
   3: Obtain the dataset from multiple files.
   4: Returns: X_train, X_test, y_train, y_test
   5: """
   6: # Data directory
   7: DATADIR = 'UCI_HAR_Dataset'
   8: SIGNALS = [
   9:     "body_acc_x",
  10:     "body_acc_y",
  11:     "body_acc_z",
  12:     "body_gyro_x",
  13:     "body_gyro_y",
  14:     "body_gyro_z",
  15:     "total_acc_x",
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    "total_acc_y",
    "total_acc_z"
    ]
# Utility function to read the data from csv file
def _read_csv(filename):
    return pd.read_csv(filename, delim_whitespace=True, header=No
ne)

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

    for signal in SIGNALS:
        filename = f'HAR/UCI_HAR_Dataset/{subset}/Inertial Signal
s/{signal}_{subset}.txt'
        signals_data.append( _read_csv(filename).to_numpy())

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

def load_y(subset):
    """
    The objective that we are trying to predict is a integer, fro
m 1 to 6,
    that represents a human activity. We return a binary represen
tation of
    every sample objective as a 6 bits vector using One Hot Encod
ing
    (https://pandas.pydata.org/pandas-docs/stable/generated/panda
s.get_dummies.html)
    """
    filename = f'HAR/UCI_HAR_Dataset/{subset}/y_{subset}.txt'
    y = _read_csv(filename)[0]
    return pd.get_dummies(y).to_numpy()

```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
  47: X_train, X_test = load_signals('train'), load_signals('test')
  48: Y_train, Y_test = load_y('train'), load_y('test')
  49:
  50:
  51:
  52:
>>> Resulting replaced keras model:

   1: def keras_fmin_fnct(space):
   2:
   3:
   4:     # Importing tensorflow
   5:
   6:     model8 = Sequential()
   7:
   8:     model8.add(Dense(space['Dense'], activation='relu', input_sha
pe=(128,9),\
   9:                      kernel_initializer='he_uniform'))
  10:     model8.add(BatchNormalization())
  11:     model8.add(Dropout(space['Dropout'], name='Dropout'))
  12:
  13:     model8.add(Dense(space['Dense_1'], activation='relu', kernel_
initializer='he_uniform'))
  14:     model8.add(BatchNormalization())
  15:     model8.add(Dropout(space['Dropout_1'], name='Dropout_1'))
  16:
  17:     # Configuring the parameters
  18:     model8.add(LSTM(space['LSTM'],kernel_initializer='he_unifor
m', \
  19:                 return_sequences=True, input_shape=(128,9), name
='LSTM_1'))
  20:     model8.add(BatchNormalization())
  21:     model8.add(Dropout(space['Dropout_2'], name='Dropout_2'))
  22:
  23:     model8.add(LSTM(space['LSTM_1'],kernel_initializer='he_unifor
m', \
  24:                 name='LSTM_2'))
  25:     model8.add(BatchNormalization())
  26:     model8.add(Dropout(space['Dropout_3'], name='Dropout_3'))
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
  27:
  28:     model8.add(Dense(6, activation='softmax'))
  29:
  30:     adam = keras.optimizers.Adam(lr=space['lr'])
  31:     rmsprop = keras.optimizers.RMSprop(lr=space['lr_1'])
  32:
  33:     choiceval = space['choiceval']
  34:
  35:     if choiceval == 'adam':
  36:         optim = adam
  37:     else:
  38:         optim = rmsprop
  39:
  40:     print(model8.summary())
  41:
  42:     model8.compile(loss='categorical_crossentropy', metrics=['acc
uracy'],optimizer=optim)
  43:
  44:     result = model8.fit(X_train, Y_train,
  45:               batch_size=256,
  46:               nb_epoch=30,
  47:               verbose=2,
  48:               validation_data=(X_test, Y_test))
  49:
  50:     score, acc = model8.evaluate(X_test, Y_test, verbose=0)
  51:
  52:     print('Test accuracy:', acc)
  53:     print('-------------------------------------------------------
--------------------------------')
  54:     return {'loss': -acc, 'status': STATUS_OK, 'model': model8}
  55:
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_9 (Dense)              (None, 128, 64)           640
_________________________________________________________________
batch_normalization_12 (Batc (None, 128, 64)           256
_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Dropout (Dropout)            (None, 128, 64)          0
_________________________________________________________________
dense_10 (Dense)             (None, 128, 128)         8320
_________________________________________________________________
batch_normalization_13 (Batc (None, 128, 128)         512
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 128)         0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 64)          49408
_________________________________________________________________
batch_normalization_14 (Batc (None, 128, 64)          256
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 64)          0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 32)               12416
_________________________________________________________________
batch_normalization_15 (Batc (None, 32)               128
_________________________________________________________________
Dropout_3 (Dropout)          (None, 32)               0
_________________________________________________________________
dense_11 (Dense)             (None, 6)                198
=================================================================
Total params: 72,134
Trainable params: 71,558
Non-trainable params: 576
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
 - 42s - loss: 0.7489 - accuracy: 0.7057 - val_loss: 2.3281 - val_accur
acy: 0.3641

Epoch 2/30
 - 37s - loss: 0.3071 - accuracy: 0.8902 - val_loss: 1.2014 - val_accur
acy: 0.7509

Epoch 3/30
 - 37s - loss: 0.2580 - accuracy: 0.9081 - val_loss: 0.5202 - val_accur
acy: 0.8235
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 4/30
 - 37s - loss: 0.1876 - accuracy: 0.9324 - val_loss: 1.0196 - val_accur
acy: 0.7414

Epoch 5/30
 - 36s - loss: 0.1824 - accuracy: 0.9304 - val_loss: 0.4071 - val_accur
acy: 0.8873

Epoch 6/30
 - 37s - loss: 0.1608 - accuracy: 0.9411 - val_loss: 0.6106 - val_accur
acy: 0.8297

Epoch 7/30
 - 38s - loss: 0.1614 - accuracy: 0.9348 - val_loss: 0.4004 - val_accur
acy: 0.8806

Epoch 8/30
 - 36s - loss: 0.1377 - accuracy: 0.9431 - val_loss: 0.3686 - val_accur
acy: 0.8928

Epoch 9/30
 - 36s - loss: 0.1487 - accuracy: 0.9357 - val_loss: 0.5298 - val_accur
acy: 0.8955

Epoch 10/30
 - 36s - loss: 0.1523 - accuracy: 0.9449 - val_loss: 0.6036 - val_accur
acy: 0.8402

Epoch 11/30
 - 36s - loss: 0.1475 - accuracy: 0.9404 - val_loss: 0.4367 - val_accur
acy: 0.8890

Epoch 12/30
 - 36s - loss: 0.1416 - accuracy: 0.9421 - val_loss: 1.3223 - val_accur
acy: 0.7262

Epoch 13/30
 - 36s - loss: 0.1678 - accuracy: 0.9389 - val_loss: 0.4114 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.8873

Epoch 14/30
 - 37s - loss: 0.1323 - accuracy: 0.9465 - val_loss: 0.3447 - val_accur
acy: 0.9138

Epoch 15/30
 - 36s - loss: 0.1283 - accuracy: 0.9453 - val_loss: 0.6144 - val_accur
acy: 0.8690

Epoch 16/30
 - 36s - loss: 0.1186 - accuracy: 0.9474 - val_loss: 0.7945 - val_accur
acy: 0.8453

Epoch 17/30
 - 36s - loss: 0.1413 - accuracy: 0.9468 - val_loss: 0.5547 - val_accur
acy: 0.8856

Epoch 18/30
 - 36s - loss: 0.1282 - accuracy: 0.9495 - val_loss: 0.4846 - val_accur
acy: 0.8992

Epoch 19/30
 - 38s - loss: 0.1311 - accuracy: 0.9486 - val_loss: 0.3743 - val_accur
acy: 0.9063

Epoch 20/30
 - 36s - loss: 0.1153 - accuracy: 0.9504 - val_loss: 0.3972 - val_accur
acy: 0.9006

Epoch 21/30
 - 36s - loss: 0.1396 - accuracy: 0.9467 - val_loss: 0.7111 - val_accur
acy: 0.8571

Epoch 22/30
 - 36s - loss: 0.1320 - accuracy: 0.9475 - val_loss: 0.3455 - val_accur
acy: 0.9040

Epoch 23/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 36s - loss: 0.1089 - accuracy: 0.9502 - val_loss: 0.4100 - val_accur
acy: 0.9070

Epoch 24/30
 - 36s - loss: 0.1294 - accuracy: 0.9470 - val_loss: 0.5173 - val_accur
acy: 0.8850

Epoch 25/30
 - 36s - loss: 0.1176 - accuracy: 0.9520 - val_loss: 0.2673 - val_accur
acy: 0.9145

Epoch 26/30
 - 36s - loss: 0.1501 - accuracy: 0.9499 - val_loss: 0.2618 - val_accur
acy: 0.9199

Epoch 27/30
 - 36s - loss: 0.1297 - accuracy: 0.9510 - val_loss: 0.3432 - val_accur
acy: 0.8948

Epoch 28/30
 - 36s - loss: 0.1162 - accuracy: 0.9505 - val_loss: 0.4709 - val_accur
acy: 0.8999

Epoch 29/30
 - 36s - loss: 0.1170 - accuracy: 0.9512 - val_loss: 0.4578 - val_accur
acy: 0.9019

Epoch 30/30
 - 36s - loss: 0.1210 - accuracy: 0.9509 - val_loss: 0.4279 - val_accur
acy: 0.8951

Test accuracy:
0.8951476216316223
---------------------------------------------------------------------
--------------
Model: "sequential_5"

_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_12 (Dense)             (None, 128, 128)          1280
_________________________________________________________________
batch_normalization_16 (Batc (None, 128, 128)          512
_________________________________________________________________
Dropout (Dropout)            (None, 128, 128)          0
_________________________________________________________________
dense_13 (Dense)             (None, 128, 32)           4128
_________________________________________________________________
batch_normalization_17 (Batc (None, 128, 32)           128
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 128)          82432
_________________________________________________________________
batch_normalization_18 (Batc (None, 128, 128)          512
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
LSTM_2 (LSTM)                (None, 32)                20608
_________________________________________________________________
batch_normalization_19 (Batc (None, 32)                128
_________________________________________________________________
Dropout_3 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_14 (Dense)             (None, 6)                 198
=================================================================
Total params: 109,926
Trainable params: 109,286
Non-trainable params: 640
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
 - 51s - loss: 1.0223 - accuracy: 0.5871 - val_loss: 1.6722 - val_accur
acy: 0.4116
Epoch 2/30
 - 49s - loss: 0.9990 - accuracy: 0.5558 - val_loss: 1.5150 - val_accur
acy: 0.4713
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 3/30

 - 48s - loss: 0.9780 - accuracy: 0.5222 - val_loss: 2.4562 - val_accur
acy: 0.3186

Epoch 4/30

 - 48s - loss: 0.8532 - accuracy: 0.5687 - val_loss: 0.8849 - val_accur
acy: 0.5029

Epoch 5/30

 - 52s - loss: 0.7973 - accuracy: 0.5982 - val_loss: 0.9462 - val_accur
acy: 0.5528

Epoch 6/30

 - 48s - loss: 0.7128 - accuracy: 0.6362 - val_loss: 0.7919 - val_accur
acy: 0.5999

Epoch 7/30

 - 48s - loss: 0.7120 - accuracy: 0.6427 - val_loss: 0.7076 - val_accur
acy: 0.6386

Epoch 8/30

 - 48s - loss: 0.6678 - accuracy: 0.6764 - val_loss: 0.7715 - val_accur
acy: 0.6271

Epoch 9/30

 - 48s - loss: 0.6129 - accuracy: 0.7135 - val_loss: 0.7165 - val_accur
acy: 0.6529

Epoch 10/30

 - 48s - loss: 0.6128 - accuracy: 0.7208 - val_loss: 0.9313 - val_accur
acy: 0.6206
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 11/30

 - 48s - loss: 0.7172 - accuracy: 0.6725 - val_loss: 0.8059 - val_accur
acy: 0.6787

Epoch 12/30

 - 49s - loss: 0.5496 - accuracy: 0.7573 - val_loss: 0.5952 - val_accur
acy: 0.7441

Epoch 13/30

 - 48s - loss: 0.5602 - accuracy: 0.7541 - val_loss: 0.5382 - val_accur
acy: 0.7730

Epoch 14/30

 - 48s - loss: 0.4837 - accuracy: 0.7726 - val_loss: 0.5866 - val_accur
acy: 0.7509

Epoch 15/30

 - 48s - loss: 0.4632 - accuracy: 0.7924 - val_loss: 0.6195 - val_accur
acy: 0.7733

Epoch 16/30

 - 49s - loss: 0.5013 - accuracy: 0.7707 - val_loss: 0.7915 - val_accur
acy: 0.7241

Epoch 17/30

 - 49s - loss: 0.5525 - accuracy: 0.7515 - val_loss: 0.7001 - val_accur
acy: 0.7272

Epoch 18/30

 - 48s - loss: 0.4670 - accuracy: 0.7775 - val_loss: 0.6714 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.7462

Epoch 19/30

 - 48s - loss: 0.4881 - accuracy: 0.7730 - val_loss: 0.7356 - val_accur
acy: 0.6919

Epoch 20/30

 - 49s - loss: 0.4654 - accuracy: 0.7735 - val_loss: 0.6283 - val_accur
acy: 0.7509

Epoch 21/30

 - 48s - loss: 0.4583 - accuracy: 0.7860 - val_loss: 0.6109 - val_accur
acy: 0.7604

Epoch 22/30

 - 48s - loss: 0.4553 - accuracy: 0.7822 - val_loss: 0.5263 - val_accur
acy: 0.7424

Epoch 23/30

 - 48s - loss: 0.4498 - accuracy: 0.7780 - val_loss: 0.5143 - val_accur
acy: 0.7693

Epoch 24/30

 - 48s - loss: 0.4517 - accuracy: 0.7801 - val_loss: 0.5168 - val_accur
acy: 0.7319

Epoch 25/30

 - 49s - loss: 0.5977 - accuracy: 0.7344 - val_loss: 1.4526 - val_accur
acy: 0.6423

Epoch 26/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 49s - loss: 0.6305 - accuracy: 0.7300 - val_loss: 0.4629 - val_accur
acy: 0.7832

Epoch 27/30

 - 48s - loss: 0.4811 - accuracy: 0.7738 - val_loss: 0.4576 - val_accur
acy: 0.7564

Epoch 28/30

 - 50s - loss: 0.4843 - accuracy: 0.7592 - val_loss: 0.4854 - val_accur
acy: 0.7306

Epoch 29/30

 - 49s - loss: 0.4865 - accuracy: 0.7780 - val_loss: 1.3366 - val_accur
acy: 0.5976

Epoch 30/30

 - 49s - loss: 0.6695 - accuracy: 0.7093 - val_loss: 0.6073 - val_accur
acy: 0.7248

Test accuracy:

0.7248048782348633

---------------------------------------------------------------------
--------------
Model: "sequential_6"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_15 (Dense)             (None, 128, 128)          1280
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
batch_normalization_20 (Batc (None, 128, 128)          512
_________________________________________________________________
Dropout (Dropout)            (None, 128, 128)          0
_________________________________________________________________
dense_16 (Dense)             (None, 128, 256)          33024
_________________________________________________________________
batch_normalization_21 (Batc (None, 128, 256)          1024
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 256)          0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 128)          197120
_________________________________________________________________
batch_normalization_22 (Batc (None, 128, 128)          512
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 32)                20608
_________________________________________________________________
batch_normalization_23 (Batc (None, 32)                128
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
Dropout_3 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_17 (Dense)             (None, 6)                 198
=================================================================
Total params: 254,406
Trainable params: 253,318
Non-trainable params: 1,088
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
 - 88s - loss: 0.9805 - accuracy: 0.6027 - val_loss: 2.1755 - val_accur
acy: 0.2433
Epoch 2/30
 - 82s - loss: 0.6614 - accuracy: 0.7145 - val_loss: 1.5252 - val_accur
acy: 0.5721
Epoch 3/30
 - 80s - loss: 0.4716 - accuracy: 0.8218 - val_loss: 0.7031 - val_accur
acy: 0.7262
Epoch 4/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 85s - loss: 0.3356 - accuracy: 0.8785 - val_loss: 0.9069 - val_accur
acy: 0.7479

Epoch 5/30

 - 86s - loss: 0.2932 - accuracy: 0.9064 - val_loss: 0.6973 - val_accur
acy: 0.8022

Epoch 6/30

 - 86s - loss: 0.2350 - accuracy: 0.9185 - val_loss: 1.2445 - val_accur
acy: 0.7258

Epoch 7/30

 - 86s - loss: 0.2182 - accuracy: 0.9272 - val_loss: 0.5125 - val_accur
acy: 0.8666

Epoch 8/30

 - 86s - loss: 0.2323 - accuracy: 0.9261 - val_loss: 0.4841 - val_accur
acy: 0.8605

Epoch 9/30

 - 83s - loss: 0.1969 - accuracy: 0.9348 - val_loss: 0.5846 - val_accur
acy: 0.8660

Epoch 10/30

 - 87s - loss: 0.2077 - accuracy: 0.9241 - val_loss: 0.4251 - val_accur
acy: 0.8924

Epoch 11/30

 - 94s - loss: 0.1828 - accuracy: 0.9380 - val_loss: 0.5047 - val_accur
acy: 0.8901
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 12/30

 - 90s - loss: 0.1841 - accuracy: 0.9359 - val_loss: 0.5326 - val_accur
acy: 0.8666

Epoch 13/30

 - 89s - loss: 0.1705 - accuracy: 0.9414 - val_loss: 0.6645 - val_accur
acy: 0.8728

Epoch 14/30

 - 86s - loss: 0.1695 - accuracy: 0.9382 - val_loss: 0.5129 - val_accur
acy: 0.8999

Epoch 15/30

 - 85s - loss: 0.1564 - accuracy: 0.9431 - val_loss: 0.5912 - val_accur
acy: 0.8833

Epoch 16/30

 - 81s - loss: 0.1470 - accuracy: 0.9440 - val_loss: 0.5170 - val_accur
acy: 0.8863

Epoch 17/30

 - 88s - loss: 0.1594 - accuracy: 0.9395 - val_loss: 0.4761 - val_accur
acy: 0.8904

Epoch 18/30

 - 84s - loss: 0.1596 - accuracy: 0.9393 - val_loss: 0.4667 - val_accur
acy: 0.8561

Epoch 19/30

 - 89s - loss: 0.1596 - accuracy: 0.9377 - val_loss: 0.7794 - val_accur
acy: 0.8544
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 20/30

 - 80s - loss: 0.1493 - accuracy: 0.9441 - val_loss: 0.5132 - val_accur
acy: 0.8873

Epoch 21/30

 - 79s - loss: 0.1528 - accuracy: 0.9453 - val_loss: 0.3489 - val_accur
acy: 0.9125

Epoch 22/30

 - 80s - loss: 0.1341 - accuracy: 0.9446 - val_loss: 0.5911 - val_accur
acy: 0.8924

Epoch 23/30

 - 83s - loss: 0.1484 - accuracy: 0.9402 - val_loss: 0.3404 - val_accur
acy: 0.9033

Epoch 24/30

 - 79s - loss: 0.1314 - accuracy: 0.9465 - val_loss: 0.4017 - val_accur
acy: 0.9155

Epoch 25/30

 - 79s - loss: 0.1356 - accuracy: 0.9444 - val_loss: 0.5213 - val_accur
acy: 0.8965

Epoch 26/30

 - 79s - loss: 0.1534 - accuracy: 0.9448 - val_loss: 0.6571 - val_accur
acy: 0.8989

Epoch 27/30

 - 79s - loss: 0.1406 - accuracy: 0.9471 - val_loss: 0.5896 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.8853

Epoch 28/30

 - 79s - loss: 0.1561 - accuracy: 0.9414 - val_loss: 0.6932 - val_accur
acy: 0.8951

Epoch 29/30

 - 79s - loss: 0.1406 - accuracy: 0.9467 - val_loss: 0.6303 - val_accur
acy: 0.8982

Epoch 30/30

 - 79s - loss: 0.1368 - accuracy: 0.9476 - val_loss: 0.5663 - val_accur
acy: 0.9091

Test accuracy:

0.9090600609779358

---------------------------------------------------------------------
--------------
Model: "sequential_7"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_18 (Dense)             (None, 128, 512)          5120

_________________________________________________________________

batch_normalization_24 (Batc (None, 128, 512)          2048

_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Dropout (Dropout)            (None, 128, 512)          0
_________________________________________________________________
dense_19 (Dense)             (None, 128, 64)           32832
_________________________________________________________________
batch_normalization_25 (Batc (None, 128, 64)           256
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 32)           12416
_________________________________________________________________
batch_normalization_26 (Batc (None, 128, 32)           128
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 64)                24832
_________________________________________________________________
batch_normalization_27 (Batc (None, 64)                256
_________________________________________________________________
Dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
dense_20 (Dense)             (None, 6)                 390
=================================================================
Total params: 78,278
Trainable params: 76,934
Non-trainable params: 1,344
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
 - 53s - loss: 3.3044 - accuracy: 0.2390 - val_loss: 1.2293 - val_accur
acy: 0.4988
Epoch 2/30
 - 51s - loss: 2.3445 - accuracy: 0.3304 - val_loss: 1.3581 - val_accur
acy: 0.4791
Epoch 3/30
 - 51s - loss: 1.7289 - accuracy: 0.4263 - val_loss: 1.3508 - val_accur
acy: 0.4937
Epoch 4/30
 - 51s - loss: 1.3198 - accuracy: 0.5001 - val_loss: 1.1408 - val_accur
acy: 0.5501
Epoch 5/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 52s - loss: 1.0186 - accuracy: 0.5645 - val_loss: 0.9809 - val_accur
acy: 0.5803

Epoch 6/30

 - 51s - loss: 0.8610 - accuracy: 0.6030 - val_loss: 0.8928 - val_accur
acy: 0.6203

Epoch 7/30

 - 50s - loss: 0.7902 - accuracy: 0.6219 - val_loss: 0.8611 - val_accur
acy: 0.5986

Epoch 8/30

 - 49s - loss: 0.7370 - accuracy: 0.6411 - val_loss: 0.7396 - val_accur
acy: 0.6189

Epoch 9/30

 - 49s - loss: 0.7226 - accuracy: 0.6495 - val_loss: 1.4459 - val_accur
acy: 0.5874

Epoch 10/30

 - 49s - loss: 0.7077 - accuracy: 0.6525 - val_loss: 0.8040 - val_accur
acy: 0.6206

Epoch 11/30

 - 53s - loss: 0.6913 - accuracy: 0.6568 - val_loss: 0.7214 - val_accur
acy: 0.6278

Epoch 12/30

 - 55s - loss: 0.6812 - accuracy: 0.6730 - val_loss: 0.9199 - val_accur
acy: 0.6186

Epoch 13/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 61s - loss: 0.6855 - accuracy: 0.6594 - val_loss: 1.0945 - val_accur
acy: 0.6274

Epoch 14/30

 - 63s - loss: 0.6812 - accuracy: 0.6676 - val_loss: 1.1320 - val_accur
acy: 0.6393

Epoch 15/30

 - 56s - loss: 0.6787 - accuracy: 0.6657 - val_loss: 0.7245 - val_accur
acy: 0.6932

Epoch 16/30

 - 58s - loss: 0.6481 - accuracy: 0.6808 - val_loss: 0.6895 - val_accur
acy: 0.7272

Epoch 17/30

 - 57s - loss: 0.6237 - accuracy: 0.7001 - val_loss: 0.6846 - val_accur
acy: 0.7238

Epoch 18/30

 - 56s - loss: 0.5874 - accuracy: 0.7246 - val_loss: 0.6408 - val_accur
acy: 0.7119

Epoch 19/30

 - 64s - loss: 0.5670 - accuracy: 0.7456 - val_loss: 0.6476 - val_accur
acy: 0.7265

Epoch 20/30

 - 59s - loss: 0.5261 - accuracy: 0.7599 - val_loss: 0.6141 - val_accur
acy: 0.7469
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 21/30

 - 60s - loss: 0.4927 - accuracy: 0.7767 - val_loss: 0.6724 - val_accur
acy: 0.7472

Epoch 22/30

 - 65s - loss: 0.4765 - accuracy: 0.7772 - val_loss: 0.7496 - val_accur
acy: 0.7360

Epoch 23/30

 - 62s - loss: 0.4589 - accuracy: 0.7879 - val_loss: 0.6601 - val_accur
acy: 0.7536

Epoch 24/30

 - 61s - loss: 0.4588 - accuracy: 0.7866 - val_loss: 0.6416 - val_accur
acy: 0.7587

Epoch 25/30

 - 62s - loss: 0.4482 - accuracy: 0.7899 - val_loss: 0.6335 - val_accur
acy: 0.7604

Epoch 26/30

 - 57s - loss: 0.4403 - accuracy: 0.7881 - val_loss: 0.6291 - val_accur
acy: 0.7594

Epoch 27/30

 - 64s - loss: 0.4100 - accuracy: 0.7961 - val_loss: 0.6318 - val_accur
acy: 0.7615

Epoch 28/30

 - 55s - loss: 0.4241 - accuracy: 0.7915 - val_loss: 0.6505 - val_accur
acy: 0.7584
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 29/30

 - 61s - loss: 0.4034 - accuracy: 0.8032 - val_loss: 0.6880 - val_accur
acy: 0.7574

Epoch 30/30

 - 58s - loss: 0.4088 - accuracy: 0.7934 - val_loss: 0.6671 - val_accur
acy: 0.7557

Test accuracy:

0.7556837201118469

---------------------------------------------------------------------
--------------
Model: "sequential_8"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_21 (Dense)             (None, 128, 256)          2560

_________________________________________________________________

batch_normalization_28 (Batc (None, 128, 256)          1024

_________________________________________________________________

Dropout (Dropout)            (None, 128, 256)          0

_________________________________________________________________

dense_22 (Dense)             (None, 128, 64)           16448
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
batch_normalization_29 (Batc (None, 128, 64)          256
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 64)          0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 128)         98816
_________________________________________________________________
batch_normalization_30 (Batc (None, 128, 128)         512
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 128)         0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 128)              131584
_________________________________________________________________
batch_normalization_31 (Batc (None, 128)              512
_________________________________________________________________
Dropout_3 (Dropout)          (None, 128)              0
_________________________________________________________________
dense_23 (Dense)             (None, 6)                774
=================================================================
Total params: 252,486
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Trainable params: 251,334

Non-trainable params: 1,152

_________________________________________________________________

None

Train on 7352 samples, validate on 2947 samples

Epoch 1/30

 - 116s - loss: 1.0469 - accuracy: 0.5778 - val_loss: 2.3730 - val_accu
racy: 0.3553

Epoch 2/30

 - 96s - loss: 0.7489 - accuracy: 0.6700 - val_loss: 0.9574 - val_accur
acy: 0.6108

Epoch 3/30

 - 90s - loss: 0.6089 - accuracy: 0.7314 - val_loss: 2.6899 - val_accur
acy: 0.3682

Epoch 4/30

 - 87s - loss: 0.4895 - accuracy: 0.8056 - val_loss: 1.0520 - val_accur
acy: 0.6223

Epoch 5/30

 - 86s - loss: 0.4084 - accuracy: 0.8353 - val_loss: 1.0424 - val_accur
acy: 0.6138

Epoch 6/30

 - 94s - loss: 0.3931 - accuracy: 0.8534 - val_loss: 1.4882 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.6088

Epoch 7/30

 - 86s - loss: 0.3676 - accuracy: 0.8630 - val_loss: 0.7014 - val_accur
acy: 0.8073

Epoch 8/30

 - 82s - loss: 0.3316 - accuracy: 0.8794 - val_loss: 0.5296 - val_accur
acy: 0.8188

Epoch 9/30

 - 85s - loss: 0.3093 - accuracy: 0.8848 - val_loss: 1.0281 - val_accur
acy: 0.7143

Epoch 10/30

 - 85s - loss: 0.3211 - accuracy: 0.8844 - val_loss: 0.6446 - val_accur
acy: 0.8015

Epoch 11/30

 - 85s - loss: 0.2939 - accuracy: 0.8893 - val_loss: 0.5809 - val_accur
acy: 0.8426

Epoch 12/30

 - 91s - loss: 0.2852 - accuracy: 0.8913 - val_loss: 0.8623 - val_accur
acy: 0.7828

Epoch 13/30

 - 93s - loss: 0.2890 - accuracy: 0.8891 - val_loss: 0.6768 - val_accur
acy: 0.8351

Epoch 14/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 87s - loss: 0.3053 - accuracy: 0.8857 - val_loss: 0.6174 - val_accur
acy: 0.8212

Epoch 15/30

 - 86s - loss: 0.2803 - accuracy: 0.8883 - val_loss: 0.5265 - val_accur
acy: 0.8565

Epoch 16/30

 - 93s - loss: 0.3026 - accuracy: 0.8842 - val_loss: 0.6969 - val_accur
acy: 0.8215

Epoch 17/30

 - 90s - loss: 0.2695 - accuracy: 0.9021 - val_loss: 0.7800 - val_accur
acy: 0.8001

Epoch 18/30

 - 112s - loss: 0.3130 - accuracy: 0.8785 - val_loss: 0.9446 - val_accu
racy: 0.8093

Epoch 19/30

 - 177s - loss: 0.2710 - accuracy: 0.9036 - val_loss: 1.3435 - val_accu
racy: 0.7163

Epoch 20/30

 - 177s - loss: 0.3007 - accuracy: 0.8916 - val_loss: 1.2368 - val_accu
racy: 0.6776

Epoch 21/30

 - 175s - loss: 0.3383 - accuracy: 0.8755 - val_loss: 0.7515 - val_accu
racy: 0.7913

Epoch 22/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 177s - loss: 0.3436 - accuracy: 0.8727 - val_loss: 0.9352 - val_accu
racy: 0.7326

Epoch 23/30

 - 119s - loss: 0.3895 - accuracy: 0.8459 - val_loss: 1.0527 - val_accu
racy: 0.7346

Epoch 24/30

 - 79s - loss: 0.4113 - accuracy: 0.8417 - val_loss: 1.1984 - val_accur
acy: 0.6827

Epoch 25/30

 - 79s - loss: 0.4357 - accuracy: 0.8353 - val_loss: 0.8129 - val_accur
acy: 0.6814

Epoch 26/30

 - 79s - loss: 0.4230 - accuracy: 0.8320 - val_loss: 0.8448 - val_accur
acy: 0.7703

Epoch 27/30

 - 80s - loss: 0.3829 - accuracy: 0.8579 - val_loss: 0.9063 - val_accur
acy: 0.7455

Epoch 28/30

 - 79s - loss: 0.3894 - accuracy: 0.8520 - val_loss: 0.7687 - val_accur
acy: 0.7618

Epoch 29/30

 - 80s - loss: 0.3680 - accuracy: 0.8647 - val_loss: 0.7661 - val_accur
acy: 0.7462
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 30/30

 - 89s - loss: 0.3898 - accuracy: 0.8504 - val_loss: 0.6455 - val_accur
acy: 0.7798

Test accuracy:

0.7797760367393494

------------------------------------------------------------------------
--------------
Model: "sequential_9"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_24 (Dense)             (None, 128, 512)          5120

_________________________________________________________________

batch_normalization_32 (Batc (None, 128, 512)          2048

_________________________________________________________________

Dropout (Dropout)            (None, 128, 512)          0

_________________________________________________________________

dense_25 (Dense)             (None, 128, 128)          65664

_________________________________________________________________

batch_normalization_33 (Batc (None, 128, 128)          512

_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Dropout_1 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 32)           20608
_________________________________________________________________
batch_normalization_34 (Batc (None, 128, 32)           128
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 128)               82432
_________________________________________________________________
batch_normalization_35 (Batc (None, 128)               512
_________________________________________________________________
Dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_26 (Dense)             (None, 6)                 774
=================================================================
Total params: 177,798
Trainable params: 176,198
Non-trainable params: 1,600
_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
None

Train on 7352 samples, validate on 2947 samples

Epoch 1/30

 - 80s - loss: 1.3749 - accuracy: 0.5634 - val_loss: 2.0836 - val_accur
acy: 0.3750

Epoch 2/30

 - 73s - loss: 0.6755 - accuracy: 0.6889 - val_loss: 1.7550 - val_accur
acy: 0.5667

Epoch 3/30

 - 73s - loss: 0.5449 - accuracy: 0.7810 - val_loss: 2.0489 - val_accur
acy: 0.4377

Epoch 4/30

 - 80s - loss: 1.0831 - accuracy: 0.5837 - val_loss: 1.5105 - val_accur
acy: 0.4296

Epoch 5/30

 - 83s - loss: 0.7936 - accuracy: 0.6258 - val_loss: 0.9933 - val_accur
acy: 0.5277

Epoch 6/30

 - 76s - loss: 0.7072 - accuracy: 0.6493 - val_loss: 0.7995 - val_accur
acy: 0.6088

Epoch 7/30

 - 70s - loss: 0.6656 - accuracy: 0.6687 - val_loss: 0.7362 - val_accur
acy: 0.6542
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 8/30

 - 69s - loss: 0.6138 - accuracy: 0.7342 - val_loss: 0.9383 - val_accur
acy: 0.6352

Epoch 9/30

 - 70s - loss: 0.5439 - accuracy: 0.7707 - val_loss: 0.6401 - val_accur
acy: 0.7279

Epoch 10/30

 - 70s - loss: 0.4730 - accuracy: 0.8088 - val_loss: 0.5231 - val_accur
acy: 0.7978

Epoch 11/30

 - 71s - loss: 0.3622 - accuracy: 0.8732 - val_loss: 0.6004 - val_accur
acy: 0.8191

Epoch 12/30

 - 70s - loss: 0.3623 - accuracy: 0.8743 - val_loss: 0.5130 - val_accur
acy: 0.8276

Epoch 13/30

 - 70s - loss: 0.2901 - accuracy: 0.9010 - val_loss: 0.7637 - val_accur
acy: 0.7655

Epoch 14/30

 - 70s - loss: 0.3137 - accuracy: 0.8949 - val_loss: 0.3208 - val_accur
acy: 0.8884

Epoch 15/30

 - 70s - loss: 0.2414 - accuracy: 0.9227 - val_loss: 0.3945 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.8816

Epoch 16/30

 - 70s - loss: 0.2064 - accuracy: 0.9295 - val_loss: 0.4029 - val_accur
acy: 0.8873

Epoch 17/30

 - 70s - loss: 0.2165 - accuracy: 0.9301 - val_loss: 0.4663 - val_accur
acy: 0.8755

Epoch 18/30

 - 70s - loss: 0.2223 - accuracy: 0.9227 - val_loss: 0.3967 - val_accur
acy: 0.8768

Epoch 19/30

 - 70s - loss: 0.2575 - accuracy: 0.9163 - val_loss: 0.3672 - val_accur
acy: 0.8711

Epoch 20/30

 - 70s - loss: 0.2129 - accuracy: 0.9238 - val_loss: 0.4131 - val_accur
acy: 0.8605

Epoch 21/30

 - 70s - loss: 0.2188 - accuracy: 0.9317 - val_loss: 0.4872 - val_accur
acy: 0.8663

Epoch 22/30

 - 70s - loss: 0.1947 - accuracy: 0.9334 - val_loss: 0.5468 - val_accur
acy: 0.8622

Epoch 23/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 70s - loss: 0.2032 - accuracy: 0.9297 - val_loss: 0.5285 - val_accur
acy: 0.8582

Epoch 24/30

 - 70s - loss: 0.2101 - accuracy: 0.9294 - val_loss: 0.5344 - val_accur
acy: 0.8633

Epoch 25/30

 - 68s - loss: 0.1981 - accuracy: 0.9329 - val_loss: 0.6843 - val_accur
acy: 0.8541

Epoch 26/30

 - 67s - loss: 0.1888 - accuracy: 0.9314 - val_loss: 0.4059 - val_accur
acy: 0.8846

Epoch 27/30

 - 67s - loss: 0.1883 - accuracy: 0.9355 - val_loss: 0.3596 - val_accur
acy: 0.8951

Epoch 28/30

 - 67s - loss: 0.2234 - accuracy: 0.9237 - val_loss: 0.4629 - val_accur
acy: 0.8697

Epoch 29/30

 - 67s - loss: 0.2224 - accuracy: 0.9232 - val_loss: 0.8168 - val_accur
acy: 0.8100

Epoch 30/30

 - 67s - loss: 0.2703 - accuracy: 0.9078 - val_loss: 0.5247 - val_accur
acy: 0.8704

Test accuracy:
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
0.8703766465187073

_________________________________________________________________
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_27 (Dense)             (None, 128, 128)          1280
_________________________________________________________________
batch_normalization_36 (Batc (None, 128, 128)          512
_________________________________________________________________
Dropout (Dropout)            (None, 128, 128)          0
_________________________________________________________________
dense_28 (Dense)             (None, 128, 128)          16512
_________________________________________________________________
batch_normalization_37 (Batc (None, 128, 128)          512
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 128)          131584
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
batch_normalization_38 (Batc (None, 128, 128)          512
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 64)                49408
_________________________________________________________________
batch_normalization_39 (Batc (None, 64)                256
_________________________________________________________________
Dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_29 (Dense)             (None, 6)                 390
=================================================================
Total params: 200,966
Trainable params: 200,070
Non-trainable params: 896
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 69s - loss: 1.6692 - accuracy: 0.4121 - val_loss: 2.3543 - val_accur
acy: 0.4584

Epoch 2/30

 - 67s - loss: 0.8762 - accuracy: 0.6223 - val_loss: 4.1424 - val_accur
acy: 0.2263

Epoch 3/30

 - 66s - loss: 0.6872 - accuracy: 0.6598 - val_loss: 4.5076 - val_accur
acy: 0.2878

Epoch 4/30

 - 65s - loss: 0.6481 - accuracy: 0.6759 - val_loss: 4.8934 - val_accur
acy: 0.2956

Epoch 5/30

 - 64s - loss: 0.6298 - accuracy: 0.6868 - val_loss: 5.3336 - val_accur
acy: 0.2782

Epoch 6/30

 - 64s - loss: 0.6159 - accuracy: 0.6893 - val_loss: 5.1988 - val_accur
acy: 0.2857

Epoch 7/30

 - 64s - loss: 0.5986 - accuracy: 0.7125 - val_loss: 4.7328 - val_accur
acy: 0.2864

Epoch 8/30

 - 65s - loss: 0.5806 - accuracy: 0.7195 - val_loss: 5.3285 - val_accur
acy: 0.2820
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 9/30

 - 65s - loss: 0.5505 - accuracy: 0.7463 - val_loss: 4.6668 - val_accur
acy: 0.3583

Epoch 10/30

 - 64s - loss: 0.5117 - accuracy: 0.7769 - val_loss: 4.8476 - val_accur
acy: 0.3247

Epoch 11/30

 - 64s - loss: 0.4699 - accuracy: 0.8054 - val_loss: 4.4200 - val_accur
acy: 0.3753

Epoch 12/30

 - 64s - loss: 0.4201 - accuracy: 0.8369 - val_loss: 3.9720 - val_accur
acy: 0.3814

Epoch 13/30

 - 65s - loss: 0.3633 - accuracy: 0.8611 - val_loss: 3.4335 - val_accur
acy: 0.4890

Epoch 14/30

 - 65s - loss: 0.3366 - accuracy: 0.8757 - val_loss: 2.2139 - val_accur
acy: 0.5691

Epoch 15/30

 - 64s - loss: 0.2960 - accuracy: 0.8938 - val_loss: 1.5775 - val_accur
acy: 0.6675

Epoch 16/30

 - 65s - loss: 0.2559 - accuracy: 0.9063 - val_loss: 1.7411 - val_accur
acy: 0.6909
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 17/30

 - 65s - loss: 0.2270 - accuracy: 0.9202 - val_loss: 2.0685 - val_accur
acy: 0.6474

Epoch 18/30

 - 65s - loss: 0.2087 - accuracy: 0.9221 - val_loss: 1.1850 - val_accur
acy: 0.7462

Epoch 19/30

 - 65s - loss: 0.1987 - accuracy: 0.9289 - val_loss: 1.6554 - val_accur
acy: 0.6895

Epoch 20/30

 - 65s - loss: 0.1918 - accuracy: 0.9274 - val_loss: 1.5537 - val_accur
acy: 0.6905

Epoch 21/30

 - 65s - loss: 0.1795 - accuracy: 0.9304 - val_loss: 0.8066 - val_accur
acy: 0.8052

Epoch 22/30

 - 65s - loss: 0.2138 - accuracy: 0.9225 - val_loss: 1.0001 - val_accur
acy: 0.7723

Epoch 23/30

 - 65s - loss: 0.1727 - accuracy: 0.9354 - val_loss: 0.4666 - val_accur
acy: 0.8680

Epoch 24/30

 - 64s - loss: 0.1757 - accuracy: 0.9310 - val_loss: 1.0784 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.7808

Epoch 25/30

 - 65s - loss: 0.1829 - accuracy: 0.9323 - val_loss: 0.8086 - val_accur
acy: 0.8120

Epoch 26/30

 - 64s - loss: 0.1677 - accuracy: 0.9332 - val_loss: 0.5143 - val_accur
acy: 0.8656

Epoch 27/30

 - 64s - loss: 0.1708 - accuracy: 0.9353 - val_loss: 0.4243 - val_accur
acy: 0.8819

Epoch 28/30

 - 64s - loss: 0.1656 - accuracy: 0.9369 - val_loss: 0.5601 - val_accur
acy: 0.8544

Epoch 29/30

 - 65s - loss: 0.1523 - accuracy: 0.9404 - val_loss: 0.4452 - val_accur
acy: 0.8806

Epoch 30/30

 - 65s - loss: 0.1539 - accuracy: 0.9366 - val_loss: 0.4414 - val_accur
acy: 0.8755

Test accuracy:

0.8754665851593018

_________________________________________________________________
______________
Model: "sequential_11"
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_30 (Dense)             (None, 128, 128)          1280
_________________________________________________________________
batch_normalization_40 (Batc (None, 128, 128)          512
_________________________________________________________________
Dropout (Dropout)            (None, 128, 128)          0
_________________________________________________________________
dense_31 (Dense)             (None, 128, 256)          33024
_________________________________________________________________
batch_normalization_41 (Batc (None, 128, 256)          1024
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 256)          0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 32)           36992
_________________________________________________________________
batch_normalization_42 (Batc (None, 128, 32)           128
_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Dropout_2 (Dropout)          (None, 128, 32)          0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 64)               24832
_________________________________________________________________
batch_normalization_43 (Batc (None, 64)               256
_________________________________________________________________
Dropout_3 (Dropout)          (None, 64)               0
_________________________________________________________________
dense_32 (Dense)             (None, 6)                390
=================================================================
Total params: 98,438
Trainable params: 97,478
Non-trainable params: 960
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
 - 48s - loss: 1.4967 - accuracy: 0.4853 - val_loss: 1.3928 - val_accur
acy: 0.3614
Epoch 2/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 45s - loss: 0.7824 - accuracy: 0.6491 - val_loss: 1.2616 - val_accur
acy: 0.4910

Epoch 3/30

 - 45s - loss: 0.6626 - accuracy: 0.6714 - val_loss: 1.9916 - val_accur
acy: 0.4584

Epoch 4/30

 - 46s - loss: 0.6227 - accuracy: 0.6937 - val_loss: 2.7206 - val_accur
acy: 0.4571

Epoch 5/30

 - 45s - loss: 0.5425 - accuracy: 0.7504 - val_loss: 2.4229 - val_accur
acy: 0.4886

Epoch 6/30

 - 45s - loss: 0.4930 - accuracy: 0.7941 - val_loss: 2.3912 - val_accur
acy: 0.5246

Epoch 7/30

 - 45s - loss: 0.4227 - accuracy: 0.8353 - val_loss: 2.0128 - val_accur
acy: 0.5833

Epoch 8/30

 - 45s - loss: 0.3596 - accuracy: 0.8694 - val_loss: 1.7832 - val_accur
acy: 0.6356

Epoch 9/30

 - 45s - loss: 0.3086 - accuracy: 0.8917 - val_loss: 1.2951 - val_accur
acy: 0.7065

Epoch 10/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 45s - loss: 0.2580 - accuracy: 0.9076 - val_loss: 1.0127 - val_accur
acy: 0.7743

Epoch 11/30

 - 45s - loss: 0.2336 - accuracy: 0.9196 - val_loss: 0.6159 - val_accur
acy: 0.8202

Epoch 12/30

 - 46s - loss: 0.2144 - accuracy: 0.9207 - val_loss: 0.7242 - val_accur
acy: 0.8164

Epoch 13/30

 - 45s - loss: 0.2081 - accuracy: 0.9232 - val_loss: 0.5309 - val_accur
acy: 0.8605

Epoch 14/30

 - 45s - loss: 0.1927 - accuracy: 0.9301 - val_loss: 0.5552 - val_accur
acy: 0.8521

Epoch 15/30

 - 45s - loss: 0.1763 - accuracy: 0.9348 - val_loss: 0.5997 - val_accur
acy: 0.8571

Epoch 16/30

 - 45s - loss: 0.1727 - accuracy: 0.9350 - val_loss: 0.3053 - val_accur
acy: 0.8897

Epoch 17/30

 - 45s - loss: 0.2179 - accuracy: 0.9271 - val_loss: 0.4554 - val_accur
acy: 0.8385
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 18/30

 - 45s - loss: 0.1790 - accuracy: 0.9294 - val_loss: 0.3138 - val_accur
acy: 0.8907

Epoch 19/30

 - 45s - loss: 0.1555 - accuracy: 0.9407 - val_loss: 0.3378 - val_accur
acy: 0.8873

Epoch 20/30

 - 45s - loss: 0.1629 - accuracy: 0.9397 - val_loss: 0.3650 - val_accur
acy: 0.8677

Epoch 21/30

 - 45s - loss: 0.1526 - accuracy: 0.9382 - val_loss: 0.3601 - val_accur
acy: 0.8911

Epoch 22/30

 - 45s - loss: 0.1475 - accuracy: 0.9402 - val_loss: 0.3848 - val_accur
acy: 0.8792

Epoch 23/30

 - 45s - loss: 0.1589 - accuracy: 0.9415 - val_loss: 0.3761 - val_accur
acy: 0.8782

Epoch 24/30

 - 45s - loss: 0.1407 - accuracy: 0.9423 - val_loss: 0.3485 - val_accur
acy: 0.8890

Epoch 25/30

 - 45s - loss: 0.1407 - accuracy: 0.9429 - val_loss: 0.3077 - val_accur
acy: 0.9080
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 26/30

 - 45s - loss: 0.1355 - accuracy: 0.9474 - val_loss: 0.3027 - val_accur
acy: 0.9013

Epoch 27/30

 - 46s - loss: 0.1300 - accuracy: 0.9464 - val_loss: 0.2876 - val_accur
acy: 0.9013

Epoch 28/30

 - 45s - loss: 0.1339 - accuracy: 0.9476 - val_loss: 0.2763 - val_accur
acy: 0.9070

Epoch 29/30

 - 45s - loss: 0.1295 - accuracy: 0.9460 - val_loss: 0.2645 - val_accur
acy: 0.9013

Epoch 30/30

 - 45s - loss: 0.1244 - accuracy: 0.9502 - val_loss: 0.3285 - val_accur
acy: 0.8935

Test accuracy:

0.8934509754180908

---------------------------------------------------------------------
--------------
Model: "sequential_12"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
dense_33 (Dense)             (None, 128, 256)          2560
_________________________________________________________________
batch_normalization_44 (Batc (None, 128, 256)          1024
_________________________________________________________________
Dropout (Dropout)            (None, 128, 256)          0
_________________________________________________________________
dense_34 (Dense)             (None, 128, 32)           8224
_________________________________________________________________
batch_normalization_45 (Batc (None, 128, 32)           128
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 128)          82432
_________________________________________________________________
batch_normalization_46 (Batc (None, 128, 128)          512
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 64)                49408
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
batch_normalization_47 (Batc (None, 64)                256
_________________________________________________________________
Dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_35 (Dense)             (None, 6)                 390
=================================================================
Total params: 144,934
Trainable params: 143,974
Non-trainable params: 960
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
 - 63s - loss: 0.9390 - accuracy: 0.6043 - val_loss: 1.8894 - val_accur
acy: 0.3546
Epoch 2/30
 - 60s - loss: 0.6531 - accuracy: 0.6933 - val_loss: 2.3851 - val_accur
acy: 0.4028
Epoch 3/30
 - 60s - loss: 0.5097 - accuracy: 0.7817 - val_loss: 1.4410 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.5484

Epoch 4/30

 - 60s - loss: 0.3586 - accuracy: 0.8662 - val_loss: 1.2662 - val_accur
acy: 0.5979

Epoch 5/30

 - 59s - loss: 0.2721 - accuracy: 0.8950 - val_loss: 1.0411 - val_accur
acy: 0.7119

Epoch 6/30

 - 60s - loss: 0.2511 - accuracy: 0.9090 - val_loss: 0.8906 - val_accur
acy: 0.7523

Epoch 7/30

 - 59s - loss: 0.2025 - accuracy: 0.9252 - val_loss: 0.7149 - val_accur
acy: 0.8212

Epoch 8/30

 - 60s - loss: 0.1982 - accuracy: 0.9263 - val_loss: 0.6452 - val_accur
acy: 0.8432

Epoch 9/30

 - 59s - loss: 0.1621 - accuracy: 0.9382 - val_loss: 0.9362 - val_accur
acy: 0.7883

Epoch 10/30

 - 59s - loss: 0.1623 - accuracy: 0.9389 - val_loss: 0.3150 - val_accur
acy: 0.8982

Epoch 11/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 60s - loss: 0.1584 - accuracy: 0.9350 - val_loss: 0.4018 - val_accur
acy: 0.8738

Epoch 12/30

 - 60s - loss: 0.1535 - accuracy: 0.9421 - val_loss: 0.3269 - val_accur
acy: 0.9050

Epoch 13/30

 - 59s - loss: 0.1404 - accuracy: 0.9459 - val_loss: 0.2725 - val_accur
acy: 0.9067

Epoch 14/30

 - 60s - loss: 0.1379 - accuracy: 0.9444 - val_loss: 0.3868 - val_accur
acy: 0.9002

Epoch 15/30

 - 61s - loss: 0.1373 - accuracy: 0.9452 - val_loss: 0.3995 - val_accur
acy: 0.8792

Epoch 16/30

 - 59s - loss: 0.1427 - accuracy: 0.9429 - val_loss: 0.3939 - val_accur
acy: 0.9002

Epoch 17/30

 - 59s - loss: 0.1408 - accuracy: 0.9444 - val_loss: 0.2993 - val_accur
acy: 0.9023

Epoch 18/30

 - 59s - loss: 0.1281 - accuracy: 0.9465 - val_loss: 0.4036 - val_accur
acy: 0.8863

Epoch 19/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 59s - loss: 0.1324 - accuracy: 0.9474 - val_loss: 0.4000 - val_accur
acy: 0.9009

Epoch 20/30

 - 59s - loss: 0.1224 - accuracy: 0.9506 - val_loss: 0.3091 - val_accur
acy: 0.9097

Epoch 21/30

 - 59s - loss: 0.1181 - accuracy: 0.9504 - val_loss: 0.3954 - val_accur
acy: 0.8985

Epoch 22/30

 - 62s - loss: 0.1272 - accuracy: 0.9501 - val_loss: 0.2356 - val_accur
acy: 0.9158

Epoch 23/30

 - 63s - loss: 0.1283 - accuracy: 0.9482 - val_loss: 0.2749 - val_accur
acy: 0.9128

Epoch 24/30

 - 63s - loss: 0.1333 - accuracy: 0.9495 - val_loss: 0.2789 - val_accur
acy: 0.9264

Epoch 25/30

 - 63s - loss: 0.1206 - accuracy: 0.9497 - val_loss: 0.2730 - val_accur
acy: 0.9074

Epoch 26/30

 - 63s - loss: 0.1209 - accuracy: 0.9468 - val_loss: 0.2651 - val_accur
acy: 0.9189
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 27/30

 - 63s - loss: 0.1137 - accuracy: 0.9533 - val_loss: 0.2610 - val_accur
acy: 0.9237

Epoch 28/30

 - 63s - loss: 0.1196 - accuracy: 0.9486 - val_loss: 0.3052 - val_accur
acy: 0.9243

Epoch 29/30

 - 63s - loss: 0.1103 - accuracy: 0.9518 - val_loss: 0.2461 - val_accur
acy: 0.9284

Epoch 30/30

 - 63s - loss: 0.1230 - accuracy: 0.9502 - val_loss: 0.2324 - val_accur
acy: 0.9270

Test accuracy:

0.9270444512367249

---------------------------------------------------------------------
--------------
Model: "sequential_13"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_36 (Dense)             (None, 128, 128)          1280

_________________________________________________________________

batch_normalization_48 (Batc (None, 128, 128)          512
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
Dropout (Dropout)            (None, 128, 128)          0
_________________________________________________________________
dense_37 (Dense)             (None, 128, 32)           4128
_________________________________________________________________
batch_normalization_49 (Batc (None, 128, 32)           128
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 128)          82432
_________________________________________________________________
batch_normalization_50 (Batc (None, 128, 128)          512
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 16)                9280
_________________________________________________________________
batch_normalization_51 (Batc (None, 16)                64
_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Dropout_3 (Dropout)          (None, 16)                0

_________________________________________________________________

dense_38 (Dense)             (None, 6)                 102

=================================================================

Total params: 98,438

Trainable params: 97,830

Non-trainable params: 608

_________________________________________________________________

None

Train on 7352 samples, validate on 2947 samples

Epoch 1/30

 - 49s - loss: 1.2715 - accuracy: 0.4695 - val_loss: 1.8053 - val_accur
acy: 0.3275

Epoch 2/30

 - 46s - loss: 0.8867 - accuracy: 0.5726 - val_loss: 1.4557 - val_accur
acy: 0.4316

Epoch 3/30

 - 46s - loss: 0.8629 - accuracy: 0.5917 - val_loss: 2.3540 - val_accur
acy: 0.4710

Epoch 4/30

 - 46s - loss: 0.8126 - accuracy: 0.6009 - val_loss: 1.8757 - val_accur
acy: 0.5307
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 5/30

 - 46s - loss: 0.7942 - accuracy: 0.6069 - val_loss: 3.0533 - val_accur
acy: 0.4754

Epoch 6/30

 - 46s - loss: 0.7493 - accuracy: 0.6115 - val_loss: 1.9592 - val_accur
acy: 0.5368

Epoch 7/30

 - 46s - loss: 0.7160 - accuracy: 0.6319 - val_loss: 1.4061 - val_accur
acy: 0.5422

Epoch 8/30

 - 46s - loss: 0.7226 - accuracy: 0.6413 - val_loss: 1.8988 - val_accur
acy: 0.4863

Epoch 9/30

 - 46s - loss: 0.7086 - accuracy: 0.6631 - val_loss: 1.2693 - val_accur
acy: 0.5636

Epoch 10/30

 - 46s - loss: 0.6914 - accuracy: 0.6824 - val_loss: 0.7826 - val_accur
acy: 0.6356

Epoch 11/30

 - 46s - loss: 0.6866 - accuracy: 0.6846 - val_loss: 0.5927 - val_accur
acy: 0.7184

Epoch 12/30

 - 46s - loss: 0.6062 - accuracy: 0.7055 - val_loss: 0.6992 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.7265

Epoch 13/30

 - 47s - loss: 0.5932 - accuracy: 0.7104 - val_loss: 0.5875 - val_accur
acy: 0.7346

Epoch 14/30

 - 46s - loss: 0.5432 - accuracy: 0.7333 - val_loss: 0.5857 - val_accur
acy: 0.7133

Epoch 15/30

 - 47s - loss: 0.5276 - accuracy: 0.7308 - val_loss: 0.5042 - val_accur
acy: 0.7591

Epoch 16/30

 - 46s - loss: 0.5412 - accuracy: 0.7359 - val_loss: 0.5172 - val_accur
acy: 0.7492

Epoch 17/30

 - 47s - loss: 0.6202 - accuracy: 0.7171 - val_loss: 0.5282 - val_accur
acy: 0.7336

Epoch 18/30

 - 46s - loss: 0.5397 - accuracy: 0.7338 - val_loss: 0.4837 - val_accur
acy: 0.7604

Epoch 19/30

 - 46s - loss: 0.5054 - accuracy: 0.7420 - val_loss: 0.5435 - val_accur
acy: 0.7258

Epoch 20/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 46s - loss: 0.4996 - accuracy: 0.7383 - val_loss: 0.5773 - val_accur
acy: 0.7367

Epoch 21/30

 - 46s - loss: 0.5118 - accuracy: 0.7295 - val_loss: 0.4464 - val_accur
acy: 0.7638

Epoch 22/30

 - 46s - loss: 0.5477 - accuracy: 0.7304 - val_loss: 0.5396 - val_accur
acy: 0.7340

Epoch 23/30

 - 46s - loss: 0.4992 - accuracy: 0.7416 - val_loss: 0.4935 - val_accur
acy: 0.7659

Epoch 24/30

 - 46s - loss: 0.4953 - accuracy: 0.7397 - val_loss: 0.5266 - val_accur
acy: 0.7635

Epoch 25/30

 - 46s - loss: 0.5096 - accuracy: 0.7349 - val_loss: 0.4777 - val_accur
acy: 0.7441

Epoch 26/30

 - 46s - loss: 0.5099 - accuracy: 0.7350 - val_loss: 0.6842 - val_accur
acy: 0.7190

Epoch 27/30

 - 46s - loss: 0.5187 - accuracy: 0.7383 - val_loss: 0.5598 - val_accur
acy: 0.7543

Epoch 28/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 46s - loss: 0.4853 - accuracy: 0.7401 - val_loss: 0.6481 - val_accur
acy: 0.7350

Epoch 29/30

 - 46s - loss: 0.4942 - accuracy: 0.7427 - val_loss: 0.7912 - val_accur
acy: 0.7204

Epoch 30/30

 - 46s - loss: 0.4921 - accuracy: 0.7406 - val_loss: 0.6470 - val_accur
acy: 0.7458

Test accuracy:

0.745843231678009

---------------------------------------------------------------------
--------------
Model: "sequential_14"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_39 (Dense)             (None, 128, 128)          1280

_________________________________________________________________

batch_normalization_52 (Batc (None, 128, 128)          512

_________________________________________________________________

Dropout (Dropout)            (None, 128, 128)          0

_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
dense_40 (Dense)             (None, 128, 64)          8256
_________________________________________________________________
batch_normalization_53 (Batc (None, 128, 64)          256
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 64)          0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 32)          12416
_________________________________________________________________
batch_normalization_54 (Batc (None, 128, 32)          128
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 32)          0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 64)               24832
_________________________________________________________________
batch_normalization_55 (Batc (None, 64)               256
_________________________________________________________________
Dropout_3 (Dropout)          (None, 64)               0
_________________________________________________________________
dense_41 (Dense)             (None, 6)                390
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
=================================================================

Total params: 48,326

Trainable params: 47,750

Non-trainable params: 576

_________________________________________________________________

None

Train on 7352 samples, validate on 2947 samples

Epoch 1/30

 - 33s - loss: 0.9828 - accuracy: 0.5875 - val_loss: 2.1382 - val_accur
acy: 0.3465

Epoch 2/30

 - 30s - loss: 0.5781 - accuracy: 0.7295 - val_loss: 1.8173 - val_accur
acy: 0.4489

Epoch 3/30

 - 30s - loss: 0.4556 - accuracy: 0.8217 - val_loss: 1.9863 - val_accur
acy: 0.3923

Epoch 4/30

 - 30s - loss: 0.3235 - accuracy: 0.8740 - val_loss: 1.3857 - val_accur
acy: 0.6247

Epoch 5/30

 - 30s - loss: 0.2468 - accuracy: 0.9101 - val_loss: 1.6015 - val_accur
acy: 0.6216
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 6/30

 - 30s - loss: 0.2180 - accuracy: 0.9150 - val_loss: 1.7451 - val_accur
acy: 0.5752

Epoch 7/30

 - 29s - loss: 0.2131 - accuracy: 0.9233 - val_loss: 0.6877 - val_accur
acy: 0.7957

Epoch 8/30

 - 30s - loss: 0.1899 - accuracy: 0.9325 - val_loss: 0.7156 - val_accur
acy: 0.8025

Epoch 9/30

 - 30s - loss: 0.1574 - accuracy: 0.9393 - val_loss: 0.5308 - val_accur
acy: 0.8429

Epoch 10/30

 - 30s - loss: 0.1675 - accuracy: 0.9347 - val_loss: 0.5830 - val_accur
acy: 0.8273

Epoch 11/30

 - 30s - loss: 0.1491 - accuracy: 0.9430 - val_loss: 0.4528 - val_accur
acy: 0.8670

Epoch 12/30

 - 30s - loss: 0.1600 - accuracy: 0.9406 - val_loss: 0.3831 - val_accur
acy: 0.8819

Epoch 13/30

 - 30s - loss: 0.1934 - accuracy: 0.9248 - val_loss: 0.3165 - val_accur
acy: 0.8948
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 14/30

 - 30s - loss: 0.1827 - accuracy: 0.9300 - val_loss: 0.6887 - val_accur
acy: 0.7866

Epoch 15/30

 - 30s - loss: 0.1879 - accuracy: 0.9271 - val_loss: 0.5012 - val_accur
acy: 0.8544

Epoch 16/30

 - 30s - loss: 0.1489 - accuracy: 0.9410 - val_loss: 0.5596 - val_accur
acy: 0.8551

Epoch 17/30

 - 29s - loss: 0.1578 - accuracy: 0.9384 - val_loss: 0.5365 - val_accur
acy: 0.8385

Epoch 18/30

 - 30s - loss: 0.1639 - accuracy: 0.9365 - val_loss: 0.4142 - val_accur
acy: 0.8724

Epoch 19/30

 - 30s - loss: 0.1361 - accuracy: 0.9399 - val_loss: 0.3950 - val_accur
acy: 0.8860

Epoch 20/30

 - 30s - loss: 0.1330 - accuracy: 0.9480 - val_loss: 0.4825 - val_accur
acy: 0.8724

Epoch 21/30

 - 30s - loss: 0.1301 - accuracy: 0.9471 - val_loss: 0.3718 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.8873

Epoch 22/30

 - 30s - loss: 0.1324 - accuracy: 0.9476 - val_loss: 0.3852 - val_accur
acy: 0.8941

Epoch 23/30

 - 30s - loss: 0.1251 - accuracy: 0.9489 - val_loss: 0.3772 - val_accur
acy: 0.8928

Epoch 24/30

 - 30s - loss: 0.1216 - accuracy: 0.9501 - val_loss: 0.3647 - val_accur
acy: 0.9053

Epoch 25/30

 - 30s - loss: 0.1153 - accuracy: 0.9483 - val_loss: 0.3821 - val_accur
acy: 0.8982

Epoch 26/30

 - 30s - loss: 0.1456 - accuracy: 0.9438 - val_loss: 0.3625 - val_accur
acy: 0.9013

Epoch 27/30

 - 30s - loss: 0.1307 - accuracy: 0.9480 - val_loss: 0.3091 - val_accur
acy: 0.9070

Epoch 28/30

 - 30s - loss: 0.1386 - accuracy: 0.9434 - val_loss: 0.2772 - val_accur
acy: 0.9101

Epoch 29/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 30s - loss: 0.1410 - accuracy: 0.9355 - val_loss: 0.2949 - val_accur
acy: 0.9101

Epoch 30/30

 - 30s - loss: 0.1426 - accuracy: 0.9415 - val_loss: 0.3343 - val_accur
acy: 0.8945

Test accuracy:

0.8944689631462097

------------------------------------------------------------------------
--------------
Model: "sequential_15"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_42 (Dense)             (None, 128, 256)          2560

_________________________________________________________________

batch_normalization_56 (Batc (None, 128, 256)          1024

_________________________________________________________________

Dropout (Dropout)            (None, 128, 256)          0

_________________________________________________________________

dense_43 (Dense)             (None, 128, 256)          65792

_________________________________________________________________

batch_normalization_57 (Batc (None, 128, 256)          1024
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 256)          0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 64)           82176
_________________________________________________________________
batch_normalization_58 (Batc (None, 128, 64)           256
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 128)               98816
_________________________________________________________________
batch_normalization_59 (Batc (None, 128)               512
_________________________________________________________________
Dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_44 (Dense)             (None, 6)                 774
=================================================================
Total params: 252,934
Trainable params: 251,526
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Non-trainable params: 1,408
_________________________________________________________________
None

Train on 7352 samples, validate on 2947 samples

Epoch 1/30

 - 88s - loss: 0.9010 - accuracy: 0.6493 - val_loss: 2.2005 - val_accur
acy: 0.3485

Epoch 2/30

 - 84s - loss: 0.4593 - accuracy: 0.8260 - val_loss: 1.3095 - val_accur
acy: 0.5870

Epoch 3/30

 - 84s - loss: 0.2797 - accuracy: 0.8927 - val_loss: 0.8625 - val_accur
acy: 0.7268

Epoch 4/30

 - 84s - loss: 0.2296 - accuracy: 0.9144 - val_loss: 0.5488 - val_accur
acy: 0.8164

Epoch 5/30

 - 84s - loss: 0.1724 - accuracy: 0.9338 - val_loss: 0.5669 - val_accur
acy: 0.8395

Epoch 6/30

 - 84s - loss: 0.1691 - accuracy: 0.9354 - val_loss: 0.4709 - val_accur
acy: 0.8527

Epoch 7/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 84s - loss: 0.1610 - accuracy: 0.9406 - val_loss: 0.3317 - val_accur
acy: 0.8965

Epoch 8/30

 - 84s - loss: 0.1599 - accuracy: 0.9381 - val_loss: 0.2541 - val_accur
acy: 0.9114

Epoch 9/30

 - 84s - loss: 0.1471 - accuracy: 0.9399 - val_loss: 0.2857 - val_accur
acy: 0.8806

Epoch 10/30

 - 84s - loss: 0.1477 - accuracy: 0.9410 - val_loss: 0.3213 - val_accur
acy: 0.8911

Epoch 11/30

 - 84s - loss: 0.1720 - accuracy: 0.9407 - val_loss: 0.3864 - val_accur
acy: 0.8935

Epoch 12/30

 - 84s - loss: 0.1406 - accuracy: 0.9429 - val_loss: 0.3205 - val_accur
acy: 0.9050

Epoch 13/30

 - 84s - loss: 0.1442 - accuracy: 0.9395 - val_loss: 0.5107 - val_accur
acy: 0.8897

Epoch 14/30

 - 84s - loss: 0.1513 - accuracy: 0.9402 - val_loss: 0.2175 - val_accur
acy: 0.9141
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 15/30

 - 84s - loss: 0.1451 - accuracy: 0.9404 - val_loss: 0.3806 - val_accur
acy: 0.8880

Epoch 16/30

 - 84s - loss: 0.1308 - accuracy: 0.9464 - val_loss: 0.3281 - val_accur
acy: 0.9077

Epoch 17/30

 - 84s - loss: 0.1351 - accuracy: 0.9442 - val_loss: 0.4222 - val_accur
acy: 0.8806

Epoch 18/30

 - 83s - loss: 0.1196 - accuracy: 0.9491 - val_loss: 0.3458 - val_accur
acy: 0.9141

Epoch 19/30

 - 84s - loss: 0.1275 - accuracy: 0.9472 - val_loss: 0.3567 - val_accur
acy: 0.9074

Epoch 20/30

 - 84s - loss: 0.1249 - accuracy: 0.9499 - val_loss: 0.5937 - val_accur
acy: 0.8629

Epoch 21/30

 - 83s - loss: 0.1161 - accuracy: 0.9498 - val_loss: 0.5268 - val_accur
acy: 0.8955

Epoch 22/30

 - 84s - loss: 0.1110 - accuracy: 0.9498 - val_loss: 0.3464 - val_accur
acy: 0.9175
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 23/30

 - 84s - loss: 0.1190 - accuracy: 0.9482 - val_loss: 0.3106 - val_accur
acy: 0.9169

Epoch 24/30

 - 84s - loss: 0.1256 - accuracy: 0.9501 - val_loss: 0.4766 - val_accur
acy: 0.8775

Epoch 25/30

 - 84s - loss: 0.1172 - accuracy: 0.9476 - val_loss: 0.4108 - val_accur
acy: 0.9138

Epoch 26/30

 - 84s - loss: 0.1165 - accuracy: 0.9514 - val_loss: 0.3977 - val_accur
acy: 0.9030

Epoch 27/30

 - 84s - loss: 0.1182 - accuracy: 0.9520 - val_loss: 0.4643 - val_accur
acy: 0.9074

Epoch 28/30

 - 83s - loss: 0.1184 - accuracy: 0.9502 - val_loss: 0.4835 - val_accur
acy: 0.8921

Epoch 29/30

 - 84s - loss: 0.1134 - accuracy: 0.9536 - val_loss: 0.5213 - val_accur
acy: 0.8870

Epoch 30/30

 - 84s - loss: 0.1148 - accuracy: 0.9517 - val_loss: 0.4620 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.8985

Test accuracy:

0.8985409140586853

_________________________________________________________________
_____________
Model: "sequential_16"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_45 (Dense)             (None, 128, 512)          5120

_________________________________________________________________

batch_normalization_60 (Batc (None, 128, 512)          2048

_________________________________________________________________

Dropout (Dropout)            (None, 128, 512)          0

_________________________________________________________________

dense_46 (Dense)             (None, 128, 256)          131328

_________________________________________________________________

batch_normalization_61 (Batc (None, 128, 256)          1024

_________________________________________________________________

Dropout_1 (Dropout)          (None, 128, 256)          0

_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
LSTM_1 (LSTM)                (None, 128, 128)          197120
_________________________________________________________________
batch_normalization_62 (Batc (None, 128, 128)          512
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 64)                49408
_________________________________________________________________
batch_normalization_63 (Batc (None, 64)                256
_________________________________________________________________
Dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_47 (Dense)             (None, 6)                 390
=================================================================
Total params: 387,206
Trainable params: 385,286
Non-trainable params: 1,920
_________________________________________________________________
None
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Train on 7352 samples, validate on 2947 samples

Epoch 1/30

 - 123s - loss: 0.8856 - accuracy: 0.6487 - val_loss: 2.0681 - val_accu
racy: 0.3957

Epoch 2/30

 - 119s - loss: 0.4622 - accuracy: 0.8271 - val_loss: 3.3842 - val_accu
racy: 0.3566

Epoch 3/30

 - 119s - loss: 0.2987 - accuracy: 0.8851 - val_loss: 2.8110 - val_accu
racy: 0.4781

Epoch 4/30

 - 120s - loss: 0.2870 - accuracy: 0.8973 - val_loss: 0.5806 - val_accu
racy: 0.8405

Epoch 5/30

 - 120s - loss: 0.2158 - accuracy: 0.9236 - val_loss: 1.0617 - val_accu
racy: 0.6630

Epoch 6/30

 - 120s - loss: 0.2059 - accuracy: 0.9225 - val_loss: 0.4880 - val_accu
racy: 0.8792

Epoch 7/30

 - 120s - loss: 0.2015 - accuracy: 0.9227 - val_loss: 0.4726 - val_accu
racy: 0.8765

Epoch 8/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 120s - loss: 0.1991 - accuracy: 0.9294 - val_loss: 0.6864 - val_accu
racy: 0.8561

Epoch 9/30

 - 120s - loss: 0.2251 - accuracy: 0.9161 - val_loss: 0.9366 - val_accu
racy: 0.8045

Epoch 10/30

 - 120s - loss: 0.2152 - accuracy: 0.9236 - val_loss: 0.8542 - val_accu
racy: 0.8500

Epoch 11/30

 - 120s - loss: 0.1832 - accuracy: 0.9309 - val_loss: 1.1536 - val_accu
racy: 0.7798

Epoch 12/30

 - 120s - loss: 0.2089 - accuracy: 0.9275 - val_loss: 0.6608 - val_accu
racy: 0.8507

Epoch 13/30

 - 120s - loss: 0.1954 - accuracy: 0.9274 - val_loss: 0.9195 - val_accu
racy: 0.8307

Epoch 14/30

 - 121s - loss: 0.1668 - accuracy: 0.9384 - val_loss: 0.4977 - val_accu
racy: 0.8870

Epoch 15/30

 - 120s - loss: 0.1818 - accuracy: 0.9313 - val_loss: 0.5490 - val_accu
racy: 0.8514

Epoch 16/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 120s - loss: 0.1616 - accuracy: 0.9372 - val_loss: 0.4084 - val_accu
racy: 0.8755

Epoch 17/30

 - 120s - loss: 0.1715 - accuracy: 0.9399 - val_loss: 0.4465 - val_accu
racy: 0.8738

Epoch 18/30

 - 120s - loss: 0.1708 - accuracy: 0.9363 - val_loss: 0.7061 - val_accu
racy: 0.8242

Epoch 19/30

 - 121s - loss: 0.1691 - accuracy: 0.9402 - val_loss: 0.7239 - val_accu
racy: 0.8314

Epoch 20/30

 - 120s - loss: 0.1789 - accuracy: 0.9353 - val_loss: 0.4306 - val_accu
racy: 0.8395

Epoch 21/30

 - 120s - loss: 0.1807 - accuracy: 0.9319 - val_loss: 0.5176 - val_accu
racy: 0.8792

Epoch 22/30

 - 121s - loss: 0.1663 - accuracy: 0.9346 - val_loss: 0.5087 - val_accu
racy: 0.8714

Epoch 23/30

 - 120s - loss: 0.1629 - accuracy: 0.9373 - val_loss: 0.4870 - val_accu
racy: 0.8721
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 24/30

 - 120s - loss: 0.1904 - accuracy: 0.9275 - val_loss: 0.5396 - val_accu
racy: 0.8565

Epoch 25/30

 - 120s - loss: 0.1840 - accuracy: 0.9324 - val_loss: 0.4397 - val_accu
racy: 0.8907

Epoch 26/30

 - 120s - loss: 0.1735 - accuracy: 0.9305 - val_loss: 0.3793 - val_accu
racy: 0.9087

Epoch 27/30

 - 120s - loss: 0.1711 - accuracy: 0.9300 - val_loss: 0.4333 - val_accu
racy: 0.8436

Epoch 28/30

 - 120s - loss: 0.1674 - accuracy: 0.9312 - val_loss: 0.4673 - val_accu
racy: 0.8728

Epoch 29/30

 - 120s - loss: 0.2365 - accuracy: 0.9151 - val_loss: 0.3795 - val_accu
racy: 0.8775

Epoch 30/30

 - 120s - loss: 0.2052 - accuracy: 0.9252 - val_loss: 0.6173 - val_accu
racy: 0.8354

Test accuracy:

0.835425853729248
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
Model: "sequential_17"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_48 (Dense)             (None, 128, 256)          2560
_________________________________________________________________
batch_normalization_64 (Batc (None, 128, 256)          1024
_________________________________________________________________
Dropout (Dropout)            (None, 128, 256)          0
_________________________________________________________________
dense_49 (Dense)             (None, 128, 256)          65792
_________________________________________________________________
batch_normalization_65 (Batc (None, 128, 256)          1024
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 256)          0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 128)          197120
_________________________________________________________________
batch_normalization_66 (Batc (None, 128, 128)          512
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
LSTM_2 (LSTM)                (None, 64)                49408
_________________________________________________________________
batch_normalization_67 (Batc (None, 64)                256
_________________________________________________________________
Dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_50 (Dense)             (None, 6)                 390
=================================================================
Total params: 318,086
Trainable params: 316,678
Non-trainable params: 1,408
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
 - 104s - loss: 0.9169 - accuracy: 0.6375 - val_loss: 2.3134 - val_accu
racy: 0.3315
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 2/30

 - 100s - loss: 0.5026 - accuracy: 0.8088 - val_loss: 2.3230 - val_accu
racy: 0.4038

Epoch 3/30

 - 101s - loss: 0.3327 - accuracy: 0.8753 - val_loss: 1.8933 - val_accu
racy: 0.4863

Epoch 4/30

 - 99s - loss: 0.2766 - accuracy: 0.8996 - val_loss: 0.8764 - val_accur
acy: 0.7564

Epoch 5/30

 - 97s - loss: 0.2141 - accuracy: 0.9184 - val_loss: 0.4924 - val_accur
acy: 0.8235

Epoch 6/30

 - 96s - loss: 0.2110 - accuracy: 0.9223 - val_loss: 0.5878 - val_accur
acy: 0.8117

Epoch 7/30

 - 97s - loss: 0.2096 - accuracy: 0.9223 - val_loss: 0.4382 - val_accur
acy: 0.8622

Epoch 8/30

 - 97s - loss: 0.2130 - accuracy: 0.9203 - val_loss: 0.3830 - val_accur
acy: 0.8449

Epoch 9/30

 - 97s - loss: 0.1990 - accuracy: 0.9233 - val_loss: 0.3609 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.8928

Epoch 10/30

 - 97s - loss: 0.1720 - accuracy: 0.9327 - val_loss: 0.5327 - val_accur
acy: 0.8616

Epoch 11/30

 - 97s - loss: 0.1980 - accuracy: 0.9260 - val_loss: 0.4247 - val_accur
acy: 0.8531

Epoch 12/30

 - 97s - loss: 0.1958 - accuracy: 0.9276 - val_loss: 0.5640 - val_accur
acy: 0.8334

Epoch 13/30

 - 97s - loss: 0.1792 - accuracy: 0.9313 - val_loss: 0.3499 - val_accur
acy: 0.8941

Epoch 14/30

 - 97s - loss: 0.1930 - accuracy: 0.9257 - val_loss: 0.3528 - val_accur
acy: 0.8955

Epoch 15/30

 - 97s - loss: 0.1876 - accuracy: 0.9275 - val_loss: 0.3750 - val_accur
acy: 0.8772

Epoch 16/30

 - 98s - loss: 0.1924 - accuracy: 0.9280 - val_loss: 0.3948 - val_accur
acy: 0.8582

Epoch 17/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 105s - loss: 0.1904 - accuracy: 0.9300 - val_loss: 0.3014 - val_accu
racy: 0.8972

Epoch 18/30

 - 97s - loss: 0.1975 - accuracy: 0.9271 - val_loss: 0.2511 - val_accur
acy: 0.9074

Epoch 19/30

 - 96s - loss: 0.1889 - accuracy: 0.9301 - val_loss: 0.3066 - val_accur
acy: 0.8945

Epoch 20/30

 - 96s - loss: 0.2040 - accuracy: 0.9233 - val_loss: 0.3409 - val_accur
acy: 0.8470

Epoch 21/30

 - 97s - loss: 0.1889 - accuracy: 0.9264 - val_loss: 0.3192 - val_accur
acy: 0.8816

Epoch 22/30

 - 97s - loss: 0.2028 - accuracy: 0.9252 - val_loss: 0.4470 - val_accur
acy: 0.8453

Epoch 23/30

 - 96s - loss: 0.1957 - accuracy: 0.9279 - val_loss: 0.3055 - val_accur
acy: 0.8870

Epoch 24/30

 - 96s - loss: 0.1955 - accuracy: 0.9238 - val_loss: 0.5077 - val_accur
acy: 0.8429

Epoch 25/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 97s - loss: 0.1910 - accuracy: 0.9249 - val_loss: 0.3295 - val_accur
acy: 0.8778

Epoch 26/30

 - 96s - loss: 0.1986 - accuracy: 0.9234 - val_loss: 0.3408 - val_accur
acy: 0.8734

Epoch 27/30

 - 96s - loss: 0.2052 - accuracy: 0.9211 - val_loss: 0.4321 - val_accur
acy: 0.8347

Epoch 28/30

 - 96s - loss: 0.2102 - accuracy: 0.9215 - val_loss: 0.4980 - val_accur
acy: 0.8456

Epoch 29/30

 - 96s - loss: 0.2046 - accuracy: 0.9188 - val_loss: 0.3981 - val_accur
acy: 0.8432

Epoch 30/30

 - 96s - loss: 0.2141 - accuracy: 0.9144 - val_loss: 0.5753 - val_accur
acy: 0.8103

Test accuracy:

0.8103155493736267

---------------------------------------------------------------------
--------------
Model: "sequential_18"

_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_51 (Dense)             (None, 128, 64)           640
_________________________________________________________________
batch_normalization_68 (Batc (None, 128, 64)           256
_________________________________________________________________
Dropout (Dropout)            (None, 128, 64)           0
_________________________________________________________________
dense_52 (Dense)             (None, 128, 64)           4160
_________________________________________________________________
batch_normalization_69 (Batc (None, 128, 64)           256
_________________________________________________________________
Dropout_1 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
LSTM_1 (LSTM)                (None, 128, 64)           33024
_________________________________________________________________
batch_normalization_70 (Batc (None, 128, 64)           256
_________________________________________________________________
Dropout_2 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
LSTM_2 (LSTM)                (None, 32)                12416
_________________________________________________________________
batch_normalization_71 (Batc (None, 32)                128
_________________________________________________________________
Dropout_3 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_53 (Dense)             (None, 6)                 198
=================================================================
Total params: 51,334
Trainable params: 50,886
Non-trainable params: 448
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
 - 33s - loss: 0.9281 - accuracy: 0.5997 - val_loss: 1.2293 - val_accur
acy: 0.5246
Epoch 2/30
 - 29s - loss: 0.6403 - accuracy: 0.7318 - val_loss: 1.5132 - val_accur
acy: 0.5836
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 3/30

 - 27s - loss: 0.4483 - accuracy: 0.8341 - val_loss: 0.6758 - val_accur
acy: 0.7543

Epoch 4/30

 - 27s - loss: 0.3683 - accuracy: 0.8653 - val_loss: 0.9401 - val_accur
acy: 0.7072

Epoch 5/30

 - 27s - loss: 0.3469 - accuracy: 0.8815 - val_loss: 0.3555 - val_accur
acy: 0.8636

Epoch 6/30

 - 28s - loss: 0.2819 - accuracy: 0.8996 - val_loss: 0.9110 - val_accur
acy: 0.7635

Epoch 7/30

 - 27s - loss: 0.2603 - accuracy: 0.9076 - val_loss: 0.4552 - val_accur
acy: 0.8592

Epoch 8/30

 - 27s - loss: 0.2505 - accuracy: 0.9093 - val_loss: 0.5853 - val_accur
acy: 0.8524

Epoch 9/30

 - 27s - loss: 0.2322 - accuracy: 0.9165 - val_loss: 0.5734 - val_accur
acy: 0.8476

Epoch 10/30

 - 27s - loss: 0.2469 - accuracy: 0.9192 - val_loss: 0.4970 - val_accur
acy: 0.8843
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 11/30

 - 27s - loss: 0.2608 - accuracy: 0.9112 - val_loss: 0.4827 - val_accur
acy: 0.8755

Epoch 12/30

 - 27s - loss: 0.2353 - accuracy: 0.9180 - val_loss: 0.4069 - val_accur
acy: 0.8996

Epoch 13/30

 - 27s - loss: 0.2126 - accuracy: 0.9236 - val_loss: 0.6712 - val_accur
acy: 0.8269

Epoch 14/30

 - 27s - loss: 0.2372 - accuracy: 0.9158 - val_loss: 0.5828 - val_accur
acy: 0.8412

Epoch 15/30

 - 27s - loss: 0.2219 - accuracy: 0.9219 - val_loss: 0.4656 - val_accur
acy: 0.8626

Epoch 16/30

 - 27s - loss: 0.1962 - accuracy: 0.9301 - val_loss: 0.5326 - val_accur
acy: 0.8877

Epoch 17/30

 - 27s - loss: 0.2079 - accuracy: 0.9234 - val_loss: 0.5640 - val_accur
acy: 0.8259

Epoch 18/30

 - 27s - loss: 0.2012 - accuracy: 0.9255 - val_loss: 0.3468 - val_accur
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acy: 0.8666

Epoch 19/30

 - 27s - loss: 0.2023 - accuracy: 0.9276 - val_loss: 0.6892 - val_accur
acy: 0.7574

Epoch 20/30

 - 27s - loss: 0.2180 - accuracy: 0.9221 - val_loss: 0.3766 - val_accur
acy: 0.8819

Epoch 21/30

 - 27s - loss: 0.1898 - accuracy: 0.9310 - val_loss: 1.0511 - val_accur
acy: 0.7550

Epoch 22/30

 - 27s - loss: 0.2278 - accuracy: 0.9193 - val_loss: 0.5079 - val_accur
acy: 0.8531

Epoch 23/30

 - 27s - loss: 0.2333 - accuracy: 0.9238 - val_loss: 0.3796 - val_accur
acy: 0.8629

Epoch 24/30

 - 27s - loss: 0.2025 - accuracy: 0.9282 - val_loss: 0.3455 - val_accur
acy: 0.8867

Epoch 25/30

 - 27s - loss: 0.2122 - accuracy: 0.9211 - val_loss: 0.3159 - val_accur
acy: 0.9046

Epoch 26/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 - 27s - loss: 0.1990 - accuracy: 0.9272 - val_loss: 0.6532 - val_accur
acy: 0.8293

Epoch 27/30

 - 27s - loss: 0.2111 - accuracy: 0.9261 - val_loss: 0.3620 - val_accur
acy: 0.8890

Epoch 28/30

 - 27s - loss: 0.2101 - accuracy: 0.9279 - val_loss: 0.5499 - val_accur
acy: 0.8514

Epoch 29/30

 - 27s - loss: 0.2213 - accuracy: 0.9226 - val_loss: 0.3256 - val_accur
acy: 0.8904

Epoch 30/30

 - 27s - loss: 0.1947 - accuracy: 0.9283 - val_loss: 0.3858 - val_accur
acy: 0.8721

Test accuracy:

0.8720732927322388

----------------------------------------------------------------------
--------------
100%|██████████| 15/15 [8:11:00<00:00, 1964.04s/trial, best loss: -0.92
70444512367249]


Time taken : 8:11:09.452154
```

7.2 Applying Convolution Networks

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

7.2.1 Model 4

7.2.1.1 Constructing the Neural Network

In [43]:
```
model4 = Sequential()
model4.add(Conv1D(64, kernel_size=3,
                  activation='relu', kernel_initializer='he_uniform',
                  input_shape=(128,9), padding='same'))
model4.add(Conv1D(64, kernel_size = 3, activation='relu',kernel_initial
izer='he_uniform', padding='same'))
model4.add(Conv1D(64, kernel_size = 3, activation='relu',kernel_initial
izer='he_uniform', padding='same'))
model4.add(BatchNormalization())
model4.add(MaxPooling1D(pool_size=2))
model4.add(Dropout(0.5))

model4.add(Conv1D(128, kernel_size = 5, activation='relu',kernel_initia
lizer='he_uniform', padding='same'))
model4.add(Conv1D(128, kernel_size = 5, activation='relu',kernel_initia
lizer='he_uniform', padding='same'))
model4.add(BatchNormalization())
model4.add(MaxPooling1D(pool_size=2))
model4.add(Dropout(0.5))

model4.add(Flatten())

model4.add(Dense(512, activation='relu',kernel_initializer='he_uniform'
))
model4.add(BatchNormalization())
model4.add(Dropout(0.5))

model4.add(Dense(256, activation='relu',kernel_initializer='he_uniform'
))
model4.add(BatchNormalization())
model4.add(Dropout(0.5))

model4.add(Dense(128, activation='relu',kernel_initializer='he_uniform'
))
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
model4.add(BatchNormalization())
model4.add(Dropout(0.5))

model4.add(Dense(64, activation='relu',kernel_initializer='he_uniform'
))
model4.add(BatchNormalization())
model4.add(Dropout(0.5))

model4.add(Dense(6, activation='softmax',kernel_initializer='he_unifor
m'))
model4.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 128, 64)           1792
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 128, 64)           12352
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 128, 64)           12352
_________________________________________________________________
batch_normalization_8 (Batch (None, 128, 64)           256
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 64, 64)            0
_________________________________________________________________
dropout_4 (Dropout)          (None, 64, 64)            0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 64, 128)           41088
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 64, 128)           82048
_________________________________________________________________
batch_normalization_9 (Batch (None, 64, 128)           512
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 32, 128)           0
_________________________________________________________________
dropout_5 (Dropout)          (None, 32, 128)           0
_________________________________________________________________
flatten (Flatten)            (None, 4096)              0
_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
dense_9 (Dense)              (None, 512)               2097664
_________________________________________________________________
batch_normalization_10 (Batc (None, 512)               2048
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_10 (Dense)             (None, 256)               131328
_________________________________________________________________
batch_normalization_11 (Batc (None, 256)               1024
_________________________________________________________________
dropout_7 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_11 (Dense)             (None, 128)               32896
_________________________________________________________________
batch_normalization_12 (Batc (None, 128)               512
_________________________________________________________________
dropout_8 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_12 (Dense)             (None, 64)                8256
_________________________________________________________________
batch_normalization_13 (Batc (None, 64)                256
_________________________________________________________________
dropout_9 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_13 (Dense)             (None, 6)                 390
=================================================================
Total params: 2,424,774
Trainable params: 2,422,470
Non-trainable params: 2,304
_________________________________________________________________
```

7.2.1.2 Running the Neural Network on Train & Test Datasets for 30 Epochs

In [45]:
```
#Refer :- https://github.com/tensorflow/tensorflow/issues/27023

import tensorflow.python.util.deprecation as deprecation
deprecation._PRINT_DEPRECATION_WARNINGS = False
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
#Compiling the Model
model4.compile(optimizer='adam', loss='categorical_crossentropy', metri
cs=['accuracy'])

#Training the Model
M4_history = model4.fit(X_train, Y_train, batch_size=256, epochs=epochs
, verbose=1, \
                        validation_data=(X_test, Y_test))
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 23s 3ms/sample - loss: 0.1
079 - acc: 0.9561 - val_loss: 0.5996 - val_acc: 0.8816
Epoch 2/30
7352/7352 [==============================] - 22s 3ms/sample - loss: 0.1
109 - acc: 0.9517 - val_loss: 0.7269 - val_acc: 0.8846
Epoch 3/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.1
048 - acc: 0.9538 - val_loss: 0.4832 - val_acc: 0.8843
Epoch 4/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
920 - acc: 0.9597 - val_loss: 0.4770 - val_acc: 0.9019
Epoch 5/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
846 - acc: 0.9607 - val_loss: 0.3924 - val_acc: 0.8928
Epoch 6/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
863 - acc: 0.9577 - val_loss: 0.6353 - val_acc: 0.8833
Epoch 7/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
962 - acc: 0.9561 - val_loss: 0.7429 - val_acc: 0.8890
Epoch 8/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
939 - acc: 0.9600 - val_loss: 0.4947 - val_acc: 0.9046
Epoch 9/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
864 - acc: 0.9601 - val_loss: 0.4944 - val_acc: 0.9182
Epoch 10/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
860 - acc: 0.9597 - val_loss: 0.5583 - val_acc: 0.8897
Epoch 11/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
922 - acc: 0.9544 - val_loss: 0.4044 - val_acc: 0.9135
Epoch 12/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
820 - acc: 0.9585 - val_loss: 0.3832 - val_acc: 0.9152
Epoch 13/30
7352/7352 [==============================] - 22s 3ms/sample - loss: 0.0
745 - acc: 0.9660 - val_loss: 0.4411 - val_acc: 0.9097
Epoch 14/30
7352/7352 [==============================] - 22s 3ms/sample - loss: 0.0
789 - acc: 0.9648 - val_loss: 0.3840 - val_acc: 0.9206
Epoch 15/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
734 - acc: 0.9650 - val_loss: 0.3535 - val_acc: 0.9230
Epoch 16/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
723 - acc: 0.9650 - val_loss: 0.3077 - val_acc: 0.9270
Epoch 17/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
730 - acc: 0.9653 - val_loss: 0.3412 - val_acc: 0.9206
Epoch 18/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
761 - acc: 0.9659 - val_loss: 0.3770 - val_acc: 0.9060
Epoch 19/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
779 - acc: 0.9644 - val_loss: 0.5569 - val_acc: 0.9121
Epoch 20/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
724 - acc: 0.9657 - val_loss: 0.4467 - val_acc: 0.9196
Epoch 21/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
756 - acc: 0.9657 - val_loss: 0.3818 - val_acc: 0.9199
Epoch 22/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
809 - acc: 0.9641 - val_loss: 0.4516 - val_acc: 0.9125
Epoch 23/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
764 - acc: 0.9661 - val_loss: 0.5267 - val_acc: 0.8924
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
704 - acc: 0.9661 - val_loss: 0.5267 - val_acc: 0.8924
Epoch 24/30

7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
709 - acc: 0.9671 - val_loss: 0.5129 - val_acc: 0.9040
Epoch 25/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
676 - acc: 0.9676 - val_loss: 0.4176 - val_acc: 0.9097
Epoch 26/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
642 - acc: 0.9689 - val_loss: 0.4538 - val_acc: 0.9104
Epoch 27/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
645 - acc: 0.9680 - val_loss: 0.4057 - val_acc: 0.9091
Epoch 28/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
641 - acc: 0.9705 - val_loss: 0.5524 - val_acc: 0.8996
Epoch 29/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
704 - acc: 0.9644 - val_loss: 0.5380 - val_acc: 0.9216
Epoch 30/30
7352/7352 [==============================] - 21s 3ms/sample - loss: 0.0
692 - acc: 0.9661 - val_loss: 0.5141 - val_acc: 0.8989
```

7.2.1.3 Number of Epochs vs Train Loss & Test Loss

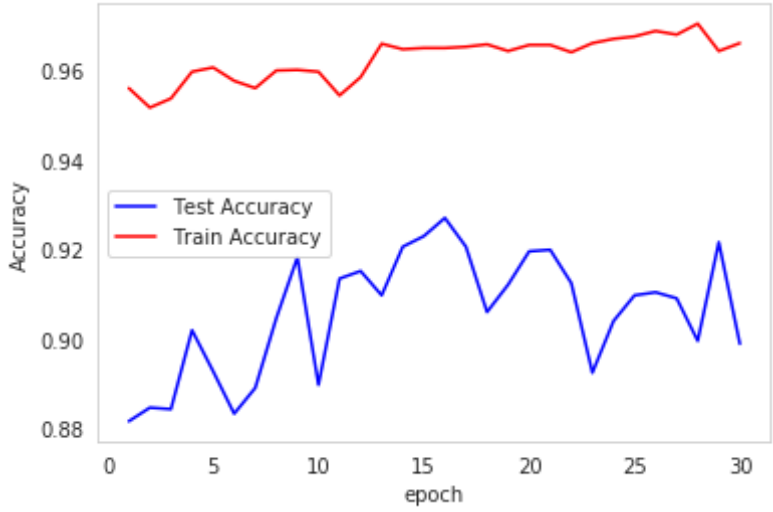
In [46]:

```
M4_score = model4.evaluate(X_test, Y_test, verbose=0)
print('Test Log Loss score:', M4_score[0])
print('Test accuracy:', M4_score[1])

fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Accuracy')

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

# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epo
chs=nb_epoch, verbose=1, validation_data=(X_test, Y_test))

# we will get val_loss and val_acc only when you pass the parameter val
idation_data
# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
# for each key in history.history we will have a list of length equal t
o number of epochs

vy = M4_history.history['val_acc']
ty = M4_history.history['acc']
plt_dynamic(x, vy, ty, ax)
```

```
Test Log Loss score: 0.5140607796788068
Test accuracy: 0.89888024
```



7.2.1.4 Normalized and Non-Normalized Confusion Matrices

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [47]:
```
#Plot a Non-Normalized confusion matrix
plot_lstm_confusion_matrix(model4, normalize=False)

#Plot a Normalized confusion matrix
plot_lstm_confusion_matrix(model4, normalize=True)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

7.2.2 Model 5 :-

7.2.2.1 Constructing the Neural Network

In [48]:
```
model5 = Sequential()
model5.add(Conv1D(64, kernel_size=3,
                  activation='relu', kernel_initializer='he_uniform',
                  input_shape=(128,9), padding='same'))
model5.add(Conv1D(64, kernel_size = 3, activation='relu',kernel_initial
izer='he_uniform', padding='same'))
model5.add(BatchNormalization())
model5.add(MaxPooling1D(pool_size=2))
model5.add(Dropout(0.5))

model5.add(Conv1D(128, kernel_size = 3, activation='relu',kernel_initia
lizer='he_uniform', padding='same'))
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
model5.add(Conv1D(128, kernel_size = 3, activation='relu',kernel_initia
lizer='he_uniform', padding='same'))
model5.add(BatchNormalization())
model5.add(MaxPooling1D(pool_size=2))
model5.add(Dropout(0.5))

model5.add(Conv1D(256, kernel_size = 3, activation='relu',kernel_initia
lizer='he_uniform', padding='same'))
model5.add(Conv1D(256, kernel_size = 3, activation='relu',kernel_initia
lizer='he_uniform', padding='same'))
model5.add(Conv1D(256, kernel_size = 3, activation='relu',kernel_initia
lizer='he_uniform', padding='same'))
model5.add(BatchNormalization())
model5.add(MaxPooling1D(pool_size=2))
model5.add(Dropout(0.5))

model5.add(Flatten())

model5.add(Dense(512, activation='relu',kernel_initializer='he_uniform'
))
model5.add(BatchNormalization())
model5.add(Dropout(0.5))

model5.add(Dense(256, activation='relu',kernel_initializer='he_uniform'
))
model5.add(BatchNormalization())
model5.add(Dropout(0.5))

model5.add(Dense(128, activation='relu',kernel_initializer='he_uniform'
))
model5.add(BatchNormalization())
model5.add(Dropout(0.5))

model5.add(Dense(64, activation='relu',kernel_initializer='he_uniform'
))
model5.add(BatchNormalization())
model5.add(Dropout(0.5))

model5.add(Dense(6, activation='softmax',kernel_initializer='he_unifor
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
m'))
model5.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_5 (Conv1D)            (None, 128, 64)           1792
_________________________________________________________________
conv1d_6 (Conv1D)            (None, 128, 64)           12352
_________________________________________________________________
batch_normalization_14 (Batc (None, 128, 64)           256
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 64, 64)            0
_________________________________________________________________
dropout_10 (Dropout)         (None, 64, 64)            0
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 64, 128)           24704
_________________________________________________________________
conv1d_8 (Conv1D)            (None, 64, 128)           49280
_________________________________________________________________
batch_normalization_15 (Batc (None, 64, 128)           512
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 32, 128)           0
_________________________________________________________________
dropout_11 (Dropout)         (None, 32, 128)           0
_________________________________________________________________
conv1d_9 (Conv1D)            (None, 32, 256)           98560
_________________________________________________________________
conv1d_10 (Conv1D)           (None, 32, 256)           196864
_________________________________________________________________
conv1d_11 (Conv1D)           (None, 32, 256)           196864
_________________________________________________________________
batch_normalization_16 (Batc (None, 32, 256)           1024
_________________________________________________________________
max_pooling1d_4 (MaxPooling1 (None, 16, 256)           0
_________________________________________________________________
dropout_12 (Dropout)         (None, 16, 256)           0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4096)              0
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
_________________________________________________________________
dense_14 (Dense)             (None, 512)               2097664
_________________________________________________________________
batch_normalization_17 (Batc (None, 512)               2048
_________________________________________________________________
dropout_13 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_15 (Dense)             (None, 256)               131328
_________________________________________________________________
batch_normalization_18 (Batc (None, 256)               1024
_________________________________________________________________
dropout_14 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_16 (Dense)             (None, 128)               32896
_________________________________________________________________
batch_normalization_19 (Batc (None, 128)               512
_________________________________________________________________
dropout_15 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_17 (Dense)             (None, 64)                8256
_________________________________________________________________
batch_normalization_20 (Batc (None, 64)                256
_________________________________________________________________
dropout_16 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_18 (Dense)             (None, 6)                 390
=================================================================
Total params: 2,856,582
Trainable params: 2,853,766
Non-trainable params: 2,816
_________________________________________________________________
```

7.2.2.2 Running the Neural Network on Train & Test Datasets for 30 Epochs

In [49]:
```
#Refer :- https://github.com/tensorflow/tensorflow/issues/27023

import tensorflow.python.util.deprecation as deprecation
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
deprecation._PRINT_DEPRECATION_WARNINGS = False

#Compiling the Model
model5.compile(optimizer='rmsprop', loss='categorical_crossentropy', me
trics=['accuracy'])

#Training the Model
M5_history = model5.fit(X_train, Y_train, batch_size=256, epochs=epochs
, verbose=1, \
                        validation_data=(X_test, Y_test))
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 39s 5ms/sample - loss: 1.5
815 - acc: 0.4460 - val_loss: 1.6207 - val_acc: 0.3237
Epoch 2/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.8
923 - acc: 0.6495 - val_loss: 2.5891 - val_acc: 0.4886
Epoch 3/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.6
307 - acc: 0.7398 - val_loss: 4.1204 - val_acc: 0.3851
Epoch 4/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.4
044 - acc: 0.8527 - val_loss: 5.7371 - val_acc: 0.2715
Epoch 5/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.2
552 - acc: 0.9136 - val_loss: 5.4007 - val_acc: 0.3000
Epoch 6/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.1
986 - acc: 0.9285 - val_loss: 4.5647 - val_acc: 0.4774
Epoch 7/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.1
675 - acc: 0.9377 - val_loss: 2.3988 - val_acc: 0.6067
Epoch 8/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.1
527 - acc: 0.9426 - val_loss: 1.3095 - val_acc: 0.7574
Epoch 9/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.1
423 - acc: 0.9430 - val_loss: 0.8915 - val_acc: 0.8605
Epoch 10/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.1
245 - acc: 0.9470 - val_loss: 1.3812 - val_acc: 0.8032

Epoch 11/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.1
236 - acc: 0.9491 - val_loss: 0.9754 - val_acc: 0.8588
Epoch 12/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.1
201 - acc: 0.9483 - val_loss: 1.1711 - val_acc: 0.8314
Epoch 13/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.1
096 - acc: 0.9509 - val_loss: 0.5620 - val_acc: 0.8992
Epoch 14/30
7352/7352 [==============================] - 38s 5ms/sample - loss: 0.1
226 - acc: 0.9491 - val_loss: 0.7672 - val_acc: 0.8724
Epoch 15/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.1
093 - acc: 0.9520 - val_loss: 0.5120 - val_acc: 0.9074
Epoch 16/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.1
066 - acc: 0.9516 - val_loss: 0.4917 - val_acc: 0.9114
Epoch 17/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.0
950 - acc: 0.9569 - val_loss: 0.7914 - val_acc: 0.8717
Epoch 18/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.0
952 - acc: 0.9551 - val_loss: 0.5157 - val_acc: 0.9009
Epoch 19/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.0
944 - acc: 0.9587 - val_loss: 0.3936 - val_acc: 0.9308
Epoch 20/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.0
955 - acc: 0.9531 - val_loss: 0.4087 - val_acc: 0.9097
Epoch 21/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.0
920 - acc: 0.9580 - val_loss: 0.7550 - val_acc: 0.8741
Epoch 22/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.0
936 - acc: 0.9582 - val_loss: 0.5081 - val_acc: 0.9118
Epoch 23/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.0
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.0
932 - acc: 0.9535 - val_loss: 0.4546 - val_acc: 0.9179

Epoch 24/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.0
845 - acc: 0.9588 - val_loss: 0.3875 - val_acc: 0.9243
Epoch 25/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.0
877 - acc: 0.9580 - val_loss: 0.3574 - val_acc: 0.9172
Epoch 26/30
7352/7352 [==============================] - 37s 5ms/sample - loss: 0.0
873 - acc: 0.9572 - val_loss: 0.5605 - val_acc: 0.9070
Epoch 27/30
7352/7352 [==============================] - 39s 5ms/sample - loss: 0.0
817 - acc: 0.9585 - val_loss: 0.4872 - val_acc: 0.9226
Epoch 28/30
7352/7352 [==============================] - 59s 8ms/sample - loss: 0.0
796 - acc: 0.9608 - val_loss: 0.4925 - val_acc: 0.9233
Epoch 29/30
7352/7352 [==============================] - 75s 10ms/sample - loss: 0.
0822 - acc: 0.9578 - val_loss: 0.6398 - val_acc: 0.9067
Epoch 30/30
7352/7352 [==============================] - 75s 10ms/sample - loss: 0.
0923 - acc: 0.9584 - val_loss: 0.3668 - val_acc: 0.9162
```

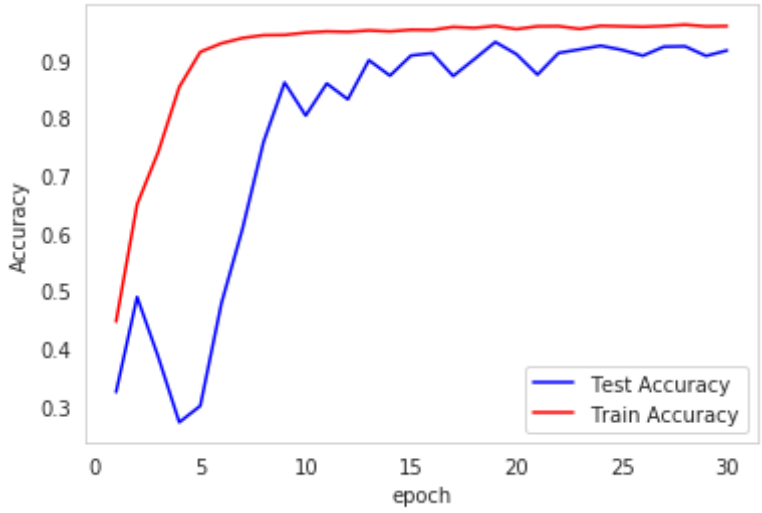
7.2.2.3 Number of Epochs vs Train Loss & Test Loss

In [50]:
```
M5_score = model5.evaluate(X_test, Y_test, verbose=0)
print('Test Log Loss score:', M5_score[0])
print('Test accuracy:', M5_score[1])

fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Accuracy')

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

# print(history.history.keys())
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epo
chs=nb_epoch, verbose=1, validation_data=(X_test, Y_test))

# we will get val_loss and val_acc only when you pass the parameter val
idation_data
# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
# for each key in history.history we will have a list of length equal t
o number of epochs

vy = M5_history.history['val_acc']
ty = M5_history.history['acc']
plt_dynamic(x, vy, ty, ax)
```

```
Test Log Loss score: 0.36675857328902295
Test accuracy: 0.916186
```



7.2.2.4 Normalized and Non-Normalized Confusion Matrices

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [51]:
```
#Plot a Non-Normalized confusion matrix
plot_lstm_confusion_matrix(model5, normalize=False)

#Plot a Normalized confusion matrix
plot_lstm_confusion_matrix(model5, normalize=True)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

8. Divide and Conquer CNN

Refer :- https://www.researchgate.net/publication/324224939_Divide_and_Conquer-Based_1D_CNN_Human_Activity_Recognition_Using_Test_Data_Sharpening

In the 'Divide and Conquer CNN' based approach we first build a CNN Model that predicts whether the activity is static or dynamic with a very high accuracy. This is followed by building models to separately predict the static and dynamic activity in consideration.

8.1 Classification into 2 Classes : Static and Dynamic Activities

8.1.1 Function to obtain Train and Test Data after Classifying Labels into Static and Dynamic Activities

In [52]:
```
def static_dynamic_classification():
    """
    Obtain the dataset from multiple files.
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    Returns: X_train, X_test, y_train, y_test
    """
    # Data directory
    DATADIR = 'UCI_HAR_Dataset'
    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"
        ]
    # 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'HAR/UCI_HAR_Dataset/{subset}/Inertial Signals/
{signal}_{subset}.txt'
            signals_data.append( _read_csv(filename).to_numpy())

        # 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 timeste
ps, 9 signals)
        return np.transpose(signals_data, (1, 2, 0))

    def load_y(subset):
        """
        The objective that we are trying to predict is a integer, from
 1 to 6,
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        that represents a human activity. We return a binary representa
tion of
        every sample objective as a 6 bits vector using One Hot Encodin
g
        (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.
get_dummies.html)
        """
        filename = f'HAR/UCI_HAR_Dataset/{subset}/y_{subset}.txt'
        y = _read_csv(filename)[0]
        y = (y>3).astype(int)

        return pd.get_dummies(y).to_numpy()

    X_train_2, X_test_2 = load_signals('train'), load_signals('test')
    Y_train_2, Y_test_2 = load_y('train'), load_y('test')

    return X_train_2, Y_train_2, X_test_2,  Y_test_2
```

In [62]:

```
X_train_2, Y_train_2, X_test_2, Y_test_2 = static_dynamic_classificatio
n()

print('Shape of the Training Data:')
print(" "*100)
print("Shape of X:" + str(X_train_2.shape))
print("Shape of Y: " + str(Y_train_2.shape))
print(" "*100)

print("-"*100)
print(" "*100)

print('Shape of the Test Data:')
print(" "*100)
print("Shape of X: " + str(X_test_2.shape))
print("Shape of Y: " + str(Y_test_2.shape))
```

```
Shape of the Training Data:


Shape of X:(7352, 128, 9)
Shape of Y: (7352, 2)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Shape of Y: (7352, 2)


--------------------------------------------------------------------------------
-----------------------------


Shape of the Test Data:


Shape of X: (2947, 128, 9)
Shape of Y: (2947, 2)
```

8.1.2 Model to Classify Labels into Static and Dynamic Activities

8.1.2.1 Constructing the Neural Network

In [56]:
```
model_class = Sequential()
model_class.add(Conv1D(64, kernel_size=3,
                 activation='relu',
                 input_shape=(128,9), padding='same'))
model_class.add(Conv1D(128, kernel_size=3, activation='relu'))
model_class.add(BatchNormalization())
model_class.add(MaxPooling1D(pool_size=2))
model_class.add(Dropout(0.3))

model_class.add(Conv1D(256, kernel_size=3, activation='relu'))
model_class.add(BatchNormalization())
model_class.add(MaxPooling1D(pool_size=2))

model_class.add(Flatten())
model_class.add(Dense(512, activation='relu',kernel_initializer='he_uni
form'))
model_class.add(BatchNormalization())
model_class.add(Dropout(0.4))
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
model_class.add(Dense(128, activation='relu',kernel_initializer='he_uni
form'))
model_class.add(BatchNormalization())
model_class.add(Dropout(0.4))

model_class.add(Dense(2, activation='softmax'))
model_class.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_12 (Conv1D)           (None, 128, 64)           1792
_________________________________________________________________
conv1d_13 (Conv1D)           (None, 126, 128)          24704
_________________________________________________________________
batch_normalization_21 (Batc (None, 126, 128)          512
_________________________________________________________________
max_pooling1d_5 (MaxPooling1 (None, 63, 128)           0
_________________________________________________________________
dropout_17 (Dropout)         (None, 63, 128)           0
_________________________________________________________________
conv1d_14 (Conv1D)           (None, 61, 256)           98560
_________________________________________________________________
batch_normalization_22 (Batc (None, 61, 256)           1024
_________________________________________________________________
max_pooling1d_6 (MaxPooling1 (None, 30, 256)           0
_________________________________________________________________
flatten_2 (Flatten)          (None, 7680)              0
_________________________________________________________________
dense_19 (Dense)             (None, 512)               3932672
_________________________________________________________________
batch_normalization_23 (Batc (None, 512)               2048
_________________________________________________________________
dropout_18 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_20 (Dense)             (None, 128)               65664
_________________________________________________________________
batch_normalization_24 (Batc (None, 128)               512
_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
dropout_19 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_21 (Dense)             (None, 2)                 258
=================================================================
Total params: 4,127,746
Trainable params: 4,125,698
Non-trainable params: 2,048
_________________________________________________________________
```

8.1.2.2 Running the Neural Network on Train & Test Datasets for 30 Epochs

In [57]:
```python
#Refer :- https://github.com/tensorflow/tensorflow/issues/27023

import tensorflow.python.util.deprecation as deprecation
deprecation._PRINT_DEPRECATION_WARNINGS = False

#Compiling the Model
model_class.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

#Training the Model
M_class_history = model_class.fit(X_train_2, Y_train_2, batch_size=256,
 epochs=30, verbose=1, \
                       validation_data=(X_test_2, Y_test_2))

#Saving the model for later use
model_class.save('HAR/UCI_HAR_Dataset/type_classification.h5')
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 27s 4ms/sample - loss: 0.0
635 - acc: 0.9743 - val_loss: 0.6187 - val_acc: 0.6854
Epoch 2/30
7352/7352 [==============================] - 25s 3ms/sample - loss: 0.0
016 - acc: 0.9996 - val_loss: 3.2048 - val_acc: 0.5294
Epoch 3/30
7352/7352 [==============================] - 26s 3ms/sample - loss: 8.9
205e-04 - acc: 0.9997 - val_loss: 4.2040 - val_acc: 0.5294
Epoch 4/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 4/30
7352/7352 [==============================] - 26s 3ms/sample - loss: 7.0
252e-04 - acc: 0.9999 - val_loss: 4.8429 - val_acc: 0.5294
Epoch 5/30
7352/7352 [==============================] - 26s 3ms/sample - loss: 0.0
014 - acc: 0.9997 - val_loss: 5.0709 - val_acc: 0.5294
Epoch 6/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 0.0
029 - acc: 0.9993 - val_loss: 5.7069 - val_acc: 0.5294
Epoch 7/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 0.0
025 - acc: 0.9995 - val_loss: 5.4646 - val_acc: 0.5294
Epoch 8/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 5.0
893e-04 - acc: 0.9999 - val_loss: 5.3402 - val_acc: 0.5294
Epoch 9/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 1.0
206e-04 - acc: 1.0000 - val_loss: 5.0594 - val_acc: 0.5294
Epoch 10/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 5.9
738e-04 - acc: 0.9997 - val_loss: 4.6989 - val_acc: 0.5294
Epoch 11/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 2.3
908e-04 - acc: 1.0000 - val_loss: 4.3466 - val_acc: 0.5297
Epoch 12/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 1.3
444e-04 - acc: 1.0000 - val_loss: 3.8980 - val_acc: 0.5378
Epoch 13/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 5.6
373e-05 - acc: 1.0000 - val_loss: 3.3642 - val_acc: 0.5582
Epoch 14/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 5.7
216e-05 - acc: 1.0000 - val_loss: 2.7236 - val_acc: 0.6033
Epoch 15/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 7.5
779e-05 - acc: 1.0000 - val_loss: 2.0493 - val_acc: 0.6627
Epoch 16/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 7.6
071e-05 - acc: 1.0000 - val_loss: 1.7998 - val_acc: 0.6960

Epoch 17/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 17/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 6.9
291e-05 - acc: 1.0000 - val_loss: 1.0240 - val_acc: 0.7923
Epoch 18/30
7352/7352 [==============================] - 27s 4ms/sample - loss: 3.1
942e-05 - acc: 1.0000 - val_loss: 0.6167 - val_acc: 0.8622
Epoch 19/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 2.7
310e-05 - acc: 1.0000 - val_loss: 0.3485 - val_acc: 0.9070
Epoch 20/30
7352/7352 [==============================] - 27s 4ms/sample - loss: 4.0
260e-05 - acc: 1.0000 - val_loss: 0.1788 - val_acc: 0.9450
Epoch 21/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 3.4
162e-05 - acc: 1.0000 - val_loss: 0.0944 - val_acc: 0.9698
Epoch 22/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 3.6
208e-05 - acc: 1.0000 - val_loss: 0.0599 - val_acc: 0.9783
Epoch 23/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 1.8
112e-05 - acc: 1.0000 - val_loss: 0.0267 - val_acc: 0.9878
Epoch 24/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 1.9
477e-05 - acc: 1.0000 - val_loss: 0.0136 - val_acc: 0.9939
Epoch 25/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 8.8
133e-05 - acc: 1.0000 - val_loss: 0.0378 - val_acc: 0.9847
Epoch 26/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 3.2
224e-05 - acc: 1.0000 - val_loss: 0.0186 - val_acc: 0.9912
Epoch 27/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 1.3
171e-05 - acc: 1.0000 - val_loss: 0.0091 - val_acc: 0.9956
Epoch 28/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 2.6
870e-05 - acc: 1.0000 - val_loss: 0.0030 - val_acc: 0.9997
Epoch 29/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 1.6
173e-05 - acc: 1.0000 - val_loss: 0.0024 - val_acc: 0.9993

Epoch 30/30
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

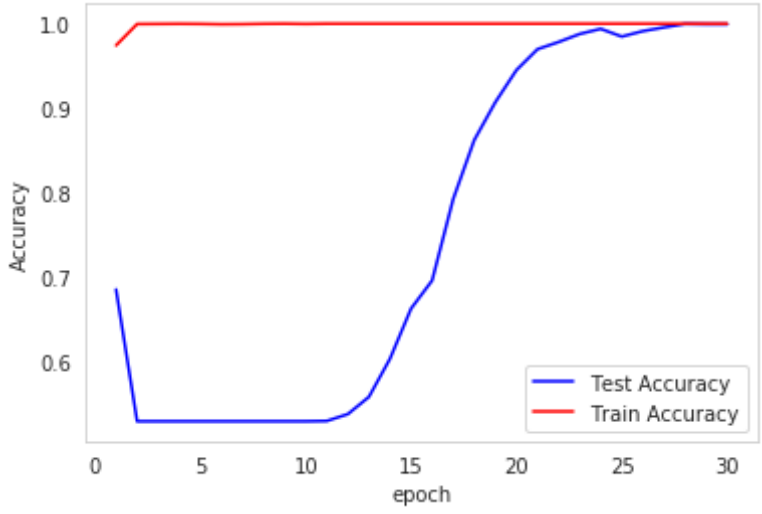
```
Epoch 30/30
7352/7352 [==============================] - 26s 4ms/sample - loss: 1.7
185e-05 - acc: 1.0000 - val_loss: 0.0020 - val_acc: 0.9993
```

8.1.2.3 Number of Epochs vs Train Loss & Test Loss

In [59]:
```
M_class_score = model_class.evaluate(X_test_2, Y_test_2, verbose=0)
print('Test Log Loss score:', M_class_score[0])
print('Test accuracy:', M_class_score[1])

fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Accuracy')

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

# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epo
chs=nb_epoch, verbose=1, validation_data=(X_test, Y_test))

# we will get val_loss and val_acc only when you pass the parameter val
idation_data
# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
# for each key in history.history we will have a list of length equal t
o number of epochs

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

```
Test Log Loss score: 0.001950174194822154
Test accuracy: 0.99932134
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



8.2 Classification into Static Activities

8.2.1 Function to obtain Train and Test Data for Static Activity Classification

In [60]:
```
def static_classification():
    """
    Obtain the dataset from multiple files.
    Returns: X_train_st, X_test_st, y_train_st, y_test_st
    """
    # Data directory
    DATADIR = 'UCI_HAR_Dataset'
    SIGNALS = [
        "body_acc_x",
        "body_acc_y",
        "body_acc_z",
        "body_gyro_x",
        "body_gyro_y",
        "body_gyro_z",
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        "total_acc_x",
        "total_acc_y",
        "total_acc_z"
        ]
    # 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'HAR/UCI_HAR_Dataset/{subset}/Inertial Signals/
{signal}_{subset}.txt'
            signals_data.append( _read_csv(filename).to_numpy())

        # 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 timeste
ps, 9 signals)
        return np.transpose(signals_data, (1, 2, 0))

    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 representa
tion of
        every sample objective as a 6 bits vector using One Hot Encodin
g
        (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.
get_dummies.html)
        """
        filename = f'HAR/UCI_HAR_Dataset/{subset}/y_{subset}.txt'
        y = _read_csv(filename)[0]
        y_filter = (y>3)
        y_final = y[y_filter]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        return pd.get_dummies(y_final).to_numpy(), y_filter


    Y_train_st, Y_train_filter = load_y('train')
    Y_test_st, Y_test_filter =   load_y('test')

    X_train_st, X_test_st = load_signals('train'), load_signals('test')
    X_train_st = X_train_st[Y_train_filter]
    X_test_st = X_test_st[Y_test_filter]

    return X_train_st, Y_train_st, X_test_st,  Y_test_st
```

In [64]:

```
X_train_st, Y_train_st, X_test_st, Y_test_st = static_classification()

print('Shape of the Training Data:')
print(" "*100)
print("Shape of X: " + str(X_train_st.shape))
print("Shape of Y: " + str(Y_train_st.shape))
print(" "*100)

print("-"*100)
print(" "*100)

print('Shape of the Test Data:')
print(" "*100)
print("Shape of X: " + str(X_test_st.shape))
print("Shape of Y: " + str(Y_test_st.shape))
```

```
Shape of the Training Data:


Shape of X: (4067, 128, 9)
Shape of Y: (4067, 3)


----------------------------------------------------------------------------------------------------
-----------------------------
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Shape of the Test Data:


Shape of X: (1560, 128, 9)
Shape of Y: (1560, 3)
```

8.2.2 Model to Classify Static Activities

8.2.2.1 Constructing the Neural Network

In [65]:
```
model_static = Sequential()

model_static.add(Conv1D(64, kernel_size=5,activation='relu', input_shap
e=(128,9)))
model_static.add(Conv1D(64, kernel_size=5, activation='relu'))
model_static.add(MaxPooling1D(pool_size=2))
model_static.add(Dropout(0.3))

model_static.add(Conv1D(256, kernel_size=3, activation='relu'))
model_static.add(MaxPooling1D(pool_size=2))
model_static.add(Dropout(0.3))

model_static.add(Flatten())
model_static.add(Dense(128, activation='relu'))
model_static.add(BatchNormalization())
model_static.add(Dropout(0.3))

model_static.add(Dense(64, activation='relu'))
model_static.add(BatchNormalization())
model_static.add(Dropout(0.3))

model_static.add(Dense(3, activation='softmax'))
model_static.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_15 (Conv1D)           (None, 124, 64)           2944
_________________________________________________________________
conv1d_16 (Conv1D)           (None, 120, 64)           20544
_________________________________________________________________
max_pooling1d_7 (MaxPooling1 (None, 60, 64)            0
_________________________________________________________________
dropout_20 (Dropout)         (None, 60, 64)            0
_________________________________________________________________
conv1d_17 (Conv1D)           (None, 58, 256)           49408
_________________________________________________________________
max_pooling1d_8 (MaxPooling1 (None, 29, 256)           0
_________________________________________________________________
dropout_21 (Dropout)         (None, 29, 256)           0
_________________________________________________________________
flatten_3 (Flatten)          (None, 7424)              0
_________________________________________________________________
dense_22 (Dense)             (None, 128)               950400
_________________________________________________________________
batch_normalization_25 (Batc (None, 128)               512
_________________________________________________________________
dropout_22 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_23 (Dense)             (None, 64)                8256
_________________________________________________________________
batch_normalization_26 (Batc (None, 64)                256
_________________________________________________________________
dropout_23 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_24 (Dense)             (None, 3)                 195
=================================================================
Total params: 1,032,515
Trainable params: 1,032,131
Non-trainable params: 384
_________________________________________________________________
```

8.2.2.2 Running the Neural Network on Train & Test Datasets for 30 Epochs

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [79]:
```
#Refer :- https://github.com/tensorflow/tensorflow/issues/27023

import tensorflow.python.util.deprecation as deprecation
deprecation._PRINT_DEPRECATION_WARNINGS = False

#Compiling the Model
model_static.compile(optimizer='adam', loss='categorical_crossentropy',
 metrics=['accuracy'])

#Training the Model
M_static_history = model_static.fit(X_train_st, Y_train_st, batch_size=
512, epochs=30, verbose=1, \
                       validation_data=(X_test_st, Y_test_st))

#Saving the model for later use
model_static.save('HAR/UCI_HAR_Dataset/static_classification.h5')
```

```
Train on 4067 samples, validate on 1560 samples
Epoch 1/30
4067/4067 [==============================] - 9s 2ms/sample - loss: 0.13
46 - acc: 0.9393 - val_loss: 0.2661 - val_acc: 0.8846
Epoch 2/30
4067/4067 [==============================] - 6s 2ms/sample - loss: 0.11
99 - acc: 0.9466 - val_loss: 0.2658 - val_acc: 0.8673
Epoch 3/30
4067/4067 [==============================] - 6s 1ms/sample - loss: 0.10
42 - acc: 0.9523 - val_loss: 0.3081 - val_acc: 0.8417
Epoch 4/30
4067/4067 [==============================] - 6s 1ms/sample - loss: 0.11
62 - acc: 0.9489 - val_loss: 0.3273 - val_acc: 0.8327
Epoch 5/30
4067/4067 [==============================] - 6s 2ms/sample - loss: 0.10
51 - acc: 0.9525 - val_loss: 0.4676 - val_acc: 0.7763
Epoch 6/30
4067/4067 [==============================] - 6s 1ms/sample - loss: 0.10
40 - acc: 0.9481 - val_loss: 0.3929 - val_acc: 0.7994
Epoch 7/30
4067/4067 [==============================] - 6s 2ms/sample - loss: 0.09
20 - acc: 0.9589 - val_loss: 0.3601 - val_acc: 0.8282
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 8/30
4067/4067 [==============================] - 6s 2ms/sample - loss: 0.09
21 - acc: 0.9557 - val_loss: 0.4140 - val_acc: 0.7949
Epoch 9/30
4067/4067 [==============================] - 6s 1ms/sample - loss: 0.08
28 - acc: 0.9631 - val_loss: 0.4628 - val_acc: 0.7904
Epoch 10/30
4067/4067 [==============================] - 6s 2ms/sample - loss: 0.08
59 - acc: 0.9604 - val_loss: 0.5036 - val_acc: 0.7942
Epoch 11/30
4067/4067 [==============================] - 6s 2ms/sample - loss: 0.07
74 - acc: 0.9671 - val_loss: 0.4936 - val_acc: 0.7795
Epoch 12/30
4067/4067 [==============================] - 6s 1ms/sample - loss: 0.08
71 - acc: 0.9612 - val_loss: 0.6404 - val_acc: 0.7288
Epoch 13/30
4067/4067 [==============================] - 6s 2ms/sample - loss: 0.07
97 - acc: 0.9648 - val_loss: 0.4167 - val_acc: 0.8263
Epoch 14/30
4067/4067 [==============================] - 6s 1ms/sample - loss: 0.09
13 - acc: 0.9587 - val_loss: 0.3763 - val_acc: 0.8801
Epoch 15/30
4067/4067 [==============================] - 6s 2ms/sample - loss: 0.10
61 - acc: 0.9540 - val_loss: 0.4689 - val_acc: 0.7788
Epoch 16/30
4067/4067 [==============================] - 6s 2ms/sample - loss: 0.08
55 - acc: 0.9624 - val_loss: 0.3015 - val_acc: 0.8744
Epoch 17/30
4067/4067 [==============================] - 6s 1ms/sample - loss: 0.07
42 - acc: 0.9673 - val_loss: 0.4067 - val_acc: 0.8314
Epoch 18/30
4067/4067 [==============================] - 6s 2ms/sample - loss: 0.07
82 - acc: 0.9634 - val_loss: 0.3221 - val_acc: 0.8647
Epoch 19/30
4067/4067 [==============================] - 6s 2ms/sample - loss: 0.07
05 - acc: 0.9688 - val_loss: 0.3336 - val_acc: 0.8641
Epoch 20/30
4067/4067 [==============================] - 6s 2ms/sample - loss: 0.06
99 - acc: 0.9693 - val_loss: 0.4232 - val_acc: 0.8122
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 21/30
4067/4067 [==============================] - 6s 2ms/sample - loss: 0.06
83 - acc: 0.9688 - val_loss: 0.3738 - val_acc: 0.8269
Epoch 22/30
4067/4067 [==============================] - 6s 1ms/sample - loss: 0.05
88 - acc: 0.9747 - val_loss: 0.3187 - val_acc: 0.8968
Epoch 23/30
4067/4067 [==============================] - 6s 2ms/sample - loss: 0.07
08 - acc: 0.9705 - val_loss: 0.4342 - val_acc: 0.8404
Epoch 24/30
4067/4067 [==============================] - 6s 2ms/sample - loss: 0.05
75 - acc: 0.9764 - val_loss: 0.4557 - val_acc: 0.8462
Epoch 25/30
4067/4067 [==============================] - 8s 2ms/sample - loss: 0.05
86 - acc: 0.9766 - val_loss: 0.4660 - val_acc: 0.8577
Epoch 26/30
4067/4067 [==============================] - 6s 2ms/sample - loss: 0.07
18 - acc: 0.9698 - val_loss: 0.4823 - val_acc: 0.8545
Epoch 27/30
4067/4067 [==============================] - 7s 2ms/sample - loss: 0.06
75 - acc: 0.9710 - val_loss: 0.6013 - val_acc: 0.8551
Epoch 28/30
4067/4067 [==============================] - 8s 2ms/sample - loss: 0.05
82 - acc: 0.9742 - val_loss: 0.5123 - val_acc: 0.8596
Epoch 29/30
4067/4067 [==============================] - 7s 2ms/sample - loss: 0.05
79 - acc: 0.9757 - val_loss: 0.4337 - val_acc: 0.8936
Epoch 30/30
4067/4067 [==============================] - 7s 2ms/sample - loss: 0.04
84 - acc: 0.9806 - val_loss: 0.3848 - val_acc: 0.9077
```

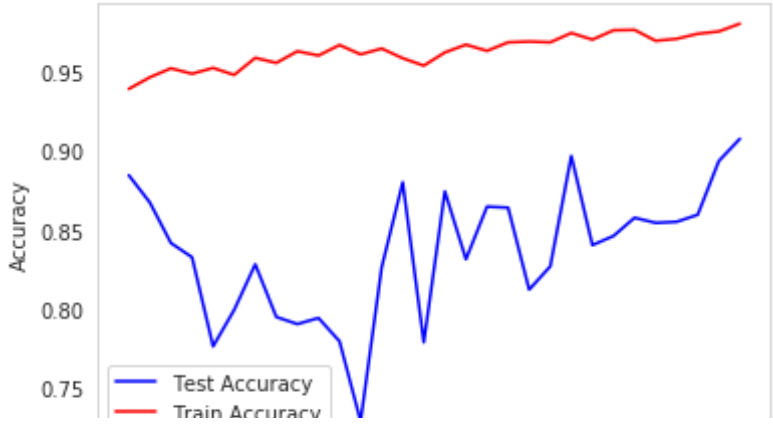
8.2.2.3 Number of Epochs vs Train Loss & Test Loss

In [80]:
```
M_static_score = model_static.evaluate(X_test_st, Y_test_st, verbose=0)

print('Test Log Loss score:', M_static_score[0])
print('Test accuracy:', M_static_score[1])
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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

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

# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epo
chs=nb_epoch, verbose=1, validation_data=(X_test, Y_test))

# we will get val_loss and val_acc only when you pass the parameter val
idation_data
# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
# for each key in history.history we will have a list of length equal t
o number of epochs

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

```
Test Log Loss score: 0.38475170127199726
Test accuracy: 0.9076923
```



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

8.3 Classification into Dynamic Activities

8.3.1 Function to obtain Train and Test Data for Dynamic Activity Classification

In [69]:
```
def dynamic_classification():
    """
    Obtain the dataset from multiple files.
    Returns: X_train, X_test, y_train, y_test
    """
    # Data directory
    DATADIR = 'UCI_HAR_Dataset'
    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"
        ]
    # 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:
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
            filename = f'HAR/UCI_HAR_Dataset/{subset}/Inertial Signals/
{signal}_{subset}.txt'
            signals_data.append( _read_csv(filename).to_numpy())

        # 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 timeste
ps, 9 signals)
        return np.transpose(signals_data, (1, 2, 0))

    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 representa
tion of
        every sample objective as a 6 bits vector using One Hot Encodin
g
        (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.
get_dummies.html)
        """
        filename = f'HAR/UCI_HAR_Dataset/{subset}/y_{subset}.txt'
        y = _read_csv(filename)[0]
        y_filter = (y<=3)
        y_final = y[y_filter]

        return pd.get_dummies(y_final).to_numpy(), y_filter


    Y_train_dy, Y_train_filter = load_y('train')
    Y_test_dy, Y_test_filter = load_y('test')

    X_train_dy, X_test_dy = load_signals('train'), load_signals('test')
    X_train_dy = X_train_dy[Y_train_filter]
    X_test_dy = X_test_dy[Y_test_filter]

    return X_train_dy, Y_train_dy, X_test_dy,  Y_test_dy
```

In [71]:

```
X_train_dy, Y_train_dy, X_test_dy, Y_test_dy = dynamic_classification()
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
print('Shape of the Training Data:')
print(" "*100)
print("Shape of X: " + str(X_train_dy.shape))
print("Shape of Y: " + str(Y_train_dy.shape))
print(" "*100)

print("-"*100)
print(" "*100)

print('Shape of the Test Data:')
print(" "*100)
print("Shape of X: " + str(X_test_dy.shape))
print("Shape of Y: " + str(Y_test_dy.shape))
```

```
Shape of the Training Data:


Shape of X: (3285, 128, 9)
Shape of Y: (3285, 3)


----------------------------------------------------------------------------------------------------
----------------------------


Shape of the Test Data:


Shape of X: (1387, 128, 9)
Shape of Y: (1387, 3)
```

8.3.2 Model to Classify Dynamic Activities

8.3.2.1 Constructing the Neural Network

In [72]:
```
model_dynamic = Sequential()
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
model_dynamic.add(Conv1D(64, kernel_size=3,
                 activation='relu', kernel_initializer='he_uniform',
                 input_shape=(128,9), padding='same'))
model_dynamic.add(Conv1D(64, kernel_size = 3, activation='relu',kernel_
initializer='he_uniform', padding='same'))
model_dynamic.add(BatchNormalization())
model_dynamic.add(MaxPooling1D(pool_size=2))
model_dynamic.add(Dropout(0.5))

model_dynamic.add(Conv1D(128, kernel_size = 3, activation='relu',kernel
_initializer='he_uniform', padding='same'))
model_dynamic.add(Conv1D(128, kernel_size = 3, activation='relu',kernel
_initializer='he_uniform', padding='same'))
model_dynamic.add(BatchNormalization())
model_dynamic.add(MaxPooling1D(pool_size=2))
model_dynamic.add(Dropout(0.5))

model_dynamic.add(Conv1D(256, kernel_size = 3, activation='relu',kernel
_initializer='he_uniform', padding='same'))
model_dynamic.add(Conv1D(256, kernel_size = 3, activation='relu',kernel
_initializer='he_uniform', padding='same'))
model_dynamic.add(Conv1D(256, kernel_size = 3, activation='relu',kernel
_initializer='he_uniform', padding='same'))
model_dynamic.add(BatchNormalization())
model_dynamic.add(MaxPooling1D(pool_size=2))
model_dynamic.add(Dropout(0.5))

model_dynamic.add(Flatten())

model_dynamic.add(Dense(512, activation='relu',kernel_initializer='he_u
niform'))
model_dynamic.add(BatchNormalization())
model_dynamic.add(Dropout(0.5))

model_dynamic.add(Dense(256, activation='relu',kernel_initializer='he_u
niform'))
model_dynamic.add(BatchNormalization())
model_dynamic.add(Dropout(0.5))
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
model_dynamic.add(Dense(128, activation='relu',kernel_initializer='he_u
niform'))
model_dynamic.add(BatchNormalization())
model_dynamic.add(Dropout(0.5))

model_dynamic.add(Dense(64, activation='relu',kernel_initializer='he_un
iform'))
model_dynamic.add(BatchNormalization())
model_dynamic.add(Dropout(0.5))

model_dynamic.add(Dense(3, activation='softmax',kernel_initializer='he_
uniform'))
model_dynamic.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_18 (Conv1D)           (None, 128, 64)           1792
_________________________________________________________________
conv1d_19 (Conv1D)           (None, 128, 64)           12352
_________________________________________________________________
batch_normalization_27 (Batc (None, 128, 64)           256
_________________________________________________________________
max_pooling1d_9 (MaxPooling1 (None, 64, 64)            0
_________________________________________________________________
dropout_24 (Dropout)         (None, 64, 64)            0
_________________________________________________________________
conv1d_20 (Conv1D)           (None, 64, 128)           24704
_________________________________________________________________
conv1d_21 (Conv1D)           (None, 64, 128)           49280
_________________________________________________________________
batch_normalization_28 (Batc (None, 64, 128)           512
_________________________________________________________________
max_pooling1d_10 (MaxPooling (None, 32, 128)           0
_________________________________________________________________
dropout_25 (Dropout)         (None, 32, 128)           0
_________________________________________________________________
conv1d_22 (Conv1D)           (None, 32, 256)           98560
_________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
conv1d_23 (Conv1D)           (None, 32, 256)           196864
_________________________________________________________________
conv1d_24 (Conv1D)           (None, 32, 256)           196864
_________________________________________________________________
batch_normalization_29 (Batc (None, 32, 256)           1024
_________________________________________________________________
max_pooling1d_11 (MaxPooling (None, 16, 256)           0
_________________________________________________________________
dropout_26 (Dropout)         (None, 16, 256)           0
_________________________________________________________________
flatten_4 (Flatten)          (None, 4096)              0
_________________________________________________________________
dense_25 (Dense)             (None, 512)               2097664
_________________________________________________________________
batch_normalization_30 (Batc (None, 512)               2048
_________________________________________________________________
dropout_27 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_26 (Dense)             (None, 256)               131328
_________________________________________________________________
batch_normalization_31 (Batc (None, 256)               1024
_________________________________________________________________
dropout_28 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_27 (Dense)             (None, 128)               32896
_________________________________________________________________
batch_normalization_32 (Batc (None, 128)               512
_________________________________________________________________
dropout_29 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_28 (Dense)             (None, 64)                8256
_________________________________________________________________
batch_normalization_33 (Batc (None, 64)                256
_________________________________________________________________
dropout_30 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_29 (Dense)             (None, 3)                 195
=================================================================
Total params: 2,856,387
Trainable params: 2,853,571
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Non-trainable params: 2,816
_________________________________________________________________
```

8.3.2.2 Running the Neural Network on Train & Test Datasets for 30 Epochs

In [73]:
```
#Refer :- https://github.com/tensorflow/tensorflow/issues/27023

import tensorflow.python.util.deprecation as deprecation
deprecation._PRINT_DEPRECATION_WARNINGS = False

#Compiling the Model
model_dynamic.compile(optimizer='rmsprop', loss='categorical_crossentro
py', metrics=['accuracy'])

#Training the Model
M_dynamic_history = model_dynamic.fit(X_train_dy, Y_train_dy, batch_siz
e=256, epochs=30, verbose=1, \
                       validation_data=(X_test_dy, Y_test_dy))

#Saving the model for later use
model_dynamic.save('HAR/UCI_HAR_Dataset/dynamic_classification.h5')
```

```
Train on 3285 samples, validate on 1387 samples
Epoch 1/30
3285/3285 [==============================] - 19s 6ms/sample - loss: 1.6
859 - acc: 0.3747 - val_loss: 1.5781 - val_acc: 0.3028
Epoch 2/30
3285/3285 [==============================] - 17s 5ms/sample - loss: 1.5
123 - acc: 0.3997 - val_loss: 1.1099 - val_acc: 0.4304
Epoch 3/30
3285/3285 [==============================] - 18s 5ms/sample - loss: 1.3
147 - acc: 0.4484 - val_loss: 1.6294 - val_acc: 0.4275
Epoch 4/30
3285/3285 [==============================] - 17s 5ms/sample - loss: 1.0
910 - acc: 0.5072 - val_loss: 2.7737 - val_acc: 0.3396
Epoch 5/30
3285/3285 [==============================] - 17s 5ms/sample - loss: 0.8
294 - acc: 0.6259 - val_loss: 3.7212 - val_acc: 0.3396
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 6/30
3285/3285 [==============================] - 17s 5ms/sample - loss: 0.6
087 - acc: 0.7242 - val_loss: 5.1330 - val_acc: 0.3396
Epoch 7/30
3285/3285 [==============================] - 17s 5ms/sample - loss: 0.3
578 - acc: 0.8612 - val_loss: 5.4871 - val_acc: 0.3403
Epoch 8/30
3285/3285 [==============================] - 17s 5ms/sample - loss: 0.2
268 - acc: 0.9291 - val_loss: 3.1848 - val_acc: 0.6446
Epoch 9/30
3285/3285 [==============================] - 16s 5ms/sample - loss: 0.1
150 - acc: 0.9650 - val_loss: 3.5875 - val_acc: 0.5169
Epoch 10/30
3285/3285 [==============================] - 16s 5ms/sample - loss: 0.0
826 - acc: 0.9760 - val_loss: 2.6668 - val_acc: 0.6842
Epoch 11/30
3285/3285 [==============================] - 16s 5ms/sample - loss: 0.0
411 - acc: 0.9909 - val_loss: 3.2185 - val_acc: 0.6691
Epoch 12/30
3285/3285 [==============================] - 20s 6ms/sample - loss: 0.0
393 - acc: 0.9896 - val_loss: 3.7678 - val_acc: 0.6683
Epoch 13/30
3285/3285 [==============================] - 18s 6ms/sample - loss: 0.0
221 - acc: 0.9951 - val_loss: 3.1886 - val_acc: 0.6799
Epoch 14/30
3285/3285 [==============================] - 17s 5ms/sample - loss: 0.0
286 - acc: 0.9915 - val_loss: 2.8483 - val_acc: 0.7123
Epoch 15/30
3285/3285 [==============================] - 17s 5ms/sample - loss: 0.0
175 - acc: 0.9945 - val_loss: 2.6982 - val_acc: 0.7037
Epoch 16/30
3285/3285 [==============================] - 19s 6ms/sample - loss: 0.0
111 - acc: 0.9963 - val_loss: 1.6320 - val_acc: 0.8097
Epoch 17/30
3285/3285 [==============================] - 31s 9ms/sample - loss: 0.0
169 - acc: 0.9963 - val_loss: 0.7250 - val_acc: 0.9048
Epoch 18/30
3285/3285 [==============================] - 28s 8ms/sample - loss: 0.0
143 - acc: 0.9957 - val_loss: 0.4526 - val_acc: 0.9387
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Epoch 19/30
3285/3285 [==============================] - 23s 7ms/sample - loss: 0.0
056 - acc: 0.9988 - val_loss: 0.6240 - val_acc: 0.9149
Epoch 20/30
3285/3285 [==============================] - 20s 6ms/sample - loss: 0.0
115 - acc: 0.9967 - val_loss: 0.1017 - val_acc: 0.9827
Epoch 21/30
3285/3285 [==============================] - 17s 5ms/sample - loss: 0.0
036 - acc: 0.9988 - val_loss: 0.0411 - val_acc: 0.9870
Epoch 22/30
3285/3285 [==============================] - 18s 6ms/sample - loss: 0.0
163 - acc: 0.9951 - val_loss: 0.1204 - val_acc: 0.9805
Epoch 23/30
3285/3285 [==============================] - 17s 5ms/sample - loss: 0.0
061 - acc: 0.9982 - val_loss: 0.3579 - val_acc: 0.9539
Epoch 24/30
3285/3285 [==============================] - 20s 6ms/sample - loss: 0.0
039 - acc: 0.9985 - val_loss: 0.2901 - val_acc: 0.9603
Epoch 25/30
3285/3285 [==============================] - 20s 6ms/sample - loss: 0.0
023 - acc: 0.9994 - val_loss: 0.2109 - val_acc: 0.9719
Epoch 26/30
3285/3285 [==============================] - 19s 6ms/sample - loss: 0.0
130 - acc: 0.9960 - val_loss: 0.2056 - val_acc: 0.9726
Epoch 27/30
3285/3285 [==============================] - 21s 6ms/sample - loss: 0.0
025 - acc: 0.9997 - val_loss: 0.0419 - val_acc: 0.9950
Epoch 28/30
3285/3285 [==============================] - 18s 5ms/sample - loss: 0.0
117 - acc: 0.9976 - val_loss: 0.0381 - val_acc: 0.9921
Epoch 29/30
3285/3285 [==============================] - 17s 5ms/sample - loss: 0.0
047 - acc: 0.9991 - val_loss: 0.2543 - val_acc: 0.9625
Epoch 30/30
3285/3285 [==============================] - 17s 5ms/sample - loss: 0.0
044 - acc: 0.9988 - val_loss: 0.2069 - val_acc: 0.9676
```

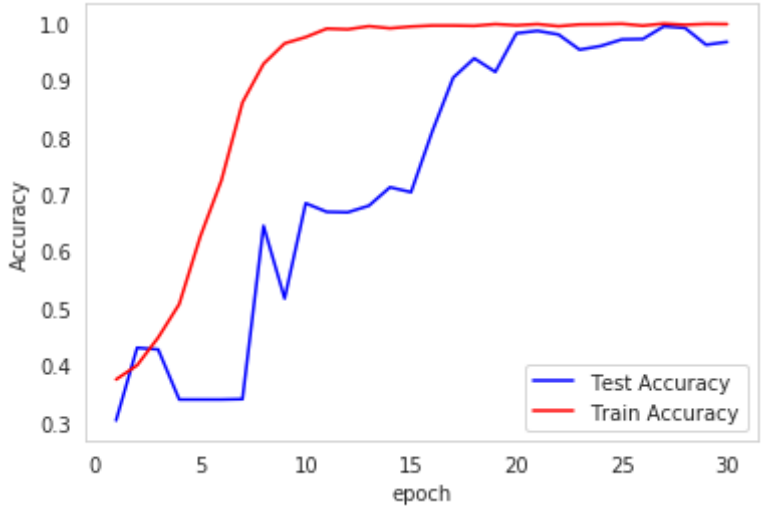
8.3.2.3 Number of Epochs vs Train Loss & Test Loss

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [74]:
```
M_dynamic_score = model_dynamic.evaluate(X_test_dy, Y_test_dy, verbose=
0)
print('Test Log Loss score:', M_dynamic_score[0])
print('Test accuracy:', M_dynamic_score[1])

fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Accuracy')

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

# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epo
chs=nb_epoch, verbose=1, validation_data=(X_test, Y_test))

# we will get val_loss and val_acc only when you pass the parameter val
idation_data
# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
# for each key in history.history we will have a list of length equal t
o number of epochs

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

```
Test Log Loss score: 0.20688932203663474
Test accuracy: 0.9675559
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



8.4 Combining final 'Divide & Conquer CNN' & Applying on Test Data

In [81]:
```
# Refer :- https://www.tensorflow.org/tutorials/keras/save_and_load

from tensorflow.keras.models import load_model

type_classifier = load_model('HAR/UCI_HAR_Dataset/type_classification.h
5')
static_model = load_model('HAR/UCI_HAR_Dataset/static_classification.h
5')
dynamic_model = load_model('HAR/UCI_HAR_Dataset/dynamic_classification.
h5')
```

In [82]:
```
# Refer :- https://machinelearningmastery.com/deep-learning-models-for-
human-activity-recognition/
# Refer :- http://www.mdpi.com/1424-8220/18/4/1055
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
type_predict = type_classifier.predict(X_test_2) #returns the class pro
babilities for individual classes
predicted_classes = np.argmax(type_predict, axis=1) #conversion of prob
abilities to actual class
actual_classes = Y_test_2.flatten() #flatten 2D array to 1D

static_classes = X_test_2[predicted_classes==1] #obtaining X features o
f Test Data for static
dynamic_classes = X_test_2[predicted_classes==0] #obtaining X features
 of Test Data for dynamic
```

In [83]:

```
static_prediction = static_model.predict(static_classes) #obtaining pro
babilities for the static classes
static_class_prediction = np.argmax(static_prediction,axis=1) #obtainin
g actual class labels
static_class_prediction = static_class_prediction + 4
#class labels obtained are from 0 to 3. Need to add 4 to obtain class l
abels from 4 to 6 (static class labels).

dynamic_prediction = dynamic_model.predict(dynamic_classes) #obtaining
 probabilities for the dynamic classes
dynamic_class_prediction = np.argmax(dynamic_prediction, axis=1) #obtai
ning actual class labels
dynamic_class_prediction = dynamic_class_prediction + 1
#class labels obtained are from 0 to 2. Need to add 1 to obtain class l
abels from 1 to 3 (dynamic class labels).

#obtaining class labels in the order of their occurrence
i,j = 0,0
final_predicted_classes = []
for label in predicted_classes:
    if label == 1:
        final_predicted_classes.append(static_class_prediction[i])
        i = i + 1
    elif label== 0:
        final_predicted_classes.append(dynamic_class_prediction[j])
        j = j + 1
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [84]:
```
final_predicted_classes = pd.get_dummies(final_predicted_classes).to_nu
mpy()
#one hot encoding of final predicted classes

from sklearn.metrics import accuracy_score
print('Accuracy of test data',accuracy_score(Y_test,final_predicted_cla
sses))  #final accuracy score
```

```
Accuracy of test data 0.9351883271123176
```

9. Conclusion

The Human Activity Recognition is a very popular application of modern times, especially with the advent of Smartphones as well as Smartwatches that have become extremely prevalent. This has very important applications such as the Health Tracking : if our device is able to determine the activity that we are performing, we can know the time that we spent on each activity and the total calories burnt during the day can also be estimated.

The data is recorded with the help of Accelerometers (that measure acceleration) as well as Gyroscopes (that measure angular velocity). We have a total of 6 time series datasets. Ours is a 6-class classification task for the 6 activities that we have. Since this is a Multiclass Classification Task, the metrics that we use are log-loss, accuracy and confusion matrices, with accuracy being the primary metric that we focus on.

In the dataset we carry out Overlapped Sampling (50% overlap), at the end of which we obtain 128 dimensional vectors, and we have a total of 561 Expert Engineered Features on which we employed various Machine Learning Models, and as we can see in the summary below, our linear models have performed much better than the tree based methodologies. However, at the same time, we see from the Confusion Matrix that there is a huge confusion between the 'Sitting' and 'Standing' classes.

After this, we work on the raw data where instead of 6 Time Series Datasets, we actually have a total of 9 Time Series Datasets, where 6 of these Time Series Datasets are for the Accelerometer Readings whereas 3 of them are for the Gyroscope Readings. At any Time t, we have 9 readings

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

ie. we have a vector of 9 values. Each window has 128 time steps. At every Time step, we have a total of 9 recordings, one corresponding to each of the Time Series.

Here, we work on the raw data and apply LSTM Modelling on top of this since we have a sequential Time series raw data. We begin by taking single LSTM layer and taking random values for the 'LSTM Units' and the 'Dropout Rate', with the help of which we train the models and we are able to find out that inspite of no expert engineered features, we are able to get a very healthy accuracy score of 89%. We are able to improve this slightly further with the help of Hyperparameter Tuning on the units and dropout rate. The same is followed with the help of CNN Models.

Since we have a limitation with the number of datapoints (only approx. 7K), we are not able to considerably improve the accuracy because we cannot build complex LSTM systems. But once again, we are able to see that there is still reasonable confusion between the 'Sitting' and 'Standing' Classes.

Finally we build a 'Divide and Conquer CNN' based approach where we first build a CNN Model that predicts whether the activity is static or dynamic with a very high accuracy. This is followed by building models to separately predict the static and dynamic activity in consideration. This is the best model that we obtain out of all the models built so far.

9.1 Metrics Obtained for ML Models obtained from Expert Engineered Features :

In [34]:
```
print('\n                    Accuracy     Error')
print('                  ----------   --------')
print('Logistic Regression : {:.04}%      {:.04}%'.format(log_reg_grid
_results['accuracy'] * 100,\
                                                 100-(log_reg_grid_res
ults['accuracy'] * 100)))

print('Linear SVC          : {:.04}%      {:.04}% '.format(lr_svc_grid
_results['accuracy'] * 100,\
                                                       100-(lr_svc_gri
d_results['accuracy'] * 100)))
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
print('rbf SVM classifier  : {:.04}%      {:.04}% '.format(rbf_svm_grid
_results['accuracy'] * 100,\
                                                          100-(rbf_svm_
grid_results['accuracy'] * 100)))

print('DecisionTree        : {:.04}%      {:.04}% '.format(dt_grid_resu
lts['accuracy'] * 100,\
                                                        100-(dt_grid_re
sults['accuracy'] * 100)))

print('Random Forest       : {:.04}%      {:.04}% '.format(rfc_grid_res
ults['accuracy'] * 100,\
                                                         100-(rfc_gri
d_results['accuracy'] * 100)))
print('GradientBoosting DT : {:.04}%      {:.04}% '.format(rfc_grid_res
ults['accuracy'] * 100,\
                                                        100-(rfc_grid_r
esults['accuracy'] * 100)))
```

```
                     Accuracy     Error
                     ----------   --------
Logistic Regression : 96.3%       3.699%
Linear SVC          : 96.4%       3.597%
rbf SVM classifier  : 96.27%      3.733%
DecisionTree        : 86.46%      13.54%
Random Forest       : 91.79%      8.212%
GradientBoosting DT : 91.79%      8.212%
```

Observations:

- Given our 561 Handcrafted/Expert Engineered Features, when we are using simple Linear Models of Logistic Regression and Linear SVC, we are getting a very good accuracy of 96.3% and 96.4% respectively.
- Since our Tree Based Models are not working very well, we can ignore the same.
- Since our models are getting confused between 'Sitting' and 'Standing', in the real world we can go to an Expert to enquire if he/she can design some special features to differentiate better between these 2 classes.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

We can choose ***Logistic regression*** or ***Linear SVC*** or ***rbf SVM***.

9.2 Metrics Obtained for LSTM Models obtained from Raw Data :

In [36]:
```
from prettytable import PrettyTable

x=PrettyTable()
x.field_names=["Model","Test Accuracy"]

print ("Accuracy for Deep Learning Models after 30 Epochs :")
print("="*100)

x.add_row(["Model 1","87.27%"])
x.add_row(["Model 2","90.73%"])
x.add_row(["Model 2 - After Tuning","91.75%"])
x.add_row(["Model 3","91.75%"])
x.add_row(["Model 3 - After Tuning","92.70%"])
x.add_row(["Model 4","89.88%"])
x.add_row(["Model 5","91.62%"])
x.add_row(["Divide & Conquer Based CNN","93.51%"])

print(x)
```

```
Accuracy for Deep Learning Models after 30 Epochs :
==========================================================================
=============================
+----------------------------+---------------+
|           Model            | Test Accuracy |
+----------------------------+---------------+
|          Model 1           |     87.27%    |
|          Model 2           |     90.73%    |
|   Model 2 - After Tuning   |     91.75%    |
|          Model 3           |     91.75%    |
|   Model 3 - After Tuning   |     92.70%    |
|          Model 4           |     89.88%    |
|          Model 5           |     91.62%    |
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
| Divide & Conquer Based CNN |     93.51%    |
+----------------------------+---------------+
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD