Human Activity Recongnition Using LSTM

This project is to build a model that predicts the human activities such as Walking, Walking_Upstairs, Walking_Downstairs, Sitting, Standing and Laying.

This dataset is collected from 30 persons(referred as subjects in this dataset), 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.

How data was recorded

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.

Feature names

1. 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.

1. From Each window, a feature vector was obtianed by calculating variables from the time and frequency domain.

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

2. 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***).

1. 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*.

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

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

   - tBodyAcc-XYZ
   - tGravityAcc-XYZ
   - tBodyAccJerk-XYZ
   - tBodyGyro-XYZ
   - tBodyGyroJerk-XYZ
   - tBodyAccMag
   - tGravityAccMag
   - tBodyAccJerkMag
   - tBodyGyroMag
   - tBodyGyroJerkMag
   - fBodyAcc-XYZ
   - fBodyAccJerk-XYZ
   - fBodyGyro-XYZ
   - fBodyAccMag
   - fBodyAccJerkMag
   - fBodyGyroMag
   - fBodyGyroJerkMag

1. 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
    - ***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.

1. 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 were taken as ***trianing data*** and remaining ***30%*** subjects recordings were taken for ***test data***

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'

Data Size :

https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones (https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones)

> 27 MB

Quick overview of the dataset :

- Accelerometer and Gyroscope readings are taken from 30 volunteers(referred as subjects) while performing the following 6 Activities.
  1. Walking
  2. WalkingUpstairs
  3. WalkingDownstairs
  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.

Problem Framework

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

Problem Statement (Objective):

- Given a new datapoint we have to predict the Activity

Importing all neccessary Libraries

In [0]:
```
import pandas as pd
import numpy as np
#from keras.models import Sequential
from keras.layers import LSTM
from keras import backend as K
#from keras.layers.core import Dense, Dropout
import pdb
from keras.layers.normalization import BatchNormalization
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
```

Loading data and Defining methods

In [26]:
```
# Load the Drive helper and mount
from google.colab import drive
# This will prompt for authorization.
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

In [0]:
```
#Loading from drive
#filtered_data=pd.read_csv('/content/drive/My Drive/Colab Notebooks/Reviews.csv')
#filtered_data=pd.read_csv('Reviews.csv')#displaying
#filtered_data.head()
#print(filtered_data.shape) #looking at the number of attributes and size of the data
#filtered_data.head()
```

In [0]:
```
#this function will draw graph in every epoch updates

# https://gist.github.com/greydanus/f6eee59eaf1d90fcb3b534a25362cea4
# https://stackoverflow.com/a/14434334
# this function is used to update the plots for each epoch and error
import matplotlib.pyplot as plt

def plt_dynamic(x, vy, ty, ax, colors=['b']):
  ax.plot(x, vy, 'b', label="Validation Loss")
  ax.plot(x, ty, 'r', label="Train Loss")
  plt.legend()
  plt.grid()
  fig.canvas.draw()
```

In [0]:
```
# Data directory
DATADIR = 'UCI_HAR_Dataset'
```

In [0]:
```
# Raw data signals
# Signals are from Accelerometer and Gyroscope
# The signals are in x,y,z directions
# Sensor signals are filtered to have only body acceleration
# excluding the acceleration due to gravity
# Triaxial acceleration from the accelerometer is total acceleration
SIGNALS = [
    "body_acc_x",
    "body_acc_y",
    "body_acc_z",
    "body_gyro_x",
    "body_gyro_y",
    "body_gyro_z",
    "total_acc_x",
    "total_acc_y",
    "total_acc_z"
]
```

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

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

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

In [0]:
```
# 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 = []
    #pdb.set_trace()
    for signal in SIGNALS:
        filename = f'/content/drive/My Drive/Colab Notebooks/UCI_HAR_Dataset/{subset}/Inertial Signals/{signal}_{subset}.txt'
        #UCI_HAR_Dataset/train/Inertial Signals/body_acc_x_train.txt-internal location where file is being placed.
        signals_data.append(
            _read_csv(filename).as_matrix()
        )
    #pdb.set_trace()
    # Transpose is used to change the dimensionality of the output,
    # aggregating the signals by combination of sample/timestep.
    # Resultant shape is (7352 train/2947 test samples, 128 timesteps, 9 signals)
    return np.transpose(signals_data, (1, 2, 0))
```

In [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 representation of
    every sample objective as a 6 bits vector using One Hot Encoding
    (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html)
    """
    #pdb.set_trace()
    filename = f'/content/drive/My Drive/Colab Notebooks/UCI_HAR_Dataset/{subset}/y_{subset}.txt'
    y = _read_csv(filename)[0]
    #pdb.set_trace()
    return pd.get_dummies(y).as_matrix()
```

In [0]:
```
def load_data():
    """
    Obtain the dataset from multiple files.
    Returns: X_train, X_test, y_train, y_test
    """
    #pdb.set_trace()
    X_train, X_test = load_signals('train'), load_signals('test')
    y_train, y_test = load_y('train'), load_y('test')
    #pdb.set_trace()
    return X_train, X_test, y_train, y_test
```

In [0]:
```
# Importing tensorflow
np.random.seed(42)
import tensorflow as tf
tf.set_random_seed(42)
```

In [0]:
```
# Configuring a session
session_conf = tf.ConfigProto(
    intra_op_parallelism_threads=1,
    inter_op_parallelism_threads=1
)
```

In [0]:
```
# Import Keras
from keras import backend as K
sess = tf.Session(graph=tf.get_default_graph(), config=session_conf)
K.set_session(sess)
```

In [0]:
```
# Utility function to count the number of classes
def _count_classes(y):
    return len(set([tuple(category) for category in y]))
```

In [0]:
```
# Loading the train and test data
#pdb.set_trace()
X_train, X_test, Y_train, Y_test = load_data()
#pdb.set_trace()
```

In [40]:
```
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))
print(n_classes)
```

```
128
9
7352
6
```

In [41]:
```
print(X_train.shape)
```

```
(7352, 128, 9)
```

LSTM with 3 layered Architecture

In [0]:
```
# Initializing parameters
epochs = 100
batch_size = 64
n_hidden_for_layer1 = 64
n_hidden_for_layer2 = 32
n_hidden_for_layer3 = 16
```

In [0]:
```
# Initiliazing the sequential model
model = Sequential()

# Configuring the parameters
model.add(LSTM(n_hidden_for_layer1,return_sequences=True ,input_shape=(timesteps, input_dim)))#layer 1
model.add(BatchNormalization())
model.add(Dropout(0.5))# Adding a dropout layer

model.add(LSTM(n_hidden_for_layer2, return_sequences=True))#layer 2
model.add(BatchNormalization())
model.add(Dropout(0.25))# Adding a dropout layer

model.add(LSTM(n_hidden_for_layer3))#layer 3
model.add(BatchNormalization())
model.add(Dropout(0.25))# Adding a dropout layer

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

model.summary()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecat
ed and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is dep
recated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_1 (LSTM)                (None, 128, 64)           18944     
_________________________________________________________________
batch_normalization_1 (Batch (None, 128, 64)           256       
_________________________________________________________________
dropout_1 (Dropout)          (None, 128, 64)           0         
_________________________________________________________________
lstm_2 (LSTM)                (None, 128, 32)           12416     
_________________________________________________________________
batch_normalization_2 (Batch (None, 128, 32)           128       
_________________________________________________________________
dropout_2 (Dropout)          (None, 128, 32)           0         
_________________________________________________________________
lstm_3 (LSTM)                (None, 16)                3136      
_________________________________________________________________
batch_normalization_3 (Batch (None, 16)                64        
_________________________________________________________________
dropout_3 (Dropout)          (None, 16)                0         
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 102       
=================================================================
Total params: 35,046
Trainable params: 34,822
Non-trainable params: 224
_________________________________________________________________
```

In [0]:
```
# Compiling the model
#import keras
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

In [0]:

```
%%time
# Training the model
history =model.fit(X_train,Y_train,batch_size=batch_size,epochs=epochs,verbose=1,validation_data=(X_test, Y_test))
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be re
moved in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 7352 samples, validate on 2947 samples
Epoch 1/100
7352/7352 [==============================] - 47s 6ms/step - loss: 1.2220 - acc: 0.6488 - val_loss: 0.9960 - val_acc: 0.6644
Epoch 2/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.8506 - acc: 0.7965 - val_loss: 0.8774 - val_acc: 0.7204
Epoch 3/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.6519 - acc: 0.7991 - val_loss: 0.6140 - val_acc: 0.7394
Epoch 4/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.5195 - acc: 0.7994 - val_loss: 0.8060 - val_acc: 0.6644
Epoch 5/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.4460 - acc: 0.7904 - val_loss: 1.0925 - val_acc: 0.5663
Epoch 6/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.3738 - acc: 0.7992 - val_loss: 0.3675 - val_acc: 0.7784
Epoch 7/100
7352/7352 [==============================] - 44s 6ms/step - loss: 0.3346 - acc: 0.8081 - val_loss: 0.4674 - val_acc: 0.7526
Epoch 8/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.3169 - acc: 0.8088 - val_loss: 0.3996 - val_acc: 0.7642
Epoch 9/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.3079 - acc: 0.8056 - val_loss: 0.4150 - val_acc: 0.7689
Epoch 10/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.2790 - acc: 0.8147 - val_loss: 0.3737 - val_acc: 0.7723
Epoch 11/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.2949 - acc: 0.8100 - val_loss: 0.8208 - val_acc: 0.7085
Epoch 12/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.2832 - acc: 0.8172 - val_loss: 0.3649 - val_acc: 0.7716
Epoch 13/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.2667 - acc: 0.8229 - val_loss: 0.4103 - val_acc: 0.7805
Epoch 14/100
7352/7352 [==============================] - 45s 6ms/step - loss: 0.2567 - acc: 0.8347 - val_loss: 0.4510 - val_acc: 0.7696
Epoch 15/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.2408 - acc: 0.8428 - val_loss: 0.4562 - val_acc: 0.7662
Epoch 16/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.2608 - acc: 0.8470 - val_loss: 0.5195 - val_acc: 0.7424
Epoch 17/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.2522 - acc: 0.8583 - val_loss: 0.4247 - val_acc: 0.8113
Epoch 18/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.2404 - acc: 0.8777 - val_loss: 0.4361 - val_acc: 0.8907
Epoch 19/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.2450 - acc: 0.8925 - val_loss: 0.4287 - val_acc: 0.9087
Epoch 20/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.2418 - acc: 0.9272 - val_loss: 0.4017 - val_acc: 0.8979
Epoch 21/100
7352/7352 [==============================] - 45s 6ms/step - loss: 0.2091 - acc: 0.9335 - val_loss: 0.3324 - val_acc: 0.8931
Epoch 22/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1654 - acc: 0.9418 - val_loss: 0.3531 - val_acc: 0.9152
Epoch 23/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1465 - acc: 0.9452 - val_loss: 0.4196 - val_acc: 0.8286
Epoch 24/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1630 - acc: 0.9357 - val_loss: 0.4674 - val_acc: 0.8836
Epoch 25/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1453 - acc: 0.9468 - val_loss: 0.3811 - val_acc: 0.9091
Epoch 26/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1384 - acc: 0.9464 - val_loss: 0.3058 - val_acc: 0.9186
Epoch 27/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1328 - acc: 0.9493 - val_loss: 0.3468 - val_acc: 0.9152
Epoch 28/100
7352/7352 [==============================] - 44s 6ms/step - loss: 0.1691 - acc: 0.9427 - val_loss: 0.5402 - val_acc: 0.8755
```

```
Epoch 29/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1407 - acc: 0.9472 - val_loss: 0.3543 - val_acc: 0.9067
Epoch 30/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1416 - acc: 0.9482 - val_loss: 0.3281 - val_acc: 0.9026
Epoch 31/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1395 - acc: 0.9457 - val_loss: 0.4422 - val_acc: 0.8802
Epoch 32/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1401 - acc: 0.9457 - val_loss: 0.3262 - val_acc: 0.9057
Epoch 33/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1344 - acc: 0.9498 - val_loss: 0.3248 - val_acc: 0.9165
Epoch 34/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1304 - acc: 0.9490 - val_loss: 0.2844 - val_acc: 0.9209
Epoch 35/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1292 - acc: 0.9494 - val_loss: 0.2849 - val_acc: 0.9308
Epoch 36/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1266 - acc: 0.9501 - val_loss: 0.3052 - val_acc: 0.9196
Epoch 37/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1283 - acc: 0.9476 - val_loss: 0.3068 - val_acc: 0.9175
Epoch 38/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1235 - acc: 0.9484 - val_loss: 0.4630 - val_acc: 0.8775
Epoch 39/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1305 - acc: 0.9461 - val_loss: 0.4033 - val_acc: 0.8992
Epoch 40/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1335 - acc: 0.9489 - val_loss: 0.3143 - val_acc: 0.9179
Epoch 41/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1242 - acc: 0.9482 - val_loss: 0.3159 - val_acc: 0.9077
Epoch 42/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1276 - acc: 0.9505 - val_loss: 0.3314 - val_acc: 0.9108
Epoch 43/100
7352/7352 [==============================] - 44s 6ms/step - loss: 0.1209 - acc: 0.9518 - val_loss: 0.2572 - val_acc: 0.9267
Epoch 44/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1150 - acc: 0.9489 - val_loss: 0.2631 - val_acc: 0.9233
Epoch 45/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1220 - acc: 0.9495 - val_loss: 0.3337 - val_acc: 0.9335
Epoch 46/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1094 - acc: 0.9533 - val_loss: 0.3418 - val_acc: 0.9257
Epoch 47/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1113 - acc: 0.9536 - val_loss: 0.3372 - val_acc: 0.9274
Epoch 48/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1192 - acc: 0.9512 - val_loss: 0.3249 - val_acc: 0.9287
Epoch 49/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1174 - acc: 0.9484 - val_loss: 0.4904 - val_acc: 0.9097
Epoch 50/100
7352/7352 [==============================] - 44s 6ms/step - loss: 0.1377 - acc: 0.9479 - val_loss: 0.6010 - val_acc: 0.8873
Epoch 51/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1355 - acc: 0.9490 - val_loss: 0.3324 - val_acc: 0.8924
Epoch 52/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1196 - acc: 0.9499 - val_loss: 0.3341 - val_acc: 0.9138
Epoch 53/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1187 - acc: 0.9547 - val_loss: 0.3608 - val_acc: 0.9125
Epoch 54/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1189 - acc: 0.9531 - val_loss: 0.2954 - val_acc: 0.9250
Epoch 55/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1088 - acc: 0.9548 - val_loss: 0.2934 - val_acc: 0.9335
Epoch 56/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1074 - acc: 0.9559 - val_loss: 0.3079 - val_acc: 0.9304
Epoch 57/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1090 - acc: 0.9543 - val_loss: 0.2689 - val_acc: 0.9321
Epoch 58/100
7352/7352 [==============================] - 44s 6ms/step - loss: 0.1235 - acc: 0.9506 - val_loss: 0.3420 - val_acc: 0.9016
Epoch 59/100
```

```
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1110 - acc: 0.9546 - val_loss: 0.3373 - val_acc: 0.9230
Epoch 60/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1079 - acc: 0.9555 - val_loss: 0.3966 - val_acc: 0.9287
Epoch 61/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1057 - acc: 0.9559 - val_loss: 0.4249 - val_acc: 0.9080
Epoch 62/100
7352/7352 [==============================] - 44s 6ms/step - loss: 0.1115 - acc: 0.9566 - val_loss: 0.6953 - val_acc: 0.8823
Epoch 63/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1168 - acc: 0.9550 - val_loss: 0.5512 - val_acc: 0.9108
Epoch 64/100
7352/7352 [==============================] - 44s 6ms/step - loss: 0.1008 - acc: 0.9591 - val_loss: 0.4559 - val_acc: 0.9175
Epoch 65/100
7352/7352 [==============================] - 44s 6ms/step - loss: 0.1068 - acc: 0.9572 - val_loss: 0.5545 - val_acc: 0.8951
Epoch 66/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1061 - acc: 0.9604 - val_loss: 0.3967 - val_acc: 0.9148
Epoch 67/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1073 - acc: 0.9535 - val_loss: 0.4156 - val_acc: 0.9250
Epoch 68/100
7352/7352 [==============================] - 44s 6ms/step - loss: 0.1077 - acc: 0.9555 - val_loss: 0.4406 - val_acc: 0.9125
Epoch 69/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1076 - acc: 0.9570 - val_loss: 0.4095 - val_acc: 0.9213
Epoch 70/100
7352/7352 [==============================] - 44s 6ms/step - loss: 0.1067 - acc: 0.9574 - val_loss: 0.5396 - val_acc: 0.9077
Epoch 71/100
7352/7352 [==============================] - 44s 6ms/step - loss: 0.1068 - acc: 0.9562 - val_loss: 0.4318 - val_acc: 0.9074
Epoch 72/100
7352/7352 [==============================] - 44s 6ms/step - loss: 0.0995 - acc: 0.9576 - val_loss: 0.4903 - val_acc: 0.9033
Epoch 73/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1031 - acc: 0.9581 - val_loss: 0.4881 - val_acc: 0.8911
Epoch 74/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1100 - acc: 0.9572 - val_loss: 0.3554 - val_acc: 0.9247
Epoch 75/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1050 - acc: 0.9577 - val_loss: 0.4240 - val_acc: 0.9084
Epoch 76/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1169 - acc: 0.9523 - val_loss: 0.3674 - val_acc: 0.9128
Epoch 77/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1003 - acc: 0.9569 - val_loss: 0.3702 - val_acc: 0.9182
Epoch 78/100
7352/7352 [==============================] - 44s 6ms/step - loss: 0.1141 - acc: 0.9553 - val_loss: 0.4252 - val_acc: 0.9040
Epoch 79/100
7352/7352 [==============================] - 44s 6ms/step - loss: 0.1113 - acc: 0.9539 - val_loss: 0.5124 - val_acc: 0.9152
Epoch 80/100
7352/7352 [==============================] - 44s 6ms/step - loss: 0.0950 - acc: 0.9614 - val_loss: 0.4883 - val_acc: 0.9260
Epoch 81/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1058 - acc: 0.9574 - val_loss: 0.4947 - val_acc: 0.9111
Epoch 82/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1020 - acc: 0.9577 - val_loss: 0.4666 - val_acc: 0.8751
Epoch 83/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.0997 - acc: 0.9570 - val_loss: 0.4091 - val_acc: 0.9192
Epoch 84/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1061 - acc: 0.9570 - val_loss: 0.4070 - val_acc: 0.8979
Epoch 85/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1019 - acc: 0.9565 - val_loss: 0.3342 - val_acc: 0.9250
Epoch 86/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1065 - acc: 0.9587 - val_loss: 0.5063 - val_acc: 0.8972
Epoch 87/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.0926 - acc: 0.9610 - val_loss: 0.4233 - val_acc: 0.9118
Epoch 88/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1198 - acc: 0.9542 - val_loss: 0.2938 - val_acc: 0.9386
Epoch 89/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1058 - acc: 0.9576 - val_loss: 0.3443 - val_acc: 0.9240
```

```
Epoch 90/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1008 - acc: 0.9585 - val_loss: 0.3751 - val_acc: 0.9209
Epoch 91/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1016 - acc: 0.9589 - val_loss: 0.3713 - val_acc: 0.9158
Epoch 92/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.0996 - acc: 0.9584 - val_loss: 0.3529 - val_acc: 0.9253
Epoch 93/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.0964 - acc: 0.9607 - val_loss: 0.3788 - val_acc: 0.9226
Epoch 94/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.0974 - acc: 0.9610 - val_loss: 0.2992 - val_acc: 0.9281
Epoch 95/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.0918 - acc: 0.9630 - val_loss: 0.3180 - val_acc: 0.9328
Epoch 96/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.0851 - acc: 0.9676 - val_loss: 0.3180 - val_acc: 0.9291
Epoch 97/100
7352/7352 [==============================] - 43s 6ms/step - loss: 0.0902 - acc: 0.9626 - val_loss: 0.3821 - val_acc: 0.9196
Epoch 98/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.0956 - acc: 0.9619 - val_loss: 0.4224 - val_acc: 0.9080
Epoch 99/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.0989 - acc: 0.9592 - val_loss: 0.4361 - val_acc: 0.9274
Epoch 100/100
7352/7352 [==============================] - 42s 6ms/step - loss: 0.0964 - acc: 0.9614 - val_loss: 0.3191 - val_acc: 0.9240
CPU times: user 1h 10min 26s, sys: 1min 10s, total: 1h 11min 36s
Wall time: 1h 11min 43s
```

In [0]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 537        0         0        0                   0
SITTING                  4      391        92        0                   1
STANDING                 0       82       450        0                   0
WALKING                  0        0         0      482                   0
WALKING_DOWNSTAIRS       1        0         0        3                 408
WALKING_UPSTAIRS         0        0         0       12                   4

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            3
STANDING                           0
WALKING                           14
WALKING_DOWNSTAIRS                 8
WALKING_UPSTAIRS                 455
```

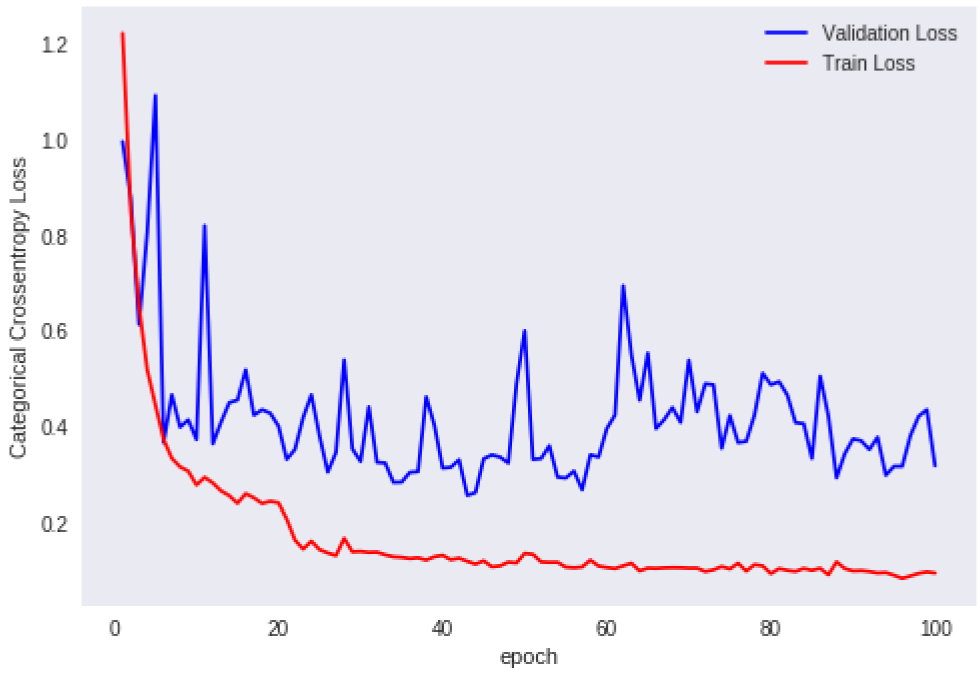
In [0]:
```
#ploting graph
score = model.evaluate(X_test, Y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

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

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

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

```
Test loss: 0.3190983775916234
Test accuracy: 0.9239904988123515
```



LSTM 2 layered with larger Dropouts

In [0]:
```
# Initializing parameters
epochs = 50
batch_size = 64
n_hidden_for_layer1 = 64
n_hidden_for_layer2 = 32
```

In [54]:

```
# Initiliazing the sequential model
model = Sequential()

# Configuring the parameters
model.add(LSTM(n_hidden_for_layer1,return_sequences=True ,input_shape=(timesteps, input_dim)))#layer 1
model.add(BatchNormalization())
model.add(Dropout(1))# Adding a dropout layer

model.add(LSTM(n_hidden_for_layer2))#layer 2
model.add(BatchNormalization())
model.add(Dropout(0.5))# Adding a dropout layer

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

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_9 (LSTM)                (None, 128, 64)           18944     
_________________________________________________________________
batch_normalization_5 (Batch (None, 128, 64)           256       
_________________________________________________________________
dropout_9 (Dropout)          (None, 128, 64)           0         
_________________________________________________________________
lstm_10 (LSTM)               (None, 32)                12416     
_________________________________________________________________
batch_normalization_6 (Batch (None, 32)                128       
_________________________________________________________________
dropout_10 (Dropout)         (None, 32)                0         
_________________________________________________________________
dense_5 (Dense)              (None, 6)                 198       
=================================================================
Total params: 31,942
Trainable params: 31,750
Non-trainable params: 192
_________________________________________________________________
```

In [0]:

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

In [56]:

```
%%time
# Training the model
history =model.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 1/50
7352/7352 [==============================] - 32s 4ms/step - loss: 1.2673 - acc: 0.5744 - val_loss: 0.9692 - val_acc: 0.6837
Epoch 2/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.9093 - acc: 0.7433 - val_loss: 0.9316 - val_acc: 0.5674
Epoch 3/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.7558 - acc: 0.7671 - val_loss: 0.6775 - val_acc: 0.7587
Epoch 4/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.6160 - acc: 0.7837 - val_loss: 1.1395 - val_acc: 0.5263
Epoch 5/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.5311 - acc: 0.7939 - val_loss: 0.4926 - val_acc: 0.7635
Epoch 6/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.4726 - acc: 0.7947 - val_loss: 0.5168 - val_acc: 0.7479
Epoch 7/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.4219 - acc: 0.7956 - val_loss: 0.4620 - val_acc: 0.7431
Epoch 8/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.3781 - acc: 0.7952 - val_loss: 0.4874 - val_acc: 0.7540
Epoch 9/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.3364 - acc: 0.8022 - val_loss: 0.4352 - val_acc: 0.7540
Epoch 10/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.3305 - acc: 0.8003 - val_loss: 0.3438 - val_acc: 0.7679
Epoch 11/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.3037 - acc: 0.8092 - val_loss: 0.5429 - val_acc: 0.7472
Epoch 12/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2822 - acc: 0.8161 - val_loss: 0.3391 - val_acc: 0.7686
Epoch 13/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.3041 - acc: 0.8120 - val_loss: 0.4169 - val_acc: 0.7615
Epoch 14/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2857 - acc: 0.8221 - val_loss: 0.3709 - val_acc: 0.7676
Epoch 15/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2802 - acc: 0.8278 - val_loss: 0.4536 - val_acc: 0.7482
Epoch 16/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.3054 - acc: 0.8195 - val_loss: 0.4373 - val_acc: 0.7679
Epoch 17/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2637 - acc: 0.8296 - val_loss: 0.4571 - val_acc: 0.7672
Epoch 18/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2600 - acc: 0.8354 - val_loss: 0.4131 - val_acc: 0.7682
Epoch 19/50
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2604 - acc: 0.8440 - val_loss: 0.3578 - val_acc: 0.7852
Epoch 20/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2552 - acc: 0.8453 - val_loss: 0.4017 - val_acc: 0.7822
Epoch 21/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2415 - acc: 0.8561 - val_loss: 0.4049 - val_acc: 0.7866
Epoch 22/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2513 - acc: 0.8562 - val_loss: 0.4077 - val_acc: 0.7720
Epoch 23/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2606 - acc: 0.8541 - val_loss: 0.3571 - val_acc: 0.7978
Epoch 24/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2493 - acc: 0.8594 - val_loss: 0.8355 - val_acc: 0.7251
Epoch 25/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2925 - acc: 0.8690 - val_loss: 0.4229 - val_acc: 0.8663
Epoch 26/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2512 - acc: 0.8823 - val_loss: 0.4078 - val_acc: 0.9030
Epoch 27/50
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2398 - acc: 0.9042 - val_loss: 0.3719 - val_acc: 0.9165
Epoch 28/50
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2132 - acc: 0.9287 - val_loss: 0.3070 - val_acc: 0.9114
Epoch 29/50
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1637 - acc: 0.9414 - val_loss: 0.2676 - val_acc: 0.9213
Epoch 30/50
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1497 - acc: 0.9457 - val_loss: 0.3102 - val_acc: 0.9209
```

```
Epoch 31/50
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1379 - acc: 0.9482 - val_loss: 0.3148 - val_acc: 0.9199
Epoch 32/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1420 - acc: 0.9478 - val_loss: 0.3738 - val_acc: 0.9046
Epoch 33/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1542 - acc: 0.9395 - val_loss: 0.4201 - val_acc: 0.8989
Epoch 34/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1413 - acc: 0.9441 - val_loss: 0.3564 - val_acc: 0.9196
Epoch 35/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1350 - acc: 0.9475 - val_loss: 0.3519 - val_acc: 0.9158
Epoch 36/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1382 - acc: 0.9460 - val_loss: 0.3891 - val_acc: 0.9189
Epoch 37/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1299 - acc: 0.9498 - val_loss: 0.3736 - val_acc: 0.9094
Epoch 38/50
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1332 - acc: 0.9489 - val_loss: 0.3137 - val_acc: 0.9203
Epoch 39/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1453 - acc: 0.9465 - val_loss: 0.5410 - val_acc: 0.8914
Epoch 40/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1469 - acc: 0.9421 - val_loss: 0.3153 - val_acc: 0.9162
Epoch 41/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1278 - acc: 0.9460 - val_loss: 0.3332 - val_acc: 0.9138
Epoch 42/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1287 - acc: 0.9472 - val_loss: 0.2859 - val_acc: 0.9311
Epoch 43/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1241 - acc: 0.9483 - val_loss: 0.3745 - val_acc: 0.9203
Epoch 44/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1388 - acc: 0.9455 - val_loss: 0.5768 - val_acc: 0.8809
Epoch 45/50
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2200 - acc: 0.9301 - val_loss: 0.5154 - val_acc: 0.8843
Epoch 46/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1601 - acc: 0.9418 - val_loss: 0.2830 - val_acc: 0.9125
Epoch 47/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1557 - acc: 0.9446 - val_loss: 0.2872 - val_acc: 0.9169
Epoch 48/50
7352/7352 [==============================] - 33s 4ms/step - loss: 0.1471 - acc: 0.9444 - val_loss: 0.2855 - val_acc: 0.9237
Epoch 49/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1328 - acc: 0.9441 - val_loss: 0.3225 - val_acc: 0.9030
Epoch 50/50
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1412 - acc: 0.9501 - val_loss: 0.2584 - val_acc: 0.9253
CPU times: user 23min 45s, sys: 28.2 s, total: 24min 14s
Wall time: 24min 27s
```

In [57]:
```python
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 537        0         0        0                   0
SITTING                  5      355       131        0                   0
STANDING                 0       45       487        0                   0
WALKING                  0        0         0      470                  22
WALKING_DOWNSTAIRS       0        0         0        0                 409
WALKING_UPSTAIRS         0        0         0        1                   1

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            0
STANDING                           0
WALKING                            4
WALKING_DOWNSTAIRS                11
WALKING_UPSTAIRS                 469
```

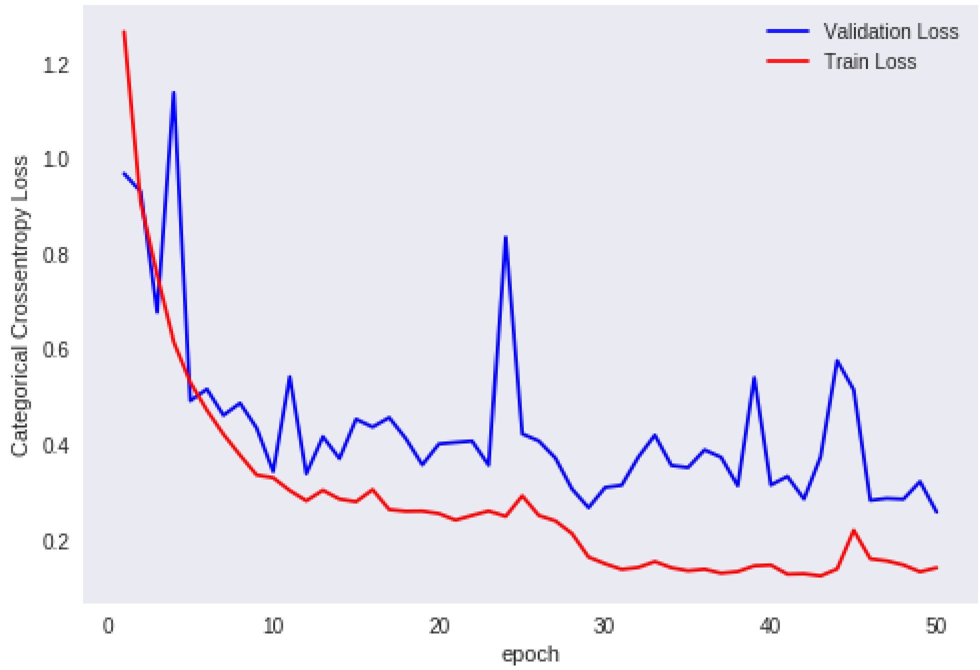
In [58]:
```python
#ploting graph
score = model.evaluate(X_test, Y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

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

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

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

```
Test loss: 0.25842093210849665
Test accuracy: 0.9253478113335596
```



LSTM 1 layered Architecture

In [0]:
```
# Initializing parameters
epochs = 75
batch_size = 64
n_hidden_for_layer1 = 128
```

In [71]:
```
# Initiliazing the sequential model
model = Sequential()

# Configuring the parameters
model.add(LSTM(n_hidden_for_layer1 ,input_shape=(timesteps, input_dim)))#Layer 1
model.add(BatchNormalization())
model.add(Dropout(2))# Adding a dropout layer

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

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_13 (LSTM)               (None, 128)               70656     
_________________________________________________________________
batch_normalization_9 (Batch (None, 128)               512       
_________________________________________________________________
dropout_13 (Dropout)         (None, 128)               0         
_________________________________________________________________
dense_8 (Dense)              (None, 6)                 774       
=================================================================
Total params: 71,942
Trainable params: 71,686
Non-trainable params: 256
_________________________________________________________________
```

In [0]:
```
# Compiling the model
#import keras
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

In [73]:

```
%%time
# Training the model
history =model.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 1/75
7352/7352 [==============================] - 32s 4ms/step - loss: 1.0788 - acc: 0.5306 - val_loss: 1.2120 - val_acc: 0.5860
Epoch 2/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.7887 - acc: 0.6085 - val_loss: 1.4075 - val_acc: 0.5524
Epoch 3/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.7283 - acc: 0.6117 - val_loss: 0.7540 - val_acc: 0.5914
Epoch 4/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.6892 - acc: 0.6291 - val_loss: 0.8100 - val_acc: 0.5772
Epoch 5/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.6373 - acc: 0.6542 - val_loss: 0.8774 - val_acc: 0.5931
Epoch 6/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.6372 - acc: 0.6425 - val_loss: 0.8003 - val_acc: 0.6210
Epoch 7/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.6181 - acc: 0.6538 - val_loss: 0.7382 - val_acc: 0.6542
Epoch 8/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.5776 - acc: 0.7062 - val_loss: 0.9943 - val_acc: 0.6644
Epoch 9/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.4412 - acc: 0.7924 - val_loss: 0.9051 - val_acc: 0.7469
Epoch 10/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2653 - acc: 0.8985 - val_loss: 0.4326 - val_acc: 0.8683
Epoch 11/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2630 - acc: 0.9095 - val_loss: 0.4133 - val_acc: 0.8551
Epoch 12/75
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2144 - acc: 0.9196 - val_loss: 0.2795 - val_acc: 0.9097
Epoch 13/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1470 - acc: 0.9406 - val_loss: 0.3585 - val_acc: 0.8816
Epoch 14/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1247 - acc: 0.9490 - val_loss: 0.2867 - val_acc: 0.9013
Epoch 15/75
7352/7352 [==============================] - 37s 5ms/step - loss: 0.1376 - acc: 0.9407 - val_loss: 0.3921 - val_acc: 0.8741
Epoch 16/75
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1291 - acc: 0.9437 - val_loss: 0.3318 - val_acc: 0.8907
Epoch 17/75
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1872 - acc: 0.9232 - val_loss: 0.2934 - val_acc: 0.9121
Epoch 18/75
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1246 - acc: 0.9438 - val_loss: 0.2751 - val_acc: 0.9077
Epoch 19/75
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1138 - acc: 0.9470 - val_loss: 0.2728 - val_acc: 0.9070
Epoch 20/75
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1147 - acc: 0.9457 - val_loss: 0.3236 - val_acc: 0.9101
Epoch 21/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1139 - acc: 0.9476 - val_loss: 0.3057 - val_acc: 0.9179
Epoch 22/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1098 - acc: 0.9470 - val_loss: 0.3130 - val_acc: 0.9125
Epoch 23/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1111 - acc: 0.9472 - val_loss: 0.3131 - val_acc: 0.9036
Epoch 24/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1519 - acc: 0.9336 - val_loss: 0.6967 - val_acc: 0.7530
Epoch 25/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1297 - acc: 0.9457 - val_loss: 0.3338 - val_acc: 0.8965
Epoch 26/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1192 - acc: 0.9494 - val_loss: 0.3244 - val_acc: 0.8938
Epoch 27/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1155 - acc: 0.9502 - val_loss: 0.2777 - val_acc: 0.9063
Epoch 28/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1125 - acc: 0.9484 - val_loss: 0.2952 - val_acc: 0.9091
Epoch 29/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1097 - acc: 0.9527 - val_loss: 0.3348 - val_acc: 0.8996
Epoch 30/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1124 - acc: 0.9455 - val_loss: 0.3149 - val_acc: 0.9128
```

```
Epoch 31/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1154 - acc: 0.9471 - val_loss: 0.3177 - val_acc: 0.9036
Epoch 32/75
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1067 - acc: 0.9512 - val_loss: 0.3517 - val_acc: 0.9002
Epoch 33/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1156 - acc: 0.9484 - val_loss: 0.4156 - val_acc: 0.8751
Epoch 34/75
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1290 - acc: 0.9440 - val_loss: 0.5976 - val_acc: 0.8422
Epoch 35/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1202 - acc: 0.9436 - val_loss: 0.2923 - val_acc: 0.9080
Epoch 36/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1084 - acc: 0.9474 - val_loss: 0.3141 - val_acc: 0.9060
Epoch 37/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1050 - acc: 0.9539 - val_loss: 0.3137 - val_acc: 0.9033
Epoch 38/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1015 - acc: 0.9512 - val_loss: 0.2806 - val_acc: 0.9108
Epoch 39/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.0995 - acc: 0.9540 - val_loss: 0.2894 - val_acc: 0.9206
Epoch 40/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.0980 - acc: 0.9555 - val_loss: 0.3010 - val_acc: 0.9162
Epoch 41/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.0989 - acc: 0.9561 - val_loss: 0.3148 - val_acc: 0.9070
Epoch 42/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.0974 - acc: 0.9563 - val_loss: 0.3332 - val_acc: 0.9175
Epoch 43/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1101 - acc: 0.9497 - val_loss: 0.3080 - val_acc: 0.9121
Epoch 44/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.0992 - acc: 0.9527 - val_loss: 0.4308 - val_acc: 0.9023
Epoch 45/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.0989 - acc: 0.9551 - val_loss: 0.3330 - val_acc: 0.9182
Epoch 46/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1090 - acc: 0.9470 - val_loss: 0.3295 - val_acc: 0.9036
Epoch 47/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1236 - acc: 0.9431 - val_loss: 0.5198 - val_acc: 0.8392
Epoch 48/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1079 - acc: 0.9504 - val_loss: 0.3464 - val_acc: 0.9162
Epoch 49/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1360 - acc: 0.9463 - val_loss: 0.2726 - val_acc: 0.9128
Epoch 50/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1142 - acc: 0.9566 - val_loss: 0.2987 - val_acc: 0.9155
Epoch 51/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1070 - acc: 0.9561 - val_loss: 0.2961 - val_acc: 0.9301
Epoch 52/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1076 - acc: 0.9548 - val_loss: 0.2890 - val_acc: 0.9260
Epoch 53/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1061 - acc: 0.9561 - val_loss: 0.2770 - val_acc: 0.9226
Epoch 54/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1049 - acc: 0.9527 - val_loss: 0.2842 - val_acc: 0.9094
Epoch 55/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1127 - acc: 0.9489 - val_loss: 0.2857 - val_acc: 0.9131
Epoch 56/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1116 - acc: 0.9476 - val_loss: 0.2922 - val_acc: 0.9114
Epoch 57/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1124 - acc: 0.9431 - val_loss: 0.3043 - val_acc: 0.9169
Epoch 58/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1101 - acc: 0.9501 - val_loss: 0.3142 - val_acc: 0.9111
Epoch 59/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1065 - acc: 0.9523 - val_loss: 0.3120 - val_acc: 0.9135
Epoch 60/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1041 - acc: 0.9501 - val_loss: 0.3123 - val_acc: 0.9128
Epoch 61/75
```

```
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1036 - acc: 0.9470 - val_loss: 0.3269 - val_acc: 0.9114
Epoch 62/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1157 - acc: 0.9459 - val_loss: 0.4796 - val_acc: 0.8714
Epoch 63/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1251 - acc: 0.9378 - val_loss: 0.3226 - val_acc: 0.9155
Epoch 64/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1088 - acc: 0.9438 - val_loss: 0.3497 - val_acc: 0.9084
Epoch 65/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1001 - acc: 0.9423 - val_loss: 0.3402 - val_acc: 0.9253
Epoch 66/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1023 - acc: 0.9476 - val_loss: 0.3360 - val_acc: 0.9165
Epoch 67/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.0983 - acc: 0.9489 - val_loss: 0.3449 - val_acc: 0.9172
Epoch 68/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.0993 - acc: 0.9512 - val_loss: 0.3678 - val_acc: 0.9213
Epoch 69/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.0998 - acc: 0.9512 - val_loss: 0.4213 - val_acc: 0.8975
Epoch 70/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.0971 - acc: 0.9558 - val_loss: 0.4250 - val_acc: 0.9135
Epoch 71/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1368 - acc: 0.9433 - val_loss: 0.3508 - val_acc: 0.9026
Epoch 72/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1123 - acc: 0.9448 - val_loss: 0.3017 - val_acc: 0.9070
Epoch 73/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1122 - acc: 0.9434 - val_loss: 0.3068 - val_acc: 0.9175
Epoch 74/75
7352/7352 [==============================] - 28s 4ms/step - loss: 0.0993 - acc: 0.9490 - val_loss: 0.3081 - val_acc: 0.9158
Epoch 75/75
7352/7352 [==============================] - 29s 4ms/step - loss: 0.0974 - acc: 0.9479 - val_loss: 0.3165 - val_acc: 0.9196
CPU times: user 35min 55s, sys: 52.1 s, total: 36min 47s
Wall time: 37min 6s
```

In [74]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 537        0         0        0                   0
SITTING                  4      358       126        0                   0
STANDING                 0       57       474        1                   0
WALKING                  0        0         0      467                  29
WALKING_DOWNSTAIRS       0        0         0        3                 413
WALKING_UPSTAIRS         0        0         2        3                   5

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            3
STANDING                           0
WALKING                            0
WALKING_DOWNSTAIRS                 4
WALKING_UPSTAIRS                 461
```

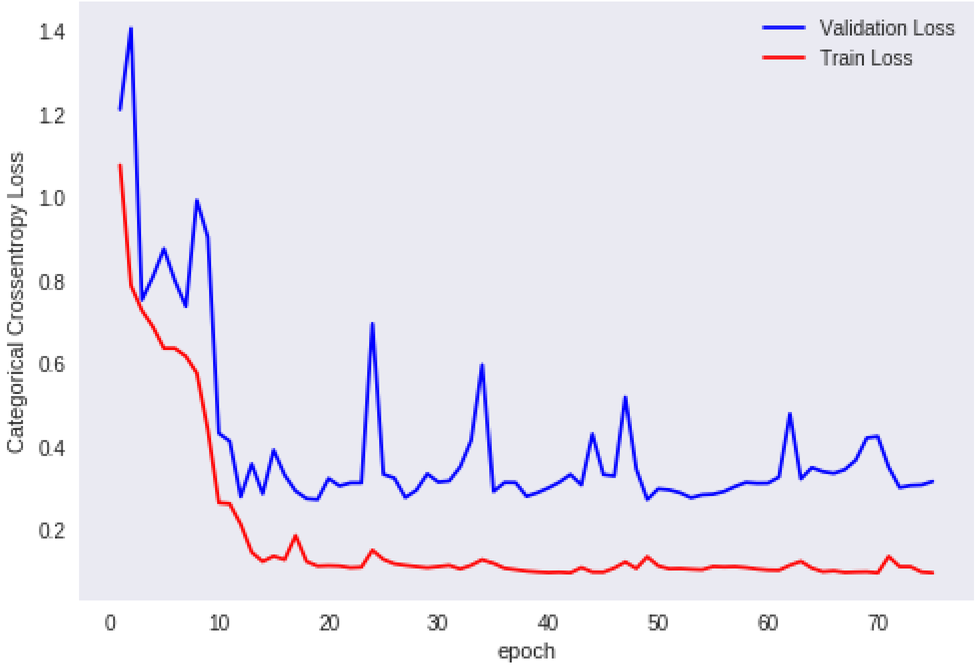
In [75]:
```
#ploting graph
score = model.evaluate(X_test, Y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

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

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

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

```
Test loss: 0.3164710504999816
Test accuracy: 0.9195792331184255
```



Conclusion :

Sr.no	No. Of LSTM Layers	Epochs	Optimiser	Accuracy	Loss
1.	3 (h1=64),(h2=32),(h3=16)	100	Adam	92.39%	31.9%
2.	2 (h1=64),(h2=32)	50	Adam	92.53%	25.84%
3.	1 (h1=64)	75	Adam	91.95%	31.64%

---xxx---