HAR_LSTM -HyperParameter Tuning

March 22, 2019

```
In [1]: # Importing Libraries

In [11]: import pandas as pd
         import numpy as np

In [12]: features = list()
         with open('UCI_HAR_Dataset/features.txt') as f:
             features = [line.split()[1] for line in f.readlines()]

In [13]: X_train = pd.read_csv('UCI_HAR_dataset/train/X_train.txt', delim_whitespace=True, head

         # add subject column to the dataframe
         X_train['subject'] = pd.read_csv('UCI_HAR_dataset/train/subject_train.txt', header=Nor

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

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

c:\users\dell\appdata\local\programs\python\python36\lib\site-packages\pandas\io\parsers.py:70
  return _read(filepath_or_buffer, kwds)


Out[13]:       tBodyAcc-mean()-X  tBodyAcc-mean()-Y  tBodyAcc-mean()-Z  \
         6142           0.277013          -0.017327          -0.108184

               tBodyAcc-std()-X  tBodyAcc-std()-Y  tBodyAcc-std()-Z  tBodyAcc-mad()-X  \
         6142         -0.988186          -0.98857          -0.99031         -0.988547

               tBodyAcc-mad()-Y  tBodyAcc-mad()-Z  tBodyAcc-max()-X  ...  \
         6142         -0.987859         -0.990558         -0.933673  ...

               angle(tBodyAccMean,gravity)  angle(tBodyAccJerkMean),gravityMean)  \
```

```
6142                       0.063378                                     -0.817828

      angle(tBodyGyroMean,gravityMean)  angle(tBodyGyroJerkMean,gravityMean)  \
6142                          0.155249                                     -0.121316

      angle(X,gravityMean)  angle(Y,gravityMean)  angle(Z,gravityMean)  \
6142              0.633324             -0.29935              -0.71443

      subject  Activity  ActivityName
6142       27         6        LAYING

[1 rows x 564 columns]
```

In [14]:
```
# get the data from txt files to pandas dataffame
X_test = pd.read_csv('UCI_HAR_dataset/test/X_test.txt', delim_whitespace=True, header=

# add subject column to the dataframe
X_test['subject'] = pd.read_csv('UCI_HAR_dataset/test/subject_test.txt', header=None,

# get y labels from the txt file
y_test = pd.read_csv('UCI_HAR_dataset/test/y_test.txt', names=['Activity'], squeeze=T
y_test_labels = y_test.map({1: 'WALKING', 2:'WALKING_UPSTAIRS',3:'WALKING_DOWNSTAIRS'
                       4:'SITTING', 5:'STANDING',6:'LAYING'})


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

Out[14]:
```
      tBodyAcc-mean()-X  tBodyAcc-mean()-Y  tBodyAcc-mean()-Z  \
1894           0.276425          -0.018626          -0.106165

      tBodyAcc-std()-X  tBodyAcc-std()-Y  tBodyAcc-std()-Z  tBodyAcc-mad()-X  \
1894         -0.996926         -0.985498         -0.990213         -0.997407

      tBodyAcc-mad()-Y  tBodyAcc-mad()-Z  tBodyAcc-max()-X  ...  \
1894          -0.98491         -0.990259         -0.940472  ...

      angle(tBodyAccMean,gravity)  angle(tBodyAccJerkMean),gravityMean)  \
1894                     0.211316                                0.179502

      angle(tBodyGyroMean,gravityMean)  angle(tBodyGyroJerkMean,gravityMean)  \
1894                         -0.653772                                0.003994

      angle(X,gravityMean)  angle(Y,gravityMean)  angle(Z,gravityMean)  \
1894             -0.521574             -0.181325             -0.161829
```

```
      subject  Activity  ActivityName
1894       18         4       SITTING

[1 rows x 564 columns]
```

0.0.1 Correlation between Feature

```
In [10]: x=train.corr()
```

```
In [17]: s=x.unstack()
```

```
In [22]: so=s.sort_values(kind='quicksort',ascending=False)
```

```
In [39]: # top 20 correlated values
         g=pd.DataFrame(so,columns=['a'])
         g.loc[g['a']!=1].head(20)
```

```
Out[39]:                                                                 a
         fBodyAccJerk-energy()-Z       tBodyAccJerk-energy()-Z       1.000000
         tBodyAccJerk-energy()-Z       fBodyAccJerk-energy()-Z       1.000000
         tBodyAccJerk-energy()-Y       fBodyAccJerk-energy()-Y       1.000000
         fBodyAccJerk-energy()-Y       tBodyAccJerk-energy()-Y       1.000000
         fBodyAccJerk-energy()-X       tBodyAccJerk-energy()-X       0.999999
         tBodyAccJerk-energy()-X       fBodyAccJerk-energy()-X       0.999999
         fBodyAcc-energy()-X           fBodyAcc-bandsEnergy()-1,24   0.999878
         fBodyAcc-bandsEnergy()-1,24   fBodyAcc-energy()-X           0.999878
         fBodyGyro-bandsEnergy()-1,24  fBodyGyro-energy()-X          0.999767
         fBodyGyro-energy()-X          fBodyGyro-bandsEnergy()-1,24  0.999767
         fBodyAcc-bandsEnergy()-1,24.1 fBodyAcc-energy()-Y           0.999661
         fBodyAcc-energy()-Y           fBodyAcc-bandsEnergy()-1,24.1 0.999661
         tBodyAccJerkMag-mean()        tBodyAccJerk-sma()            0.999656
         tBodyAccJerkMag-sma()         tBodyAccJerk-sma()            0.999656
         tBodyAccJerk-sma()            tBodyAccJerkMag-mean()        0.999656
                                       tBodyAccJerkMag-sma()         0.999656
         tBodyAcc-energy()-X           fBodyAcc-energy()-X           0.999611
         fBodyAcc-energy()-X           tBodyAcc-energy()-X           0.999611
         fBodyGyro-energy()-Z          fBodyGyro-bandsEnergy()-1,24.2 0.999523
         fBodyGyro-bandsEnergy()-1,24.2 fBodyGyro-energy()-Z         0.999523
```

0.0.2 Count of Acceleration and gyroscope features

```
In [76]: mydict={}
         g=set([col.split('-')[0] for col in train.columns])
         mydict={key: 0 for key in g}
         for i in [col.split('-')[0] for col in train.columns]:
             if i in mydict.keys():
                 mydict[i]+=1
         from collections import Counter
         count=Counter(mydict).most_common(10)
```

```
In [77]: pd.DataFrame(count,columns=['Feature','Count'])
```

```
Out[77]:            Feature  Count
         0      fBodyAccJerk     79
         1          fBodyAcc     79
         2         fBodyGyro     79
         3         tBodyGyro     40
         4          tBodyAcc     40
         5       tGravityAcc     40
         6      tBodyAccJerk     40
         7     tBodyGyroJerk     40
         8      tBodyGyroMag     13
         9  fBodyBodyGyroMag     13
```

There are many Acceleration and gyroscope features and few gravity features

0.0.3 Description

```
In [44]: train.describe()
```

```
Out[44]:        tBodyAcc-mean()-X  tBodyAcc-mean()-Y  tBodyAcc-mean()-Z  \
         count        7352.000000        7352.000000        7352.000000
         mean            0.274488          -0.017695          -0.109141
         std             0.070261           0.040811           0.056635
         min            -1.000000          -1.000000          -1.000000
         25%             0.262975          -0.024863          -0.120993
         50%             0.277193          -0.017219          -0.108676
         75%             0.288461          -0.010783          -0.097794
         max             1.000000           1.000000           1.000000

                tBodyAcc-std()-X  tBodyAcc-std()-Y  tBodyAcc-std()-Z  tBodyAcc-mad()-X  \
         count       7352.000000       7352.000000       7352.000000       7352.000000
         mean          -0.605438         -0.510938         -0.604754         -0.630512
         std            0.448734          0.502645          0.418687          0.424073
         min           -1.000000         -0.999873         -1.000000         -1.000000
         25%           -0.992754         -0.978129         -0.980233         -0.993591
         50%           -0.946196         -0.851897         -0.859365         -0.950709
         75%           -0.242813         -0.034231         -0.262415         -0.292680
         max            1.000000          0.916238          1.000000          1.000000

                tBodyAcc-mad()-Y  tBodyAcc-mad()-Z  tBodyAcc-max()-X   ...  \
         count       7352.000000       7352.000000       7352.000000   ...
         mean          -0.526907         -0.606150         -0.468604   ...
         std            0.485942          0.414122          0.544547   ...
         min           -1.000000         -1.000000         -1.000000   ...
         25%           -0.978162         -0.980251         -0.936219   ...
         50%           -0.857328         -0.857143         -0.881637   ...
         75%           -0.066701         -0.265671         -0.017129   ...
         max            0.967664          1.000000          1.000000   ...
```

```
       fBodyBodyGyroJerkMag-kurtosis()  angle(tBodyAccMean,gravity)  \
count                      7352.000000                  7352.000000
mean                         -0.625294                     0.008684
std                           0.307584                     0.336787
min                          -0.999765                    -0.976580
25%                          -0.845573                    -0.121527
50%                          -0.711692                     0.009509
75%                          -0.503878                     0.150865
max                           0.956845                     1.000000

       angle(tBodyAccJerkMean),gravityMean)  angle(tBodyGyroMean,gravityMean)  \
count                           7352.000000                       7352.000000
mean                               0.002186                          0.008726
std                                0.448306                          0.608303
min                               -1.000000                         -1.000000
25%                               -0.289549                         -0.482273
50%                                0.008943                          0.008735
75%                                0.292861                          0.506187
max                                1.000000                          0.998702

       angle(tBodyGyroJerkMean,gravityMean)  angle(X,gravityMean)  \
count                           7352.000000           7352.000000
mean                              -0.005981             -0.489547
std                                0.477975              0.511807
min                               -1.000000             -1.000000
25%                               -0.376341             -0.812065
50%                               -0.000368             -0.709417
75%                                0.359368             -0.509079
max                                0.996078              1.000000

       angle(Y,gravityMean)  angle(Z,gravityMean)      subject     Activity
count           7352.000000           7352.000000  7352.000000  7352.000000
mean               0.058593             -0.056515    17.413085     3.643362
std                0.297480              0.279122     8.975143     1.744802
min               -1.000000             -1.000000     1.000000     1.000000
25%               -0.017885             -0.143414     8.000000     2.000000
50%                0.182071              0.003181    19.000000     4.000000
75%                0.248353              0.107659    26.000000     5.000000
max                0.478157              1.000000    30.000000     6.000000

[8 rows x 563 columns]
```

In [45]:
```
# Activities are the class labels
# It is a 6 class classification
ACTIVITIES = {
    0: 'WALKING',
    1: 'WALKING_UPSTAIRS',
```

```
    2: 'WALKING_DOWNSTAIRS',
    3: 'SITTING',
    4: 'STANDING',
    5: 'LAYING',
}

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

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

0.0.4 Data

In [15]:
```
# Data directory
DATADIR = 'UCI_HAR_Dataset'
```

In [16]:
```
# 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 [17]:
```
# Utility function to read the data from csv file
def _read_csv(filename):
    return pd.read_csv(filename, delim_whitespace=True, header=None)

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

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

```
        )

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

In [18]:
```
def load_y(subset):
    """
    The objective that we are trying to predict is a integer, from 1 to 6,
    that represents a human activity. We return a binary representation of
    every sample objective as a 6 bits vector using One Hot Encoding
    (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html)
    """
    filename = f'UCI_HAR_Dataset/{subset}/y_{subset}.txt'
    y = _read_csv(filename)[0]

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

In [19]:
```
def load_data():
    """
    Obtain the dataset from multiple files.
    Returns: X_train, X_test, y_train, y_test
    """
    X_train, X_test = load_signals('train'), load_signals('test')
    y_train, y_test = load_y('train'), load_y('test')

    return X_train, X_test, y_train, y_test
```

In [20]:
```
# Importing tensorflow
np.random.seed(42)
import tensorflow as tf
tf.set_random_seed(42)
```

In [21]:
```
# Configuring a session
session_conf = tf.ConfigProto(
    intra_op_parallelism_threads=1,
    inter_op_parallelism_threads=1
)
```

In [22]:
```
# Import Keras
from keras import backend as K
sess = tf.Session(graph=tf.get_default_graph(), config=session_conf)
K.set_session(sess)
```

```
Using TensorFlow backend.
```

In [23]:
```
# Importing libraries
from keras.models import Sequential
```

```
from keras.layers import LSTM
from keras.layers.core import Dense, Dropout
```

In [24]:
```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 32
```

In [25]:
```
# Utility function to count the number of classes
def _count_classes(y):
    return len(set([tuple(category) for category in y]))
```

In [26]:
```
# Loading the train and test data
X_train, X_test, Y_train, Y_test = load_data()
```

```
c:\users\dell\appdata\local\programs\python\python36\lib\site-packages\ipykernel_launcher.py:1
  if sys.path[0] == '':
```

In [27]:
```
timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = _count_classes(Y_train)

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

```
128
9
7352
```

- Defining the Architecture of LSTM

Model 1: 64 hidden layers

In [88]:
```
# Initiliazing the sequential model
from hyperopt import Trials, STATUS_OK, tpe
from hyperas import optim
from hyperas.distributions import choice, uniform
model = Sequential()
# Configuring the parameters
model.add(LSTM(64, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_10 (LSTM)               (None, 64)                18944
_________________________________________________________________
dropout_6 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_6 (Dense)              (None, 6)                 390
=================================================================
Total params: 19,334
Trainable params: 19,334
Non-trainable params: 0
_________________________________________________________________
```

In [89]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [90]:
```
# Training the model
history=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
3408/7352 [============>...] - ETA: 7:47 - loss: 1.7549 - acc: 0.250 - ETA: 4:08 - loss: 1.7856
Epoch 2/30
3440/7352 [=============>...] - ETA: 29s - loss: 1.0312 - acc: 0.37 - ETA: 30s - loss: 0.9927 -
Epoch 3/30
3440/7352 [=============>...] - ETA: 33s - loss: 0.5445 - acc: 0.81 - ETA: 34s - loss: 0.8037 -
Epoch 4/30
3440/7352 [=============>...] - ETA: 32s - loss: 0.6001 - acc: 0.93 - ETA: 32s - loss: 0.6813 -
Epoch 5/30
3440/7352 [=============>...] - ETA: 32s - loss: 0.8302 - acc: 0.68 - ETA: 32s - loss: 0.6811 -
Epoch 6/30
3440/7352 [=============>...] - ETA: 32s - loss: 0.3064 - acc: 0.81 - ETA: 31s - loss: 0.3585 -
Epoch 7/30
3440/7352 [=============>...] - ETA: 32s - loss: 0.3356 - acc: 0.87 - ETA: 31s - loss: 0.3782 -
Epoch 8/30
3440/7352 [=============>...] - ETA: 29s - loss: 0.1509 - acc: 1.00 - ETA: 31s - loss: 0.3069 -
Epoch 9/30
3440/7352 [=============>...] - ETA: 29s - loss: 0.1559 - acc: 0.93 - ETA: 30s - loss: 0.1282 -
Epoch 10/30
3440/7352 [=============>...] - ETA: 28s - loss: 0.2665 - acc: 0.93 - ETA: 29s - loss: 0.1982 -
```

```
Epoch 11/30
3440/7352 [=============>...] - ETA: 28s - loss: 0.3164 - acc: 0.93 - ETA: 29s - loss: 0.2913 -
Epoch 12/30
3440/7352 [=============>...] - ETA: 32s - loss: 0.3355 - acc: 0.93 - ETA: 32s - loss: 0.2066 -
Epoch 13/30
3440/7352 [=============>...] - ETA: 29s - loss: 0.0741 - acc: 0.93 - ETA: 30s - loss: 0.1051 -
Epoch 14/30
3440/7352 [=============>...] - ETA: 29s - loss: 0.1279 - acc: 0.87 - ETA: 29s - loss: 0.3064 -
Epoch 15/30
3440/7352 [=============>...] - ETA: 30s - loss: 0.0889 - acc: 1.00 - ETA: 30s - loss: 0.0717 -
Epoch 16/30
3440/7352 [=============>...] - ETA: 32s - loss: 0.1650 - acc: 0.93 - ETA: 31s - loss: 0.1129 -
Epoch 17/30
3440/7352 [=============>...] - ETA: 36s - loss: 0.1318 - acc: 0.87 - ETA: 34s - loss: 0.2239 -
Epoch 18/30
3440/7352 [=============>...] - ETA: 29s - loss: 0.5613 - acc: 0.87 - ETA: 28s - loss: 0.4599 -
Epoch 19/30
3440/7352 [=============>...] - ETA: 36s - loss: 0.0800 - acc: 0.93 - ETA: 34s - loss: 0.0683 -
Epoch 20/30
3440/7352 [=============>...] - ETA: 30s - loss: 0.2934 - acc: 0.81 - ETA: 29s - loss: 0.2094 -
Epoch 21/30
3440/7352 [=============>...] - ETA: 28s - loss: 0.0435 - acc: 1.00 - ETA: 29s - loss: 0.2092 -
Epoch 22/30
3440/7352 [=============>...] - ETA: 30s - loss: 0.0450 - acc: 1.00 - ETA: 29s - loss: 0.0570 -
Epoch 23/30
3440/7352 [=============>...] - ETA: 29s - loss: 0.2933 - acc: 0.93 - ETA: 29s - loss: 0.1505 -
Epoch 24/30
3440/7352 [=============>...] - ETA: 34s - loss: 0.0852 - acc: 0.93 - ETA: 33s - loss: 0.1504 -
Epoch 25/30
3440/7352 [=============>...] - ETA: 32s - loss: 0.0434 - acc: 1.00 - ETA: 30s - loss: 0.0593 -
Epoch 26/30
3440/7352 [=============>...] - ETA: 32s - loss: 0.1625 - acc: 0.87 - ETA: 31s - loss: 0.2222 -
Epoch 27/30
3440/7352 [=============>...] - ETA: 31s - loss: 0.0419 - acc: 1.00 - ETA: 31s - loss: 0.0805 -
Epoch 28/30
3440/7352 [=============>...] - ETA: 31s - loss: 0.1203 - acc: 0.93 - ETA: 31s - loss: 0.1254 -
Epoch 29/30
3440/7352 [=============>...] - ETA: 32s - loss: 0.3680 - acc: 0.87 - ETA: 32s - loss: 0.3036 -
Epoch 30/30
3440/7352 [=============>...] - ETA: 33s - loss: 0.0083 - acc: 1.00 - ETA: 32s - loss: 0.1619 -
```

In [91]:
```python
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
print()
```

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

```
LAYING                    508       0       4       0                   0
SITTING                     0     376     115       0                   0
STANDING                    0      81     449       2                   0
WALKING                     0       0       0     472                  23
WALKING_DOWNSTAIRS          0       0       0       1                 419
WALKING_UPSTAIRS            0       5       0      12                  26

Pred                WALKING_UPSTAIRS
True
LAYING                            25
SITTING                            0
STANDING                           0
WALKING                            1
WALKING_DOWNSTAIRS                 0
WALKING_UPSTAIRS                 428
```

In [92]:
```python
score = model.evaluate(X_test, Y_test)
```

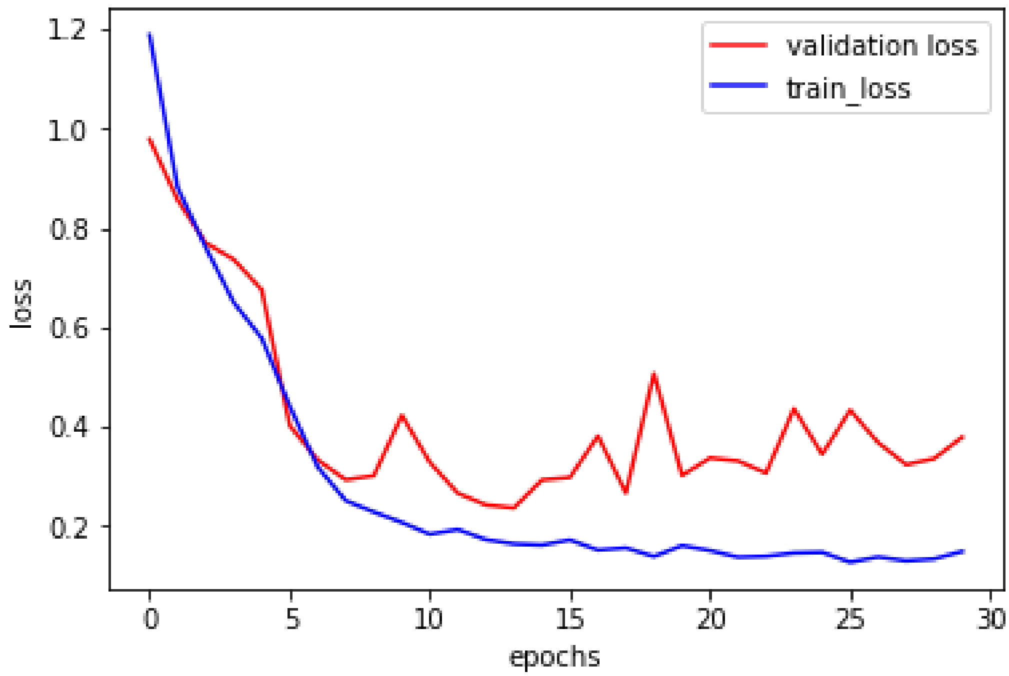
```
2947/2947 [==============================] - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - ET
```

In [50]:
```python
import matplotlib.pyplot as plt
def plot_dynamic(epochs,val_loss,train_loss):
    x=[x for x in range(epochs)]
    plt.plot(x,val_loss,color='r',label='validation loss')
    plt.plot(x,train_loss,color='b',label='train_loss')
    plt.legend()
    plt.xlabel('epochs')
    plt.ylabel('loss')
    plt.show()
```

In [93]:
```python
score
```

Out[93]:
```
[0.37837991104009505, 0.8998982015609094]
```

In [105]:
```python
#plotting train and test loss
val_loss=history.history['val_loss']
train_loss=history.history['loss']
plot_dynamic(epochs,val_loss,train_loss)
print('Test loss',score[0])
print('Test accuracy',score[1])
```




```
Test loss 0.37837991104009505
Test accuracy 0.8998982015609094
```

0.1 Model 2: 128 hidden layers

In [41]:
```python
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(128, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.4))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_7 (LSTM)                (None, 128)               70656
_________________________________________________________________
dropout_7 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 774
=================================================================
```

```
Total params: 71,430
Trainable params: 71,430
Non-trainable params: 0
_________________________________________________________________
```

In [42]:
```python
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [43]:
```python
# Training the model
history=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 52s 7ms/step - loss: 1.2749 - acc: 0.4230 - val_lo
Epoch 2/30
7352/7352 [==============================] - 51s 7ms/step - loss: 0.8672 - acc: 0.6001 - val_lo
Epoch 3/30
7352/7352 [==============================] - 51s 7ms/step - loss: 0.7140 - acc: 0.6926 - val_lo
Epoch 4/30
7352/7352 [==============================] - 51s 7ms/step - loss: 0.4559 - acc: 0.8307 - val_lo
Epoch 5/30
7352/7352 [==============================] - 51s 7ms/step - loss: 0.3152 - acc: 0.8893 - val_lo
Epoch 6/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.2271 - acc: 0.9242 - val_lo
Epoch 7/30
7352/7352 [==============================] - 51s 7ms/step - loss: 0.2024 - acc: 0.9279 - val_lo
Epoch 8/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1812 - acc: 0.9382 - val_lo
Epoch 9/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.2227 - acc: 0.9222 - val_lo
Epoch 10/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1840 - acc: 0.9376 - val_lo
Epoch 11/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1596 - acc: 0.9450 - val_lo
Epoch 12/30
7352/7352 [==============================] - 51s 7ms/step - loss: 0.1559 - acc: 0.9423 - val_lo
Epoch 13/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1477 - acc: 0.9483 - val_lo
Epoch 14/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1578 - acc: 0.9438 - val_lo
Epoch 15/30
```

```
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1452 - acc: 0.9453 - val_lo
Epoch 16/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1378 - acc: 0.9457 - val_lo
Epoch 17/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1388 - acc: 0.9491 - val_lo
Epoch 18/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1331 - acc: 0.9517 - val_lo
Epoch 19/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1418 - acc: 0.9478 - val_lo
Epoch 20/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1207 - acc: 0.9540 - val_lo
Epoch 21/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1257 - acc: 0.9499 - val_lo
Epoch 22/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1438 - acc: 0.9489 - val_lo
Epoch 23/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1350 - acc: 0.9498 - val_lo
Epoch 24/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1275 - acc: 0.9512 - val_lo
Epoch 25/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1295 - acc: 0.9516 - val_lo
Epoch 26/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1279 - acc: 0.9524 - val_lo
Epoch 27/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1416 - acc: 0.9529 - val_lo
Epoch 28/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.1143 - acc: 0.9543 - val_lo
Epoch 29/30
7352/7352 [==============================] - 55s 7ms/step - loss: 0.1414 - acc: 0.9516 - val_lo
Epoch 30/30
7352/7352 [==============================] - 56s 8ms/step - loss: 0.1351 - acc: 0.9497 - val_lo
```

In [46]:
```
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 517        0        20        0                   0
SITTING                  0      384       105        0                   0
STANDING                 0       85       447        0                   0
WALKING                  0        9         1      443                  19
WALKING_DOWNSTAIRS       0        0         0        0                 420
WALKING_UPSTAIRS         0        2         6        2                   1

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            2
```

```
STANDING                        0
WALKING                        24
WALKING_DOWNSTAIRS              0
WALKING_UPSTAIRS              460
```

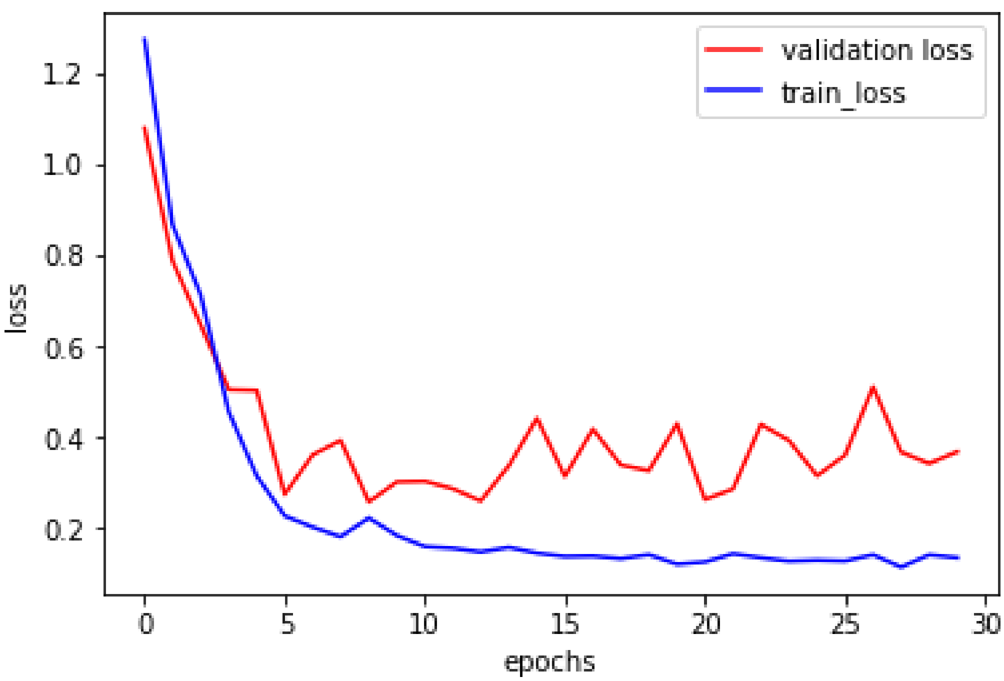
```
In [47]: score = model.evaluate(X_test, Y_test)
```

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

```
In [48]: score
```

```
Out[48]: [0.3682935545481282, 0.9063454360366474]
```

```
In [51]: #plotting train and test loss
         val_loss=history.history['val_loss']
         train_loss=history.history['loss']
         plot_dynamic(epochs,val_loss,train_loss)
         print('Test loss',score[0])
         print('Test accuracy',score[1])
```




```
Test loss 0.3682935545481282
Test accuracy 0.9063454360366474
```

0.2 Model 3: 256 hidden layer

In [52]:
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(256, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.4))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_8 (LSTM)                (None, 256)               272384
_________________________________________________________________
dropout_8 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_5 (Dense)              (None, 6)                 1542
=================================================================
Total params: 273,926
Trainable params: 273,926
Non-trainable params: 0
_________________________________________________________________
```

In [53]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [54]:
```
# Training the model
history=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 118s 16ms/step - loss: 1.2889 - acc: 0.4414 - val_
Epoch 2/30
7352/7352 [==============================] - 117s 16ms/step - loss: 1.1250 - acc: 0.5076 - val_
Epoch 3/30
7352/7352 [==============================] - 117s 16ms/step - loss: 0.7406 - acc: 0.6786 - val_
Epoch 4/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.4854 - acc: 0.8214 - val_
Epoch 5/30
```

```
7352/7352 [==============================] - 117s 16ms/step - loss: 0.3035 - acc: 0.8954 - val_
Epoch 6/30
7352/7352 [==============================] - 117s 16ms/step - loss: 0.2416 - acc: 0.9149 - val_
Epoch 7/30
7352/7352 [==============================] - 117s 16ms/step - loss: 0.1964 - acc: 0.9245 - val_
Epoch 8/30
7352/7352 [==============================] - 117s 16ms/step - loss: 0.1899 - acc: 0.9332 - val_
Epoch 9/30
7352/7352 [==============================] - 117s 16ms/step - loss: 0.1645 - acc: 0.9412 - val_
Epoch 10/30
7352/7352 [==============================] - 117s 16ms/step - loss: 0.1622 - acc: 0.9441 - val_
Epoch 11/30
7352/7352 [==============================] - 117s 16ms/step - loss: 0.1526 - acc: 0.9456 - val_
Epoch 12/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.2000 - acc: 0.9274 - val_
Epoch 13/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.1445 - acc: 0.9449 - val_
Epoch 14/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.1565 - acc: 0.9440 - val_
Epoch 15/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.1365 - acc: 0.9478 - val_
Epoch 16/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.1331 - acc: 0.9482 - val_
Epoch 17/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.1383 - acc: 0.9479 - val_
Epoch 18/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.1421 - acc: 0.9480 - val_
Epoch 19/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.1624 - acc: 0.9344 - val_
Epoch 20/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.1305 - acc: 0.9509 - val_
Epoch 21/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.1391 - acc: 0.9489 - val_
Epoch 22/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.1421 - acc: 0.9478 - val_
Epoch 23/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.1445 - acc: 0.9461 - val_
Epoch 24/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.1420 - acc: 0.9467 - val_
Epoch 25/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.1488 - acc: 0.9421 - val_
Epoch 26/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.1343 - acc: 0.9504 - val_
Epoch 27/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.1338 - acc: 0.9497 - val_
Epoch 28/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.1518 - acc: 0.9480 - val_
Epoch 29/30
```

```
7352/7352 [==============================] - 118s 16ms/step - loss: 0.1503 - acc: 0.9459 - val_
Epoch 30/30
7352/7352 [==============================] - 119s 16ms/step - loss: 0.1684 - acc: 0.9444 - val_
```

```
In [55]: print(confusion_matrix(Y_test, model.predict(X_test)))
```

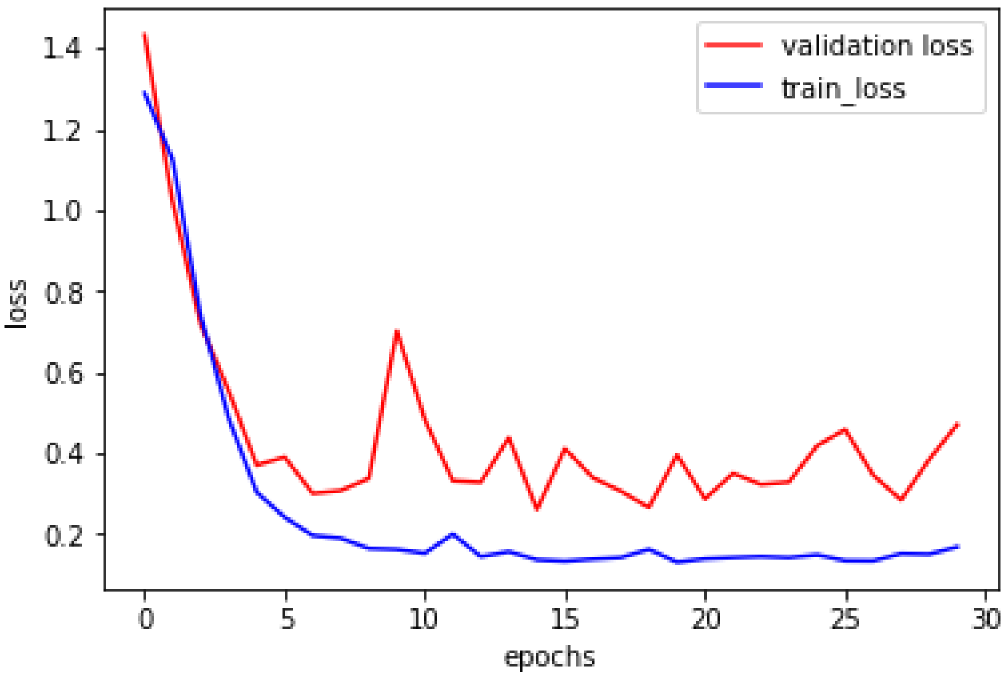
```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 517        0        20        0                   0
SITTING                  0      340       149        0                   0
STANDING                 0       45       487        0                   0
WALKING                  0        0         0      465                  29
WALKING_DOWNSTAIRS       0        0         0        1                 416
WALKING_UPSTAIRS         0        2        19        8                   2

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            2
STANDING                           0
WALKING                            2
WALKING_DOWNSTAIRS                 3
WALKING_UPSTAIRS                 440
```

```
In [56]: score = model.evaluate(X_test, Y_test)
         score
```

```
2947/2947 [==============================] - 9s 3ms/step
```

```
Out[56]: [0.4706485792249341, 0.9043094672548354]
```

```
In [57]: #plotting train and test loss
         val_loss=history.history['val_loss']
         train_loss=history.history['loss']
         plot_dynamic(epochs,val_loss,train_loss)
         print('Test loss',score[0])
         print('Test accuracy',score[1])
```




```
Test loss 0.4706485792249341
Test accuracy 0.9043094672548354
```

0.2.1 Model 4 : 2 LSTM Layers

Finding the best dropout rate

```
In [28]: scores=[]
         for i in (0.3,0.5,0.7):
             model = Sequential()
             # Configuring the parameters
             model.add(LSTM(128,return_sequences=True, input_shape=(timesteps, input_dim)))
             # Adding a dropout layer
             model.add(Dropout(i))
             model.add(LSTM(256))
             model.add(Dropout(i))
             # Adding a dense output layer with sigmoid activation
             model.add(Dense(n_classes, activation='sigmoid'))
             model.summary()
             model.compile(loss='categorical_crossentropy',
                       optimizer='rmsprop',
                       metrics=['accuracy'])

             history=model.fit(X_train,
```

```
                  Y_train,
                  batch_size=batch_size,
                  validation_data=(X_test, Y_test),
                  epochs=epochs)
            score = model.evaluate(X_test, Y_test)
            scores.append(score)
            print("The score for model with dropout of {} is {}".format(i,score))
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 128, 128)          70656
_________________________________________________________________
dropout_1 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
lstm_2 (LSTM)                (None, 256)               394240
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 1542
=================================================================
Total params: 466,438
Trainable params: 466,438
Non-trainable params: 0
_________________________________________________________________
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 223s 30ms/step - loss: 1.0058 - acc: 0.5492 - val_
Epoch 2/30
7352/7352 [==============================] - 222s 30ms/step - loss: 0.6842 - acc: 0.7035 - val_
Epoch 3/30
7352/7352 [==============================] - 222s 30ms/step - loss: 0.4184 - acc: 0.8390 - val_
Epoch 4/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.2207 - acc: 0.9214 - val_
Epoch 5/30
7352/7352 [==============================] - 223s 30ms/step - loss: 0.1854 - acc: 0.9340 - val_
Epoch 6/30
7352/7352 [==============================] - 222s 30ms/step - loss: 0.1677 - acc: 0.9419 - val_
Epoch 7/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.1560 - acc: 0.9418 - val_
Epoch 8/30
7352/7352 [==============================] - 222s 30ms/step - loss: 0.1404 - acc: 0.9434 - val_
Epoch 9/30
7352/7352 [==============================] - 222s 30ms/step - loss: 0.1470 - acc: 0.9453 - val_
Epoch 10/30
7352/7352 [==============================] - 222s 30ms/step - loss: 0.1288 - acc: 0.9476 - val_
Epoch 11/30
7352/7352 [==============================] - 218s 30ms/step - loss: 0.1338 - acc: 0.9479 - val_
```

```
Epoch 12/30
7352/7352 [==============================] - 219s 30ms/step - loss: 0.1374 - acc: 0.9509 - val_
Epoch 13/30
7352/7352 [==============================] - 218s 30ms/step - loss: 0.1376 - acc: 0.9490 - val_
Epoch 14/30
7352/7352 [==============================] - 218s 30ms/step - loss: 0.1322 - acc: 0.9504 - val_
Epoch 15/30
7352/7352 [==============================] - 218s 30ms/step - loss: 0.1377 - acc: 0.9509 - val_
Epoch 16/30
7352/7352 [==============================] - 218s 30ms/step - loss: 0.1326 - acc: 0.9512 - val_
Epoch 17/30
7352/7352 [==============================] - 224s 31ms/step - loss: 0.1231 - acc: 0.9502 - val_
Epoch 18/30
7352/7352 [==============================] - 218s 30ms/step - loss: 0.1647 - acc: 0.9499 - val_
Epoch 19/30
7352/7352 [==============================] - 217s 30ms/step - loss: 0.1200 - acc: 0.9550 - val_
Epoch 20/30
7352/7352 [==============================] - 218s 30ms/step - loss: 0.1371 - acc: 0.9518 - val_
Epoch 21/30
7352/7352 [==============================] - 218s 30ms/step - loss: 0.1363 - acc: 0.9520 - val_
Epoch 22/30
7352/7352 [==============================] - 218s 30ms/step - loss: 0.1421 - acc: 0.9512 - val_
Epoch 23/30
7352/7352 [==============================] - 216s 29ms/step - loss: 0.1323 - acc: 0.9544 - val_
Epoch 24/30
7352/7352 [==============================] - 216s 29ms/step - loss: 0.1457 - acc: 0.9480 - val_
Epoch 25/30
7352/7352 [==============================] - 217s 29ms/step - loss: 0.1474 - acc: 0.9508 - val_
Epoch 26/30
7352/7352 [==============================] - 216s 29ms/step - loss: 0.1336 - acc: 0.9520 - val_
Epoch 27/30
7352/7352 [==============================] - 216s 29ms/step - loss: 0.1419 - acc: 0.9493 - val_
Epoch 28/30
7352/7352 [==============================] - 216s 29ms/step - loss: 0.1155 - acc: 0.9600 - val_
Epoch 29/30
7352/7352 [==============================] - 216s 29ms/step - loss: 0.1359 - acc: 0.9494 - val_
Epoch 30/30
7352/7352 [==============================] - 216s 29ms/step - loss: 0.1310 - acc: 0.9523 - val_
2947/2947 [==============================] - 15s 5ms/step
The score for model with dropout of 0.3 is [0.6947863878794325, 0.8829317950458093]

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 128, 128)          70656
_________________________________________________________________
dropout_3 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
lstm_4 (LSTM)                (None, 256)               394240
```

```
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 1542
=================================================================
Total params: 466,438
Trainable params: 466,438
Non-trainable params: 0
_________________________________________________________________
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 224s 31ms/step - loss: 1.0125 - acc: 0.5430 - val_
Epoch 2/30
7352/7352 [==============================] - 222s 30ms/step - loss: 0.7514 - acc: 0.6672 - val_
Epoch 3/30
7352/7352 [==============================] - 227s 31ms/step - loss: 0.5793 - acc: 0.7586 - val_
Epoch 4/30
7352/7352 [==============================] - 222s 30ms/step - loss: 0.3778 - acc: 0.8622 - val_
Epoch 5/30
7352/7352 [==============================] - 227s 31ms/step - loss: 0.2536 - acc: 0.9149 - val_
Epoch 6/30
7352/7352 [==============================] - 221s 30ms/step - loss: 0.1777 - acc: 0.9363 - val_
Epoch 7/30
7352/7352 [==============================] - 221s 30ms/step - loss: 0.1686 - acc: 0.9380 - val_
Epoch 8/30
7352/7352 [==============================] - 221s 30ms/step - loss: 0.1643 - acc: 0.9408 - val_
Epoch 9/30
7352/7352 [==============================] - 221s 30ms/step - loss: 0.1470 - acc: 0.9460 - val_
Epoch 10/30
7352/7352 [==============================] - 222s 30ms/step - loss: 0.1548 - acc: 0.9436 - val_
Epoch 11/30
7352/7352 [==============================] - 226s 31ms/step - loss: 0.1428 - acc: 0.9455 - val_
Epoch 12/30
7352/7352 [==============================] - 240s 33ms/step - loss: 0.1402 - acc: 0.9470 - val_
Epoch 13/30
7352/7352 [==============================] - 236s 32ms/step - loss: 0.1523 - acc: 0.9461 - val_
Epoch 14/30
7352/7352 [==============================] - 234s 32ms/step - loss: 0.1370 - acc: 0.9463 - val_
Epoch 15/30
7352/7352 [==============================] - 240s 33ms/step - loss: 0.1484 - acc: 0.9478 - val_
Epoch 16/30
7352/7352 [==============================] - 253s 34ms/step - loss: 0.2068 - acc: 0.9150 - val_
Epoch 17/30
7352/7352 [==============================] - 251s 34ms/step - loss: 0.1414 - acc: 0.9467 - val_
Epoch 18/30
7352/7352 [==============================] - 250s 34ms/step - loss: 0.1384 - acc: 0.9455 - val_
Epoch 19/30
7352/7352 [==============================] - 252s 34ms/step - loss: 0.1465 - acc: 0.9468 - val_
```

```
Epoch 20/30
7352/7352 [==============================] - 248s 34ms/step - loss: 0.1442 - acc: 0.9418 - val_
Epoch 21/30
7352/7352 [==============================] - 223s 30ms/step - loss: 0.1566 - acc: 0.9449 - val_
Epoch 22/30
7352/7352 [==============================] - 222s 30ms/step - loss: 0.1489 - acc: 0.9486 - val_
Epoch 23/30
7352/7352 [==============================] - 222s 30ms/step - loss: 0.1507 - acc: 0.9463 - val_
Epoch 24/30
7352/7352 [==============================] - 222s 30ms/step - loss: 0.1384 - acc: 0.9510 - val_
Epoch 25/30
7352/7352 [==============================] - 223s 30ms/step - loss: 0.1324 - acc: 0.9514 - val_
Epoch 26/30
7352/7352 [==============================] - 222s 30ms/step - loss: 0.1393 - acc: 0.9499 - val_
Epoch 27/30
7352/7352 [==============================] - 222s 30ms/step - loss: 0.1359 - acc: 0.9527 - val_
Epoch 28/30
7352/7352 [==============================] - 222s 30ms/step - loss: 0.1312 - acc: 0.9525 - val_
Epoch 29/30
7352/7352 [==============================] - 222s 30ms/step - loss: 0.1455 - acc: 0.9506 - val_
Epoch 30/30
7352/7352 [==============================] - 222s 30ms/step - loss: 0.1470 - acc: 0.9509 - val_
2947/2947 [==============================] - 16s 5ms/step
The score for model with dropout of 0.5 is [0.7302716798681125, 0.9009161859518154]
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_5 (LSTM)                (None, 128, 128)          70656
_________________________________________________________________
dropout_5 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
lstm_6 (LSTM)                (None, 256)               394240
_________________________________________________________________
dropout_6 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 1542
=================================================================
Total params: 466,438
Trainable params: 466,438
Non-trainable params: 0
_________________________________________________________________
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 226s 31ms/step - loss: 1.0732 - acc: 0.5167 - val_
Epoch 2/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.7896 - acc: 0.6258 - val_
Epoch 3/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.7158 - acc: 0.6583 - val_
```

```
Epoch 4/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.7187 - acc: 0.6944 - val_
Epoch 5/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.4998 - acc: 0.7833 - val_
Epoch 6/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.2782 - acc: 0.9066 - val_
Epoch 7/30
7352/7352 [==============================] - 224s 31ms/step - loss: 0.2326 - acc: 0.9181 - val_
Epoch 8/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.1811 - acc: 0.9399 - val_
Epoch 9/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.1981 - acc: 0.9354 - val_
Epoch 10/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.1796 - acc: 0.9340 - val_
Epoch 11/30
7352/7352 [==============================] - 225s 31ms/step - loss: 0.1709 - acc: 0.9423 - val_
Epoch 12/30
7352/7352 [==============================] - 224s 31ms/step - loss: 0.1720 - acc: 0.9426 - val_
Epoch 13/30
7352/7352 [==============================] - 224s 31ms/step - loss: 0.1648 - acc: 0.9404 - val_
Epoch 14/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.1736 - acc: 0.9436 - val_
Epoch 15/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.1538 - acc: 0.9483 - val_
Epoch 16/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.1636 - acc: 0.9418 - val_
Epoch 17/30
7352/7352 [==============================] - 224s 31ms/step - loss: 0.1684 - acc: 0.9480 - val_
Epoch 18/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.1572 - acc: 0.9489 - val_
Epoch 19/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.1561 - acc: 0.9444 - val_
Epoch 20/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.1574 - acc: 0.9467 - val_
Epoch 21/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.1660 - acc: 0.9453 - val_
Epoch 22/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.1565 - acc: 0.9465 - val_
Epoch 23/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.1894 - acc: 0.9445 - val_
Epoch 24/30
7352/7352 [==============================] - 223s 30ms/step - loss: 0.1857 - acc: 0.9476 - val_
Epoch 25/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.1554 - acc: 0.9460 - val_
Epoch 26/30
7352/7352 [==============================] - 224s 31ms/step - loss: 0.1830 - acc: 0.9404 - val_
Epoch 27/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.1662 - acc: 0.9456 - val_
```

```
Epoch 28/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.1458 - acc: 0.9475 - val_
Epoch 29/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.1537 - acc: 0.9464 - val_
Epoch 30/30
7352/7352 [==============================] - 224s 30ms/step - loss: 0.1793 - acc: 0.9456 - val_
2947/2947 [==============================] - 16s 6ms/step
The score for model with dropout of 0.7 is [0.47057243221619627, 0.9039701391245334]
```

In [40]:
```python
drop=[0.3,0.5,0.7]
count=0
for i in drop:

    print("The accuracy for model with dropout rate of {} is {}".format(i,scores[coun
    count+=1
```

```
The accuracy for model with dropout rate of 0.3 is 0.8829317950458093
The accuracy for model with dropout rate of 0.5 is 0.9009161859518154
The accuracy for model with dropout rate of 0.7 is 0.9039701391245334
```

In [60]:
```python
#model with dropout rate of 0.7
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(256, return_sequences=True,input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.7))
model.add(LSTM(128))
model.add(Dropout(0.7))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()


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

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_9 (LSTM)                (None, 128, 256)          272384
_________________________________________________________________
dropout_9 (Dropout)          (None, 128, 256)          0
_________________________________________________________________
lstm_10 (LSTM)               (None, 128)               197120
_________________________________________________________________
```

```
dropout_10 (Dropout)         (None, 128)              0
_________________________________________________________________
dense_6 (Dense)              (None, 6)                774
=================================================================
Total params: 470,278
Trainable params: 470,278
Non-trainable params: 0
_________________________________________________________________
```

```
In [61]: history=model.fit(X_train,
                   Y_train,
                   batch_size=batch_size,
                   validation_data=(X_test, Y_test),
                   epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 253s 34ms/step - loss: 1.3047 - acc: 0.4320 - val_
Epoch 2/30
7352/7352 [==============================] - 243s 33ms/step - loss: 0.8754 - acc: 0.6017 - val_
Epoch 3/30
7352/7352 [==============================] - 241s 33ms/step - loss: 0.7189 - acc: 0.6730 - val_
Epoch 4/30
7352/7352 [==============================] - 262s 36ms/step - loss: 0.4576 - acc: 0.8409 - val_
Epoch 5/30
7352/7352 [==============================] - 281s 38ms/step - loss: 0.4332 - acc: 0.8619 - val_
Epoch 6/30
7352/7352 [==============================] - 276s 38ms/step - loss: 0.3160 - acc: 0.9033 - val_
Epoch 7/30
7352/7352 [==============================] - 246s 34ms/step - loss: 0.2249 - acc: 0.9260 - val_
Epoch 8/30
7352/7352 [==============================] - 242s 33ms/step - loss: 0.1800 - acc: 0.9353 - val_
Epoch 9/30
7352/7352 [==============================] - 242s 33ms/step - loss: 0.1874 - acc: 0.9363 - val_
Epoch 10/30
7352/7352 [==============================] - 247s 34ms/step - loss: 0.1803 - acc: 0.9408 - val_
Epoch 11/30
7352/7352 [==============================] - 246s 33ms/step - loss: 0.1970 - acc: 0.9328 - val_
Epoch 12/30
7352/7352 [==============================] - 246s 34ms/step - loss: 0.1736 - acc: 0.9414 - val_
Epoch 13/30
7352/7352 [==============================] - 242s 33ms/step - loss: 0.1683 - acc: 0.9456 - val_
Epoch 14/30
7352/7352 [==============================] - 241s 33ms/step - loss: 0.2021 - acc: 0.9404 - val_
Epoch 15/30
7352/7352 [==============================] - 241s 33ms/step - loss: 0.1623 - acc: 0.9429 - val_
Epoch 16/30
```

```
7352/7352 [==============================] - 243s 33ms/step - loss: 0.1603 - acc: 0.9442 - val_
Epoch 17/30
7352/7352 [==============================] - 241s 33ms/step - loss: 0.1441 - acc: 0.9472 - val_
Epoch 18/30
7352/7352 [==============================] - 240s 33ms/step - loss: 0.1587 - acc: 0.9411 - val_
Epoch 19/30
7352/7352 [==============================] - 240s 33ms/step - loss: 0.8286 - acc: 0.7108 - val_
Epoch 20/30
7352/7352 [==============================] - 243s 33ms/step - loss: 0.2845 - acc: 0.9134 - val_
Epoch 21/30
7352/7352 [==============================] - 251s 34ms/step - loss: 0.1800 - acc: 0.9408 - val_
Epoch 22/30
7352/7352 [==============================] - 262s 36ms/step - loss: 0.1544 - acc: 0.9434 - val_
Epoch 23/30
7352/7352 [==============================] - 273s 37ms/step - loss: 0.1632 - acc: 0.9441 - val_
Epoch 24/30
7352/7352 [==============================] - 272s 37ms/step - loss: 0.1591 - acc: 0.9445 - val_
Epoch 25/30
7352/7352 [==============================] - 277s 38ms/step - loss: 0.1490 - acc: 0.9393 - val_
Epoch 26/30
7352/7352 [==============================] - 247s 34ms/step - loss: 0.1512 - acc: 0.9457 - val_
Epoch 27/30
7352/7352 [==============================] - 245s 33ms/step - loss: 0.1707 - acc: 0.9456 - val_
Epoch 28/30
7352/7352 [==============================] - 245s 33ms/step - loss: 0.1640 - acc: 0.9494 - val_
Epoch 29/30
7352/7352 [==============================] - 245s 33ms/step - loss: 0.1988 - acc: 0.9378 - val_
Epoch 30/30
7352/7352 [==============================] - 249s 34ms/step - loss: 0.1877 - acc: 0.9395 - val_
```

```
In [63]: print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 537        0         0        0                   0
SITTING                  1      430        58        0                   0
STANDING                 0      120       412        0                   0
WALKING                  0        0         0      471                  21
WALKING_DOWNSTAIRS       0        0         0        1                 418
WALKING_UPSTAIRS         0        1         0        5                   6

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            2
STANDING                           0
WALKING                            4
```

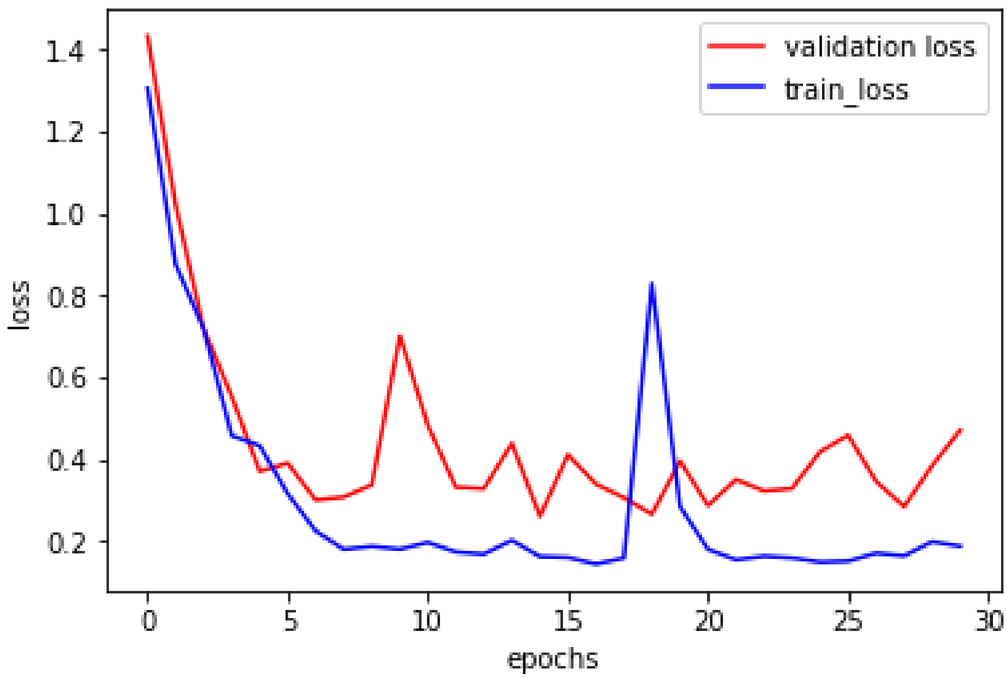
```
WALKING_DOWNSTAIRS                      1
WALKING_UPSTAIRS                      459
```

```
In [64]: score = model.evaluate(X_test, Y_test)
         score
```

```
2947/2947 [==============================] - 22s 8ms/step
```

```
Out[64]: [0.3385070033687646, 0.9253478113335596]
```

```
In [65]: train_loss=history.history['loss']
         plot_dynamic(epochs,val_loss,train_loss)
         print('Test loss',score[0])
         print('Test accuracy',score[1])
```




```
Test loss 0.3385070033687646
Test accuracy 0.9253478113335596
```

0.3 Conclusion:

Objective: To build a model that predicts the human activities such as Walking, Walking_Upstairs, Walking_Downstairs, Sitting, Standing or Laying.

1. 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.
2. We have features gained from accelerometer and gyroscope. we have raw features and also 561 engineered features by experts. We use raw features for deeplearning and 561 features for machine learning algorithms.
3. We perform EDA on the dataset to know about the distribution of the data, get more insights about the data and apply tsne to know if the data is seperable.
4. We apply machine learning algorithms like RBF SVM, Logistic Regression,Decision Tree, Random Forest for 561 features and compare .
5. We use accuracy and log loss as performance metric and find that SVM,SVC and LR gives the maximum accuracy of 96%.
6. We then apply deep learning algorithms (LSTM) on initial features and get an accuracy of 88%.
7. We try to improve the score by hyperparameter tuning the LSTM model.

In [67]:
```
from prettytable import PrettyTable

x=PrettyTable()

x.field_names=['Algorithm','LSTM-Layers','Hidden Layers','Dropout','Accuracy']
x.add_row(["LSTM","1",64,0.5, 0.899])
x.add_row(["LSTM","1",128,0.4, 0.906])
x.add_row(["LSTM","1",256,0.4, 0.904])
x.add_row(["LSTM","2",'256,128',0.7, 0.9253])


print(x)
```

```
+-----------+-------------+---------------+---------+----------+
| Algorithm | LSTM-Layers | Hidden Layers | Dropout | Accuracy |
+-----------+-------------+---------------+---------+----------+
|    LSTM   |      1      |       64      |   0.5   |  0.899   |
|    LSTM   |      1      |      128      |   0.4   |  0.906   |
|    LSTM   |      1      |      256      |   0.4   |  0.904   |
|    LSTM   |      2      |    256,128    |   0.7   |  0.9253  |
+-----------+-------------+---------------+---------+----------+
```

In [ ]: