LSTM Hyperparameter Tuning

Step by step Process

1.Lets start as we know 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.So after performing lots of EDA and visulization on dataset we have performed both Classical ML models and Deep learning RNN models and we know we haved two types of data: 1.Expert Engg features - perform Classical ML models 2.Raw time series data - perform Deep learning RNN models

3.After doing all above in this in this we will performs some hyperparameter tuning on our deep learning models.

Lets Start working :

In [1]:

```
# Importing Libraries
```

In [26]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
import seaborn as sns
from sklearn.metrics import confusion_matrix
```

In [2]:

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

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

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

Data

In [3]:

```
# Data directory
DATADIR = 'UCI_HAR_Dataset'
```

In [4]:

```
# Raw data signals
# Signals are from Accelerometer and Gyroscope
# The signals are in x,y,z directions
# Sensor signals are filtered to have only body acceleration
# excluding the acceleration due to gravity
# Triaxial acceleration from the accelerometer is total acceleration
```

```python
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 [5]:

```python
# 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 [6]:

```python
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 [7]:

```python
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 [9]:

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

In [10]:

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

```
)
```

In [12]:

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

In [13]:

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

In [33]:

```
# utility function
def plt_dynamic(x, vy, ty):
    plt.figure(figsize=(10,5))
    plt.plot(x, vy, 'b', label="Validation Loss")
    plt.plot(x, ty, 'r', label="Train Loss")
    plt.xlabel('Epochs')
    plt.ylabel('Crossentropy Loss')
    plt.title('Crossentropy Loss VS Epochs')
    plt.legend()
    plt.grid()
    plt.show()
```

In [14]:

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

In [15]:

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

In [16]:

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

In [17]:

```
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

1 LSTM with 32 Units

In [46]:

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

# Compiling the model
model1.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])

# Training the model
history1 = model1.fit(X_train, Y_train, batch_size=batch_size,validation_data=(X_test, Y_test),epoc
hs=epochs)
```

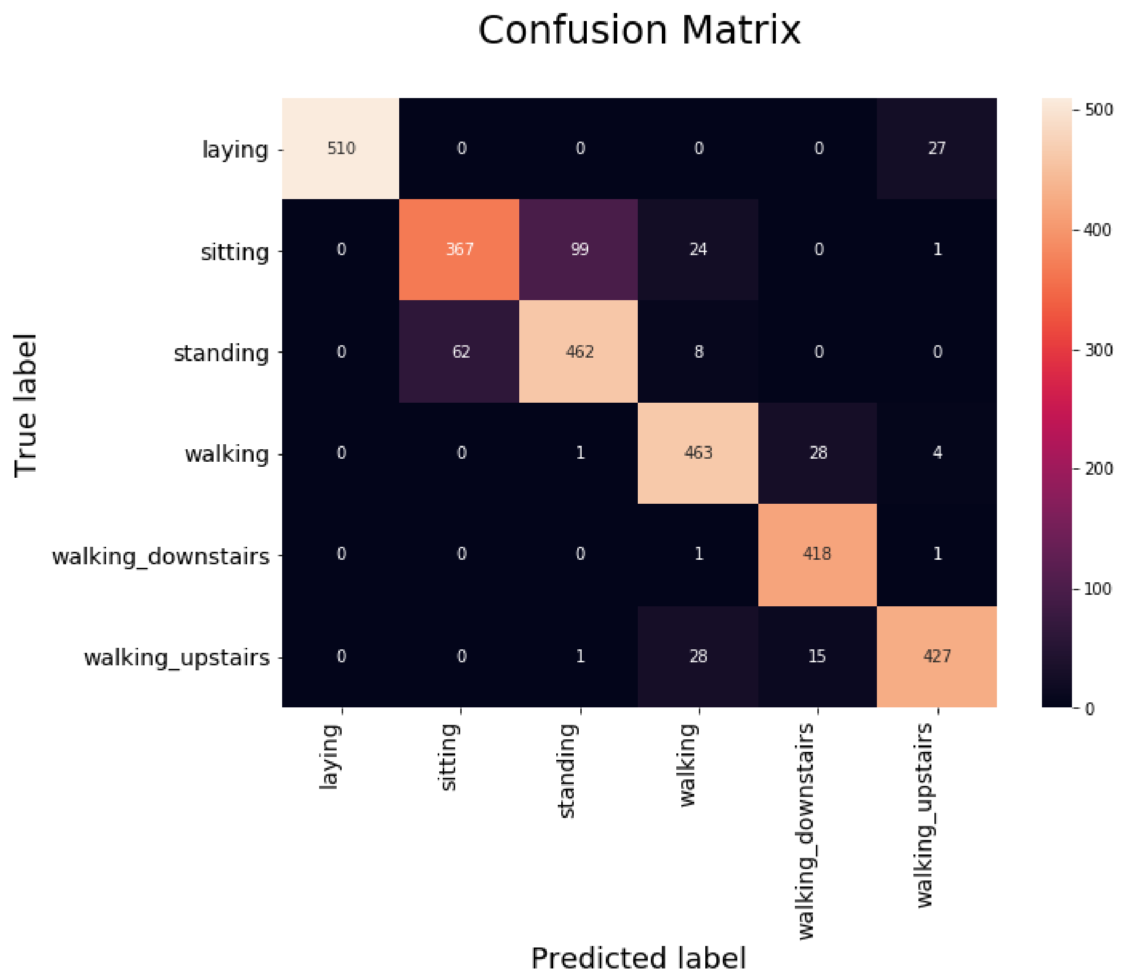
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_8 (LSTM)                (None, 32)                5376      
_________________________________________________________________
dropout_8 (Dropout)          (None, 32)                0         
_________________________________________________________________
dense_6 (Dense)              (None, 6)                 198       
=================================================================
Total params: 5,574
Trainable params: 5,574
Non-trainable params: 0
_________________________________________________________________
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 37s 5ms/step - loss: 1.3700 - acc: 0.4323 - val_loss:
1.1422 - val_acc: 0.5019
Epoch 2/30
7352/7352 [==============================] - 34s 5ms/step - loss: 1.0386 - acc: 0.5223 - val_loss:
0.8877 - val_acc: 0.6023
Epoch 3/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.8403 - acc: 0.6094 - val_loss:
0.8038 - val_acc: 0.6410
Epoch 4/30
7352/7352 [==============================] - 33s 4ms/step - loss: 0.7603 - acc: 0.6341 - val_loss:
0.7887 - val_acc: 0.6098
Epoch 5/30
7352/7352 [==============================] - 33s 5ms/step - loss: 0.7190 - acc: 0.6498 - val_loss:
0.7401 - val_acc: 0.6142
Epoch 6/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.7620 - acc: 0.6484 - val_loss:
0.7331 - val_acc: 0.6206
Epoch 7/30
7352/7352 [==============================] - 36s 5ms/step - loss: 0.8683 - acc: 0.6326 - val_loss:
0.7364 - val_acc: 0.6481
Epoch 8/30
7352/7352 [==============================] - 36s 5ms/step - loss: 0.6668 - acc: 0.7061 - val_loss:
0.6517 - val_acc: 0.7024
Epoch 9/30
7352/7352 [==============================] - 36s 5ms/step - loss: 0.7004 - acc: 0.7150 - val_loss:
0.9810 - val_acc: 0.6284
Epoch 10/30
7352/7352 [==============================] - 37s 5ms/step - loss: 0.6010 - acc: 0.7666 - val_loss:
0.6001 - val_acc: 0.7608
Epoch 11/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.5037 - acc: 0.8108 - val_loss:
0.5779 - val_acc: 0.7893
Epoch 12/30
7352/7352 [==============================] - 36s 5ms/step - loss: 0.4216 - acc: 0.8576 - val_loss:
0.4980 - val_acc: 0.8436
Epoch 13/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.3439 - acc: 0.8938 - val_loss:
0.5197 - val_acc: 0.8493
Epoch 14/30
7352/7352 [==============================] - 34s 5ms/step - loss: 0.2921 - acc: 0.9064 - val_loss:
0.4793 - val_acc: 0.8670
Epoch 15/30
```

```
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2762 - acc: 0.9197 - val_loss: 0.5259 - val_acc: 0.8470
Epoch 16/30
7352/7352 [==============================] - 33s 5ms/step - loss: 0.2465 - acc: 0.9271 - val_loss: 0.8381 - val_acc: 0.8358
Epoch 17/30
7352/7352 [==============================] - 36s 5ms/step - loss: 0.2639 - acc: 0.9226 - val_loss: 0.4094 - val_acc: 0.8816
Epoch 18/30
7352/7352 [==============================] - 36s 5ms/step - loss: 0.3222 - acc: 0.9102 - val_loss: 0.3858 - val_acc: 0.8914
Epoch 19/30
7352/7352 [==============================] - 36s 5ms/step - loss: 0.2124 - acc: 0.9319 - val_loss: 0.4049 - val_acc: 0.8897
Epoch 20/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.2017 - acc: 0.9319 - val_loss: 0.3426 - val_acc: 0.8955
Epoch 21/30
7352/7352 [==============================] - 33s 4ms/step - loss: 0.3140 - acc: 0.9037 - val_loss: 0.5085 - val_acc: 0.8897
Epoch 22/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2601 - acc: 0.9199 - val_loss: 0.3746 - val_acc: 0.9009
Epoch 23/30
7352/7352 [==============================] - 34s 5ms/step - loss: 0.1865 - acc: 0.9378 - val_loss: 0.6343 - val_acc: 0.8700
Epoch 24/30
7352/7352 [==============================] - 37s 5ms/step - loss: 0.1846 - acc: 0.9393 - val_loss: 0.3892 - val_acc: 0.8962
Epoch 25/30
7352/7352 [==============================] - 36s 5ms/step - loss: 0.1733 - acc: 0.9396 - val_loss: 0.3167 - val_acc: 0.8962
Epoch 26/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.1716 - acc: 0.9353 - val_loss: 0.2777 - val_acc: 0.9030
Epoch 27/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.1598 - acc: 0.9448 - val_loss: 0.3779 - val_acc: 0.8819
Epoch 28/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.1569 - acc: 0.9434 - val_loss: 0.2861 - val_acc: 0.9108
Epoch 29/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.1624 - acc: 0.9389 - val_loss: 0.3922 - val_acc: 0.8918
Epoch 30/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.1772 - acc: 0.9437 - val_loss: 0.4369 - val_acc: 0.8982
```

In [47]:

```python
# Final evaluation of the model
scores = model1.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores[0]))
print("Test Accuracy: %f%%" % (scores[1]*100))

# Confusion Matrix
Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_test, axis=1)])
Y_predictions = pd.Series([ACTIVITIES[y] for y in np.argmax(model1.predict(X_test), axis=1)])

# Code for drawing seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_names, columns=class_names )
fig = plt.figure(figsize=(10,7))
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")

# Setting tick labels for heatmap
heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotation=0, ha='right', fontsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=90, ha='right', fontsize=14)
plt.ylabel('True label',size=18)
plt.xlabel('Predicted label',size=18)
plt.title("Confusion Matrix\n",size=24)
plt.show()
```

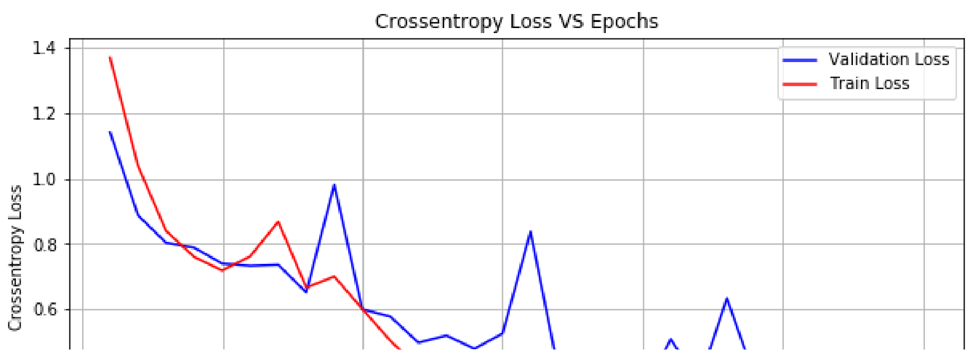
```
Test Score: 0.436930
Test Accuracy: 89.820156%
```

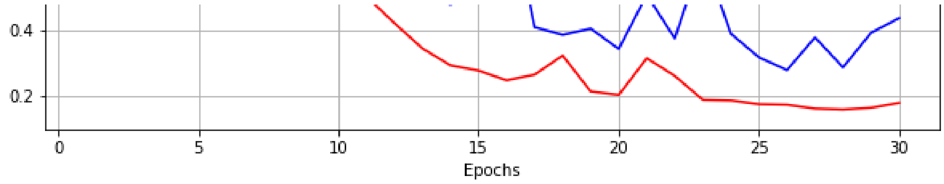


In [48]:

```python
# Final evaluation of the model
scores = model1.evaluate(X_test, Y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))

# Test and train accuracy of the model
model_1_test = scores[1]
model_1_train = max(history1.history['acc'])

# Plotting Train and Test Loss VS no. of epochs
# list of epoch numbers
x = list(range(1,31))

# Validation loss
vy = history1.history['val_loss']
# Training loss
ty = history1.history['loss']

# Calling the function to draw the plot
plt_dynamic(x, vy, ty)
```

Accuracy: 89.82%





1 LSTM with 42 Units

In [30]:

```
# Initiliazing the sequential model
model2 = Sequential()
# Configuring the parameters
model2.add(LSTM(42, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model2.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model2.add(Dense(n_classes, activation='sigmoid'))
print(model2.summary())

# Compiling the model
model2.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])

# Training the model
history2 = model2.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y_test),epochs
=epochs)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_2 (LSTM)                (None, 42)                8736      
_________________________________________________________________
dropout_2 (Dropout)          (None, 42)                0         
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 258       
=================================================================
Total params: 8,994
Trainable params: 8,994
Non-trainable params: 0
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 39s 5ms/step - loss: 1.2947 - acc: 0.4499 - val_loss:
1.1957 - val_acc: 0.4571
Epoch 2/30
7352/7352 [==============================] - 39s 5ms/step - loss: 1.0003 - acc: 0.5594 - val_loss:
0.8970 - val_acc: 0.6196
Epoch 3/30
7352/7352 [==============================] - 36s 5ms/step - loss: 0.7672 - acc: 0.6571 - val_loss:
0.7434 - val_acc: 0.6634
Epoch 4/30
7352/7352 [==============================] - 34s 5ms/step - loss: 0.7791 - acc: 0.6485 - val_loss:
0.7595 - val_acc: 0.6970
Epoch 5/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.6105 - acc: 0.7588 - val_loss:
0.6652 - val_acc: 0.7387
Epoch 6/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.5855 - acc: 0.7833 - val_loss:
0.5378 - val_acc: 0.7781
Epoch 7/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.6056 - acc: 0.8017 - val_loss:
0.5069 - val_acc: 0.8633
Epoch 8/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.3929 - acc: 0.8848 - val_loss:
0.5762 - val_acc: 0.8300
Epoch 9/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.3592 - acc: 0.8962 - val_loss:
0.4852 - val_acc: 0.8507
Epoch 10/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.2607 - acc: 0.9219 - val_loss:
0.3340 - val_acc: 0.8979
Epoch 11/30
```

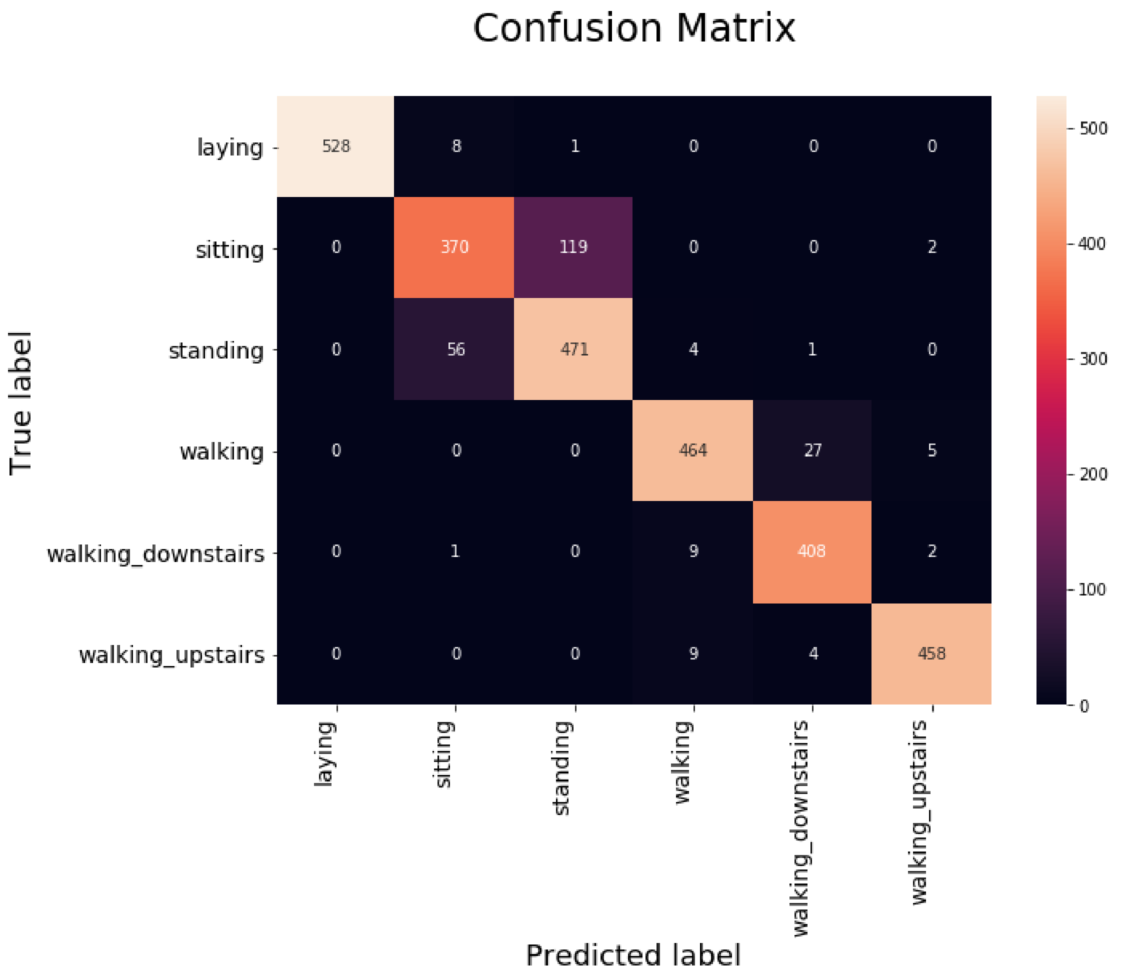
```
7352/7352 [==============================] - 34s 5ms/step - loss: 0.2327 - acc: 0.9280 - val_loss:
0.3171 - val_acc: 0.8955
Epoch 12/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.2309 - acc: 0.9263 - val_loss:
0.8502 - val_acc: 0.8246
Epoch 13/30
7352/7352 [==============================] - 38s 5ms/step - loss: 0.2203 - acc: 0.9282 - val_loss:
0.4000 - val_acc: 0.8935
Epoch 14/30
7352/7352 [==============================] - 38s 5ms/step - loss: 0.2015 - acc: 0.9361 - val_loss:
0.3798 - val_acc: 0.8880
Epoch 15/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.1964 - acc: 0.9399 - val_loss:
0.3339 - val_acc: 0.9104
Epoch 16/30
7352/7352 [==============================] - 38s 5ms/step - loss: 0.2422 - acc: 0.9295 - val_loss:
0.3901 - val_acc: 0.8982
Epoch 17/30
7352/7352 [==============================] - 37s 5ms/step - loss: 0.1763 - acc: 0.9397 - val_loss:
0.2982 - val_acc: 0.9118
Epoch 18/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.1564 - acc: 0.9425 - val_loss:
0.4010 - val_acc: 0.9016
Epoch 19/30
7352/7352 [==============================] - 38s 5ms/step - loss: 0.1671 - acc: 0.9442 - val_loss:
0.3645 - val_acc: 0.9019
Epoch 20/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.1623 - acc: 0.9421 - val_loss:
0.3848 - val_acc: 0.9114
Epoch 21/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.1604 - acc: 0.9460 - val_loss:
0.4165 - val_acc: 0.9121
Epoch 22/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.1496 - acc: 0.9464 - val_loss:
0.3730 - val_acc: 0.9158
Epoch 23/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1710 - acc: 0.9470 - val_loss:
0.3039 - val_acc: 0.9125
Epoch 24/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.1478 - acc: 0.9463 - val_loss:
0.3399 - val_acc: 0.9077
Epoch 25/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1519 - acc: 0.9467 - val_loss:
0.4050 - val_acc: 0.9077
Epoch 26/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1525 - acc: 0.9468 - val_loss:
0.2773 - val_acc: 0.9145
Epoch 27/30
7352/7352 [==============================] - 38s 5ms/step - loss: 0.1489 - acc: 0.9513 - val_loss:
0.4474 - val_acc: 0.9026
Epoch 28/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1361 - acc: 0.9489 - val_loss:
0.2907 - val_acc: 0.9203
Epoch 29/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.1542 - acc: 0.9510 - val_loss:
0.4495 - val_acc: 0.9084
Epoch 30/30
7352/7352 [==============================] - 37s 5ms/step - loss: 0.1580 - acc: 0.9476 - val_loss:
0.4272 - val_acc: 0.9158
```

In [31]:

```python
# Final evaluation of the model
scores2 = model2.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores2[0]))
print("Test Accuracy: %f%%" % (scores2[1]*100))

# Confusion Matrix
Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_test, axis=1)])
Y_predictions = pd.Series([ACTIVITIES[y] for y in np.argmax(model2.predict(X_test), axis=1)])

# Code for drawing seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_names, columns=class
_names )
fig = plt.figure(figsize=(10,7))
```

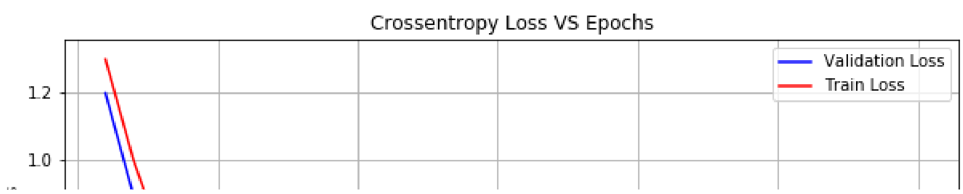
```
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")

# Setting tick labels for heatmap
heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotation=0, ha='right', fontsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=90, ha='right', fontsize=14)
plt.ylabel('True label',size=18)
plt.xlabel('Predicted label',size=18)
plt.title("Confusion Matrix\n",size=24)
plt.show()
```

```
Test Score: 0.427173
Test Accuracy: 91.584662%
```



In [34]:

```
# Test and train accuracy of the model
model_2_test = scores2[1]
model_2_train = max(history2.history['acc'])

# Plotting Train and Test Loss VS no. of epochs
# list of epoch numbers
x = list(range(1,31))

# Validation loss
vy = history2.history['val_loss']
# Training loss
ty = history2.history['loss']

# Calling the function to draw the plot
plt_dynamic(x, vy, ty)
```



1 LSTM with 64 Units with large dropout

In [37]:

```
# Initiliazing the sequential model
model3 = Sequential()
# Configuring the parameters
model3.add(LSTM(64, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model3.add(Dropout(0.7))
# Adding a dense output layer with sigmoid activation
model3.add(Dense(n_classes, activation='sigmoid'))
print(model3.summary())

# Compiling the model
model3.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])

# Training the model
history3 = model3.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y_test),epochs
=epochs)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 64)                18944
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 390
=================================================================
Total params: 19,334
Trainable params: 19,334
Non-trainable params: 0
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 44s 6ms/step - loss: 1.3289 - acc: 0.4153 - val_loss:
1.1952 - val_acc: 0.4842
Epoch 2/30
7352/7352 [==============================] - 40s 5ms/step - loss: 1.0808 - acc: 0.5403 - val_loss:
0.9875 - val_acc: 0.5840
Epoch 3/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.7958 - acc: 0.6654 - val_loss:
0.8333 - val_acc: 0.6464
Epoch 4/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.6894 - acc: 0.7082 - val_loss:
0.7353 - val_acc: 0.6983
Epoch 5/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.6138 - acc: 0.7357 - val_loss:
0.8443 - val_acc: 0.6882
Epoch 6/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.5816 - acc: 0.7579 - val_loss:
0.6961 - val_acc: 0.7577
Epoch 7/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.5487 - acc: 0.7843 - val_loss:
1.4815 - val_acc: 0.6057
Epoch 8/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.5442 - acc: 0.8078 - val_loss:
0.7764 - val_acc: 0.7550
Epoch 9/30
```

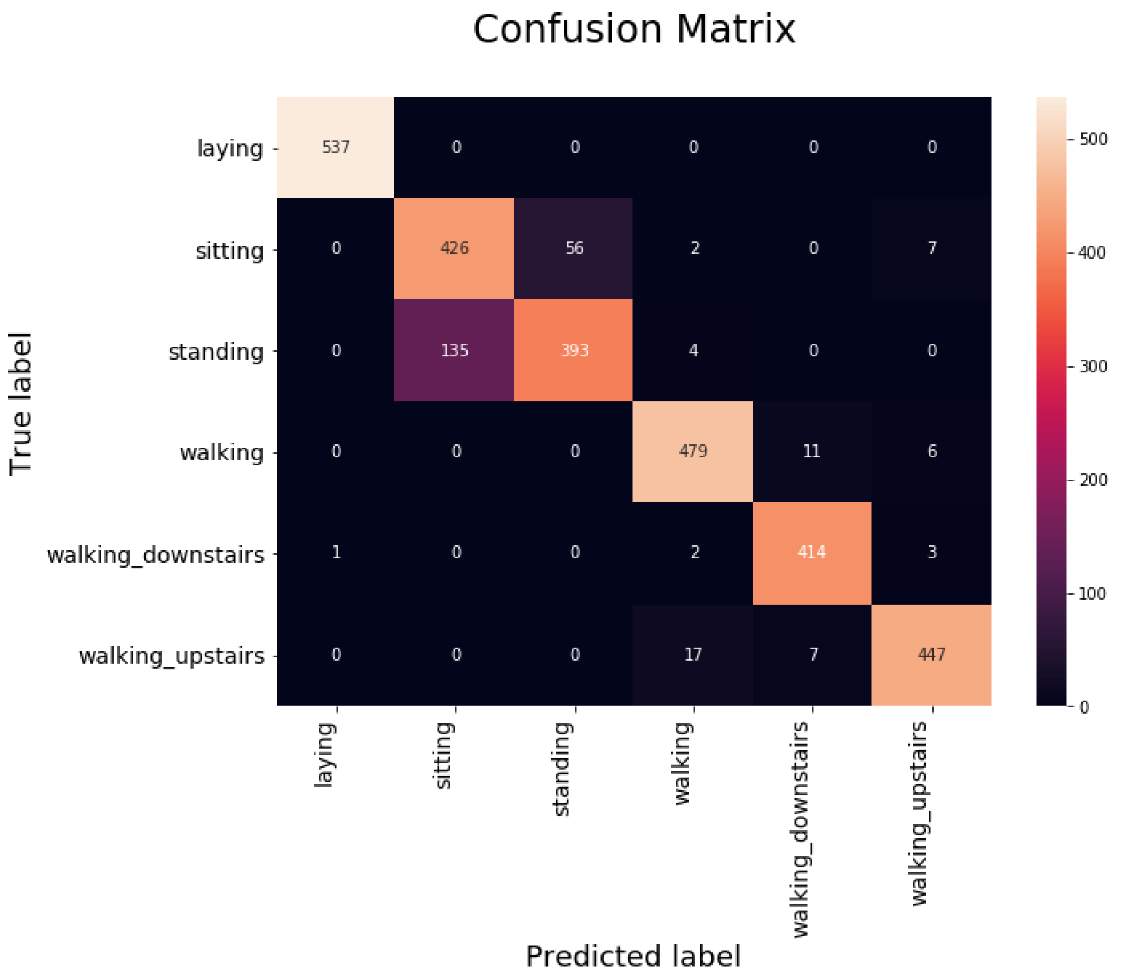
```
7352/7352 [==============================] - 35s 5ms/step - loss: 0.4581 - acc: 0.8562 - val_loss: 0.5762 - val_acc: 0.8517
Epoch 10/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.4103 - acc: 0.8826 - val_loss: 0.5640 - val_acc: 0.8436
Epoch 11/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.3345 - acc: 0.9044 - val_loss: 0.4452 - val_acc: 0.8819
Epoch 12/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.3613 - acc: 0.9025 - val_loss: 0.3402 - val_acc: 0.9009
Epoch 13/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.2813 - acc: 0.9102 - val_loss: 0.3724 - val_acc: 0.9013
Epoch 14/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.2385 - acc: 0.9305 - val_loss: 0.3165 - val_acc: 0.9080
Epoch 15/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.2228 - acc: 0.9340 - val_loss: 0.3286 - val_acc: 0.9145
Epoch 16/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.2520 - acc: 0.9272 - val_loss: 0.8850 - val_acc: 0.8402
Epoch 17/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.2275 - acc: 0.9338 - val_loss: 0.3064 - val_acc: 0.9114
Epoch 18/30
7352/7352 [==============================] - 37s 5ms/step - loss: 0.1871 - acc: 0.9391 - val_loss: 0.3734 - val_acc: 0.9135
Epoch 19/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.2178 - acc: 0.9353 - val_loss: 0.3587 - val_acc: 0.9101
Epoch 20/30
7352/7352 [==============================] - 36s 5ms/step - loss: 0.1981 - acc: 0.9410 - val_loss: 0.3054 - val_acc: 0.9084
Epoch 21/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.1817 - acc: 0.9444 - val_loss: 0.3525 - val_acc: 0.9030
Epoch 22/30
7352/7352 [==============================] - 36s 5ms/step - loss: 0.2005 - acc: 0.9410 - val_loss: 0.2881 - val_acc: 0.9264
Epoch 23/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.1742 - acc: 0.9431 - val_loss: 0.4975 - val_acc: 0.9043
Epoch 24/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.1950 - acc: 0.9422 - val_loss: 0.3071 - val_acc: 0.9189
Epoch 25/30
7352/7352 [==============================] - 36s 5ms/step - loss: 0.1665 - acc: 0.9457 - val_loss: 0.4143 - val_acc: 0.8904
Epoch 26/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.1777 - acc: 0.9453 - val_loss: 0.7865 - val_acc: 0.8761
Epoch 27/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.1783 - acc: 0.9440 - val_loss: 0.3182 - val_acc: 0.9148
Epoch 28/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.1646 - acc: 0.9483 - val_loss: 0.2716 - val_acc: 0.9247
Epoch 29/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.2131 - acc: 0.9422 - val_loss: 0.3630 - val_acc: 0.9046
Epoch 30/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.2133 - acc: 0.9359 - val_loss: 0.2799 - val_acc: 0.9148
```

In [38]:

```python
# Final evaluation of the model
scores3 = model3.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores3[0]))
print("Test Accuracy: %f%%" % (scores3[1]*100))

# Confusion Matrix
Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_test, axis=1)])
Y_predictions = pd.Series([ACTIVITIES[y] for y in np.argmax(model3.predict(X_test), axis=1)])
```

```
# Code for drawing seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_names, columns=class
_names )
fig = plt.figure(figsize=(10,7))
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")

# Setting tick labels for heatmap
heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotation=0, ha='right', fontsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=90, ha='right', fontsize=14)
plt.ylabel('True label',size=18)
plt.xlabel('Predicted label',size=18)
plt.title("Confusion Matrix\n",size=24)
plt.show()
```

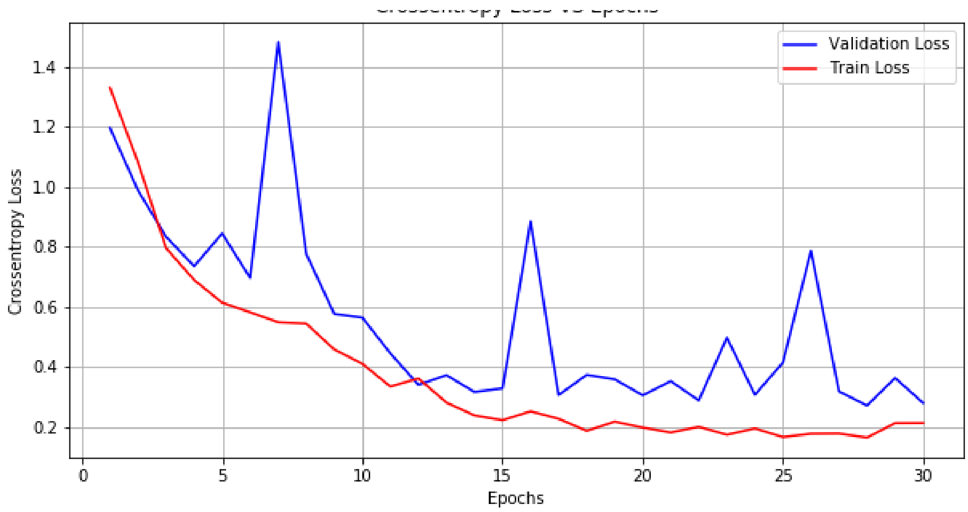
```
Test Score: 0.279943
Test Accuracy: 91.482864%
```



In [39]:

```
# Test and train accuracy of the model
model_3_test = scores3[1]
model_3_train = max(history3.history['acc'])

# Plotting Train and Test Loss VS no. of epochs
# list of epoch numbers
x = list(range(1,31))

# Validation loss
vy = history3.history['val_loss']
# Training loss
ty = history3.history['loss']

# Calling the function to draw the plot
plt_dynamic(x, vy, ty)
```

Crossentropy Loss VS Epochs



2 LSTM Layers with 32 Units

In [40]:

```
# Initiliazing the sequential model
model4 = Sequential()
# Configuring the parameters
model4.add(LSTM(32,return_sequences=True, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model4.add(Dropout(0.5))

# Configuring the parameters
model4.add(LSTM(32))
# Adding a dropout layer
model4.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model4.add(Dense(n_classes, activation='sigmoid'))
print(model4.summary())

# Compiling the model
model4.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])

# Training the model
history4 = model4.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y_test),epochs
=epochs)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_4 (LSTM)                (None, 128, 32)           5376
_________________________________________________________________
dropout_4 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
lstm_5 (LSTM)                (None, 32)                8320
_________________________________________________________________
dropout_5 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 198
=================================================================
Total params: 13,894
Trainable params: 13,894
Non-trainable params: 0
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 70s 9ms/step - loss: 1.2283 - acc: 0.4827 - val_loss:
1.0618 - val_acc: 0.4866
Epoch 2/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.8534 - acc: 0.6099 - val_loss:
1.0135 - val_acc: 0.5456
Epoch 3/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.7211 - acc: 0.6493 - val_loss:
0.7843 - val_acc: 0.6094
Epoch 4/30
```

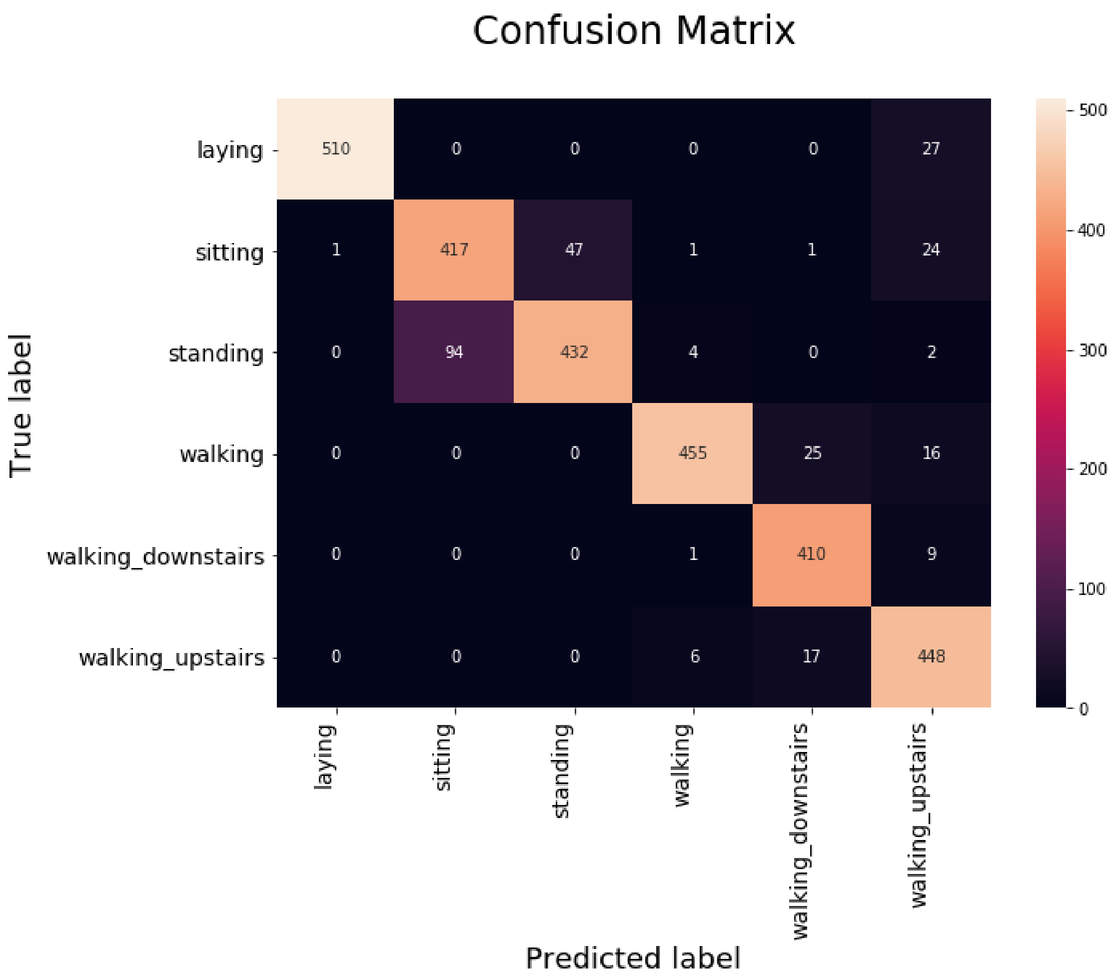
```
7352/7352 [==============================] - 65s 9ms/step - loss: 0.6762 - acc: 0.6669 - val_loss:
0.7229 - val_acc: 0.6179
Epoch 5/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.6600 - acc: 0.6647 - val_loss:
0.6788 - val_acc: 0.6210
Epoch 6/30
7352/7352 [==============================] - 66s 9ms/step - loss: 0.5906 - acc: 0.7160 - val_loss:
0.6746 - val_acc: 0.7296
Epoch 7/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.5347 - acc: 0.7671 - val_loss:
0.5615 - val_acc: 0.7370
Epoch 8/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.4404 - acc: 0.7961 - val_loss:
0.5352 - val_acc: 0.7384
Epoch 9/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.3666 - acc: 0.8550 - val_loss:
0.5541 - val_acc: 0.8405
Epoch 10/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.3070 - acc: 0.9064 - val_loss:
0.6806 - val_acc: 0.8079
Epoch 11/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.2319 - acc: 0.9298 - val_loss:
0.4894 - val_acc: 0.8748
Epoch 12/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.2052 - acc: 0.9350 - val_loss:
0.5480 - val_acc: 0.8755
Epoch 13/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.1892 - acc: 0.9414 - val_loss:
0.4811 - val_acc: 0.8761
Epoch 14/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.1807 - acc: 0.9387 - val_loss:
0.4882 - val_acc: 0.8728
Epoch 15/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.1653 - acc: 0.9460 - val_loss:
0.4981 - val_acc: 0.8799
Epoch 16/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.1518 - acc: 0.9457 - val_loss:
0.4923 - val_acc: 0.8941
Epoch 17/30
7352/7352 [==============================] - 66s 9ms/step - loss: 0.1593 - acc: 0.9479 - val_loss:
0.5772 - val_acc: 0.8884
Epoch 18/30
7352/7352 [==============================] - 69s 9ms/step - loss: 0.1574 - acc: 0.9436 - val_loss:
0.4696 - val_acc: 0.8924
Epoch 19/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.1415 - acc: 0.9502 - val_loss
: 0.4920 - val_acc: 0.8938
Epoch 20/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.1528 - acc: 0.9502 - val_loss:
0.5691 - val_acc: 0.8928
Epoch 21/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.1461 - acc: 0.9494 - val_loss:
0.6179 - val_acc: 0.8856
Epoch 22/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.1449 - acc: 0.9531 - val_loss:
0.4345 - val_acc: 0.9019
Epoch 23/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.1583 - acc: 0.9486 - val_loss:
0.5218 - val_acc: 0.8870
Epoch 24/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.1414 - acc: 0.9494 - val_loss:
0.4950 - val_acc: 0.8985
Epoch 25/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.1645 - acc: 0.9480 - val_loss:
0.4609 - val_acc: 0.9002
Epoch 26/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.1409 - acc: 0.9480 - val_loss:
0.5459 - val_acc: 0.8867
Epoch 27/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.1572 - acc: 0.9461 - val_loss:
0.5245 - val_acc: 0.8907
Epoch 28/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.1379 - acc: 0.9527 - val_loss:
0.5630 - val_acc: 0.8996
Epoch 29/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.1426 - acc: 0.9529 - val_loss:
0.6665 - val_acc: 0.8887
```

```
Epoch 30/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.1608 - acc: 0.9509 - val_loss:
0.5688 - val_acc: 0.9067
```

In [41]:

```python
# Final evaluation of the model
scores4 = model4.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores4[0]))
print("Test Accuracy: %f%%" % (scores4[1]*100))

# Confusion Matrix
Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_test, axis=1)])
Y_predictions = pd.Series([ACTIVITIES[y] for y in np.argmax(model4.predict(X_test), axis=1)])

# Code for drawing seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_names, columns=class
_names )
fig = plt.figure(figsize=(10,7))
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")

# Setting tick labels for heatmap
heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotation=0, ha='right', fontsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=90, ha='right', fontsize=14)
plt.ylabel('True label',size=18)
plt.xlabel('Predicted label',size=18)
plt.title("Confusion Matrix\n",size=24)
plt.show()
```

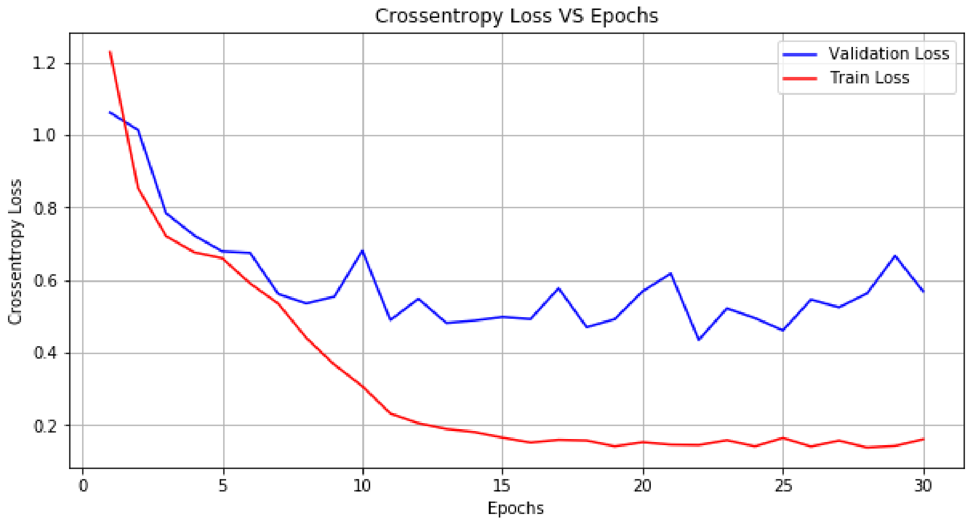
```
Test Score: 0.568783
Test Accuracy: 90.668476%
```



In [42]:

```python
# Test and train accuracy of the model
model_4_test = scores4[1]
model_4_train = max(history4.history['acc'])
```

```
model_4_train = max(history4.history['acc'])

# Plotting Train and Test Loss VS no. of epochs
# list of epoch numbers
x = list(range(1,31))

# Validation loss
vy = history4.history['val_loss']
# Training loss
ty = history4.history['loss']

# Calling the function to draw the plot
plt_dynamic(x, vy, ty)
```



2 LSTM Layers with 64 Units with larger dropout

In [43]:

```
# Initiliazing the sequential model
model5 = Sequential()
# Configuring the parameters
model5.add(LSTM(64,return_sequences=True, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model5.add(Dropout(0.7))

# Configuring the parameters
model5.add(LSTM(64))
# Adding a dropout layer
model5.add(Dropout(0.7))
# Adding a dense output layer with sigmoid activation
model5.add(Dense(n_classes, activation='sigmoid'))
print(model5.summary())

# Compiling the model
model5.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])

# Training the model
history5 = model5.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y_test),epochs
=epochs)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_6 (LSTM)                (None, 128, 64)           18944     
_________________________________________________________________
dropout_6 (Dropout)          (None, 128, 64)           0         
_________________________________________________________________
lstm_7 (LSTM)                (None, 64)                33024     
_________________________________________________________________
dropout_7 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_5 (Dense)              (None, 6)                 390       
=================================================================
Total params: 52,358
```

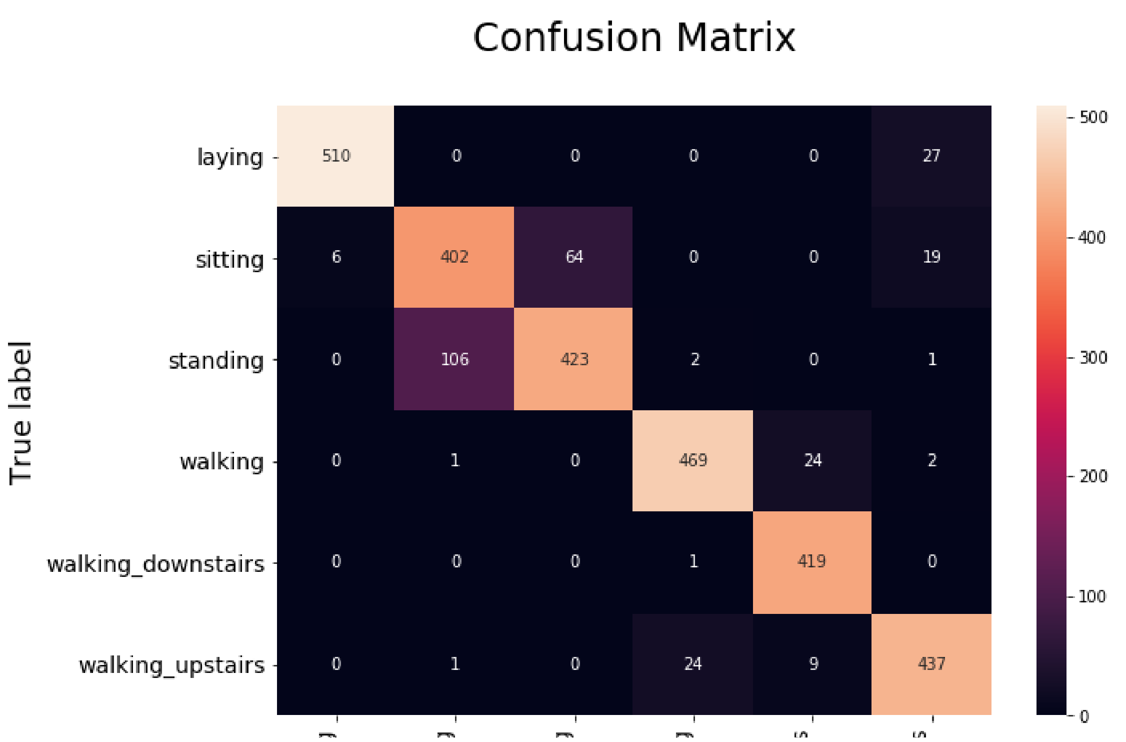
```
Trainable params: 52,358
Non-trainable params: 0
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 86s 12ms/step - loss: 1.2639 - acc: 0.4765 - val_loss
: 1.0098 - val_acc: 0.5738
Epoch 2/30
7352/7352 [==============================] - 83s 11ms/step - loss: 0.8303 - acc: 0.6251 - val_loss
: 0.8178 - val_acc: 0.6423
Epoch 3/30
7352/7352 [==============================] - 86s 12ms/step - loss: 0.7374 - acc: 0.6699 - val_loss
: 0.8662 - val_acc: 0.6023
Epoch 4/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.5859 - acc: 0.7743 - val_loss
: 0.5922 - val_acc: 0.8059
Epoch 5/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.4481 - acc: 0.8694 - val_loss
: 0.4113 - val_acc: 0.8741
Epoch 6/30
7352/7352 [==============================] - 83s 11ms/step - loss: 0.3208 - acc: 0.9082 - val_loss
: 0.5145 - val_acc: 0.8656
Epoch 7/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.2599 - acc: 0.9218 - val_loss
: 0.4321 - val_acc: 0.8880
Epoch 8/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.2220 - acc: 0.9272 - val_loss
: 0.4801 - val_acc: 0.8802
Epoch 9/30
7352/7352 [==============================] - 84s 11ms/step - loss: 0.2299 - acc: 0.9287 - val_loss
: 0.4148 - val_acc: 0.8918
Epoch 10/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1985 - acc: 0.9361 - val_loss
: 0.3505 - val_acc: 0.9148
Epoch 11/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1831 - acc: 0.9397 - val_loss
: 0.3722 - val_acc: 0.9165
Epoch 12/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1715 - acc: 0.9434 - val_loss
: 0.4501 - val_acc: 0.9094
Epoch 13/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1774 - acc: 0.9422 - val_loss
: 0.4963 - val_acc: 0.8985
Epoch 14/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1588 - acc: 0.9445 - val_loss
: 0.5815 - val_acc: 0.9033
Epoch 15/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1591 - acc: 0.9455 - val_loss
: 0.4792 - val_acc: 0.8962
Epoch 16/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1727 - acc: 0.9460 - val_loss
: 0.5234 - val_acc: 0.9111
Epoch 17/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1718 - acc: 0.9467 - val_loss
: 0.5157 - val_acc: 0.9067
Epoch 18/30
7352/7352 [==============================] - 86s 12ms/step - loss: 0.1532 - acc: 0.9490 - val_loss
: 0.5536 - val_acc: 0.9033
Epoch 19/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1711 - acc: 0.9474 - val_loss
: 0.5503 - val_acc: 0.9213
Epoch 20/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1644 - acc: 0.9452 - val_loss
: 0.5486 - val_acc: 0.9080
Epoch 21/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1774 - acc: 0.9472 - val_loss
: 0.4828 - val_acc: 0.9114
Epoch 22/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1609 - acc: 0.9484 - val_loss
: 0.3316 - val_acc: 0.9237
Epoch 23/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1655 - acc: 0.9471 - val_loss
: 0.3622 - val_acc: 0.9213
Epoch 24/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1534 - acc: 0.9467 - val_loss
: 0.4632 - val_acc: 0.9179
```

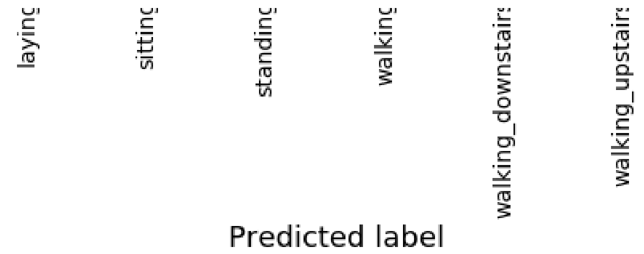
```
Epoch 25/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1723 - acc: 0.9465 - val_loss
: 0.4597 - val_acc: 0.9121
Epoch 26/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1515 - acc: 0.9487 - val_loss
: 0.4589 - val_acc: 0.9182
Epoch 27/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.1550 - acc: 0.9442 - val_loss
: 0.3841 - val_acc: 0.9148
Epoch 28/30
7352/7352 [==============================] - 86s 12ms/step - loss: 0.1519 - acc: 0.9468 - val_loss
: 0.5507 - val_acc: 0.9053
Epoch 29/30
7352/7352 [==============================] - 86s 12ms/step - loss: 0.1488 - acc: 0.9509 - val_loss
: 0.6676 - val_acc: 0.9009
Epoch 30/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1295 - acc: 0.9520 - val_loss
: 0.7193 - val_acc: 0.9026
```

In [44]:

```
# Final evaluation of the model
scores5 = model5.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores5[0]))
print("Test Accuracy: %f%%" % (scores5[1]*100))

# Confusion Matrix
Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_test, axis=1)])
Y_predictions = pd.Series([ACTIVITIES[y] for y in np.argmax(model5.predict(X_test), axis=1)])

# Code for drawing seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_names, columns=class
_names )
fig = plt.figure(figsize=(10,7))
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")

# Setting tick labels for heatmap
heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotation=0, ha='right', fontsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=90, ha='right', fontsize=14)
plt.ylabel('True label',size=18)
plt.xlabel('Predicted label',size=18)
plt.title("Confusion Matrix\n",size=24)
plt.show()
```

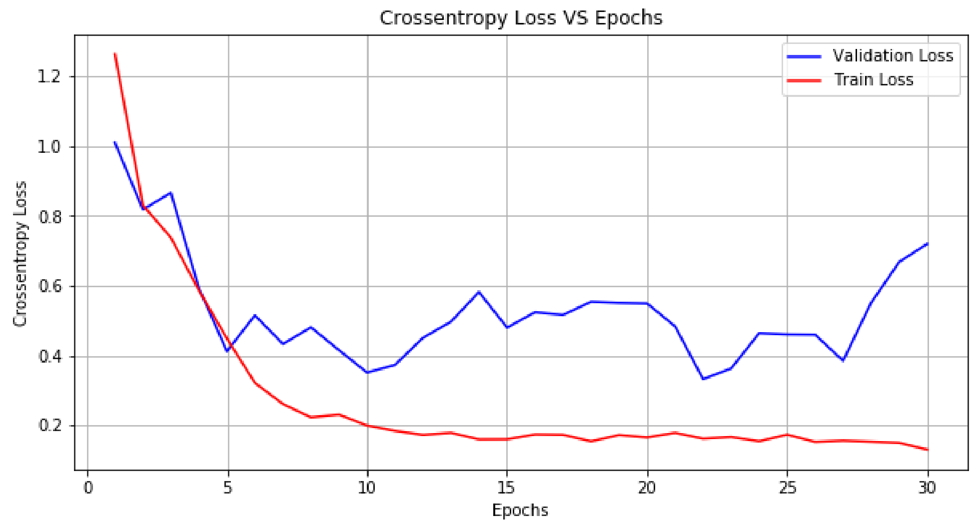
```
Test Score: 0.719278
Test Accuracy: 90.261283%
```





In [45]:

```python
# Test and train accuracy of the model
model_5_test = scores5[1]
model_5_train = max(history5.history['acc'])

# Plotting Train and Test Loss VS no. of epochs
# list of epoch numbers
x = list(range(1,31))

# Validation loss
vy = history5.history['val_loss']
# Training loss
ty = history5.history['loss']

# Calling the function to draw the plot
plt_dynamic(x, vy, ty)
```



Final Results

In [49]:

```python
from prettytable import PrettyTable

print('Performance Table')
x = PrettyTable()
x.field_names =["Models","Train","Test"]

x.add_row(["Model having 1 LSTM layer with 32 LSTM Units",model_1_train,model_1_test])
x.add_row(["Model having 1 LSTM layer with 42 LSTM Units",model_2_train,model_2_test])
x.add_row(["Model having 1 LSTM layer with 64 LSTM Units with larger Dropout",model_3_train,model_
3_test])
x.add_row(["Model having 2 LSTM layer with 32 LSTM Units",model_4_train,model_4_test])
x.add_row(["Model having 2 LSTM layer with 64 LSTM Units and with larger Dropout",model_5_train,mo
del_5_test])

print(x)
```

```
Performance Table
+-----------------------------------------------------------------------+---------------------+------
----------+
|                                 Models                                 |        Train        |
est         |
```

```
+--------------------------------------------------------------------+--------------------+-----
----------+
|             Model having 1 LSTM layer with 32 LSTM Units             | 0.9447769314472253 |
0.8982015609093994 |
|             Model having 1 LSTM layer with 42 LSTM Units             | 0.9513057671381937 |
0.9158466236851035 |
|   Model having 1 LSTM layer with 64 LSTM Units with larger Dropout   | 0.9483133841131665 | 0.914
8286392941974 |
|             Model having 2 LSTM layer with 32 LSTM Units             | 0.9530739934711643 |
0.9066847641669494 |
| Model having 2 LSTM layer with 64 LSTM Units and with larger Dropout | 0.9519858541893362 |
0.9026128266033254 |
+--------------------------------------------------------------------+--------------------+-----
----------+
```