Human Activity Recognition

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

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
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]:
```
# 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).values
        )

    # 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]:
```
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).values
```

In [7]:
```
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 [8]:
```
# Utility function to read the data from csv file
def _read_csv(filename):
    return pd.read_csv(filename, delim_whitespace=True, header=None)
```

In [39]:
```
#Function to plot Train and cross validation loss
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('categorical_crossentropy Loss')
  plt.title('\ncategorical_crossentropy Loss VS Epochs')
  plt.legend()
  plt.grid()
  plt.show()
```

In [10]:
```
# Importing tensorflow
np.random.seed(42)
import tensorflow as tf
tf.set_random_seed(42)
```

In [11]:
```
# 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)
```

```
Using TensorFlow backend.
```

In [13]:
```
# Importing libraries
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers.core import Dense, Dropout
```

In [14]:
```
# Initializing parameters
epochs = 1
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]:
```
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(7352, 128, 9)
(2947, 128, 9)
(7352, 6)
(2947, 6)
```

In [18]:
```
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
```

In [19]:
```
from hyperopt import Trials, STATUS_OK, tpe
from hyperas import optim
from hyperas.distributions import choice, uniform
```

A. Tuning parameters using Grid search for 1 Layer LSTM

In [20]:
```
import numpy
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
```

In [21]:
```
from __future__ import print_function
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
import numpy
from sklearn.model_selection import RandomizedSearchCV
from keras.layers.normalization import BatchNormalization
from keras.wrappers.scikit_learn import KerasClassifier
from keras.optimizers import SGD
from keras.constraints import maxnorm
from keras.layers import Bidirectional, CuDNNLSTM

# fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)
```

Tuning Batch size, Epochs, Dropout rate & n_hidden

In [22]:
```
def create_model(dropout_rate=0.0,n_hidden=32):
    # default values
    model = Sequential()
# Configuring the parameters
    model.add(CuDNNLSTM(n_hidden, input_shape=(timesteps, input_dim)))
#Adding batch normnalization
    model.add(BatchNormalization())
# Adding a dropout layer
    model.add(Dropout(dropout_rate))
# Adding a dense output layer with sigmoid activation
    model.add(Dense(n_classes, activation='sigmoid'))
    model.compile(loss='categorical_crossentropy',
              optimizer='RMSprop',
              metrics=['accuracy'])
    return model

# create model
model = KerasClassifier(build_fn=create_model,verbose=1)

# define the grid search parameters
batch_size = [32,64,128,256]
epochs = [10,30,50]
dropout_rate = [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
n_hidden = [16,32,64,128]

param_distributions  = dict(batch_size=batch_size, epochs=epochs,dropout_rate=dropout_rate,n_hidden=n_hidden)

grid = RandomizedSearchCV(estimator=model, param_distributions =param_distributions ,cv=3)
grid_result = grid.fit(X_train, Y_train)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
acc: 0.951 - ETA: 0s - loss: 0.1455 - acc: 0.942 - ETA: 0s - loss: 0.1433 - acc: 0.944 - ETA: 0s - loss: 0.1
404 - acc: 0.942 - ETA: 0s - loss: 0.1337 - acc: 0.944 - ETA: 0s - loss: 0.1288 - acc: 0.948 - ETA: 0s - los
s: 0.1305 - acc: 0.945 - ETA: 0s - loss: 0.1331 - acc: 0.944 - ETA: 0s - loss: 0.1322 - acc: 0.945 - ETA: 0s
- loss: 0.1317 - acc: 0.945 - ETA: 0s - loss: 0.1344 - acc: 0.944 - ETA: 0s - loss: 0.1334 - acc: 0.944 - ET
A: 0s - loss: 0.1341 - acc: 0.945 - ETA: 0s - loss: 0.1352 - acc: 0.944 - ETA: 0s - loss: 0.1359 - acc: 0.94
3 - ETA: 0s - loss: 0.1357 - acc: 0.944 - 1s 116us/step - loss: 0.1356 - acc: 0.9445
```

```
Epoch 46/50
7352/7352 [==============================] - ETA: 0s - loss: 0.0815 - acc: 0.968 - ETA: 0s - loss: 0.1394 -
acc: 0.941 - ETA: 0s - loss: 0.1528 - acc: 0.937 - ETA: 0s - loss: 0.1387 - acc: 0.943 - ETA: 0s - loss: 0.1
402 - acc: 0.941 - ETA: 0s - loss: 0.1414 - acc: 0.940 - ETA: 0s - loss: 0.1364 - acc: 0.942 - ETA: 0s - los
s: 0.1373 - acc: 0.941 - ETA: 0s - loss: 0.1383 - acc: 0.941 - ETA: 0s - loss: 0.1368 - acc: 0.941 - ETA: 0s
- loss: 0.1342 - acc: 0.943 - ETA: 0s - loss: 0.1299 - acc: 0.945 - ETA: 0s - loss: 0.1329 - acc: 0.943 - ET
A: 0s - loss: 0.1310 - acc: 0.945 - ETA: 0s - loss: 0.1276 - acc: 0.946 - ETA: 0s - loss: 0.1334 - acc: 0.94
5 - ETA: 0s - loss: 0.1347 - acc: 0.945 - 1s 117us/step - loss: 0.1344 - acc: 0.9456
Epoch 47/50
7352/7352 [==============================] - ETA: 0s - loss: 0.2012 - acc: 0.906 - ETA: 0s - loss: 0.1473 -
acc: 0.937 - ETA: 0s - loss: 0.1192 - acc: 0.952 - ETA: 0s - loss: 0.1157 - acc: 0.951 - ETA: 0s - loss: 0.1
189 - acc: 0.949 - ETA: 0s - loss: 0.1236 - acc: 0.948 - ETA: 0s - loss: 0.1192 - acc: 0.950 - ETA: 0s - los
s: 0.1184 - acc: 0.949 - ETA: 0s - loss: 0.1158 - acc: 0.950 - ETA: 0s - loss: 0.1172 - acc: 0.950 - ETA: 0s
```

In [27]:
```
n_hidden=32
epochs=50
batch_size=64
```

In [29]:
```
model = Sequential()
# Configuring the parameters
model.add(CuDNNLSTM(n_hidden, input_shape=(timesteps, input_dim)))
model.add(BatchNormalization())
# Adding a dropout layer
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes,kernel_initializer='uniform' ,activation='softmax'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
cu_dnnlstm_32 (CuDNNLSTM)    (None, 32)                5504
_________________________________________________________________
batch_normalization_33 (Batc (None, 32)                128
_________________________________________________________________
dropout_34 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_34 (Dense)             (None, 6)                 198
=================================================================
Total params: 5,830
Trainable params: 5,766
Non-trainable params: 64
_________________________________________________________________
```

In [30]:
```
model.compile(loss='categorical_crossentropy',
              optimizer='RMSprop',
              metrics=['accuracy'])
```

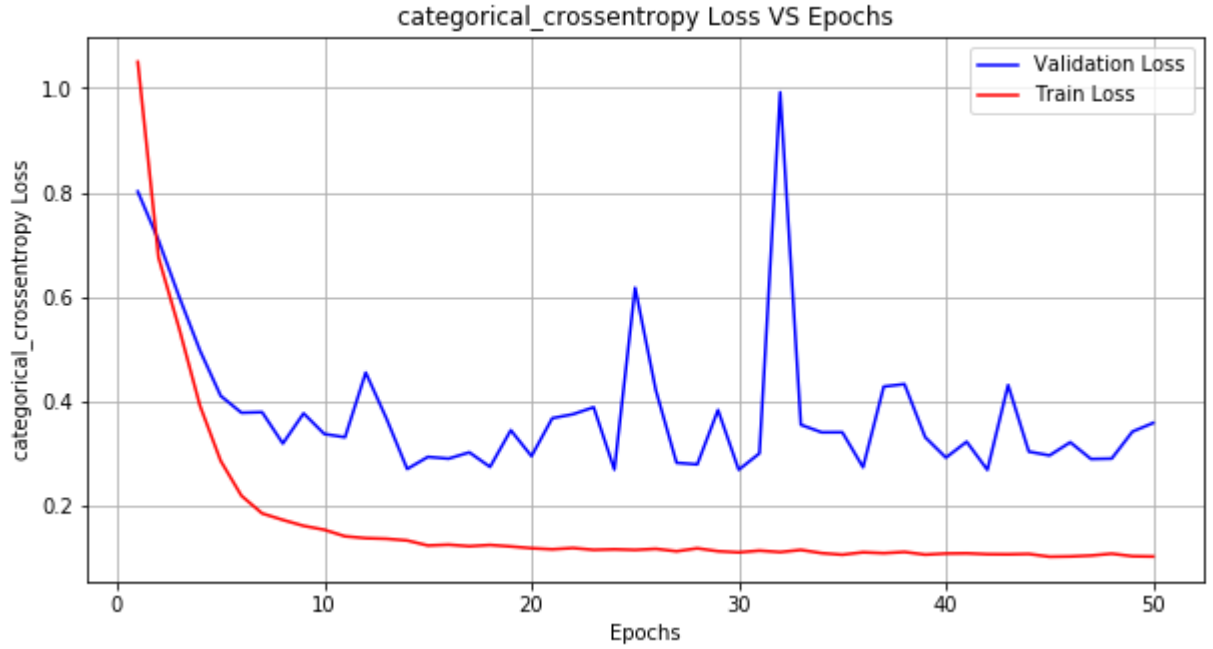
In [31]:
```
# Training the model
history=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
l_acc: 0.9091
Epoch 49/50
7352/7352 [==============================] - ETA: 0s - loss: 0.0371 - acc: 0.968 - ETA: 0s - loss: 0.1092 -
acc: 0.941 - ETA: 0s - loss: 0.1025 - acc: 0.947 - ETA: 0s - loss: 0.0985 - acc: 0.948 - ETA: 0s - loss: 0.0
970 - acc: 0.951 - ETA: 0s - loss: 0.1006 - acc: 0.952 - ETA: 0s - loss: 0.1027 - acc: 0.952 - ETA: 0s - los
s: 0.1048 - acc: 0.951 - ETA: 0s - loss: 0.1007 - acc: 0.953 - ETA: 0s - loss: 0.1027 - acc: 0.952 - ETA: 0s
- loss: 0.1012 - acc: 0.953 - ETA: 0s - loss: 0.1022 - acc: 0.952 - ETA: 0s - loss: 0.1043 - acc: 0.952 - ET
A: 0s - loss: 0.1037 - acc: 0.954 - ETA: 0s - loss: 0.1028 - acc: 0.955 - ETA: 0s - loss: 0.1027 - acc: 0.95
4 - ETA: 0s - loss: 0.1038 - acc: 0.953 - 1s 144us/step - loss: 0.1043 - acc: 0.9533 - val_loss: 0.3427 - va
l_acc: 0.9148
Epoch 50/50
7352/7352 [==============================] - ETA: 1s - loss: 0.0797 - acc: 0.953 - ETA: 0s - loss: 0.1006 -
acc: 0.955 - ETA: 0s - loss: 0.0995 - acc: 0.954 - ETA: 0s - loss: 0.0978 - acc: 0.954 - ETA: 0s - loss: 0.0
922 - acc: 0.956 - ETA: 0s - loss: 0.0904 - acc: 0.958 - ETA: 0s - loss: 0.0964 - acc: 0.955 - ETA: 0s - los
s: 0.1022 - acc: 0.953 - ETA: 0s - loss: 0.1036 - acc: 0.952 - ETA: 0s - loss: 0.1010 - acc: 0.953 - ETA: 0s
- loss: 0.0995 - acc: 0.954 - ETA: 0s - loss: 0.0988 - acc: 0.955 - ETA: 0s - loss: 0.1012 - acc: 0.953 - ET
A: 0s - loss: 0.1017 - acc: 0.953 - ETA: 0s - loss: 0.1016 - acc: 0.953 - ETA: 0s - loss: 0.1007 - acc: 0.95
3 - ETA: 0s - loss: 0.1046 - acc: 0.953 - 1s 142us/step - loss: 0.1038 - acc: 0.9542 - val_loss: 0.3592 - va
l_acc: 0.9155
```

In [40]:
```
# Plotting Train and Test Loss VS no. of epochs
# list of epoch numbers
x = list(range(1,51))

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

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



In [35]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 536        0         0        0                   0
SITTING                  0      357       133        0                   0
STANDING                 0       60       472        0                   0
WALKING                  0        0         1      472                  14
WALKING_DOWNSTAIRS       0        0         0        1                 417
WALKING_UPSTAIRS         0        0         0        9                  18

Pred                WALKING_UPSTAIRS
True
LAYING                             1
SITTING                            1
STANDING                           0
WALKING                            9
WALKING_DOWNSTAIRS                 2
WALKING_UPSTAIRS                 444
```

In [36]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - 0s 117us/
step
```

In [37]:
```
score
```

Out[37]:
```
[0.3591722217254519, 0.9155072955548015]
```

2. Trying 2 LSTM layer

In [42]:
```
batch_size=64
```

In [50]:
```
def create_model(dropout_rate=0.0,n_hidden=32):
    # default values
    model = Sequential()
# Configuring the parameters
    model.add(CuDNNLSTM(64, input_shape=(timesteps, input_dim),return_sequences=True))
#Adding batch normnalization
    model.add(BatchNormalization())
# Adding a dropout layer
    model.add(Dropout(0.1))
#adding 2nd lstm layer
    model.add(CuDNNLSTM(n_hidden))
    model.add(Dropout(dropout_rate))
# Adding a dense output layer with sigmoid activation
    model.add(Dense(n_classes, activation='sigmoid'))
    model.compile(loss='categorical_crossentropy',
              optimizer='RMSprop',
              metrics=['accuracy'])
    return model

# create model
model = KerasClassifier(build_fn=create_model,verbose=1)

# define the grid search parameters
dropout_rate = [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
n_hidden = [16,32,64,128]
epochs=[30,40,50]

param_distributions  = dict(dropout_rate=dropout_rate,n_hidden=n_hidden,epochs=epochs)

grid = RandomizedSearchCV(estimator=model, param_distributions =param_distributions ,cv=3)
grid_result = grid.fit(X_train, Y_train)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
860 - ETA: 1s - loss: 0.4392 - acc: 0.861 - ETA: 1s - loss: 0.4383 - acc: 0.862 - ETA: 0s - loss: 0.4370 - a
cc: 0.861 - ETA: 0s - loss: 0.4372 - acc: 0.861 - ETA: 0s - loss: 0.4382 - acc: 0.861 - ETA: 0s - loss: 0.43
89 - acc: 0.860 - ETA: 0s - loss: 0.4399 - acc: 0.860 - ETA: 0s - loss: 0.4394 - acc: 0.859 - ETA: 0s - los
s: 0.4385 - acc: 0.860 - ETA: 0s - loss: 0.4368 - acc: 0.860 - ETA: 0s - loss: 0.4354 - acc: 0.861 - ETA: 0s
```

```
- loss: 0.4352 - acc: 0.861 - ETA: 0s - loss: 0.4340 - acc: 0.861 - ETA: 0s - loss: 0.4333 - acc: 0.862 - ET
A: 0s - loss: 0.4307 - acc: 0.862 - ETA: 0s - loss: 0.4311 - acc: 0.862 - ETA: 0s - loss: 0.4310 - acc: 0.86
3 - ETA: 0s - loss: 0.4308 - acc: 0.863 - ETA: 0s - loss: 0.4305 - acc: 0.864 - ETA: 0s - loss: 0.4292 - ac
c: 0.865 - ETA: 0s - loss: 0.4299 - acc: 0.864 - 3s 407us/step - loss: 0.4298 - acc: 0.8648
Best: 0.931039 using {'n_hidden': 16, 'epochs': 30, 'dropout_rate': 0.7}
0.915805 (0.014062) with: {'n_hidden': 128, 'epochs': 50, 'dropout_rate': 0.5}
0.925462 (0.017827) with: {'n_hidden': 32, 'epochs': 30, 'dropout_rate': 0.6}
0.925326 (0.007990) with: {'n_hidden': 32, 'epochs': 50, 'dropout_rate': 0.5}
0.931039 (0.007495) with: {'n_hidden': 16, 'epochs': 30, 'dropout_rate': 0.7}
0.659004 (0.353180) with: {'n_hidden': 32, 'epochs': 30, 'dropout_rate': 0.7}
0.897443 (0.021852) with: {'n_hidden': 16, 'epochs': 40, 'dropout_rate': 0.6}
0.762106 (0.068636) with: {'n_hidden': 32, 'epochs': 40, 'dropout_rate': 0.9}
0.918526 (0.011429) with: {'n_hidden': 32, 'epochs': 40, 'dropout_rate': 0.0}
0.926415 (0.008997) with: {'n_hidden': 128, 'epochs': 50, 'dropout_rate': 0.0}
0.928319 (0.007833) with: {'n_hidden': 64, 'epochs': 50, 'dropout_rate': 0.6}
```

In [51]:
```
epochs=30
batch_size=64
```

In [53]:
```
model = Sequential()
# Configuring the parameters
model.add(CuDNNLSTM(64, input_shape=(timesteps, input_dim),return_sequences=True))
#Adding batch normnalization
model.add(BatchNormalization())
# Adding a dropout layer
model.add(Dropout(0.1))
#adding 2nd lstm layer
model.add(CuDNNLSTM(16))
model.add(Dropout(0.7))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
cu_dnnlstm_100 (CuDNNLSTM)   (None, 128, 64)           19200
_________________________________________________________________
batch_normalization_69 (Batc (None, 128, 64)           256
_________________________________________________________________
dropout_104 (Dropout)        (None, 128, 64)           0
_________________________________________________________________
cu_dnnlstm_101 (CuDNNLSTM)   (None, 16)                5248
_________________________________________________________________
dropout_105 (Dropout)        (None, 16)                0
_________________________________________________________________
dense_69 (Dense)             (None, 6)                 102
=================================================================
Total params: 24,806
Trainable params: 24,678
Non-trainable params: 128
_________________________________________________________________
```

In [54]:
```
model.compile(loss='categorical_crossentropy',
              optimizer='RMSprop',
              metrics=['accuracy'])
```

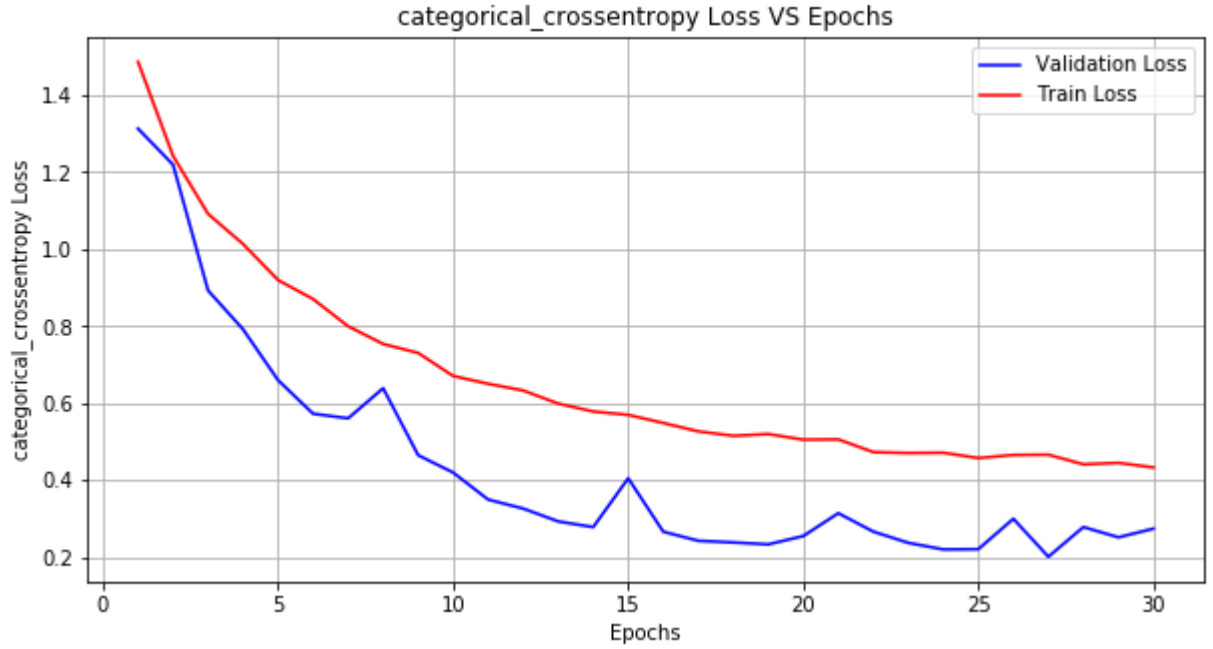
In [55]:
```
# Training the model
history=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
A: 0s - loss: 0.4592 - acc: 0.871 - ETA: 0s - loss: 0.4505 - acc: 0.869 - ETA: 0s - loss: 0.4485 - acc: 0.86
9 - ETA: 0s - loss: 0.4479 - acc: 0.869 - ETA: 0s - loss: 0.4494 - acc: 0.867 - ETA: 0s - loss: 0.4464 - ac
c: 0.868 - ETA: 0s - loss: 0.4454 - acc: 0.868 - ETA: 0s - loss: 0.4452 - acc: 0.869 - ETA: 0s - loss: 0.444
5 - acc: 0.868 - ETA: 0s - loss: 0.4456 - acc: 0.868 - ETA: 0s - loss: 0.4463 - acc: 0.868 - ETA: 0s - loss:
0.4492 - acc: 0.867 - ETA: 0s - loss: 0.4496 - acc: 0.866 - ETA: 0s - loss: 0.4477 - acc: 0.865 - ETA: 0s -
loss: 0.4470 - acc: 0.866 - ETA: 0s - loss: 0.4460 - acc: 0.866 - 2s 260us/step - loss: 0.4452 - acc: 0.8662
- val_loss: 0.2518 - val_acc: 0.9135
Epoch 30/30
7352/7352 [==============================] - ETA: 1s - loss: 0.3742 - acc: 0.890 - ETA: 1s - loss: 0.3952 -
acc: 0.893 - ETA: 1s - loss: 0.4021 - acc: 0.888 - ETA: 1s - loss: 0.4153 - acc: 0.875 - ETA: 1s - loss: 0.4
206 - acc: 0.876 - ETA: 1s - loss: 0.4218 - acc: 0.874 - ETA: 1s - loss: 0.4357 - acc: 0.870 - ETA: 1s - los
s: 0.4308 - acc: 0.871 - ETA: 1s - loss: 0.4296 - acc: 0.872 - ETA: 1s - loss: 0.4374 - acc: 0.867 - ETA: 1s
- loss: 0.4341 - acc: 0.869 - ETA: 0s - loss: 0.4301 - acc: 0.869 - ETA: 0s - loss: 0.4281 - acc: 0.867 - ET
A: 0s - loss: 0.4242 - acc: 0.871 - ETA: 0s - loss: 0.4244 - acc: 0.871 - ETA: 0s - loss: 0.4261 - acc: 0.86
9 - ETA: 0s - loss: 0.4251 - acc: 0.870 - ETA: 0s - loss: 0.4252 - acc: 0.870 - ETA: 0s - loss: 0.4246 - ac
c: 0.870 - ETA: 0s - loss: 0.4256 - acc: 0.869 - ETA: 0s - loss: 0.4254 - acc: 0.869 - ETA: 0s - loss: 0.424
2 - acc: 0.871 - ETA: 0s - loss: 0.4250 - acc: 0.870 - ETA: 0s - loss: 0.4269 - acc: 0.869 - ETA: 0s - loss:
0.4263 - acc: 0.869 - ETA: 0s - loss: 0.4264 - acc: 0.868 - ETA: 0s - loss: 0.4276 - acc: 0.867 - ETA: 0s -
loss: 0.4311 - acc: 0.867 - ETA: 0s - loss: 0.4333 - acc: 0.868 - 2s 260us/step - loss: 0.4333 - acc: 0.8689
- val_loss: 0.2743 - val_acc: 0.9101
```

In [56]:
```
# Plotting Train and Test Loss VS no. of epochs
# list of epoch numbers
x = list(range(1,31))

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

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



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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 537        0         0        0                   0
SITTING                  6      327       140        0                   0
STANDING                 0       45       487        0                   0
WALKING                  0        0         0      471                  24
WALKING_DOWNSTAIRS       0        0         0        2                 418
WALKING_UPSTAIRS         0        4         0       18                   7

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                           18
STANDING                           0
WALKING                            1
WALKING_DOWNSTAIRS                 0
WALKING_UPSTAIRS                 442
```

In [58]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - E
TA:  - ETA:  - ETA:  - ETA:  - 1s 221us/step
```

In [59]:
```
score
```

Out[59]:
```
[0.2743394161926277, 0.9100780454699695]
```

In [ ]:
```

```

3. Trying 3x3 LSTM layer

In [61]:
```
def create_model(dropout_rate=0.0,n_hidden=32):
    # default values
    model = Sequential()
# Configuring the parameters
    model.add(CuDNNLSTM(64, input_shape=(timesteps, input_dim),return_sequences=True))
#Adding batch normnalization
    model.add(BatchNormalization())
# Adding a dropout layer
    model.add(Dropout(0.1))
#adding 2nd lstm layer
    model.add(CuDNNLSTM(16,return_sequences=True))
    model.add(Dropout(0.7))

#adding 3rd lstm layer
    model.add(CuDNNLSTM(n_hidden))
    model.add(Dropout(dropout_rate))
# Adding a dense output layer with sigmoid activation
    model.add(Dense(n_classes, activation='sigmoid'))
    model.compile(loss='categorical_crossentropy',
              optimizer='RMSprop',
              metrics=['accuracy'])
    return model

# create model
model = KerasClassifier(build_fn=create_model,verbose=1)

# define the grid search parameters
dropout_rate = [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
n_hidden = [16,32,64,128]
epochs=[30,40,50]
batch_size = [32,64,128,256]

param_distributions  = dict(dropout_rate=dropout_rate,n_hidden=n_hidden,epochs=epochs,batch_size=batch_size)

grid = RandomizedSearchCV(estimator=model, param_distributions =param_distributions ,cv=3)
grid_result = grid.fit(X_train, Y_train)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
```

```
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
Epoch 1/50
4901/4901 [==============================] - ETA: 6:37 - loss: 1.8041 - acc: 0.109 - ETA: 1:36 - loss: 1.731
4 - acc: 0.339 - ETA: 53s - loss: 1.6692 - acc: 0.446 - ETA: 36s - loss: 1.6049 - acc: 0.49 - ETA: 26s - los
s: 1.5360 - acc: 0.51 - ETA: 21s - loss: 1.4774 - acc: 0.54 - ETA: 17s - loss: 1.4206 - acc: 0.57 - ETA: 14s
- loss: 1.3925 - acc: 0.58 - ETA: 11s - loss: 1.3697 - acc: 0.58 - ETA: 10s - loss: 1.3340 - acc: 0.58 - ET
A: 8s - loss: 1.3024 - acc: 0.5953 - ETA: 7s - loss: 1.2757 - acc: 0.597 - ETA: 6s - loss: 1.2483 - acc: 0.6
04 - ETA: 5s - loss: 1.2337 - acc: 0.602 - ETA: 4s - loss: 1.2109 - acc: 0.602 - ETA: 4s - loss: 1.1884 - ac
c: 0.607 - ETA: 3s - loss: 1.1690 - acc: 0.611 - ETA: 2s - loss: 1.1517 - acc: 0.611 - ETA: 2s - loss: 1.135
2 - acc: 0.611 - ETA: 2s - loss: 1.1179 - acc: 0.613 - ETA: 1s - loss: 1.1021 - acc: 0.616 - ETA: 1s - loss:
1.0922 - acc: 0.616 - ETA: 0s - loss: 1.0804 - acc: 0.617 - ETA: 0s - loss: 1.0685 - acc: 0.616 - ETA: 0s -
loss: 1.0575 - acc: 0.616 - ETA: 0s - loss: 1.0483 - acc: 0.615 - 7s 1ms/step - loss: 1.0458 - acc: 0.6152
Epoch 2/50
4901/4901 [==============================] - ETA: 1s - loss: 0.7234 - acc: 0.687 - ETA: 1s - loss: 0.7231 -
acc: 0.722 - ETA: 1s - loss: 0.7396 - acc: 0.680 - ETA: 1s - loss: 0.7576 - acc: 0.660 - ETA: 1s - loss: 0.7
687 - acc: 0.651 - ETA: 1s - loss: 0.7645 - acc: 0.647 - ETA: 1s - loss: 0.7601 - acc: 0.642 - ETA: 1s - los
s: 0.7510 - acc: 0.647 - ETA: 0s - loss: 0.7505 - acc: 0.646 - ETA: 0s - loss: 0.7483 - acc: 0.642 - ETA: 0s
- loss: 0.7514 - acc: 0.649 - ETA: 0s - loss: 0.7470 - acc: 0.651 - ETA: 0s - loss: 0.7406 - acc: 0.652 - ET
A: 0s - loss: 0.7406 - acc: 0.646 - ETA: 0s - loss: 0.7371 - acc: 0.646 - ETA: 0s - loss: 0.7378 - acc: 0.64
8 - ETA: 0s - loss: 0.7457 - acc: 0.647 - ETA: 0s - loss: 0.7400 - acc: 0.649 - ETA: 0s - loss: 0.7385 - ac
```

In [67]:
```
epochs=40
batch_size=32
```

In [68]:
```
model = Sequential()
# Configuring the parameters
model.add(CuDNNLSTM(64, input_shape=(timesteps, input_dim),return_sequences=True))
#Adding batch normnalization
model.add(BatchNormalization())
# Adding a dropout layer
model.add(Dropout(0.1))
#adding 2nd lstm layer
model.add(CuDNNLSTM(16,return_sequences=True))
model.add(Dropout(0.7))

#adding 3rd lstm layer
model.add(CuDNNLSTM(128))
model.add(Dropout(0.3))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
cu_dnnlstm_200 (CuDNNLSTM)   (None, 128, 64)           19200
_________________________________________________________________
batch_normalization_103 (Bat (None, 128, 64)           256
_________________________________________________________________
dropout_203 (Dropout)        (None, 128, 64)           0
_________________________________________________________________
cu_dnnlstm_201 (CuDNNLSTM)   (None, 128, 16)           5248
_________________________________________________________________
dropout_204 (Dropout)        (None, 128, 16)           0
_________________________________________________________________
cu_dnnlstm_202 (CuDNNLSTM)   (None, 128)               74752
_________________________________________________________________
dropout_205 (Dropout)        (None, 128)               0
_________________________________________________________________
dense_102 (Dense)            (None, 6)                 774
=================================================================
Total params: 100,230
Trainable params: 100,102
Non-trainable params: 128
_________________________________________________________________
```

In [69]:
```
model.compile(loss='categorical_crossentropy',
              optimizer='RMSprop',
              metrics=['accuracy'])
```

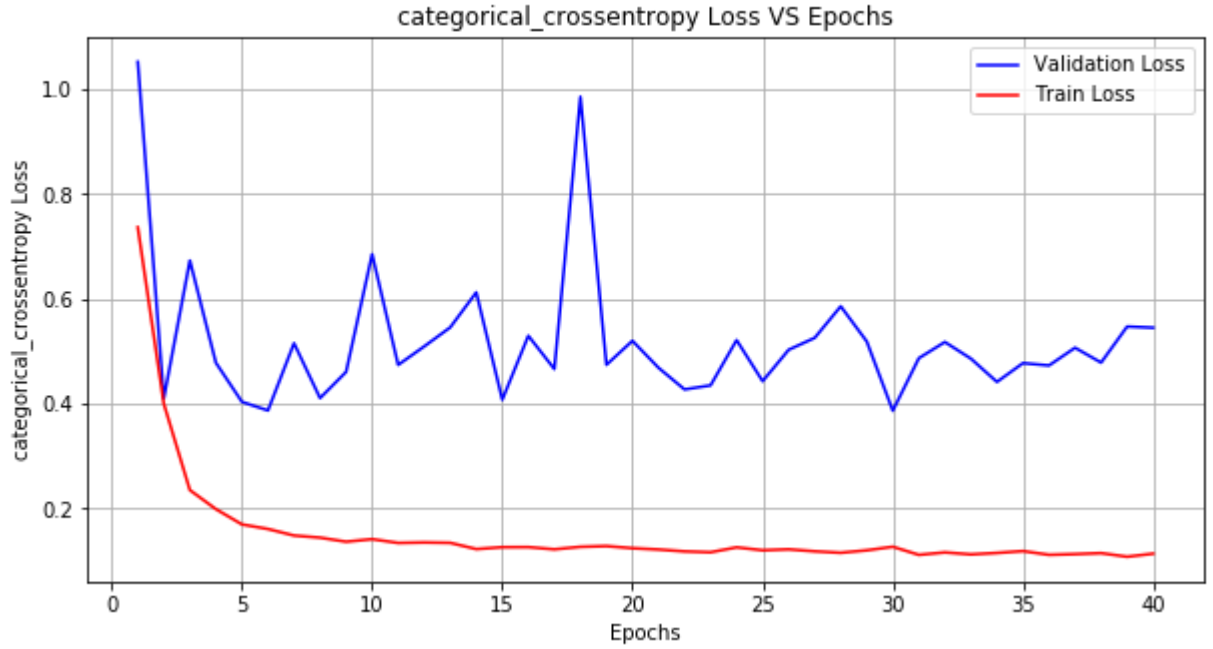
In [70]:
```
# Training the model
history=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
loss: 0.1103 - acc: 0.957 - ETA: 3s - loss: 0.1112 - acc: 0.958 - ETA: 2s - loss: 0.1130 - acc: 0.957 - ETA:
2s - loss: 0.1139 - acc: 0.956 - ETA: 2s - loss: 0.1202 - acc: 0.955 - ETA: 2s - loss: 0.1194 - acc: 0.955 -
ETA: 2s - loss: 0.1181 - acc: 0.955 - ETA: 2s - loss: 0.1194 - acc: 0.954 - ETA: 2s - loss: 0.1185 - acc: 0.
954 - ETA: 2s - loss: 0.1178 - acc: 0.954 - ETA: 2s - loss: 0.1162 - acc: 0.954 - ETA: 2s - loss: 0.1167 - a
cc: 0.954 - ETA: 2s - loss: 0.1148 - acc: 0.955 - ETA: 2s - loss: 0.1126 - acc: 0.956 - ETA: 2s - loss: 0.11
31 - acc: 0.955 - ETA: 2s - loss: 0.1128 - acc: 0.956 - ETA: 2s - loss: 0.1128 - acc: 0.956 - ETA: 2s - los
s: 0.1136 - acc: 0.955 - ETA: 1s - loss: 0.1124 - acc: 0.956 - ETA: 1s - loss: 0.1128 - acc: 0.955 - ETA: 1s
- loss: 0.1115 - acc: 0.956 - ETA: 1s - loss: 0.1114 - acc: 0.956 - ETA: 1s - loss: 0.1106 - acc: 0.956 - ET
A: 1s - loss: 0.1112 - acc: 0.955 - ETA: 1s - loss: 0.1113 - acc: 0.955 - ETA: 1s - loss: 0.1116 - acc: 0.95
4 - ETA: 1s - loss: 0.1109 - acc: 0.955 - ETA: 1s - loss: 0.1116 - acc: 0.954 - ETA: 1s - loss: 0.1124 - ac
c: 0.954 - ETA: 1s - loss: 0.1123 - acc: 0.954 - ETA: 1s - loss: 0.1125 - acc: 0.953 - ETA: 1s - loss: 0.112
8 - acc: 0.953 - ETA: 1s - loss: 0.1122 - acc: 0.953 - ETA: 1s - loss: 0.1123 - acc: 0.953 - ETA: 0s - loss:
0.1116 - acc: 0.953 - ETA: 0s - loss: 0.1128 - acc: 0.953 - ETA: 0s - loss: 0.1120 - acc: 0.954 - ETA: 0s -
loss: 0.1116 - acc: 0.954 - ETA: 0s - loss: 0.1113 - acc: 0.954 - ETA: 0s - loss: 0.1105 - acc: 0.954 - ETA:
0s - loss: 0.1098 - acc: 0.954 - ETA: 0s - loss: 0.1089 - acc: 0.955 - ETA: 0s - loss: 0.1090 - acc: 0.955 -
ETA: 0s - loss: 0.1091 - acc: 0.955 - ETA: 0s - loss: 0.1083 - acc: 0.955 - ETA: 0s - loss: 0.1105 - acc: 0.
955 - ETA: 0s - loss: 0.1134 - acc: 0.955 - ETA: 0s - loss: 0.1131 - acc: 0.955 - ETA: 0s - loss: 0.1135 - a
cc: 0.954 - ETA: 0s - loss: 0.1134 - acc: 0.954 - 6s 768us/step - loss: 0.1137 - acc: 0.9544 - val_loss: 0.5
453 - val_acc: 0.9131
```

In [71]:
```
# Plotting Train and Test Loss VS no. of epochs
# list of epoch numbers
x = list(range(1,41))

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

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



In [72]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 537        0         0        0                   0
SITTING                  0      428        39        1                   0
STANDING                 0      107       425        0                   0
WALKING                  0        0         0      464                  23
WALKING_DOWNSTAIRS       0        0         0        0                 420
WALKING_UPSTAIRS         0        1         0        1                  52

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                           23
STANDING                           0
WALKING                            9
WALKING_DOWNSTAIRS                 0
WALKING_UPSTAIRS                 417
```

In [73]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - E
TA:  - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - 1s 351us/step
```

In [74]:
```
score
```

Out[74]:
```
[0.5453052728528218, 0.9131319986426875]
```

Summary

In [2]:
```
from prettytable import PrettyTable
x=PrettyTable()
x.field_names = ["Lstm layer","Train_acc","Test_acc"]
x.add_row(['1 layer','0.9542','0.9155'])
x.add_row(['2 layer','0.8689','0.9101'])
x.add_row(['3 layer','0.9544','0.9131'])

print(x)
```

```
+------------+-----------+----------+
| Lstm layer | Train_acc | Test_acc |
+------------+-----------+----------+
|  1 layer   |   0.9542  |  0.9155  |
|  2 layer   |   0.8689  |  0.9101  |
|  3 layer   |   0.9544  |  0.9131  |
+------------+-----------+----------+
```