HAR_LSTM (2)

March 28, 2020

In [0]:
```
# Importing Libraries
```

In [1]:
```
%tensorflow_version 1.x
```

```
TensorFlow 1.x selected.
```

In [0]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

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

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

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

0.0.1 Data

In [0]:
```
# Data directory
DATADIR = '/content/drive/My Drive/Colab Notebooks/HumanActivityRecognition/HAR/UCI_HA
```

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

In [0]:
```
# 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'/content/drive/My Drive/Colab Notebooks/HumanActivityRecognition/
        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 [0]:
```
def load_y(subset):
    """
    The objective that we are trying to predict is a integer, from 1 to 6,
    that represents a human activity.
    """
    filename = f'/content/drive/My Drive/Colab Notebooks/HumanActivityRecognition/HAR/
    y = _read_csv(filename)[0]

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

In [0]:
```
def load_data():
    """
```

```
    Obtain the dataset from multiple files.
    Returns: X_train, X_test, y_train, y_test
    """
    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 [0]: # Importing tensorflow
        np.random.seed(42)
        import tensorflow as tf
        tf.set_random_seed(42)
```

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

```
In [13]: # 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 [0]: # Importing libraries
        from keras.models import Sequential
        from keras.layers import LSTM
        from keras.layers.core import Dense, Dropout
```

```
In [0]: # Initializing parameters

        epochs = 30
        batch_size = 16
        n_hidden = 32
```

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

```
In [17]: # Loading the train and test data
         X_train, X_test, Y_train, Y_test = load_data()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:11: FutureWarning: Method .as_mati
  # This is added back by InteractiveShellApp.init_path()
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:9: FutureWarning: Method .as_matri
  if __name__ == '__main__':
```

```
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))
         print(n_classes)
```

```
128
9
7352
6
```

```
In [0]: def plt_dynamic(x, vy, ty, ax, colors=['b']):
            ax.plot(x, vy, 'b', label="Validation Loss")
            ax.plot(x, ty, 'r', label="Train Loss")
            plt.legend()
            plt.grid()
            fig.canvas.draw()
```

- Defining the Architecture of LSTM

```
In [0]: # Initiliazing the sequential model
        model = Sequential()
        # Configuring the parameters
        model.add(LSTM(n_hidden, 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()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 32)                5376
_________________________________________________________________
```

```
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 198
=================================================================
Total params: 5,574
Trainable params: 5,574
Non-trainable params: 0
_________________________________________________________________
```

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

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The nar

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend
```

```
In [0]: # Training the model
        model.fit(X_train,
                  Y_train,
                  batch_size=batch_size,
                  validation_data=(X_test, Y_test),
                  epochs=epochs)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/ops/math_
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend

Train on 7352 samples, validate on 2947 samples
Epoch 1/30
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend

7352/7352 [==============================] - 35s 5ms/step - loss: 1.3084 - acc: 0.4425 - val_lo
Epoch 2/30
7352/7352 [==============================] - 34s 5ms/step - loss: 0.9628 - acc: 0.5876 - val_lo
Epoch 3/30
```

```
7352/7352 [==============================] - 35s 5ms/step - loss: 0.7721 - acc: 0.6538 - val_lo
Epoch 4/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.6918 - acc: 0.6669 - val_lo
Epoch 5/30
7352/7352 [==============================] - 34s 5ms/step - loss: 0.6357 - acc: 0.6938 - val_lo
Epoch 6/30
7352/7352 [==============================] - 34s 5ms/step - loss: 0.5906 - acc: 0.7067 - val_lo
Epoch 7/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.5678 - acc: 0.7552 - val_lo
Epoch 8/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.5142 - acc: 0.7825 - val_lo
Epoch 9/30
7352/7352 [==============================] - 34s 5ms/step - loss: 0.4729 - acc: 0.7856 - val_lo
Epoch 10/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.4340 - acc: 0.8028 - val_lo
Epoch 11/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.4110 - acc: 0.8146 - val_lo
Epoch 12/30
7352/7352 [==============================] - 36s 5ms/step - loss: 0.3979 - acc: 0.8467 - val_lo
Epoch 13/30
7352/7352 [==============================] - 36s 5ms/step - loss: 0.3420 - acc: 0.9011 - val_lo
Epoch 14/30
7352/7352 [==============================] - 37s 5ms/step - loss: 0.2987 - acc: 0.9151 - val_lo
Epoch 15/30
7352/7352 [==============================] - 36s 5ms/step - loss: 0.2498 - acc: 0.9268 - val_lo
Epoch 16/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.3114 - acc: 0.9170 - val_lo
Epoch 17/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.2965 - acc: 0.9135 - val_lo
Epoch 18/30
7352/7352 [==============================] - 36s 5ms/step - loss: 0.2385 - acc: 0.9320 - val_lo
Epoch 19/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.2526 - acc: 0.9241 - val_lo
Epoch 20/30
7352/7352 [==============================] - 34s 5ms/step - loss: 0.2222 - acc: 0.9334 - val_lo
Epoch 21/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.2048 - acc: 0.9387 - val_lo
Epoch 22/30
7352/7352 [==============================] - 36s 5ms/step - loss: 0.2315 - acc: 0.9321 - val_lo
Epoch 23/30
7352/7352 [==============================] - 34s 5ms/step - loss: 0.2376 - acc: 0.9242 - val_lo
Epoch 24/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.1921 - acc: 0.9372 - val_lo
Epoch 25/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.1867 - acc: 0.9419 - val_lo
Epoch 26/30
7352/7352 [==============================] - 34s 5ms/step - loss: 0.1973 - acc: 0.9422 - val_lo
Epoch 27/30
```

```
7352/7352 [==============================] - 34s 5ms/step - loss: 0.1676 - acc: 0.9470 - val_lo
Epoch 28/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.1757 - acc: 0.9391 - val_lo
Epoch 29/30
7352/7352 [==============================] - 34s 5ms/step - loss: 0.1689 - acc: 0.9434 - val_lo
Epoch 30/30
7352/7352 [==============================] - 34s 5ms/step - loss: 0.1523 - acc: 0.9471 - val_lo
```

```
Out[0]: <keras.callbacks.History at 0x7fe676d0f438>
```

```
In [0]: # Confusion Matrix
        print(confusion_matrix(Y_test, model.predict(X_test)))
```

Pred	LAYING	SITTING	...	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
True			...		
LAYING	510	0	...	0	0
SITTING	3	384	...	0	0
STANDING	0	86	...	0	0
WALKING	2	0	...	25	6
WALKING_DOWNSTAIRS	0	0	...	409	10
WALKING_UPSTAIRS	2	1	...	0	461

[6 rows x 6 columns]

```
In [0]: score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 1s 366us/step
```

```
In [0]: score
```

```
Out[0]: [0.3678714334553279, 0.9056667797760435]
```

Hyperparameter Tuning

https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/

```
In [0]: import keras
        from keras.models import Sequential
        from keras.layers import Dense,Dropout
        import keras
        from keras.wrappers.scikit_learn import KerasClassifier
        from sklearn.model_selection import GridSearchCV
```

```
In [0]: def create_model(dropout_rate=0.0,n_hidden=1):
            model = Sequential()
            model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
```

```
            model.add(Dropout(dropout_rate))
            model.add(Dense(n_classes, activation='softmax'))
            model.compile(loss='binary_crossentropy', optimizer='rmsprop', metrics=['accuracy']
            return model
```

In [0]:
```
# define the grid search parameters
model = KerasClassifier(build_fn=create_model)
n_hidden = [20, 25, 30, 35, 40, 45, 50]
dropout_rate = [0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
param_grid = dict(dropout_rate=dropout_rate, n_hidden=n_hidden)
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, 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']
```

```
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:706: UserWarni
  "timeout or by a memory leak.", UserWarning


WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The nar

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/ops/nn_im
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend

Epoch 1/1
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend

7352/7352 [==============================] - 16s 2ms/step - loss: 0.3306 - acc: 0.8700
Best: 0.881619 using {'dropout_rate': 0.5, 'n_hidden': 40}
```

In [0]:
```
#https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchC
a=grid_result.best_params_['dropout_rate']
p = grid_result.best_params_['n_hidden']
```

```
        print(grid_result.best_score_)
        print(a)
        print(p)
```

```
0.8816190913489047
0.5
40
```

```
In [0]: model = Sequential()
        model.add(LSTM(units=p, input_shape=(timesteps, input_dim)))
        model.add(Dropout(rate=a))
        model.add(Dense(n_classes, activation='softmax'))
        model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy
```

```
In [0]: # Training the model
        history=model.fit(X_train,
                  Y_train,
                  batch_size=batch_size,
                  validation_data=(X_test, Y_test),
                  epochs=30)
        score = model.evaluate(X_test, Y_test, verbose=0)
        print('Test loss:', score[0])
        print('Test accuracy:', score[1])
```

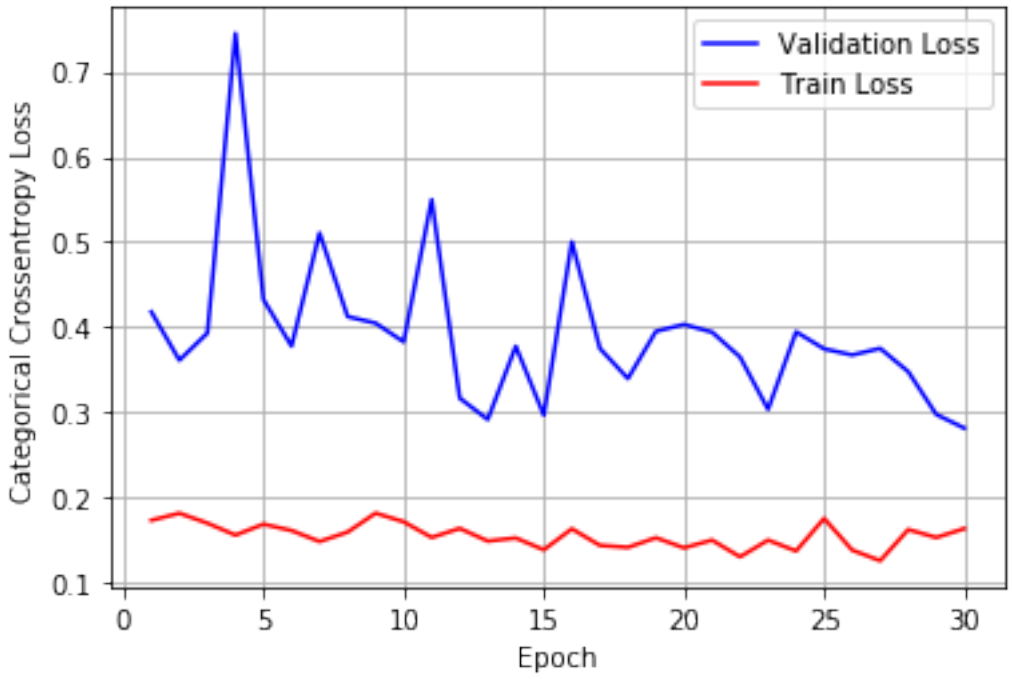
```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1733 - acc: 0.9412 - val_lo
Epoch 2/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1816 - acc: 0.9395 - val_lo
Epoch 3/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.1699 - acc: 0.9471 - val_lo
Epoch 4/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.1559 - acc: 0.9455 - val_lo
Epoch 5/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.1687 - acc: 0.9418 - val_lo
Epoch 6/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.1613 - acc: 0.9453 - val_lo
Epoch 7/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.1483 - acc: 0.9474 - val_lo
Epoch 8/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1594 - acc: 0.9448 - val_lo
Epoch 9/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.1817 - acc: 0.9433 - val_lo
Epoch 10/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1716 - acc: 0.9470 - val_lo
Epoch 11/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1531 - acc: 0.9497 - val_lo
Epoch 12/30
```

```
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1638 - acc: 0.9433 - val_lo
Epoch 13/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1489 - acc: 0.9472 - val_lo
Epoch 14/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.1525 - acc: 0.9463 - val_lo
Epoch 15/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1386 - acc: 0.9501 - val_lo
Epoch 16/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1633 - acc: 0.9487 - val_lo
Epoch 17/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1438 - acc: 0.9512 - val_lo
Epoch 18/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1415 - acc: 0.9493 - val_lo
Epoch 19/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1528 - acc: 0.9449 - val_lo
Epoch 20/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1409 - acc: 0.9499 - val_lo
Epoch 21/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1500 - acc: 0.9457 - val_lo
Epoch 22/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1304 - acc: 0.9520 - val_lo
Epoch 23/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.1500 - acc: 0.9467 - val_lo
Epoch 24/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1372 - acc: 0.9538 - val_lo
Epoch 25/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1755 - acc: 0.9472 - val_lo
Epoch 26/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.1383 - acc: 0.9523 - val_lo
Epoch 27/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1258 - acc: 0.9508 - val_lo
Epoch 28/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1624 - acc: 0.9494 - val_lo
Epoch 29/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.1531 - acc: 0.9510 - val_lo
Epoch 30/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.1635 - acc: 0.9471 - val_lo
Test loss: 0.281517137235211
Test accuracy: 0.9280624363759755
```

```
In [0]: import matplotlib.pyplot as plt
        score = model.evaluate(X_test, Y_test, verbose=0)
        print('Test score:', score[0])
        print('Test accuracy:', score[1])

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

```
        x = list(range(1,epochs+1)) #List of epoch numbers
        val_loss = history.history['val_loss'] #Validation Loss
        loss = history.history['loss'] #Training Loss
        plt_dynamic(x, val_loss, loss, ax) #Display the model
```

```
Test score: 0.281517137235211
Test accuracy: 0.9280624363759755
```




```
In [0]: # Confusion Matrix
        print(confusion_matrix(Y_test, model.predict(X_test)))
```

Pred	LAYING	SITTING	...	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
True			...		
LAYING	511	0	...	0	0
SITTING	0	367	...	2	1
STANDING	0	45	...	0	1
WALKING	0	0	...	13	7
WALKING_DOWNSTAIRS	0	0	...	412	5
WALKING_UPSTAIRS	2	3	...	9	420

[6 rows x 6 columns]

From Hyperparameter Tuning, we got our desired result as accuracy is 93%
2 LSTM Layers having 32 LSTM units + Dropout (0.8)

```
In [0]: # Initializing parameters
        epochs = 30
        batch_size = 32
        n_hidden = n_hidden
```

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

        #https://adventuresinmachinelearning.com/keras-lstm-tutorial/
        #Multilayer LSTM

        model.add(LSTM(32, return_sequences=True,input_shape=(timesteps, input_dim)))
        model.add(Dropout(0.8))
        model.add(LSTM(32,input_shape=(timesteps, 32)))
        model.add(BatchNormalization())
        model.add(Dropout(0.8))
        model.add(Dense(n_classes, activation='sigmoid'))
        model.summary()
```

```
WARNING:tensorflow:Large dropout rate: 0.8 (>0.5). In TensorFlow 2.x, dropout() uses dropout ra
WARNING:tensorflow:Large dropout rate: 0.8 (>0.5). In TensorFlow 2.x, dropout() uses dropout ra
Model: "sequential_13"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_24 (LSTM)               (None, 128, 32)           5376
_________________________________________________________________
dropout_15 (Dropout)         (None, 128, 32)           0
_________________________________________________________________
lstm_25 (LSTM)               (None, 32)                8320
_________________________________________________________________
batch_normalization_4 (Batch (None, 32)                128
_________________________________________________________________
dropout_16 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_12 (Dense)             (None, 6)                 198
=================================================================
Total params: 14,022
Trainable params: 13,958
Non-trainable params: 64
_________________________________________________________________
```

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

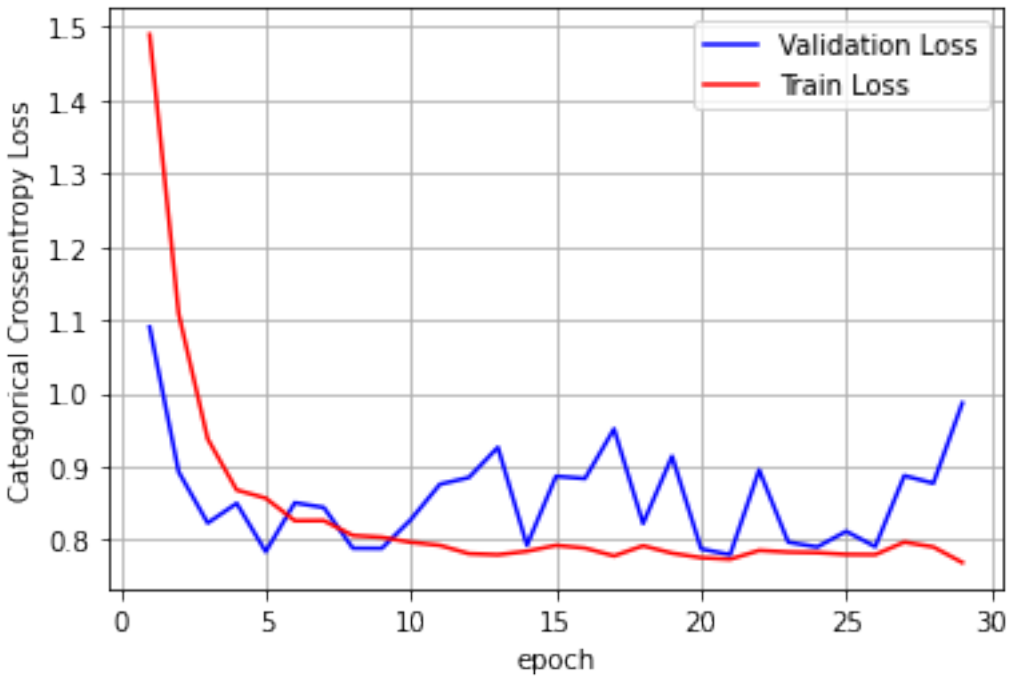
```
In [0]: # Training the model
        history=model.fit(X_train,
```

```
                    Y_train,
                    batch_size=batch_size,
                    validation_data=(X_test, Y_test),
                    epochs=epochs)
        score = model.evaluate(X_test, Y_test, verbose=0)
        print('Test loss:', score[0])
        print('Test accuracy:', score[1])

Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 37s 5ms/step - loss: 1.4897 - acc: 0.4070 - val_lo
Epoch 2/30
7352/7352 [==============================] - 32s 4ms/step - loss: 1.1085 - acc: 0.5170 - val_lo
Epoch 3/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.9380 - acc: 0.5239 - val_lo
Epoch 4/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.8682 - acc: 0.5229 - val_lo
Epoch 5/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.8571 - acc: 0.5231 - val_lo
Epoch 6/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.8264 - acc: 0.5209 - val_lo
Epoch 7/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.8265 - acc: 0.5299 - val_lo
Epoch 8/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.8062 - acc: 0.5331 - val_lo
Epoch 9/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.8038 - acc: 0.5329 - val_lo
Epoch 10/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7969 - acc: 0.5222 - val_lo
Epoch 11/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7927 - acc: 0.5310 - val_lo
Epoch 12/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7814 - acc: 0.5331 - val_lo
Epoch 13/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7799 - acc: 0.5355 - val_lo
Epoch 14/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7847 - acc: 0.5254 - val_lo
Epoch 15/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7927 - acc: 0.5393 - val_lo
Epoch 16/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7890 - acc: 0.5331 - val_lo
Epoch 17/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7783 - acc: 0.5413 - val_lo
Epoch 18/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7922 - acc: 0.5434 - val_lo
Epoch 19/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7821 - acc: 0.5484 - val_lo
Epoch 20/30
```

```
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7759 - acc: 0.5411 - val_lo
Epoch 21/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7738 - acc: 0.5388 - val_lo
Epoch 22/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7857 - acc: 0.5435 - val_lo
Epoch 23/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7835 - acc: 0.5369 - val_lo
Epoch 24/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7829 - acc: 0.5457 - val_lo
Epoch 25/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7803 - acc: 0.5378 - val_lo
Epoch 26/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7801 - acc: 0.5419 - val_lo
Epoch 27/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7971 - acc: 0.5450 - val_lo
Epoch 28/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7906 - acc: 0.5377 - val_lo
Epoch 29/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7693 - acc: 0.5416 - val_lo
Epoch 30/30
7352/7352 [==============================] - 32s 4ms/step - loss: nan - acc: 0.5253 - val_loss
Test loss: nan
Test accuracy: 0.168306752629793
```

```
In [0]: import matplotlib.pyplot as plt
        score = model.evaluate(X_test, Y_test, verbose=0)
        print('Test score:', score[0])
        print('Test accuracy:', score[1])

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

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

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

```
Test score: nan
Test accuracy: 0.168306752629793
```




```
In [0]: # Confusion Matrix
        print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                WALKING
True
LAYING                  537
SITTING                 491
STANDING                532
WALKING                 496
WALKING_DOWNSTAIRS      420
WALKING_UPSTAIRS        471
```

2 LSTM Layers having 32 LSTM units + Dropout (0.7)

```
In [0]: # Initializing parameters
        epochs = 30
        batch_size = 32
        n_hidden = n_hidden
```

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

        #https://adventuresinmachinelearning.com/keras-lstm-tutorial/
        #Multilayer LSTM
```

```
        model.add(LSTM(32, return_sequences=True,input_shape=(timesteps, input_dim)))
        model.add(Dropout(0.7))
        model.add(LSTM(32,input_shape=(timesteps, 32)))
        model.add(BatchNormalization())
        model.add(Dropout(0.7))
        model.add(Dense(n_classes, activation='sigmoid'))
        model.summary()
```

```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_26 (LSTM)               (None, 128, 32)           5376
_________________________________________________________________
dropout_17 (Dropout)         (None, 128, 32)           0
_________________________________________________________________
lstm_27 (LSTM)               (None, 32)                8320
_________________________________________________________________
batch_normalization_5 (Batch (None, 32)                128
_________________________________________________________________
dropout_18 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_13 (Dense)             (None, 6)                 198
=================================================================
Total params: 14,022
Trainable params: 13,958
Non-trainable params: 64
_________________________________________________________________
```

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

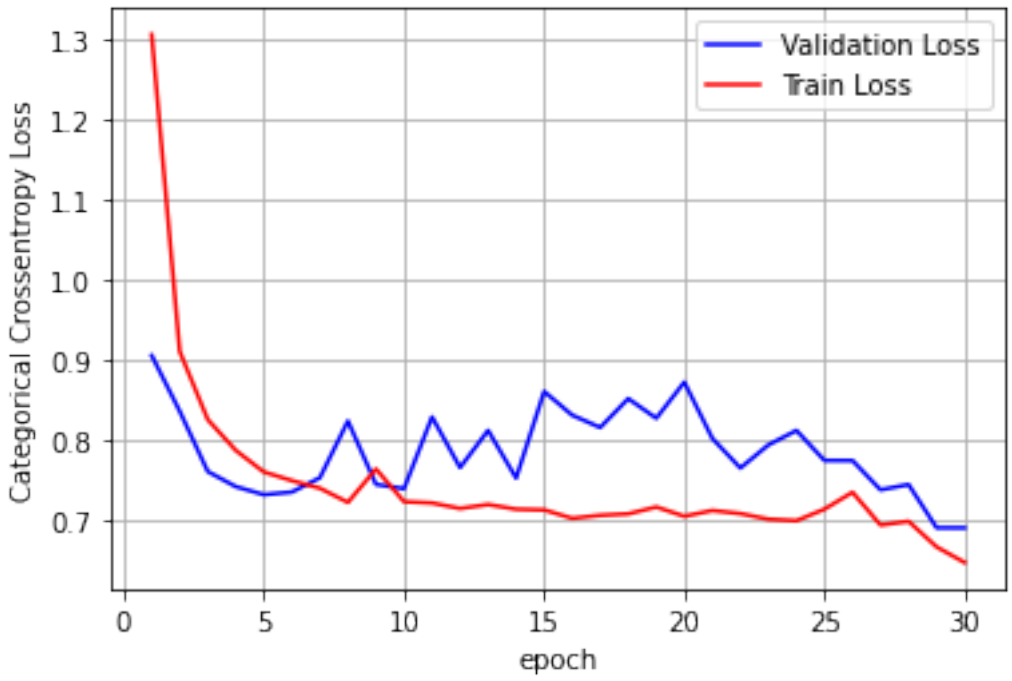
```
In [0]: # Training the model
        history=model.fit(X_train,
                  Y_train,
                  batch_size=batch_size,
                  validation_data=(X_test, Y_test),
                  epochs=epochs)
        score = model.evaluate(X_test, Y_test, verbose=0)
        print('Test loss:', score[0])
        print('Test accuracy:', score[1])
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 37s 5ms/step - loss: 1.3051 - acc: 0.4592 - val_lo
```

```
Epoch 2/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.9109 - acc: 0.5654 - val_lo
Epoch 3/30
7352/7352 [==============================] - 33s 5ms/step - loss: 0.8262 - acc: 0.5479 - val_lo
Epoch 4/30
7352/7352 [==============================] - 33s 5ms/step - loss: 0.7878 - acc: 0.5192 - val_lo
Epoch 5/30
7352/7352 [==============================] - 33s 5ms/step - loss: 0.7610 - acc: 0.5309 - val_lo
Epoch 6/30
7352/7352 [==============================] - 33s 4ms/step - loss: 0.7503 - acc: 0.5181 - val_lo
Epoch 7/30
7352/7352 [==============================] - 33s 5ms/step - loss: 0.7408 - acc: 0.5272 - val_lo
Epoch 8/30
7352/7352 [==============================] - 33s 4ms/step - loss: 0.7232 - acc: 0.5320 - val_lo
Epoch 9/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7647 - acc: 0.5267 - val_lo
Epoch 10/30
7352/7352 [==============================] - 33s 4ms/step - loss: 0.7241 - acc: 0.5341 - val_lo
Epoch 11/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7225 - acc: 0.5261 - val_lo
Epoch 12/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7156 - acc: 0.5331 - val_lo
Epoch 13/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7205 - acc: 0.5390 - val_lo
Epoch 14/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7147 - acc: 0.5328 - val_lo
Epoch 15/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7139 - acc: 0.5377 - val_lo
Epoch 16/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7034 - acc: 0.5356 - val_lo
Epoch 17/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7072 - acc: 0.5473 - val_lo
Epoch 18/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7089 - acc: 0.5394 - val_lo
Epoch 19/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7175 - acc: 0.5445 - val_lo
Epoch 20/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7059 - acc: 0.5529 - val_lo
Epoch 21/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7131 - acc: 0.5464 - val_lo
Epoch 22/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7093 - acc: 0.5611 - val_lo
Epoch 23/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7023 - acc: 0.5530 - val_lo
Epoch 24/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7004 - acc: 0.5562 - val_lo
Epoch 25/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7148 - acc: 0.5635 - val_lo
```

```
Epoch 26/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7358 - acc: 0.5622 - val_lo
Epoch 27/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.6950 - acc: 0.5804 - val_lo
Epoch 28/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.6997 - acc: 0.5891 - val_lo
Epoch 29/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.6678 - acc: 0.6124 - val_lo
Epoch 30/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.6484 - acc: 0.6279 - val_lo
Test loss: 0.6915614812784582
Test accuracy: 0.6012894468951476
```

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

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

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

        # print(history.history.keys())
        # dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
        # history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, v

        # we will get val_loss and val_acc only when you pass the paramter validation_data
        # val_loss : validation loss
        # val_acc : validation accuracy

        # loss : training loss
        # acc : train accuracy
        # for each key in histrory.histrory we will have a list of length equal to number of e

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

Test score: 0.6915614812784582
Test accuracy: 0.6012894468951476
```




```
In [0]: # Confusion Matrix
        print(confusion_matrix(Y_test, model.predict(X_test)))
```

Pred	LAYING	SITTING	WALKING	WALKING_UPSTAIRS
True				
LAYING	537	0	0	0
SITTING	7	484	0	0
STANDING	0	530	0	2
WALKING	0	0	329	167
WALKING_DOWNSTAIRS	0	0	410	10
WALKING_UPSTAIRS	0	0	49	422

2 LSTM Layers having 32 LSTM units + Dropout (0.6)

```
In [0]: # Initializing parameters
        epochs = 30
        batch_size = 32
        n_hidden = n_hidden

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

        #https://adventuresinmachinelearning.com/keras-lstm-tutorial/
        #Multilayer LSTM
```

```
        model.add(LSTM(32, return_sequences=True,input_shape=(timesteps, input_dim)))
        model.add(LSTM(16,input_shape=(timesteps, 32)))
        model.add(Dropout(0.6))
        model.add(Dense(n_classes, activation='sigmoid'))
        model.summary()
```

```
WARNING:tensorflow:Large dropout rate: 0.6 (>0.5). In TensorFlow 2.x, dropout() uses dropout ra
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_6 (LSTM)                (None, 128, 32)           5376
_________________________________________________________________
lstm_7 (LSTM)                (None, 16)                3136
_________________________________________________________________
dropout_4 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 102
=================================================================
Total params: 8,614
Trainable params: 8,614
Non-trainable params: 0
_________________________________________________________________
```

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

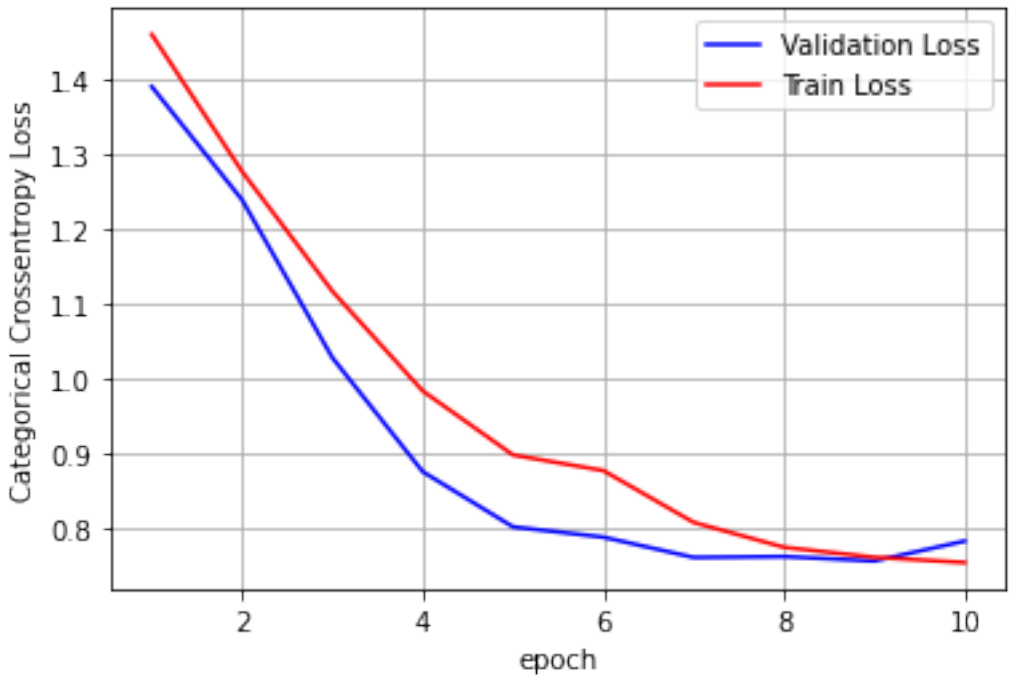
```
In [0]: # Training the model
        history=model.fit(X_train,
                  Y_train,
                  batch_size=batch_size,
                  validation_data=(X_test, Y_test),
                  epochs=epochs)
        score = model.evaluate(X_test, Y_test, verbose=0)
        print('Test loss:', score[0])
        print('Test accuracy:', score[1])
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/10
7352/7352 [==============================] - 30s 4ms/step - loss: 1.4606 - acc: 0.3541 - val_lo
Epoch 2/10
7352/7352 [==============================] - 28s 4ms/step - loss: 1.2772 - acc: 0.4445 - val_lo
Epoch 3/10
7352/7352 [==============================] - 29s 4ms/step - loss: 1.1170 - acc: 0.5462 - val_lo
Epoch 4/10
```

```
7352/7352 [==============================] - 29s 4ms/step - loss: 0.9835 - acc: 0.5845 - val_lo
Epoch 5/10
7352/7352 [==============================] - 29s 4ms/step - loss: 0.8978 - acc: 0.6122 - val_lo
Epoch 6/10
7352/7352 [==============================] - 29s 4ms/step - loss: 0.8770 - acc: 0.6058 - val_lo
Epoch 7/10
7352/7352 [==============================] - 27s 4ms/step - loss: 0.8076 - acc: 0.6306 - val_lo
Epoch 8/10
7352/7352 [==============================] - 27s 4ms/step - loss: 0.7743 - acc: 0.6310 - val_lo
Epoch 9/10
7352/7352 [==============================] - 26s 4ms/step - loss: 0.7609 - acc: 0.6450 - val_lo
Epoch 10/10
7352/7352 [==============================] - 27s 4ms/step - loss: 0.7539 - acc: 0.6507 - val_lo
Test loss: 0.7825939169891171
Test accuracy: 0.6223277909738717
```

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

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

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

        # print(history.history.keys())
        # dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
        # history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, v

        # we will get val_loss and val_acc only when you pass the paramter validation_data
        # val_loss : validation loss
        # val_acc : validation accuracy

        # loss : training loss
        # acc : train accuracy
        # for each key in histrory.histrory we will have a list of length equal to number of ep

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

```
Test score: 0.7825939169891171
Test accuracy: 0.6223277909738717
```




```
In [0]: # Confusion Matrix
        print(confusion_matrix(Y_test, model.predict(X_test)))
```

Pred	LAYING	SITTING	STANDING	WALKING
True				
LAYING	510	0	27	0
SITTING	0	365	116	10
STANDING	0	51	463	18
WALKING	0	0	0	496
WALKING_DOWNSTAIRS	0	0	0	420
WALKING_UPSTAIRS	0	0	0	471

2 LSTM Layers having 32 LSTM units + Dropout (0.5)

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

        #https://adventuresinmachinelearning.com/keras-lstm-tutorial/
        #Multilayer LSTM

        model.add(LSTM(32,return_sequences=True,input_shape=(timesteps, input_dim)))
        model.add(Dropout(0.5))
        model.add(LSTM(32,input_shape=(timesteps, 32)))
        model.add(BatchNormalization())
```

```
        model.add(Dropout(0.5))
        model.add(Dense(n_classes, activation='sigmoid'))
        model.summary()
```

```
Model: "sequential_15"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_28 (LSTM)               (None, 128, 32)           5376
_________________________________________________________________
dropout_19 (Dropout)         (None, 128, 32)           0
_________________________________________________________________
lstm_29 (LSTM)               (None, 32)                8320
_________________________________________________________________
batch_normalization_6 (Batch (None, 32)                128
_________________________________________________________________
dropout_20 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_14 (Dense)             (None, 6)                 198
=================================================================
Total params: 14,022
Trainable params: 13,958
Non-trainable params: 64
_________________________________________________________________
```

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

```
In [0]: # Training the model
        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, verbose=0)
        print('Test loss:', score[0])
        print('Test accuracy:', score[1])
```

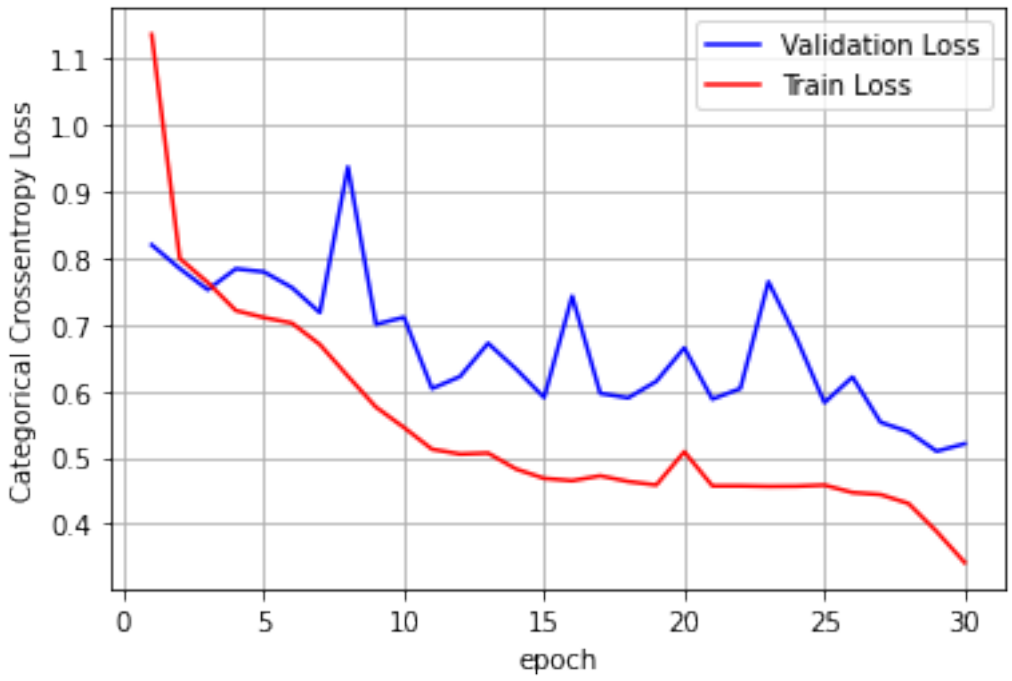
```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 38s 5ms/step - loss: 1.1372 - acc: 0.5460 - val_lo
Epoch 2/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7995 - acc: 0.6140 - val_lo
Epoch 3/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7634 - acc: 0.6027 - val_lo
Epoch 4/30
```

```
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7211 - acc: 0.6030 - val_lo
Epoch 5/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7106 - acc: 0.5915 - val_lo
Epoch 6/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.7023 - acc: 0.5811 - val_lo
Epoch 7/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.6699 - acc: 0.6011 - val_lo
Epoch 8/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.6222 - acc: 0.6196 - val_lo
Epoch 9/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.5757 - acc: 0.6371 - val_lo
Epoch 10/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.5449 - acc: 0.6503 - val_lo
Epoch 11/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.5121 - acc: 0.6605 - val_lo
Epoch 12/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.5049 - acc: 0.6624 - val_lo
Epoch 13/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.5063 - acc: 0.6647 - val_lo
Epoch 14/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.4823 - acc: 0.6672 - val_lo
Epoch 15/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.4682 - acc: 0.6674 - val_lo
Epoch 16/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.4647 - acc: 0.6672 - val_lo
Epoch 17/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.4721 - acc: 0.6699 - val_lo
Epoch 18/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.4635 - acc: 0.6732 - val_lo
Epoch 19/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.4582 - acc: 0.6766 - val_lo
Epoch 20/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.5083 - acc: 0.6711 - val_lo
Epoch 21/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.4569 - acc: 0.6798 - val_lo
Epoch 22/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.4569 - acc: 0.6744 - val_lo
Epoch 23/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.4562 - acc: 0.6794 - val_lo
Epoch 24/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.4564 - acc: 0.6825 - val_lo
Epoch 25/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.4581 - acc: 0.6785 - val_lo
Epoch 26/30
7352/7352 [==============================] - 33s 4ms/step - loss: 0.4469 - acc: 0.6940 - val_lo
Epoch 27/30
7352/7352 [==============================] - 33s 4ms/step - loss: 0.4438 - acc: 0.7024 - val_lo
Epoch 28/30
```

```
7352/7352 [==============================] - 32s 4ms/step - loss: 0.4303 - acc: 0.7190 - val_lo
Epoch 29/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.3883 - acc: 0.7739 - val_lo
Epoch 30/30
7352/7352 [==============================] - 32s 4ms/step - loss: 0.3414 - acc: 0.8028 - val_lo
Test loss: 0.5203237037180645
Test accuracy: 0.7712928401764506
```

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

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

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

        # print(history.history.keys())
        # dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
        # history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, v

        # we will get val_loss and val_acc only when you pass the paramter validation_data
        # val_loss : validation loss
        # val_acc : validation accuracy

        # loss : training loss
        # acc : train accuracy
        # for each key in histrory.histrory we will have a list of length equal to number of e

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

```
Test score: 0.5203237037180645
Test accuracy: 0.7712928401764506
```




```
In [0]: # Confusion Matrix
        print(confusion_matrix(Y_test, model.predict(X_test)))
```

Pred	LAYING	SITTING	STANDING	WALKING	WALKING_UPSTAIRS
True					
LAYING	537	0	0	0	0
SITTING	8	368	98	0	17
STANDING	0	74	458	0	0
WALKING	0	0	5	450	41
WALKING_DOWNSTAIRS	0	0	0	1	419
WALKING_UPSTAIRS	0	2	1	8	460

2 LSTM Layers having 32 LSTM units + Dropout (0.4)

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

        #https://adventuresinmachinelearning.com/keras-lstm-tutorial/
        #Multilayer LSTM

        model.add(LSTM(32, return_sequences=True,input_shape=(timesteps, input_dim)))
        model.add(LSTM(16,input_shape=(timesteps, 32)))
        model.add(Dropout(0.4))
        model.add(Dense(n_classes, activation='sigmoid'))
        model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_10 (LSTM)               (None, 128, 32)           5376
_________________________________________________________________
lstm_11 (LSTM)               (None, 16)                3136
_________________________________________________________________
dropout_6 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_6 (Dense)              (None, 6)                 102
=================================================================
Total params: 8,614
Trainable params: 8,614
Non-trainable params: 0
_________________________________________________________________
```

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

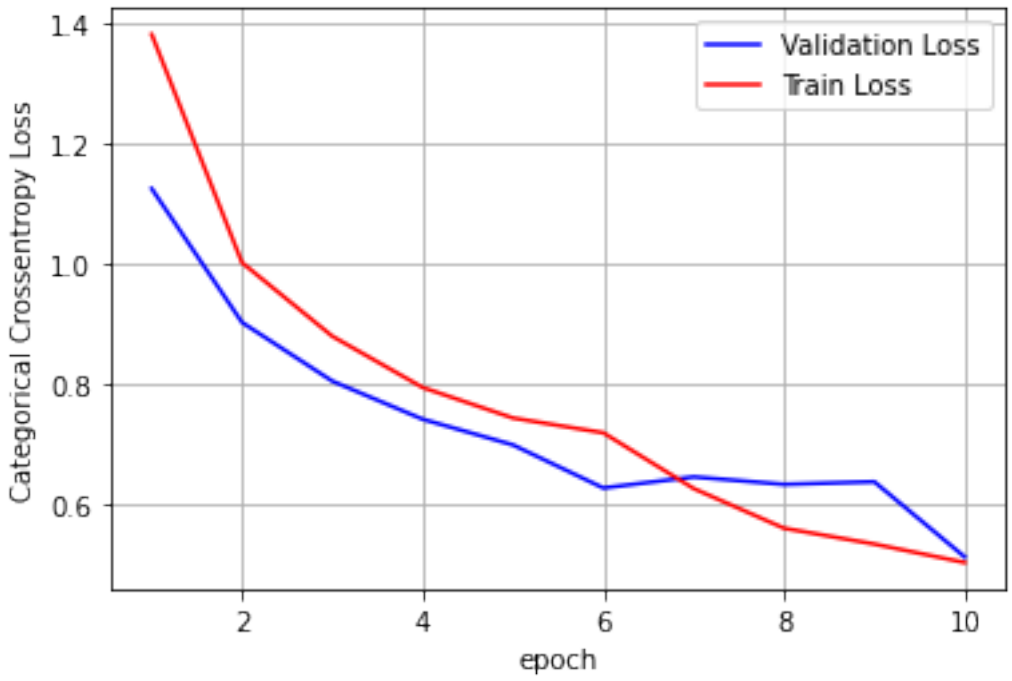
```
In [0]: # Training the model
        history=model.fit(X_train,
                  Y_train,
                  batch_size=batch_size,
                  validation_data=(X_test, Y_test),
                  epochs=epochs)
        score = model.evaluate(X_test, Y_test, verbose=0)
        print('Test loss:', score[0])
        print('Test accuracy:', score[1])
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/10
7352/7352 [==============================] - 28s 4ms/step - loss: 1.3801 - acc: 0.4603 - val_lo
Epoch 2/10
7352/7352 [==============================] - 26s 4ms/step - loss: 1.0014 - acc: 0.5684 - val_lo
Epoch 3/10
7352/7352 [==============================] - 25s 3ms/step - loss: 0.8792 - acc: 0.6249 - val_lo
Epoch 4/10
7352/7352 [==============================] - 26s 3ms/step - loss: 0.7942 - acc: 0.6629 - val_lo
Epoch 5/10
7352/7352 [==============================] - 26s 4ms/step - loss: 0.7436 - acc: 0.6930 - val_lo
Epoch 6/10
7352/7352 [==============================] - 26s 3ms/step - loss: 0.7190 - acc: 0.7035 - val_lo
Epoch 7/10
7352/7352 [==============================] - 26s 3ms/step - loss: 0.6266 - acc: 0.7406 - val_lo
```

```
Epoch 8/10
7352/7352 [==============================] - 25s 3ms/step - loss: 0.5607 - acc: 0.7655 - val_lo
Epoch 9/10
7352/7352 [==============================] - 26s 3ms/step - loss: 0.5344 - acc: 0.7825 - val_lo
Epoch 10/10
7352/7352 [==============================] - 26s 3ms/step - loss: 0.5038 - acc: 0.7958 - val_lo
Test loss: 0.5131604391759155
Test accuracy: 0.7692568714148641
```

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

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

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

        # print(history.history.keys())
        # dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
        # history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, v

        # we will get val_loss and val_acc only when you pass the paramter validation_data
        # val_loss : validation loss
        # val_acc : validation accuracy

        # loss : training loss
        # acc : train accuracy
        # for each key in histrory.histrory we will have a list of length equal to number of ep

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

```
Test score: 0.5131604391759155
Test accuracy: 0.7692568714148641
```




```
In [0]: # Confusion Matrix
        print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  ...  WALKING_DOWNSTAIRS  WALKING_UPSTAIRS
True                                 ...
LAYING                 537        0  ...                   0                 0
SITTING                  0      463  ...                   7                 0
STANDING                 0      182  ...                   1                 3
WALKING                  0        1  ...                   2                43
WALKING_DOWNSTAIRS       0        1  ...                 167               234
WALKING_UPSTAIRS         0        0  ...                  96               310

[6 rows x 6 columns]
```

2 LSTM Layers having 64 LSTM units + Dropout (0.5)

```
In [0]: # Initializing parameters
        epochs = 30
        batch_size = 32
        n_hidden = n_hidden
```

```
In [0]: from keras.layers.normalization import BatchNormalization
        # Initiliazing the sequential model
        model = Sequential()
```

```
        #https://adventuresinmachinelearning.com/keras-lstm-tutorial/
        #Multilayer LSTM

        model.add(LSTM(128, return_sequences=True,input_shape=(timesteps, input_dim)))
        model.add(Dropout(0.5))
        model.add(LSTM(128,input_shape=(timesteps,input_dim)))
        model.add(BatchNormalization())
        model.add(Dropout(0.5))
        model.add(Dense(n_classes, activation='sigmoid'))
        model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 128, 128)          70656
_________________________________________________________________
dropout_3 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
lstm_4 (LSTM)                (None, 128)               131584
_________________________________________________________________
batch_normalization_2 (Batch (None, 128)               512
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 774
=================================================================
Total params: 203,526
Trainable params: 203,270
Non-trainable params: 256
_________________________________________________________________
```

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

In [0]: # patient early stopping
        from keras.callbacks import EarlyStopping
        es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=20)

In [0]: # ModelCheckpoint
        from keras.callbacks import ModelCheckpoint
        mc = ModelCheckpoint('30epoch_model.h5', monitor='val_acc', mode='max', verbose=1, save

In [0]: # Training the model
        history=model.fit(X_train,
```

```
                Y_train,
                batch_size=batch_size,
                validation_data=(X_test, Y_test),
                epochs=epochs, callbacks=[es, mc])
        score = model.evaluate(X_test, Y_test, verbose=0)
        print('Test loss:', score[0])
        print('Test accuracy:', score[1])

Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 109s 15ms/step - loss: 1.1494 - acc: 0.4736 - val_

Epoch 00001: val_acc improved from -inf to 0.41534, saving model to 30epoch_model.h5
Epoch 2/30
7352/7352 [==============================] - 98s 13ms/step - loss: 0.9132 - acc: 0.5966 - val_l

Epoch 00002: val_acc improved from 0.41534 to 0.63658, saving model to 30epoch_model.h5
Epoch 3/30
7352/7352 [==============================] - 99s 13ms/step - loss: 0.7147 - acc: 0.6382 - val_l

Epoch 00003: val_acc did not improve from 0.63658
Epoch 4/30
7352/7352 [==============================] - 100s 14ms/step - loss: 0.6770 - acc: 0.6609 - val_

Epoch 00004: val_acc improved from 0.63658 to 0.67832, saving model to 30epoch_model.h5
Epoch 5/30
7352/7352 [==============================] - 99s 13ms/step - loss: 0.6310 - acc: 0.7046 - val_l

Epoch 00005: val_acc did not improve from 0.67832
Epoch 6/30
7352/7352 [==============================] - 99s 13ms/step - loss: 0.7059 - acc: 0.6884 - val_l

Epoch 00006: val_acc improved from 0.67832 to 0.73023, saving model to 30epoch_model.h5
Epoch 7/30
7352/7352 [==============================] - 99s 13ms/step - loss: 0.4577 - acc: 0.7901 - val_l

Epoch 00007: val_acc improved from 0.73023 to 0.76315, saving model to 30epoch_model.h5
Epoch 8/30
7352/7352 [==============================] - 98s 13ms/step - loss: 0.3393 - acc: 0.8640 - val_l

Epoch 00008: val_acc improved from 0.76315 to 0.86121, saving model to 30epoch_model.h5
Epoch 9/30
7352/7352 [==============================] - 99s 13ms/step - loss: 0.2080 - acc: 0.9256 - val_l

Epoch 00009: val_acc improved from 0.86121 to 0.89243, saving model to 30epoch_model.h5
Epoch 10/30
7352/7352 [==============================] - 99s 13ms/step - loss: 0.1537 - acc: 0.9410 - val_l
```

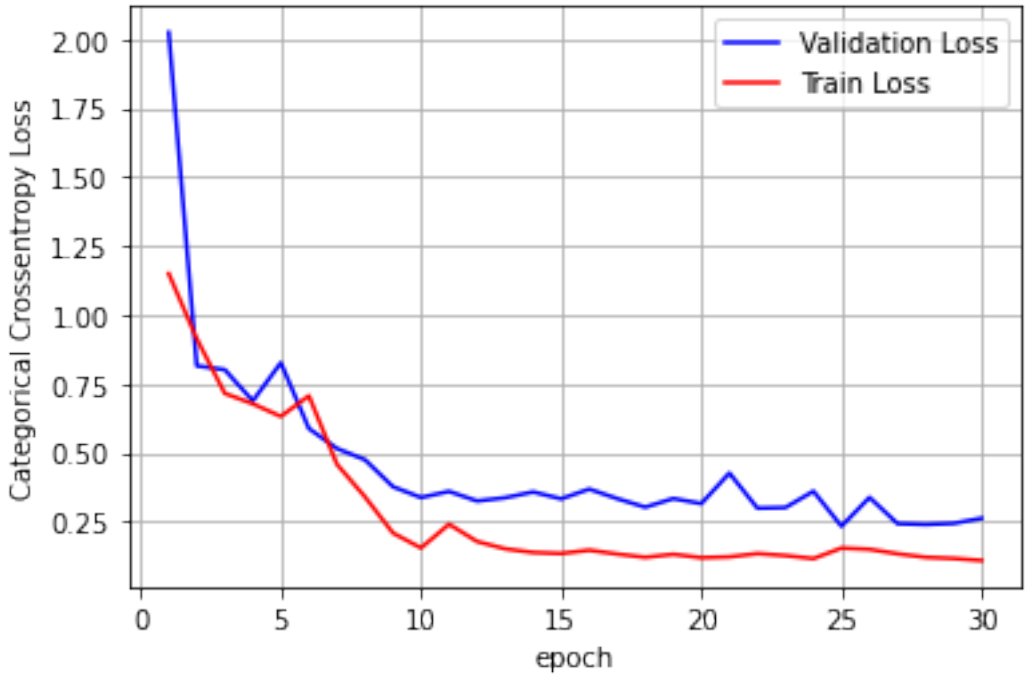
```
Epoch 00010: val_acc improved from 0.89243 to 0.90126, saving model to 30epoch_model.h5
Epoch 11/30
7352/7352 [==============================] - 99s 13ms/step - loss: 0.2399 - acc: 0.9142 - val_l

Epoch 00011: val_acc did not improve from 0.90126
Epoch 12/30
7352/7352 [==============================] - 99s 13ms/step - loss: 0.1771 - acc: 0.9353 - val_l

Epoch 00012: val_acc improved from 0.90126 to 0.90736, saving model to 30epoch_model.h5
Epoch 13/30
7352/7352 [==============================] - 98s 13ms/step - loss: 0.1506 - acc: 0.9381 - val_l

Epoch 00013: val_acc did not improve from 0.90736
Epoch 14/30
7352/7352 [==============================] - 100s 14ms/step - loss: 0.1373 - acc: 0.9436 - val

Epoch 00014: val_acc did not improve from 0.90736
Epoch 15/30
7352/7352 [==============================] - 100s 14ms/step - loss: 0.1336 - acc: 0.9433 - val

Epoch 00015: val_acc did not improve from 0.90736
Epoch 16/30
7352/7352 [==============================] - 100s 14ms/step - loss: 0.1457 - acc: 0.9434 - val

Epoch 00016: val_acc did not improve from 0.90736
Epoch 17/30
7352/7352 [==============================] - 100s 14ms/step - loss: 0.1316 - acc: 0.9445 - val

Epoch 00017: val_acc did not improve from 0.90736
Epoch 18/30
7352/7352 [==============================] - 101s 14ms/step - loss: 0.1197 - acc: 0.9508 - val

Epoch 00018: val_acc improved from 0.90736 to 0.91347, saving model to 30epoch_model.h5
Epoch 19/30
7352/7352 [==============================] - 100s 14ms/step - loss: 0.1307 - acc: 0.9461 - val

Epoch 00019: val_acc did not improve from 0.91347
Epoch 20/30
7352/7352 [==============================] - 101s 14ms/step - loss: 0.1184 - acc: 0.9501 - val

Epoch 00020: val_acc improved from 0.91347 to 0.91449, saving model to 30epoch_model.h5
Epoch 21/30
7352/7352 [==============================] - 100s 14ms/step - loss: 0.1214 - acc: 0.9521 - val

Epoch 00021: val_acc did not improve from 0.91449
Epoch 22/30
7352/7352 [==============================] - 101s 14ms/step - loss: 0.1333 - acc: 0.9491 - val
```

```
Epoch 00022: val_acc improved from 0.91449 to 0.91754, saving model to 30epoch_model.h5
Epoch 23/30
7352/7352 [==============================] - 100s 14ms/step - loss: 0.1261 - acc: 0.9474 - val

Epoch 00023: val_acc did not improve from 0.91754
Epoch 24/30
7352/7352 [==============================] - 99s 13ms/step - loss: 0.1152 - acc: 0.9514 - val_

Epoch 00024: val_acc did not improve from 0.91754
Epoch 25/30
7352/7352 [==============================] - 99s 13ms/step - loss: 0.1533 - acc: 0.9436 - val_

Epoch 00025: val_acc did not improve from 0.91754
Epoch 26/30
7352/7352 [==============================] - 99s 13ms/step - loss: 0.1488 - acc: 0.9372 - val_

Epoch 00026: val_acc improved from 0.91754 to 0.91822, saving model to 30epoch_model.h5
Epoch 27/30
7352/7352 [==============================] - 99s 13ms/step - loss: 0.1326 - acc: 0.9448 - val_

Epoch 00027: val_acc did not improve from 0.91822
Epoch 28/30
7352/7352 [==============================] - 100s 14ms/step - loss: 0.1198 - acc: 0.9482 - val

Epoch 00028: val_acc improved from 0.91822 to 0.91958, saving model to 30epoch_model.h5
Epoch 29/30
7352/7352 [==============================] - 99s 13ms/step - loss: 0.1157 - acc: 0.9527 - val_

Epoch 00029: val_acc improved from 0.91958 to 0.92603, saving model to 30epoch_model.h5
Epoch 30/30
7352/7352 [==============================] - 99s 13ms/step - loss: 0.1081 - acc: 0.9544 - val_

Epoch 00030: val_acc did not improve from 0.92603
Test loss: 0.2615530221774433
Test accuracy: 0.9199185612487275
```

```
In [0]: import matplotlib.pyplot as plt
        score = model.evaluate(X_test, Y_test, verbose=0)
        print('Test score:', score[0])
        print('Test accuracy:', score[1])

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

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

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

```
Test score: 0.2615530221774433
Test accuracy: 0.9199185612487275
```



In [0]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  ...  WALKING_DOWNSTAIRS  WALKING_UPSTAIRS
True                                 ...
LAYING                 537        0  ...                   0                 0
SITTING                  5      384  ...                   0                 4
STANDING                 0       98  ...                   0                 1
WALKING                  0        0  ...                  25                 2
WALKING_DOWNSTAIRS       0        0  ...                 417                 2
WALKING_UPSTAIRS         0        0  ...                   0               471

[6 rows x 6 columns]
```

In [0]:
```
# Save the model
model.save("30epoch_model.h5")
```

```
In [0]: from numpy import loadtxt
        from keras.models import load_model
        models = load_model('30epoch_model.h5')

In [0]: models.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 128, 128)          70656
_________________________________________________________________
dropout_3 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
lstm_4 (LSTM)                (None, 128)               131584
_________________________________________________________________
batch_normalization_2 (Batch (None, 128)               512
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 774
=================================================================
Total params: 203,526
Trainable params: 203,270
Non-trainable params: 256
_________________________________________________________________
```

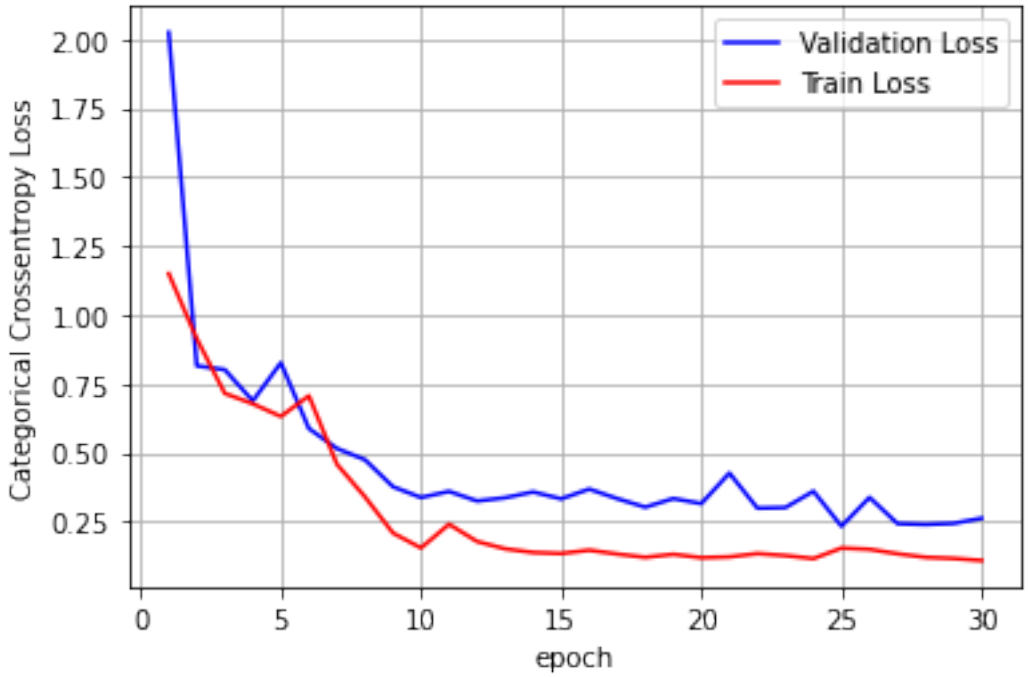
```
In [0]: # Compiling the model
        models.compile(loss='categorical_crossentropy',
                       optimizer='adam',
                       metrics=['accuracy'])

In [0]: # patient early stopping
        from keras.callbacks import EarlyStopping
        es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=20)

In [0]: # ModelCheckpoint
        from keras.callbacks import ModelCheckpoint
        mc = ModelCheckpoint('80epoch_model.h5', monitor='val_acc', mode='max', verbose=1, save

In [0]: # Training the model
        history5=models.fit(X_train,
                  Y_train,
                  batch_size=batch_size,
                  validation_data=(X_test, Y_test),
                  epochs=10, callbacks=[es, mc])
        score = models.evaluate(X_test, Y_test, verbose=0)
        print('Test loss:', score[0])
        print('Test accuracy:', score[1])
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/10
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1179 - acc: 0.9487 - val

Epoch 00001: val_acc improved from -inf to 0.92738, saving model to 80epoch_model.h5
Epoch 2/10
7352/7352 [==============================] - 99s 13ms/step - loss: 0.1437 - acc: 0.9437 - val_

Epoch 00002: val_acc did not improve from 0.92738
Epoch 3/10
7352/7352 [==============================] - 99s 14ms/step - loss: 0.1181 - acc: 0.9538 - val_

Epoch 00003: val_acc did not improve from 0.92738
Epoch 4/10
7352/7352 [==============================] - 100s 14ms/step - loss: 0.1186 - acc: 0.9497 - val

Epoch 00004: val_acc did not improve from 0.92738
Epoch 5/10
7352/7352 [==============================] - 99s 14ms/step - loss: 0.1625 - acc: 0.9354 - val_

Epoch 00005: val_acc did not improve from 0.92738
Epoch 6/10
7352/7352 [==============================] - 99s 13ms/step - loss: 0.1477 - acc: 0.9407 - val_

Epoch 00006: val_acc did not improve from 0.92738
Epoch 7/10
7352/7352 [==============================] - 101s 14ms/step - loss: 0.1170 - acc: 0.9463 - val

Epoch 00007: val_acc did not improve from 0.92738
Epoch 8/10
7352/7352 [==============================] - 106s 14ms/step - loss: 0.1133 - acc: 0.9528 - val

Epoch 00008: val_acc did not improve from 0.92738
Epoch 9/10
7352/7352 [==============================] - 102s 14ms/step - loss: 0.1179 - acc: 0.9513 - val

Epoch 00009: val_acc did not improve from 0.92738
Epoch 10/10
7352/7352 [==============================] - 105s 14ms/step - loss: 0.1115 - acc: 0.9531 - val

Epoch 00010: val_acc did not improve from 0.92738
Test loss: 0.3136977848725028
Test accuracy: 0.9148286392941974
```

```
In [0]: import matplotlib.pyplot as plt
        score = models.evaluate(X_test, Y_test, verbose=0)
        print('Test score:', score[0])
```

```
        print('Test accuracy:', score[1])

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

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

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

```
Test score: 0.3136977848725028
Test accuracy: 0.9148286392941974
```




```
In [0]: # Save the model
        models.save("80epoch_model.h5")
```

2 LSTM Layers having 128 LSTM units + Dropout (0.5)

```
In [0]: # Initializing parameters
        epochs = 30
        batch_size = 32
        n_hidden = n_hidden
```

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

        #https://adventuresinmachinelearning.com/keras-lstm-tutorial/
        #Multilayer LSTM

        model.add(LSTM(128, return_sequences=True,input_shape=(timesteps, input_dim)))
        model.add(Dropout(0.5))
        model.add(LSTM(128,input_shape=(timesteps, 32)))
        model.add(BatchNormalization())
        model.add(Dropout(0.5))
        model.add(Dense(n_classes, activation='sigmoid'))
        model.summary()
```

```
Model: "sequential_16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_30 (LSTM)               (None, 128, 128)          70656
_________________________________________________________________
dropout_21 (Dropout)         (None, 128, 128)          0
_________________________________________________________________
lstm_31 (LSTM)               (None, 128)               131584
_________________________________________________________________
batch_normalization_7 (Batch (None, 128)               512
_________________________________________________________________
dropout_22 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_15 (Dense)             (None, 6)                 774
=================================================================
Total params: 203,526
Trainable params: 203,270
Non-trainable params: 256
_________________________________________________________________
```

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

```
In [0]: # patient early stopping
        from keras.callbacks import EarlyStopping
        es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=20)
```

```
In [0]: # ModelCheckpoint
        from keras.callbacks import ModelCheckpoint
        mc = ModelCheckpoint('128layer_model.h5', monitor='val_acc', mode='max', verbose=1, sav
```

```
In [0]: # Training the model
        history=model.fit(X_train,
                  Y_train,
                  batch_size=batch_size,
                  validation_data=(X_test, Y_test),
                  epochs=epochs, callbacks=[es, mc])
        score = model.evaluate(X_test, Y_test, verbose=0)
        print('Test loss:', score[0])
        print('Test accuracy:', score[1])

Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 93s 13ms/step - loss: 0.9964 - acc: 0.5520 - val_l

Epoch 00001: val_acc improved from -inf to 0.59688, saving model to 128layer_model.h5
Epoch 2/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.7406 - acc: 0.6270 - val_l

Epoch 00002: val_acc did not improve from 0.59688
Epoch 3/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.6767 - acc: 0.6398 - val_l

Epoch 00003: val_acc improved from 0.59688 to 0.61113, saving model to 128layer_model.h5
Epoch 4/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.6673 - acc: 0.6425 - val_l

Epoch 00004: val_acc improved from 0.61113 to 0.61927, saving model to 128layer_model.h5
Epoch 5/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.6483 - acc: 0.6474 - val_l

Epoch 00005: val_acc did not improve from 0.61927
Epoch 6/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.6422 - acc: 0.6564 - val_l

Epoch 00006: val_acc did not improve from 0.61927
Epoch 7/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.7330 - acc: 0.6291 - val_l

Epoch 00007: val_acc improved from 0.61927 to 0.61995, saving model to 128layer_model.h5
Epoch 8/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.6433 - acc: 0.6530 - val_l

Epoch 00008: val_acc did not improve from 0.61995
Epoch 9/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.6379 - acc: 0.6581 - val_l

Epoch 00009: val_acc did not improve from 0.61995
Epoch 10/30
```

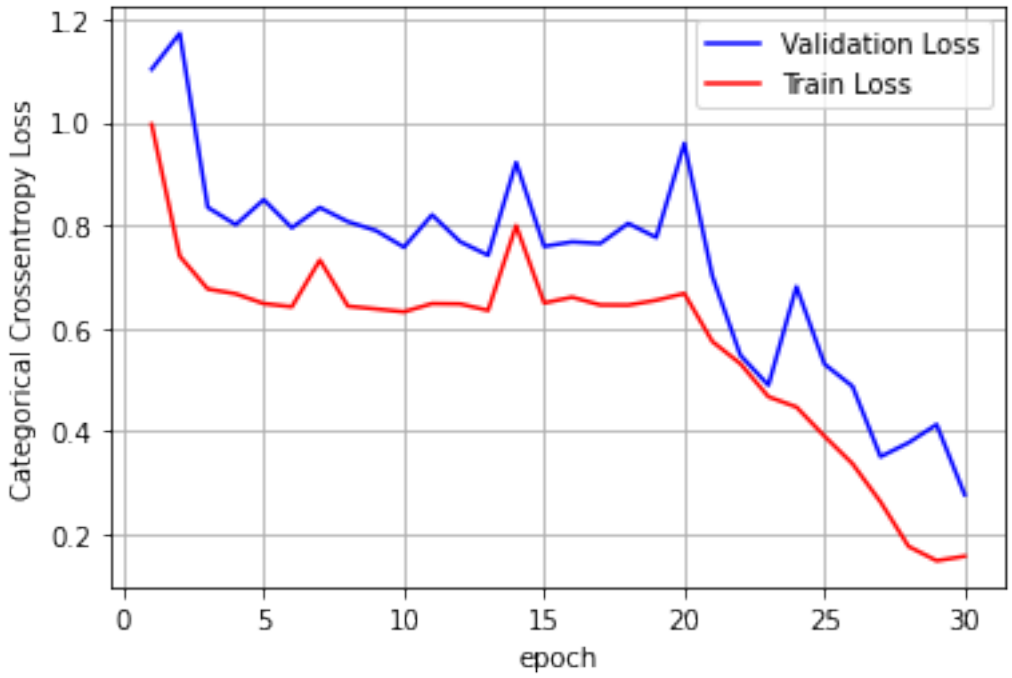
```
7352/7352 [==============================] - 89s 12ms/step - loss: 0.6324 - acc: 0.6575 - val_l

Epoch 00010: val_acc improved from 0.61995 to 0.62979, saving model to 128layer_model.h5
Epoch 11/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.6484 - acc: 0.6615 - val_l

Epoch 00011: val_acc did not improve from 0.62979
Epoch 12/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.6477 - acc: 0.6642 - val_l

Epoch 00012: val_acc did not improve from 0.62979
Epoch 13/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.6352 - acc: 0.6601 - val_l

Epoch 00013: val_acc did not improve from 0.62979
Epoch 14/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.7999 - acc: 0.6264 - val_l

Epoch 00014: val_acc did not improve from 0.62979
Epoch 15/30
7352/7352 [==============================] - 87s 12ms/step - loss: 0.6497 - acc: 0.6582 - val_l

Epoch 00015: val_acc did not improve from 0.62979
Epoch 16/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.6608 - acc: 0.6670 - val_l

Epoch 00016: val_acc did not improve from 0.62979
Epoch 17/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.6458 - acc: 0.6658 - val_l

Epoch 00017: val_acc improved from 0.62979 to 0.63251, saving model to 128layer_model.h5
Epoch 18/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.6455 - acc: 0.6591 - val_l

Epoch 00018: val_acc did not improve from 0.63251
Epoch 19/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.6549 - acc: 0.6605 - val_l

Epoch 00019: val_acc did not improve from 0.63251
Epoch 20/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.6680 - acc: 0.6529 - val_l

Epoch 00020: val_acc did not improve from 0.63251
Epoch 21/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.5748 - acc: 0.6604 - val_l

Epoch 00021: val_acc did not improve from 0.63251
Epoch 22/30
```

```
7352/7352 [==============================] - 89s 12ms/step - loss: 0.5317 - acc: 0.6591 - val_l

Epoch 00022: val_acc did not improve from 0.63251
Epoch 23/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.4675 - acc: 0.6613 - val_l

Epoch 00023: val_acc did not improve from 0.63251
Epoch 24/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.4476 - acc: 0.6757 - val_l

Epoch 00024: val_acc did not improve from 0.63251
Epoch 25/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.3909 - acc: 0.7699 - val_l

Epoch 00025: val_acc improved from 0.63251 to 0.75534, saving model to 128layer_model.h5
Epoch 26/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.3375 - acc: 0.8035 - val_l

Epoch 00026: val_acc improved from 0.75534 to 0.78045, saving model to 128layer_model.h5
Epoch 27/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.2634 - acc: 0.8607 - val_l

Epoch 00027: val_acc improved from 0.78045 to 0.92467, saving model to 128layer_model.h5
Epoch 28/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.1769 - acc: 0.9404 - val_l

Epoch 00028: val_acc did not improve from 0.92467
Epoch 29/30
7352/7352 [==============================] - 87s 12ms/step - loss: 0.1492 - acc: 0.9448 - val_l

Epoch 00029: val_acc did not improve from 0.92467
Epoch 30/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.1578 - acc: 0.9357 - val_l

Epoch 00030: val_acc did not improve from 0.92467
Test loss: 0.27744862166333983
Test accuracy: 0.9178825924669155
```

```
In [0]: import matplotlib.pyplot as plt
        score = model.evaluate(X_test, Y_test, verbose=0)
        print('Test score:', score[0])
        print('Test accuracy:', score[1])

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

        # list of epoch numbers
```

```
        x = list(range(1,epochs+1))

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

```
Test score: 0.27744862166333983
Test accuracy: 0.9178825924669155
```



In [0]:
```
        # Confusion Matrix
        print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  ...  WALKING_DOWNSTAIRS  WALKING_UPSTAIRS
True                                 ...
LAYING                 537        0  ...                   0                 0
SITTING                  5      305  ...                   0                19
STANDING                 0       26  ...                   0                 0
WALKING                  0        0  ...                  19                 8
WALKING_DOWNSTAIRS       0        0  ...                 419                 1
WALKING_UPSTAIRS         0        0  ...                  22               449

[6 rows x 6 columns]
```

In [0]:
```
        # Save the model
        model.save("128layer_model.h5")
```

```
In [0]: from numpy import loadtxt
        from keras.models import load_model
        model_1 = load_model('128layer_model.h5')

In [0]: model_1.summary()

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 128, 128)          70656
_________________________________________________________________
dropout_1 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
lstm_2 (LSTM)                (None, 128)               131584
_________________________________________________________________
batch_normalization_1 (Batch (None, 128)               512
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 774
=================================================================
Total params: 203,526
Trainable params: 203,270
Non-trainable params: 256
_________________________________________________________________


In [0]: # patient early stopping
        from keras.callbacks import EarlyStopping
        es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=20)

In [0]: # ModelCheckpoint
        from keras.callbacks import ModelCheckpoint
        mc = ModelCheckpoint('30epoch_model.h5', monitor='val_acc', mode='max', verbose=1, save

In [0]: # Training the model
        history3=model_1.fit(X_train,
                  Y_train,
                  batch_size=batch_size,
                  validation_data=(X_test, Y_test),
                  epochs=30, callbacks=[es, mc])
        score = model_1.evaluate(X_test, Y_test, verbose=0)
        print('Test loss:', score[0])
        print('Test accuracy:', score[1])

Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.2056 - acc: 0.9416 - val_
```

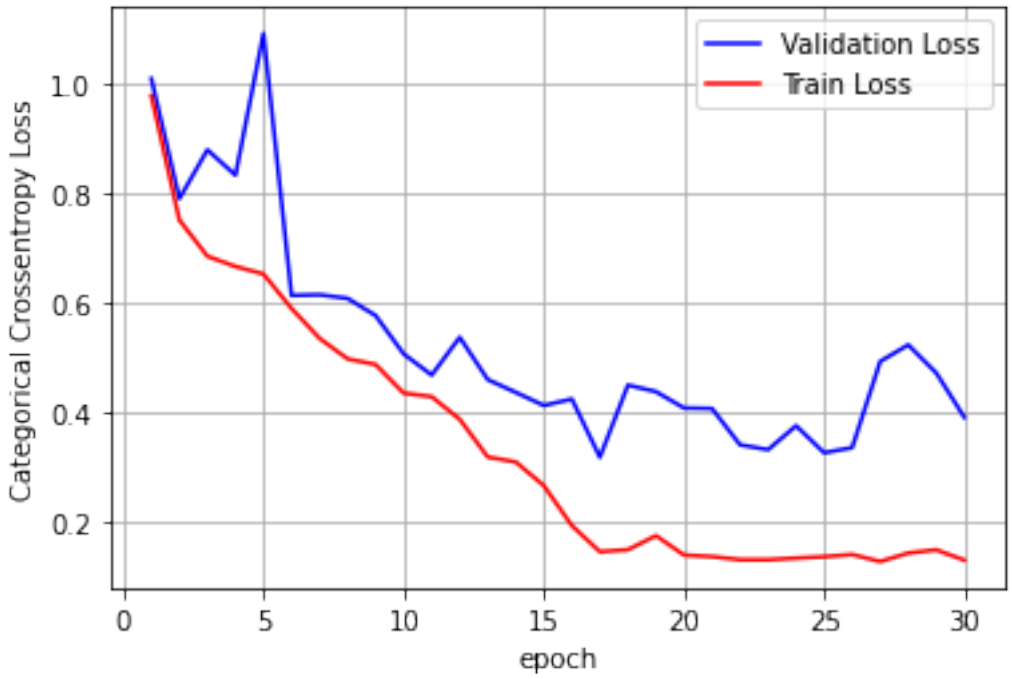
```
Epoch 00001: val_acc improved from -inf to 0.91110, saving model to 30epoch_model.h5
Epoch 2/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.1639 - acc: 0.9415 - val_l

Epoch 00002: val_acc improved from 0.91110 to 0.92162, saving model to 30epoch_model.h5
Epoch 3/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.1543 - acc: 0.9446 - val_l

Epoch 00003: val_acc did not improve from 0.92162
Epoch 4/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.1392 - acc: 0.9510 - val_l

Epoch 00004: val_acc did not improve from 0.92162
Epoch 5/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.1407 - acc: 0.9510 - val_l

Epoch 00005: val_acc did not improve from 0.92162
Epoch 6/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.1413 - acc: 0.9479 - val_l

Epoch 00006: val_acc improved from 0.92162 to 0.92263, saving model to 30epoch_model.h5
Epoch 7/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.1339 - acc: 0.9538 - val_l

Epoch 00007: val_acc improved from 0.92263 to 0.92297, saving model to 30epoch_model.h5
Epoch 8/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.1452 - acc: 0.9514 - val_l

Epoch 00008: val_acc improved from 0.92297 to 0.92569, saving model to 30epoch_model.h5
Epoch 9/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.1361 - acc: 0.9497 - val_l

Epoch 00009: val_acc did not improve from 0.92569
Epoch 10/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.1127 - acc: 0.9531 - val_l

Epoch 00010: val_acc did not improve from 0.92569
Epoch 11/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.1424 - acc: 0.9502 - val_l

Epoch 00011: val_acc did not improve from 0.92569
Epoch 12/30
7352/7352 [==============================] - 93s 13ms/step - loss: 0.1173 - acc: 0.9524 - val_l

Epoch 00012: val_acc did not improve from 0.92569
Epoch 13/30
7352/7352 [==============================] - 93s 13ms/step - loss: 0.1990 - acc: 0.9376 - val_l
```

```
Epoch 00013: val_acc did not improve from 0.92569
Epoch 14/30
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1220 - acc: 0.9533 - val_l

Epoch 00014: val_acc did not improve from 0.92569
Epoch 15/30
7352/7352 [==============================] - 95s 13ms/step - loss: 0.1411 - acc: 0.9513 - val_l

Epoch 00015: val_acc did not improve from 0.92569
Epoch 16/30
7352/7352 [==============================] - 101s 14ms/step - loss: 0.1622 - acc: 0.9521 - val

Epoch 00016: val_acc did not improve from 0.92569
Epoch 17/30
7352/7352 [==============================] - 93s 13ms/step - loss: 0.1218 - acc: 0.9525 - val_l

Epoch 00017: val_acc did not improve from 0.92569
Epoch 18/30
7352/7352 [==============================] - 95s 13ms/step - loss: 0.2778 - acc: 0.9421 - val_l

Epoch 00018: val_acc did not improve from 0.92569
Epoch 19/30
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1952 - acc: 0.9381 - val_l

Epoch 00019: val_acc did not improve from 0.92569
Epoch 20/30
7352/7352 [==============================] - 92s 13ms/step - loss: 0.1765 - acc: 0.9372 - val_l

Epoch 00020: val_acc did not improve from 0.92569
Epoch 21/30
7352/7352 [==============================] - 92s 13ms/step - loss: 0.1493 - acc: 0.9465 - val_l

Epoch 00021: val_acc did not improve from 0.92569
Epoch 22/30
7352/7352 [==============================] - 92s 13ms/step - loss: 0.1338 - acc: 0.9467 - val_l

Epoch 00022: val_acc did not improve from 0.92569
Epoch 00022: early stopping
Test loss: 0.4867024210958139
Test accuracy: 0.9178825924669155
```

```
In [0]: score = model_1.evaluate(X_test, Y_test, verbose=0)
        print('Test score:', score[0])
        print('Test accuracy:', score[1])

        fig,ax = plt.subplots(1,1)
```

```
        ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

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

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

```
Test score: 0.4867024210958139
Test accuracy: 0.9178825924669155
```




```
In [0]: # Confusion Matrix
        print(confusion_matrix(Y_test, model_1.predict(X_test)))
```

Pred	LAYING	SITTING	...	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
True			...		
LAYING	512	0	...	0	0
SITTING	0	380	...	0	2
STANDING	0	67	...	0	0
WALKING	0	2	...	10	1
WALKING_DOWNSTAIRS	0	0	...	420	0
WALKING_UPSTAIRS	0	4	...	18	448

```
[6 rows x 6 columns]
```

```
In [0]: # Save the model
        model_1.save("30epoch_model.h5")
```

```
In [0]: from numpy import loadtxt
        from keras.models import load_model
        model_2 = load_model('30epoch_model.h5')
```

```
In [0]: model_2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 128, 128)          70656
_________________________________________________________________
dropout_1 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
lstm_2 (LSTM)                (None, 128)               131584
_________________________________________________________________
batch_normalization_1 (Batch (None, 128)               512
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 774
=================================================================
Total params: 203,526
Trainable params: 203,270
Non-trainable params: 256
_________________________________________________________________
```

```
In [0]: # patient early stopping
        from keras.callbacks import EarlyStopping
        es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=20)
```

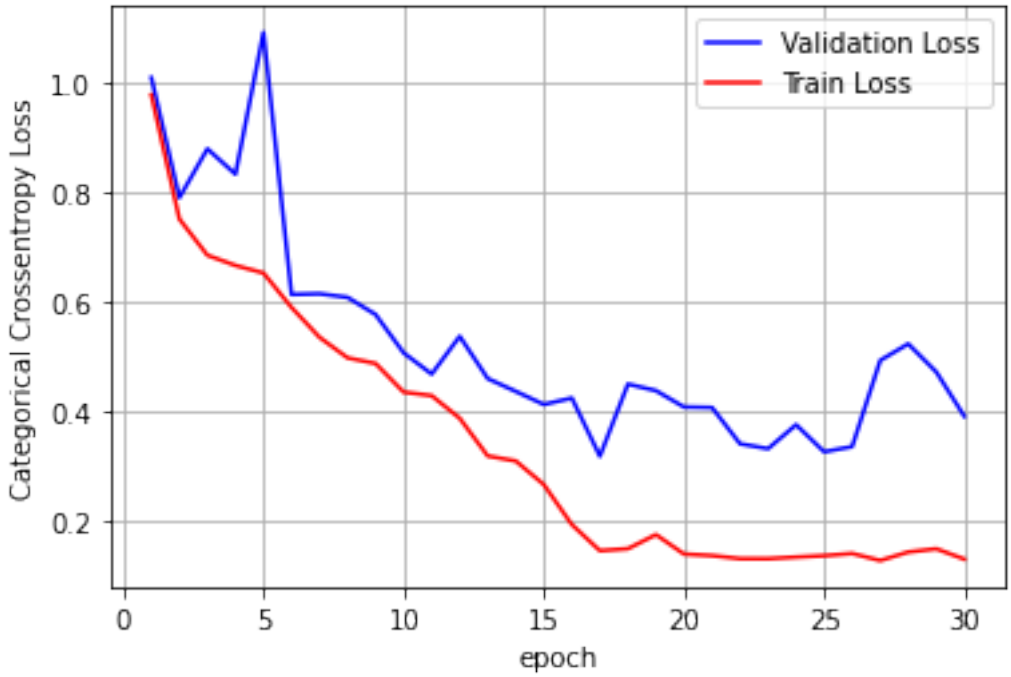
```
In [0]: # ModelCheckpoint
        from keras.callbacks import ModelCheckpoint
        mc = ModelCheckpoint('60epoch_model.h5', monitor='val_acc', mode='max', verbose=1, save
```

```
In [0]: # Training the model
        history4=model_2.fit(X_train,
                  Y_train,
                  batch_size=batch_size,
                  validation_data=(X_test, Y_test),
                  epochs=15, callbacks=[es, mc])
        score = model_2.evaluate(X_test, Y_test, verbose=0)
        print('Test loss:', score[0])
        print('Test accuracy:', score[1])
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/15
7352/7352 [==============================] - 95s 13ms/step - loss: 0.1421 - acc: 0.9518 - val_l

Epoch 00001: val_acc improved from -inf to 0.90804, saving model to 60epoch_model.h5
Epoch 2/15
7352/7352 [==============================] - 93s 13ms/step - loss: 0.1268 - acc: 0.9518 - val_l

Epoch 00002: val_acc improved from 0.90804 to 0.92331, saving model to 60epoch_model.h5
Epoch 3/15
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1192 - acc: 0.9502 - val_l

Epoch 00003: val_acc did not improve from 0.92331
Epoch 4/15
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1290 - acc: 0.9510 - val_l

Epoch 00004: val_acc did not improve from 0.92331
Epoch 5/15
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1259 - acc: 0.9514 - val_l

Epoch 00005: val_acc improved from 0.92331 to 0.92976, saving model to 60epoch_model.h5
Epoch 6/15
7352/7352 [==============================] - 95s 13ms/step - loss: 0.1277 - acc: 0.9546 - val_l

Epoch 00006: val_acc did not improve from 0.92976
Epoch 7/15
7352/7352 [==============================] - 93s 13ms/step - loss: 0.1248 - acc: 0.9493 - val_l

Epoch 00007: val_acc did not improve from 0.92976
Epoch 8/15
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1197 - acc: 0.9540 - val_l

Epoch 00008: val_acc did not improve from 0.92976
Epoch 9/15
7352/7352 [==============================] - 98s 13ms/step - loss: 0.1470 - acc: 0.9527 - val_l

Epoch 00009: val_acc did not improve from 0.92976
Epoch 10/15
7352/7352 [==============================] - 99s 14ms/step - loss: 0.1283 - acc: 0.9554 - val_l

Epoch 00010: val_acc did not improve from 0.92976
Epoch 11/15
7352/7352 [==============================] - 97s 13ms/step - loss: 0.1559 - acc: 0.9509 - val_l

Epoch 00011: val_acc did not improve from 0.92976
Epoch 12/15
7352/7352 [==============================] - 97s 13ms/step - loss: 0.1999 - acc: 0.9518 - val_l
```

```
Epoch 00012: val_acc did not improve from 0.92976
Epoch 13/15
7352/7352 [==============================] - 99s 13ms/step - loss: 0.1229 - acc: 0.9518 - val_l

Epoch 00013: val_acc did not improve from 0.92976
Epoch 14/15
7352/7352 [==============================] - 99s 13ms/step - loss: 0.1284 - acc: 0.9486 - val_l

Epoch 00014: val_acc did not improve from 0.92976
Epoch 15/15
7352/7352 [==============================] - 103s 14ms/step - loss: 0.1468 - acc: 0.9470 - val_

Epoch 00015: val_acc did not improve from 0.92976
Test loss: 0.5388459812360064
Test accuracy: 0.8937902952154734
```

```
In [0]: score = model_2.evaluate(X_test, Y_test, verbose=0)
        print('Test score:', score[0])
        print('Test accuracy:', score[1])

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

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

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

Test score: 0.5388459812360064
Test accuracy: 0.8937902952154734
```




```
In [0]: # Save the model
        model_2.save("60epoch_model.h5")

In [0]: from numpy import loadtxt
        from keras.models import load_model
        model_3 = load_model('60epoch_model.h5')

In [0]: model_3.summary()

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 128, 128)          70656
_________________________________________________________________
dropout_1 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
lstm_2 (LSTM)                (None, 128)               131584
_________________________________________________________________
batch_normalization_1 (Batch (None, 128)               512
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 774
=================================================================
```

```
Total params: 203,526
Trainable params: 203,270
Non-trainable params: 256
_________________________________________________________________
```

```
In [0]: # Compiling the model
        model_3.compile(loss='categorical_crossentropy',
                        optimizer='rmsprop',
                        metrics=['accuracy'])
```

```
In [0]: # patient early stopping
        from keras.callbacks import EarlyStopping
        es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=20)
```

```
In [0]: # ModelCheckpoint
        from keras.callbacks import ModelCheckpoint
        mc = ModelCheckpoint('70epoch_model.h5', monitor='val_acc', mode='max', verbose=1, save
```

```
In [0]: # Training the model
        history4=model_3.fit(X_train,
                  Y_train,
                  batch_size=batch_size,
                  validation_data=(X_test, Y_test),
                  epochs=15, callbacks=[es, mc])
        score = model_3.evaluate(X_test, Y_test, verbose=0)
        print('Test loss:', score[0])
        print('Test accuracy:', score[1])
```

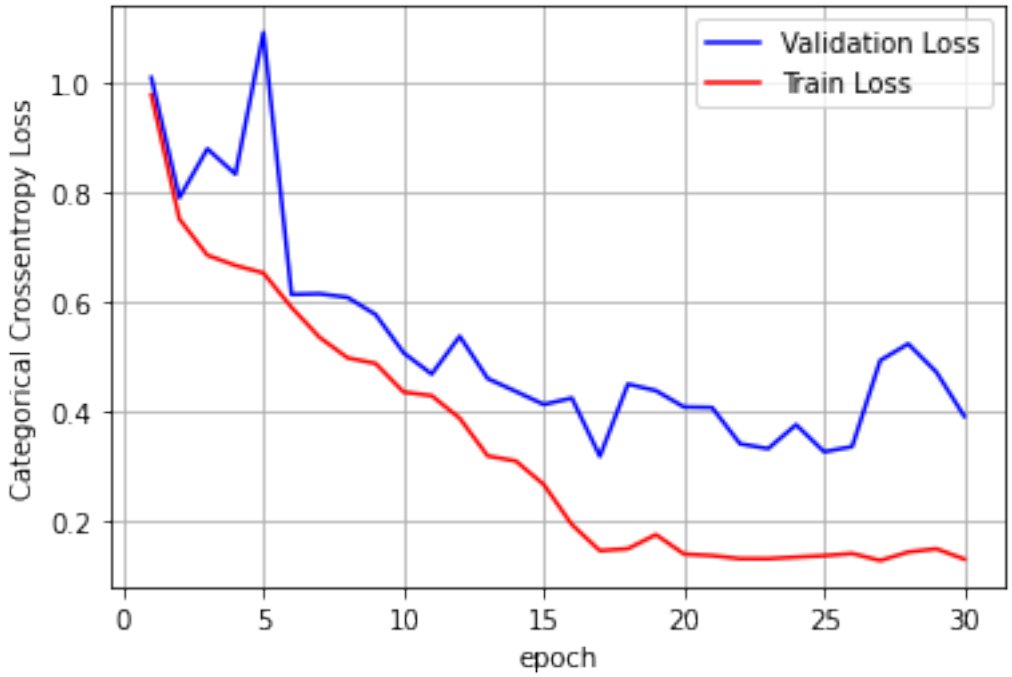
```
Train on 7352 samples, validate on 2947 samples
Epoch 1/15
7352/7352 [==============================] - 98s 13ms/step - loss: 0.1583 - acc: 0.9452 - val_

Epoch 00001: val_acc improved from -inf to 0.92806, saving model to 70epoch_model.h5
Epoch 2/15
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1946 - acc: 0.9484 - val_

Epoch 00002: val_acc did not improve from 0.92806
Epoch 3/15
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1410 - acc: 0.9470 - val_

Epoch 00003: val_acc did not improve from 0.92806
Epoch 4/15
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1260 - acc: 0.9493 - val_

Epoch 00004: val_acc did not improve from 0.92806
Epoch 5/15
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1395 - acc: 0.9501 - val_
```

```
Epoch 00005: val_acc did not improve from 0.92806
Epoch 6/15
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1317 - acc: 0.9506 - val_l

Epoch 00006: val_acc did not improve from 0.92806
Epoch 7/15
7352/7352 [==============================] - 95s 13ms/step - loss: 0.1175 - acc: 0.9553 - val_l

Epoch 00007: val_acc did not improve from 0.92806
Epoch 8/15
7352/7352 [==============================] - 97s 13ms/step - loss: 0.1336 - acc: 0.9525 - val_l

Epoch 00008: val_acc did not improve from 0.92806
Epoch 9/15
7352/7352 [==============================] - 98s 13ms/step - loss: 0.1311 - acc: 0.9483 - val_l

Epoch 00009: val_acc did not improve from 0.92806
Epoch 10/15
7352/7352 [==============================] - 103s 14ms/step - loss: 0.1435 - acc: 0.9509 - val

Epoch 00010: val_acc did not improve from 0.92806
Epoch 11/15
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1466 - acc: 0.9523 - val_l

Epoch 00011: val_acc did not improve from 0.92806
Epoch 12/15
7352/7352 [==============================] - 95s 13ms/step - loss: 0.2185 - acc: 0.9426 - val_l

Epoch 00012: val_acc did not improve from 0.92806
Epoch 13/15
7352/7352 [==============================] - 95s 13ms/step - loss: 0.1757 - acc: 0.9498 - val_l

Epoch 00013: val_acc did not improve from 0.92806
Epoch 14/15
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1457 - acc: 0.9484 - val_l

Epoch 00014: val_acc did not improve from 0.92806
Epoch 15/15
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1295 - acc: 0.9490 - val_l

Epoch 00015: val_acc did not improve from 0.92806
Test loss: 0.4395436131623223
Test accuracy: 0.9127926705123854
```

```
In [0]: score = model_3.evaluate(X_test, Y_test, verbose=0)
        print('Test score:', score[0])
        print('Test accuracy:', score[1])
```

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

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

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

```
Test score: 0.4395436131623223
Test accuracy: 0.9127926705123854
```




```
In [0]: # Save the model
        model_3.save("70epoch_model.h5")
```

```
In [0]: from numpy import loadtxt
        from keras.models import load_model
        model_4 = load_model('70epoch_model.h5')
```

```
In [0]: model_4.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 128, 128)          70656
_________________________________________________________________
dropout_1 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
lstm_2 (LSTM)                (None, 128)               131584
_________________________________________________________________
batch_normalization_1 (Batch (None, 128)               512
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 774
=================================================================
Total params: 203,526
Trainable params: 203,270
Non-trainable params: 256
_________________________________________________________________
```

```
In [0]: # Compiling the model
        model_4.compile(loss='categorical_crossentropy',
                        optimizer='rmsprop',
                        metrics=['accuracy'])
```

```
In [0]: # patient early stopping
        from keras.callbacks import EarlyStopping
        es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=20)
```

```
In [0]: # ModelCheckpoint
        from keras.callbacks import ModelCheckpoint
        mc = ModelCheckpoint('80epoch_model.h5', monitor='val_acc', mode='max', verbose=1, save
```

```
In [0]: # Training the model
        history5=model_4.fit(X_train,
                  Y_train,
                  batch_size=batch_size,
                  validation_data=(X_test, Y_test),
                  epochs=10, callbacks=[es, mc])
        score = model_4.evaluate(X_test, Y_test, verbose=0)
        print('Test loss:', score[0])
        print('Test accuracy:', score[1])
```

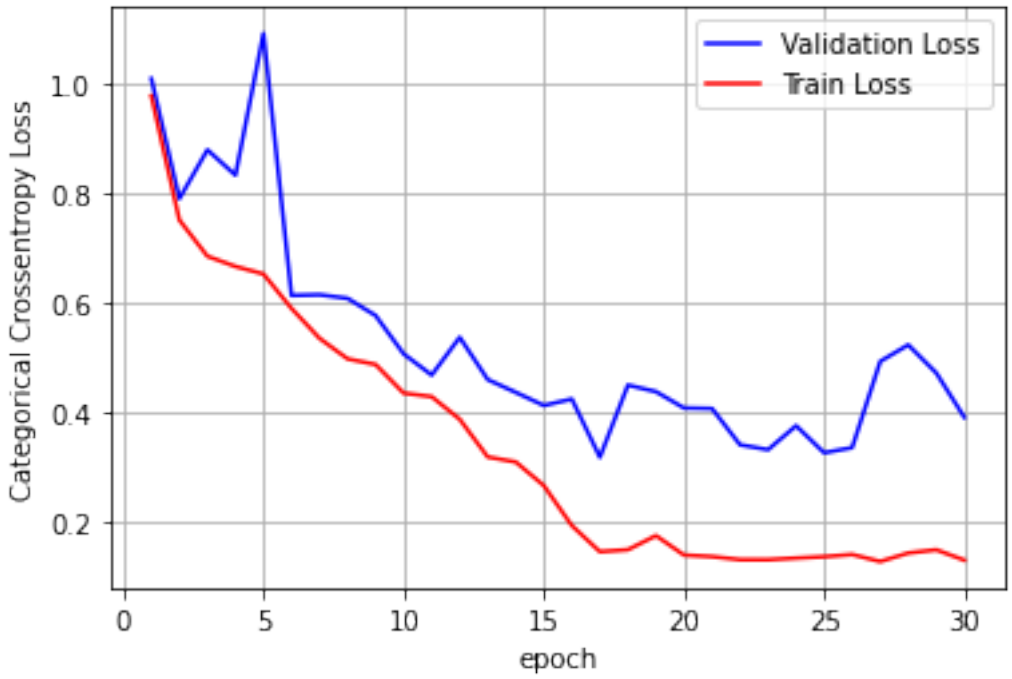
```
Train on 7352 samples, validate on 2947 samples
Epoch 1/10
7352/7352 [==============================] - 100s 14ms/step - loss: 0.1495 - acc: 0.9520 - val

Epoch 00001: val_acc improved from -inf to 0.91211, saving model to 80epoch_model.h5
Epoch 2/10
```

```
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1297 - acc: 0.9513 - val_l

Epoch 00002: val_acc improved from 0.91211 to 0.92094, saving model to 80epoch_model.h5
Epoch 3/10
7352/7352 [==============================] - 95s 13ms/step - loss: 0.1227 - acc: 0.9524 - val_l

Epoch 00003: val_acc did not improve from 0.92094
Epoch 4/10
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1190 - acc: 0.9524 - val_l

Epoch 00004: val_acc did not improve from 0.92094
Epoch 5/10
7352/7352 [==============================] - 93s 13ms/step - loss: 0.1088 - acc: 0.9574 - val_l

Epoch 00005: val_acc did not improve from 0.92094
Epoch 6/10
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1119 - acc: 0.9555 - val_l

Epoch 00006: val_acc did not improve from 0.92094
Epoch 7/10
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1244 - acc: 0.9546 - val_l

Epoch 00007: val_acc did not improve from 0.92094
Epoch 8/10
7352/7352 [==============================] - 95s 13ms/step - loss: 0.1289 - acc: 0.9499 - val_l

Epoch 00008: val_acc did not improve from 0.92094
Epoch 9/10
7352/7352 [==============================] - 95s 13ms/step - loss: 0.1216 - acc: 0.9572 - val_l

Epoch 00009: val_acc did not improve from 0.92094
Epoch 10/10
7352/7352 [==============================] - 95s 13ms/step - loss: 0.1183 - acc: 0.9484 - val_l

Epoch 00010: val_acc did not improve from 0.92094
Test loss: 0.45956417720007264
Test accuracy: 0.9046487953851374
```

```
In [0]: score = model_4.evaluate(X_test, Y_test, verbose=0)
        print('Test score:', score[0])
        print('Test accuracy:', score[1])

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

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

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

```
Test score: 0.45956417720007264
Test accuracy: 0.9046487953851374
```




```
In [0]: # Save the model
        model_4.save("80epoch_model.h5")
```

```
In [0]: from numpy import loadtxt
        from keras.models import load_model
        model_5 = load_model('80epoch_model.h5')
```

```
In [0]: model_5.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 128, 128)          70656
_________________________________________________________________
dropout_1 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
```

```
lstm_2 (LSTM)                (None, 128)               131584
_________________________________________________________________
batch_normalization_1 (Batch (None, 128)               512
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 774
=================================================================
Total params: 203,526
Trainable params: 203,270
Non-trainable params: 256
_________________________________________________________________
```

```
In [0]: # Compiling the model
        model_5.compile(loss='categorical_crossentropy',
                        optimizer='rmsprop',
                        metrics=['accuracy'])
```

```
In [0]: # patient early stopping
        from keras.callbacks import EarlyStopping
        es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=20)
```

```
In [0]: # ModelCheckpoint
        from keras.callbacks import ModelCheckpoint
        mc = ModelCheckpoint('90epoch_model.h5', monitor='val_acc', mode='max', verbose=1, save
```

```
In [0]: # Training the model
        history4=model_5.fit(X_train,
                  Y_train,
                  batch_size=batch_size,
                  validation_data=(X_test, Y_test),
                  epochs=10, callbacks=[es, mc])
        score = model_5.evaluate(X_test, Y_test, verbose=0)
        print('Test loss:', score[0])
        print('Test accuracy:', score[1])
```

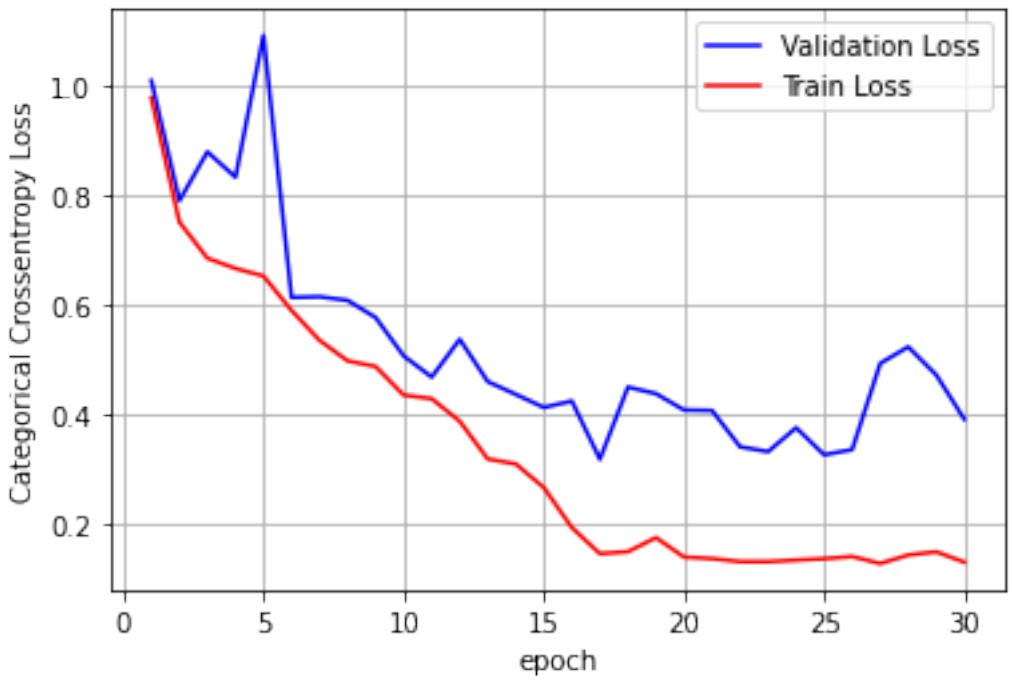
```
Train on 7352 samples, validate on 2947 samples
Epoch 1/10
7352/7352 [==============================] - 103s 14ms/step - loss: 0.1144 - acc: 0.9539 - val_

Epoch 00001: val_acc improved from -inf to 0.91788, saving model to 90epoch_model.h5
Epoch 2/10
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1511 - acc: 0.9543 - val_l

Epoch 00002: val_acc improved from 0.91788 to 0.92433, saving model to 90epoch_model.h5
Epoch 3/10
7352/7352 [==============================] - 97s 13ms/step - loss: 0.2194 - acc: 0.9467 - val_l
```

```
Epoch 00003: val_acc did not improve from 0.92433
Epoch 4/10
7352/7352 [==============================] - 101s 14ms/step - loss: 0.1172 - acc: 0.9567 - val

Epoch 00004: val_acc improved from 0.92433 to 0.92467, saving model to 90epoch_model.h5
Epoch 5/10
7352/7352 [==============================] - 103s 14ms/step - loss: 0.1302 - acc: 0.9486 - val

Epoch 00005: val_acc did not improve from 0.92467
Epoch 6/10
7352/7352 [==============================] - 103s 14ms/step - loss: 0.1995 - acc: 0.9461 - val

Epoch 00006: val_acc improved from 0.92467 to 0.92874, saving model to 90epoch_model.h5
Epoch 7/10
7352/7352 [==============================] - 103s 14ms/step - loss: 0.1234 - acc: 0.9547 - val

Epoch 00007: val_acc did not improve from 0.92874
Epoch 8/10
7352/7352 [==============================] - 101s 14ms/step - loss: 0.1173 - acc: 0.9529 - val

Epoch 00008: val_acc did not improve from 0.92874
Epoch 9/10
7352/7352 [==============================] - 101s 14ms/step - loss: 0.1340 - acc: 0.9542 - val

Epoch 00009: val_acc did not improve from 0.92874
Epoch 10/10
7352/7352 [==============================] - 101s 14ms/step - loss: 0.1387 - acc: 0.9529 - val

Epoch 00010: val_acc improved from 0.92874 to 0.92908, saving model to 90epoch_model.h5
Test loss: 0.2910023735648925
Test accuracy: 0.9290804207668816
```

```
In [0]: score = model_5.evaluate(X_test, Y_test, verbose=0)
        print('Test score:', score[0])
        print('Test accuracy:', score[1])

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

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

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

```
Test score: 0.2910023735648925
```

```
Test accuracy: 0.9290804207668816
```




```
In [0]: # Save the model
        model_5.save("90epoch_model.h5")
```

- With a simple 2 layer architecture with 128 LSTM units we got 93% accuracy and a loss of 0.29.