HAR_LSTM_done

January 6, 2020

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

[0]:
```
%tensorflow_version 1.x
```

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

[4]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id
=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redire
ct_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20http
s%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.c
om%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.reado
nly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
ůůůůůůůůůů
Mounted at /content/drive
```

[0]:
```
dir_path="/content/drive/My Drive/Colab Notebooks/AppliedAI/
 ↪HumanActivityRecognition/HAR/"
```

[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

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

[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"
]
```

[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'{dir_path}UCI_HAR_Dataset/{subset}/Inertial Signals/
 ↪{signal}_{subset}.txt'
        signals_data.append(
            _read_csv(filename).as_matrix()
        )

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

[10]:
```
print(dir_path)
```

```
/content/drive/My Drive/Colab Notebooks/AppliedAI/HumanActivityRecognition/HAR/
```

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

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

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

    return X_train, X_test, y_train, y_test
```

[13]:
```python
# Importing tensorflow
np.random.seed(42)
import tensorflow as tf
from keras import backend as K
tf.set_random_seed(42)
```

```
Using TensorFlow backend.
```

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

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

[0]:
```python
# Importing libraries
from keras.models import Sequential
from keras.layers import LSTM, Conv1D, MaxPooling1D,InputLayer,TimeDistributed,\
                          ConvLSTM2D
```

```
from keras.utils import to_categorical
from keras.optimizers import Adam
from keras.layers.core import Dense, Dropout, Flatten
from keras.layers.normalization import BatchNormalization
from keras.callbacks import ModelCheckpoint
```

[0]:

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

[0]:

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

[40]:

```
# 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_matrix will be removed in a future version. Use .values instead.
  # This is added back by InteractiveShellApp.init_path()
```

[41]:

```
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

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

[42]:

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

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

```
128
9
7352
```

[0]:

```
#container to hold all the parameters and results
d = {
        "layers"    : [],
        "n_hidden"  : [],
        "dropout"  : [],
        "parameters_trained" : [],
        "train_acc" : [],
        "test_acc" : []
    }
```

[0]:
```
def get_trainable_parameters(model):
    trainable_count = int(np.sum([K.count_params(p) for p in set(model.
 ↪trainable_weights)]))
    return trainable_count
```

0.0.2 Defining the Architecture of LSTM

[0]:
```
def define_compile_model(n_hidden=[32],layer=1,dropout=[0.5]):
    assert(len(n_hidden)==layer)
    global timesteps,input_dim,n_classes,d
    ret_seq = True if layer > 1 else False
    d["layers"].append(layer)
    d["n_hidden"].append(n_hidden)
    d["dropout"].append(dropout)
    model = Sequential()
    model.add(LSTM(n_hidden[0],input_shape=(timesteps, input_dim),
                   return_sequences=ret_seq))
    model.add(BatchNormalization())
    model.add(Dropout(dropout[0]))
    for i in range(1,layer):
        if i==layer-1:
            ret_seq = False
        model.add(LSTM(n_hidden[i],return_sequences=ret_seq))
        model.add(BatchNormalization())
        model.add(Dropout(dropout[i]))
    model.add(Dense(n_classes,activation='sigmoid'))
    model.summary()
    # Compiling the model
    model.compile(loss='categorical_crossentropy',optimizer='rmsprop',
                  metrics=['accuracy'])    # also tried adam
    return model
```

[25]:
```
# https://machinelearningmastery.com/
 ↪how-to-develop-rnn-models-for-human-activity-recognition-time-series-classification/
 ↪
# Trying out different custom models

# Model 1 : LSTM

model = Sequential()
```

```
model.add(LSTM(128, input_shape=(timesteps,input_dim),return_sequences=True))
model.add(Dropout(0.5))
model.add(LSTM(128))
model.add(Dropout(0.5))
model.add(Dense(100, activation='relu'))
model.add(Dense(n_classes, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam',
              metrics=['accuracy'])
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:4432: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:148: The name
tf.placeholder_with_default is deprecated. Please use
tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3733: calling dropout (from
tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed
in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 -
keep_prob`.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecated.
Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3576: The name tf.log is
deprecated. Please use tf.math.log instead.
```

[0]:

```
# Model 2: CNN-LSTM

n_steps, n_length = 4, 32
X_train = X_train.reshape((X_train.shape[0], n_steps, n_length, input_dim))
X_test = X_test.reshape((X_test.shape[0], n_steps, n_length, input_dim))
```

[0]:

```
model = Sequential()
model.add(TimeDistributed(Conv1D(filters=64, kernel_size=3,
                 activation='relu'), input_shape=(None,n_length,input_dim)))
model.add(TimeDistributed(Conv1D(filters=32, kernel_size=3,
                  activation='relu')))
model.add(TimeDistributed(Dropout(0.5)))
model.add(TimeDistributed(MaxPooling1D(pool_size=2)))
model.add(TimeDistributed(Flatten()))
model.add(LSTM(100))
model.add(Dropout(0.5))
model.add(Dense(100, activation='relu'))
model.add(Dense(n_classes, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam',
              metrics=['accuracy'])
```

[0]:

```
# Model 3: ConvLSTM2D

timesteps, input_dim, n_classes = X_train.shape[1], X_train.shape[2],\
                                  Y_train.shape[1]
# reshape into subsequences (samples, time steps, rows, cols, channels)
n_steps, n_length = 4, 32
X_train = X_train.reshape((X_train.shape[0], n_steps, 1, n_length, input_dim))
X_test = X_test.reshape((X_test.shape[0], n_steps, 1, n_length, input_dim))
```

[0]:

```
# define model
model = Sequential()
model.add(ConvLSTM2D(filters=64, kernel_size=(1,3),
                     activation='relu',
                     input_shape=(n_steps, 1, n_length, input_dim)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(n_classes, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam',
              metrics=['accuracy'])
```

[46]:

```
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
time_distributed_11 (TimeDis (None, None, 30, 64)      1792
_________________________________________________________________
time_distributed_12 (TimeDis (None, None, 28, 32)      6176
_________________________________________________________________
time_distributed_13 (TimeDis (None, None, 28, 32)      0
_________________________________________________________________
time_distributed_14 (TimeDis (None, None, 14, 32)      0
_________________________________________________________________
time_distributed_15 (TimeDis (None, None, 448)         0
_________________________________________________________________
lstm_5 (LSTM)                (None, 100)               219600
```

```
_________________________________________________________________
dropout_9 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_9 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_10 (Dense)             (None, 6)                 606
=================================================================
Total params: 238,274
Trainable params: 238,274
Non-trainable params: 0
_________________________________________________________________
```

0.0.3 fit and predict model ..

[0]:

```python
def fit_model(X_train,y_train,batch_size,X_test,y_test,epochs,model):
    global d
    # Training the modela
    model.fit(X_train,Y_train,batch_size=batch_size,
              validation_data=(X_test, Y_test),epochs=epochs)
    d["parameters_trained"].append(get_trainable_parameters(model))
    d["train_acc"].append(model.history.history['acc'][-1])
    # Confusion Matrix
    print(confusion_matrix(Y_test, model.predict(X_test)))
    return model

def predict_model(X_test,y_test):
    #evaluate the model :: socre[1] contains the accuracy for the given data ...
    score = model.evaluate(X_test,Y_test)
    d["test_acc"].append(score[1])
    return model
```

0.0.4 LSTM with layer = 1 , n_hidden = 32 and dropout = [0.5]

[0]:

```python
model = define_compile_model(n_hidden=[32,32,32],layer=3,
                             dropout=[0.2,0.2,0.2])
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:4432: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:148: The name
tf.placeholder_with_default is deprecated. Please use
tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3733: calling dropout (from
tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed
in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 -
keep_prob`.
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 128, 32)           5376
_________________________________________________________________
batch_normalization_1 (Batch (None, 128, 32)           128
_________________________________________________________________
dropout_1 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
lstm_2 (LSTM)                (None, 128, 32)           8320
_________________________________________________________________
batch_normalization_2 (Batch (None, 128, 32)           128
_________________________________________________________________
dropout_2 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
lstm_3 (LSTM)                (None, 32)                8320
_________________________________________________________________
batch_normalization_3 (Batch (None, 32)                128
_________________________________________________________________
dropout_3 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 198
=================================================================
Total params: 22,598
Trainable params: 22,406
Non-trainable params: 192
_________________________________________________________________
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecated.
Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3576: The name tf.log is
deprecated. Please use tf.math.log instead.
```

[47]:

```
model = fit_model(X_train,Y_train,batch_size,X_test,Y_test,15,model)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow_core/python/ops/math_grad.py:1424: where (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a future
version.
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.py:1033: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:1020: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.

Train on 7352 samples, validate on 2947 samples
Epoch 1/15
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:190: The name
tf.get_default_session is deprecated. Please use
tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:207: The name tf.global_variables
is deprecated. Please use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:216: The name
tf.is_variable_initialized is deprecated. Please use
tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:223: The name
tf.variables_initializer is deprecated. Please use
tf.compat.v1.variables_initializer instead.

7352/7352 [==============================] - 18s 2ms/step - loss: 0.4835 - acc:
0.8032 - val_loss: 0.5491 - val_acc: 0.8310
Epoch 2/15
7352/7352 [==============================] - 5s 626us/step - loss: 0.1771 - acc:
0.9338 - val_loss: 0.4791 - val_acc: 0.8500
Epoch 3/15
7352/7352 [==============================] - 5s 629us/step - loss: 0.1387 - acc:
0.9441 - val_loss: 0.3689 - val_acc: 0.8945
Epoch 4/15
7352/7352 [==============================] - 5s 643us/step - loss: 0.1294 - acc:
0.9472 - val_loss: 0.6042 - val_acc: 0.8551
Epoch 5/15
7352/7352 [==============================] - 5s 638us/step - loss: 0.1340 - acc:
```

```
0.9506 - val_loss: 0.4288 - val_acc: 0.8948
Epoch 6/15
7352/7352 [==============================] - 5s 633us/step - loss: 0.1163 - acc:
0.9508 - val_loss: 0.4495 - val_acc: 0.8877
Epoch 7/15
7352/7352 [==============================] - 5s 635us/step - loss: 0.1135 - acc:
0.9525 - val_loss: 0.4045 - val_acc: 0.8924
Epoch 8/15
7352/7352 [==============================] - 5s 628us/step - loss: 0.1151 - acc:
0.9514 - val_loss: 0.4384 - val_acc: 0.8928
Epoch 9/15
7352/7352 [==============================] - 5s 640us/step - loss: 0.1090 - acc:
0.9520 - val_loss: 0.4849 - val_acc: 0.8985
Epoch 10/15
7352/7352 [==============================] - 5s 649us/step - loss: 0.0986 - acc:
0.9563 - val_loss: 0.4449 - val_acc: 0.9046
Epoch 11/15
7352/7352 [==============================] - 5s 633us/step - loss: 0.1029 - acc:
0.9570 - val_loss: 0.5560 - val_acc: 0.8768
Epoch 12/15
7352/7352 [==============================] - 5s 640us/step - loss: 0.1087 - acc:
0.9548 - val_loss: 0.3356 - val_acc: 0.9080
Epoch 13/15
7352/7352 [==============================] - 5s 645us/step - loss: 0.0956 - acc:
0.9567 - val_loss: 0.5069 - val_acc: 0.9013
Epoch 14/15
7352/7352 [==============================] - 5s 646us/step - loss: 0.1087 - acc:
0.9548 - val_loss: 0.5025 - val_acc: 0.8873
Epoch 15/15
7352/7352 [==============================] - 5s 643us/step - loss: 0.1041 - acc:
0.9550 - val_loss: 0.4978 - val_acc: 0.9013
Pred                LAYING  SITTING  ...  WALKING_DOWNSTAIRS  WALKING_UPSTAIRS
True                                 ...
LAYING                 510        0  ...                   0                 1
SITTING                  0      412  ...                   0                 7
STANDING                 0       89  ...                   0                 0
WALKING                  0        0  ...                  35                 7
WALKING_DOWNSTAIRS       0        0  ...                 401                 7
WALKING_UPSTAIRS         0        0  ...                  28               436

[6 rows x 6 columns]
```

[49]:

```
model = predict_model(X_test,Y_test)
```

```
2947/2947 [==============================] - 0s 87us/step
```

0.0.5 printing d value

[0]:
```
print(d)
```

```
{'layers': [], 'n_hidden': [], 'dropout': [], 'parameters_trained': [302982,
302982], 'train_acc': [0.9606909684439608, 0.9639553862894451], 'test_acc':
[0.9104173736002714, 0.8730912792670512]}
```

[0]:
```
print(d)
```

```
{'layers': [], 'n_hidden': [], 'dropout': [], 'parameters_trained': [249026,
238274, 238274], 'train_acc': [0.9560663764961915, 0.9575625680087051,
0.9672198041349293], 'test_acc': [0.9121140142517815, 0.9243298269426535,
0.8598574821852731]}
```

- With a simple 2 layer architecture we got 90.09% accuracy and a loss of 0.30
- We can further imporve the performace with Hyperparameter tuning

[0]:
```
##################################
# hyper paratemer tunning
# hidden units , dropout, lstm layers (but be careful that it will not overfit)␣
 ↪(use different dropout rate .. )
# use pretty table to conclude epochs can also be changed ..
# pretty table columns - layers  , n_hidden, dropout used for each layer,␣
 ↪train_Acc, test,Acc
####################################

from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["layers", "n_hidden", "dropout",␣
 ↪"parameters_trained","train_acc","test_acc"]
for i in range(len(d["layers"])):
    x.
 ↪add_row([d["layers"][i],d["n_hidden"][i],d["dropout"][i],d["parameters_trained"][i],d["trai
print(x)
```

```
+--------+----------+-----------------+-------------------+--------------------
+--------------------+
| layers | n_hidden |     dropout     | parameters_trained |     train_acc
|      test_acc      |
+--------+----------+-----------------+-------------------+--------------------
+--------------------+
|   1    |    32    |      [0.5]      |        5574        | 0.9434167573449401
| 0.8842891075670173 |
|   2    |    32    |    [0.5, 0.5]   |       13894        | 0.9473612622415669
| 0.8934509670851714 |
|   3    |    64    | [0.3, 0.3, 0.3] |       85382        | 0.953346028291621З
| 0.9131319986426875 |
+---------+----------+-----------------+-------------------+--------------------
+--------------------+
```

[0]:

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["layers", "n_hidden", "dropout",␣
 ↪"parameters_trained","train_acc","test_acc"]
for i in range(len(d["layers"])):
    x.
 ↪add_row([d["layers"][i],d["n_hidden"][i],d["dropout"][i],d["parameters_trained"][i],d["trai
print(x)
```

```
+---------+------------------+----------------------+-------------------+-------
-------------+--------------------+
| layers |     n_hidden     |       dropout        | parameters_trained |
train_acc       |      test_acc      |
+---------+------------------+----------------------+-------------------+-------
-------------+--------------------+
|   1    |       [32]       |        [0.5]         |        5574        |
0.9434167573449401 | 0.8842891075670173 |
|   1    |       [64]       |        [0.5]         |       19462        |
0.9480413492927094 | 0.8842891075670173 |
|   1    |       [64]       |        [0.3]         |       19462        |
0.9500816104461371 | 0.9267051238547676 |
|   1    |      [128]       |        [0.5]         |       71686        |
0.6228237214363439 | 0.168306752629793  |
|   1    |      [128]       |        [0.5]         |       71686        |
0.9465451577801959 | 0.9280624363759755 |
|   2    |     [32, 32]     |      [0.4, 0.3]      |       14022        |
0.9506256800870512 | 0.9287410926365796 |
|   1    |      [256]       |        [0.5]         |       274438       |
0.9396082698585418 | 0.8992195453003053 |
|   2    |     [64, 32]     |      [0.5, 0.4]      |       31750        |
0.9476332970620239 | 0.9124533423820834 |
|   2    |     [64, 64]     |      [0.5, 0.5]      |       52614        |
0.9426006528835691 | 0.9267051238547676 |
|   3    |   [32, 32, 32]   |   [0.5, 0.4, 0.3]    |       22406        |
0.9472252448313384 | 0.9178825924669155 |
|   3    |   [64, 32, 16]   |   [0.5, 0.3, 0.1]    |       34822        |
0.9430087051142546 | 0.9243298269426535 |
|   4    | [32, 32, 32, 32] | [0.4, 0.4, 0.4, 0.4] |       30790        |
0.9367519042437432 | 0.8948082796063793 |
+---------+------------------+----------------------+-------------------+-------
-------------+--------------------+
```

[0]:

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["layers", "n_hidden", "dropout",␣
 ↪"parameters_trained","train_acc","test_acc"]
for i in range(len(d["layers"])):
    x.
 ↪add_row([d["layers"][i],d["n_hidden"][i],d["dropout"][i],d["parameters_trained"][i],d["trai
print(x)
```

```
+--------+--------------+-----------------+--------------------+----------------
----+--------------------+
| layers |   n_hidden   |     dropout     | parameters_trained |     train_acc
|      test_acc      |
+--------+--------------+-----------------+--------------------+----------------
----+--------------------+
|   3    | [32, 32, 32] | [0.2, 0.2, 0.2] |       22406        |
0.9413764961915125 | 0.8988802171700034 |
+--------+--------------+-----------------+--------------------+----------------
----+--------------------+
```

[0]:

```
table = PrettyTable(['layers','n_hidden','dropout','parameters_trained',
                     'train_acc','test_acc'])
```

[0]:

```
table.add_row([4,[100,100,6],[0.5,0.5],54706,0.9163492927094669,0.
 ↪9066847641669494])
table.add_row([3,[64,100,6],[0.5],249026,0.9560663764961915,0.9121140142517815])
table.add_row([7,[64,32,100,100,6],[0.5,0.5],238274,0.9572198041349293,0.
 ↪9298574821852731])
table.add_row([7,[64,32,100,100,6],[0.5,0.5],238274,0.9675625680087051,0.
 ↪9443298269426535])
```

[0]:

```
print(table)
```

```
+--------+-----------------------+------------+--------------------+------------
--------+--------------------+
| layers |        n_hidden       |  dropout   | parameters_trained |
train_acc      |      test_acc      |
+--------+-----------------------+------------+--------------------+------------
--------+--------------------+
|   4    |     [100, 100, 6]     | [0.5, 0.5] |       54706        |
0.9163492927094669 | 0.9066847641669494 |
|   3    |      [64, 100, 6]     |   [0.5]    |       249026       |
0.9560663764961915 | 0.9121140142517815 |
|   7    | [64, 32, 100, 100, 6] | [0.5, 0.5] |       238274       |
0.9572198041349294 | 0.9298574821852731 |
|   7    | [64, 32, 100, 100, 6] | [0.5, 0.5] |       238274       |
0.9675625680087051 | 0.9443298269426535 |
+--------+----------------------+------------+-------------------+------------
--------+-------------------+
```

[0]:

```
# Conclusions

# We clearly see that the model performs fairly well on shallow LSTM layers
# but as we increase the depth of the model architecture it suffers overfitting

# Also, we experimented by adding large values of dropouts, and reducing the
# number of training epochs but still the deep networks are not gettings any
# better substantially

# The best model having two LSTM layers each having 32 hidden units
```

[0]:

```
# At last we experimented with 2 other kinds of custome models instead of
# vanilla LSTMS's. They were CNN-LSTM and ConvLSTM. We were able to achieve our
# target accuracy of >94% using the CNN-LSTM model after palying around with
# the hyperparams and number of epochs to avoid overfitting
```

[0]:

```

```

[0]:

```
# Trying out a completely new Github implementation
```

[0]:

```
# Using code directly from :
# https://github.com/heeryoncho/sensors2018cnnhar/
```

[0]:

```
from scipy import ndimage
```

[0]:

```
dir_path = dir_path+'UCI_HAR_Dataset/'
```

[11]:

```
print(dir_path)
```

```
/content/drive/My Drive/Colab
Notebooks/AppliedAI/HumanActivityRecognition/HAR/UCI_HAR_Dataset/
```

[0]:

```
def load_x(train_or_test):
    global dir_path
    if train_or_test is "train":
        x_path = dir_path + 'train/X_train.txt'
    elif train_or_test is "test":
        x_path = dir_path + 'test/X_test.txt'

    with open(x_path) as f:
        container = f.readlines()

    result = []
    for line in container:
        tmp1 = line.strip()
        tmp2 = tmp1.replace('  ', ' ')     # removes inconsistent blank spaces
        tmp_ary = list(map(float, tmp2.split(' ')))
        result.append(tmp_ary)
    return np.array(result)
```

[0]:

```
def load_y(train_or_test):
    global dir_path
    if train_or_test is "train":
        y_path = dir_path + 'train/y_train.txt'
    elif train_or_test is "test":
        y_path = dir_path + 'test/y_test.txt'

    with open(y_path) as f:
        container = f.readlines()

    result = []
    for line in container:
        num_str = line.strip()
        result.append(int(num_str))
    return np.array(result)
```

[0]:

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL']='2'
import numpy as np
import random
from numpy.random import seed
from sklearn.metrics import accuracy_score,confusion_matrix
from keras.optimizers import Adam
from keras.models import Sequential, load_model
from keras.layers import Conv1D, MaxPooling1D, Dense, Flatten, Dropout
from keras.callbacks import ModelCheckpoint
from keras.utils import plot_model
import keras.backend as K
```

[0]:

```
X_train_all = load_x("train")
y_train_all = load_y("train")

X_test_all = load_x("test")
y_test_all = load_y("test")
```

[31]:

```
# --------------------------------------
# Only dynamic HAR data are selected
# --------------------------------------

# Select dynamic HAR train data

dynamic_1 = np.where(y_train_all == 1)[0]
dynamic_2 = np.where(y_train_all == 2)[0]
dynamic_3 = np.where(y_train_all == 3)[0]
dynamic = np.concatenate([dynamic_1, dynamic_2, dynamic_3])

X_train = X_train_all[dynamic]
```

```
y_train = y_train_all[dynamic]

# Convert (1, 2, 3) labels to (0, 1, 2)
y_train  = y_train - 1

print("\n+++ DATA STATISTICS +++\n")
print("train_dynamic shape: ", X_train.shape)

# Select dynamic HAR test data

dynamic_1 = np.where(y_test_all == 1)[0]
dynamic_2 = np.where(y_test_all == 2)[0]
dynamic_3 = np.where(y_test_all == 3)[0]
dynamic = np.concatenate([dynamic_1, dynamic_2, dynamic_3])

X_test = X_test_all[dynamic]
y_test = y_test_all[dynamic]

# Convert (1, 2, 3) labels to (0, 1, 2)
y_test  = y_test - 1

print("test_dynamic shape: ", X_test.shape)
```

```
+++ DATA STATISTICS +++

train_dynamic shape:  (3285, 561)
test_dynamic shape:  (1387, 561)
```

[32]:

```
dir_path
```

[32]:

```
'/content/drive/My Drive/Colab
Notebooks/AppliedAI/HumanActivityRecognition/HAR/UCI_HAR_Dataset/'
```

[35]:

```
# Display dynamic model accuracy

print("\n+++ DYNAMIC MODEL ACCURACY (See Table 8 in paper) +++\n")

model_path = dir_path+"models/dynamic.hdf5"
model = load_model(model_path)

pred_train = model.predict(np.expand_dims(X_train, axis=2), batch_size=32)
print("------ TRAIN ACCURACY ------")
print(accuracy_score(y_train, np.argmax(pred_train, axis=1)))
print(confusion_matrix(y_train, np.argmax(pred_train, axis=1)))

pred_test = model.predict(np.expand_dims(X_test, axis=2), batch_size=32)
print("------ TEST ACCURACY ------")
```

```python
print(accuracy_score(y_test, np.argmax(pred_test, axis=1)))
print(confusion_matrix(y_test, np.argmax(pred_test, axis=1)))
```

```
+++ DYNAMIC MODEL ACCURACY (See Table 8 in paper) +++

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:66: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:4432: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:4267: The name tf.nn.max_pool is
deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:148: The name
tf.placeholder_with_default is deprecated. Please use
tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3733: calling dropout (from
tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed
in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 -
keep_prob`.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:190: The name
tf.get_default_session is deprecated. Please use
tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:197: The name tf.ConfigProto is
deprecated. Please use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:203: The name tf.Session is
deprecated. Please use tf.compat.v1.Session instead.
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:207: The name tf.global_variables
is deprecated. Please use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:216: The name
tf.is_variable_initialized is deprecated. Please use
tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:223: The name
tf.variables_initializer is deprecated. Please use
tf.compat.v1.variables_initializer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecated.
Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:1033: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:1020: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.


/usr/local/lib/python3.6/dist-packages/keras/engine/saving.py:350: UserWarning:
Error in loading the saved optimizer state. As a result, your model is starting
with a freshly initialized optimizer.
  warnings.warn('Error in loading the saved optimizer '
```

```
------ TRAIN ACCURACY ------
0.9863013698630136
[[1223    3    0]
 [   4 1038   31]
 [   0    7  979]]
------ TEST ACCURACY ------
0.9798125450612833
[[495   0   1]
 [  2 469   0]
 [  2  23 395]]
```

[38]:

```
static_1 = np.where(y_train_all == 4)[0]
static_2 = np.where(y_train_all == 5)[0]
static_3 = np.where(y_train_all == 6)[0]
static = np.concatenate([static_1, static_2, static_3])
```

```
X_train = X_train_all[static]
y_train = y_train_all[static]

# Convert (4, 5, 6) labels to (0, 1, 2)
y_train  = y_train - 4

print("\n+++ DATA STATISTICS +++\n")
print("train_static shape: ", X_train.shape)

# Select static HAR test data

static_1 = np.where(y_test_all == 4)[0]
static_2 = np.where(y_test_all == 5)[0]
static_3 = np.where(y_test_all == 6)[0]
static = np.concatenate([static_1, static_2, static_3])

X_test = X_test_all[static]
y_test = y_test_all[static]

# Convert (4, 5, 6) labels to (0, 1, 2)
y_test  = y_test - 4

print("test_static shape: ", X_test.shape)


# Display static model accuracy

print("\n+++ STATIC MODEL ACCURACY (See Table 8 in paper) +++\n")

model_path = dir_path+"models/static.hdf5"
model = load_model(model_path)

pred_train = model.predict(np.expand_dims(X_train, axis=2), batch_size=32)
print("------ TRAIN ACCURACY ------")
print(accuracy_score(y_train, np.argmax(pred_train, axis=1)))
print(confusion_matrix(y_train, np.argmax(pred_train, axis=1)))

pred_test = model.predict(np.expand_dims(X_test, axis=2), batch_size=32)
print("------ TEST ACCURACY ------")
print(accuracy_score(y_test, np.argmax(pred_test, axis=1)))
print(confusion_matrix(y_test, np.argmax(pred_test, axis=1)))
```

```
+++ DATA STATISTICS +++

train_static shape:  (4067, 561)
test_static shape:  (1560, 561)
```

```
+++ STATIC MODEL ACCURACY (See Table 8 in paper) +++


/usr/local/lib/python3.6/dist-packages/keras/engine/saving.py:350: UserWarning:
Error in loading the saved optimizer state. As a result, your model is starting
with a freshly initialized optimizer.
  warnings.warn('Error in loading the saved optimizer '

------ TRAIN ACCURACY ------
0.9923776739611507
[[1275   11    0]
 [  20 1354    0]
 [   0    0 1407]]
------ TEST ACCURACY ------
0.966025641025641
[[453  38   0]
 [ 14 518   0]
 [  1   0 536]]
```

[0]: