```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import random
import tensorflow as tf
from keras import layers
import keras
! pip install -q -U keras-tuner
import keras_tuner as kt
import warnings
warnings.filterwarnings('ignore')

base_dir = '/kaggle/input/mhealth/data/mHealth_subject'

df = pd.DataFrame()
for i in range(10):
    df_tmp = pd.read_csv(base_dir + str(i+1) + '.csv', header=0)
    df = pd.concat([df, df_tmp])

# View top 5 rows of dataframe
df.head()
```

[148…

	acceleration from the chest sensor (X axis)	acceleration from the chest sensor (Y axis)	acceleration from the chest sensor (Z axis)	electrocardiogram signal (lead 1)	electrocardiogram signal (lead 2)	acceleration from the left-ankle sensor (X axis)	acceleration from the left-ankle sensor (Y axis)	acceler fron left-a sens
0	-9.8184	0.009971	0.29563	0.004186	0.004186	2.1849	-9.6967	0.6
1	-9.8489	0.524040	0.37348	0.004186	0.016745	2.3876	-9.5080	0.6
2	-9.6602	0.181850	0.43742	0.016745	0.037677	2.4086	-9.5674	0.6
3	-9.6507	0.214220	0.24033	0.079540	0.117220	2.1814	-9.4301	0.5
4	-9.7030	0.303890	0.31156	0.221870	0.205130	2.4173	-9.3889	0.7

5 rows × 24 columns

```
df.Label.value_counts()
```

[150…

```
Label
0.0     872550
1.0      30720
2.0      30720
3.0      30720
4.0      30720
9.0      30720
10.0     30720
11.0     30720
5.0      30720
7.0      29441
8.0      29337
6.0      28315
12.0     10342
Name: count, dtype: int64
```

```
# Remove data with null class (=0)
df = df[df["Label"] != 0]
df = df[df["Label"] != 6]
df = df[df["Label"] != 7]
df = df[df["Label"] != 8]
df = df[df["Label"] != 12]
df.Label.value_counts()
```

```
Label
1.0     30720
2.0     30720
3.0     30720
4.0     30720
9.0     30720
10.0    30720
11.0    30720
5.0     30720
Name: count, dtype: int64
```

```
split_point = int(len(df) * 0.8)
train_data = df.iloc[:split_point, :]
test_data = df.iloc[split_point:, :]

def concat(data):

    # Select right arm data
    rgt_arm = data.iloc[:,15:24]
    rgt_arm.columns=['Ax', 'Ay', 'Az', 'Gx', 'Gy', 'Gz', 'Mx', 'My', 'Mz']

    # Extract labels
    labels = data.iloc[:, -1]
    labels = labels.to_frame()
    labels.columns=['Activity_Label']

    df = rgt_arm

    return df, labels

## Standardize
def standardize(X):
    # Calculate mean and standard deviation along columns (axis=0)
    X_mean = X.mean(axis=0)
    X_std = X.std(axis=0)

    # Standardize each column of X
    standardized_X = (X - X_mean) / X_std

    return standardized_X

# Generate input data and labels
X, y = concat(df)
train_X, train_y = concat(train_data)
test_X, test_y = concat(test_data)
print(type(X))

trainc = train_y
testc = test_y

print(type(train_y))
print(type(trainc))
```

```
# Standardize the right arm data
X_ = standardize(X.copy())  # Avoid modifying original data
train_X_ = standardize(train_X.copy())  # Avoid modifying original data
test_X_ = standardize(test_X.copy())  # Avoid modifying original data

from scipy import stats
from sklearn import metrics

%matplotlib inline

train_X_.head()
```

`159...

	Ax	Ay	Az	Gx	Gy	Gz	Mx	My	Mz
6656	-0.521092	-0.154170	0.173719	-1.060556	-1.544968	0.032539	-0.068058	-0.008686	-1.283513
6657	-0.542376	-0.219229	0.173719	-1.060556	-1.544968	0.042813	-0.050876	-0.000969	-1.283513
6658	-0.540843	-0.212305	0.173719	-1.060556	-1.544968	0.032539	-0.068058	-0.008686	-1.283513
6659	-0.537881	-0.214352	0.139189	-1.060556	-1.536464	0.032590	-0.062369	-0.008705	-1.283513
6660	-0.542242	-0.198806	0.139189	-1.060556	-1.536464	0.022471	-0.062372	-0.012631	-1.283513

```
TIME_STEPS = 23 #sliding window length
STEP = 10 #Sliding window step size
N_FEATURES = 9

#function to create time series datset for seuence modeling
def create_dataset(X, y, time_steps, step):
    Xs, ys = [], []

    for i in range(0, len(X) - time_steps, step):
        x = X.iloc[i:(i + time_steps)].values
        labels = y.iloc[i: i + time_steps]

        mode_result = stats.mode(labels)

        if np.isscalar(mode_result.mode):
            mode_value = mode_result.mode
        elif len(mode_result.mode) > 0:
            mode_value = mode_result.mode[0]
        else:
            mode_value = labels.values[0]

        ys.append(mode_value)
        Xs.append(x)

    return np.array(Xs), np.array(ys).reshape(-1, 1)

train_X_, train_y = create_dataset(train_X_, train_y, 23, step=10)
test_X_, test_y = create_dataset(test_X_, test_y, 23, step=10)
```

```
import tensorflow as tf
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.utils import to_categorical

# Define activity labels as a list
labelss = ['Standing still', 'Sitting and relaxing', 'Lying down', 'Walking',
        'Climbing stairs', 'Cycling', 'Jogging', 'Running']

# Decrease all labels by 1
train_y_remapped = train_y - 1
test_y_remapped = test_y - 1

# Define the mapping dictionary
label_mapping = {
   0: 0,
   1: 1,
   2: 2,
   3: 3,
   4: 4,
   8: 5,  # Map 8 to 5
   9: 6,  # Map 9 to 6
   10: 7
}

# Apply the label remapping using list comprehension
train_y_remapped = np.array([label_mapping.get(label, label) for label in
train_y_remapped.flatten()]).reshape(train_y_remapped.shape)
test_y_remapped = np.array([label_mapping.get(label, label) for label in
test_y_remapped.flatten()]).reshape(test_y_remapped.shape)

# Convert NumPy array train_y_remapped into a DataFrame
train_y_df = pd.DataFrame(train_y_remapped, columns=trainc.columns)

# Convert NumPy array test_y_remapped into a DataFrame
test_y_df = pd.DataFrame(test_y_remapped, columns=testc.columns)



# Define the model
model = keras.Sequential()
model.add(keras.Input(shape=(23, 9)))


model.add(LSTM(512, return_sequences=True, activation='relu'))
model.add(LSTM(256, return_sequences=False, activation='relu'))


# Dense layer for feature extraction
model.add(Dense(128, activation='relu'))


# Output layer with softmax activation for probability distribution
model.add(Dense(len(labelss), activation='softmax'))


print(model.summary())
```

Model: "sequential_9"

Layer (type)	Output Shape	Param #
lstm_18 (LSTM)	(None, 23, 512)	1,069,056
lstm_19 (LSTM)	(None, 256)	787,456
dense_18 (Dense)	(None, 128)	32,896
dense_19 (Dense)	(None, 8)	1,032

Total params: 1,890,440 (7.21 MB)

Trainable params: 1,890,440 (7.21 MB)

Non-trainable params: 0 (0.00 B)

None

```
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model for 15 epochs
model.fit(train_X_, to_categorical(train_y_df, num_classes=len(labelss)), epochs=15)

# Save the model to an H5 file
model.save('jmd.h5')
```

,