In [57]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import tensorflow as tf
from keras import layers
import keras

%matplotlib inline
tf.random.set_seed(42)
```

In [58]:
```
df = pd.read_csv('data.csv')
df
```

Out[58]:

	alx	aly	alz	glx	gly	glz	arx	ary	arz	g
0	2.1849	-9.6967	0.63077	0.103900	-0.84053	-0.68762	-8.6499	-4.5781	0.187760	-0.4490
1	2.3876	-9.5080	0.68389	0.085343	-0.83865	-0.68369	-8.6275	-4.3198	0.023595	-0.4490
2	2.4086	-9.5674	0.68113	0.085343	-0.83865	-0.68369	-8.5055	-4.2772	0.275720	-0.4490
3	2.1814	-9.4301	0.55031	0.085343	-0.83865	-0.68369	-8.6279	-4.3163	0.367520	-0.4568
4	2.4173	-9.3889	0.71098	0.085343	-0.83865	-0.68369	-8.7008	-4.1459	0.407290	-0.4568
...	...	...	...	...	...	...	...	...	...	
1215740	1.7849	-9.8287	0.29725	-0.341370	-0.90056	-0.61493	-3.7198	-8.9071	0.294230	0.0411
1215741	1.8687	-9.8766	0.46236	-0.341370	-0.90056	-0.61493	-3.7160	-8.7455	0.448140	0.0411
1215742	1.6928	-9.9290	0.16631	-0.341370	-0.90056	-0.61493	-3.8824	-9.1155	0.450480	0.0411
1215743	1.5279	-9.6306	0.30458	-0.341370	-0.90056	-0.61493	-3.5564	-9.1441	0.594880	0.0411
1215744	1.6614	-9.8398	0.18088	-0.332100	-0.90432	-0.61886	-3.9035	-8.9324	0.761710	0.0352

1215745 rows × 14 columns

In [59]:

```
from sklearn.utils import resample

df_majority = df[df.Activity==0]
df_minorities = df[df.Activity!=0]

df_majority_downsampled = resample(df_majority,n_samples=30000, random_state
df = pd.concat([df_majority_downsampled, df_minorities])
df.Activity.value_counts()
```

Out[59]:

```
11    30720
10    30720
9     30720
5     30720
4     30720
3     30720
2     30720
1     30720
0     30000
7     29441
8     29337
6     28315
12    10342
Name: Activity, dtype: int64
```

In [60]:

```
#Dropping feature have data outside 98% confidence interval
df1 = df.copy()

for feature in df1.columns[:-2]:
  lower_range = np.quantile(df[feature],0.01)
  upper_range = np.quantile(df[feature],0.99)
  print(feature,'range:',lower_range,'to',upper_range)

  df1 = df1.drop(df1[(df1[feature]>upper_range) | (df1[feature]<lower_range)
  print('shape',df1.shape)
```

```
alx range: -11.473120000000002 to 19.233
shape (365733, 14)
aly range: -19.378999999999998 to 2.4478719999999976
shape (359934, 14)
alz range: -18.95 to 14.19623999999999
shape (356240, 14)
glx range: -0.74212 to 0.80705
shape (349347, 14)
gly range: -1.0694 to 0.96623
shape (342811, 14)
glz range: -1.1061 to 0.8290799999999999
shape (337361, 14)
arx range: -21.492 to 9.097647999999998
shape (332280, 14)
ary range: -18.694000000000006 to 11.948059999999998
shape (326215, 14)
arz range: -10.367 to 11.823119999999996
shape (323650, 14)
grx range: -1.0196 to 0.95686
shape (320165, 14)
gry range: -1.1417 to 0.90965
shape (315329, 14)
grz range: -0.69828 to 1.125
shape (310906, 14)
```

In [61]:

```
label_map = {
    0: 'Nothing',
    1: 'Standing still',
    2: 'Sitting and relaxing',
    3: 'Lying down',
    4: 'Walking',
    5: 'Climbing stairs',
    6: 'Waist bends forward',
    7: 'Frontal elevation of arms',
    8: 'Knees bending (crouching)',
    9: 'Cycling',
    10: 'Jogging',
    11: 'Running',
    12: 'Jump front & back'
}
```

In [62]:
```
train = df1[(df1['subject'] != 'subject10') & (df1['subject'] != 'subject9'
test = df1.drop(train.index, axis=0)
train.shape,test.shape
```

Out[62]:
```
((246483, 14), (64423, 14))
```

In [63]:
```
X_train = train.drop(['Activity','subject'],axis=1)
y_train = train['Activity']
X_test = test.drop(['Activity','subject'],axis=1)
y_test = test['Activity']
X_train.shape,y_train.shape,X_test.shape,y_test.shape
```

Out[63]:
```
((246483, 12), (246483,), (64423, 12), (64423,))
```

In [64]:
```
from scipy import stats

#function to create time series datset for seuence modeling
def create_dataset(X, y, time_steps, step=1):
    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]
        Xs.append(x)
        ys.append(stats.mode(labels)[0][0])
    return np.array(Xs), np.array(ys).reshape(-1, 1)
```

In [65]:
```
X_train,y_train = create_dataset(X_train, y_train, 100, step=50)
X_train.shape, y_train.shape
```

Out[65]:
```
((4928, 100, 12), (4928, 1))
```

In [66]:
```
X_test,y_test = create_dataset(X_test, y_test, 100, step=50)
X_test.shape, y_test.shape
```

Out[66]:
```
((1287, 100, 12), (1287, 1))
```

In [67]:

```
model = keras.Sequential()
model.add(layers.Input(shape=[100,12]))
model.add(layers.Conv1D(filters=32, kernel_size=3, padding="same"))
model.add(layers.BatchNormalization())
model.add(layers.ReLU())
model.add(layers.Conv1D(filters=64, kernel_size=3, padding="same"))
model.add(layers.BatchNormalization())
model.add(layers.ReLU())
model.add(layers.MaxPool1D(2))
model.add(layers.LSTM(64))
model.add(layers.Dense(units=128, activation='relu'))
model.add(layers.Dense(13, activation='softmax'))
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_6 (Conv1D)            (None, 100, 32)           1184
_________________________________________________________________
batch_normalization_6 (Batch (None, 100, 32)           128
_________________________________________________________________
re_lu_6 (ReLU)               (None, 100, 32)           0
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 100, 64)           6208
_________________________________________________________________
batch_normalization_7 (Batch (None, 100, 64)           256
_________________________________________________________________
re_lu_7 (ReLU)               (None, 100, 64)           0
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 50, 64)            0
_________________________________________________________________
lstm_3 (LSTM)                (None, 64)                33024
_________________________________________________________________
dense_6 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_7 (Dense)              (None, 13)                1677
=================================================================
Total params: 50,797
Trainable params: 50,605
Non-trainable params: 192
_________________________________________________________________
```

In [68]:

```
tf.keras.utils.plot_model(model, show_shapes=True)
```

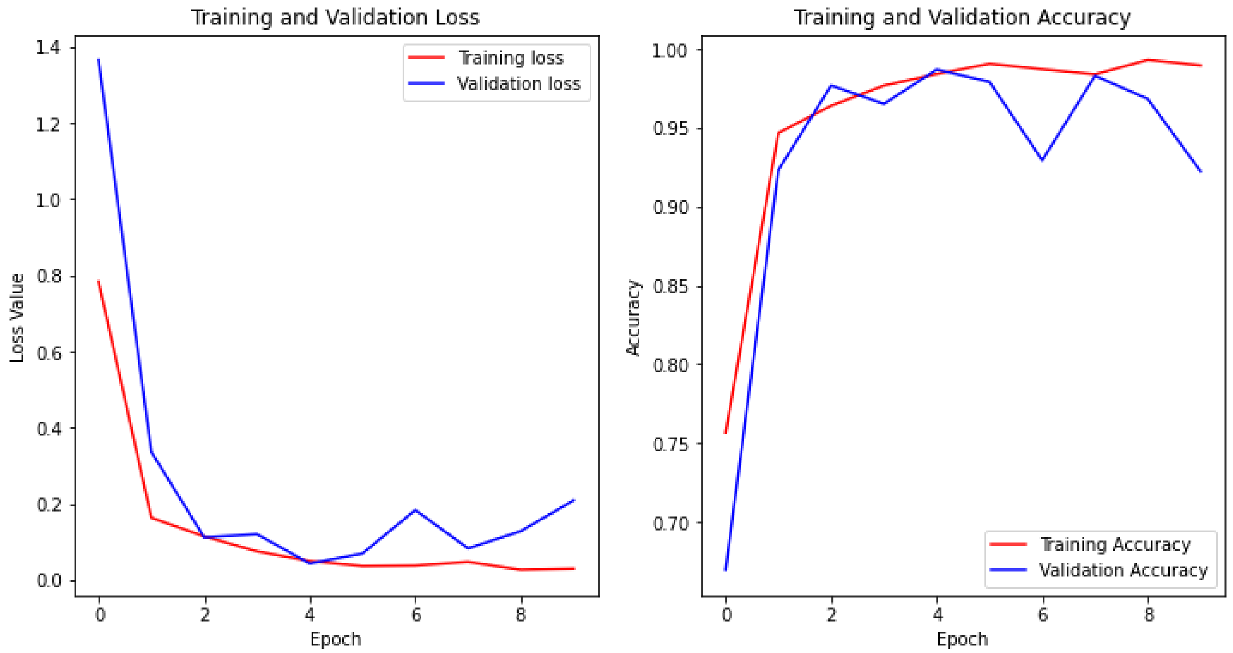
```
('You must install pydot (`pip install pydot`) and install graphviz (see
instructions at https://graphviz.gitlab.io/download/) (https://graphviz.g
itlab.io/download/)) ', 'for plot_model/model_to_dot to work.')
```

In [69]:
```
callbacks = [keras.callbacks.ModelCheckpoint("model.h5", save_best_only=True
             keras.callbacks.EarlyStopping(monitor="val_loss", patience=50,

model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metr

model_history = model.fit(X_train,y_train, epochs= 10, validation_data=(X_te
```

```
Epoch 1/10
154/154 [==============================] - 5s 21ms/step - loss: 0.7834 -
sparse_categorical_accuracy: 0.7567 - val_loss: 1.3659 - val_sparse_categ
orical_accuracy: 0.6698
Epoch 2/10
154/154 [==============================] - 3s 19ms/step - loss: 0.1632 -
sparse_categorical_accuracy: 0.9466 - val_loss: 0.3360 - val_sparse_categ
orical_accuracy: 0.9231
Epoch 3/10
154/154 [==============================] - 3s 18ms/step - loss: 0.1144 -
sparse_categorical_accuracy: 0.9639 - val_loss: 0.1118 - val_sparse_categ
orical_accuracy: 0.9767
Epoch 4/10
154/154 [==============================] - 3s 18ms/step - loss: 0.0751 -
sparse_categorical_accuracy: 0.9767 - val_loss: 0.1201 - val_sparse_categ
orical_accuracy: 0.9650
Epoch 5/10
154/154 [==============================] - 3s 17ms/step - loss: 0.0496 -
sparse_categorical_accuracy: 0.9840 - val_loss: 0.0431 - val_sparse_categ
orical_accuracy: 0.9868
Epoch 6/10
154/154 [==============================] - 3s 19ms/step - loss: 0.0365 -
sparse_categorical_accuracy: 0.9905 - val_loss: 0.0693 - val_sparse_categ
orical_accuracy: 0.9790
Epoch 7/10
154/154 [==============================] - 3s 22ms/step - loss: 0.0377 -
sparse_categorical_accuracy: 0.9870 - val_loss: 0.1835 - val_sparse_categ
orical_accuracy: 0.9293
Epoch 8/10
154/154 [==============================] - 3s 20ms/step - loss: 0.0474 -
sparse_categorical_accuracy: 0.9838 - val_loss: 0.0828 - val_sparse_categ
orical_accuracy: 0.9829
Epoch 9/10
154/154 [==============================] - 3s 19ms/step - loss: 0.0265 -
sparse_categorical_accuracy: 0.9929 - val_loss: 0.1280 - val_sparse_categ
orical_accuracy: 0.9681
Epoch 10/10
154/154 [==============================] - 3s 20ms/step - loss: 0.0295 -
sparse_categorical_accuracy: 0.9894 - val_loss: 0.2090 - val_sparse_categ
orical_accuracy: 0.9223
```

In [70]:

```
train_loss = model_history.history['loss']
val_loss = model_history.history['val_loss']
train_accuracy = model_history.history['sparse_categorical_accuracy']
val_accuracy = model_history.history['val_sparse_categorical_accuracy']

plt.figure(figsize=(12,6))

plt.subplot(1,2,1)
plt.plot(train_loss, 'r', label='Training loss')
plt.plot(val_loss, 'b', label='Validation loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss Value')
plt.legend()

plt.subplot(1,2,2)
plt.plot(train_accuracy, 'r', label='Training Accuracy')
plt.plot(val_accuracy, 'b', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```



In [71]:

```
model = keras.models.load_model('./model.h5')

train_loss, train_acc = model.evaluate(X_train,y_train)
test_loss, test_acc = model.evaluate(X_test,y_test)

print("Train accuracy", round(train_acc*100, 2),'%')
print("Train loss", train_loss)
print("Test accuracy", round(test_acc*100, 2),'%')
print("Test loss", test_loss)
```

```
154/154 [==============================] - 1s 7ms/step - loss: 0.0141 - s
parse_categorical_accuracy: 0.9976
41/41 [==============================] - 1s 7ms/step - loss: 0.0431 - spa
rse_categorical_accuracy: 0.9868
Train accuracy 99.76 %
Train loss 0.014134098775684834
Test accuracy 98.68 %
Test loss 0.043090686202049255
```

In [72]:

```
pred = model.predict(X_test)
pred = np.argmax(pred, axis = 1)
pred = pred.reshape(-1,1)
```

In [73]:

```
pred.shape,y_test.shape
```

Out[73]:

```
((1287, 1), (1287, 1))
```

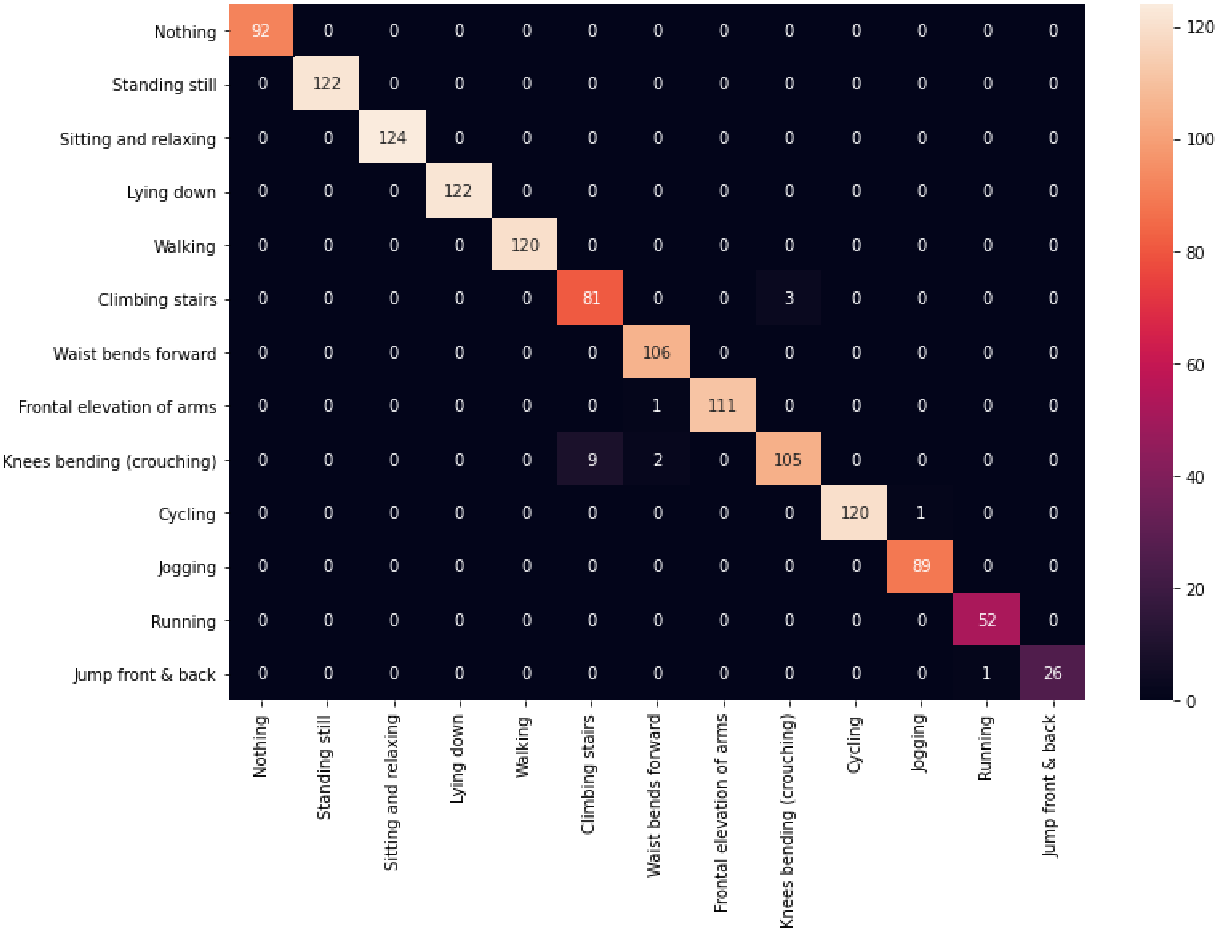
In [74]:

```
from sklearn.metrics import confusion_matrix, classification_report

print(classification_report(y_test,pred))
print('*'*50)
print(confusion_matrix(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        92
           1       1.00      1.00      1.00       122
           2       1.00      1.00      1.00       124
           3       1.00      1.00      1.00       122
           4       1.00      1.00      1.00       120
           5       0.90      0.96      0.93        84
           6       0.97      1.00      0.99       106
           7       1.00      0.99      1.00       112
           8       0.97      0.91      0.94       116
           9       1.00      0.99      1.00       121
          10       0.99      1.00      0.99        89
          11       0.98      1.00      0.99        52
          12       1.00      0.96      0.98        27

    accuracy                           0.99      1287
   macro avg       0.99      0.99      0.99      1287
weighted avg       0.99      0.99      0.99      1287

**************************************************
[[ 92   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0 122   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0 124   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0 122   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 120   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0  81   0   0   3   0   0   0   0]
 [  0   0   0   0   0   0 106   0   0   0   0   0   0]
 [  0   0   0   0   0   0   1 111   0   0   0   0   0]
 [  0   0   0   0   0   9   2   0 105   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0 120   1   0   0]
 [  0   0   0   0   0   0   0   0   0   0  89   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  52   0]
 [  0   0   0   0   0   0   0   0   0   0   0   1  26]]
```

In [75]:

```
plt.figure(figsize=(12,8))
conf_matrix = confusion_matrix(y_test,pred)
sns.heatmap(conf_matrix, xticklabels= label_map.values(), yticklabels= label
plt.show()
```



	Nothing	Standing still	Sitting and relaxing	Lying down	Walking	Climbing stairs	Waist bends forward	Frontal elevation of arms	Knees bending (crouching)	Cycling	Jogging	Running	Jump front & back
Nothing	92	0	0	0	0	0	0	0	0	0	0	0	0
Standing still	0	122	0	0	0	0	0	0	0	0	0	0	0
Sitting and relaxing	0	0	124	0	0	0	0	0	0	0	0	0	0
Lying down	0	0	0	122	0	0	0	0	0	0	0	0	0
Walking	0	0	0	0	120	0	0	0	0	0	0	0	0
Climbing stairs	0	0	0	0	0	81	0	0	3	0	0	0	0
Waist bends forward	0	0	0	0	0	0	106	0	0	0	0	0	0
Frontal elevation of arms	0	0	0	0	0	0	1	111	0	0	0	0	0
Knees bending (crouching)	0	0	0	0	0	9	2	0	105	0	0	0	0
Cycling	0	0	0	0	0	0	0	0	0	120	1	0	0
Jogging	0	0	0	0	0	0	0	0	0	0	89	0	0
Running	0	0	0	0	0	0	0	0	0	0	0	52	0
Jump front & back	0	0	0	0	0	0	0	0	0	0	0	1	26

In [ ]: