In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from Data_manipulation_utils import *
from model_functions import *
import scipy
```

Loading Data

In [2]:

```
X = np.loadtxt('x.csv', delimiter = ',')
Y = np.loadtxt('y.csv', delimiter = ',')
```

In [3]:

```
print("Shape of raw data")
print("Shape of X : ", X.shape)
print("Shape of Y : ", Y.shape)
```

```
Shape of raw data
Shape of X :  (814400, 6)
Shape of Y :  (814400,)
```

Generation Dataset to Train from raw data

In [4]:

```
x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12 = split_data_labelwise(X, Y)
```

Spectrogram readings of Dataset

In [6]:

```
s = 15
l = 15
```

In [11]:

```
s_x1, y1 = feature_extraction_from_rawdata(x1, label = 1, S = s, L = l)
s_x2, y2 = feature_extraction_from_rawdata(x2, label = 2, S = s, L = l)
s_x3, y3 = feature_extraction_from_rawdata(x3, label = 3, S = s, L = l)
s_x4, y4 = feature_extraction_from_rawdata(x4, label = 4, S = s, L = l)
s_x5, y5 = feature_extraction_from_rawdata(x5, label = 5, S = s, L = l)
s_x6, y6 = feature_extraction_from_rawdata(x6, label = 6, S = s, L = l)
s_x7, y7 = feature_extraction_from_rawdata(x7, label = 7, S = s, L = l)
s_x8, y8 = feature_extraction_from_rawdata(x8, label = 8, S = s, L = l)
s_x9, y9 = feature_extraction_from_rawdata(x9, label = 9, S = s, L = l)
s_x10, y10 = feature_extraction_from_rawdata(x10, label = 10, S = s, L = l)
s_x11, y11 = feature_extraction_from_rawdata(x11, label = 11, S = s, L = l)
s_x12, y12 = feature_extraction_from_rawdata(x12, label = 12, S = s, L = l)
```

Time Domain Features

In [5]:

```
tx1 = time_domain_features_from_rawdata(x1)
tx2 = time_domain_features_from_rawdata(x2)
tx3 = time_domain_features_from_rawdata(x3)
tx4 = time_domain_features_from_rawdata(x4)
tx5 = time_domain_features_from_rawdata(x5)
tx6 = time_domain_features_from_rawdata(x6)
tx7 = time_domain_features_from_rawdata(x7)
tx8 = time_domain_features_from_rawdata(x8)
tx9 = time_domain_features_from_rawdata(x9)
tx10 = time_domain_features_from_rawdata(x10)
tx11 = time_domain_features_from_rawdata(x11)
tx12 = time_domain_features_from_rawdata(x12)
```

In [12]:

```
Xf = np.hstack((s_x1, s_x2, s_x3, s_x4, s_x5, s_x6, s_x7, s_x8, s_x9, s_x10, s_x11, s_x12))
Xt = np.hstack((tx1, tx2, tx3, tx4, tx5, tx6, tx7, tx8, tx9, tx10, tx11, tx12))
Xtf = np.vstack((Xf, Xt))

Ytf = np.hstack((y1, y2, y3, y4, y5, y6, y7, y8, y9, y10, y11, y12))
```

In [13]:

```
print("Shape of Xtf :", Xtf.shape)
print("Shape of Ytf :", Ytf.shape)
```

```
Shape of Xtf : (460, 3166)
Shape of Ytf : (1, 3166)
```

Frequency Domain Dataset with Data Augmentation

In [14]:

```
s = 100
l = 100
```

In [15]:

```
Sx1, y1 = feature_extraction_from_rawdata(x1, label = 1, S = s, L = l)
Sx2, y2 = feature_extraction_from_rawdata(x2, label = 2, S = s, L = l)
Sx3, y3 = feature_extraction_from_rawdata(x3, label = 3, S = s, L = l)
Sx4, y4 = feature_extraction_from_rawdata(x4, label = 4, S = s, L = l)
Sx5, y5 = feature_extraction_from_rawdata(x5, label = 5, S = s, L = l)
Sx6, y6 = feature_extraction_from_rawdata(x6, label = 6, S = s, L = l)
Sx7, y7 = feature_extraction_from_rawdata(x7, label = 7, S = s, L = l)
Sx8, y8 = feature_extraction_from_rawdata(x8, label = 8, S = s, L = l)
Sx9, y9 = feature_extraction_from_rawdata(x9, label = 9, S = s, L = l)
Sx10, y10 = feature_extraction_from_rawdata(x10, label = 10, S = s, L = l)
Sx11, y11 = feature_extraction_from_rawdata(x11, label = 11, S = s, L = l)
Sx12, y12 = feature_extraction_from_rawdata(x12, label = 12, S = s, L = l)
```

Data Augmentation

Local Averaging with window_size = 4

In [17]:

```
Sx1, y1 = local_averaging(Sx1, 4, 1, shuffle = True)
Sx2, y2 = local_averaging(Sx2, 4, 2, shuffle = True)
Sx3, y3 = local_averaging(Sx3, 4, 3, shuffle = True)
Sx4, y4 = local_averaging(Sx4, 4, 4, shuffle = True)
Sx5, y5 = local_averaging(Sx5, 4, 5, shuffle = True)
Sx6, y6 = local_averaging(Sx6, 4, 6, shuffle = True)
Sx7, y7 = local_averaging(Sx7, 4, 7, shuffle = True)
Sx8, y8 = local_averaging(Sx8, 4, 8, shuffle = True)
Sx9, y9 = local_averaging(Sx9, 4, 9, shuffle = True)
Sx10, y10 = local_averaging(Sx10, 4, 10, shuffle = True)
Sx11, y11 = local_averaging(Sx11, 4, 11, shuffle = True)
Sx12, y12 = local_averaging(Sx12, 4, 12, shuffle = True)
```

Local Averaging with window_size = 2

In [18]:

```
Sx1, y1 = local_averaging(Sx1, 2, 1, shuffle = False)
Sx2, y2 = local_averaging(Sx2, 2, 2, shuffle = False)
Sx3, y3 = local_averaging(Sx3, 2, 3, shuffle = False)
Sx4, y4 = local_averaging(Sx4, 2, 4, shuffle = False)
Sx5, y5 = local_averaging(Sx5, 2, 5, shuffle = False)
Sx6, y6 = local_averaging(Sx6, 2, 6, shuffle = False)
Sx7, y7 = local_averaging(Sx7, 2, 7, shuffle = False)
Sx8, y8 = local_averaging(Sx8, 2, 8, shuffle = False)
Sx9, y9 = local_averaging(Sx9, 2, 9, shuffle = False)
Sx10, y10 = local_averaging(Sx10, 2, 10, shuffle = False)
Sx11, y11 = local_averaging(Sx11, 2, 11, shuffle = False)
Sx12, y12 = local_averaging(Sx12, 2, 12, shuffle = False)
```

Final Dataset to Train

In [19]:

```
X_fa = np.hstack((Sx1, Sx2, Sx3, Sx4, Sx5, Sx6, Sx7, Sx8, Sx9, Sx10, Sx11, Sx12))
Y_fa = np.hstack((y1, y2, y3, y4, y5, y6, y7, y8, y9, y10, y11, y12))
```

In [21]:

```
print("Shape of X_train : ", X_fa.shape)
print("Shape of Y_train : ", Y_fa.shape)
```

```
Shape of X_train :  (400, 12766)
Shape of Y_train :  (1, 12766)
```

Training Model with only frequency domain features with Data Augmentation

In [22]:

```
# Fitting SVM to the Training set
X_fa = X_fa.T
Y_fa = Y_fa.T

from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_fa, Y_fa)

y_pred = classifier.predict(X_fa)

from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(Y_fa, y_pred)
print(cm)

acc = accuracy_score(Y_fa,y_pred)
print("Training Model with only frequency domain features with Data Augmentation :")
print("Accuracy of model :", acc*100, "%")
```

```
c:\users\jaimi\appdata\local\programs\python\python36\lib\site-packages\skle
arn\utils\validation.py:578: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_sample
s, ), for example using ravel().
  y = column_or_1d(y, warn=True)

[[1163  103    9    0    0    0    0    0    0    0    0    0]
 [ 193 1013   48    0    0    0    0    0    0    0    0    0]
 [  39   78 1101    0    0    1    0    0    0    0    0    0]
 [   0    0    0 1011   41  241    0    0    0    0    0    0]
 [   0    0    0  503  685  150    0    0    0    0    0    0]
 [   0    0    0  710   44  579    0    0    0    0    0    0]
 [   2    3    0    0    0    0  834    0    0    0    0    0]
 [   1    2    0    0    0    0    0  827    0    0    0    0]
 [   0    0    0    0    0    0    0    0  846    0    1    0]
 [   0    0    0    0    0    0    0    0    2  839    0    1]
 [   0    0    0    0    0    0    0    0    2    1  847    5]
 [   0    0    0    0    0    0    0    0    1    1    1  838]]
Accuracy of model : 82.89989033369889 %
```

Training Dataset with 60 frequency + 60 time features :

In [23]:

```
# Fitting SVM to the Training set

Xtf = Xtf.T
Ytf = Ytf.T

from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(Xtf, Ytf)

y_pred = classifier.predict(Xtf)

from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(Ytf, y_pred)
print(cm)

acc = accuracy_score(Ytf,y_pred)
print("Training Dataset with 60 frequency + 60 time features :")
print("Accuracy of model :", acc*100, "%")
```

```
c:\users\jaimi\appdata\local\programs\python\python36\lib\site-packages\skle
arn\utils\validation.py:578: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_sample
s, ), for example using ravel().
  y = column_or_1d(y, warn=True)

[[474   1   0   0   0   0   0   0   0   0   0   0]
 [  2 452   0   0   0   0   0   0   0   0   0   0]
 [  0   0 419   0   0   0   0   0   0   0   0   0]
 [  0   0   0 469  24   0   0   0   0   0   0   0]
 [  0   0   0  22 516   0   0   0   0   0   0   0]
 [  0   0   0   0   0 533   0   0   0   0   0   0]
 [  0   0   0   0   0   0  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  30   0   0   0   0]
 [  0   0   0   0   0   0   0   0  47   0   0   0]
 [  0   0   0   0   0   0   0   0   0  42   0   0]
 [  0   0   0   0   0   0   0   0   0   0  55   0]
 [  0   0   0   0   0   0   0   0   0   0   0  41]]
Training Dataset with 60 frequency + 60 time features :
Accuracy of model : 98.45230574857865 %
```

In [ ]:

```

```