Window size:1024
Stride: 1024
Raw data with detrend(lambda:300)
Lowpass filter(300Hz)
Use all channels
Use rectified signal. Shuffle and split data to 25% for test and 75% for training.

```
model = keras.models.Sequential()
model.add(layers.InputLayer(input_shape=X[:,:,:].shape[1:]))
model.add(layers.Bidirectional(layers.LSTM(32,return_sequences=True,
                                           recurrent_regularizer=reg)))
model.add(layers.Conv1D(filters=32, kernel_size=kernel_size,strides=st,
                        padding='same',
                        kernel_regularizer=reg,
                        kernel_initializer=kernel_initializer
                       ))
model.add(layers.LayerNormalization(axis=axis))
model.add(layers.ELU())
model.add(layers.AveragePooling1D(2))
model.add(layers.Dropout(drop_rate))
model.add(layers.Conv1D(filters=16, kernel_size=kernel_size,strides=st,
                        padding='same',
                        kernel_regularizer=reg,
                        kernel_initializer=kernel_initializer
                       ))
model.add(layers.LayerNormalization(axis=axis))
model.add(layers.ELU())
model.add(layers.AveragePooling1D(2))
model.add(layers.Dropout(drop_rate))
model.add(layers.Conv1D(filters=8, kernel_size=kernel_size,strides=st,
                        padding='same',
                        kernel_regularizer=reg,
                        kernel_initializer=kernel_initializer
                       ))
model.add(layers.LayerNormalization(axis=axis))
model.add(layers.ELU())
model.add(layers.AveragePooling1D(2))
model.add(layers.Dropout(drop_rate))
model.add(layers.Conv1D(filters=4, kernel_size=kernel_size,strides=st,
                        padding='same',
                        kernel_regularizer=reg,
                        kernel_initializer=kernel_initializer
                       ))
model.add(layers.LayerNormalization(axis=axis))
```

```
model.add(layers.ELU())
model.add(layers.GlobalAveragePooling1D())
model.add(layers.Dropout(drop_rate))
model.add(layers.Dense(3,activation='softmax',kernel_regularizer=reg))
```

file	label 1
G07_Freezing_Trial1_trial_1_emg.csv	7
G08_FoG_1_trial_1_emg.csv	27
G08_FoG_2_trial_1_emg.csv	70
P551_M050_2_B_FoG_trial_1_emg.csv	6
P812_M050_2_B_FoG_trial_1_emg.csv	1

```
kernel_size=11
reg=regularizers.l2(1e-4)
drop_rate = 0.2
kernel_initializer = 'glorot_normal'
mo = 0.8
st = 1
axis = 2
```

drop G07_Freezing_Trial1_trial_1_emg.csv and P551_M050_2_B_FoG_trial_1_emg.csv

$$\text{cost\_matrix} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Train (acc 0.970)

	Predicted 1	Predicted 2	Predicted 6	acc
Actual 1	72	2	1	0.960
Actual 2	2	183	1	0.983
Actual 6	5	0	109	0.956

Test (acc 0.912)

	Predicted 1	Predicted 2	Predicted 6	acc
Actual 1	17	2	4	0.739
Actual 2	0	59	0	0.949
Actual 6	3	2	39	0.886

cost_matrix = $\begin{bmatrix} 0 & 1.2 & 1.3 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$

Train (acc 0.976)

	Predicted 1	Predicted 2	Predicted 6	acc
Actual 1	73	0	2	0.973
Actual 2	4	182	0	0.978
Actual 6	3	0	111	0.973

Test (acc 0.936)

	Predicted 1	Predicted 2	Predicted 6	acc
Actual 1	21	0	2	0.913
Actual 2	1	57	1	0.966
Actual 6	2	2	40	0.909

No file drop

cost_matrix = $\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$

Train (acc 0.935)

	Predicted 1	Predicted 2	Predicted 6	acc
Actual 1	72	5	9	0.837
Actual 2	6	178	2	0.956
Actual 6	2	1	110	0.973

Test (acc 0.813)

	Predicted 1	Predicted 2	Predicted 6	acc
Actual 1	16	6	3	0.64
Actual 2	3	50	6	0.847
Actual 6	5	1	39	0.886

cost_matrix = $\begin{bmatrix} 0 & 1.2 & 1.2 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$

Train (acc 0.887)

	Predicted 1	Predicted 2	Predicted 6	acc
Actual 1	67	4	15	0.779
Actual 2	11	167	8	0.897
Actual 6	6	1	106	0.938

Test (acc 0.806)

	Predicted 1	Predicted 2	Predicted 6	acc
Actual 1	18	5	2	0.720
Actual 2	3	50	6	0.847
Actual 6	8	1	36	0.800