Use data include all patients. Pick the last 20% data out from each class. Shuffle and split the reset data to 20% for test and 80% for training. Use data, whose label1 and label2 are same. Use continuous wavelet transform to extract the features and feed it to CNN model.

label	**1**	**2**	**3**	**6**
number	2370	5348	47	2941

The number of data with label 3 is too less, so I didn't use it.

Train (acc 1.000)

	Predicted 1	Predicted 2	Predicted 6
Actual 1	1503	0	0
Actual 2	0	3425	0
Actual 6	0	0	1884

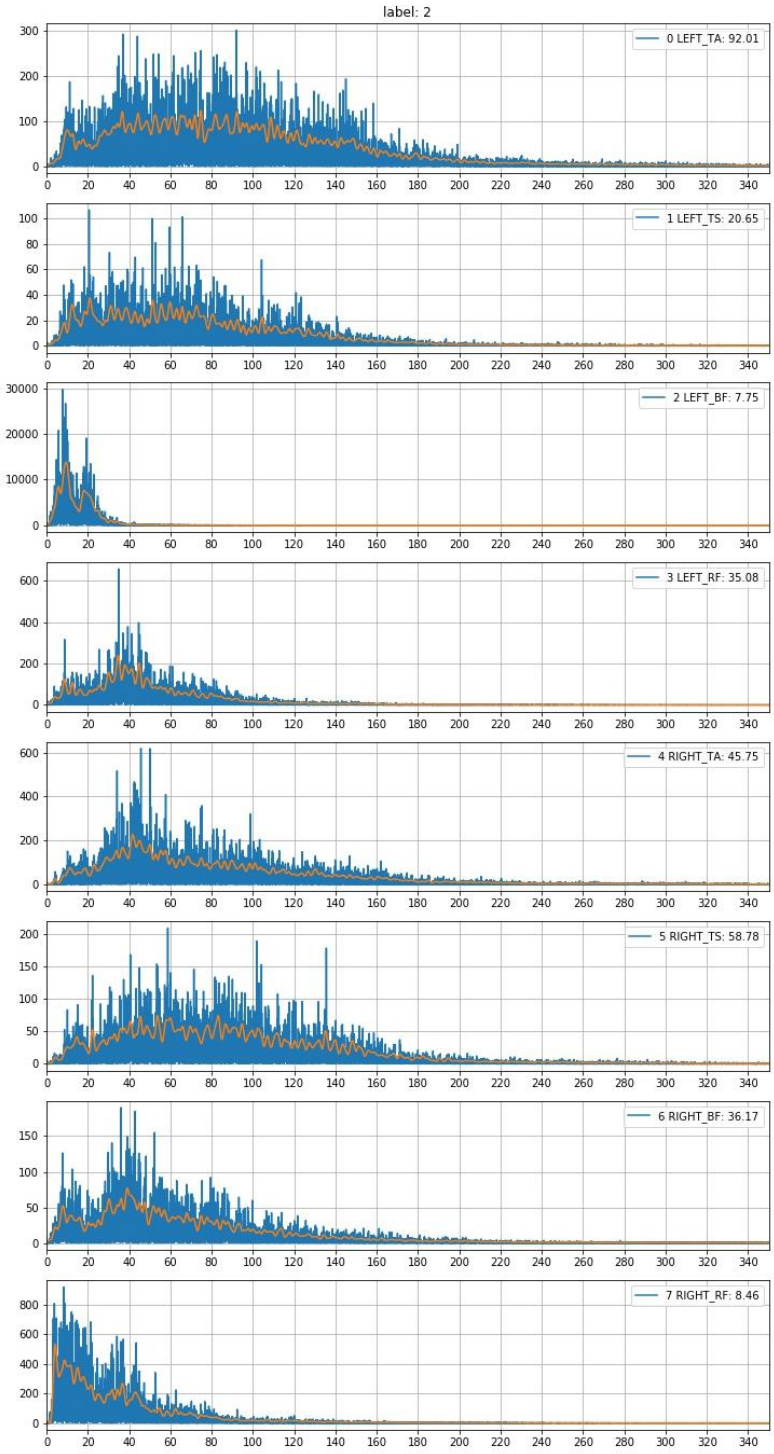
Test (acc 0.966)

	Predicted 1	Predicted 2	Predicted 6
Actual 1	371	10	10
Actual 2	15	823	8
Actual 6	9	5	453

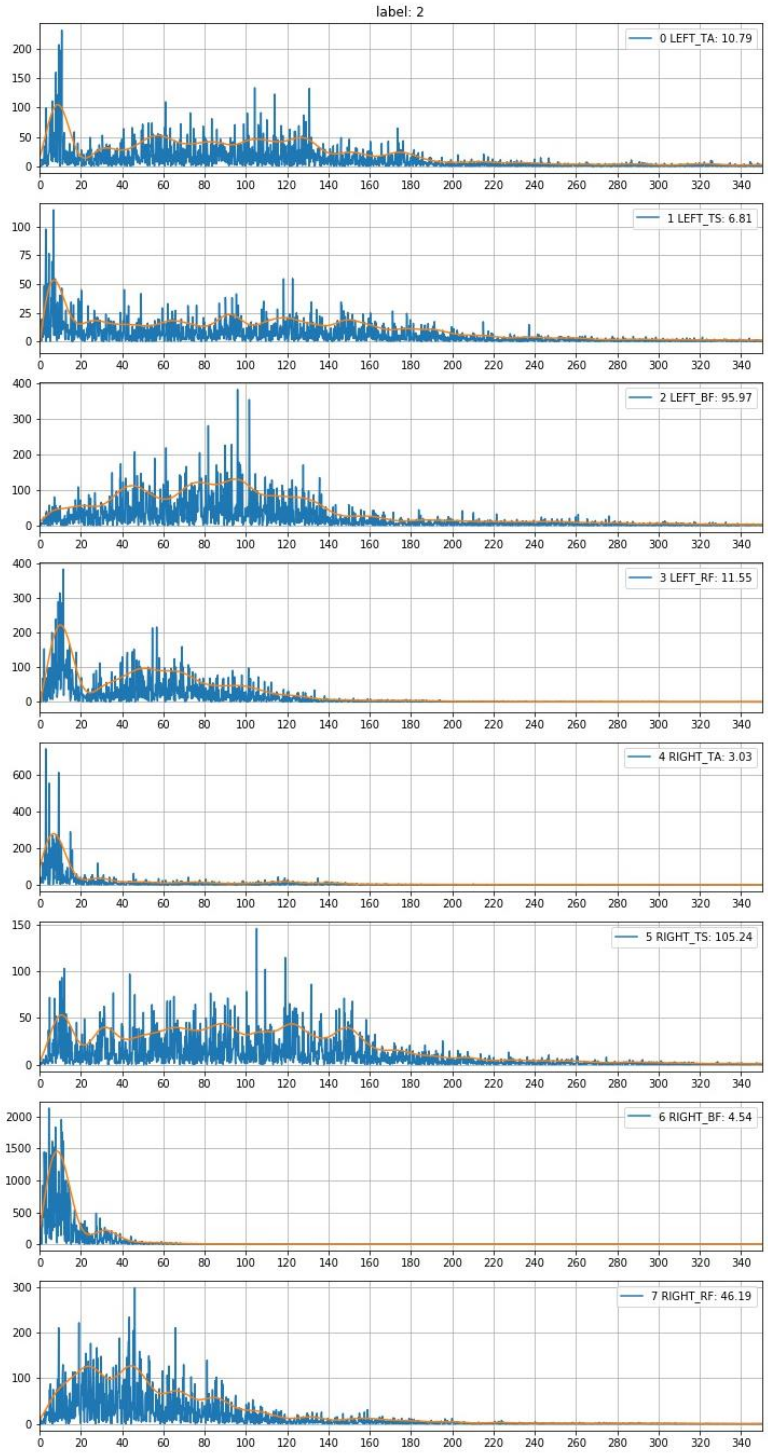
Last 20% (acc 0.711)

	Predicted 1	Predicted 2	Predicted 6
Actual 1	191	184	101
Actual 2	121	898	58
Actual 6	56	98	436

Generation is not good. I checked the Power Spectral Density of the data from each label. I didn't find a certain pattern. They differ from each trial too much. In some trials there are some very obvious components of low frequency, but in others even with same label there can be not. I'm not sure, whether these components of low frequency are disturbance. If I filter them, the results will not be better. The amplitude of EMG signal vary also very widely. Some can reach over 1000, some are just around 300. I tried to scale the signal, it can make the model better, but the generation is still not good.



P812_M050_2_B_FoG_trial_1_emg



P379_M050_2_OFF_B_FoG_trial_2_emg

Parameters:

input_ = layers.Input(shape=X_full.shape[1:])
max_pool = layers.MaxPooling2D((2,2))(input_)

```
cnn1 = layers.Conv2D(16,(2,6),strides=(1,1),
                     kernel_initializer=TruncatedNormal(),
                     padding='same')(max_pool)
cnn1 = layers.Activation('relu')(cnn1)
cnn1 = layers.BatchNormalization()(cnn1)
cnn1 = layers.MaxPooling2D((2,2))(cnn1)

cnn2 = layers.Conv2D(32,(2,6),strides=(1,1),
                     kernel_initializer=TruncatedNormal(),
                     padding='same')(cnn1)
cnn2 = layers.Activation('relu')(cnn2)
cnn2 = layers.BatchNormalization()(cnn2)
cnn2 = layers.MaxPooling2D(2)(cnn2)

cnn3 = layers.Conv2D(64,(2,6),strides=(1,1),
                     kernel_initializer=TruncatedNormal(),
                     padding='same')(cnn2)
cnn3 = layers.Activation('relu')(cnn3)
cnn3 = layers.BatchNormalization()(cnn3)
cnn3 = layers.MaxPooling2D(2)(cnn3)

cnn4 = layers.Conv2D(128,(2,6),strides=(1,1),
                     kernel_initializer=TruncatedNormal(),
                     padding='same')(cnn3)
cnn4 = layers.Activation('relu')(cnn4)
cnn4 = layers.BatchNormalization()(cnn4)
cnn4 = layers.MaxPooling2D(2)(cnn4)

flatten = layers.Flatten()(cnn4)
output = layers.Dense(3,activation = 'softmax')(flatten)
```