Window size:1024
Stride: 256
Raw data with detrend(lambda:300)
Only use channels LEFT_TA, LEFT_TS, RIGHT_TA, RIGHT_TS.
No scale on raw signal. Not shuffle and pick the last 20% data of each labels out for validation.
Shuffle and split the rest to 80% for training and 20% for test.

Label: 2,6

Train (acc 0.997)

	Predicted 2	Predicted 6
Actual 2	632	1
Actual 6	2	398

Test (acc 0.980)

	Predicted 2	Predicted 6
Actual 2	156	4
Actual 6	1	98

Last 20% (acc 0.924)

	Predicted 2	Predicted 6
Actual 2	191	16
Actual 6	9	117

Label:1,6

Train (acc 0.998)

	Predicted 1	Predicted 6
Actual 1	276	1
Actual 6	0	398

Test (acc 0 946)

	Predicted 1	Predicted 6
Actual 1	65	3
Actual 6	6	95

Last 20% (acc 0.817)

	Predicted 1	Predicted 6
Actual 1	58	30
Actual 6	9	117

Label:1,2

Train (acc 0.989)

	Predicted 1	Predicted 2
Actual 1	280	0
Actual 2	10	620

Test (acc 0 942)

	Predicted 1	Predicted 2
Actual 1	63	2
Actual 2	11	152

Last 20% (acc 0.823)

	Predicted 1	Predicted 2
Actual 1	68	20
Actual 2	32	175

Label:1,2,6

Train (acc 0.996)

	Predicted 1	Predicted 2	Predicted 6
Actual 1	271	0	2
Actual 2	1	639	1
Actual 6	0	0	395

Test (acc 0.978)

	Predicted 1	Predicted 2	Predicted 6
Actual 1	70	0	2
Actual 2	4	148	0
Actual 6	1	0	103

Last 20% (acc 0.838)

	Predicted 1	Predicted 2	Predicted 6
Actual 1	60	14	14
Actual 2	16	178	13
Actual 6	5	6	115

rate=0.2
kernel_size=7
kernel_size2=5
stride=1
reg=0
acti='relu'

```
input_ = layers.Input(shape=X[:,:,[0,1,4,5]].shape[1:])

cnn1 = layers.Conv1D(256,kernel_size,strides=stride,
                     kernel_regularizer=regularizers.l2(reg),
                      padding='same')(input_)
cnn1 = layers.Activation(acti)(cnn1)
cnn1 = layers.MaxPooling1D(2)(cnn1)
cnn1 = layers.Dropout(rate)(cnn1)

cnn2 = layers.Conv1D(128,kernel_size,strides=stride,
                     kernel_regularizer=regularizers.l2(reg),
                      padding='same')(cnn1)
cnn2 = layers.Activation(acti)(cnn2)
cnn2 = layers.BatchNormalization(momentum=0.8)(cnn2)
cnn2 = layers.MaxPooling1D(2)(cnn2)
cnn2 = layers.Dropout(rate)(cnn2)

cnn3 = layers.Conv1D(64,kernel_size,strides=stride,
                     kernel_regularizer=regularizers.l2(reg),
                      padding='same')(cnn2)
cnn3 = layers.Activation(acti)(cnn3)
cnn3 = layers.BatchNormalization(momentum=0.8)(cnn3)
cnn3 = layers.MaxPooling1D(2)(cnn3)
cnn3 = layers.Dropout(rate)(cnn3)

cnn4 = layers.Conv1D(32,kernel_size2,strides=stride,
                     kernel_regularizer=regularizers.l2(reg),
                      padding='same')(cnn3)
cnn4 = layers.Activation(acti)(cnn4)
cnn4 = layers.BatchNormalization(momentum=0.8)(cnn4)
cnn4 = layers.MaxPooling1D(2)(cnn4)
cnn4 = layers.Dropout(rate)(cnn4)

cnn5 = layers.Conv1D(16,kernel_size,strides=stride,
                     kernel_regularizer=regularizers.l2(reg),
                      padding='same')(cnn4)
cnn5 = layers.Activation(acti)(cnn5)
cnn5 = layers.BatchNormalization(momentum=0.8)(cnn5)
cnn5 = layers.MaxPooling1D(2)(cnn5)
cnn5 = layers.Dropout(rate)(cnn5)

cnn6 = layers.Conv1D(8,kernel_size2,strides=stride,
                     kernel_regularizer=regularizers.l2(reg),
                                        padding='same')(cnn5)
cnn6 = layers.Activation(acti)(cnn6)
cnn6 = layers.BatchNormalization(momentum=0.8)(cnn6)
cnn6 = layers.MaxPooling1D(2)(cnn6)
cnn6 = layers.Dropout(rate)(cnn6)

cnn7 = layers.Conv1D(4,kernel_size2,strides=stride,
                                      kernel_regularizer=regularizers.l2(reg),
                                        padding='same')(cnn6)
cnn7 = layers.Activation(acti)(cnn7)
cnn7 = layers.BatchNormalization(momentum=0.8)(cnn7)
cnn7 = layers.MaxPooling1D(2)(cnn7)
cnn7 = layers.Dropout(rate)(cnn7)

flatten = layers.Flatten()(cnn7)
dropout = layers.Dropout(rate)(flatten)
output = layers.Dense(2,activation = 'softmax')(dropout)
model = Model(inputs=[input_],outputs=[output])
```