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.

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.

CNN:

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

ANN:

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 0.927)

	Predicted 1	Predicted 2	Predicted 6
Actual 1	326	41	24
Actual 2	23	810	13
Actual 6	12	11	444

Last 20% (acc 0.719)

	Predicted 1	Predicted 2	Predicted 6
Actual 1	161	184	131
Actual 2	80	935	62
Actual 6	42	102	446

Combined:

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 0.964)

	Predicted 1	Predicted 2	Predicted 6
Actual 1	366	17	8
Actual 2	17	822	7
Actual 6	9	2	456

Last 20% (acc 0.740)

	Predicted 1	Predicted 2	Predicted 6
Actual 1	198	147	131
Actual 2	91	925	61
Actual 6	46	80	464

Parameters:

CNN:

```
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)



ANN:
model = models.Sequential()
model.add(layers.BatchNormalization())
model.add(layers.Dense(128,#activation='elu',
                           #kernel_initializer='lecun_normal',
                           #kernel_regularizer = regularizers.l2(0.001),
                                 #use_bias=False
                          ))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Dense(64,#activation='elu',
                           #kernel_initializer='lecun_normal',
                           #kernel_regularizer = regularizers.l2(0.001),
                                 # use_bias=False
                          ))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Dense(32,#activation='elu',
                           #kernel_initializer='lecun_normal',
                           #kernel_regularizer = regularizers.l2(0.001),
                                 #use_bias=False
```

```
                        ))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Dense(16,#activation='elu',
                        #kernel_regularizer = regularizers.l2(0.001),
                                #use_bias=False
                        ))
model.add(layers.BatchNormalization())
model.add(layers.Dense(3,activation='softmax'))

combined:
input_ann = layers.Input(shape=feature.shape[1:],name='input_ann')
ann = model_ann.layers[0](input_ann)
for layer in model_ann.layers[1:-2]:
    ann = layer(ann)
    ann.trainable = False

input_cnn = layers.Input(shape=X_full.shape[1:])
cnn = model_cnn.layers[0](input_cnn)
for layer in model_cnn.layers[1:-1]:
    cnn = layer(cnn)
    cnn.trainable = False
concat = layers.Concatenate()([ann,cnn])
dense1 = layers.Dense(1024,activation='relu',name='dense_concat')(concat)
dropout = layers.Dropout(0.2)(dense1)
output = layers.Dense(3,activation='softmax',name='output')(concat)

model = Model(inputs=[input_ann,input_cnn],outputs=[output])
```