東南大學

医学信号获取和处理 课程设计

题　目　利用深度卷积网络与小波变换实现对心电数据的学习分类

生物科学与医学工程 院（系） 生物医学工程 专业

学　号　171848

学生姓名　张 冰

指导教师　汪永平

起止日期　2018.6.30

设计地点　东南大学

The Biomedical Signal Final Project

Zhang Bing 171848

Purpose：

1．This is the last project of this semester, finally!
2．This project will be involved with:
   1) ECG data sets
   2) Wavelet transform and scalogram
   3) Convolution Neural Network
   4) Some basic concepts of image processing
3．Please go to the following link and study "Signal Classification with Wavelet Analysis and Convolutional Neural Networks(CNN)*"
https://www.mathworks.com/help/wavelet/examples/signal-classification-with-wavelet-analysis-and-convolutional-neural-networks.html
*We will go over the concept of Convolution Neural Networks (CNN) in our next lesson.
4．Practice the examples demonstrated in the studying materials and configure a CNN for ECG classification.
5．Randomly download three ECG data sets from public ECG database, use the trained CNN to classify the downloaded data and report the results.
6．Due date: June 30, 2018.

BACKGROUND REIVEW：

Wavelet Analysis
Using continuous wavelet analysis, spectral features evolve over time can be explore and common time-varying patterns can be identified in two signals and perform time-localized filtering. Using discrete wavelet analysis, we can analyze signals and images at different resolutions to detect changepoints, discontinuities, and other events not readily visible in raw data. Signal statistics can be compared on multiple scales and hidden patterns will be revealed by performing fractal analysis of data.

Scalogram
In signal processing, a scalogram is a visual method of displaying a wavelet transform. There are 3 axes: x representing time, y representing scale, and z representing coefficient value. The z axis is often shown by varying the colour or brightness.

Convolution Neural Network
In machine learning, a convolutional neural network is a class of deep artificial neural networks, most commonly applied to analyzing visual imagery. Convolutional networks are inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field. Applications of CNN are developed for image and video recognition, recommender systems and natural language processing.

Convolution Neural Network

In machine learning, a convolutional neural network is a class of deep artificial neural networks, most commonly applied to analyzing visual imagery. Convolutional networks are inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field. Applications of CNN are developed for image and video recognition, recommender systems and natural language processing.

METHODS AND IMPLEMENTATION:

We use the GoogLeNet as Model to predict the scalogram pictures translated from ECG digital data with continuous wavelet translate (Figure.1).

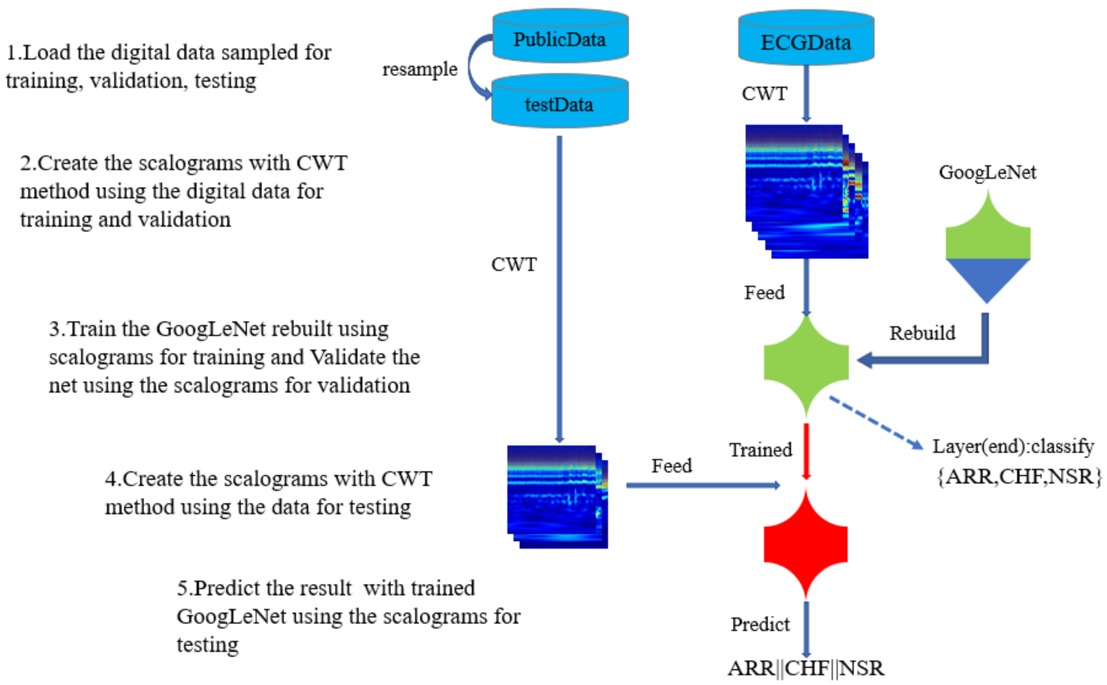

Figure.1 The schematic diagram of the algorithm.

Compilation of data with ECGData

In this example, ECGdata obtained from three groups of people: persons with cardiac arrhythmia (ARR), persons with congestive heart failure (CHF), and persons with normal sinus rhythms (NSR). In total 162 ECG data recorded from three PhysioNet databases: MIT-BIH Arrhythmia Database, MIT-BIH Normal Sinus Rhythm Database, and The BIDMC Congestive Heart Failure Database. More specifically, 96 recordings from persons with arrhythmia, 30 recordings from persons with congestive heart failure, and 36 recordings from persons with normal sinus rhythms. ECGData is a structure array with two fields: Data and Labels. The Data field is a 162-by-65536 matrix where each row is an ECG recording sampled at 128 hertz. Labels is a 162-by-1 cell array of diagnostic labels, one for each row of Data. The three diagnostic categories are: 'ARR', 'CHF', and 'NSR'.

TestData were download randomly from three databases mentioned above. Because the sampling frequency of each databases is different, resample function was used to rebuild the frequency parameters of all to 128Hz.

Create Time-Frequency Representations

After making the folders, create time-frequency representations of the ECG signals. These representations are called scalograms. A scalogram is the absolute value of the CWT coefficients of a signal.To create the scalograms, precompute a CWT filter bank. Precomputing the CWT filter bank is the preferred method when obtaining the CWT of many signals using the same parameters.
helperCreateRGBfromTF, a function to create the scalograms as RGB images and write them to the appropriate subdirectory in dataDir.

Rebuild GoogLeNet To Suit Categories

Load the pretrained GoogLeNet neural network (Figure 2). Each layer in the network architecture can be considered a filter. The earlier layers identify more common features of images, such as blobs, edges, and colors. The later layers focus on more specific features in order to differentiate categories.
To retrain GoogLeNet to our ECG classification problem, replace the last four layers of the network. The first of the four layers, 'pool5-drop_7x7_s1' is a dropout layer. A dropout layer randomly sets input elements to zero with a given probability. The dropout layer is used to help prevent overfitting. The default probability is 0.5. See dropoutLayer for more information. The three remaining layers, 'loss3-classifier', 'prob', and 'output', contain information on how to combine the features that the network extracts into class probabilities and labels. By default, the last three layers are configured for 1000 categories.
Add four new layers to the layer graph: a dropout layer with a probability of 60% dropout, a fully connected layer, a softmax layer, and a classification output layer. Set the final fully connected layer to have the same size 3 as the number of classes in the new data set. To learn faster in the new layers than in the transferred layers, increase the learning rate factors of the fully connected layer. Store the GoogLeNet image dimensions in inputSize.

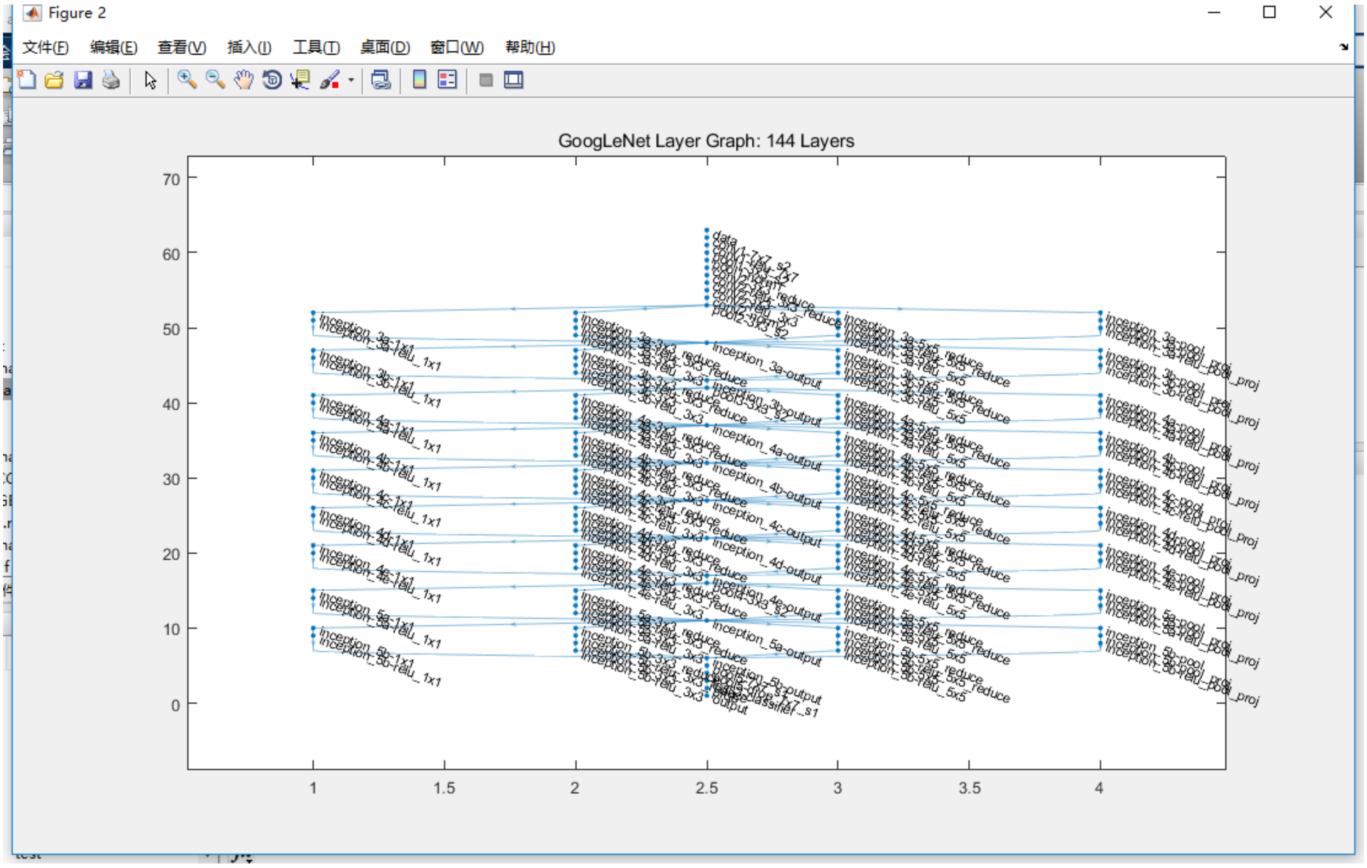

Figure.2 The layer graph of GoogLeNet.

Train Network with Scalograms.

Use the trainingOptions function to specify the training options. Set MiniBatchSize to 10, MaxEpochs to 10, and InitialLearnRate to 0.0001. Visualize training progress by setting Plots to training-progress. Use the stochastic gradient descent with momentum optimizer. By default, training is done on a GPU if one is available. For purposes of reproducibility, train the network using only one CPU, by setting the ExecutionEnvironment to cpu, and set the random seed to the default value. Run-times will be faster if you are able to use a GPU. Because my computer didn't use Navida graphics card, I couldn't set up GPU. The data in the training process, as shown in Figure 3 and Figure 4.

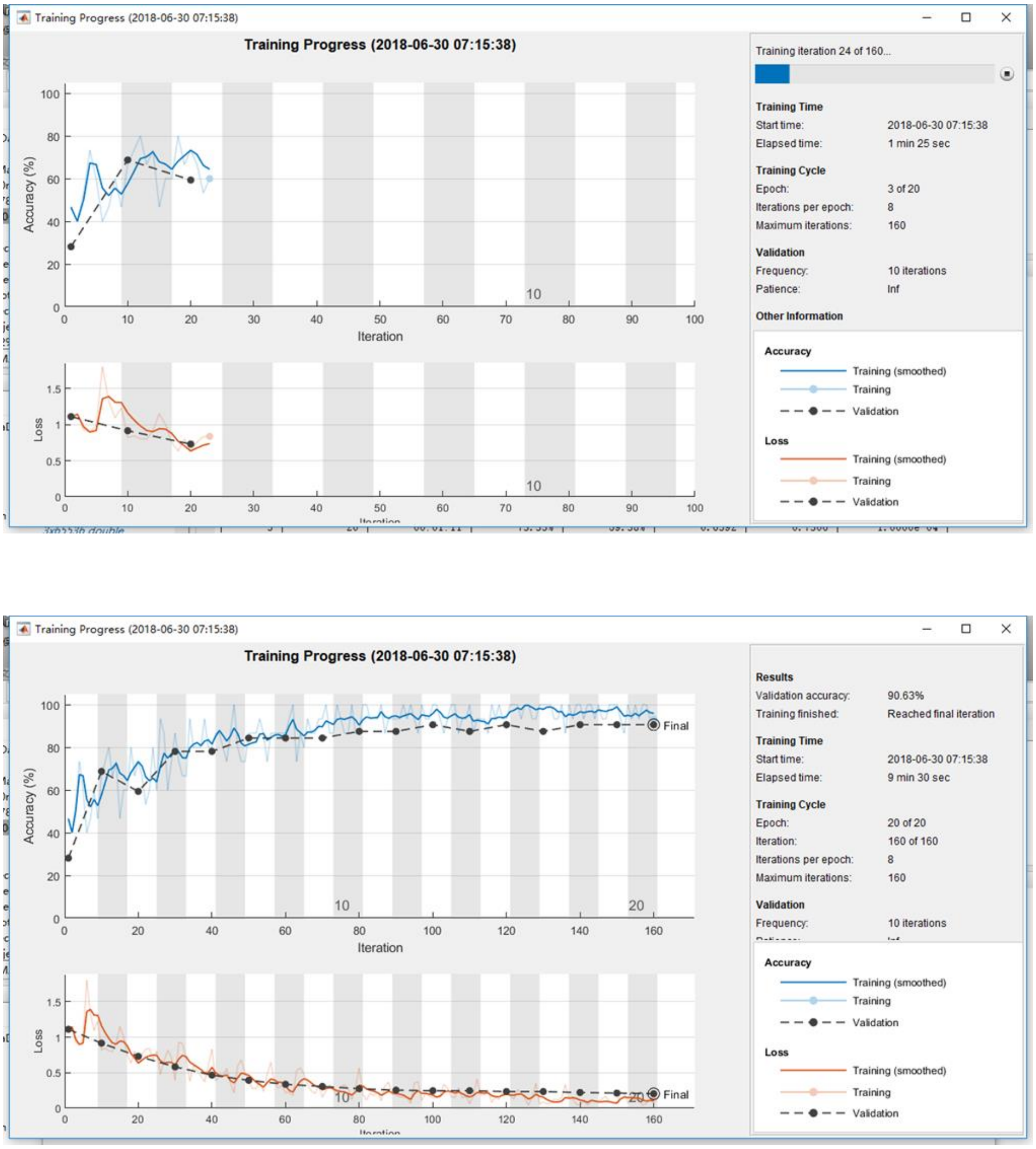

Figure.3 Graphical display of training accuracy and loss function

Initializing image normalization.

Epoch	Iteration	Time Elapsed (hh:mm:ss)	Mini-batch Accuracy	Validation Accuracy	Mini-batch Loss	Validation Loss	Base Learning Rate
1	1	00:00:03	46.67%	28.13%	1.0992	1.1112	1.0000e-04
2	10	00:00:32	66.67%	68.75%	0.8230	0.9156	1.0000e-04
3	20	00:01:11	73.33%	59.38%	0.6392	0.7306	1.0000e-04

Initializing image normalization.

Epoch	Iteration	Time Elapsed (hh:mm:ss)	Mini-batch Accuracy	Validation Accuracy	Mini-batch Loss	Validation Loss	Base Learning Rate
1	1	00:00:03	46.67%	28.13%	1.0992	1.1112	1.0000e-04
2	10	00:00:32	66.67%	68.75%	0.8230	0.9156	1.0000e-04
3	20	00:01:07	73.33%	59.38%	0.6392	0.7306	1.0000e-04
4	30	00:01:42	86.67%	78.13%	0.3958	0.5820	1.0000e-04
5	40	00:02:17	80.00%	78.13%	0.8311	0.4660	1.0000e-04
7	50	00:02:54	86.67%	84.38%	0.3828	0.3923	1.0000e-04
8	60	00:03:29	86.67%	84.38%	0.2167	0.3366	1.0000e-04
9	70	00:04:04	93.33%	84.38%	0.1547	0.3062	1.0000e-04
10	80	00:04:39	86.67%	87.50%	0.5641	0.2738	1.0000e-04
12	90	00:05:14	93.33%	87.50%	0.1834	0.2540	1.0000e-04
13	100	00:05:50	100.00%	90.63%	0.1459	0.2460	1.0000e-04
14	110	00:06:25	100.00%	87.50%	0.0651	0.2465	1.0000e-04
15	120	00:07:00	93.33%	90.63%	0.2447	0.2344	1.0000e-04
17	130	00:07:36	100.00%	87.50%	0.0860	0.2340	1.0000e-04
18	140	00:08:11	100.00%	90.63%	0.0407	0.2204	1.0000e-04
19	150	00:08:49	100.00%	90.63%	0.0665	0.2126	1.0000e-04
20	160	00:09:24	93.33%	90.63%	0.1281	0.2028	1.0000e-04

Figure.4 Table display of training accuracy and loss function

Results and Conclusion：

The classification of testData and predict result were as shown in Figure 5. The predict result sequence was the same as the order of testData sequence. The whole flow output was attached at the end.

imgsTest | imgsTest.Files

imgsTest.Files

	1
1	E:\张大白租界\医学信号作业\BS_ProjectFn_张冰_171848\testData\test\ARR.jpg
2	E:\张大白租界\医学信号作业\BS_ProjectFn_张冰_171848\testData\test\CHF.jpg
3	E:\张大白租界\医学信号作业\BS_ProjectFn_张冰_171848\testData\test\NSR.jpg
4	

```
***************以下为预测结果***************
    "预测分类:"    "ARR"    ",正确分类："    "E:\张大白租界\医学信号作业\BS_P…"

    "预测分类:"    "CHF"    ",正确分类："    "E:\张大白租界\医学信号作业\BS_P…"

    "预测分类:"    "NSR"    ",正确分类："    "E:\张大白租界\医学信号作业\BS_P…"
```

Figure.5 The classification of testData and predict result

As the project showed, wavelet transform can decompose one-dimensional ECG data into multi-frequency data in a more detailed way, which makes the data display more detailed. Through this project, we could apply the mature and stable neural network framework to our own applications to solve complex graphics problems .

Attach code：

```
%文件名称 : BS_ProjectFn_zhangbing_171848
%实现功能 : 利用小波变换及卷积神经网络对 ECG 数据进行分类
%
%参考资料 : Signal Classification with Wavelet Analysis and Convolutional Neural Networks
%https://ww2.mathworks.cn/help/wavelet/examples/signal-classification-with-
%wavelet-analysis-and-convolutional-neural-networks.html
%
%作者信息 : 171848-张冰
%           537405288@qq.com
%           18795969032
%修订时间 : 2018 年 6 月 30 日 2 点 06 分
%调用格式 : 无
%参数释义 : 无

%项目路径
addpath(genpath(pwd));
clc;

% ====================数据路径及数据集准备====================
% 训练数据的操作路径
practiceDir = 'practiceData';
practiceDataDir = 'practice';
testDir = 'testData';
testDataDir = 'test';

% 载入网上的训练数据 mat 文件
% mat 文件格式分 data 和 label 两部分
% data 为 162 * 65536 的格式，即为 162 个心电数据样本，采样频率 128HZ
% label 为 162 * 1 的格式，记录了这 162 个心电数据的病理类型。
load(fullfile(practiceDir,'physionet_ECG_data-master','ECGData.mat'))
disp(fullfile(practiceDir,'physionet_ECG_data-master','ECGData.mat'))

% 载入网上随机选择的测试数据的 mat 文件
% 100m 为 ARR
% 16786m 为 NRC
% chf06m 为 CHF
% 所有载入后都是 val 变量名，所有除以 200 的幅度之后重新命名
load(fullfile(testDir,'dataMat','100m.mat'))
ARR100 = val/200;
load(fullfile(testDir,'dataMat','16786m.mat'));
NSR16786 = val/200;
load(fullfile(testDir,'dataMat','chf06m.mat'));
CHF06 = val/200;
clear val;
```

```
%因为采样率不同，首先将采样率都降为 128Hz
ARR_test = resample(ARR100,128,360);
NSR_test = resample(NSR16786,128,128);
CHF_test = resample(CHF06,128,250);
signList(1,:) = ARR_test(1:65536)';
signList(2,:) = NSR_test(1:65536)';
signList(3,:) = CHF_test(1:65536)';

clear ARR_test NSR_test CHF_test ARR100 NSR16786 CHF06
display("ECG 数字数据准备完毕");
% ===================训练数据预处理===================

% 利用辅助函数创建训练数据的训练文件夹
disp("是否需要重新创建存储训练数据图片的文件夹：");
iflag = input('"是"输入 1，"否"输入 0:');
if iflag == 1
    helperCreateECGDirectories(ECGData,practiceDir,practiceDataDir);
    disp("训练数据路径准备完毕，进入下一步...");
else
    disp("训练数据路径准备完毕，进入下一步...");
end

% 利用辅助函数画三个类型数据 ARR，CHF，NSR 的第一个数据图
% 数据图选取三个数据的前一千个数据
helperPlotReps(ECGData);

% 小波变化 scalogram 的例子
% 做连续小波变换 CWT
% 数据长度 1000，频率 128，使用 8 频度分解以 12 为数量
% Fs = 128;
% fb = cwtfilterbank('SignalLength',1000,'SamplingFrequency',Fs,'VoicesPerOctave',12);
% sig = ECGData.Data(1,1:1000);
% [cfs,frq] = wt(fb,sig);
% t =(0:999)/ Fs; figure; pcolor(t,frq,abs(cfs))
% set(gca,'yscale','log'); shading interp ; axis tight ;
% title('Scalogram'); xlabel('Time(s)'); ylabel('Frequency(Hz)')

% 利用辅助函数创建训练数据连续小波变换的图像
disp("是否需要重新创建训练数据小波变化图：");
iflag = input('"是"输入 1，"否"输入 0:');
if iflag == 1
    helperCreateRGBfromTF(ECGData,practiceDir,practiceDataDir);
    disp("训练数据 CWT scalogram 图像绘制完毕，进入下一步...");
else
    disp("训练数据 CWT scalogram 图像绘制完毕，进入下一步...");
end

% 将绘图数据路径、文件夹名称（即病情 label）等信息存储在 matlab 中
```

```
allImages = imageDatastore(fullfile(practiceDir,practiceDataDir),...
                                          'IncludeSubfolders',true,'LabelSource',...
                                          'foldernames');
disp("CWT scalogram 存入 matlab，进入下一步...");

% 将随机种子设为全局默认值
% rng default
% 将随机种子设为固定值
rng(1);
% 将每个图像数据按标签随机分为两份
% 一份为训练数据，一份为验证数据，比例为 0.8
% imgsTrain 和 imgsValidation 也是 imgDataStore 模式，存储图像信息
[imgsTrain,imgsValidation] = splitEachLabel(allImages,0.8,'randomized');
disp(['Number of training images: ',num2str(numel(imgsTrain.Files))]);
disp(['Number of validation images: ',num2str(numel(imgsValidation.Files))]);
disp("googlenet 喂入数据准备完毕，进入下一步...");

% ====================训练 GoogLeNet====================

% 加载预训练的 GoogLeNet
net = googlenet;
% 提取 GoogLeNet 中的神经网络图层图
lgraph = layerGraph(net);
% 提取 GoogLeNet 的神经层数：144 层
numberOfLayers = numel(lgraph.Layers);
% 把 GoogLeNet 的神经网络图画出来
figure('Units','normalized','Position',[0.1 0.1 0.8 0.8]);
plot(lgraph);
title(['GoogLeNet Layer Graph: ',num2str(numberOfLayers),' Layers']);

% 重新配置神经网络
% 删除 GoogLeNet 最后四层
% pool5-drop_7x7_s1 是 droplayer 用来防止过拟合，概率 0.5
% 'loss3-classifier','prob','output'用来分类，适用于 1000 类
lgraph = removeLayers(lgraph,{ 'pool5-drop_7x7_s1',...
                                                'loss3-classifier','prob','output' });
% 我们要分类的数量就是 3 个 label 的数量
numClasses = numel(categories(imgsTrain.Labels));
% 设置新的最后四层
newLayers = [
    % 新的 drop 层，概率 0.6
    dropoutLayer(0.6,'Name','newDropout')
    % 新的全连接层
    fullyConnectedLayer(numClasses,'Name','fc','WeightLearnRateFactor',...
                        5,'BiasLearnRateFactor',5)
    % 新的 softmax 层，可以使值小的部分也被取到
    softmaxLayer('Name','softmax')
    % 分类输出层
```

```
    classificationLayer('Name','classoutput')];
% 将新的四层加在处理过的 GoogLeNet 后
lgraph = addLayers(lgraph,newLayers); % 合并
lgraph = connectLayers(lgraph,'pool5-7x7_s1','newDropout'); % 链接
inputSize = net.Layers(1).InputSize; % 按 GoogleNet 格式训练

% 设置训练参数
options = trainingOptions('sgdm',...
    'MiniBatchSize',15,...
    'MaxEpochs',20,...
    'InitialLearnRate',1e-4,...
    'ValidationData',imgsValidation,...
    'ValidationFrequency',10,...
    'ValidationPatience',Inf,...
    'Verbose',1,...
    'ExecutionEnvironment','cpu',...
    'Plots','training-progress');

% 对编辑后的 GoogLenet 进行训练
% imgsTrain 训练图像存储集
% lgraph     神经网络图层
% options    训练参数
rng default
trainedGN = trainNetwork(imgsTrain,lgraph,options);

trainedGN.Layers(end-2:end);
% 输出结果类的格式为{'ARR','CHF','NSR'}的数组
cNames = trainedGN.Layers(end).ClassNames;
disp("测试分类矩阵：");
disp(cNames);

% 用验证集确认准确度
% YPred 类别
% probs 预测打分
[YPred,probs] = classify(trainedGN,imgsValidation);
accuracy = mean(YPred==imgsValidation.Labels);
display(['GoogLeNet Accuracy: ',num2str(accuracy)]);
% 输出预测值，按照结果类打分，分数高的为预测结果
display(predict(trainedGN,imgsValidation));

% ====================预处理测试 GoogLeNet 的网上数据====================

% 对测试数据转化成图像
[~,signalLength] = size(signList);
fb = cwtfilterbank('SignalLength',signalLength,'VoicesPerOctave',12);
r = size(signList,1);
% 记好下载的数据类型作为文件名
testLabel = ["ARR";"NSR"; "CHF"];
```

```
for ii = 1:r
        cfs = abs(fb.wt(signList(ii,:)));
        im = ind2rgb(im2uint8(rescale(cfs)),jet(128));
        imgLoc = fullfile(fullfile(testDir,testDataDir));
        imFileName = strcat(char(testLabel(ii)),'.jpg');
        imwrite(imresize(im,[224 224]),fullfile(imgLoc,imFileName));
end
imgsTest = imageDatastore(fullfile(testDir,testDataDir),...
                                        'IncludeSubfolders',true,...
                                        'LabelSource','foldernames');
disp("测试图像数据集存储完毕，进入下一步...");

% ====================用训练好的 GoogLeNet 进行预测====================

% 输出测试结果
result = predict(trainedGN,imgsTest);
[max_result, result_index] = max(result,[],2);
disp("打印预测结果的打分矩阵")
disp(result);
testResult = [cNames(result_index(1)); ...
                        cNames(result_index(2)); ...
                        cNames(result_index(3))];
disp("***************以下为预测结果***************")
disp(["预测分类:",testResult(1),",正确分类：",imgsTest.Files(1)]);
disp(["预测分类:",testResult(2),",正确分类：",imgsTest.Files(1)]);
disp(["预测分类:",testResult(3),",正确分类：",imgsTest.Files(1)]);
```