数字信号处理技术与应用大作业实验报告

一、实验目的

本实验旨在利用槽楔松紧度测试仪的敲击声音数据，基于语音信号处理技术，分析并提取不同松紧度对应声音信号的特征，将其根据压力值大小进行分类，获得松紧度和敲击声音频率的关系。

二、实验原理

1、 双门限法实现端点检测[1]

端点检测技术是指从包含语音的一段信号中确定出语音的起始点和结束点。在本实验中，端点检测可以从时域上分离每一次敲击的信号，并选定合适的时域信号长度，从而排除无声段的噪声干扰。

双门限法基于短时能量和短时平均过零率进行端点检测，利用过零率检测清音，用短时能量检测浊音。端点检测具体判断方法见实验内容部分。

信号{x(n)}的短时能量定义为：

$$E_n = \sum_m [x(m) * w(n-m)]^2$$

短时平均过零率定义为:

$$Z_n = \frac{1}{2} * \sum_m |sgn[x(m)] - sgn[x(m-1)]| * w(n-m)$$

其中:

$$\mathrm{w(n)} = \begin{cases} \frac{1}{2} * N, 0 \le n \le N-1 \\ 0, 其他 \end{cases}$$

实验过程中一般取第一段语音之前的背景噪音的参数作为低门限，而高门限根据实验效果进行调整，一般取低门限的 2 到 3 倍。

2、 梅尔频率倒谱系数（MFCC）实现特征提取[2]

梅尔频率倒谱表示一个语音的短时功率谱，是一个语音的对数功率谱在频率的一个非线性梅尔刻度上进行线性余弦转换所得。梅尔倒谱系数是在 Mel 标度频率域提取出来的倒谱参数，Mel 标度描述了人耳频率的非线性特性，它与频率的关系可用下式近似表示：

$$f_{mel}(f) = 2595 * \log(1 + \frac{f}{700})$$

设语音信号{x(n)}的 DFT 为 $X_a(k)$，定义 M 个三角滤波器组 $H_m(k)$:

$$H_m(k) = \begin{cases} 0, k < f(m-1) \\ \frac{k - f(m-1)}{f(m) - f(m-1)}, f(m-1) \le k \le f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)}, f(m) \le k \le f(m+1) \\ 0, k > f(m+1) \end{cases}$$

其中：

$$\sum_{m=0}^{M-1} H_m(k) = 1$$

Mel 滤波器中心频率定义为：

$$\mathrm{f(m)} = \frac{\mathrm{N}}{F_s} * B^{-1}(B(f_l) + m * \frac{B(f_h) - B(f_l)}{M+1})$$

其中 f_h 和 f_l 分别为滤波器组的最高频率和最低频率，F_s 为采样频率，M 为滤波器数目，N 为 FFT 变换点数，式中：

$$B^{-1}(b) = 700 * (e^{\left(\frac{b}{1125}\right)} - 1)$$

每个滤波器组输出的对数能量为：

$$\mathrm{S(m)} = \ln\left(\sum_{k=0}^{N-1} |X_a(k)|^2 * H_m(k)\right), 0 \le m < M$$

经余弦变换得到 MFCC 系数：

$$\mathrm{C(n)} = \sum_{m=0}^{M-1} S(m) * \cos\left(\pi n * \frac{m+0.5}{M}\right), 0 \le n < M$$

标准的倒谱参数 MFCC 只反映了语音参数的静态特性，语音的动态特性可以用这些静态特征的差分谱来描述。实验中主要选取倒谱参数的一阶差分和二阶差分参数作为倒谱参数的补充以提高系统的识别性能。

3、 卷积神经网络实现分类

卷积神经网络（CNN）是一类包含卷积计算且具有深度结构的前馈神经网络。在语音处理领域，卷积神经网络的表现被证实优于隐马尔可夫模型（HMM）、高斯混合模型（GMM ）和其它一些深度算法[3]。

卷积神经网络包含输入层、隐含层、输出层，其中隐含层主要包含卷积层、池化层和全连接层。基本组成单元为神经元节点，每个节点执行运算：

$$\mathrm{t} = \mathrm{f}(\mathrm{W} * A^T + b)$$

其中：f 为激活函数，W 为权值向量，A 为输入向量，b 为偏置。卷积神经网络采用梯度下降算法更新权值和偏置，以最小化损失函数。

三、实验内容

1、 时域上分离每一次敲击的信号，选定合适的时域信号长度。

采用双门限法对信号进行端点检测，实现过程见图 1。先对数据进行分帧，设置帧长、帧移位等参数，计算第一次信号前的噪声占的帧数，再调用 vad_ezml 函数进行端点检测。

```
load(filename);
fs=51200;
start=startpoint(ind);                  % 起始点
x=data/max(abs(data));                  % 幅度归一化
N=length(x);
time=(0:N-1)/fs;
% 数据分帧
wlen=500;                               % 帧长
inc=200;                                % 帧移位(相邻两帧非重叠部分)
fn=fix((N-wlen)/inc)+1;                 % 帧数
NIS=fix((start-wlen)/inc)+1;            % 前导噪声帧数
frameTime=(((1:fn)-1)*inc+wlen/2)/fs;   % 计算每帧对应的时间
[voiceseg,vsl]=vad_ezm1(x,wlen,inc,NIS);% 端点检测
```

为了使不同长度的信号最后输出矩阵维度相同，计算最长信号长度作为矩阵维度，较短的信号输出补零值。

```
% 输出矩阵初始化
dur=0;
for j=1 : vsl       % 计算最长信号的长度
    i=fix(fs*frameTime(voiceseg(j).end))-fix(fs*frameTime(voiceseg(j).begin));
    if dur <= i
        dur=i;
    end
end
prodata=zeros(dur,vsl);
```

最后根据得到的每段信号的起始点，在图上分别用直线和虚线标出，再保存输出矩阵。

```
% 画出起止点位置
plot(time,x);
title('端点检测');
ylabel('幅值');axis([0 max(time) -1 1]);   grid;
xlabel('时间/s');
for k=1 : vsl
    nx1=voiceseg(k).begin;
    nx2=voiceseg(k).end;
    a=fix((fs*frameTime(nx1)));
    b=fix((fs*frameTime(nx2)));
    nxl=b-a+1;
    prodata(1:nxl,k)=x(fix(fs*frameTime(nx1)):fix(fs*frameTime(nx2)));
     line([frameTime(nx1) frameTime(nx1)],[-1.5 1.5],'color','r','LineStyle','-');
     line([frameTime(nx2) frameTime(nx2)],[-1.5 1.5],'color','r','LineStyle','--');
end
% 保存输出矩阵
matname=[num2str(filename1(ind)),'pro.mat'];
save(matname,'prodata');
```

vad_ezml 函数输入信号 x、帧长 wlen、帧移位 inc、前导噪声帧数 NIS，输出分离出的信号数组 voiceseg 和数组大小 vsl（即敲击信号个数）。在 vad_ezml 函数中，先初始化相应变量，并计算每帧数据和初始噪声，设定门限的阈值。

```
function [voiceseg,vsl]=vad_ezml(x,wlen,inc,NIS)
x=x(:);                  % 把x转换成列数组
maxsilence = 15;         % 保持语音段时静音长度允许的最大值
minlen  = 5;             % 语音长度允许的最小值
status  = 0;             % 表示状态
count   = 0;             % 统计语音段长度
silence = 0;             % 统计静音段长度
y=enframe(x,wlen,inc)';  % 分帧
fn=size(y,2);            % 帧数
amp=sum(y.^2);           % 求取短时平均能量
zcr=zc2(y,fn);           % 计算短时平均过零率
ampth=mean(amp(1:NIS));  % 计算初始噪声区间能量的平均值
zcrth=mean(zcr(1:NIS));  % 计算初始噪声区间过零率的平均值
% 设置能量和过零率的阈值
amp2=ampth;
amp1=2*ampth;
zcr2=2*zcrth;
```

开始端点检测时，首先根据短时能量和过零率分别确定高低两个门限，将端点检测分为四段：静音段、过渡段、语音段、结束。在静音段，若能量或过零率超过低门限，就开始标记起始点，进入过渡段。在过渡段，若能量和过零率都回落到低门限下，则恢复为静音段，若任一个超过高门限，则进入语音段。在语音段，如果能量和过零率都降低到门限以下，且总的计时长度小于最短时间门限，则认为是噪音，继续扫描后面数据，否则标记为结束端点。

```
%开始端点检测
xn=1;
for n=1:fn
   switch status
   case {0,1}                              % 0 = 静音, 1 = 可能开始
      if amp(n) > amp1                     % 确信进入语音段
         x1(xn) = max(n-count(xn)-1,1);
         status  = 2;
         silence(xn) = 0;
         count(xn)    = count(xn) + 1;
      elseif amp(n) > amp2 || zcr(n) > zcr2% 可能处于语音段
         status = 1;
         count(xn)  = count(xn) + 1;
      else                                 % 静音状态
         status  = 0;
         count(xn)    = 0;
         x1(xn)=0;
         x2(xn)=0;
      end

case 2                                    % 2 = 语音段
   if amp(n) > amp2 && zcr(n) > zcr2  % 保持在语音段
      count(xn) = count(xn) + 1;
      silence(xn) = 0;
   else                                   % 语音将结束
      silence(xn) = silence(xn)+1;
      if silence(xn) < maxsilence     % 静音还不够长，语音尚未结束
         count(xn)  = count(xn) + 1;
      elseif count(xn) < minlen       % 语音长度太短，认为是静音或噪声
         status  = 0;
         silence(xn) = 0;
         count(xn)    = 0;
      else                                % 语音结束
         status  = 3;
         x2(xn)=x1(xn)+count(xn);
      end
   end
case 3                                  % 语音结束，为下一个语音准备
     status  = 0;
     xn=xn+1;
     count(xn)    = 0;
     silence(xn)=0;
     x1(xn)=0;
     x2(xn)=0;
end
```

2、 对选定的信号进行频域变换，比较并分析不同松紧度信号的特征

调用 fft 函数对信号 cutdata 进行快速傅里叶变换，并调用 abs 函数求其幅度特性。根据采样频率，将采样点数变换为时间和频率，作出信号的时域图和频域图。

```
%对选定信号进行频域变换
fs=51200;
x=(0:length(cutdata)-1)/fs;
y=abs(fft(cutdata));
xf=(0:length(cutdata)-1)'*fs/length(y);
subplot(2,1,1)
plot(x,cutdata);
xlabel('时间/s');
ylabel('幅度');
title('时域图');
subplot(2,1,2);
plot(xf,y);
xlabel('频率/Hz');
ylabel('幅度');
title('频域图');
```

3、 提取信号的特征

Matlab 语音处理工具箱 voicebox 提供了计算 MFCC 的内置函数，可得到梅尔倒谱参数 coeffs，一阶差分参数 delta，二阶差分参数 deltaDelta。最后发现不同松紧度的信号输出维度基本为 37*13，除了压力值为 2000 时为 35*13，压力值为 4000 时为 39*13。为了使矩阵维度相同，不足的部分补充随机数，超出的部分去掉。

```
%计算MFCC
cutdata=filter(0.0625,1,cutdata);%预加重
[coeffs,delta,deltaDelta,loc]= mfcc(cutdata,fs,'WindowLength',128,...
    'OverlapLength',32,"LogEnergy","Ignore",'DeltaWindowLength',9);
%处理矩阵使其维度相同
if idx==5    %(35*13)-> (37*13)
    a=coeffs(end-1:end,:)+randn(2,13);
    coeffs=[coeffs;a];
    a=delta(end-1:end,:)+randn(2,13);
    delta=[delta;a];
    a=deltaDelta(end-1:end,:)+randn(2,13);
    deltaDelta=[deltaDelta;a];
end
if idx==10%(39*13)-> (37*13)
    coeffs(38:39,:)=[];
    delta(38:39,:)=[];
    deltaDelta(38:39,:)=[];
end
```

最后，保存输出矩阵，矩阵名前缀为 coeffs 为对应压力值信号的梅尔倒谱矩阵，前缀为 delta 为对应的一阶差分矩阵，前缀为 deltaDelta 为对应的二阶差分矩阵。

4、 根据特征进行分类，计算分类精度

构建深度神经网络模型，将得到的三个矩阵合在一起构成 3*37*13 的矩阵作为输入，输入依次经过卷积层 1（变成 6*30*10 的矩阵）、归一化层、线性整流层、池化层 1（变成 6*15*5 矩阵）、卷积层 2（变成 36*8*2 的矩阵）、归一化层、线性整流层、池化层 2（变成 36*4*1

的矩阵）、一维化层（变成 144 维向量）、dropout 层、全连接层（变成 10 维向量），最后通过 logsoftmax 激活函数输出。

```
class Net(nn.Module):
    def __init__(self, class_number):
        super(Net, self).__init__()  # 3*37*13
        self.conv1 = nn.Conv2d(3, 6, kernel_size=(8, 4))  # 6*30*10
        self.batchnormal1 = nn.BatchNorm2d(6)
        self.relu = nn.ReLU()
        self.maxpool1 = nn.MaxPool2d(kernel_size=(2, 2), stride=2)  # 6*15*5
        self.conv2 = nn.Conv2d(6, 36, kernel_size=(8, 4))  # 36*8*2
        self.batchnormal2 = nn.BatchNorm2d(36)
        self.maxpool2 = nn.MaxPool2d(kernel_size=(2, 2), stride=2)  # 36*4*1
        self.flatten = nn.Flatten()  # 144
        self.fc = nn.Linear(144, class_number)  # 10
        self.dropout = nn.Dropout2d(0.01)
        self.logsoftmax = nn.LogSoftmax(dim=-1)

    def forward(self, input):
        output = self.conv1(input)
        output = self.batchnormal1(output)
        output = self.relu(output)
        output = self.maxpool1(output)
        output = self.conv2(output)
        output = self.batchnormal2(output)
        output = self.relu(output)
        output = self.maxpool2(output)
        output = self.flatten(output)
        output = self.dropout(output)
        output = self.fc(output)
        output = self.logsoftmax(output)
```

将数据进行处理、打乱、区分训练集和测试集，随后开始迭代 30 次，每次迭代中遍历训练集中的部分数据，将特征参数输入模型，计算 10 维输出中最大的数对应的维度作为预测的类别，与标签进行比较，计算损失函数，再用梯度下降算法更新模型参数。随后遍历测试集中的部分数据，其他操作与训练集基本相同，区别在于不需要更新模型参数。最后得到每次迭代训练集损失函数 train_loss，训练集分类精度 train_accuracy，测试集损失函数 test_loss，测试集分类精度 test_accuracy。最后画图显示结果。

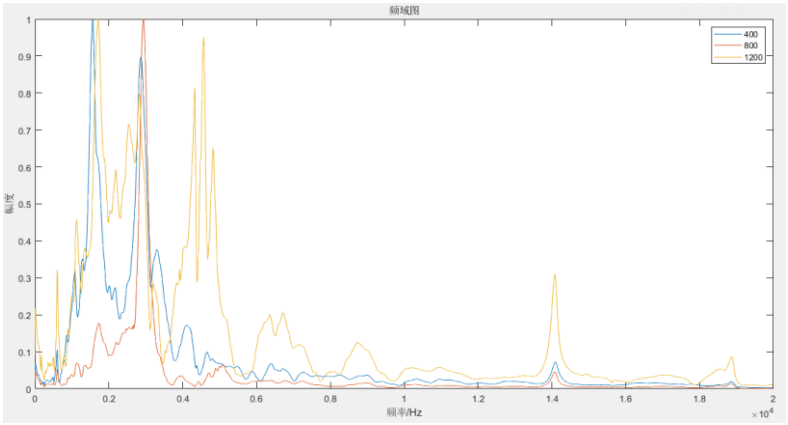
```
train_accuracy, train_loss, test_accuracy, test_loss = [], [], [], []
for epoch in range(30):
    train_correct, train_total, test_correct, test_total = 0, 0, 0, 0
    loss, accuracy = 0, 0
    for datas, labels in train_loader:  # 输入训练集数据
        optimizer.zero_grad()
        outputs = model(datas)  # 输入数据特征参数
        _, predicted = torch.max(outputs.data, 1)  # 计算预测标签
        train_total += labels.size(0)
        train_correct += (predicted == labels).sum().item()  # 预测对了加一
        loss = criterion(outputs, labels)  # 计算训练集损失函数
        loss.backward()  # 梯度下降法调正参数
        optimizer.step()
        accuracy = 100 * torch.true_divide(train_correct, train_total)  # 计算训练集分类精度
    print("epoch:", epoch, " train loss:", loss, " train accuracy:", accuracy)
    train_loss.append(loss)
    train_accuracy.append(accuracy)
    loss, accuracy = 0, 0
    with torch.no_grad():  # 设置为不调整参数
        for datas, labels in test_loader:  # 输入测试集数据
            outputs = model(datas)  # 输入数据特征参数
            _, predicted = torch.max(outputs.data, 1)  # 计算预测标签
            test_total += labels.size(0)
            test_correct += (predicted == labels).sum().item()
            loss = criterion(outputs, labels)  # 计算测试集损失函数
            accuracy = 100 * torch.true_divide(test_correct, test_total)  # 计算测试集分类精度
        print("test loss:", loss, " test accuracy:", accuracy)
    test_loss.append(loss)
    test_accuracy.append(accuracy)
```

5、获得松紧度（压力值）和频率的变化关系

将每个压力值下信号频谱峰值对应的频率视为可能的声音频率。首先，对每个压力值下分离的每一次敲击信号作 fft 变换得到信号频谱，再取平均得到每个压力值下信号的平均频谱。调用 findpeaks 函数，可获得频谱中幅值和间隔满足一定要求的峰值所在的位置，视为该压力值下可能的声音频率。注意到声音频率里面会有本征频率糅合在一起，且压力值越高对应的声音频率越高，因此采用累试法。具体操作步骤为，压力值从小到大，依次绘出相邻两压力值对应的频谱曲线，从左至右曲线差异较大且出现峰值处视为声音频率，若未出现峰值则调整 findpeaks 函数中的参数，结合具体频谱波形，依次分析每个压力值下的声音频率。代码如下图所示：

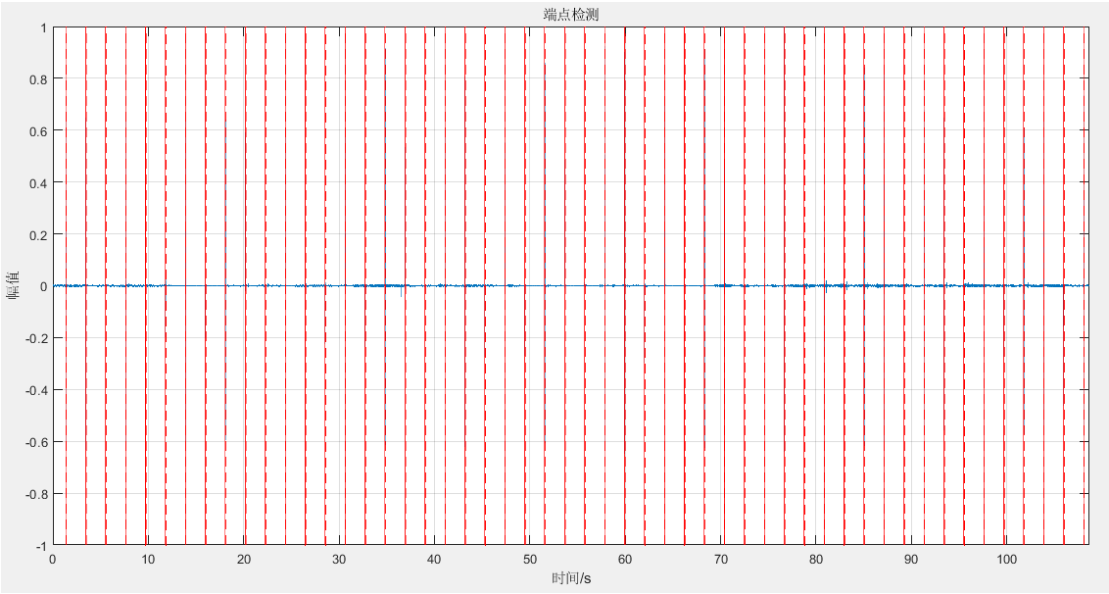
```
for idx=1:length(filename1)
    filename=[num2str(filename1(idx)),filename2];
    load(filename);
    average=zeros(size(prodata,1),1);
    for ind=1:size(prodata,2)  % 对每段信号作fft变换
        cutdata=prodata(:,ind);
        y=abs(fft(cutdata));
        average=average+y;
    end
    average=average/size(prodata,2); % 求每个压力值下的平均频谱
    fs=51200;
    x=(0:size(prodata,1)-1)/fs;
    xf=(0:size(prodata,1)-1)'*fs/length(average);
    ii=find(xf>=20000);
    average(ii)=NaN;    % 只在低频段的频谱中获取声音频率
    average=average/max(average);  % 频谱归一化
    % 找到频谱中的峰值，满足峰值大于mph且峰值间隔大于mpd，具体根据需要调整参数
    mph=0.5;
    mpd=10;
    [pks,locs]= findpeaks(average,'minpeakheight',mph,'minpeakdistance',mpd);
    a=fs/length(average)*locs % 计算峰值最大的频率
    plot(xf,average);
    xlabel('频率/Hz');
    ylabel('幅度');
    title('频域图');
    hold on
end
legend('400','800','1200','1600','2000','2400','2800','3200','3600','4000')
```

以确定前三种压力值的声音频率为例，绘出频谱如下图，确定压力值 400 频率 1578.2Hz，压力值 800 频率 2493.2Hz，压力值 1200 频率 3213Hz，以此类推。

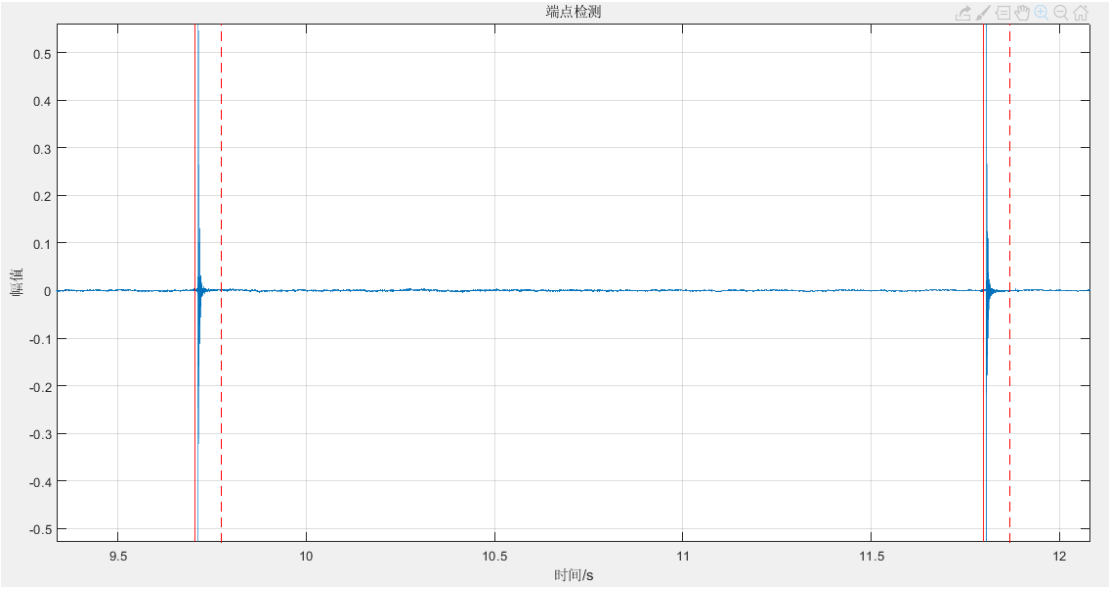


四、实验结果

1、 以压力值为 4000 为例，语音端点检测后得到的结果如下图：



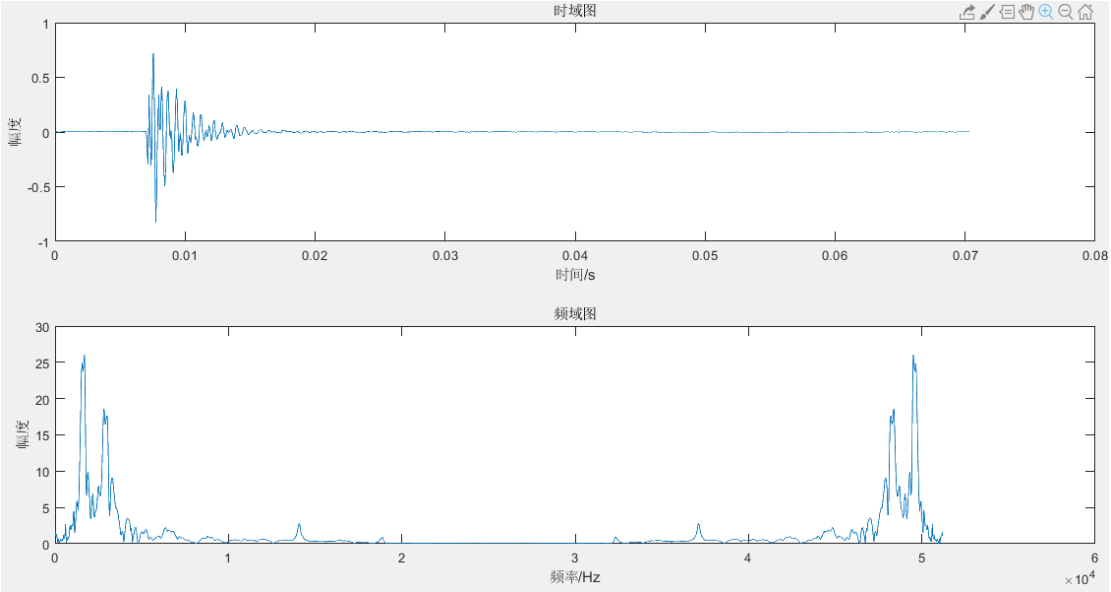
部分放大后：



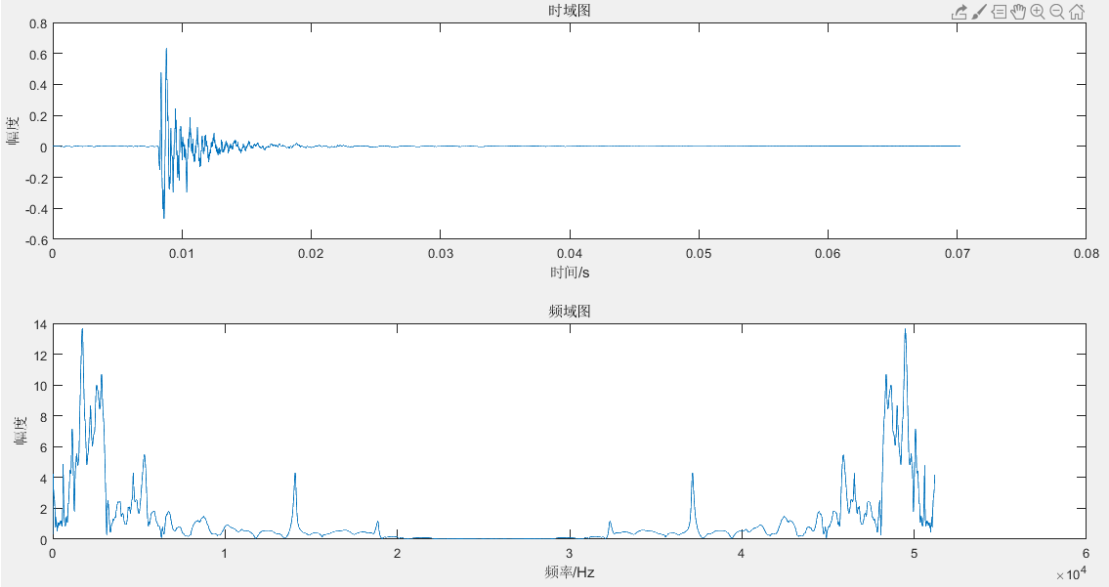
可见分离信号效果较好，基本截取了敲击信号段，滤除了大部分噪声段。

2、 选定信号进行频域变换，分析特征

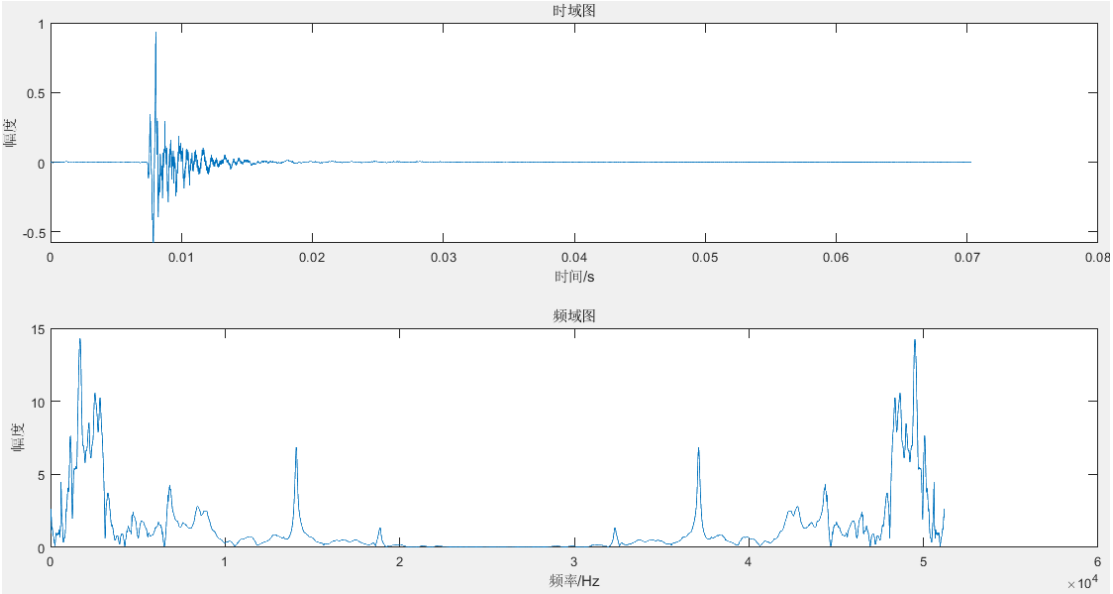
压力值为 400 时其中一个信号幅频特性：



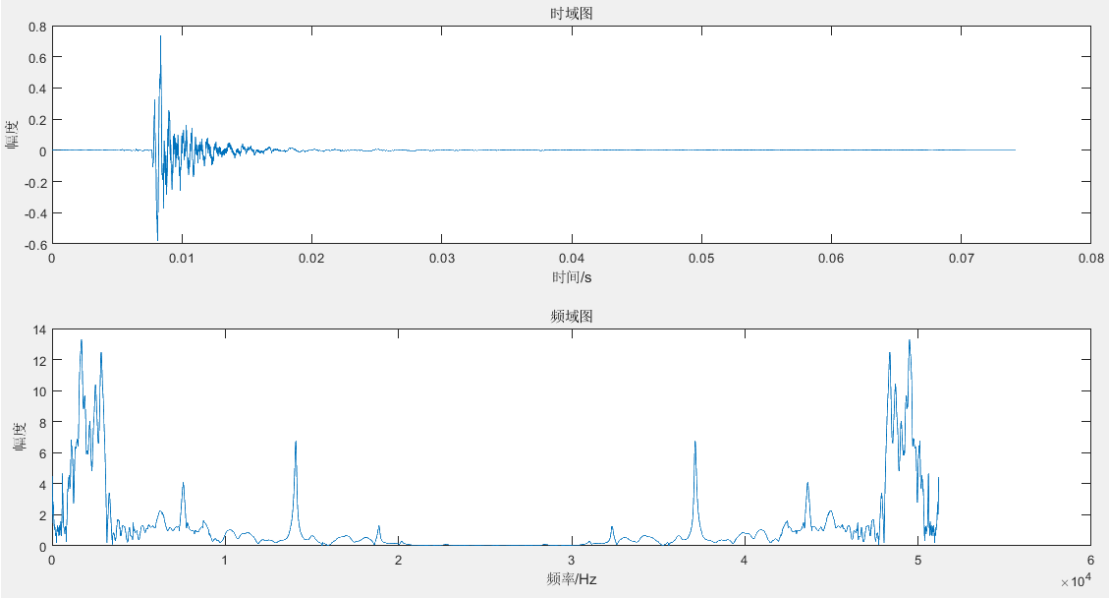
压力值为 1600 时其中一个信号幅频特性：



压力值为 2800 时其中一个信号幅频特性：



压力值为 4000 时其中一个信号幅频特性：

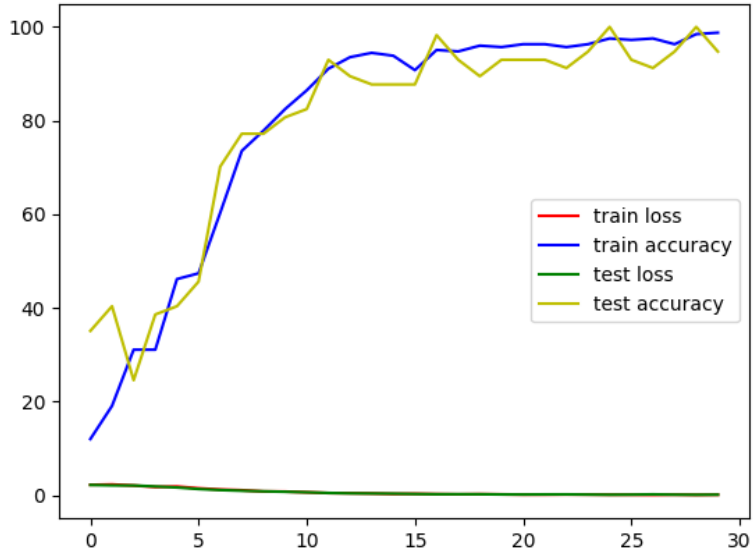


分析可知随着压力值增大，低频段的幅值较大的频域区间逐渐向高频段移动。

3、 使用神经网络进行分类

程序输出和绘制的精度和损失曲线如下图所示。

```
test_for_matlab
test loss: tensor(0.1747)  test accuracy: tensor(91.2281)
epoch: 23  train loss: tensor(0.1472, grad_fn=<NllLossBackward>)  train accuracy: tensor(96.3077)
test loss: tensor(0.1386)  test accuracy: tensor(94.7368)
epoch: 24  train loss: tensor(0.1164, grad_fn=<NllLossBackward>)  train accuracy: tensor(97.5385)
test loss: tensor(0.1026)  test accuracy: tensor(100.)
epoch: 25  train loss: tensor(0.1052, grad_fn=<NllLossBackward>)  train accuracy: tensor(97.2308)
test loss: tensor(0.1255)  test accuracy: tensor(92.9825)
epoch: 26  train loss: tensor(0.0603, grad_fn=<NllLossBackward>)  train accuracy: tensor(97.5385)
test loss: tensor(0.1798)  test accuracy: tensor(91.2281)
epoch: 27  train loss: tensor(0.1231, grad_fn=<NllLossBackward>)  train accuracy: tensor(96.3077)
test loss: tensor(0.1173)  test accuracy: tensor(94.7368)
epoch: 28  train loss: tensor(0.0752, grad_fn=<NllLossBackward>)  train accuracy: tensor(98.4615)
test loss: tensor(0.0916)  test accuracy: tensor(100.)
```



由下图可见，迭代次数在 20 此后精度基本达到了 0.9，损失基本降到 0.2 以下。迭代到 23 次后，训练精度在 0.96 以上，测试精度在 0.94 附近波动，最高达到百分之百的测试精度。

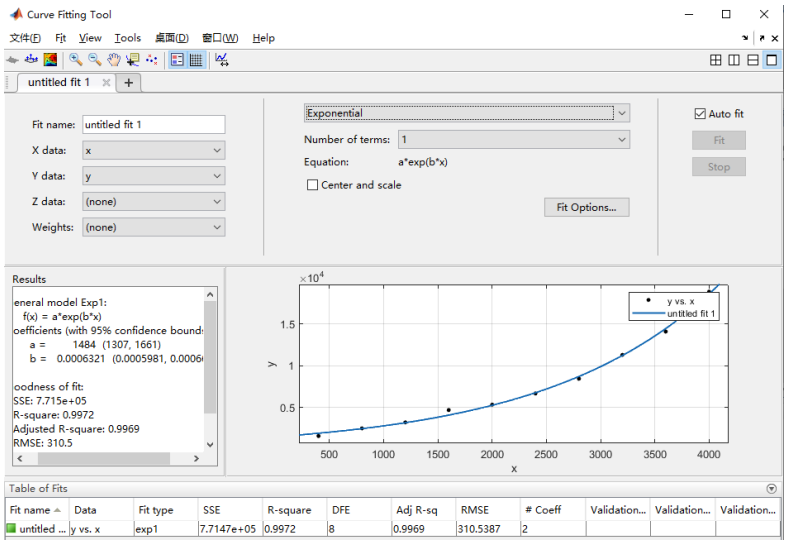
4、获得松紧度（压力值）和频率的变化关系

经分析得到不同压力值对应频率如下表所示：

压力值	400	800	1200	1600	2000	2400	2800	3200	3600	4000
频率/Hz	1578	2493	3213	4692	5331	6683	8446	11289	14090	18863

通过 matlab 的 cftool 工具进行拟合如下图所示，得到压力值 y 和频率的变化关系为：

$$y = 1484 * e^{0.0006321*x}$$



五、参考文献

[1]夏敏磊. 语音端点检测技术研究[D].浙江大学,2005.

[2]王让定,柴佩琪.语音倒谱特征的研究[J].计算机工程,2003(13):31-33.

[3] Abdel-Hamid, O., Mohamed, A.R., Jiang, H. and Penn, G., 2012. Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. In Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on (pp. 4277-4280). IEEE