音频音高识别

Y30190698　　夏源祥

一、实验目的

视唱练耳是对于一位音乐学习者是十分基础且十分重要的一门基础学科，这一门学科分为视唱与练耳部分，所谓视唱，即是通过根据乐谱唱出乐谱所包含的音高，音准和节奏；而练耳则是通过聆听一段乐曲或几个音符以判断其音名。视唱练耳过往一直是由教师与学生之间直接面对面进行教学，这种教学方式有它独有的优点即老师可以在身边即使辅导，但这种教学方式首先对老师的要求极为苛刻，老师也是经过十几年的视唱练耳专业训练，利用十几年的学习经验来教授学生，所以视唱练耳的授课老师都是专业的教授者，而这种专业性和技术性都集中在钢琴演奏的技术，传统的视唱练耳教学就是通过钢琴作为教学工具和媒介，这也就造成了教学中的音色可能比较单一；

美国作为第三次科学技术革命爆发的发源地，计算机作为辅助教育得到了更早的开发，那个时代的美国视唱练耳教师就已经可以通过计算机来对学生进行相应的辅导，随着科技的发展，国外的视唱练耳软件已经成为一个完善的训练系统，利用这个系统，学生和计算机进行交互。一些发达国家，重视视唱练耳数字化的发展，*Clint Randles* 等人关于衡量音乐教育研究中提出了新的音乐教学在音乐教育中的重要性，也在文中提出了计算机在音乐教育研究领域中的重要作用；*RenéeCrawford* 等人探讨了在音乐教育中的信息通信和技术（*ICT*）的可用性和使用，以及音乐技术资源和设施； *Lorenzo- Quiles* 等人分析了基于数字学习对象的教育干预计划以及有关数字化音乐在教学中的效果分析。在实际应用上，国外拥有“*EarMaster*”，“*Overture*”，“*Sibelius*”等知名的视唱练耳软件，国内对于视唱练耳的发展起步较晚，虽然在学术上重视对视唱练耳数字化的发展，但是对于专业的系统设计方面国内仍有不足。

目前市面上流传较广的视唱练耳辅助软件均是基于传统的音高识别即通过与标准音高进行比对来判断待测音高是否准确。本实验尝试采用人声作为训练样本来构建神经网络系统测试其功能效果。采用基于 *BP* 神经网络对音频进行音高识别。

二、BP 神经网络的基本结构

BP(*back propagation*)神经网络是1986年由*Rumelhart*和*McClelland*为首的科学家提出的概念，是一种按照误差逆向传播算法训练的多层前馈神经网络，是目前应用最广泛的神经网络。其结构如下图1.1所示。*BP*神经网络属于有监督的学习，利用一组已知目标的训练样本集作为输入，使用随机数组作为权值及阈值计算结果，根据结果与目标的误差利用梯度下降的学习规则，反向逐层修改权重系数及阈值，使得样本训练集的结果与目标的误差达到最小甚至为零，神经网络就训练完成。

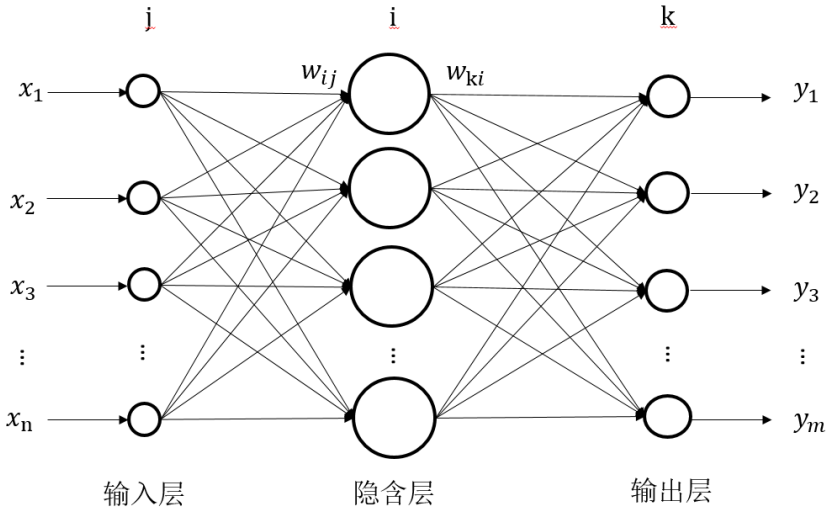

图1.1 BP神经网络结构

1 人工神经元

神经元是神经网络的基本单位，神经元的构成如4.1图所示。

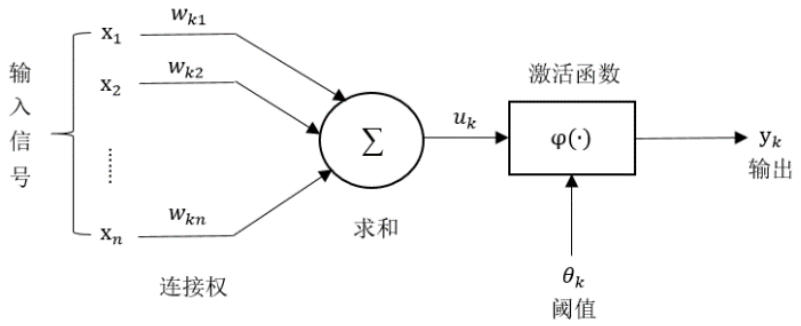

图1.2 神经元结构图

其中$\{x_1,x_2 \ldots x_n\}$为输入向量，$\{w_{k1},w_{k2} \ldots w_{kn}\}$为权重系数。

$$u_k=\sum_{j=1}^{n} w_{kj}x_j \tag{1-1}$$

$$y_k=\varphi(u_k-\theta_k) \tag{1-2}$$

从上图1.2，式（1.1）和式（1.2）我们可以得知，一个神经元的输出y_k与输入向量$\{x_1,x_2...x_n\}$，权重系数$\{w_{k1},w_{k2}...w_{kn}\}$，阈值$\theta_k$和激活函数$\varphi(x)$有关，经过不断修改这些参数可以调整输出$y_k$。一般一个完整的神经网络会有数个神经元，将输入参数输入进入神经元，经过神经元处理之后的输出又将被送到其他神经元用于进一步的处理。

2 权重系数与阈值

神经网络中的神经元以权重系数与阈值作为连接的桥梁。互相连接的两个神经元以权重系数w_{kj}来连接，每一个神经元的输出在传入另一个神经元时都要乘以一个相应的权重，权重的大小也表示了相互连接的神经元的连接强度，权重系数决定了输入对输出的影响程度。

阈值θ_k是每个神经元的额外输入，一般情况下我们都将神经元的阈值的值视为1，但其与神经元之间的连接也由权重系数控制，权重系数的大小也会影响阈值与神经元之间的联系，阈值的目的是改变权重与输入乘积结果的范围。

3 激活函数

激活函数是将非线性函数引入神经网络中，其目的是神经元的输入值缩小到较小的范围。由于输入向量经过神经元处理之后的结果可能比较大，神经网络的目的是解决分类问题，大多数情况需要输出结果是1或者0，代表是或者否。而引入激活函数就可以实现这样的功能，不同的激活函数具有不同的功能函数可以将巨大的数缩小到合适的范围去解决相应的问题。

（1）*sigmoid*函数

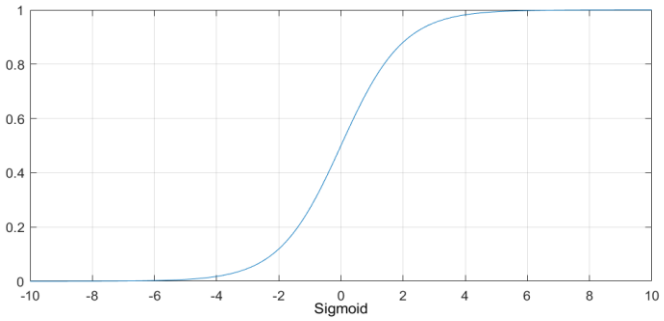

图1.3 sigmoid函数

*sigmoid*函数是BP神经网络中最常见的激活函数，它的功能是将输入的任意实数值压缩到0和1之间，其常用于二分类的神经网络中，将输入在两个类别中进行区分。但*sigmoid*函数存在易饱和的问题，当输入特征值过大或过小时，神经元的梯度就会接近0，使得反向传播算法中的权重系数和阈值未得到修改。

（2） *softmax*函数

$$h_\theta(x^i)=\begin{bmatrix} p(y^i=1|x^i;\theta) \\ p(y^i=2|x^i;\theta) \\ \cdots \\ p(y^i=k|x^i;\theta) \end{bmatrix}=\frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^i}}\begin{bmatrix} e^{\theta_1^T x^i} \\ e^{\theta_2^T x^i} \\ \cdots \\ e^{\theta_k^T x^i} \end{bmatrix} \tag{1-3}$$

*softmax*函数常作为多分类问题的最后一层激活函数，把多个输入分别映射到（0,1）之间，且总和为1。从概率学上来谈，其每一个输出数值则等于该输入分到相对应类别的概率，数值最大的那一类对应的类别则是输入最大可能归属的类别。

（3）*tanh*函数

$$f(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}} \tag{1-4}$$

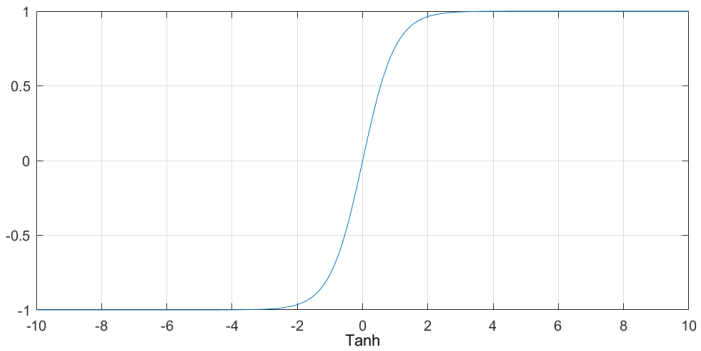

图1.4 tanh函数

*tanh*函数能将任意实数值压缩到-1和1之间。该方法也可用于二分类的问题中，相比*sigmoid*函数来说，*tanh*函数是0均值的，在实际应用中，后者比前者更好。但是*tanh*函数与*sigmoid*函数一样也存在输入值过大或小时的梯度为0的问题。

（4）*ReLU*函数

$$f(x)=max(0,x) \tag{1-5}$$

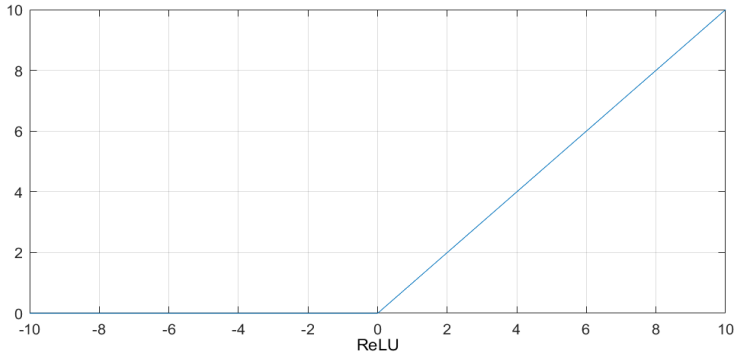

图1.5 ReLU函数

由于*ReLU*函数属于线性函数，且梯度不会饱和，所以收敛速度相比*sigmoid*函数和*tanh*函数就会快很多，而且相比这两个函数来说，*ReLU*函数的计算成本较低，只需要一个阈值就可得到计算值。但*ReLU*也存在危险，如图所示在*x*小于0

时，该函数的梯度是0，就会导致权重系数和阈值无法更新，训练结果误差较大。

4 神经网络的构成

一般的神经网络由输入层、隐含层、输出层三层构成：

输入层的作用是用于输入待分类信息的特征参数，一般有多少个特征就会有多少维度的输入层。

输出层的作用是用于输出待分类信息的判别结果，根据结果可以设定相应维度的输出层。

隐含层则是放置神经元的夹层，输入层的特征参数经过隐含层，在各个神经元之间传输，利用权重系数、阈值和激活函数的处理得到输出层的最终结果。根据神经网络设计的需要隐含层可能不止一层，会出现两层或两层以上的隐含层，隐含层的层数越多，输出层的分类结果越精确，但是同时也会造成整个神经网络的计算复杂、成本增加等问题。隐含层中的每一层都有数个神经元与另外一层的神经元相互连接，隐含层中每层的神经元数量并不是越多越好，选择合适的神经元数量可以更加准确的对待分类信息进行分类。

5 损失函数（*loss function*）

损失函数又称为代价函数（*cost function*），其用于计算神经网络在训练中的损失。一般情况下不同的激活函数都会有各自损失的计算方式。通过计算损失反向传播修改权重系数及阈值使得输入经过神经网络之后能无限接近目标值。

交叉熵损失函数主要常用于分类问题中，其主要原因是它能够更好的克服如*sigmoid*函数中出现的梯度为零的缺点。交叉熵损失函数在二分类中式子如下：

$$J= -[\hat{y}*lny+(1-\hat{y})*ln\,(1-y)] \tag{1-6}$$

其中$\hat{y}$为训练样本的目标值，y为训练样本的实际输出值，J为当前样本的损失值。

在0和1的二分类中的J的取值如图1.6所示。

当目标值$\hat{y}$=1时：

$$J= -\hat{y}*lny \tag{1-7}$$

J的计算值如上图的直线所示，实际输出y的值越接近1，损失值J则越小，否则J的值越大。

当目标值$\hat{y}$=0时：

$$J= -(1-\hat{y})*ln\,(1-y) \tag{1-8}$$

J的计算值如上图的虚线所示，实际输出y的值越接近0，损失值J则越小，否则J的值越大。

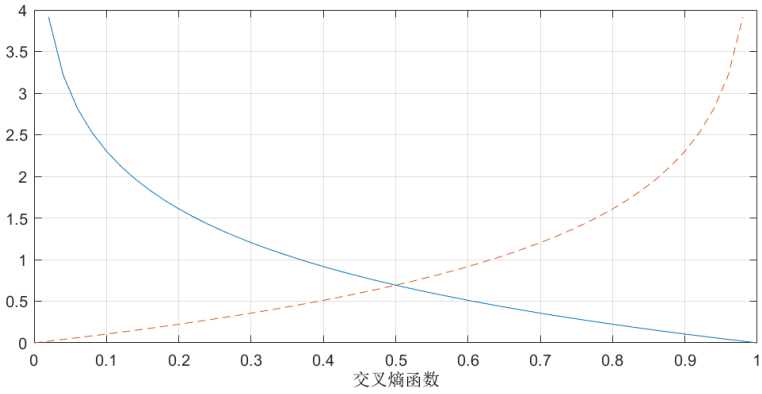

图1.6交叉熵损失函数值

6 前向传播和反向传播

前向传播的意义在于将样本的输入特征从输出层传入神经网络的隐含层之中，经过神经元的权重系数、阈值和激活函数的一系列处理，可以计算出该样本实际结果，若实际结果与目标结果的误差较大，则反向传播将输出误差以某种形式向隐含层方向传播，并将误差分摊给各层神经元，各层神经元以此误差信号作为修改权值的依据。对权重系数、阈值的数值进行调整，使得样本的实际结果不断逼近目标结果。

三、基于 BP 神经网络的音高识别

1 特征提取

由上述实验可知每段音频经过处理之后其仅包含0~1024*Hz*的频谱特征，提取音名*C*3至*B*5共36个音名的对应频率的正负3*Hz*的频率段的频谱特征作为输入特征，之所以选择3Hz，是因为单个音符在正负偏差超过3*Hz*以上时，我们人耳可以很容易的辨别出这个音符的错误。在部分音名之间频率距离大于13*Hz*小于21*Hz*时选取中间合适的7*Hz*频率作为输入特征，在距离大于21*Hz*小于35*Hz*时选取中间两个合适的7*Hz*频率作为输入特征，

同理，在在距离大于35*Hz*小于49*Hz*时选取中间三个合适的7*Hz*频率作为输入特征，这样可以增加待测音频段的测试准确率。本实验中提取出共70段的频谱能量作为接下来实验的训练样本输入特征。其70维输入特征对应的频率如3.1表所示。

输入特征共70个，对于表格第二行表示该特征对应的标准音名，并用灰色标亮显示。

选取偏差极小的音频段作为训练样本，并根据选取的训练样本构造相应的输出目标结果，每段音频的目标结果为36维，对应如3.2表所示。

采用*one-hot*编码，每一段音频对应的频率对应的音名相对的维数置1，其他

置0。并以此来构造训练样本的目标结果。如若一段音频的实际对应的音名为C4，其目标结果为{0,0,0,0, 0,0,0,0, 0,0,0,0, 1,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0}。

表3.1 输入特征对应表（单位：Hz）

序号	音名	频率范围	序号	音名	频率范围	序号	音名	频率范围	序号	音名	频率范围	序号	音名	频率范围
1	C3	128-134	**17**	E4	327-333	**23**		400-406	**39**		595-601	**55**		793-799
2	C#3	136-142	**18**	F4	346-352	**24**	G#4	412-418	**40**		608-614	**56**		805-811
3	D3	144-150	**19**		356-362	**25**		424-430	**41**	D#5	619-625	**57**		816-822
4	D#3	153-159	**20**	F#4	367-373	**26**	A4	437-443	**42**		631-637	**58**	G#5	828-834
5	E3	162-168	**21**		378-384	**27**		450-456	**43**		644-650	**59**		840-846
6	F3	172-178	**22**	G4	389-395	**28**	A#4	463-469	**44**	E5	656-665	**60**		852-858
7	F#3	182-188	**23**		400-406	**29**		477-483	**45**		669-675	**61**		864-870
8	G3	193-199	**24**	G#4	412-418	**30**	B4	491-497	**46**		682-688	**62**	A5	877-883
9	G#3	205-211	**25**		424-430	**31**		505-511	**47**	F5	695-701	**63**		890-896
10	A3	217-223	**26**	A4	437-443	**32**	C5	520-526	**48**		709-715	**64**		903-909
11	A#3	230-236	**27**		450-456	**33**		530-536	**49**		723-729	**65**		916-922
12	B3	244-250	**28**	A#4	463-469	**34**		541-547	**50**	F#5	737-743	**66**	A#5	929-935
13	C4	259-265	**29**		477-483	**35**	C#5	551-557	**51**		748-754	**67**		943-949
14	C#4	274-280	**30**	B4	491-497	**36**		562-568	**52**		759-765	**68**		957-963
15	D4	291-297	**21**		378-384	**37**		573-579	**53**		770-776	**69**		971-977
16	D#4	308-314	**22**	G4	389-395	**38**	D5	584-590	**54**	G5	781-787	**70**	B5	985-991

表3.2 输出特征对应表

维度	音名	维度	音名	维度	音名	维度	音名	维度	音名	维度	音名
1	C3	7	F#3	13	C4	19	F#4	25	C5	31	F#5
2	C#3	8	G3	14	C#4	20	G4	26	C#5	32	G5
3	D3	9	G#3	15	D4	21	G#4	27	D5	33	G#5
4	D#3	10	A3	16	D#4	22	A4	28	D#5	34	A5
5	E3	11	A#3	17	E4	23	A#4	29	E5	35	A#5

2 BP 神经网络的搭建

本实验的*BP*神经网络设计如下图所示：

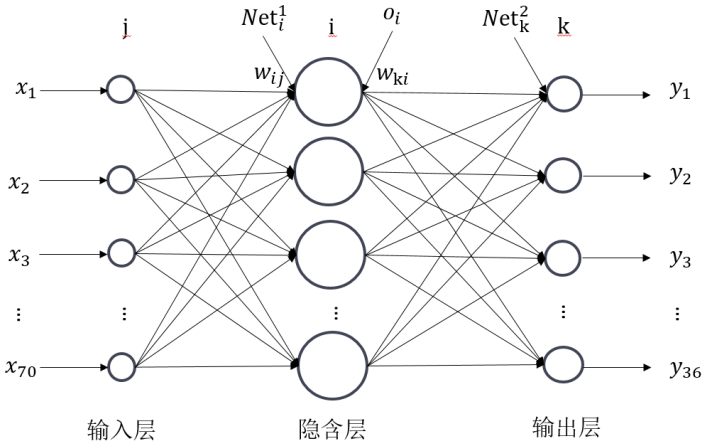

图3.1 本文的神经网络结构

输入层的维度为70，隐含层的维度为150，输出层的维度为36。

由上小节内容已经了解到本实验将采用音频段的70段频谱能量归一化之后作为输入层的输入特征。

输出层的输出采用*one-hot*编码的多分类表示，在输出层中采用*softmax*作为激活函数，使用交叉熵损失函数的学习规则对隐含层与输出层之间的权重及阈值进行调整。

在本实验中，隐含层的输出序列$O^i=\{o_1,o_2,o_3...o_{150}\}$作为输出层的输入特征，i表示第几个样本，由式（1-3）可知

$$y_k=\frac{1}{\sum_{k=1}^{36}e^{a_k}}\begin{bmatrix}e^{a_1}\\e^{a_2}\\...\\e^{a_{36}}\end{bmatrix}=\frac{1}{\sum_{k=1}^{36}e^{w_k^T O^i}}\begin{bmatrix}e^{w_1^T O^i}\\e^{w_2^T O^i}\\...\\e^{w_{36}^T O^i}\end{bmatrix} \tag{3-1}$$

对*softmax*函数进行求导则有：

当$i=j$ 时:

$$\frac{\partial\frac{e^{a_i}}{\sum_{k=1}^{36}e^{a_k}}}{\partial e^{a_j}}=\frac{e^{a_i}(\sum_{k=1}^{36}e^{a_k}-e^{a_j})}{(\sum_{k=1}^{36}e^{a_k})^2}=y_i(1-y_j) \tag{3-2}$$

当$i\neq j$时:

$$\frac{\partial\frac{e^{a_i}}{\sum_{k=1}^{36}e^{a_k}}}{\partial e^{a_j}}=\frac{0-e^{a_i}e^{a_j}}{(\sum_{k=1}^{36}e^{a_k})^2}=-y_jy_i \tag{3-3}$$

由单个样本的交叉熵式（3-4），对交叉熵函数求导为：

$$L=-\sum_N \hat{y}_i \log(y_i) \tag{3-4}$$

$$\frac{\partial L}{\partial o_i}=-\sum_k \hat{y}_k \frac{\partial \log(y_k)}{\partial o_k}=-\sum_k \hat{y}_k \frac{1}{y_k}*\frac{\partial y_k}{\partial o_i} \tag{3-5}$$

联合式（3-2）、（3-3）和（3-5），并根据本实验采用*one-hot*编码。

$$\frac{\partial L}{\partial o_i}=-\hat{y}_i(1-y_i)-\sum_{k\neq j}\hat{y}_k\frac{-(y_k y_i)}{y_k}=y_i(\hat{y}_i+\sum_{k\neq 1}\hat{y}_k)-\hat{y}_i=y_i-\hat{y}_i \tag{3-6}$$

根据梯度下降规则对权值系数和阈值进行调整：

$$w_{ik}=w_{ik}+alpha*(y_k-\hat{y}_k)*o_i \tag{3-7}$$

$$b_k=b_k+alpha*(y_k-\hat{y}_k) \tag{3-8}$$

其中*alpha*为梯度下降值。

在隐含层中选用*sigmoid*函数作为激活函数，对隐含层采用广义增量规则对权值系数和阈值进行调整，即：

$$w_{ij}=w_{ij}+alpha*\delta_i*x_j \tag{3-9}$$

式（3-9）中的δ_i的定义为：

$$\delta_i=\varphi'(x_i)*e_i \tag{3-10}$$

在本实验中采用*sigmoid*函数作为隐含层的激活函数，则有：

$$\delta_i=\varphi'(v_i)*e_i=\varphi(v_i)*[1-\varphi'(v_i)]*e_i \tag{3-11}$$

$$e_i=\sum_{k=1}^{36} w_{ik}*(y_k-\hat{y}_k) \tag{3-12}$$

综合以上式子可以得到隐含层的权值系数和阈值调整系数为：

$$w_{ij}=w_{ij}+alpha*\varphi(v_i)*[1-\varphi'(v_i)]*\sum_{k=1}^{36} w_{ik}*(y_k-\hat{y}_k) \tag{3-13}$$

$$b_i=b_i+alpha*\varphi(v_i)*[1-\varphi'(v_i)]*\sum_{k=1}^{36} w_{ik}*(y_k-\hat{y}_k) \tag{3-14}$$

3 训练神经网络

本实验共收录两位实验者的音频共55项，其中《小星星》15项，音阶共40项，选取正确率较高的《小星星》7项、音阶唱录25项作为拟训练音频，在这32项拟训练音频中经过分帧提取正确率较高的训练样本集共238个。将这238个训练样本进行特征提取，构造目标结果之后分别导入神经网络进行训练。

该神经网络以小批量的梯度下降方式对神经网络的权重系数及阈值进行调整，本实验设置每34个样本修改一次权重系数和阈值，以遍历整个238个训练样本为一轮，记录每一轮的总误差值，当总误差值小于等于0.01时，停止训练，并记录此时的权重系数及阈值用于后面实验的进行。

设置下降梯度初始系数为0.9，当轮数大于10000时下降梯度系数为0.8。本次实验训练完成共进行了31856轮训练，238个样本总误差值为0.01，符合预期的设定。

四、实验结果

为检验该神经网络的训练情况，使用非训练样本音频进行数据处理之后导入该神经网络进行测试，记录测试结果。

本文将以两段《小星星》音频作为测试样本对该神经网络进行测试，首先将音频进行数据处理，其包括带通滤波以及音频切分。计算音频段对应频率的方法计算测试音频的各段实际所对应的频率，按照就近原则将各音频段进行人工识别。再将切分完成的音频的各音频段提取70维输入特征，提取特征完成之后将其导入神经网络之后进行计算判断其对应的音名。比较人工识别及神经网络识别，以判别该神经网络的识别效果。

测试音频《小星星》1：

表4.1 测试音频1的实际频率及人工识别（单位:Hz）

音频段	实际频率	人工识别	音频段	实际频率	人工识别	音频段	实际频率	人工识别
1	281.446	C#4	**14**	**411.624**	**G#4**	**28**	**276.355**	**C#4**
2	**259.628**	**C4**	**15**	**403.624**	**G4**	**29**	**270.537**	**C#4**
3	**412.351**	**G#4**	16	374.534	F#4	**30**	**417.442**	**G#4**
4	410.897	G#4	**17**	**354.171**	**F4**	**31**	**415.26**	**G#4**
5	**444.35**	**A4**	**18**	**336.717**	**E4**	**32**	**461.077**	**A#4**
			19	**340.353**	**F4**	33	446.532	A4
6	**420.351**	**G#4**	20	304.718	D#4	34	395.624	G4
7	366.534	F#4	**21**	**410.897**	**G#4**	**35**	**359.262**	**F4**
8	361.444	F#4	22	399.988	G4	**36**	**360.716**	**F4**
9	**342.535**	**F4**	23	369.443	F#4	**37**	**342.535**	**F4**
10	**338.899**	**E4**	24	365.08	F#4	**38**	**338.172**	**E4**
11	**306.9**	**D#4**	**25**	**344.717**	**F4**	**39**	**302.536**	**D4**
12	**306.173**	**D#4**	**26**	**345.444**	**F4**	**40**	**306.9**	**D#4**
13	**272.719**	**C#4**	**27**	**309.809**	**D#4**	**41**	**274.901**	**C#4**

测试音频段共42个音频段，正确切分出41段，未正确切分以使用空白进行表示。其第三行是基于就近原则的人工识别，其数字按照表4.6进行对照。

将该音频输入神经网络接入如下表4.2所示。

根据两表的分类情况进行比较，人工分类与神经网络分类完全相符合的个数共30个，将其数据进行加粗表示，其比例占整首歌约73.17%。剩下11个音频段的分类的误差均在正负1，有4个属于高一个音分类错误，有7个属于高半音的分类错误，该歌曲整体情况属于可接受范围之内。根据测试音频2所示的两个表格可知，人工识别与神经网络识别完全相符合的个数为31个，其比例占整首歌约73.81%。11个分类错误的中，有1个属于高半音分类错误，有一个属于高一个音的分类错误，两个分类严重错误，其他均为高（低）八音的分类错误。根据推测分类出现高（低）八音的情况是由于实际的频率计算方式是通过遍历音频段的频

谱选择其最高的频谱能量的K值计算其对应的频率，但由于存在谐波因素的影响及声音受口腔发声的影响等问题，均可能导致实际频率高（低）八音的原因，因此在本实验中均认为高（低）八音均是由此造成，此分类属于正确。则整首歌的正确分类占比为90.47%。

表4.2 基于BP神经网络的测试音频段1的识别

音频段	分类音名	音频段	分类音名	音频段	分类音名	音频段	分类音名	音频段	分类音名	音频段	分类音名
1	C4	7	F4	**14**	**G#4**	**21**	**G#4**	**28**	**C#4**	**35**	**F4**
2	**C4**	8	F4	**15**	**G4**	22	G#4	**29**	**C#4**	**36**	**F4**
3	**G#4**	**9**	**F4**	16	G4	23	G4	**30**	**G#4**	**37**	**F4**
4	A4	**10**	**E4**	**17**	**F4**	24	G4	**31**	**G#4**	**38**	**E4**
5	**A4**	**11**	**D#4**	**18**	**E4**	**25**	**F4**	**32**	**A#4**	**39**	**D4**
		12	**D#4**	**19**	**F4**	**26**	**F4**	33	A#4	**40**	**D#4**
6	**G#4**	**13**	**C#4**	20	D4	**27**	**D#4**	34	G#4	**41**	**C#4**

测试音频《小星星》2：

表4.3 测试音频2的实际频率及人工分类（单位:Hz）

音频段	实际频率	人工识别	音频段	实际频率	人工识别	音频段	实际频率	人工识别
1	409.258	G#4	**15**	**315.486**	**D#4**	**29**	**385.418**	**G4**
2	412.437	G#4	**16**	**315.486**	**D#4**	**30**	**196.285**	**G3**
3	**306.745**	**D#4**	17	570.577	D5	**31**	**314.692**	**D#4**
4	**310.718**	**D#4**	**18**	**286.083**	**D4**	**32**	**313.102**	**D#4**
5	**344.889**	**F4**	**19**	**259.859**	**C4**	**33**	**353.631**	**F4**
6	**345.684**	**F4**	**20**	**260.654**	**C4**	**34**	**352.836**	**F4**
7	**306.745**	**D#4**	21	240.787	B3	**35**	**309.129**	**D#4**
8	**277.342**	**C#4**	**22**	**313.897**	**D#4**	**36**	**278.931**	**C#4**
9	**278.931**	**C#4**	**23**	**313.102**	**D#4**	**37**	**282.905**	**C#4**
10	**268.6**	**C4**	**24**	**286.083**	**D4**	38	526.87	C5
11	**267.806**	**C4**	25	561.041	C#5	**39**	**265.422**	**C4**
12	**242.376**	**B3**	**26**	**259.859**	**C4**	40	239.992	A#3
13	487.136	B4	**27**	**261.448**	**C4**	41	238.403	A#3
14	**216.947**	**A3**	28	233.635	A#3	42	212.178	G#3

表 4.4 基于 BP 神经网络的测试音频段 2 的分类

音频段	分类音名	音频段	分类音名	音频段	分类音名	音频段	分类音名	音频段	分类音名	音频段	分类音名
1	G#3	**8**	**C#4**	**15**	**D#4**	**22**	**D#4**	**29**	**G4**	**36**	**C#4**
2	G#3	**9**	**C#4**	**16**	**D#4**	**23**	**D#4**	**30**	**G3**	**37**	**C#4**
3	**D#4**	**10**	**C4**	17	C#4	**24**	**D4**	**31**	**D#4**	38	C4
4	**D#4**	**11**	**C4**	**18**	**D4**	25	C#4	**32**	**D#4**	**39**	**C4**
5	**F4**	**12**	**B3**	**19**	**C4**	**26**	**C4**	**33**	**F4**	40	B3
6	**F4**	13	B3	**20**	**C4**	**27**	**C4**	**34**	**F4**	41	A#4
7	**D#4**	**14**	**A3**	21	A#4	28	A#4	**35**	**D#4**	42	A3

根据这两首测试音频的测试结果观察，认为该神经网络属于训练完成且正确率较高。可以用于日常使用。输入音频通过该*BP*神经系统可以得到该音频对应的相应曲谱。将待测音频输入该系统，得到此音频曲谱（简谱）如下：

表 4.5 待测音频的输出简谱

序号	简谱	序号	简谱	序号	简谱	序号	简谱	序号	简谱	序号	简谱
1	1	**2**	1	**3**	5	**4**	5	**5**	6	**6**	6
7	5	**8**	4	**9**	4	**10**	3	**11**	3	**12**	2
13	2	**14**	1	**15**	5	**16**	5	**17**	#4	**18**	#4
19	3	**20**	3	**21**	2	**22**	5	**23**	5	**24**	4
25	4	**26**	3	**27**	3	**28**	2	**29**	1	**30**	1
31	5	**32**	5	**33**	6	**34**	#5	**35**	5	**36**	4
37	4	**38**	3	**39**	3	**40**	2	**41**	2	**42**	1

其表中1对应为C大调中的C4，其余2至7分别对应D4至B4,其带有“#”为对应音符升半音。

表4.6 音高与基频的对应表（单位：Hz）

音名	频率	音名	频率	音名	频率	音名	频率
A0	27.5	G2	97.9989	F4	349.228	D#6	1244.51
A#0	29.1352	G#2	103.826	F#4	369.994	E6	1318.51
B0	30.8677	A2	110	G4	391.995	F6	1396.91
C1	32.7032	A#2	116.541	G#4	415.305	F#6	1479.98
C#1	34.6478	B2	123.471	A4	440	G6	1567.98
D1	36.7081	C3	130.813	A#4	466.164	G#6	1661.22
D#1	38.8909	C#3	138.591	B4	493.883	A6	1760
E1	41.2034	D3	146.832	C5	523.251	A#6	1864.66
F1	43.6535	D#3	155.563	C#5	554.365	B6	1975.53
F#1	46.2493	E3	164.814	D5	587.33	C7	2093
G1	48.9994	F3	174.614	D#5	622.254	C#7	2217.46
G#1	51.9131	F#3	184.997	E5	659.255	D7	2349.32

A1	55	G3	195.998	F5	698.456	D#7	2489.02
A#1	58.2705	G#3	207.652	F#5	739.989	E7	2637.02
B1	61.7354	A3	220	G5	783.991	F7	2793.83
C2	65.4064	A#3	233.082	G#5	830.609	F#7	2959.96
C#2	69.2957	B3	246.942	A5	880	G7	3135.96
D2	73.4162	C4	261.626	A#5	932.328	G#7	3322.44
D#2	77.7817	C#4	277.183	B5	987.767	A7	3520
E2	82.4069	D4	293.665	C6	1046.5	A#7	3729.31
F2	87.3071	D#4	311.127	C#6	1108.73	B7	3951.07
F#2	92.4986	E4	329.628	D6	1174.66		

五、心得体会

通过这次实践的操作，我掌握了除本门学科所需的一些知识点外，利用一些课外的知识进行了实践。虽然在程序编写和调试过程中遇到了很多问题，但是最终还是在同学的帮助下，以及自己的坚持下完成了这个大作业。在编写过程中，由于很多知识点的欠缺，我都在一边写程序一边查询资料，也幸亏这次实验操作让我加深了对模式识别的了解，虽然在程序设计上没有使用到模式识别某些知识，但是在一开始的设计上也尝试了使用模式识别的方法。关于模式识别的知识点得到了巩固。

附录

1. 数据预处理

```
function [x] = yuchuli(y,fs)
y=double(y);
longx=length(y);


%带通滤波器(60hz-2400hz)
n=50;
wn=[2*60/fs,2*2400/fs];
window=chebwin(n+1,30);
b=fir1(n,wn,window);
x=filter(b,1,y);
end
```

2. 短时傅里叶

```
function d=stftms1of1(x,fs,nfft) %x是这一行的数据，,fs是采样频率，nfft是快速傅里叶变换的点数(归一化的短时傅里叶)
max = 0;
[~,xc]=size(x);
length = fix(1024*nfft/fs);
d = zeros(1,length);
win = hanning(xc);
u =win .* x(:);
t = fft(u , nfft);
t = abs(t);
d(1,:) =t(1:length); %取0~1024HZ的fft
for num = 1:length
      if d(1,num) > max
            max = d(1,num);
      end
end
for num = 1:length
       d(1,num) = d(1,num) / max;
end

end
```

3. 端点检测

```
%基于端点检测的切分
clear;
%分帧she
[y,fs] = audioread('D:\workplace\bishe\yinjie39.mp3');%读取音频
subplot 211,plot(y);title('音频原图');

y=double(y);
longx=length(y);

%带通滤波器（60hz-2400hz）
n=50;
wn=[2*60/fs,2*2400/fs];
window=chebwin(n+1,30);
b=fir1(n,wn,window);
x=filter(b,1,y);
%subplot 212,plot(x);title('音频预处理结果示意图');
%axis([-inf,inf, -0.4, 0.4]);

%预加重
%x=double(x);
%x=filter([1 -0.9375], 1, x);
%subplot 312,plot(x);title('音频预处理后的示意图');

win = 500;%45.5ms为一帧；帧长为500
inc = 200;%帧移
f = enframe(x,win,inc);
[fn,len]=size(f);

%find the Max of every frame
frameMax=zeros(fn+5,1);
num = 1; %frameMax的指标
for row = 1:fn
    for column = 1:len
        if f(row,column) > frameMax(num)
            frameMax(num) = f(row,column);
        end
    end
    num= num + 1;
end


%smooth frameMax
for num = 1:fn
```

```
frameMax(num)=frameMax(num)+frameMax(num+1)+frameMax(num+2)+frameMax(num+3)+frameMax(num+
4);
end

%do the difference
diff = zeros(fn,1);
diff(1) = frameMax(1);
diffMax =diff(1);
for num =2:fn
    diff(num) = frameMax(num) - frameMax(num-1);
    if diff(num) > diffMax
        diffMax = diff(num);
    end
end
for num =1:fn
    diff(num)=diff(num)/diffMax;
end



%find the endpoint
endpoint = zeros(length(x),1); %端点在原图中的位置
endpoint1 = zeros(length(x),1); %端点在原图中的位置
endpointframe =zeros(fn,1); %端点在帧数中的位置
num = 1;
count = 1;
distance = 0;
while(num < fn)
    %distance = distance + 1;
    if diff(num) > 0.11 || diff(num) < -0.45
        %if distance >60
            endpoint((num-1)*inc,1) = 0.25;
            endpoint1((num-1)*inc,1) = -0.25;
            endpointframe(num-1) = 1;
            num= num + 125;
            count = count + 1; %计数起始点的数目。
            distance = 0;
        %end
    end
    num = num + 1;
end

%calculation the localtion of every endpoint in x
endpointlocal = zeros(count,1);
```

```
count = 1;
for num = 1:length(x)
    if endpoint(num) == 0.25
        endpointlocal(count) = num;
        count = count + 1;
    end
    if num == length(x)
        %if num - endpointlocal(t-1) >20000
            endpointlocal(count) = num;
        %end
    end

end
CountNum = count - 1; %最后分了多少帧

%build a new array to storage the every endpoint data
diff1Max = 0;
for num = 2:length(endpointlocal)
    if (endpointlocal(num) - endpointlocal(num-1)) >diff1Max
        diff1Max = (endpointlocal(num) - endpointlocal(num-1));
    end
end
%frameing array
framelength = diff1Max + 1;   %帧的最大实际长度
if mod(framelength,2) == 1     %如果帧长为奇数，将其+1变成偶数方便fft
    framelength =framelength + 1;
end

framearray = zeros(length(endpointlocal)-1,framelength);
row = 1;
column =2;
framearray(row,1) = x(endpointlocal(1));

for num = endpointlocal(1)+1:length(x)
    if endpoint(num) == 0
        framearray(row,column) = x(num);
        column = column + 1;
    else
        row = row + 1;
        column = 1;
        framearray(row,column) = x(num);
        column = column + 1;
    end
end
```

```
%plot(diff);hold on ;plot(endpointframe);

subplot 211;plot(x);hold on;plot(endpoint,'b');
subplot 212;plot(framearray(2,:));

%到这个点为止，做完基于端点检测的分帧


%将每一帧进行傅里叶变换得到st矩阵，每一行是每一帧的傅里叶变换

for num = 1 : CountNum
    every_frame=framearray(num,:);
    st(num,:)=stftms1(every_frame,fs,framelength);
end
%energy = Extraction_parameter1(every_frame,fs,framelength);

%提取参数Extraction parameter
for num = 1 : CountNum
    every_frame=st(num,:);
    %energy(: ,num) = Extraction_parameter1(every_frame,fs,framelength);
    energy (:,num)= Extraction_parameterof70(every_frame,fs,framelength);
end
canshu = zeros(CountNum,1);
for num = 1 : CountNum
    for i = 1:70
        if energy (i,num) == 1
            canshu(num,1) = i;
        end
    end
end
```

4. 特征提取

```
function energy = Extraction_parameterof70(x,fs,nfft) %x是每一帧0~1024HZ对应的nfft,nfft,fs,用于求相对应的K
nx = length(x);
energy = zeros(70,1);
parameter = zeros(70,1);
F = [128,134;136,142;144,150;153,159;162,168;172,178;182,188;193,199;205,211;217,223;...
     230,236;244,250;259,265;274,280;291,297;308,314;327,333;346,352;356,362;367,373;...
     378,384;389,395;400,406;412,418;424,430;437,443;450,456;463,469;477,483;491,497;...
     505,511;520,526;530,536;541,547;551,557;562,568;573,579;584,590;595,601;608,614;...
     619,625;631,637;644,650;656,665;669,675;682,688;695,701;709,715;723,729;737,743;...
     748,754;759,765;770,776;781,787;793,799;805,811;816,822;828,834;840,846;852,858;...
     864,870;877,883;890,896;903,909;916,922;929,935;943,949;957,963;971,977;985,991];
```

```
F = fix(F .* (nfft/fs));
K = zeros (70,20);
for i = 1 : 70
    num = 1;
    for j =F(i,1) : F(i,2)
        K(i,num) = j;
        num = num + 1;
    end
end
for num = 1 : nx
    switch num
        case
{K(1,1),K(1,2),K(1,3),K(1,4),K(1,5),K(1,6),K(1,7),K(1,8),K(1,9),K(1,10),K(1,11),K(1,12),K(1,13),K(1,14),K(1,15),K(1,16),K(1,17),K(1,18),K(1,19),K(1,20)}
            parameter(1) = parameter(1) + x(1,num);
        case
{K(2,1),K(2,2),K(2,3),K(2,4),K(2,5),K(2,6),K(2,7),K(2,8),K(2,9),K(2,10),K(2,11),K(2,12),K(2,13),K(2,14),K(2,15),K(2,16),K(2,17),K(2,18),K(2,19),K(2,20)}
            parameter(2) = parameter(2) + x(1,num);
        case
{K(3,1),K(3,2),K(3,3),K(3,4),K(3,5),K(3,6),K(3,7),K(3,8),K(3,9),K(3,10),K(3,11),K(3,12),K(3,13),K(3,14),K(3,15),K(3,16),K(3,17),K(3,18),K(3,19),K(3,20)}
            parameter(3) = parameter(3) + x(1,num);

        …

    end
end
parameterMax = max(parameter);
energy =    parameter./parameterMax;
end
```

5. 参数训练

```
function [W1,W2,B1,B2,E] = MultiClass1(W1,W2,X,D,B1,B2,epoch) %w1是sigmoid的系数,w2是softmax的系数,x训练样本,d训练样本标签,epoch轮数,B1是sigmoid阈值,B2是softmax阈值
alpha = 0.9 %梯度下降系数

if epoch >    10000
    alpha = 0.8;
end

if epoch > 50000
    alpha = 0.6;
```

```
end

if epoch > 100000
    alpha = 0.4;
end

if epoch > 250000
    alpha = 0.2;
end
if epoch > 500000
    alpha = 0.1;
end

N = 238; %训练样本数
dw1 = zeros(150,70);%sigmoid的阈值
dw2 = zeros(36,150);%softmax的阈值
db1 = zeros(150,1);
db2 = zeros(36,1);
count = 0 ;
E = 0;
for k = 1 : N
    x = X(:,k);
    d = D(:,k);

    v1 = (W1 * x) + B1;
    y1 = Sigmoid(v1);
    v  = (W2 * y1) + B2;
    y  = Softmax(v);

    count = count + 1;

    e = d - y; %输出误差
    E = E + sum(abs(e));
    delta = e; %softmax激活函数的δ = φ'(v) * e
    e1 = W2' * delta; %sigmoid函数的输出误差
    delta1 = y1 .* (1 - y1) .* e1;%sigmoid函数的δ


    dw1 = dw1 + alpha * delta1 * x';
    dw2 = dw2 + alpha * delta * y1';

    db1 = db1 + alpha .* delta1;
    db2 = db2 + alpha .* delta;
```

```
    if count == 34
        dw1 = dw1 ./ count;
        dw2 = dw2 ./ count;
        db1 = db1 ./ count;
        db2 = db2 ./ count;

        W1 = W1 + dw1;
        W2 = W2 + dw2;
        B1 = B1 + db1;
        B2 = B2 + db2;

        dw1 = zeros(150,70);
        dw2 = zeros(36,150);
        db1 = zeros(150,1);
        db2 = zeros(36,1);
        count = 0;
    end
end
end
```