第5章 chapter 5

基于深度信念网络的轴承故障诊断方法

作为第二类深度学习方法——深度信念网络，是一种采用逐层训练方式训练的深度神经网络。本章主要提出一种基于深度信念网络的轴承故障诊断方法，具体包括基于深度信念网络的轴承故障诊断网络结构与故障诊断建模机理；基于深度信念网络的轴承故障诊断模型构建流程与构建算法；结合美国凯斯西储大学(CWRU)提供的轴承数据集，完成基于深度信念网络的轴承故障诊断模型实验。

5.1 基于深度信念网络故障诊断工作原理

5.1.1 基于深度信念网络的轴承故障诊断网络结构

基于深度信念网络 DBN 的轴承故障诊断网络结构如图 5.1 所示，它是具有许多隐藏层的前馈神经网络，这些隐藏层可以使 DBN 对数据中的复杂关系具有更强大的建模能力。

基于 DBN 的轴承故障诊断网络结构的输入层对应轴承振动信号的输入，中间是多个受限玻尔兹曼机 RBM 构成的隐藏层，输出层对应故障诊断分类结果输出。从输入层到输出分类层，每两个相邻层形成一个 RBM。

5.1.2 基于深度信念网络的轴承故障诊断建模机理

基于 DBN 的轴承故障诊断建模过程包括 DBN 无监督学习和 DBN 有

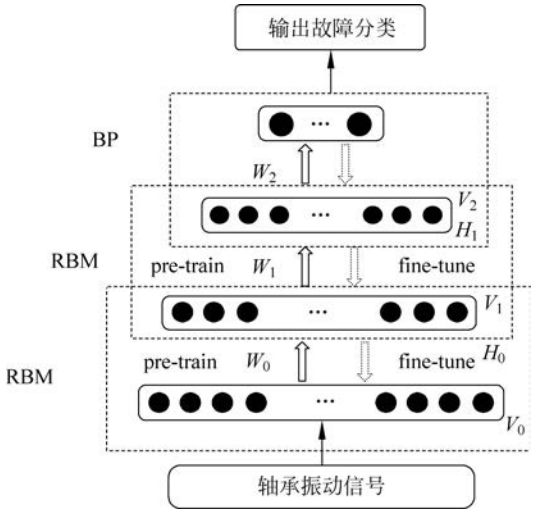

图 5.1　基于 DBN 的轴承故障诊断网络结构

监督反向调参两个阶段。

1. DBN 无监督学习训练过程

在这个训练阶段，通过一个非监督贪婪逐层训练方法去预训练模型的权值。首先把数据向量 $\boldsymbol{x}$ 和第一层隐藏层 h_1 作为一个 RBM，使用 CD-K 算法训练出这个 RBM 的参数（连接 x 和 h_1 的权重，$\boldsymbol{x}$ 和 h_1 各个节点的偏置等），然后固定这个 RBM 的参数，把 h_1 视作可见向量，把 h_2 视作隐藏向量，训练第二个 RBM，得到其参数，然后固定这些参数。接着训练更高层的 RBM。

RBM 是两层神经网络。数据输入层或可见层由可见单元 $\boldsymbol{v}=(v_1,v_2,v_3,\cdots,v_n)$ 组成，隐藏层由隐藏单元 $\boldsymbol{h}=(h_1,h_2,h_3,\cdots,h_m)$ 组成，通过权重矩阵 $\boldsymbol{w}_{n*m}$ 连接每个可见单元和每个隐藏单元。

可见单元和隐藏单元的联合分布符合玻尔兹曼分布，通过能量函数 $E(V,H)$ 的概率密度为

$$P(V,H)=\frac{1}{Z}\mathrm{e}^{-E(V,H)} \tag{5.1}$$

$$Z=\sum_{V,H}\mathrm{e}^{-E(V,H)} \tag{5.2}$$

其能量函数 $E(V,H)$ 为

$$E(V,H)=-\sum_{i=1}^{n}a_i v_i-\sum_{j=1}^{m}b_j h_j-\sum_{i=1}^{n}\sum_{j=1}^{m}v_i w_{ij}h_j \tag{5.3}$$

其中 v_i 和 h_j 分别表示可见层中的第 i 个神经元和隐藏层中的第 j 个神经元的状态，其中 a_i 和 b_j 是它们的偏差，而 $w_{ij}=w_{ji}$ 是第 i 个神经元和第 j 个神经元之间的双向权重。

在一个 RBM 中，隐藏神经元 h_j 被激活的概率如下：

$$P(h_j \mid v)=\sigma(a_j+\sum_{i=1}^{n}w_{ij}v_i) \tag{5.4}$$

在一个 RBM 中，隐藏神经元 v_i 被激活的概率如下：

$$P(v_i \mid h)=\sigma(b_i+\sum_{j=1}^{m}w_{ij}h_j) \tag{5.5}$$

其中，σ 为 Sigmoid 函数：$\sigma(x)=1/(1+e^{-x})$，也可以设定为其他函数。

RBM 的训练过程实际上是求出一个最能产生训练样本的概率分布。也就是说，要求一个分布，在这个分布里，训练样本的概率最大。由于这个分布的决定性因素在于参数 w、b，所以我们训练 RBM 的目标就是寻找最佳的权值。

RBM 的权值连接是单向的，因此 RBM 无需监督即可从输入数据中学习特征信息。训练的过程是通过调整权重和偏差来增加得到输入数据 $P(V)$的可能性。对比散度算法(Contrastive Divergence，CD)是训练 RBM 的一个快速学习算法。它只需要 k 步 Gibbs(吉布斯)采样(简记为 CD-K)就可以得到一个足够好的近似模型，甚至当 $K=1$ 时就可以达到很好的训练效果。

CD-K 算法训练过程为，在可视层会产生一个向量 $\boldsymbol{v}$，通过它将值传递到隐藏层。反过来，可视层的输入会被随机的选择，以尝试去重构原始的输入信号。最后，这些新的可视的神经元激活单元将前向传递重构隐藏层激活单元，获得 $\boldsymbol{h}$(在训练过程中，首先将可视向量值映射给隐单元；然后可视单元由隐藏层单元重建；这些新可视单元再次映射给隐单元，这样就获取新的隐单元。执行这种反复步骤称为 Gibbs 采样)。而隐藏层激活单元和可视层输入之间的相关性差别就作为权值更新的主要依据。各个参数更新策略的方法，使用如下的式子：

$$w_{ij}^{\text{epoch}+1}=w_{ij}^{\text{epoch}}+\alpha\text{CD} \tag{5.6}$$

$$a_i^{\text{epoch}+1}=a_i^{\text{epoch}}+\alpha\frac{1}{m}\sum_{i=1}^{m}(h_i^{\text{epoch}-1}-h_i^{\text{epoch}}) \tag{5.7}$$

$$b_i^{\text{epoch}+1}=b_i^{\text{epoch}}+\alpha\frac{1}{n}\sum_{i=1}^{n}(v_i^{\text{epoch}-1}-v_i^{\text{epoch}}) \tag{5.8}$$

其中，α 为学习率。对比散度 CD=$<h_i^{\text{epoch}-1}v_j^{\text{epoch}-1}>-<h_i^{\text{epoch}}v_j^{\text{epoch}}>$，$<\ >$表示分布的平均。

2. DBN 有监督反向调参训练过程

在无监督学习之后，可以获得特征模型。在 DBN 的最后一层设置 BP 网络，接收 RBM 的输出特征向量作为它的输入特征向量，有监督地训练实体关系分类器。而且每层 RBM 网络只能确保自身层内的权值对该层特征向量映射达到最优，并不是对整个 DBN 的特征向量映射达到最优，所以反向传播网络还将错误信息自顶向下传播至每层 RBM，微调整个 DBN 网络。RBM 网络训练模型的过程可以看作对一个深层 BP 网络权值参数的初始化，使 DBN 克服了 BP 网络因随机初始化权值参数而容易陷入局部最优和训练时间长的缺点。

通常用损失函数 $L(Y,f(x))$来评估模型好坏程度，即预测值 $f(x)$与真实值的不一致程度，损失函数的值越小，模型的鲁棒性也就越好，对新数据的预测能力也就越强。传统的损失函数有：均方误差损失函数(Mean Squared Error Loss Function)、交叉熵损失函数(Cross-Entropy Loss Function)等。

均方误差损失函数的基本形式如下：

$$L(Y,f(x))=\frac{1}{D_N}\sum_{i=1}^{D_N}(h_i^N-Y_i)^2 \tag{5.9}$$

交叉熵损失函数的基本形式如下：

$$L(Y,f(x))=-\frac{1}{D_N}\sum_{i=1}^{D_N}Y_i\ln h_i^N \tag{5.10}$$

其中，Y 是目标，D_N 是输出层的节点个数，h^N 是模型的输出。

当使用 Sigmoid 作为激活函数时，常用交叉熵损失函数而不用均方误差损失函数，因为它可以完美解决平方损失函数权重更新过慢的问题，具有“误差大的时候，权重更新快；误差小的时候，权重更新慢”的良好性质。

目前的 DBN 模型使用基于梯度的学习方法进行模型的训练，而在训练过程中也主要是通过计算损失函数的梯度来更新网络参数。常用的梯度的学习方法有：随机梯度下降法(Stochastic Gradient Descent，SGD)、Momentum 标

准动量优化方法、NAG(Nesterov Accelerated Gradient)牛顿加速梯度动量优化方法、Adam 自适应学习率优化算法等。

随机梯度下降法(SGD)：更新模型参数的表达式如下：

$$W_{t+1}=W_t-\eta_t g_t \tag{5.11}$$

其中，学习率为 η_t；$g_t=\nabla L(W_t;X^{(i_s)};Y^{(i_s)})$，表示一个随机的梯度方向，$i_s\in\{1,2,\cdots,n\}$，$X^{(i_s)}$ 从训练集中取出的一个大小为 n 的小批量$\{X^{(1)},X^{(2)},\cdots,X^{(n)}\}$样本，对应的真实值分别为 $Y^{(i_s)}$；W_t 表示 t 时刻的模型参数。

Momentum 标准动量优化方法：更新模型参数的表达式如下：

$$v_t=\alpha v_{t-1}+\eta_t\nabla L(W_t;X^{(i_s)};Y^{(i_s)}) \tag{5.12}$$

$$W_{t+1}=W_t-v_t \tag{5.13}$$

其中，v_t 表示 t 时刻积攒的速度；α 表示动力的大小，一般取值为 0.9；$\Delta L(W_t;X^{(i_s)};Y^{(i_s)})$含义同 SGD 算法；$W_t$ 表示 t 时刻模型参数。

NAG 牛顿加速梯度动量优化方法：更新模型参数的表达式如下：

$$v_t=\alpha v_{t-1}+\eta_t\nabla L(W_t-\alpha v_{t-1}) \tag{5.14}$$

$$W_{t+1}=W_t-v_t \tag{5.15}$$

其中，v_t 表示 t 时刻积攒的速度；α 表示动力的大小；η_t 表示学习率；W_t 表示 t 时刻的模型参数；$\nabla L(W_t-\alpha v_{t-1})$表示损失函数关于 W_t 的梯度。

5.2 基于深度信念网络的轴承故障诊断模型构建

5.2.1 基于深度信念网络的轴承故障诊断模型构建流程

基于深度信念网络 DBN 的轴承故障诊断模型构建流程如图 5.2 所示。

5.2.2 基于深度信念网络的轴承故障诊断模型构建算法

基于深度信念网络 DBN 的轴承故障诊断模型构建算法如下所示，整个算法包括使用贪心无监督方法和逐层基于反向传播的监督学习两部分组成。

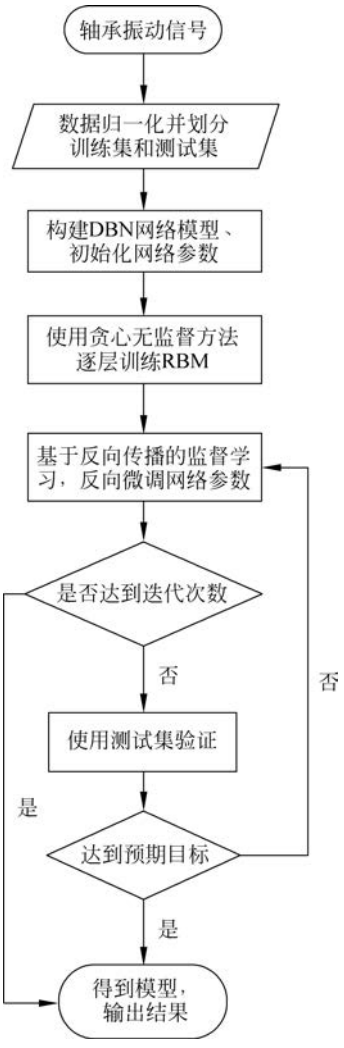

图 5.2　轴承故障诊断的 DBN 模型流程图

算法 5.1　基于 DBN 的轴承故障诊断模型建模算法

输入：轴承振动信号数据集 X

隐藏层 h，层数 N，迭代次数 Q_1，Q_2

参数空间 $\boldsymbol{w}$，偏置 a、b，学习率 α

标注数据的个数 L，未标注数据的个数 U

输出：包含训练后参数空间的 DBN 网络

方法：

使用贪婪无监督方法逐层构建网络

for k=1;k<N do

for i=1;i<Q_1 do

for j=1;j<U do

计算非线性正向和反向状态

$$P(h_j \mid v) = \sigma\left(a_j + \sum_{i=1}^{n} w_{ij} v_i\right)$$

$$P(v_i \mid h) = \sigma\left(b_i + \sum_{j=1}^{m} w_{ij} h_j\right)$$

CD 算法更新权重和偏置：

$$w_{ij}^{\text{epoch}+1} = w_{ij}^{\text{epoch}} + \alpha \text{CD}$$

$$a_i^{\text{epoch}+1} = a_i^{\text{epoch}} + \alpha \frac{1}{m} \sum_{i=1}^{m} (h_i^{\text{epoch}-1} - h_i^{\text{epoch}})$$

$$b_i^{\text{epoch}+1} = b_i^{\text{epoch}} + \alpha \frac{1}{n} \sum_{i=1}^{n} (v_i^{\text{epoch}-1} - v_i^{\text{epoch}})$$

基于反向传播的监督学习

计算代价函数并反向微调网络参数以寻找模型最优解

for i=1;i<Q_2 do

构建损失函数：$L(Y, f(x)) = \frac{1}{D_N} \sum_{i=1}^{D_N} (h_i^N - Y_i)^2$

目标函数：$\theta^* = \text{argmin}_\theta L(f(X;\theta))$

使用 NAG 优化器算法更新权重：

$$v_t = \alpha v_{t-1} + \eta_t \nabla L(W_t, -\alpha v_{t-1})$$

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \boldsymbol{v}_t$$

5.3 基于深度信念网络的轴承故障诊断模型实验

5.3.1 基于深度信念网络的轴承故障诊断数据源

轴承故障诊断模型实验数据来自美国凯斯西储大学(CWRU)提供的轴承数据集。本次实验对象为驱动端轴承，轴承型号为 SKF6205，轴承故障由电火花加工制作而成，系统的采样频率为 12kHz 以及转速 1797r/min。被诊断的轴承一共有 3 种缺陷位置，分别是滚动体损伤，外圈损伤与内圈损伤，损伤直径的大小分别为 0.007 英寸，0.014 英寸和 0.021 英寸，共计 9 种

损伤状态。轴承的一个旋转周期对应的采样点数为400，使用包含2个完整周期的800个数据点进行诊断实验。为了便于提高训练速度和分类精度，对每段信号均做标准归一化处理，标准归一化处理如式(5.16)所示。

$$X=(X-\text{mean})/\text{std} \tag{5.16}$$

式中X为信号数据集，mean为其均值，std为其标准差。经过处理后数据符合标准正态分布，即均值为0，标准差为1。

本实验用到3组数据集A、B、C，分别是在负载为1HP、2HP和3HP下的数据集。每组数据集均包含以下故障数据划分，如表5.1所示。每个数据集分为80%的训练样本与20%的测试样本。

表5.1　实验对象的每组数据集分类

损伤	无	滚　动　体			内　　圈			外　　圈		
标签	1	2	3	4	5	6	7	8	9	10
损伤直径(in)	0	0.007	0.014	0.021	0.007	0.014	0.021	0.007	0.014	0.021

输入端节点数为800，输出端为10。学习因子设置为0.01，使用Dropout抑制过拟合，Dropout率为0.5，每个RBM的预训练次数为2。使用NAG动量优化方法加速训练过程，动量因子设置为0.9。对于数据集A、B和C，训练次数上限20000次。由于神经网络的初值是随机的，为了验证每次训练结果的可靠性，对于每个数据集，均训练10次。

5.3.2　基于深度信念网络的轴承故障诊断模型构建实验

本实验是基于Python语言，采用以TensorFlow为底层代码实现，计算机硬件基本配置为i7-8750H处理器，8GB内存，Windows系统。

1. 不同网络深度下模型对比实验

我们按DNN网络隐藏层的个数，分别选择2层、3层、4层和5层的网络模型进行实验。实验数据使用数据集A、B、C，训练集1575组，测试集390组，不同深度DBN的实验结果如表5.2～表5.4所示。

表5.2　数据集A下不同深度的DBN的实验结果

深度	2层	3层	4层	5层
正确率	84.65%	99.91%	98.46%	95.89%

表 5.3 数据集 B 下不同深度的 DBN 的实验结果

深度	2 层	3 层	4 层	5 层
正确率	79.90%	100%	99.85%	93.76%

表 5.4 数据集 C 下不同深度的 DBN 的实验结果

深度	2 层	3 层	4 层	5 层
正确率	74.42%	100%	99.04%	95.64%

从表 5.2 中可以看出，在网络层数较少为 2 层时，模型的分类性能较差且在不同的数据集上效果差异较大，如数据集 A 下的分类正确率比数据集 C 下的分类正确率多了大约 10%；网络层数增加后，训练的时间逐渐增加，网络的性能有所提升，但层数增加到一定数量，性能反而有所下降。这主要是由于隐藏层的增加造成误差在反向传播的过程中累加过大导致。通过实验得到层数为 3 时 DBN 的分类性能最佳，此时的准确率在 3 个数据集 A、B、C 分别达到了 99.91%、100%和 100%。

2. 不同小批量下模型对比实验

模型训练过程中，当 mini-batch 中包含的样本数目越少，其均值的波动范围也就越大，也就越满足给训练模型的均值方差带来干扰的要求。此外，为了使均值方差不出现与整体训练样本过大的偏离，应该保证 mini-batch 的值要大于需要分类的数量，否则 mini-batch 在训练时的均值方差将始终偏离整体的均值方差，不利于模型的训练。经过测试分析得知较大的 mini-batch 不适合用来训练模型，学习效率慢，且会出现欠拟合；而较小的 mini-batch 可以提高学习速度，但容易出现过拟合。

结合大小不同的 mini-batch，在数据集 A 上进行模型实验，实验结果如图 5.3 所示。

对应不同的 mini-batch，最终分类准确率依次分别为，0.996153846、0.997720798、0.997008547、0.998974359、0.913461538、0.906153846。从图 5.3 中可以看出，在 mini-batch 的值大于 100 时，出现了欠拟合，且波动较大。在 mini-batch 的值小于 100 时，整体分类准确率较好，多次可以达到 100%的分类准确率，但 mini-batch 的值过小，同样会出现轻微的过拟合并伴随一定的波动。在 mini-batch 的值为 100 时，可以获得稳定的高分类准确率。

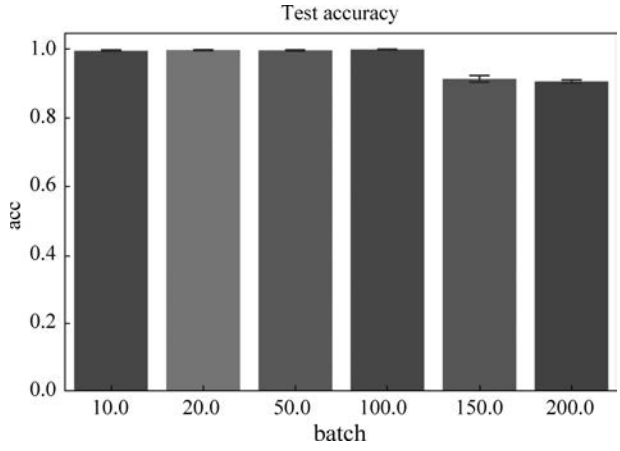

图 5.3　不同大小的 mini-batch 的平均分类准确率

对比不同 mini-batch 下最接近均值的训练过程如图 5.4 所示，在 15000 次内均达到了收敛点。随着 mini-batch 的增大训练速度下降、训练前期波动较大，在 mini-batch 大于 100 时最终分类准确率很低。

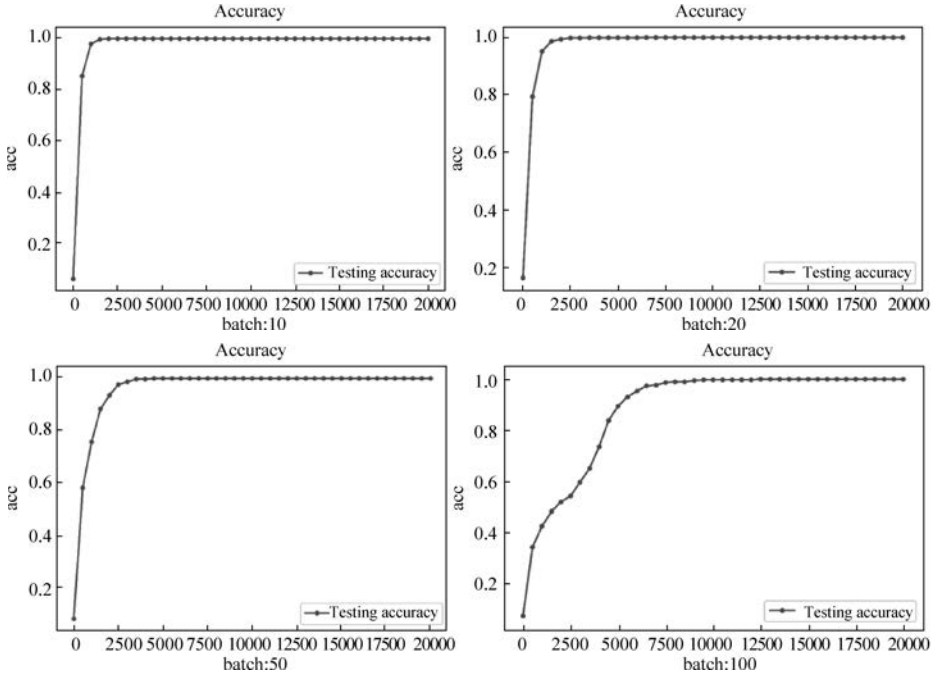

图 5.4　不同 mini-batch 下分类准确率随迭代次数的变化

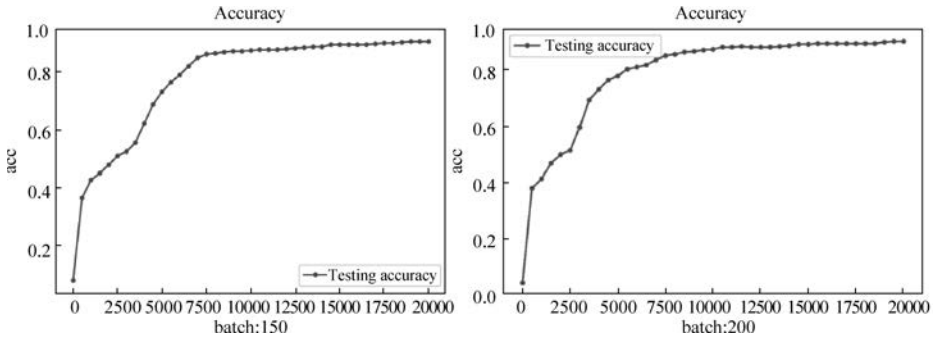

图 5.4 （续）

从图 5.5 我们可以看出，不同的 mini-batch 下，模型平均迭代 10000 次所用时间随着 mini-batch 的增大而减小，这意味着在保证分类准确率的前提下，增大 mini-batch 可以提高训练速度。所以我们选择使用的 mini-batch 的值为 100。

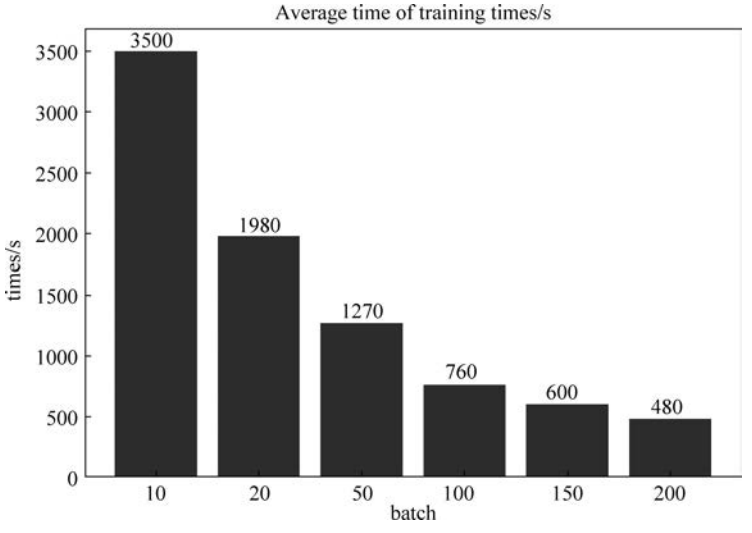

图 5.5 不同大小的 mini-batch 平均迭代 10000 次所用时间

3. 不同数据规模大小下模型对比实验

为了测试在不同大小训练集下的结果，我们对原始数据集进行重叠采样的数据增强。在实验中针对数据集 A 分别使用样本总量为 521，784，1575，3156，6318 和 12644 的训练样本训练 DBN 模型，观察数据集增强技术对 DBN 诊断能力的影响。由于神经网络的权值初值是随机生成的，为了验

证 DBN 的稳定性，每个试验重复进行 20 次，试验结果如表 5.5 所示。

表 5.5　不同大小训练集下的分类准确率

样本集大小	分类准确率	样本集大小	分类准确率
521	95.00%	3156	99.87%
784	93.33%	6318	99.51%
1575	99.90%	12644	99.50%

由表 5.5 可以看出该模型在使用较少训练数据的情况下，也能达到很高的识别率。当使用样本总量大于或等于原始数据集大小 1575 时，分类准确率均大于 99.50%，表明诊断模型已具有较高的稳定性，说明该模型具有较强的学习能力。

5.4　一种基于 MCELF 的 DBN 轴承故障诊断加速方法

针对深度信念网络故障诊断模型训练时间过长，效率低下等问题，我们提出一种基于多级复合指数损失函数(Multi-Composite Exponential Loss Function，MCELF)的深度信念网络轴承故障诊断加速方法，该方法通过 MCELF 来放大代价函数，进而放大梯度达到故障诊断模型训练加速目的。

5.4.1　基于 MCELF 的故障诊断加速方法

由于 DBN 模型通常采用基于梯度的方法来进行模型的训练，因此可以通过改变损失函数来加快 DBN 模型的学习效率。我们通过定义多级复合指数的方法来放大损失函数，进而放大梯度，多级复合指数损失函数(MCELF)公式如下：

$$L' = k\overbrace{\exp(\exp(\cdots\exp(L)))}^{n} \tag{5.17}$$

式中，k 为放大倍数，n 为指数复合级数。

引入 MCELF 后，其对应代价函数梯度计算如下：

$$\begin{aligned}\frac{\partial J'(\theta)}{\partial\theta} &= \frac{k}{n}\sum_{i=1}^{n}\frac{\partial L'}{\partial L}\frac{\partial L}{\partial f}\frac{\partial f}{\partial\theta} \\ &= \frac{k}{n}\sum_{i=1}^{n}\overbrace{\exp(\exp(\cdots\exp(L)))}^{n}\cdots\exp(\exp(L))\exp(L)\frac{\partial L}{\partial f}\frac{\partial f}{\partial\theta}\end{aligned} \tag{5.18}$$

我们通过这种方式将原有梯度放大了 $k\overbrace{\exp(\exp(\cdots\exp(L)))}^{n}\cdots\exp(L)$ 倍，且保持原有损失函数基本性质，其在损失函数由大到小的训练过程中仍具有“误差大的时候，梯度大；误差小的时候，梯度小”的良好性质。

5.4.2 基于 MCELF 的加速 DBN 故障诊断的模型构建算法

基于 MCELF 的 DBN 故障诊断的模型构建算法如下所示。

算法 5.2 基于 MCELF 的 DBN 故障诊断的模型构建算法

输入：数据集 X，标签集 Y

隐藏层 h，层数 N，每一层的单元个数 $D_1, D_2, \cdots, D_N$，迭代次数 Q_1, Q_2

参数空间 $W=\{w_1, w_2, \cdots, w_N\}$，偏置 a、b，优化器动量因子 ϑ，学习率 α

标注数据的个数 L，未标注数据的个数 U

输出：包含训练后参数空间的 DBN 网络模型

方法：

1. 使用贪婪无监督方法逐层构建网络

for k=1;k<N do

for i=1;i<Q_1 do

for j=1;j<U do

计算非线性正向和反向状态

$$P(h_j \mid v)=\sigma\left(a_j+\sum_{i=1}^{n} w_{ij} v_i\right)$$

$$P(v_i \mid h)=\sigma\left(b_i+\sum_{j=1}^{m} w_{ij} h_j\right)$$

CD 算法更新权重和偏置

$$w_{ij}^{\text{epoch}+1}=w_{ij}^{\text{epoch}}+\alpha \text{CD}$$

$$a_i^{\text{epoch}+1}=a_i^{\text{epoch}}+\alpha\frac{1}{m}\sum_{i=1}^{m}(h_i^{\text{epoch}-1}-h_i^{\text{epoch}})$$

$$b_i^{\text{epoch}+1}=b_i^{\text{epoch}}+\alpha\frac{1}{n}\sum_{i=1}^{n}(v_i^{\text{epoch}-1}-v_i^{\text{epoch}})$$

2. 基于反向传播的监督学习

计算代价函数并反向微调网络参数以寻找模型最优解

for i=1;i<Q_2 do

损失函数：$L'=k\sum_{n}^{k}\overbrace{\exp(\exp(\cdots\exp(L)))}^{n}$

目标函数：$\min J'(\theta)=\frac{1}{n}\sum_{i=1}^{n}L'(y_i,f(x_i;\theta))$

使用优化器更新权重

5.4.3　基于 MCELF 的加速 DBN 故障诊断的模型实验

继续使用 5.3 节中的 CWRU 数据集，以及其 DBN 故障诊断模型，输入端节点数为 800，两层隐藏层节点数为 500、500，输出端为 10。小批量 mini-batch 的大小设置为 50，学习因子设置为 0.01，每个 RBM 的预训练次数为 2。使用 NAG 动量优化方法加速训练过程，动量因子设置为 0.9。由于神经网络的初值是随机的，为了验证每次训练结果的可靠性，对于每个数据集，多次测试，取其最接近平均值的单次训练作对比。

我们使用 MSE 和 Cross-Entropy 两种损失函数在 DBN 模型上做实验，首先在 NAG 优化器下分别对 MCELF 的参数 n 和 k 进行实验验证其加速效果，然后再验证其在不同的优化器下的加速效果。

1. 不同级 MCELF 下模型加速对比实验

首先使用 $n=1$，即 1 级复合指数损失函数在两种损失函数即 1-MSE 和 1-Cross-Entropy 做实验。为了更好地观察加速效果，我们在图中以实际的第 50 次迭代结果开始展示，对比加速前后实验结果，如图 5.6 和图 5.7 所示：

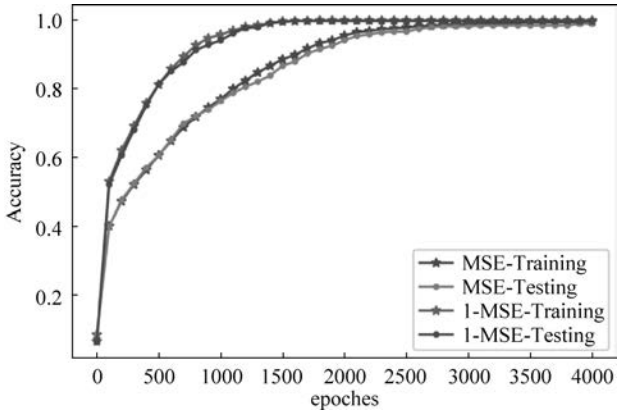

图 5.6　对比 1 级 MSE 损失函数加速前后效果

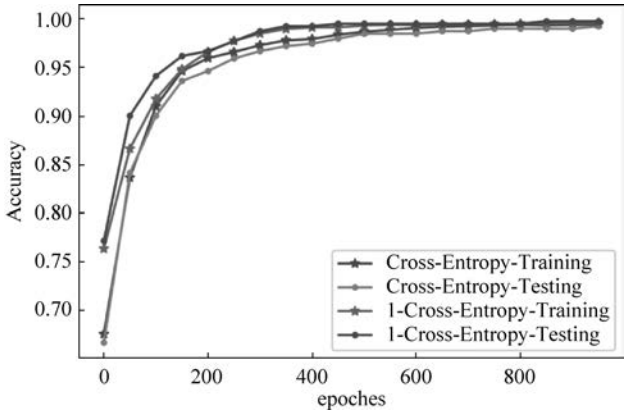

图 5.7　对比 1 级 Cross-Entropy 损失函数加速前后效果

使用 1-MSE 的训练集在大约 2000 次迭代时达到最终分类准确率为 0.9994、测试集最终分类准确率为 0.9974，使用 1-Cross-Entropy 的训练集大约在 900 次迭代时达到最终分类准确率为 0.9958、测试集最终分类准确率为 0.9974。结合实验结果和训练过程图可以看出，使用改进后的损失函数不仅可以加快模型的训练速度，更快达到稳定点，而且还略微提高了分类准确率。并且加速后的模型在整个训练过程中训练集和测试集的差值很小，说明该方法也具有良好的泛化能力。

接着使用 $n=2$，即 2 级复合指数损失函数在两种损失函数即 2-MSE 和 2-Cross-Entropy 做实验。为了更好地观察加速效果，我们在图中以实际的第 50 次迭代结果开始展示，对比不同级别加速前后实验结果，如图 5.8 和图 5.9 所示：

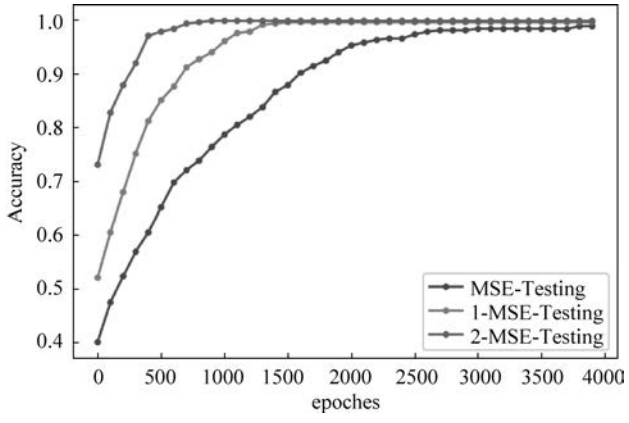

图 5.8　不同级别 MSE 损失函数下加速对比图

由图 5.8 可知，在 2-MSE 具有不错的表现，其训练速度和分类准确率（为 100%，且十分稳定）甚至超过了 1-Cross-Entropy。而图 5.9 中使用 2-Cross-Entropy 出现了梯度爆炸的现象，并且多次重复实验结果均出现了该现象，说明此实验中 Cross-Entropy 的指数加速上限为 1-Cross-Entropy。在使用 3-MSE 做实验时同样出现了梯度爆炸的现象，所以此类问题中 MSE 的指数加速上限为 2-MSE。

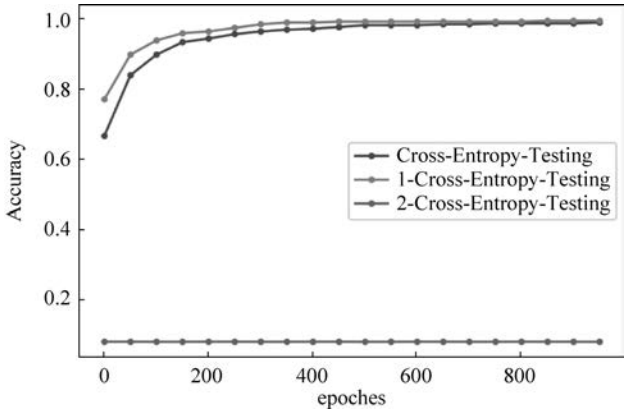

图 5.9　不同级别 Cross-Entropy 损失函数下加速对比图

2. 不同 k 倍 MCELF 下模型加速对比实验

由于过大的 k 值会使模型训练出现陷入局部极小值和梯度爆炸的情况，需要找到合适的 k 值来取得更高的分类准确率和稳定性。这里分别使用值为 10、20、30、40、50 的 k 值，来放大 2-MSE 并观察其分类效果。实验结果如图 5.10 所示。

从图 5.10 可以看出，2-MSE 的 k 值小于或等于 20 时，分类效果很好且具有极高的稳定性。而 k 值大于或等于 20 时，分类效果出现了偏差和波动，k 值越大，其效果越差。说明本实验中在 2-MSE 最优的 k 值在 20 到 30 之间。

我们分别使用值为 1、2、3、4、5 的 k 值，来放大 1-Cross-Entropy 并观察其分类效果。实验结果如图 5.11 所示。

从图 5.11 可以看出，1-Cross-Entropy 的 k 值等于 1 时，分类效果很好且具有极高的稳定性。而 k 值大于 1 时，分类效果出现了偏差和波动，k 值越大，其效果越差。说明本实验中在 1-Cross-Entropy 最优的 k 值在 1 到 2 之间。

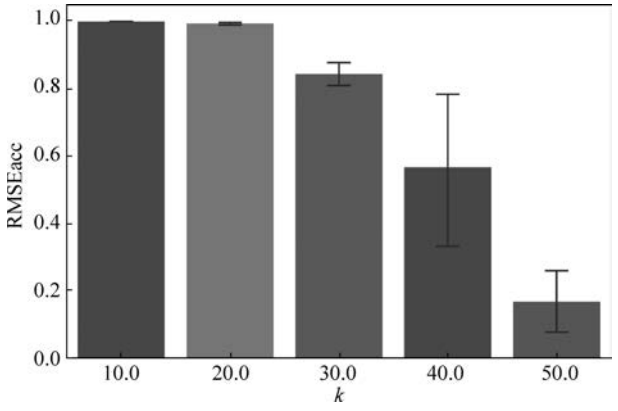

图 5.10　使用不同 k 值 2-MSE 的 10 次训练平均分类准确率及其波动

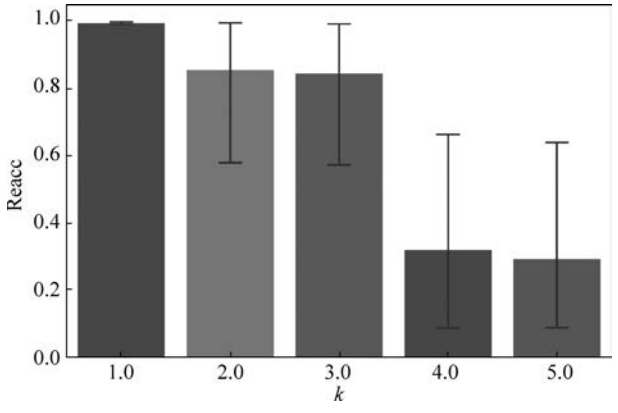

图 5.11　使用不同 k 值 1-Cross-Entropy 的 10 次训练平均分类准确率及其波动

3. MCELF 在不同优化器下模型加速对比实验

由于不同优化器对模型的训练有不同的加速方式，为验证其与 MCELF 的关系，我们结合 MSE 和 Cross-Entropy 两种损失函数，在不同级别 MCELF 下，观察 Adam 和 RMSProp 两种不同优化器的故障诊断模型加速效果。

由图 5.12 至图 5.15 可以看出，该加速方法结合不同的优化器具有不同的效果。对 NAG 优化器的加速效果最为明显；由于 Adam 优化器本身就具有较快的训练速度，该方法在提高稳定性和最终分类准确率上具有不错的效果；而对于 RMSProp 优化器，不同的复合指数 MSE 都取得了相似的

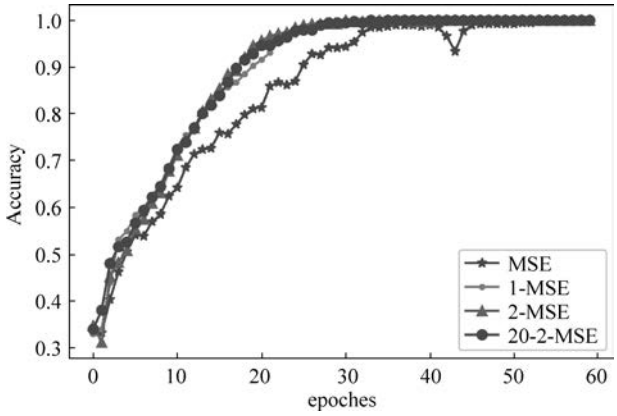

图 5.12　使用 MSE、1-MSE、2-MSE 和 20-2-MSE 损失函数在 Adam 优化器下的对比图

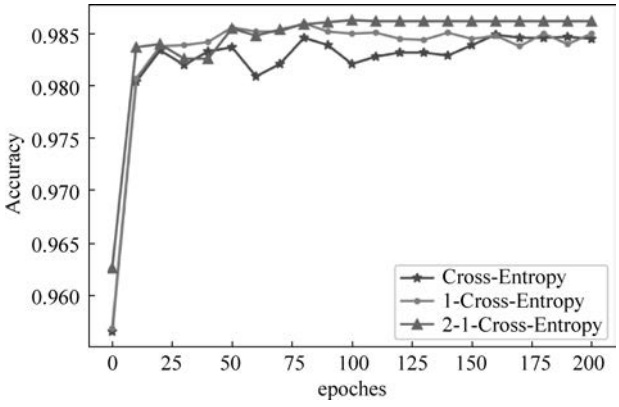

图 5.13　使用 Cross-Entropy、1-Cross-Entropy 和 2-1-Cross-Entropy 损失函数在 Adam 优化器下的对比图

效果，而不同复合指数 Cross-Entropy 并没有性能上的提升，其最终结果略差于其他的优化器。总体而言，该加速方法在不同的优化器下都有一定的加速优化效果，且最终都可以在几乎相同的迭代次数到达最优点，表明了其最终的加速上限都是十分接近的，可能存在统一的加速上限，可以在具体应用中通过寻找适当的 k 和 n 值获得最优的分类效果。

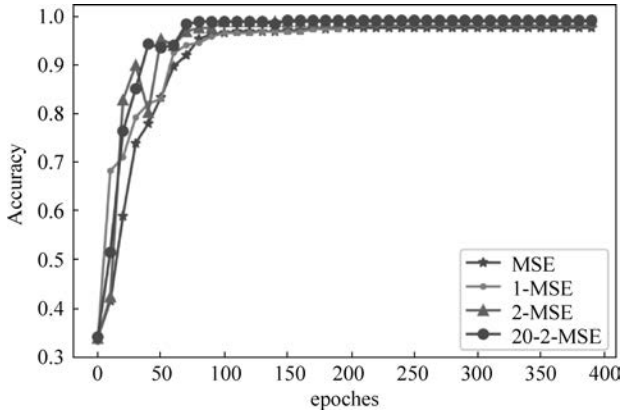

图 5.14 使用 MSE、1-MSE、2-MSE 和 20-2-MSE 损失函数在 RMSProp 优化器下的对比图

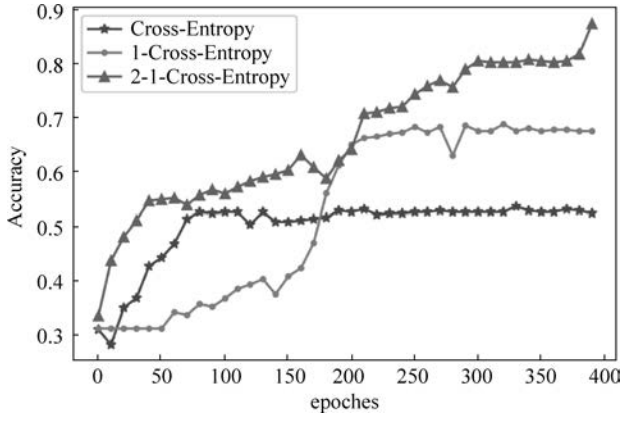

图 5.15 使用 Cross-Entropy、1-Cross-Entropy 和 2-1-Cross-Entropy 损失函数在 RMSProp 优化器下的对比图