# 

# 卷积神经网络（CNN）中深层残差网络（ResNet）在简单数据集上的表现

# 姓名：钱俊帆

# 学号：3230100542

# 专业：临床医学（5+3，儿科方向）

目录

[第一章 卷积神经网络（CNN） 4](#_Toc169292303)

[1.1 模型架构 4](#_Toc169292304)

[1.2 训练方法 4](#_Toc169292305)

[1.2.1 数据归一化 4](#_Toc169292306)

[1.2.2 数据集加载处理 4](#_Toc169292307)

[1.2.3 损失函数 4](#_Toc169292308)

[1.2.4 优化器 5](#_Toc169292309)

[1.2.5 训练过程 5](#_Toc169292310)

[1.3 训练结果和讨论 5](#_Toc169292311)

[第二章 深层残差网络（ResNet） 7](#_Toc169292312)

[2.1 模型架构 7](#_Toc169292313)

[2.2 训练方法 8](#_Toc169292314)

[2.3 训练结果与讨论 8](#_Toc169292315)

[第三章 总结 9](#_Toc169292316)

[参考书目 11](#_Toc169292317)

# 绪论

卷积神经网络在图像的特征提取上发挥了重要作用，其广泛运用于图像分类任务，人脸识别任务等等。而本文章则关注在MNIST数据集[[1]](#footnote-1)上，测试一般相对浅层的神经网络和深层残差网络（ResNet）的表现，并得出相关的结论。

注：

1. 本实验的model权重保存在附件的my\_pt文件夹里面，命名规范为{迭代次数}\_{模型类别}.pt。
2. 本实验的训练及模型架构源代码分别在networks.py和deep\_networks.py中。
3. 本实验训练时的train\_loss, test\_loss的相关图像保存在loss\_pic文件夹下，命名和权重相同。
4. 本实验的模型测试准确度的源代码位于test.py,deep\_test.py文件中。
5. 卷积神经网络（CNN）
   1. 模型架构

Input shape: [1, 28, 28]

卷积: out\_channel = 64, kernel\_size = 5, padding = 1 🡪 shape: [64, 28, 28]

激活函数: Tanh

最大池化: kernel\_size = 2 🡪 shape: [64, 13, 13]

卷积: out\_channel = 32, kernel\_size = 3, padding = 1 🡪 shape: [32, 13, 13]

激活函数: Tanh

最大池化: kernel\_size = 2 🡪 shape: [32, 6, 6]

卷积: out\_channel = 16, kernel\_size = 3, padding = 1 🡪 shape: [16, 6, 6]

激活函数: Tanh

（全连接层）展开

线性层: in\_futures: 16 \* 6 \* 6, out\_features:128

激活函数: Tanh

线性层: in\_futures: 128, out\_features:10

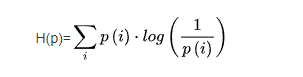
* 1. 训练方法
     1. 数据归一化

使得数据均值为0， 方差为1

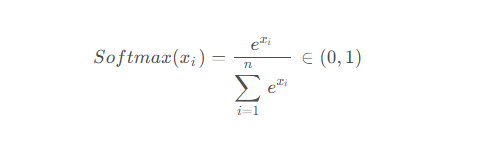
* + 1. 数据集加载处理

使用DataLoader, batch\_size = 64

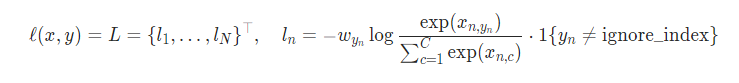
* + 1. 损失函数

采用了CrossEntropyLoss, 但是pytorch的交叉熵损失和数学定义略有不同，下面进行详细解释并给出数学原理：

这是交叉熵的定义，所以传入参数应当为概率，然后和真实的概率计算交叉熵损失，继而反向求导来减少交叉熵损失使得预测概率和真实概率相近。

而再pytorch中可以使用softmax来解释数据成概率，下面是softmax的公式：

即将所有数据归一化，其相加的值为1即可。

然而发现很有趣的一点，我在代码里直接传入了最后的输出的10个特征值，而没有经过softmax将其解释成概率，这是因为pytorch在实现CrossEntropyLoss的时候并不是和数学上的概念完全对应，下面是pytorch官方文档对CrossEntropyLoss的解释：

可以发现pytorch在实现的时候已经使用了softmax，因此我们输入不需要再显式的使用softmax，不过在使用model进行预测的时候还是要自己加上的。

同时，softmax使用负对数似然（NLL）作为损失在这里和交叉熵损失的效果相同。

* + 1. 优化器

优化器是实现自动反向传播并且更新参数的过程，使用了SGD优化器，即最朴素的梯度下降方法,其中超参数 leaning\_rate = 1e-2

* + 1. 训练过程

数据归一化加载以后传入model,产生10个值，其索引对应相关的数字0-9，和期望值计算交叉熵损失，最后使用梯度下降算法更新权重。

* 1. 训练结果和讨论

其中蓝色线是训练集损失，黄色线是测试集损失。（注：由于代码原因，测试集的损失是在训练集损失计算后，更新权重再计算的，所以测试集损失会领先一个epoch）

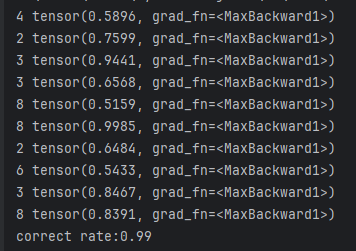
而将数据分段查看后

从这几张图片可以看出，一开始随着训练的进行，再训练集和测试集上的损失都在减小，随着不断的迭代，训练集的loss还在减小，而测试集的loss开始增大，说明了model开始过拟合。

而这里对训练集和测试即大小的相对关系做出解释：

首先，二者的大小关系由于loss计算的先后问题没有呈现出测试集大于训练集的情况，即每次先先计算训练loss，反向传播后才计算测试集loss。[[2]](#footnote-2)

其次，再本数据集中由于数据集比较简单，所以本身测试集和训练集的loss相差不大，所以主要受上面的影响很大。

最后编写了一个程序计算了model在测试集熵预测的准确率：

使用的是200\_model.pt这个模型，在1000个样本中预测，列出了其中被误判的几个样本，同时计算得到准确率为99%，可见对于简单数据集，使用简单的卷积神经网络已经足够应付相应任务了。

1. 深层残差网络（ResNet）
   1. 模型架构

Input shape: [1, 28, 28]

卷积: out\_channel = 32, kernel\_size = 3, padding = 1 🡪 shape: [32, 28, 28]

激活函数: ReLU[[3]](#footnote-3)

最大池化: kernel\_size = 2 🡪 shape: [32, 13, 13]

ResNet块（ResNet Block）[[4]](#footnote-4)

输入：x

卷积: out\_channel = 32, kernel\_size = 3, padding = 1 🡪 shape: [32, 28, 28]

批量归一化

激活函数: ReLU 🡪 输出：out

返回：x + out

在model中这个ResNet块重复了20次，一次构建了一个深度的卷积神经网络

最大池化: kernel\_size = 2 🡪 shape: [32, 7, 7]

（全连接层）展开

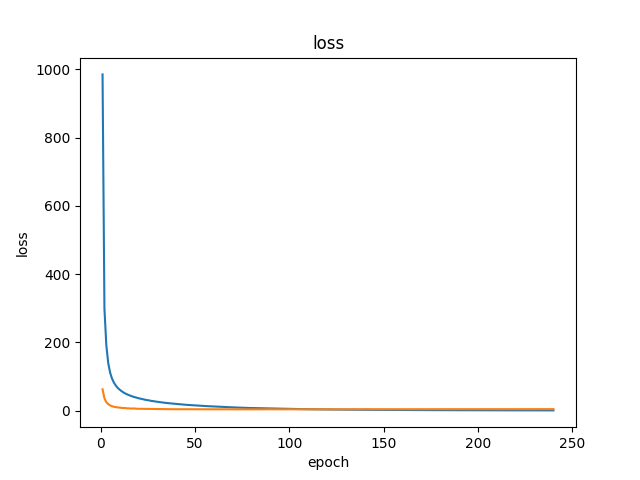
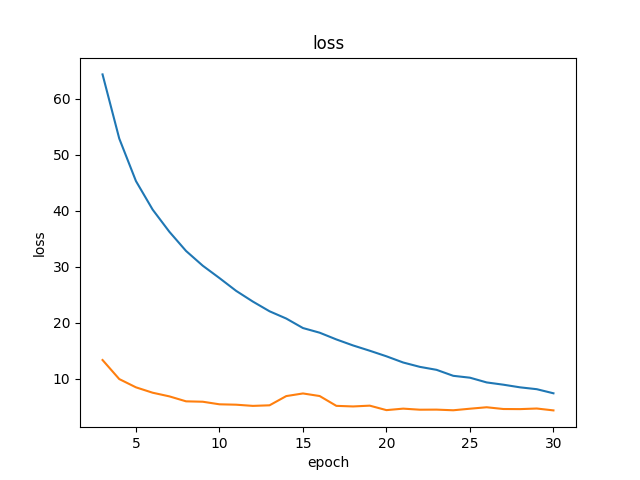
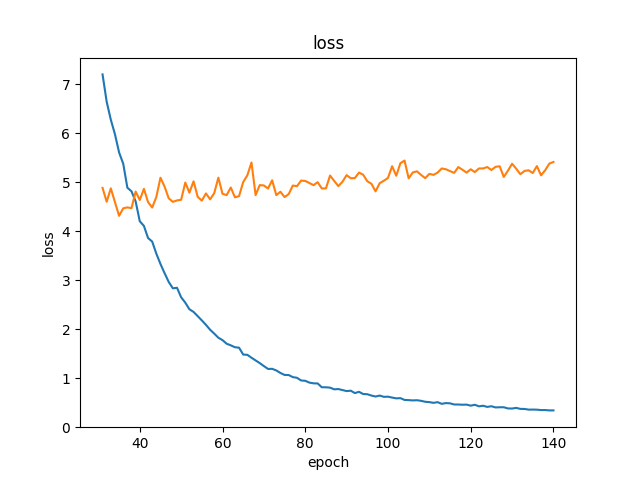
线性层: in\_futures: 32 \* 7 \* 7, out\_features:128

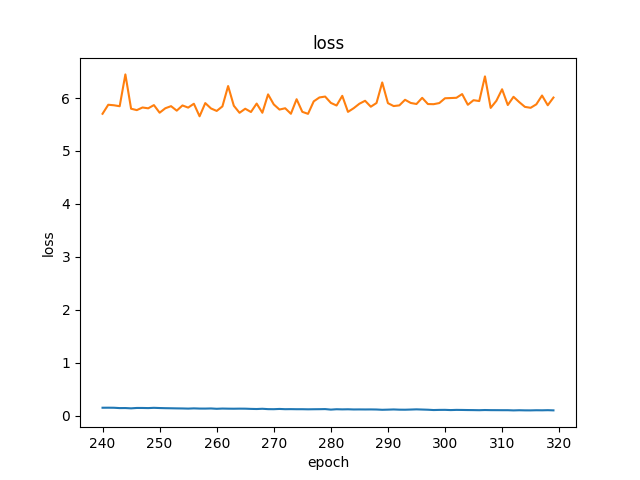
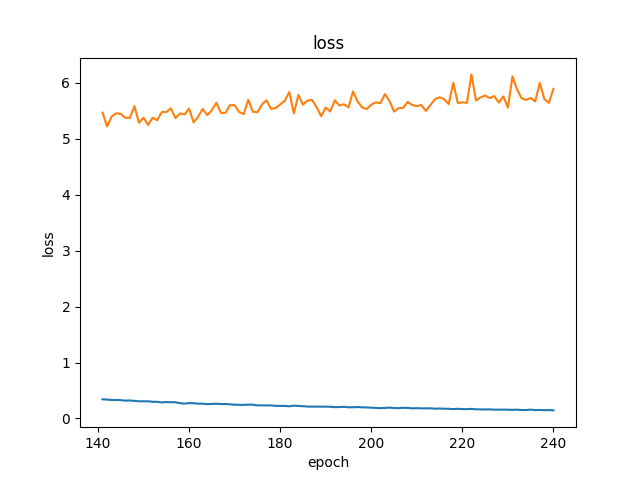
激活函数: ReLU

线性层: in\_futures: 128, out\_features:10

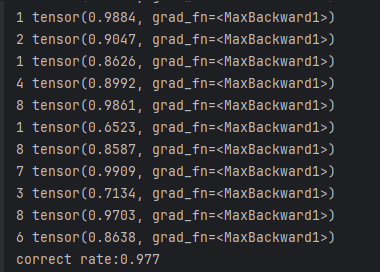
* 1. 训练方法

这里的训练方法和第一张相同，仅将learning rate改成了3e-3从而更小心的训练深层的神经网络。

* 1. 训练结果与讨论

其中蓝色线是训练集损失，黄色线是测试集损失。

从中可以看见，模型在训练早期已经开始过拟合，但是从后面的曲线可以看出，似乎过拟合的趋势没有继续增大，而是保持了平稳。这可能是由于数据集的简单，只有一个颜色通道，从而在图片信息中不会学到过多的多余特征，从而导致了过拟合现象不明显，这种现象在第一个神经网络中也有发现。

同时看一下模型在训练集下面的准确率：

使用的模型是240\_deep\_model.pt，可以发现他的准确率似乎不如第一个简单的神经网络模型，下面做出原因的总结

1. 总结

在上面的数据结果可以看出，似乎更深层次的神经网络的表现反而不好，有以下几点原因：

1. 深度的网络拥有太多的权重，去记忆那些图片中不重要的特征从而导致了过拟合。
2. 本次数据集使用的是MNIST数据集，是一种比较简单的数据集，使用简单的神经网络已经能够很好的完成任务，而相对复杂的神经网络在简单数据集上可能更加容易导致过拟合。

但总的来说，这两种神经网络在该数据集上的表现都很良好，并不需要过度的吹毛求疵，应当在更加复杂的数据集上去寻找这两种神经网络的优缺点。

但可以肯定的是，能够使用简单的model很好的实现相应的目的，没有过多必要使用过于复杂的模型，从而能够节约部署的简单性和算力。

# 参考书目

Eli Stevens, Luca Antiga, Tjomas Viehmann, 《PyTorch深度学习实战》, 人民邮电出版社, 2022.

1. MNIST 数据库是一个大型手写数字数据库(包含0~9十个数字),包含 60,000 张训练图像和 10,000 张测试图像 [↑](#footnote-ref-1)
2. 这里是代码没有完善的原因，请见谅 [↑](#footnote-ref-2)
3. 为了model更好的处于激活函数的敏感区域，这里一开始model权重的随机化使用了均值为0方式。 [↑](#footnote-ref-3)
4. 在每个ResNet块中还加入了一个批量归一化的层，使得深度神经网络能够更好的收敛。 [↑](#footnote-ref-4)