**华东师范大学数据科学与工程学院实验报告**

|  |  |  |
| --- | --- | --- |
| **课程名称**：当代人工智能 | **年级**：2021级 | **上机实践成绩**： |
| **指导教师**：李翔 | **姓名**：李睿恩 | **学号**：10215501434 |
| **上机实践名称**：图像分类及经典CNN实现 | | **上机实践日期**：2023.11.23 |
| **上机实践编号**：3 | **组号**：无 | **上机实践时间**：8:30 |

1. 实验目的

* 通过使用CNN搭建神经网络，完成图像分类任务。

1. 实验任务

* 导入MNIST数据集，对该数据集进行图像分类。
* 使用至少四种CNN模型实现图像分类。LeNet，AlexNet，ResNet为必须使用的架构。

1. 实验环境

本次实验基于Colaboratory环境，调用其上的T4 GPU完成。Colaboratory是由Google开发的一种基于云端的交互式环境，提供了免费的计算资源（包括CPU、GPU和TPU），可让用户在浏览器中编写和执行代码，而无需进行任何配置和安装。

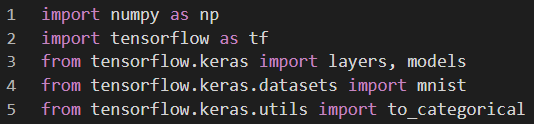
本次实验基于Python的编程语言，采用了TensorFlow模块，搭建了所有神经网络。

1. 实验过程

4.1 实验准备

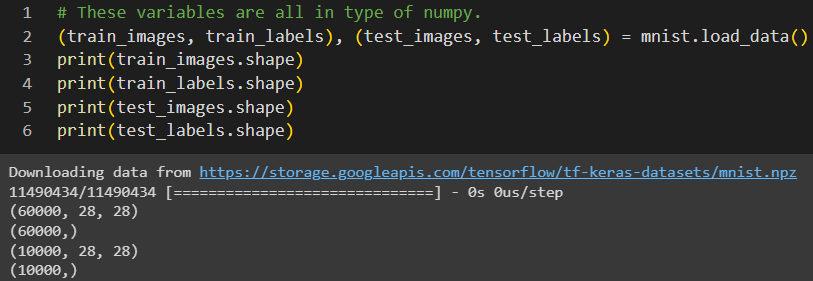
无论我们在搭建何种架构，有一些前置准备操作总是必须事先完成的。

在本次实验中，我们将采用TensorFlow模块以实现神经网络的搭建。为了配合TensorFlow的使用，我们还会调用numpy模块进行数据处理。图1展示了导入的必要模块。



**图1 导入必要模块**

本次实验使用了MNIST数据集以完成图像分类任务。我们调用了TensorFlow中已经存在的MNIST数据集以完成实验。TensorFlow中的MNIST数据集已经分好了训练集与测试集。图2为导入MNIST数据集以及观测其中数据的形状，可以发现我们在训练集中有60000张规模为28×28的图片，测试集中则有10000张同等规模的图片。

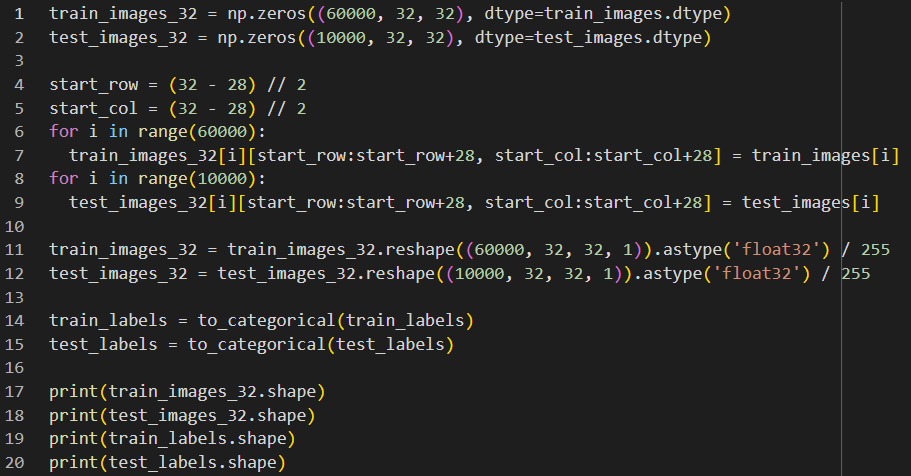


**图2 导入MNIST数据集**

MNIST数据集的特点在于每张图片的大小均为28×28，这对神经网络来说规模过小，神经网络可能无法搭建过深。为了能够更好地搭建适配的神经网络，并且为了可以让过滤器捕获到图像边缘的特征，我们用0来对图像边缘进行填充，使其扩展到了32×32。这个流程被称为填充（Padding）。

另外，我们还将训练集与测试集的标签转化为了One-hot编码。这主要是因为，我们搭建的神经网络的最后一层通常都是基于Softmax或Sigmoid激活函数的生成10个神经元的全连接层，每一个神经元都是一个0到1之间的实数。这意味着，对于每张图片而言，最后我们得到的实际上是它属于每一类的概率。一个符合直觉的认知是，一张图片被预测属于正确类别的概率越高，说明其准确率越高，因此我们将所有正确的标签转化为了One-hot编码，将图片通过神经网络得到的结果与正确的One-hot编码作对比，从而得到最后的准确率。具体操作见图3.

接下来我们将进行各神经网络的搭建。



**图3 图像与标签的数据预处理**

4.2 LeNet

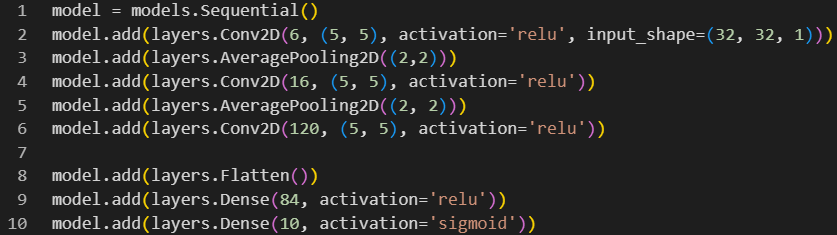
LeNet是一个在1998年由Yann LeCun创建的卷积神经网络架构，被广泛用于手写数字识别中。由于它本身就被应用于MNIST数据集，因此我们可以非常准确地还原LeNet架构。它的架构如表1所示。

注意，由于实际使用的激活函数并不是决定神经网络架构最重要的因素，而且激活函数也并没有任何强制的限制，因此我们在本次实验中描述架构的图表中均没有标出激活函数，但实际上在每个卷积层与全连接层后，均应该有激活函数作为非线性层。

**表1 LeNet架构**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 层 | 类型 | 图 | 尺寸 | 内核尺寸 | 步幅 |
| OUT | 全连接 | -- | 10 | -- | -- |
| F6 | 全连接 | -- | 84 | -- | -- |
| C5 | 卷积 | 120 | 1×1 | 5×5 | 1 |
| S4 | 平均池化 | 16 | 5×5 | 2×2 | 2 |
| C3 | 卷积 | 16 | 10×10 | 5×5 | 1 |
| S2 | 平均池化 | 6 | 14×14 | 2×2 | 2 |
| C1 | 卷积 | 6 | 28×28 | 5×5 | 1 |
| In | 输入 | 1 | 32×32 | -- | -- |

代码实现如图4。



**图4 LeNet实现代码**

在本次实验中，我们为每一个神经网络编译的模型都是选用categorical\_crossentropy作为损失函数，采用Adam的优化器，以默认值（0.001）作为学习率的初始值。我们会将模型应用于我们处理过的训练集中，并要求从我们处理好的数据中取出80%作为真正的训练集，剩余的20%作为验证集，要求对该训练集的学习要进行5轮（epochs=5），要求该训练集在每次权重更新之前都只是看到64个样本（batch\_size=64）。在模型训练完毕后，我们将其用于测试集，观察其最终的结果。

在LeNet的实现中，我们经历上述步骤，训练耗时30.53秒，在测试集上得到了98.9%的准确率。这说明我们的LeNet搭建效果较好。

4.3 AlexNet

AlexNet的CNN架构以大比分赢得了2012年的ILSVRC竞赛。它和LeNet架构很相似，只是比LeNet更大更深。它直接将卷积层堆叠到其它层之上，而不是在每个卷积层之上堆叠池化层。值得注意的是，由于AlexNet并不是为MNIST数据集量身打造的，因此我们无法完全还原其原版架构，但我们仍然可以基于其神经网络层的排列顺序，将它改造为适合MNIST数据集的神经网络架构。

我们所实际使用的架构如表2所示。为了能够适配MNIST数据集，我们没有遵循传统架构，而是改变了其除去层的类型与顺序以外的几乎所有内容。

**表2 AlexNet架构**

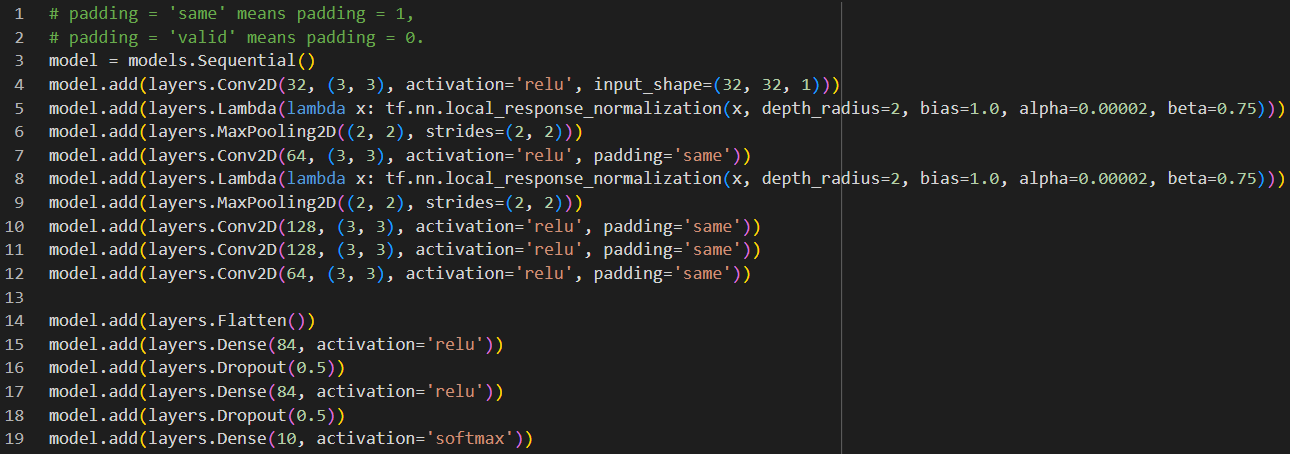
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 层 | 类型 | 图 | 尺寸 | 内核尺寸 | 步幅 | 填充 |
| OUT | 全连接 | -- | 10 | -- |  | -- |
| F9 | 全连接 | -- | 84 | -- |  | -- |
| F8 | 全连接 | -- | 84 | -- |  | -- |
| C7 | 卷积 | 64 | 8×8 | 3×3 | 1 | 1 |
| C6 | 卷积 | 128 | 8×8 | 3×3 | 1 | 1 |
| C5 | 卷积 | 128 | 8×8 | 3×3 | 1 | 1 |
| S4 | 最大池化 | 64 | 8×8 | 2×2 | 2 | 0 |
| C3 | 卷积 | 64 | 16×16 | 3×3 | 1 | 1 |
| S2 | 最大池化 | 32 | 16×16 | 2×2 | 2 | 0 |
| C1 | 卷积 | 32 | 32×32 | 3×3 | 1 | 1 |
| In | 输入 | 1 | 32×32 | -- | -- | -- |

一个过深的神经网络很可能会导致对训练集的过拟合。因此，除去层数的增加，AlexNet的另一个非常重要的特性是其使用了淘汰（Dropout）策略与数据增强（Data Augmentation）策略。淘汰策略指的是神经网络会在某一层随机删除一些输出的神经元，从而减少特征的捕获，该方法可以有效地防止过拟合。数据增强策略指的是将图片进行各种偏移、翻转、改变光照条件等操作，随机移动训练数据。

在我们的神经网络搭建中，我们使用了淘汰策略，但我们没有使用数据增强策略。保留淘汰策略，不但是为了尽可能还原AlexNet，更是为了防止对训练集的过拟合。但我们没有使用数据增强策略，则是考虑到了MNIST数据集本身的特性。MNIST数据集是手写数字数据集，它的分类非常依赖于图像中数字的形状与位置，一些数据增强操作很可能会对数据集本身的性质造成毁灭性的打击，比如如果将数字6旋转180度，得到的数字就会变成9；对数字8进行大幅度的偏移，可能会导致该数字变为数字3……这都必然会导致模型的训练受到出乎意料的影响。为此，我们并没有使用数据增强策略。

AlexNet的另一个特性在于它使用了局部相应归一化（Local Response Normalization, LRN）。它主要被用于激活函数后的归一化过程，它对局部神经元的响应进行归一化，使得响应较大的神经元抑制响应较小的神经元，从而进一步增强了模型的泛化性能。

代码实现如图5.



**图5 AlexNet实现代码**

模型经过了52.06秒的训练后，我们将模型用于测试集，最终得到了98.8%的准确率。这说明我们的AlexNet搭建较好。

4.4 ResNet

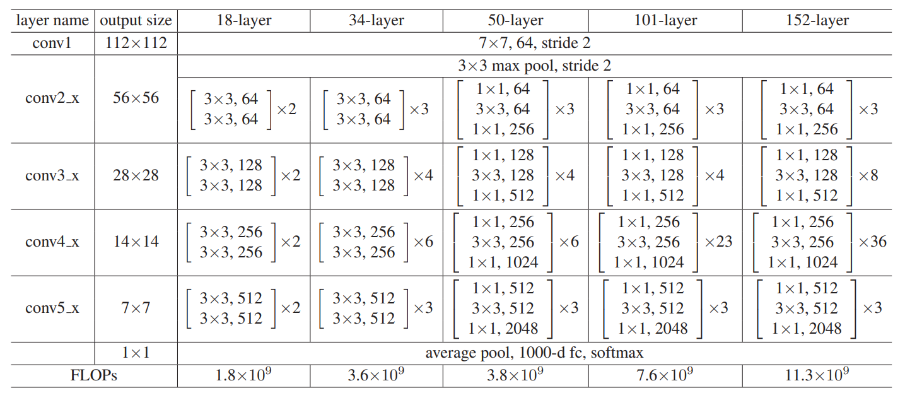
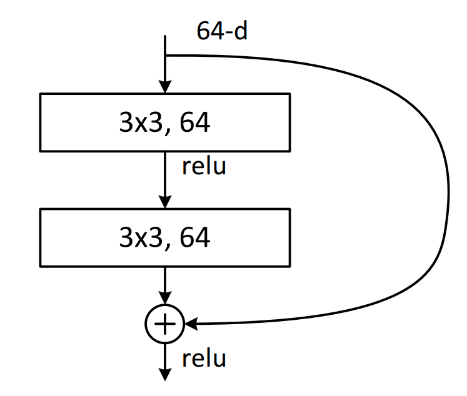
ResNet是一个由深度学习领域的专家何恺明等人开发并获得2015年ILSVRC竞赛冠军的残差网络。原版的神经网络使用了一个由152层组成的非常深的卷积神经网络。能够训练如此深层的网络的关键是使用了跳过连接（快捷连接）：输入到一个层中的信号也被添加到位于堆栈上方的层的输出端。

直观来看，一个更深的神经网络似乎预测效果会比一个浅层的神经网络要好。但是，在实际实验中，一个更深的神经网络的预测效果远不如一个浅层的神经网络。这一方面是因为梯度消失与梯度爆炸问题，另一方面则是因为过深的神经网络会造成神经网络的负优化问题，不但没有提升训练精度，还扭曲了网络空间，升高了训练集上的损失函数值。为此，我们希望深层神经网络是恒等映射，并希望深层神经网络可以相比于恒等映射更进一步产生优化。因此，我们诞生了残差单元的思想。

对于较为浅层的ResNet而言，残差单元会主要由输入端、两个卷积层与一个输出端构成。在残差单元中，输入*x*会被添加到网络的输出端，求和后的结果才是被激活函数执行的对象。这样的架构可以确保，如果神经网络即将走向负优化，那么先前的最优解仍然可以被保留在输出的结果中，缓解了梯度问题。何恺明原作论文中的残差单元的结构见图6.

但在实际执行时，残差单元的第一个卷积层后紧跟着的分批归一化（Batch Normalization）与ReLU激活层，第二个卷积层后紧跟着的分批归一化。在输入数据经过多层神经网络后，数据会发生内部协变量偏移（Internal Covariate Shift），导致数据分布发生变化，而使用了分批归一化后，我们可以使得网络更容易学到适应数据的有效数据，并且间接减少过拟合。

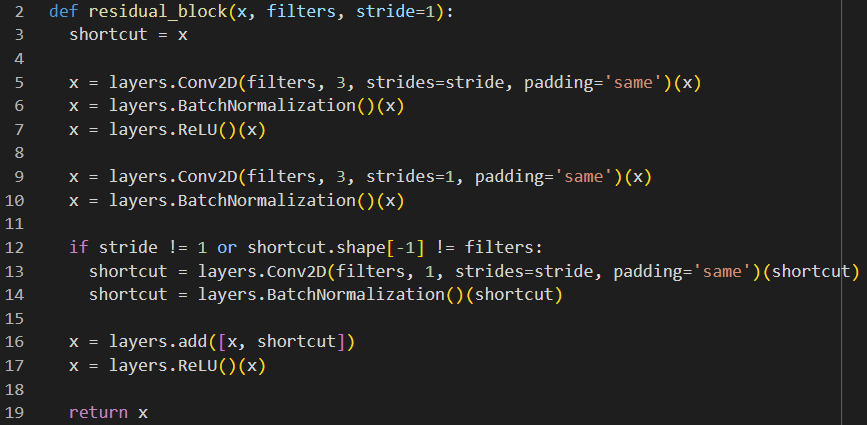
基于此，我们可以学习传统的ResNet架构。传统的ResNet架构由输入层、卷积层、最大池化层、平均池化层、全连接层构成，其中有大量的卷积层实质上构成了上述所提到的残差单元。由于MNIST数据集规模较小，并不会适应非常深层的神经网络，因此我们在这里搭建ResNet-18以对其进行还原。各层数下的ResNet架构如图7.



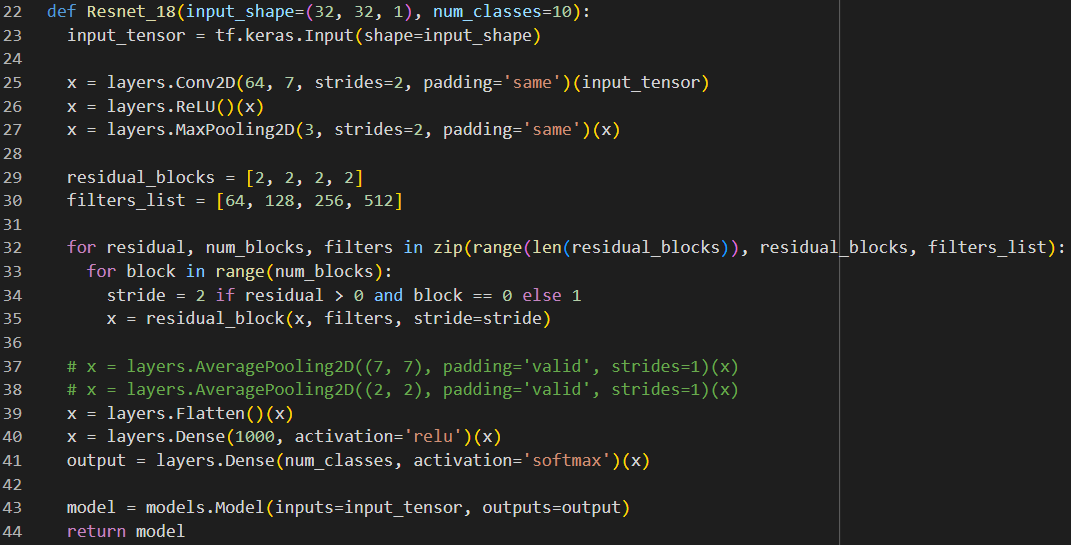
**图6 残差单元的结构 图7 各层数下的ResNet架构**

以下是我们对ResNet-18的搭建。其中，图8所示的residual\_block是一个函数，它封装了残差单元，实现了残差单元的功能。图9所示的Resnet\_18函数则根据图7中18-layer所示的方式搭建了一个ResNet-18。值得注意的是，图7中的这些卷积层之间存在一些通道数量的改变，因此我们需要在卷积层通道数变化的时刻注意通道数量的变换。

另外，我们删去了传统流程中的最后两层平均池化层，这一方面是因为MNIST数据集本身图像过小，一步步地卷积、激活、池化已经让我们得到的特征规模极小，已经不能再进一步地降维了，另一方面是这并不是ResNet最核心的部分。



**图8 残差单元**



**图9 ResNet-18实现代码**

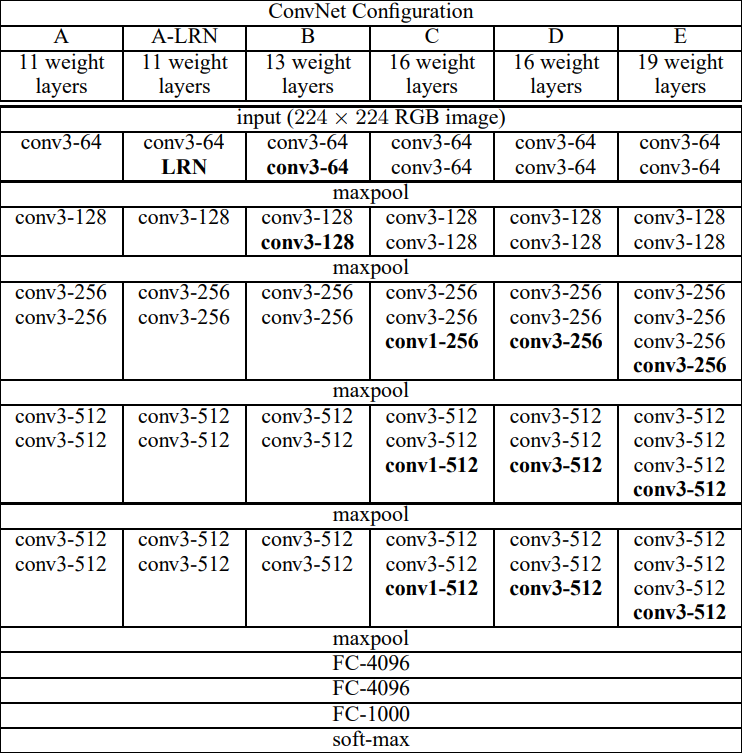
模型经过148.23秒的训练后，我们将模型用于测试集，最终得到了98.1%的准确率。这说明我们的ResNet搭建较好。

4.5 \*VGGNet

VGGNet曾被称为是通过简单堆叠卷积构建网络的巅峰之作。它是由牛津大学的Visual Geometry Group提出的一种架构，该架构获得了2014年ILSVRC分类任务的亚军以及定位任务的冠军。它的实质是使用了比AlexNet更小的卷积核和更深的网络。

它的架构主要是简单的由卷积层、池化层和全连接层的堆积，具体架构见图10.

我们原计划实现一个目前较为常见的VGGNet-16。但是，令我们很遗憾的是，由于其中涉及了过多的池化层，图片的维数逐渐越来越小，直到最后图片的维数已经无法实现全部的最大池化层。如果强行用0在图片周围进行填充，那么模型在验证集上的准确率就会只有10%左右，这意味着神经网络没有学习到任何特征。因此，我们将最后一组卷积层与最大池化层舍弃了。并且，考虑到图片的特征过少，我们也删去了一层全连接层。



**图10 各层数下的VGGNet**

代码实现如图11.



**图11 VGGNet实现代码**

模型训练了143.97秒后，我们将模型用于测试集，最终得到了99%的准确率。这说明我们的VGGNet搭建较好。

4.6 \*GoogLeNet

GoogLeNet架构是由来自Google研究部的Christian Szegedy等人开发的，赢得了2014年的ILSVRC挑战。这一伟大的性能很大程度上源于它的网络比以前的CNN更深。这是通过一个被称为初始化模块的子网使之成为可能。

初始化模块指的是，一个输入的内容会分别给到四个神经网络层中。这四层分别服从：

A. 使用了1×1内核，步幅为1，填充为1的卷积层。

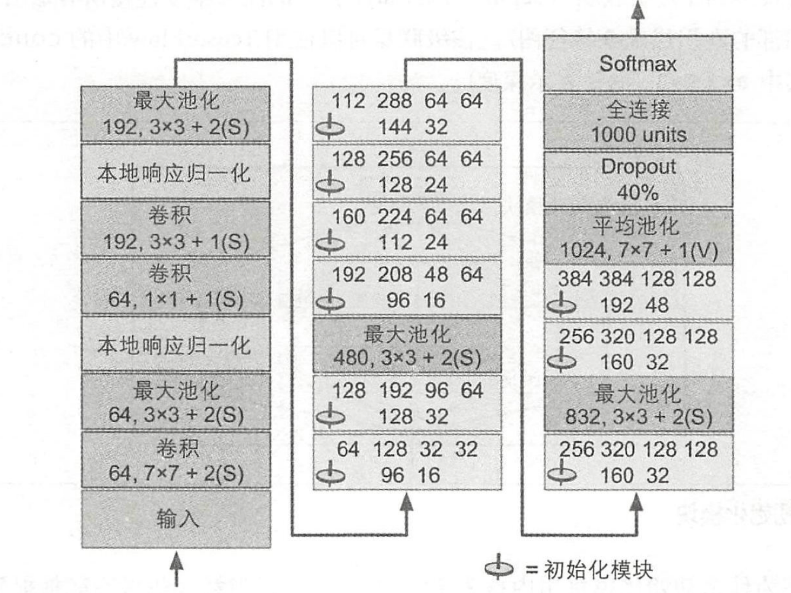
B. 使用了1×1内核，步幅为1，填充为1的卷积层，之后再使用了一次3×3内核，步幅为1，填充为1的卷积层。

C. 使用了1×1内核，步幅为1，填充为1的卷积层，之后再使用了一次5×5内核，步幅为1，填充为1卷积层。

D. 使用了3×3内核，步幅为1，填充为1的最大池化层，之后再使用了一次1×1内核，步幅为1，填充为1的卷积层。

四层得到的结果会进入最后的深度级联层（Depth Concatenation Layer），即从所有的四个顶部的卷积层堆叠特征图。整个的一个完整模块被称为初始化模块。这样做的好处是可以捕捉到不同尺寸的图像模式。

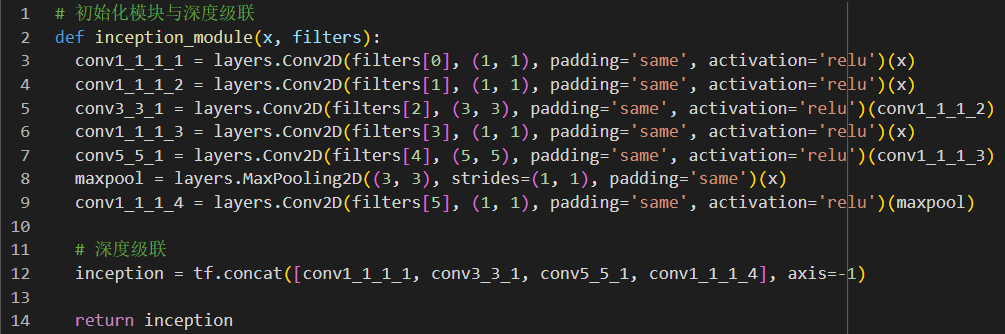
它的原始架构还采用了局部响应归一化与Dropout操作提升泛化能力，如图12所示。

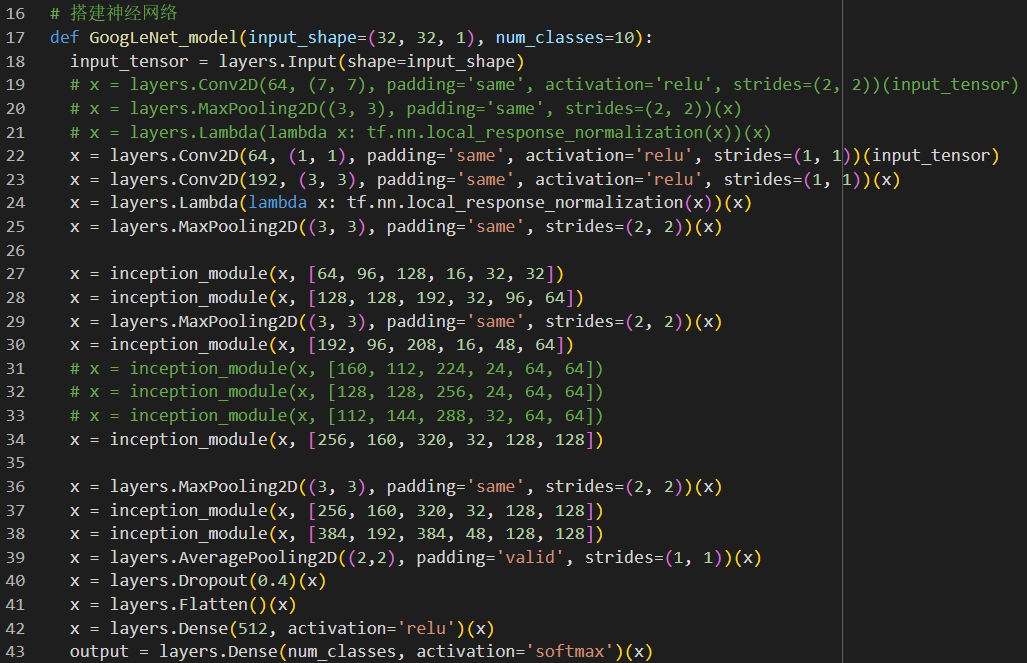


**图12 GoogLeNet原始架构**

然而，其中有一些卷积层与池化层存在的目的在于减少计算量、增加计算速度，使用大量的初始化模块也是为了学习到更丰富的特征。然而，手写数字集本身规模已经很小，过多的层数不但可能无法实现，过多的层数也可能导致神经网络的恶化，训练效果极差。为此，尽管我们会保留大部分的初始化模块、局部响应归一化与Dropout操作作为核心，但我们还是删去了部分层。实现代码如图13.

模型经过了507.73秒的训练后，我们将模型用于测试集，最终得到了99%的准确率。这说明我们的GoogLeNet搭建较好。



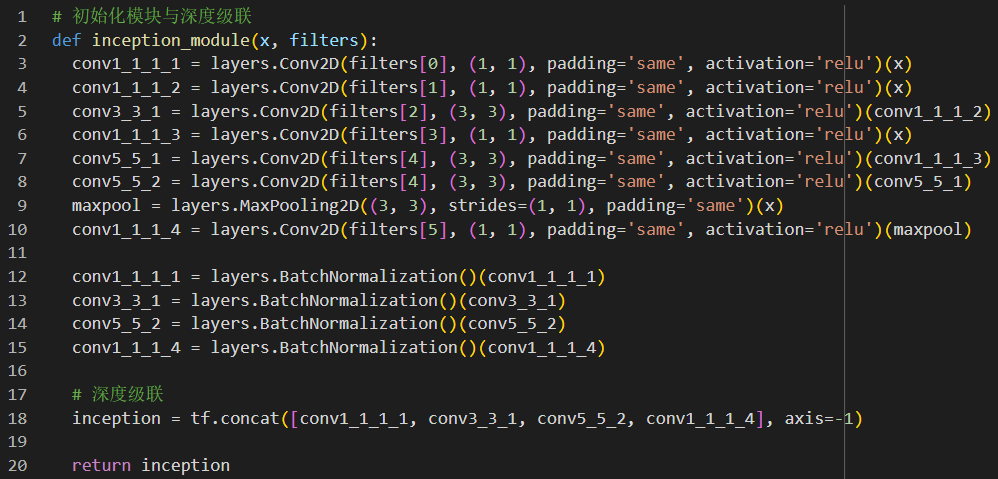


**图13 GoogLeNet的实现代码**

4.7 \*InceptionV2

实际上，GoogLeNet还有一个名称InceptionV1，这主要是因为它创新性地提出了深度级联层。在之后的一年，Google研究部的成员就开发出了InceptionV2神经网络架构。它与GoogLeNet的区别在于，它使用了两个3×3的卷积层代替了一个5×5的卷积层，减少了计算量，并且在当时创新性地提出了分批归一化的方式，让模型更有效地学习数据，并且间接减少了数据的过拟合。

主要修改的地方如图14所示。



**图14 InceptionV2的实现代码**

模型经过了572.33秒的训练后，我们将模型用于测试集，最终得到了98.6%的准确率。这说明我们InceptionV2搭建较好。

1. 总结与分析

我们使用的6种卷积神经网络架构在测试集上均有着大于98%的准确率，这一方面是由于MNIST数据集的特征可能过于简单，易于学习，另一方面也意味着我们的神经网络搭建较为成功，结果对比见表3。但是，在搭建的过程中，我们还是遇到了不少的问题。

**表3 不同卷积神经网络架构在MNIST数据集的表现**

|  |  |  |  |
| --- | --- | --- | --- |
| 架构 | 规模 | 测试集准确率 | 训练时间（秒） |
| LeNet-5 | 相对较浅 | 98.9% | 30.53 |
| AlexNet | 相对较浅 | 98.8% | 52.06 |
| ResNet-18 | 相对较深 | 98.1% | 148.23 |
| VGGNet-16 | 相对较浅 | 99.0% | 143.97 |
| GoogLeNet | 相对较深 | 99.0% | 507.73 |
| InceptionV2 | 相对较深 | 98.6% | 572.33 |

有的神经网络在搭建的过程中会提示有关矩阵规模的问题，这有可能是因为我们使用了过多的池化层导致的图像规模的降维，而当我们降维到一定程度后，图片就会小到无法再进一步降维（如ResNet，VGGNet，GoogLeNet、InceptionV2），此时我们需要删去一些池化层。有的神经网络会在我们搭建原版架构时在验证集上展现出仅仅10%左右的准确率（如GoogLeNet），而实际上，随机地猜测所有图片的分类，其期望准确率也是10%，这说明神经网络完全没有学习到图像的特征，这则可能是因为神经网络层数过深导致了梯度消失问题，从而导致模型几乎没有被优化，或者是学习率并没有被较好地选择，此时我们需要让神经网络层变浅，或者尝试修改学习率。

尽管这些数据在测试集上均有大于98%的准确率，但还是可以发现其准确率存在一些微弱的区别。直觉上，我们会认为一个非常深层的神经网络可以学习到更多的特征，从而带来更高的准确率，但在此次试验中，恰恰是一些非常深层的，使用了大量技术的模型在测试集上没有最优秀的表现（如ResNet、InceptionV2）。这可能是因为过于深层的神经网络会遇到梯度消失、梯度爆炸或负优化问题，最终的训练效果反而不如浅层神经网络；也可能是因为神经网络没有使用正则化等技术提升模型的泛化能力，使得模型还是过于注重训练集本身的特征，导致了一定程度的过拟合；也可能是因为图像规模本身较小，而模型本身不适配于尺寸过小的图片。这启发我们在使用神经网络时，可以不要执着于过于深层的神经网络，而是注重于寻找最合适于特定数据集的网络架构。

并且，过深的神经网络还会带来过大的计算负担，使得训练时间变长，对硬件需求变高。在上述代码中，我们观察到了一些很深层的神经网络的训练时间非常漫长，最长可以超过550秒，而如果我们将上述代码全部使用CPU运行，那么这可能是一个更加漫长的训练过程。考虑到这个因素，我们应该结合精度与时间综合考虑选择神经网络架构。

综上所示，影响模型结果的主要因素有很多。一方面，虽然直觉上理解，深层神经网络有着更好的表现，但本次实验体现出了一个过深的神经网络可能会导致梯度消失、梯度爆炸、负优化与计算需求过高等问题，因此一个神经网络的深度会非常重要。一方面，神经网络防止过拟合的操作有很多，如淘汰策略，分批归一化等，他们并不总是灵丹妙药，他们反而可能因为对图像特征的学习发生了变化导致训练的结果反而走向欠拟合。一方面，如LeNet这样的神经网络从一开始就是专门为MNIST数据集这样的小型数据集制作的，而其他神经网络则是基于其他数据集诞生的，这意味着这些神经网络实际上可能对数据集本身存在要求，因此数据数量与数据质量都可能影响模型表现。因此，神经网络架构需要针对特定数据集来选择，并且不可以盲目追求深层神经网络，此外还需要适量地提升泛化能力。

1. 参考文献
2. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.
3. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*.
4. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
5. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
6. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).