18051810　陈锐

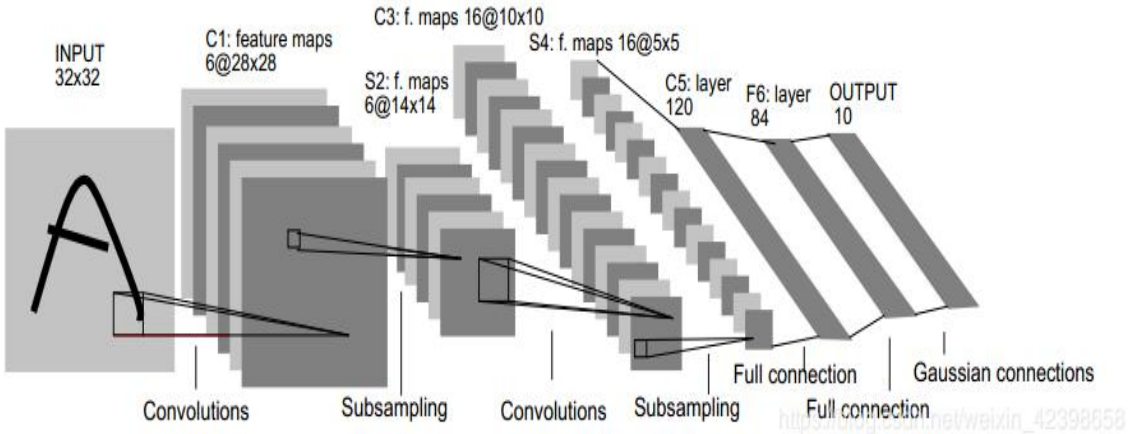
LeNet：

发展背景：1962 年 Hubel 以及 Wiesel 通过生物学研究表明，从视网膜传递脑中的视觉信息是通过多层次的感受野（Receptive Field）激发完成的，并首先提出了感受野的概念。Yann LeCun 1998 年发表了一篇（Gradient-based learning applied to document recognition，1998），文中首次提出卷积-池化-全连接的神经网络结构，由 LeCun 提出的七层网络命名为 LeNet-5，采用了基于梯度的反向传播算法对网络进行有监督的训练。

框架构成：

上图包含输入层总共 8 层网络，分别为：

输入层（INPUT）、卷积层（Convolutions,C1）、池化层（Subsampling，S2）、卷积层（C3）、池化层（Subsampling，S4）、卷积层（C5）、全连接层（F6）、输出层（径向基层）



上图包含输入层总共 8 层网络，分别为：

Input Layer：1*32*32 图像

Conv1 Layer：包含 6 个卷积核，kernal size：5*5，parameters:（5*5+1）*6=156 个

Subsampling Layer：average pooling，size：2*2

Activation Function：tanh　步长:2

Conv3 Layer：包含 16 个卷积核，kernal size：5*5　->16 个 Feature Map

Subsampling Layer：average pooling，size：2*2　步长:2

Conv5 Layer：包含 120 个卷积核，kernal size：5*5

Fully Connected Layer：Activation Function：sigmoid

Output Layer：Gaussian connection(RBF)径向基函数

每层之间的参数计算：

输入层：

C1 层：输出值为 32*32 像素的图片。

输入图片：32*32

卷积核大小：5*5

卷积核种类：6

输出 featuremap 大小：28*28 （32-5+1）

神经元数量：28*28*6

可训练参数：（5*5+1）*6（每个滤波器 5*5=25 个 unit 参数和一个 bias 参数，一共 6 个滤波器）

连接数：（5*5+1）*6*28*28

S2 层是一个下采样层：

输入：28*28

采样区域：2*2

采样方式：4 个输入相加，乘以一个可训练参数，再加上一个可训练偏置。结果通过 sigmoid

采样种类：6

输出 featureMap 大小：14*14（28/2）

神经元数量：14*14*6

可训练参数：2*6（和的权+偏置）

连接数：（2*2+1）*6*14*14

S2 中每个特征图的大小是 C1 中特征图大小的 1/4

3. C3 层也是一个卷积层

输入：S2 中所有 6 个或者几个特征 map 组合

卷积核大小：5*5

卷积核种类：16

输出 featureMap 大小：10*10

C3 中的每个特征 map 是连接到 S2 中的所有 6 个或者几个特征 map 的，表示本层的特征 map 是上一层提取到的特征 map 的不同组合

存在的一个方式是：C3 的前 6 个特征图以 S2 中 3 个相邻的特征图子集为输入。接下来 6 个特征图以 S2 中 4 个相邻特征图子集为输入。然后的 3 个以不相邻的 4 个特征图子集为输入。最后一个将 S2 中所有特征图为输入。

则：可训练参数：6*（3*25+1）+6*（4*25+1）+3*（4*25+1）+（25*6+1）=1516

连接数：10*10*1516=151600

4. S4 层是一个下采样层

输入：10*10

采样区域：2*2

采样方式：4 个输入相加，乘以一个可训练参数，再加上一个可训练偏置。结果通过 sigmoid

采样种类：16

输出 featureMap 大小：5*5（10/2）

神经元数量：5*5*16=400

可训练参数：2*16=32（和的权+偏置）

连接数：16*（2*2+1）*5*5=2000

S4 中每个特征图的大小是 C3 中特征图大小的 1/4

5. C5 层是一个卷积层

输入：S4 层的全部 16 个单元特征 map（与 s4 全相连）

卷积核大小：5*5

卷积核种类：120

输出 featureMap 大小：1*1（5-5+1）

可训练参数/连接：120*（16*5*5+1）=48120

6. F6 层全连接层

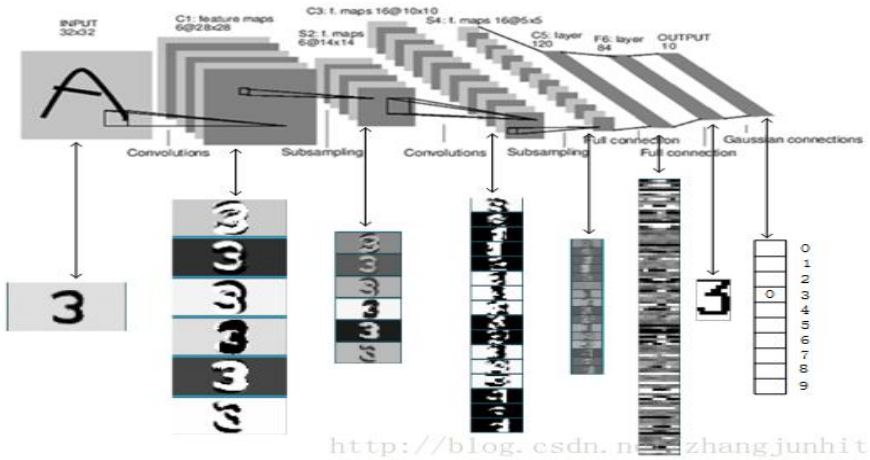
输入：c5 120 维向量

计算方式：计算输入向量和权重向量之间的点积，再加上一个偏置，结果通过 sigmoid 函数

可训练参数:84*(120+1)=10164

7. **输出层**：该层有十个神经元，基于欧几里得距离，F6 层为 84 个输入用$/dpi150x_j$表示，而输出有 10 个用$/dpi150y_i$表示，说明所有输入和权值的距离平方和为依据判断，如果越相近距离越小，输出越小则去哪个。

例子;



应用：

LeNet 是在 1998 年提出的，用于手写体数字的识别

优点：

传统的机器学习进行图像分类，需要手工的设计特征提取器从图像集中提取特征，然后输入到机器学习算法中进行学习。

使用梯度下降的方法训练的多层神经网络，可以直接从大量的数据集中学习到复杂的，高维度以及非线性的特征，这些特征远比人工设计的特征要好很多。

最后，可以在神经网络的全连接层直接利用前面网络层提取到的特征进行分类，不用像传统的机器学习分类那样，分成两个步，而是进行 end-to-end 的学习。

缺点：

LeNet 的设计较为简单，因此其处理复杂数据的能力有限；此外，在近年来的研究中许多学者已经发现全连接层的计算代价过大，而使用全部由卷积层组成的神经网络。

现在在研究中已经很少将 LeNet 使用在实际应用上，对卷积神经网络的设计往往在某个或多个方向上进行优化，如包含更少的参数（以减轻计算代价）、更快的训练速度、更少的训练数据要求等。

实践操作：从网络上找到相关操作实践教程

```
from keras.models import Sequential
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dense
from keras import backend as K
from sklearn import cifar10
class LeNet:
    @staticmethod
    def build(width,height,depth,classes):
        model = Sequential()
        inputShape = (height,width,depth)

        if K.image_data_format() == "channels_first":
            inputShape = (depth,height,width)

        # first set of CONV => RELU => POOL
        model.add(Conv2D(20,(5,5),padding="same",input_shape=inputShape))
        model.add(Activation("relu"))
        model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))

        # second set of CONV => RELU => POOL_layers
        model.add(Conv2D(50,(5,5),padding="same"))
        model.add(Activation("relu"))
        model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))

        # set of FC => RELU layers
        model.add(Flatten())
        model.add(Dense(500))
        model.add(Activation("relu"))
```

相关参数还有点不清楚，无法完全理解代码

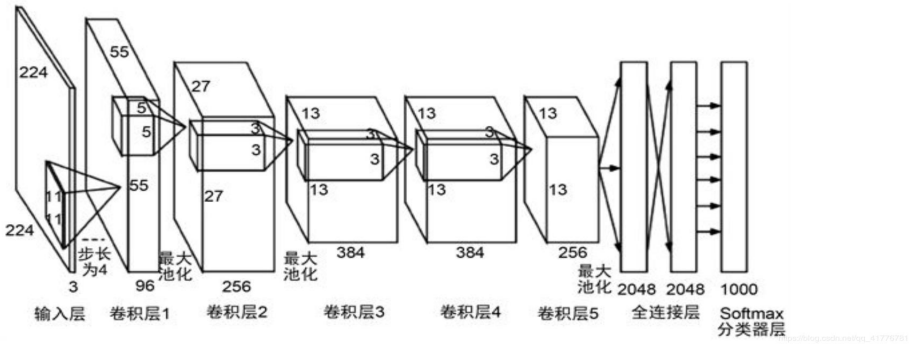
AlexNet：

背景：

2012 年，深度学习三巨头之一的 Geoffrey Hinton 的学生 Alex Krizhevsky 率先提出了 AlexNet（ImageNet Classification with Deep Convolutional Neural Networks，2012），

并在当年度的 ILSVRC（ImageNet 大规模视觉挑战赛）以显著的优势获得当届冠军，top-5 的错误率降至了 16.4%，相比于第二名 26.2% 的错误率有了极大的提升。这一成绩引起了学界和业界的极大关注，计算机视觉也开始逐渐进入深度学习主导的时代。

框架构成：



卷积层 C1

该层的处理流程是： 卷积-->ReLU-->池化-->归一化。

卷积，输入是 227×227

，使用 96 个 11×11×3 的卷积核，得到的 FeatureMap 为 55×55×96

- 。
- ReLU，将卷积层输出的 FeatureMap 输入到 ReLU 函数中。
- 池化，使用 3×3

步长为 2 的池化单元（重叠池化，步长小于池化单元的宽度），输出为 27×27×96（(55-3)/2+1=27

-)
- 局部响应归一化，使用 k=2,n=5,α=10-4,β=0.75

进行局部归一化，输出的仍然为 27×27×96，输出分为两组，每组的大小为 27×27×48

-

卷积层 C2

该层的处理流程是：卷积-->ReLU-->池化-->归一化

- 卷积，输入是 2 组 27×27×48

。使用 2 组，每组 128 个尺寸为 5×5×48 的卷积核，并作了边缘填充 padding=2，卷积的步长为 1. 则输出的 FeatureMap 为 2 组，每组的大小为 27×27 times128. （(27+2* 2-5)/1+1=27 ）

- ReLU，将卷积层输出的 FeatureMap 输入到 ReLU 函数中
- 池化运算的尺寸为 13×13

，步长为 2，池化后图像的尺寸为(27-3)/2+1=13，输出为 13×13×256

- • 局部响应归一化，使用 k=2,n=5,α=10-4,β=0.75

进行局部归一化，输出的仍然为 13×13×256，输出分为 2 组，每组的大小为 13×13×128

-

卷积层 C3

该层的处理流程是： 卷积-->ReLU

- 卷积，输入是 13×13×256

，使用 2 组共 384 尺寸为 3×3×256 的卷积核，做了边缘填充 padding=1，卷积的步长为 1.则输出的 FeatureMap 为 13×13 times384

-

ReLU，将卷积层输出的 FeatureMap 输入到 ReLU 函数中

卷积层 C4

该层的处理流程是： 卷积-->ReLU

该层和 C3 类似。

- 卷积，输入是 13×13×384

，分为两组，每组为 13×13×192.使用 2 组，每组 192 个尺寸为 3×3×192 的卷积核，做了边缘填充 padding=1，卷积的步长为 1.则输出的 FeatureMap 为 13×13 times384，分为两组，每组为 13×13×192

-

ReLU，将卷积层输出的 FeatureMap 输入到 ReLU 函数中

卷积层 C5

该层处理流程为：卷积-->ReLU-->池化

- 卷积，输入为 13×13×384

，分为两组，每组为 13×13×192。使用 2 组，每组为 128 尺寸为 3×3×192 的卷积核，做了边缘填充 padding=1，卷积的步长为 1.则输出的 FeatureMap 为 13×13×256

- - ReLU，将卷积层输出的 FeatureMap 输入到 ReLU 函数中
- 池化，池化运算的尺寸为 3×3，步长为 2，池化后图像的尺寸为 (13-3)/2+1=6

,即池化后的输出为 6×6×256

-

全连接层 FC6

该层的流程为：（卷积）全连接 -->ReLU -->Dropout

卷积->全连接： 输入为 6×6×256

,该层有 4096 个卷积核，每个卷积核的大小为 6×6×256。由于卷积核的尺寸刚好与待处理特征图（输入）的尺寸相同，即卷积核中的每个系数只与特征图（输入）尺寸的一个像素值相乘，一一对应，因此，该层被称为全连接层。由于卷积核与特征图的尺寸相同，卷积运算后只有一个值，因此，卷积后的像素层尺寸为 4096×1×1

- ，即有 4096 个神经元。

ReLU,这 4096 个运算结果通过 ReLU 激活函数生成 4096 个值

Dropout,抑制过拟合，随机的断开某些神经元的连接或者是不激活某些神经元

全连接层 FC7

流程为：全连接-->ReLU-->Dropout

- 全连接，输入为 4096 的向量

ReLU,这 4096 个运算结果通过 ReLU 激活函数生成 4096 个值

Dropout,抑制过拟合，随机的断开某些神经元的连接或者是不激活某些神经元

- **输出层**

第七层输出的 4096 个数据与第八层的 1000 个神经元进行全连接，经过训练后输出 1000 个 float 型的值，这就是预测结果。

AlexNet 参数数量

卷积层的参数 = 卷积核的数量 * 卷积核 + 偏置

C1：96 个 11×11×3

的卷积核，96×11×11×3+96=34848

- C2：2 组，每组 128 个 5×5×48

的卷积核，(128×5×5×48+128)×2=307456

- C3：384 个 3×3×256

的卷积核，3×3×256×384+384=885120

- C4：2 组，每组 192 个 3×3×192

的卷积核，(3×3×192×192+192)×2=663936

- C5：2 组，每组 128 个 3×3×192

的卷积核，(3×3×192×128+128)×2=442624

- FC6：4096 个 6×6×256

的卷积核，6×6×256×4096+4096=37752832

- FC7：4096* 4096+4096=16781312
- output：4096* 1000=4096000

应用：识别图像以及对于图像进行分类。

优点：更深的网络结构

使用层叠的卷积层，即卷积层+卷积层+池化层来提取图像的特征

使用 Dropout 抑制过拟合

使用数据增强 Data Augmentation 抑制过拟合

使用 Relu 替换之前的 sigmoid 的作为激活函数

多 GPU 训练

缺点：如果移除一个卷积层，那么整个 AlexNet 网络都会退化

没有对该框架进行实践操作。

VGG：

背景：2014 年，牛津大学计算机视觉组（Visual Geometry Group）和 Google DeepMind 公司的研究员一起研发出了新的深度卷积神经网络：VGGNet，并取得了 ILSVRC2014 比赛分类项目的第二名（第一名是 GoogLeNet，也是同年提出的）和定位项目的第一名。

VGGNet 探索了卷积神经网络的深度与其性能之间的关系，成功地构筑了 16~19 层深的卷积

神经网络，证明了增加网络的深度能够在一定程度上影响网络最终的性能，使错误率大幅下降，同时拓展性又很强，迁移到其它图片数据上的泛化性也非常好。到目前为止，VGG 仍然被用来提取图像特征。

VGGNet 可以看成是加深版本的 AlexNet，都是由卷积层、全连接层两大部分构成。

优点：

1、**结构简洁**

VGG 由 5 层卷积层、3 层全连接层、softmax 输出层构成，层与层之间使用 max-pooling（最大化池）分开，所有隐层的激活单元都采用 ReLU 函数。

2、**小卷积核和多卷积子层**

VGG 使用多个较小卷积核（3x3）的卷积层代替一个卷积核较大的卷积层，一方面可以减少参数，另一方面相当于进行了更多的非线性映射，可以增加网络的拟合/表达能力。

小卷积核是 VGG 的一个重要特点，虽然 VGG 是在模仿 AlexNet 的网络结构，但没有采用 AlexNet 中比较大的卷积核尺寸（如 7x7），而是通过降低卷积核的大小（3x3），增加卷积子层数来达到同样的性能（VGG：从 1 到 4 卷积子层，AlexNet：1 子层）。

3、**小池化核**

相比 AlexNet 的 3x3 的池化核，VGG 全部采用 2x2 的池化核。

4、通道数多

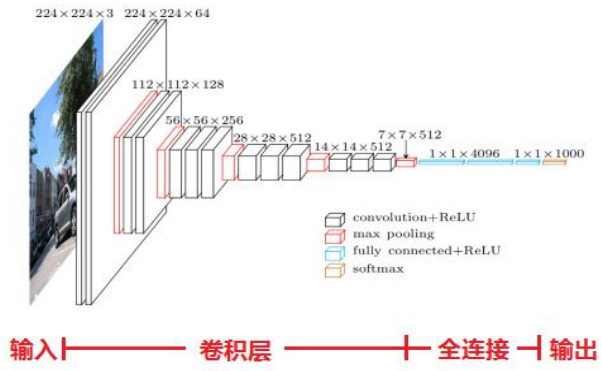
VGG 网络第一层的通道数为 64，后面每层都进行了翻倍，最多到 512 个通道，通道数的增加，使得更多的信息可以被提取出来。

5、层数更深、特征图更宽

由于卷积核专注于扩大通道数、池化专注于缩小宽和高，使得模型架构上更深更宽的同时，控制了计算量的增加规模。

缺点：GG 耗费更多计算资源，并且使用了更多的参数（这里不是 3x3 卷积的锅），导致更多的内存占用（140M）。其中绝大多数的参数都是来自于第一个全连接层。

结构：



1、输入 224x224x3 的图片，经 64 个 3x3 的卷积核作两次卷积+ReLU，卷积后的尺寸变为 224x224x64

2、作 max pooling（最大化池化），池化单元尺寸为 2x2（效果为图像尺寸减半），池化后

的尺寸变为 112x112x64

3、经 128 个 3x3 的卷积核作两次卷积+ReLU，尺寸变为 112x112x128

4、作 2x2 的 max pooling 池化，尺寸变为 56x56x128

5、经 256 个 3x3 的卷积核作三次卷积+ReLU，尺寸变为 56x56x256

6、作 2x2 的 max pooling 池化，尺寸变为 28x28x256

7、经 512 个 3x3 的卷积核作三次卷积+ReLU，尺寸变为 28x28x512

8、作 2x2 的 max pooling 池化，尺寸变为 14x14x512

9、经 512 个 3x3 的卷积核作三次卷积+ReLU，尺寸变为 14x14x512

10、作 2x2 的 max pooling 池化，尺寸变为 7x7x512

11、与两层 1x1x4096，一层 1x1x1000 进行全连接+ReLU（共三层）

12、通过 softmax 输出 1000 个预测结果

应用:可以应用在现实生活中的各种识别上，比如各种品种的花卉识别，以及图形识别。

GoogleNet：

背景：2014 年，GoogLeNet 和 VGG 是当年 ImageNet 挑战赛(ILSVRC14)的双雄，GoogLeNet 获得了第一名、VGG 获得了第二名，这两类模型结构的共同特点是层次更深了。VGG 继承了 LeNet 以及 AlexNet 的一些框架结构，而 GoogLeNet 则做了更加大胆的网络结构尝试，虽然深度只有 22 层，但大小却比 AlexNet 和 VGG 小很多，GoogleNet 参数为 500 万个，AlexNet 参数个数是 GoogleNet 的 12 倍，VGGNet 参数又是 AlexNet 的 3 倍，因此在内存或计算资源有限时，GoogleNet 是比较好的选择；从模型结果来看，GoogLeNet 的性能却更加优越。

为何能够提升性能：

一般来说，提升网络性能最直接的办法就是增加网络深度和宽度，深度指网络层次数量、宽度指神经元数量。**但这种方式存在以下问题：**

（1）参数太多，如果训练数据集有限，很容易产生过拟合；

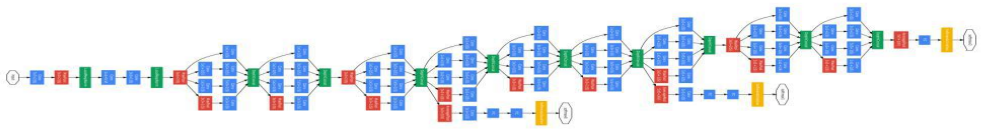
（2）网络越大、参数越多，计算复杂度越大，难以应用；

（3）网络越深，容易出现梯度弥散问题（梯度越往后穿越容易消失），难以优化模型。

如何解决：

解决这些问题的方法当然就是在增加网络深度和宽度的同时减少参数，为了减少参数，自然就想到将全连接变成稀疏连接。但是在实现上，全连接变成稀疏连接后实际计算量并不会有质的提升，因为大部分硬件是针对密集矩阵计算优化的，稀疏矩阵虽然数据量少，但是计算所消耗的时间却很难减少。

结构：



0、输入

原始输入图像为 224x224x3，且都进行了零均值化的预处理操作（图像每个像素减去均值）。

1、第一层（卷积层）

使用 7x7 的卷积核（滑动步长 2，padding 为 3），64 通道，输出为 112x112x64，卷积后进行 ReLU 操作

经过 3x3 的 max pooling（步长为 2），输出为((112 - 3+1)/2)+1=56，即 56x56x64，再进行 ReLU 操作

2、第二层（卷积层）

使用 3x3 的卷积核（滑动步长为 1，padding 为 1），192 通道，输出为 56x56x192，卷积后进行 ReLU 操作

经过 3x3 的 max pooling（步长为 2），输出为((56 - 3+1)/2)+1=28，即 28x28x192，再进行 ReLU 操作

3a、第三层（Inception 3a 层）

分为四个分支，采用不同尺度的卷积核来进行处理

（1）64 个 1x1 的卷积核，然后 RuLU，输出 28x28x64

（2）96 个 1x1 的卷积核，作为 3x3 卷积核之前的降维，变成 28x28x96，然后进行 ReLU 计算，再进行 128 个 3x3 的卷积（padding 为 1），输出 28x28x128

（3）16 个 1x1 的卷积核，作为 5x5 卷积核之前的降维，变成 28x28x16，进行 ReLU 计算后，再进行 32 个 5x5 的卷积（padding 为 2），输出 28x28x32

（4）pool 层，使用 3x3 的核（padding 为 1），输出 28x28x192，然后进行 32 个 1x1 的卷积，输出 28x28x32。

将四个结果进行连接，对这四部分输出结果的第三维并联，即 64+128+32+32=256，最终输出 28x28x256

3b、第三层（Inception 3b 层）

（1）128 个 1x1 的卷积核，然后 RuLU，输出 28x28x128

（2）128 个 1x1 的卷积核，作为 3x3 卷积核之前的降维，变成 28x28x128，进行 ReLU，再进行 192 个 3x3 的卷积（padding 为 1），输出 28x28x192

（3）32 个 1x1 的卷积核，作为 5x5 卷积核之前的降维，变成 28x28x32，进行 ReLU 计算后，再进行 96 个 5x5 的卷积（padding 为 2），输出 28x28x96

（4）pool 层，使用 3x3 的核（padding 为 1），输出 28x28x256，然后进行 64 个 1x1 的卷积，输出 28x28x64。

将四个结果进行连接，对这四部分输出结果的第三维并联，即 128+192+96+64=480，最终输出输出为 28x28x480

第四层（4a,4b,4c,4d,4e）、第五层（5a,5b）……，与 3a、3b 类似，在此就不再重复。