卷积神经网络

Kuangnan Fang

Email: xmufkn@xmu.edu.cn

2022 年 5 月 16 日

什么是卷积神经网络？

- 卷积神经网络（Convolutional Neural Network，CNN或ConvNet）是一种具有局部连接、权重共享等特性的深层前馈神经网络。
- 卷积神经网络是受生物学上感受野机制的启发而提出的。感受野（Receptive Field）机制主要是指听觉、视觉等神经系统中一些神经元的特性，即神经元只接受其所支配的刺激区域内的信号。
- 在视觉神经系统中，视觉皮层中的神经细胞的输出依赖于视网膜上的光感受器。视网膜上的光感受器受刺激兴奋时，将神经冲动信号传到视觉皮层，但不是所有视觉皮层中的神经元都会接受这些信号。一个神经元的感受野是指视网膜上的特定区域，只有这个区域内的刺激才能够激活该神经元。

卷积神经网络的发展历程

对卷积神经网络的研究始于二十世纪80至90年代，时间延迟网络和LeNet-5是最早出现的卷积神经网络；在二十一世纪后，随着深度学习理论的提出和数值计算设备的改进，卷积神经网络得到了快速发展。卷积神经网络主要使用在图像和视频分析的各种任务（比如图像分类、人脸识别、物体识别、图像分割等）上，其准确率一般也远远超出了其他的神经网络模型。近年来卷积神经网络也广泛地应用到自然语言处理、推荐系统等领域。

卷积神经网络的基本结构

一个卷积神经网络主要包含5层基本结构：输入层、卷积层、池化层、全连接层、输出层。

- 输入层：具体数据的输入。如图像分类问题中图像像素矩阵的输入。在卷积神经网络中，输入/输出数据称之为特征图（feature map）。
- 卷积层：使用卷积核进行特征提取和特征映射，获得更多图像的抽象特征。卷积运算的主要目的是使原信号特征增强，并降低噪音。
- 池化层：进行下采样降维，减少网络中参数。
- 全连接层：在尾部进行拟合，减少特征信息的损失，为后续分类任务做准备。
- 输出层：得到输入样本所属种类的概率分布情况。

其中，卷积层、激活层和池化层可叠加重复使用，这是卷积神经网络的核心结构。

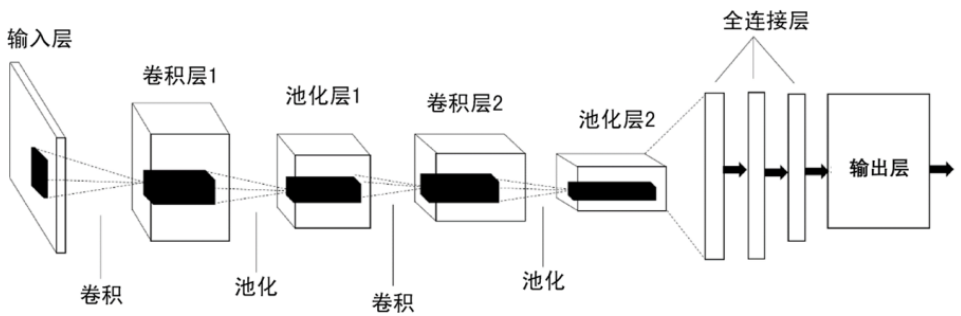


图 1: 卷积神经网络结构示意图

卷积层

卷积（Convolutional）是一种特殊的线性运算，用来代替一般的矩阵乘法运算。在图像处理中，卷积就是利用一个卷积核逐行逐列地扫描像素矩阵，并与像素矩阵中的元素相乘后相加，由此得到一个新的像素矩阵，这个计算过程就称为卷积。如图2中的例子，将一个矩阵使用卷积核转换成一个的输出矩阵。

卷积操作就是用的卷积核从上至下逐行扫描的输入矩阵，每扫描一次卷积核就与相应位置的数值进行线性组合。从而形成了一个新的图像矩阵，称为输出特征图（Output Feature Map），如上图中的输出矩阵。但输出矩阵的维度不尽相同，和卷积核的维度与卷积方式有关。卷积核的大小一般为3 × 3或者5 × 5的矩阵，而不同方法的卷积计算会在下面具体讨论。

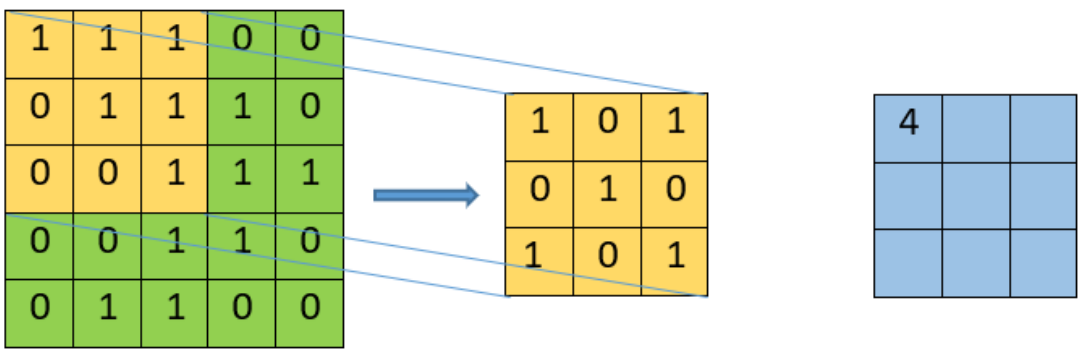


图 2: 卷积计算示例（张量矩阵、卷积核与输出矩阵）

卷积运算原理

卷积运算共有三种不同类型：full卷积、same卷积和valid卷积。下面以3×3的二维张量x和2×2的卷积核K分别演示三种不同卷积计算方法。

- full卷积的计算过程为：卷积核在张量矩阵上按照先行后列的顺序移动，对应位置元素相乘并求和。只要矩阵和卷积核元素有一个位置重叠就要计算，并将落在矩阵外的元素全部视为0。计算过程如图3所示。

卷积运算原理

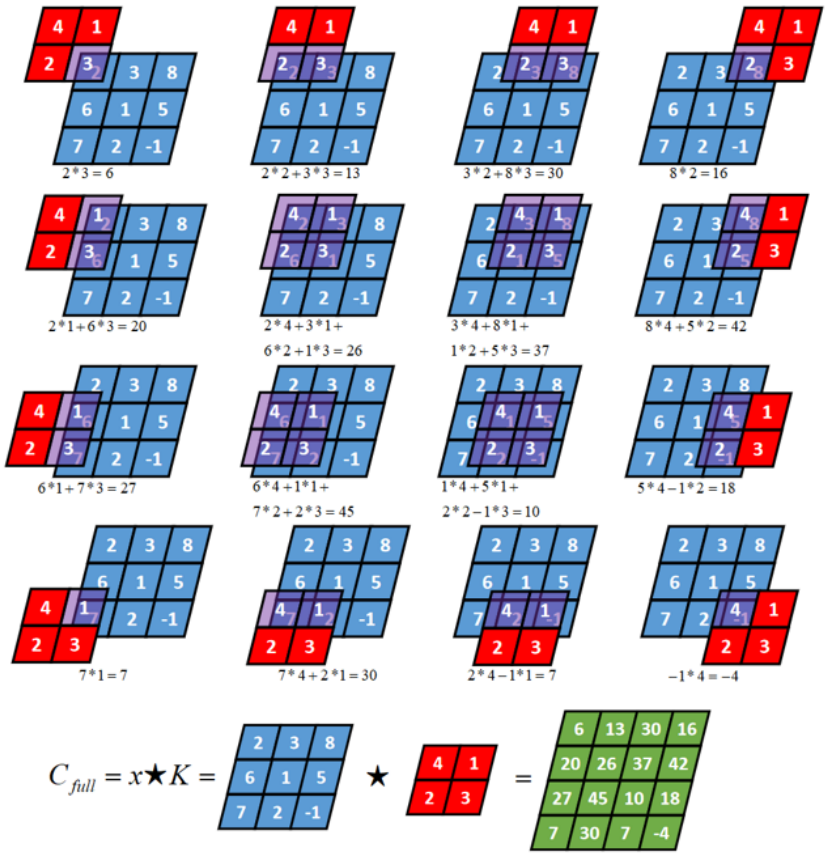


图 3: Full卷积计算方式和结果

这种卷积计算的实际意义未可知，但对于某些数据是有效的，对有些却无效。full卷积作为一种简单直观的基础算法是需要了解的。

卷积运算原理

- same卷积是在实际应用中较为常见的类型。same卷积计算可以保证卷积前后矩阵成同样的维度（步长为1时成立）。在进行same卷积时，首先要为卷积核指定一个起始点，然后按照先行后列的顺序遍历矩阵，对应位置的值相乘并求和。假设卷积核的高为H，宽为W，则起始点可按照图4中的规则确定。

高度（H）	宽度（W）	起点位置
偶数	偶数	$\frac{H-2}{2},\frac{W-2}{2}$
偶数	奇数	$\frac{H-2}{2},\frac{W-1}{2}$
奇数	偶数	$\frac{H-1}{2},\frac{W-2}{2}$
奇数	奇数	$\frac{H-1}{2},\frac{W-1}{2}$

图 4: 起始位置确定规则表

卷积运算原理

按照公式，2×2的卷积核的起始点为$(0,0)$。当卷积核的中心与张量矩阵的边角重合时，开始做卷积运算。此时卷积核K的运动范围比full模式小了一圈。计算过程如图5所示。

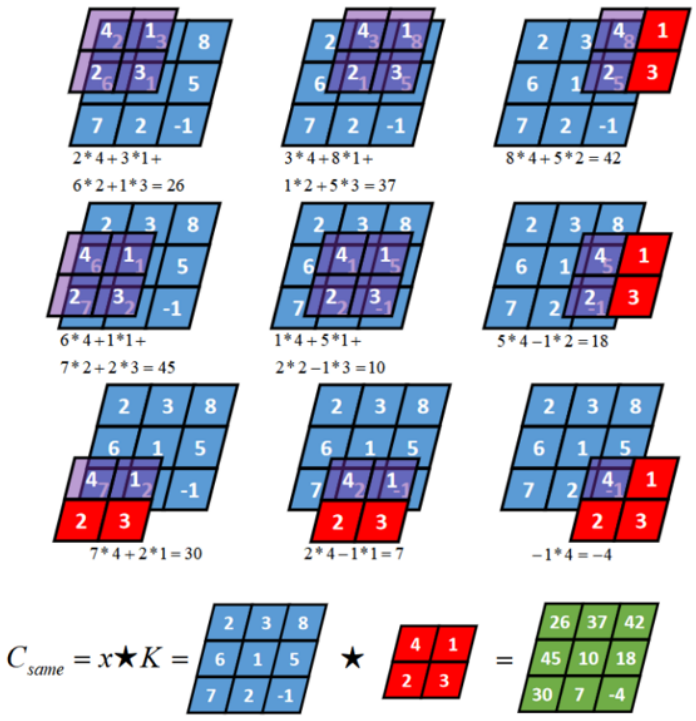


图 5: same卷积计算方式和结果

卷积运算原理

- valid卷积:上述的两种卷积方式都会有卷积核延伸至矩阵之外的情况，因此会采取用0来填充后才可以进行计算。
  而valid卷积仅考虑卷积核能完全覆盖的部分（即卷积核在矩阵内部移动）

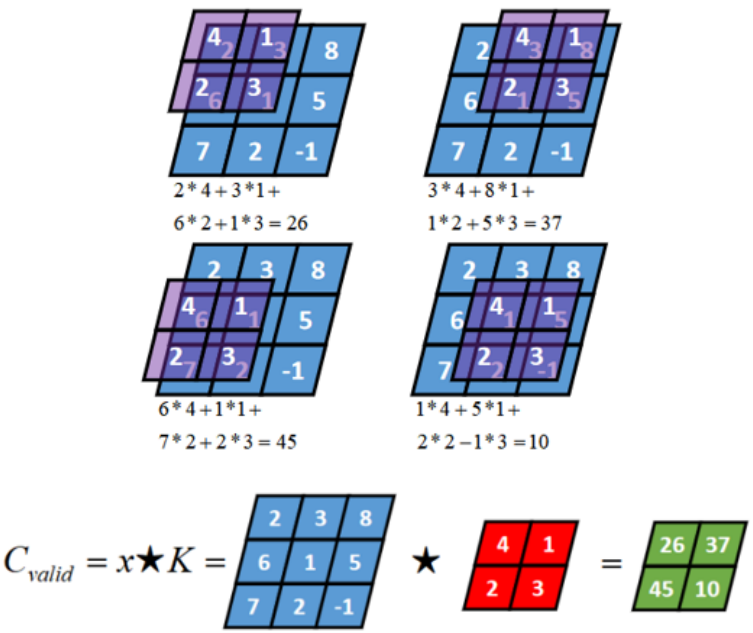


图 6: valid卷积的计算过程

卷积运算原理

可以发现，经过valid卷积后的矩阵维度变小了，似乎在数学上十分合理，但也带来了一些问题。维度的变化受到卷积核和原始矩阵维度的影响，因此改变矩阵维度后输出结果也会受到影响，因此使用valid卷积需要注意矩阵维度变化所带来的影响。下图2-6就生动的表明了不同卷积方式的差异。

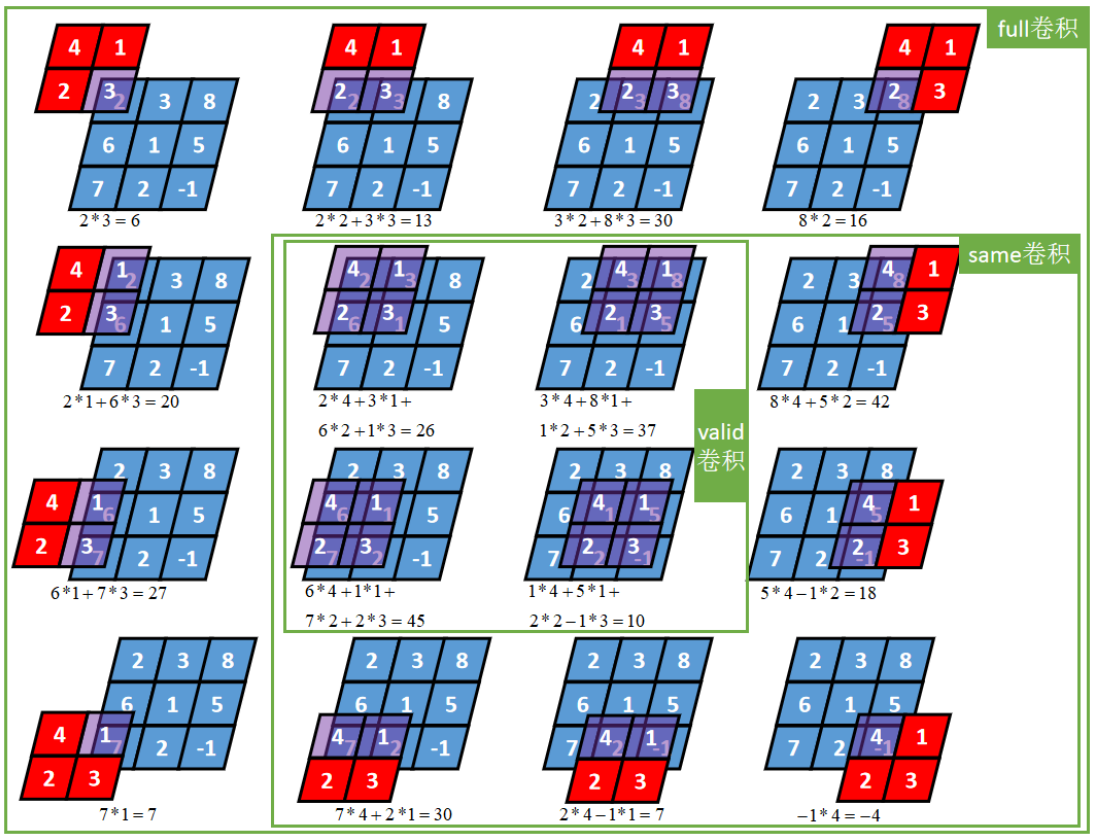


图 7: 不同卷积方式计算过程的不同

多通道卷积原理

上文以单通道图像的像素矩阵为例，简要介绍了full卷积、same卷积和valid卷积。但是在实际应用中，更为常见的是三通道像素矩阵的彩色图像，接下来介绍基本的多通道卷积原理。

基本的多通道卷积:此处以3行3列2深度的三维张量与一个2行2列2深度的三维卷积核为例。在卷积计算过程中，像素矩阵的第一层对准卷积核的第一层，像素矩阵的第二层对准卷积核的第二层，再将重叠部分相乘并求和。通过这种计算方法，将所有通道所得的结果加和，无论有多少个通道，经过卷积计算后都会变成一个单通道矩阵。因此，多通道卷积核可以将多通道的像素矩阵变成单通道。

- 1.单个张量与多个卷积核的卷积,将一个多通道矩阵变为一个单通道矩阵时，会产生大量的信息损失，因此，在实际应用中，很少只使用一个卷积核进行卷积，通常使用多个卷积核进行卷积来提取图像的特征，再将多个通道提取的特征叠加，得到一个新的多通道图像。可以看到，一个张量和多个卷积核的卷积事实上是在重复之前的基本多通道卷积，使用每一个卷积核做一次基本多通道卷积，然后将结果合并成一个深度为3的张量。在面对不同情况时，可以根据模型精度调整多通道矩阵卷积生成的矩阵。

多通道卷积原理

- 2.在每一通道上分别卷积:图8介绍卷积核如何在张量的每一通道上分别卷积。与基本的多通道卷积类似，在计算过程中，像素矩阵的第一层对准卷积核的第一层，像素矩阵的第二层对准卷积核的第二层。但不将每一层的结果相加，而是每个通道分别计算卷积结果。使用这种卷积计算方式，可以保证原始矩阵的通道数保持不变。

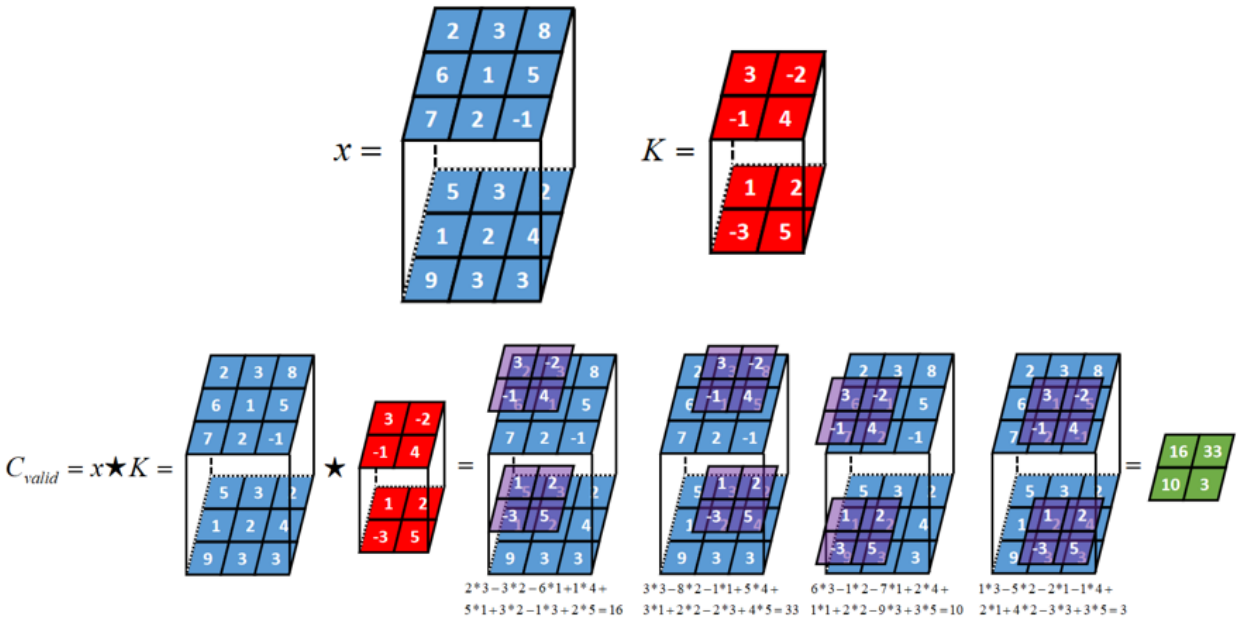


图 8: 在每一通道上分别卷积

多通道卷积原理

- 3.单个张量与多个卷积核在通道上分别卷积同样地，和单个张量与多个卷积核的卷积的方式类似，如图，仍然以3行3列2深度的三维张量与三个2行2列2深度的三维卷积核采用valid卷积为例，分别卷积就可以得到三个2行2列的输出矩阵，再将它们在深度方向上拼接，最终就得到了一个2行2列3深度的输出矩阵。

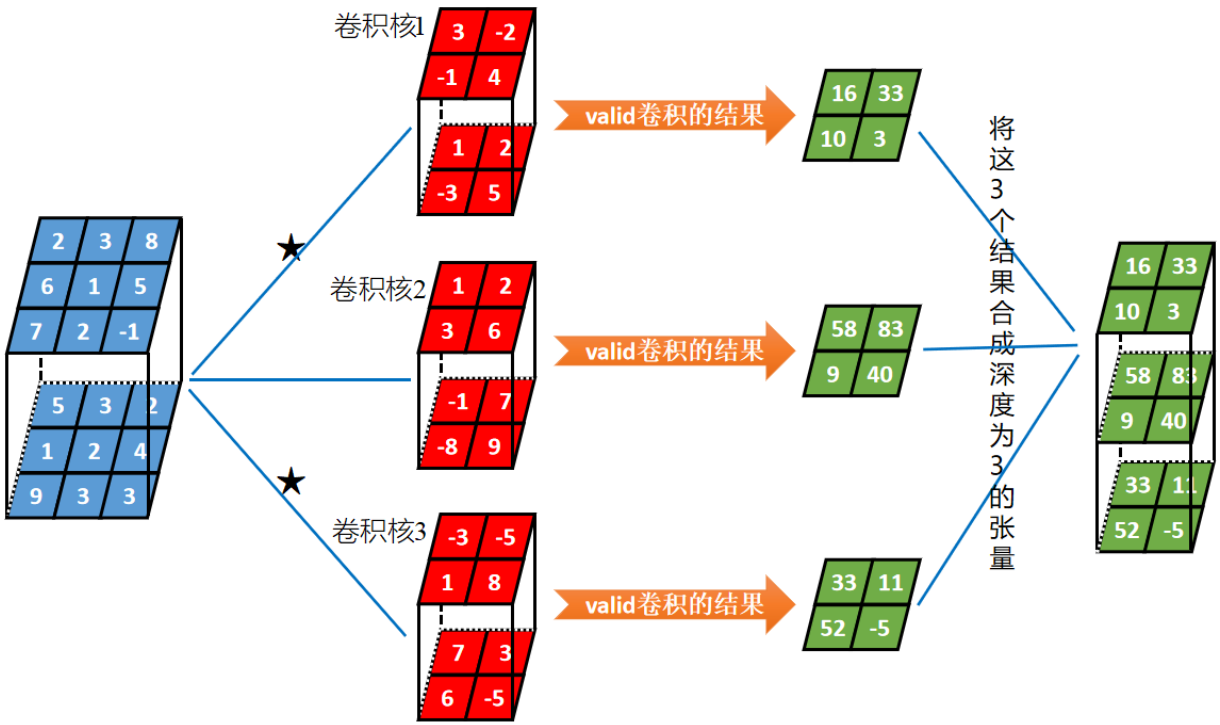


图 9: 单个张量与多个卷积核在通道上分别卷积

卷积运算的性质

卷积运算有三个重要的性质，分别是稀疏连接、参数共享和平移不变性。

- 1.卷积运算的稀疏连接我们已经清楚地了解到，在全连接神经网络中，上一层的每一个神经元都与下一层每一个神经元完全连接，这种连接方式会产生巨量参数。而卷积运算采用稀疏连接的方式。假设神经网络有个输入和个输出，全连接网络需要个参数，而稀疏连接则是限制每一个输出仅连接到部分输入，从而减少网络的参数量。通过稀疏连接，可以达到减少参数的目的，这样做有以下好处：一是降低计算复杂度与成本；二是减少发生过拟合的风险。

卷积运算的性质

- 2.卷积运算的参数共享参数共享则是指相同的参数可以被运用在多个函数中。相对于全连接网络中每个神经元都需要学习一个单独的参数集合这种复杂方式，卷积运算中，每一层神经元只需要学习一个卷积核大小的参数，大大降低了计算复杂度。
- 3.卷积运算的平移不变性卷积元算的平移不变性则非常常用。在处理图像数据时，当卷积在图像中的某个位置学习到存在的特征时，它可以在之后的任何地方识别这个特征。而对于全连接网络而言，如果特征出现在新的位置，只能重新学习。这样就大大提升网络的训练效率。

卷积运算的意义

（大概了解过深度学习发展历程的人都知道，卷积神经网络的应用最早是在LeCun所创建用来识别手写数字的LeNet-5。后来又被AlexNet应用在ImageNet比赛中，拔群的识别效果让卷积神经网络成了AI的热点。也随着研究的逐渐深入，CNN算法中的黑箱性也得到了一定程度的解释。

在图像识别的例子中，卷积运算并不复杂，但是为什么要进行卷积操作？想要了解这一点，我们首先需要明确人类识别物体时是先对图像边缘开始敏感的，通过对物体边界的识别然后将其组装成一个整体，这就是人可以区分物体的基本原理。所以说，模拟人类识别物体的机制，神经网络算法首先要识别物体的边界，再进一步的区分和细化对物体的认识。而卷积层就刚好是做到了这一点。

卷积运算的意义

图 10: 卷积效果图

图10（b）是将图10（a）经过卷积核滑动整个图像后出来的效果，可以看到经过经过卷积滑动后，图像中物体的边界被强化，而其他部分则相应被忽略，这个卷积核很好的提取出图像中物体的轮廓。这也就解释了为什么有必要使用多个卷积核对同一个像素矩阵分别作卷积：不同的卷积核可以提取出图像中不同物体的边界，这也为我们从视觉图像检测物体提供了凭据。越多的边界被强化，也就为卷积神经网络提供了越多的特征。

池化层

池化（Pooling）操作是对卷积得到结果的进一步处理，池化操作会将平面内某一位置及其相邻位置的特征值进行统计汇总，并把汇总后的结果作为这一位置在该平面内的值输出。池化可以将输入张量每一位置的矩形领域内的最大值或者平均值作为该平面内的输出值，按照汇总的方式可以区分为最大值池化或者平均值池化。池化操作在图像处理中类似减少参数量的操作。

池化层的作用是降低参数，而降低参数的方法当然也只有删除参数了。一般我们有最大池化和平均池化，而最大池化更为常用。需要注意的是，池化层一般放在卷积层后面，所以池化层池化的是卷积层的输出。下面分别介绍不同类型的池化操作。

same池化

same池化可以分为same最大值池化和same平均值池化。最大值池化和平均值池化的步骤基本一致，仅在数据汇总时的操作分别为取最大值或取均值。因此，接下来仅展示单通道的same最大值池化和多通道的same平均值池化。

- same最大值池化：图11表示了一个3×3的张量X和2×2的池化窗口。

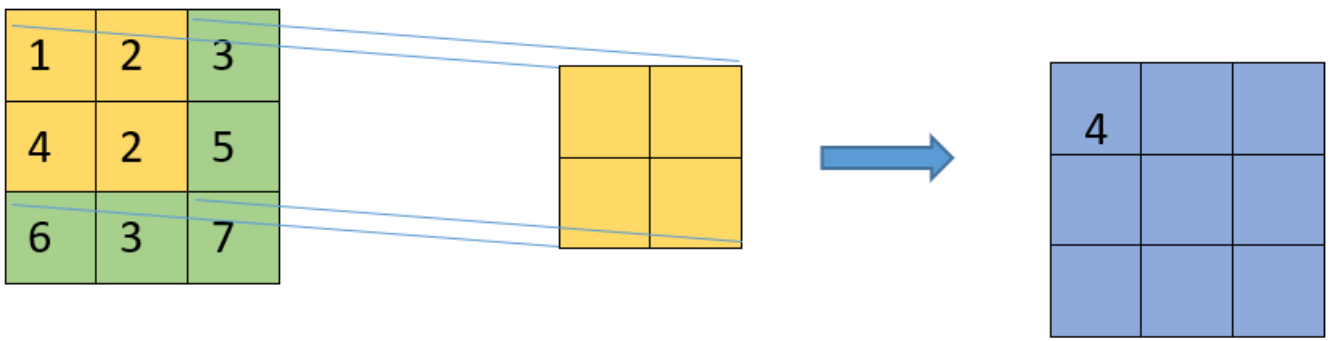


图 11: 张量X、池化窗口和输出矩阵

same池化

- 与same卷积类似，same最大值池化也需要指定起点，其指定起始点——与same卷积的起始点规则类似。假设池化窗口的高等于H，宽等于W，起始点的确定规则如下图12所示。

高度（H）	宽度（W）	起点位置
偶数	偶数	$\frac{H-2}{2},\frac{W-2}{2}$
偶数	奇数	$\frac{H-2}{2},\frac{W-1}{2}$
奇数	偶数	$\frac{H-1}{2},\frac{W-2}{2}$
奇数	奇数	$\frac{H-1}{2},\frac{W-1}{2}$

图 12: 张量X、池化窗口和输出矩阵

same池化

- 将起始点按照先行后列的顺序在输入张量X上进行移动，移动方式与same卷积一致。然后取池化窗口内的最大值作为该窗口区域的输出值，将得到的输出值组合起来就得到了池化操作最后的结果。接下来我们通过示例来展示same最大值池化的计算过程，如图13所示：

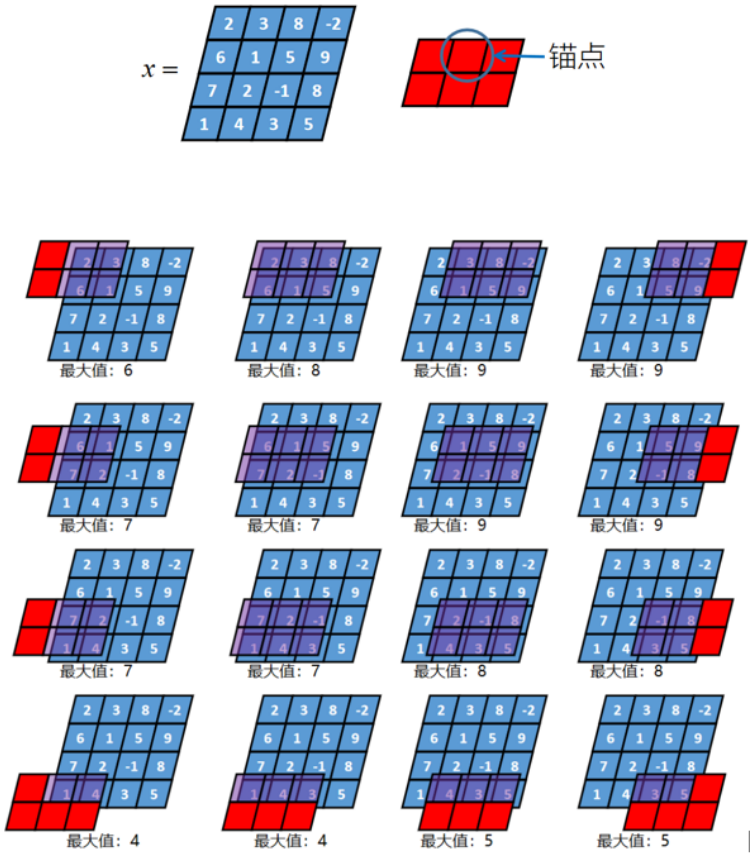


图 13: same最大值池化的计算过程

same池化

- 多通道张量的same最大值池化:下面以池化窗口沿行和沿列的移动步长均为2，给出最大值池化的例子。最终结果如图14所示。

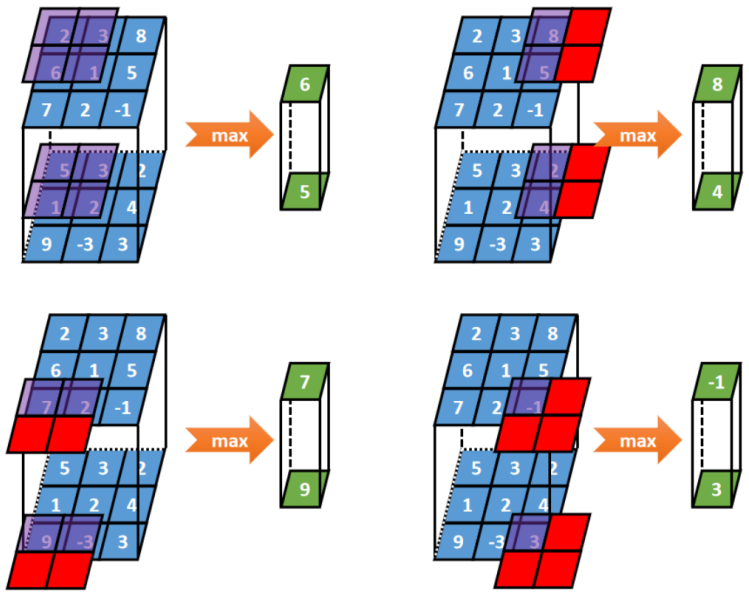


图 14: same最大值池化过程

vaild 池化

与same池化不同，valid池化的池化窗口仅在张量内部移动。以图15所示的张量和池化窗口为例，介绍valid最大值池化。

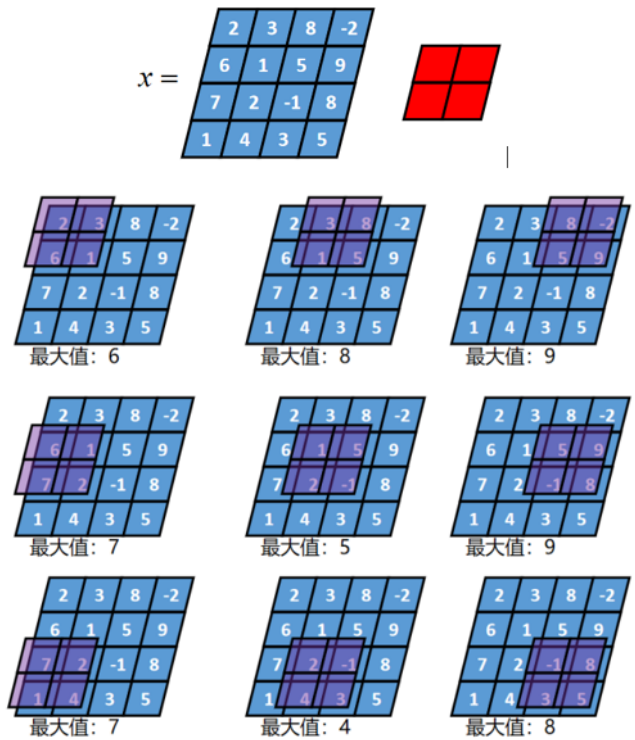


图 15: valid最大值池化的过程

vaild 池化

对比卷积与池化，它们有两个核心区别：

- （1）卷积核的权重需要在人为设定或计算过程中，通过机器学习算法自动优化得到，是一个位置的参数；而池化仅仅求最大值，没有未知参数需要估计，也不需要参数优化的过程。因此，相对而言池化操作十分简单。
- （2）不管输入的像素矩阵有多少通道，只要进行卷积操作，一个卷积核参与计算只会产生一个通道；而池化是分层运算，输出的像素矩阵的通道取决于输入像素矩阵的通道数。

归一化层

批归一化（Batch Normalization，BN）是由Google的DeepMind团队提出的在深度网络各层之间进行数据批量归一化的算法。

在神经网络中, 数据的分布会对训练效果会产生影响。而由于训练过程中各层输入的分布随着前几层参数的变化而变化，这就使得训练深度神经网络变得复杂。以Tanh激活函数为例，随着$|x|$的增加，会趋于0，也就是说神经网络对于较大的输入值不敏感。在回归分析中，人们经常对数据进行归一化处理，得到均值为0和标准差为1的特征。在训练神经网络之前，可以对数据进行归一化处理，使得输入的变化范围不会太大,让输入值经过激励函数的敏感部分,同时保持训练和测试数据的分布相同。批归一化就是将各层的数据强制拉回均值为0，方差为1的分布，使得各层的分布一致。而在神经网络内部，每一层我们都会有输入和输出，经过网络每一层计算后的数据，它们的分布是不同的。这就需要批归一化将这个标准化过程从输入层扩展到所有隐藏层。

BN算法

输入：每个 $Mini-Batch$ 中输入 x 的值 $B=\{x_1,\ldots,x_m\}$；

需要学习的参数：γ,β

输出：BN 结果 $\{y_i=BN_{\gamma,\beta}(x_i)\}$

$$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{//计算批量数据均值}$$

$$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i-\mu_B)^2 \qquad \text{//计算批量数据方差}$$

$$\hat{x}_i \leftarrow \frac{x_i-\mu_B}{\sqrt{\sigma_B^2+\varepsilon}} \qquad \text{//归一化数据，}\ \varepsilon \text{ 是很小的常数}$$

$$y_i \leftarrow \gamma\hat{x}_i+\beta \equiv BN_{\gamma,\beta}(x_i) \qquad \text{//尺度变化与偏移}$$

BN 算法总体可以分为两个过程：第一个过程是对数据进行批归一化，这是为了使分布一致。图像数据有可能会出现方差为0的情况，因此为了计算的正常进行，要在分母上加上一个ε（一个很小的、数值由Tensorflow固定的正数）。第二个过程是对批归一化后的数据进行一定的缩放和平移，这一步是因为从激活函数的角度来说，如果各隐藏层的输入均值在靠近0的区域即处于激活函数的线性区域，将不利于训练好的非线性神经网络，得到的模型效果也不会太好，所以增加两个调节参数（scale和shift）。这两个参数是通过学习得到的，经过线性变换后可使各隐藏层输入的均值和方差为任意值。

经典的卷积神经网络

从LeNet-5在手写数字识别上的成功，到AlexNet在ImageNet图像分类大赛中的一鸣惊人，再到ResNet另辟蹊径地提出“shortcut connection”，卷积神经网络发展到现在，网络层数不断加深，其能力也在不断加强，本节将介绍卷积神经网络发展历程中经典的几个网络。

LeNet-5是一个简单有效的卷积神经网络，奠定了现在卷积神经网络的基本结构，但是它也存在一些缺陷，其过拟合问题导致LeNet-5的泛化能力较差。AlexNet针对LeNet-5的缺陷，引入了dropout操作，一定程度上减轻了过拟合问题，同时，Alexnet还引入了GPU训练，极大地提高了网络训练速度。AlexNet获得了2012年IamgeNet图像分类大赛的冠军，它引领了一波卷积神经网络研究的热潮，是卷积神经网络研究史上的一个里程碑。VGGNet在AlexNet的基础上增加了网络的深度，但是随着网络深度的增加，网络退化问题随之出现，因此VGGNet的网络深度通常设定为16-19层。ResNet在残差单元中设置了“shortcut connection”，能让网络中低层的特征图跳过一些层，直接映射到高层的网络当中。这种独特的残差单元架构在一定程度上缓解了随网络深度增加而引发的退化问题。DenseNet一定程度上借鉴了ResNet的思想，在其设置的DenseBlock中，将在前的层与后续的每一层都进行连接，加强了对特征图的重复利用。

LeNet-5

LeNet-5是一种典型的卷积神经网络的结构，由Yann LeCun在1998年发明，用于MNIST手写数字识别，是第一个成功应用于手写数字识别的卷积神经网络，推动了后续卷积神经网络的发展。

LeNet-5一共有7层网络，不包含输入，每层都包含可训练参数。MNIST中图像的大小为28×28，图像归一化后填充0成为32×32像素的图像，这是为了笔画末端或者拐点等具有一定固定模型的潜在特征能存在于C1层卷积感受野的中间。LeNet-5的网络结构如图16所示。

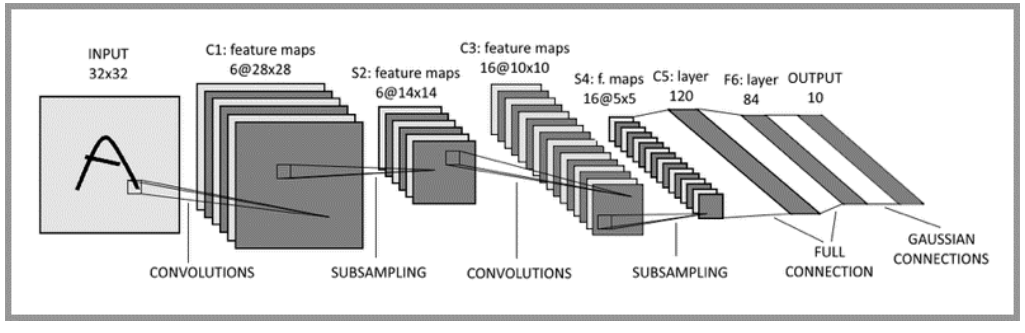


图 16: LeNet-5的网络结构

LeNet-5

- 第一层：输入层。LeNet-5的输入是一个$32 \times 32 \times 1$的1通道灰度图像。
- 第二层：卷积层。输入是$32 \times 32 \times 1$，使用6个大小为5×5、步长为1的卷积核进行valid卷积，得到的特征图（Feature Map）为$28 \times 28 \times 6$（$32 - 5 + 1 = 28$）。
- 第三层：池化层。对第二层的卷积层输出的$28 \times 28 \times 6$的抽象矩阵使用大小为2×2、步长为2的池化单元进行valid池化，输出为$14 \times 14 \times 6$（$(28 - 2)/2 + 1 = 14$）。
- 第四层：卷积层。对第三层的池化层的输出，使用16个大小为5×5、步长为1的卷积核进行valid卷积，得到的特征图为$10 \times 10 \times 16$（$14 - 5 + 1 = 10$）。
- 第五层：池化层。对第四层的卷积层输出的$10 \times 10 \times 16$的抽象矩阵使用大小为2×2、步长为2的池化单元进行valid池化，输出为$5 \times 5 \times 16$（$(10 - 2)/2 + 1 = 5$）。
- 第六层：全连接层。将第五层的池化层输出的矩阵拉成一维向量，这个向量的长度为$5 \times 5 \times 16 = 400$。将该向量经过一个全连接神经网络处理，该全连接神经网络共有2个隐含层，其中输入层有400个神经元，第一个隐含层有120个神经元，第二个隐含层有84个神经元。
- 第七层：输出层。第六层的输出与第七层的10个神经元进行全连接，经过训练后输出10个float型的值，产生的10类所属标签的概率分布情况，用来识别手写数字。

AlexNet

AlexNet由Hinton的团队所提出，以压倒性的优势获得了2012年Imagenet图像分类大赛的冠军，是这项竞赛中第一个使用卷积神经网络的模型。该网络由原论文的第一作者Alex Krizhevsky的名字命名。在AlexNet之后，涌现出了大量的卷积神经网络，可以说AlexNet深刻影响到了卷积神经网络的发展。

AlexNet 的输入是ImageNet 中归一化后的 RGB 图像样本，每张图像的尺寸被裁切到了，AlexNet中包含5个卷积层和3个全连接层，输出为1000类的Softmax层，具体的网络结构如图2-18所示。受当时GPU能力的限制，整个网络模型被分割在两块NVidia GTX580 GPU上运行,并只在一些特定的层通信。

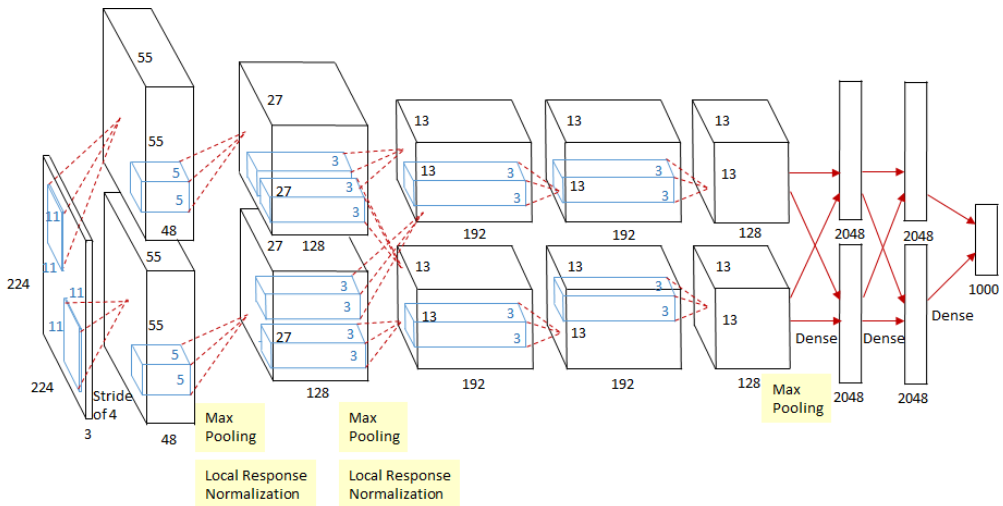


图 17: AlexNet网络结构

AlexNet

AlexNet的网络结构类似于LeNet-5，但AlexNet将卷积神经网络的基本原理应用到更深更宽的网络模型中，其创新点如下：

- 1.相较于以往的卷积神经网络，AlexNet具有更深的网络结构。
- 2.AlexNet使用ReLU函数作为激活函数，并且验证了使用ReLU函数的效果好于sigmoid函数。
- 3. AlexNet使用Dropout与数据增强（Data Augmentation）两种正则化方法来抑制过拟合。在数据增强的处理中，AlexNet中对数据做了以下操作：
  - （1）随机裁剪，对的图片进行随机裁剪到，然后进行水平翻转；测试的时候，对左上、右上、左下、右下、中间分别做了5次裁剪，然后翻转，共10个裁剪，之后对结果求平均；
  - （2）对RGB空间做PCA（主成分分析），然后对主成分做一个的高斯扰动。

AlexNet

- 4. AlexNet采用了多GPU训练的方法。AlexNet使用了2块GPU进行模型的训练很大程度上是受限于当时GPU的性能，而随着GPU性能的提升，对于AlexNet这种小型的卷积神经网络在一块GPU上就可以完成训练。对很多大型神经网络而言，多GPU训练还是主要的训练方式。AlexNet使用的多GPU训练的方法对大型神经网络模型的多GPU训练具有一定的启发作用。
- 5. AlexNet提出了End-End的图像处理的方法，即AlexNet的输入只需要是原始的RGB图像数据。而在以往的图像处理任务中，首先需要对图像进行特征提取的操作。
- 6. AlexNet还提出了局部响应归一化（Local Response Normalization, LRN），但是这个方法已经被BN（Batch Normalization）所替代。

VGGNet

2014年，牛津大学计算机视觉组和Google DeepMind公司的研究员共同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，同年提出）和定位项目的第一名。VGGNet结构简单，模型的泛化能力好，因而受到研究人员的青睐而被广泛使用，到现在依然经常被用作图像的特征提取。
下面介绍VGG网络的基础结构：
在当时，其他卷积神经网络更多使用感受野大的卷积核（如5×5,7×7）。与此不同的是，VGGNet在整个网络中大量使用大小的卷积核（VGG-16中引入卷积核大小为1×1的卷积层）。VGGNet用多个小卷积核的卷积层来替换大卷积核的层，原论文中指出：堆叠2层和3层3×3卷积层（层间没有空间池化）的有效感受野分别为5×5,7×7, 。

VGG网络的基础结构

以用3层3×3卷积层来替换1层7×7卷积层为例，假设通道数为C，3层3×3卷积层的参数个数为$3(3^2C^2) = 27C^2$，而1层7×7卷积层的参数个数为$7^2C^2 = 49C^2$。因此这样堆叠3×3卷积层的好处主要有两点：

- 1.减少网络参数量；
- 2.增加网络层数，提高网络的非线性表达能力。

VGG网络的基础结构

表中展示了6个VGGNet网络结构，层数设置共有四种，分别是11层、13层、16层、19层。为了简洁，表中没有显示ReLU激活函数，但其实际上存在于每一个卷积层之后。所有VGGNet都包含五个卷积层组，和三个全连接层。每个卷积层组由一个或多个卷积层构成，在每个卷积层组之后都需进行最大值池化操作。根据网络结构设置的不同，每个卷积层组包含的卷积层的个数、卷积层的卷积核的大小存在一定差别。

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

VGG网络的基础结构

VGGNet的网络结构与AlexNet具有一定的相似性，都由5个卷积部分和3个全连接层构成，但不同的是VGGNet每个卷积部分是由卷积层组构成，而AlexNet每个卷积部分由1个卷积层构成。以VGG-19为例，其深度为19层，具体结构如表中E列所示，下面介绍每层的具体结构：

- （1）输入层：VGGNet输入224 × 224的RGB图像，因此输入大小（input size）是224 × 224 × 3。
- （2）第1个卷积层组+池化层：第1个卷积层组由2个卷积层构成，每层使用64个3 × 3大小的卷积核，步长为1，使用same卷积，使用ReLU函数激活，output为224 × 224 × 64。池化层采用最大值池化，池化窗口大小为2 × 2，步长为2，output为112 × 112 × 64。
- （3）第2个卷积层组+池化层：第2个卷积层组由2个卷积层构成，每层使用128个3 × 3 大小的卷积核，步长为1，使用same卷积，使用ReLU函数激活，output为112 × 112 × 128。池化层采用最大值池化，池化窗口大小为2 × 2，步长为2，output为56 × 56 × 128。

VGG网络的基础结构

- （4）第3个卷积层组+池化层：第3个卷积层组由4个卷积层构成，每层使用256个3× 大小的卷积核，步长为1，使用same卷积，使用ReLU函数激活，output为56 × 56 × 256。池化层采用最大值池化，池化窗口大小为2 × 2，步长为2，output为28 × 28 × 256。
- （5）第4个卷积层组+池化层：第4个卷积层组由4个卷积层构成，每层使用512个3 × 3大小的卷积核，步长为1，使用same卷积，使用ReLU 函数激活，output为28 × 28 × 512。池化层采用最大值池化，池化窗口大小为2 × 2，步长为2，output为14 × 14 × 512。
- （6）第5个卷积层组+池化层：第5个卷积层组由4个卷积层构成，每层使用512个3 × 3大小的卷积核，步长为1，使用same卷积，使用ReLU 函数激活，output为14 × 14 × 512。池化层采用最大值池化，池化窗口大小为2 × 2，步长为2，output为7 × 7 × 512。
- （7）最后3层为全连接层，前两层全连接层含有4096个神经元，层后连接ReLU函数，最后一个全连接层含有1000个神经元，层后连接softmax函数，输出1000分类的概率。

ResNet

残差神经网络（ResNet：Residual Neural Network），由微软研究院何恺明团队在2015年提出，在同年的COCO和ILSVRC竞赛中获得了检测、分割、分类、定位任务的冠军，并且原论文《Deep Residual Learning for Image Recognition》获得了2016年CVPR best paper award。

ResNet通过设置“旁路连接（shortcuts connections）”，能使信息跳过若干个卷积层，直接输入到后续的层中，提高了信息的传播效率。对于很深的神经网络，这种架构能降低神经网络的优化难度、提高模型准确率。

退化问题

存在这样一种认知：更多的神经网络层数可能意味着更准确的模型。因为对于一个层的浅层网络与一个更深的层网络，深层网络只需在前层与浅层网络保持一致，即使在后层全为恒等函数（保持输入与输出相等），也可保证深层网络表现至少与浅层网络一样好。但是，与这种认知相反的事实是，当神经网络层数增加时，模型准确率一开始会随之上升，但到达一个层数临界点后模型准确率反而下降。并且这种准确率的下降不是由过拟合导致的，因为此时模型的训练误差与测试误差都会增加。这个现象也被称为退化问题（Degradation Problem）。

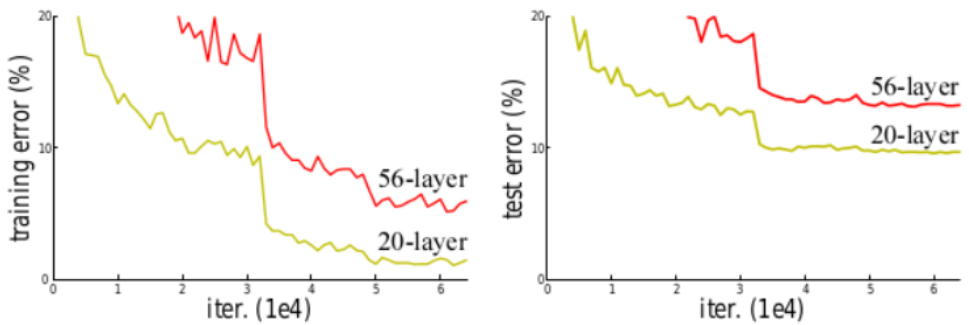


图 18: 训练CIFAR-10模型的性能退化

由图18可以看到，56层的CIFAR-10训练误差与测试误差均高于20层的CIFAR-10。这种现象可能的解释是：深层的神经网络并不能很好地学习到上文提及的恒等函数。而ResNet中的残差单元能够在一定程度上解决上述问题。

ResNet的残差单元

残差网络由很多个残差单元串联构成，一个残差单元由多个叠加的卷积层和一个跨层的旁路连接构成。

假设$H(X)$是我们想拟合的一个函数，则$H(X)$可以分解为一个恒等函数X和一个残差函数$F(X)$的和，即$H(X) = F(X) + X$。ResNet中残差单元学习的就是残差函数$F(X) = H(X) - X$，残差网络的名称也由此得来。对于一个残差单元，初始输入信息（残差单元中第一个卷积层的input）为X，最终的激活函数（ReLU）的输入为$F(X) + X$，将激活函数的输出作为整个残差单元的输出。其中，初始输入信息X经过多个卷积层作用后得到$F(X)$，同时X又能跳过卷积层直接传到后面的层中，因此最终激活函数的输入为$F(X) + X$。

ResNet的残差单元

当深度网络需要学习恒等函数时，基于ResNet中的残差单元架构，只需使用跨层的旁路连接，同时令$F(X)=0$，此时最终的激活函数的输入为$F(X)+X=X$。同时，跨层的旁路连接能够直接令初始信息X直接传到后续的层中，一定程度上减少了信息的丢失和损耗，保护了信息的完整性。残差单元的架构一定程度上解决了深层网络的梯度消失问题或梯度爆炸问题。图2-21是原论文中神经网络结构的对比图。图中左侧为VGG-19模型结构，中间为34层一般神经网络结构，右侧为34层残差神经网络结构。

ResNet的残差单元

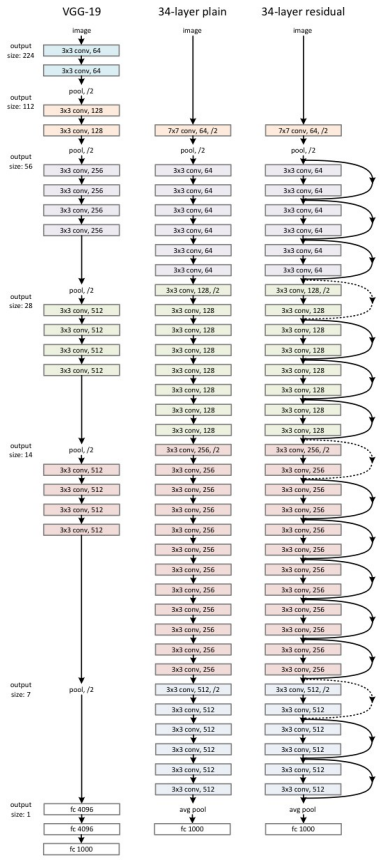


图 19: ResNet神经网络结构示例

ResNet的残差单元

图19是原论文中神经网络结构的对比图。图中左侧为VGG-19模型结构，中间为34层一般神经网络结构，右侧为34层残差神经网络结构。

图20中左侧图像对应表2中34层残差网络中的单元结构的示例，由两层 3×3的卷积网络串联，右侧图像同样为一个示例，对应50/101/152层残差网络中的单元结构，该残差单元由3层卷积网络串联，对应的卷积核分别为1×1、3×3、1×1。

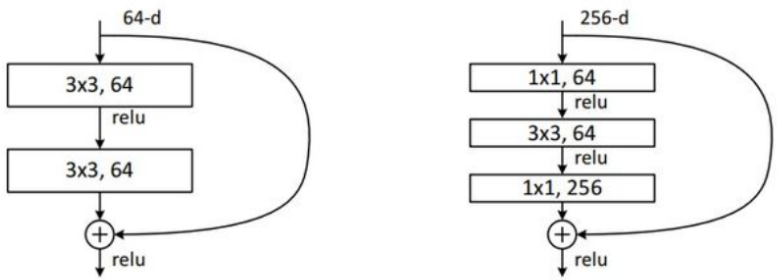


图 20: 两种残差单元结构

ResNet的结构

图21为ResNet原论文的结构示例，其中分别列出了18层、34层、50层、101层、152层ResNet的结构。表中con1行表示ResNet的输入层；conv2_x包含了池化层与残差单元；conv3_x至conv5_x均包含残差单元，具体数目如表中所示；在con5_x之后的一行表示全连接层；表中最后一行FLOPs表示对于不同层数的ResNet，单张图片一次训练所需的计算量。

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

图 21: 原论文中ResNet结构示例

ResNet的结构

下面我们以50层的ResNet为例，详细介绍内部结构。

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

图 22: 原论文中ResNet结构示例

图22中矩形方框conv1、conv2、conv3、conv4、conv5分别对应表1中的layer name，每个矩形方框下的括号对应表1中的output size。正方形方框R、B、D、S分别代表relu函数、Batch noemalization、Dropout、Softmax函数。图中直线箭头表示信息的传播方向，肘型实线箭头表示ResNet的shortcut connection，肘型虚线箭头表示当output size维度改变时ResNet的shortcut connection。字母p、s、ps分别表示padding、stride、pool size。

ResNet的结构

- （1）conv1层：conv1层使用64个卷积核，每个卷积核为7 × 7，步长为2（s=2），conv1层的输入是224 × 224大小的图片，输出的图片维度为112 × 112。
- （2）conv2层：输入信息在conv2层先进行3 × 3的最大值池化（max pool），步长设定为2。conv2层一共由3个残差单元构成，每个残差单元都由3个卷积层组成。在每个残差单元中，三个卷积层卷积核的大小分别为1 × 1、3 × 3、1 × 1，通道数分别为64、64、256。conv2层输出的图片维度为56 × 56。
- （3）conv3层：conv3层由4个残差单元构成，每个残差单元都由3个卷积层组成。在每个残差单元中，三个卷积层卷积核的大小分别为1 × 1、3 × 3、1 × 1，通道数分别为128、128、512。conv3层输出的图片维度为28 × 28。

ResNet的结构

- （4）conv4层：conv4层由6个残差单元构成，每个残差单元都由3个卷积层组成。在每个残差单元中，三个卷积层卷积核的大小分别为1 × 1、3 × 3、1 × 1，通道数分别为256、256、1024。conv4层输出的图片维度为14 × 14。
- （5）conv5层：conv5层由3个残差单元构成，每个残差单元都由3个卷积层组成。在每个残差单元中，三个卷积层卷积核的大小分别为1 × 1、3 × 3、1 × 1，通道数分别为512、512、2048。conv5层输出的图片维度为7 × 7。
- （6）fc层：最后一层为全连接层，在最后一层中，首先对输入数据进行average 池化，执行dropout操作后输出到1000分类，最后通过Softmax函数得到最终的概率输出。

DenseNet

DenseNet由康奈尔大学的 Gao Huang 和清华大学的 Zhuang Liu等人在2016年提出，原论文《Densely Connected Convolutional Networks》获得了2017年CVPR best paper award。在当时，提升神经网络性能的主要方式有两种：加宽网络结构、加深网络层数，而DenseNet却另辟蹊径，加强对特征的利用，一定程度上缓解了梯度消失问题，大幅度减少了网络的参数量。

在DenseNet之前，有很多研究涉及到了卷积神经网络的梯度消失问题，如ResNet通过设置“shortcut connection”将信息跨层传输，随机深度的ResNet通过在训练时随机地舍弃卷积层来使信息和梯度更好地传输。这些方法的核心就是创造层与层之间的“近路（short path）”，将特征图（feature map）进行跨层的传输。DenseNet一定程度上借鉴了这种思想，加强了网络中特征的传递，将特征图重复使用。

Dense Block

传统卷积前馈神经网络将第i层的输出作为第$i+1$层的输入，记第i层的输出为X_i，层间关系可以表示为

$$X_{i+1} = H_i(X_i)$$

其中$H_i(\cdot)$表示第i层的非线性变换。对于ResNet，在残差单元中增加了旁路连接，即

$$X_{i+1} = H_i(X_i) + X_i$$

如上一节所介绍，ResNet的优势是信息可以通过恒等函数到达后续的层中，但这种输出的叠加方式是简单的加和。为了进一步优化信息流的传播，DenseNet提出了更复杂的连接模式：Dense Block。

在Dense Block中，在前面的层与后续的每一层都进行连接。对于一个包含有L个卷积层的Dense Block，其中第i层($i \leq L$)接收到了前面所有层的特征图作为输入，输入个数共有i个（因此，包含L个卷积层的Dense Block的总连接个数为$1 + \cdots + L = \frac{L(L+1)}{2}$）。此时第$i$层的输出为$X_i = H_i([X_0, X_1, \ldots, X_{i-1}])$，其中$[X_0, X_1, \ldots, X_{i-1}]$代表前面所有层产生的特征图形成的拼接（concatenation）。

Dense Block

在Dense Block中，非线性变换$H_i(\cdot)$由三部分组成，分别为batch normalization（BN）、Relu函数、3×3卷积。对于每一个$H_i(\cdot)$，需要设定超参数增长率（growth rate），若增长率设定为k，表示非线性变换$H_i(\cdot)$产生k个特征图。一般情况下使用较小的k能得到不错的性能（如$k = 12$）。若初始input的通道数为k_0，则第i层的输入的特征图数目为$k_0 + (i-1)k$。

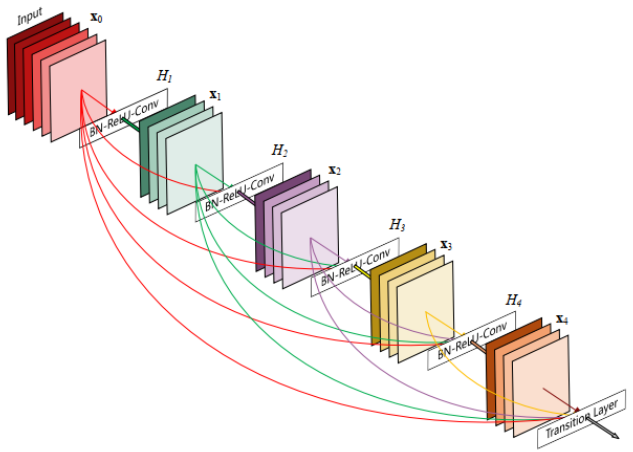


图 23: Dense Block的结构

图23展示了包含5个卷积层的Dense Block的具体结构。其中x_1, x_2, x_3, x_4对应Dense Block的输入，对应1至4层的输出，超参数增长率设定为$k = 4$。

DenseNet与ResNet对比

对ImageNet数据集使用DenseNet和ResNet的表现进行对比。图24中左侧图像的横轴代表模型学习到的参数个数，纵轴代表预测的top-1 error；右侧图像的横轴代表模型在测试集的计算次数，纵轴代表预测的top-1 error，可以发现，DenseNet在图像预测准确率、参数节约度、计算节约度上都优于ResNet。

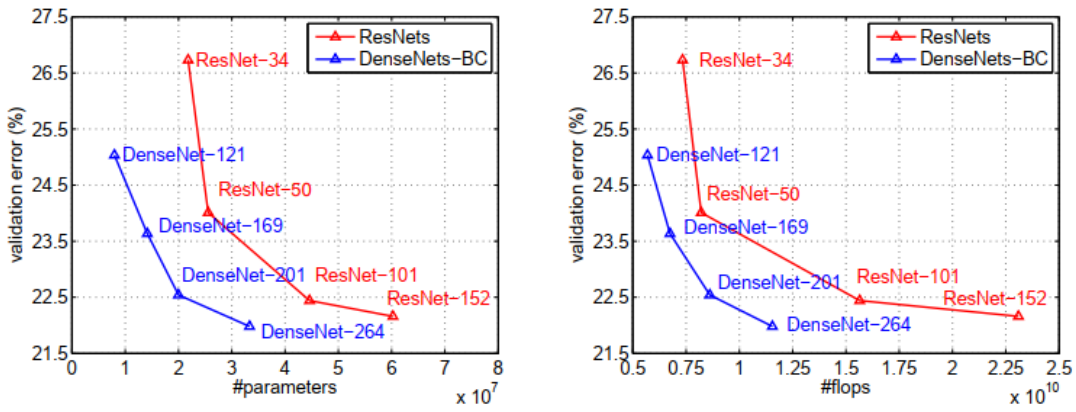


图 24: DenseNet与ResNet对比

DenseNet网络结构

对于卷积神经网络，一个核心的组成部分是下采样层（down-sampling layers）,它能改变特征图的维度。但是在DenseNet中，对不同层的特征图进行拼接操作时，需要不同层的特征图维度保持一致。所有，为了实现down sampling操作，需要将DenseNet分成多个Dense Block。在Dense Block内保持特征图维度不变，在Dense Block之间进行down sampling操作。

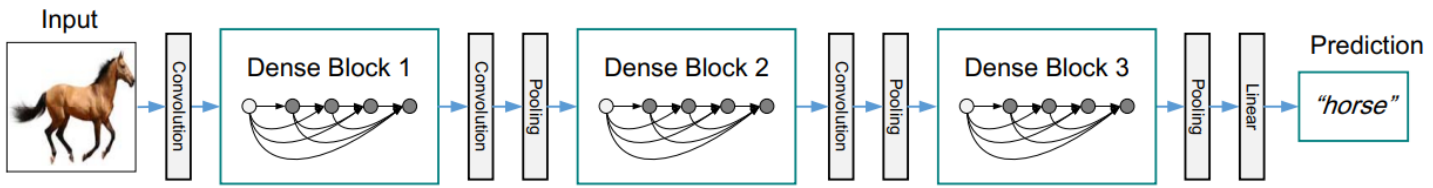


图 25: DenseNet与ResNet对比

图25给出了一个DenseNet网络结构示例，整个DenseNet由3个Dense Block组成。每个Dense Block中的卷积层由圆圈表示。前2个Dense Block连接卷积层与池化层，第3个Dense Block连接池化层与线性层，最后输出对于分类问题的预测。

DenseNet网络结构

图26是DenseNet对于ImageNet数据集设计的四种网络结构，其中增长率被设定为$k = 32$，表中每一个conv都代表batch normalization、Relu函数和卷积三项操作的复合。下面以DenseNet-169为例详细说明网络内部结构。

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

图 26: 原论文中DenseNet的网络结构

DenseNet-169

- （1）第1层为输入层，输入为224 × 224 × 3大小的图像，卷积层使用7 × 7 大小的卷积核，进行same卷积，步长为2，通道数为64，输出为112 × 112 × 64。使用batch normalization操作并用ReLU函数激活。最后进行最大值池化，步长为2，池化窗口为3 × 3，池化层的输出作为第一个Dense Block的输入。
- （2）Dense Block (1) 层：由6组1 × 1、3 × 3的卷积层构成，共12层。其中的卷积层也被称为Bottleneck层，作用是将特征图的数量减少到设定值，提高计算效率。Bottleneck层可以选择使用也可以选择不使用，对于使用了Bottleneck层的DenseNet，称为DenseNet-B。因为增长率k设定为32，所以Dense Block (1) 的输出为56 × 56 × 32。
- （3）Transition Layer (1) 层：该层由一个2 × 2Bottleneck层和一个平均池化层构成。输入为Dense Block (1) 的输出，首先输入进行batch normalization操作，然后将信息通过Relu函数在传输到Bottleneck层。在Bottleneck层中可以预先设定压缩系数$(0 < \theta \leq 1)$，对增长率k进行压缩，此时该层的输出为$56 \times 56 \times 32 \times \theta$，在池化层进行平均值池化后最终的输出为$28 \times 28 \times 32 \times \theta$。对于压缩系数$\theta < 1$的DenseNet-B，称其为DenseNet-BC。为了下文叙述的方便，这里固定$\theta = 1$。

DenseNet-169

- （4）后续的Dense Block (2) 层、Dense Block (3) 、Dense Block (4)层的内部结构与Dense Block (1) 层类似，内部分别包含12组、32组、32组和的卷积层。Transition Layer (2) 与Transition Layer (3)内部结构也与Transition Layer (1)相似。因为增长率k固定为32，因此Dense Block (2) 层的输出为$28 \times 28 \times 32$，Transition Layer (2)层的输出为$14 \times 14 \times 32$，Dense Block (3)层的输出为$7 \times 7 \times 32$，Transition Layer (3)层的输出为$7 \times 7 \times 32$，Dense Block (4) 的输出为$7 \times 7 \times 32$
- （5）Classification层：该层输入为Dense Block (4) 的输出，输入信息经过batch normalization操作、通过Relu函数之后进入7×7，步长为1的池化层，执行全局平均池化，池化层的输出为$1 \times 1 \times 32$ 。最后，池化层的输出进入1000分类的全连接层，由softmax函数输出对应的分类概率。

MobileNet

常规卷积神经网络庞大且复杂，训练需要很大的计算量，难以应用在某些现实场景当中，如嵌入式终端或移动终端。在这些终端上，神经网络的主要应用场景广泛地集中但不限于物体识别和分类、人脸识别、地标检测等（如图27所示）。因此在这类终端上运行的神经网络需要在保证模型性能的同时降低模型大小、并且提高运行速度。基于此，谷歌团队在2017年提出了MoblieNet V1 ，为移动式和嵌入设备提供了高效的轻量级神经网络架构。随后，谷歌团队在MobileNet V1的基础上加以改进，在2018年、2019年先后提出了MobileNet V2、MobileNet V3。本节仅介绍MobileNet V1，为了方便起见，简称为MobileNet。

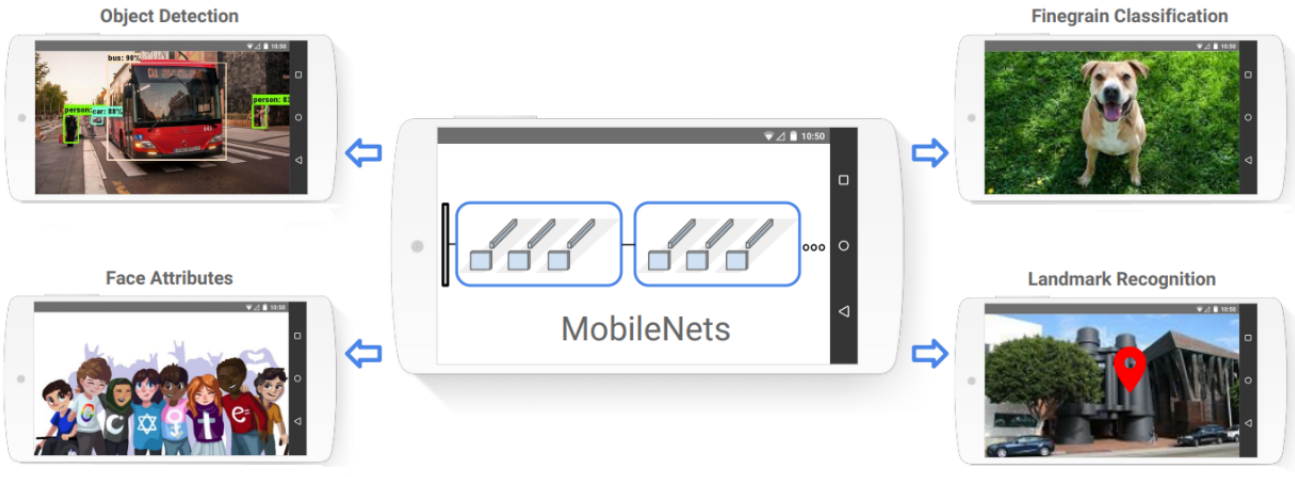


图 27: MobileNet应用场景

深度可分离卷积（Deepwise Separable Convolution）

MobileNet的独特架构是深度可分离卷积。深度可分离卷积把标准卷积操作分解为两部分：深度卷积（depthwise convolution）和逐点卷积（pointwise convolution），并且能起到与标准卷积差不多的整体效果。标准卷积操作能起到的效果有两点：第一，过滤特征（基于卷积核）、第二，组合特征，产生新的特征表达。因此，深度可分离卷积中的深度卷积用来过滤特征、逐点卷积用来生成深度卷积输出的线性组合。相比于标准卷积操作，采用深度可分离卷积能够大幅缩减模型参数数量，降低计算量。

对于标准卷积而言，其卷积核作用于所有输入的通道上，与此不同的是，深度卷积对于每一个输入通道都采用不同的卷积核。图2-28左侧图像是一个三通道深度卷积的示例，可以发现有三个不同的卷积核分别作用于三个输入通道上。如图28右侧图像所示，逐点卷积是标准的卷积，用于整合深度卷积的输出，起到转换通道的作用。

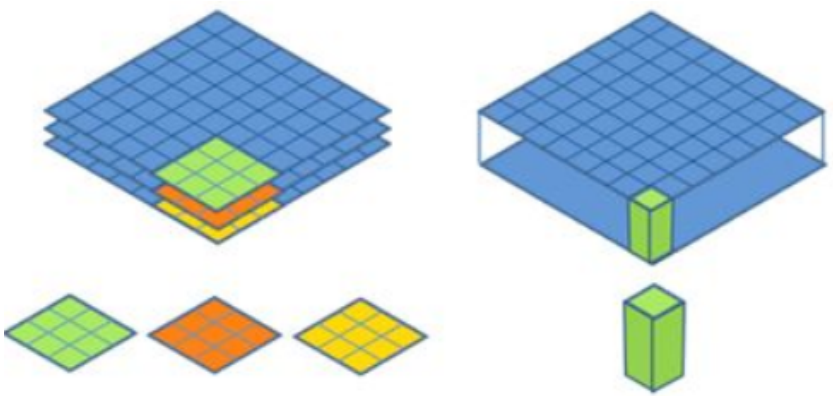


图 28: 深度卷积（左）和逐点卷积（右）示意图

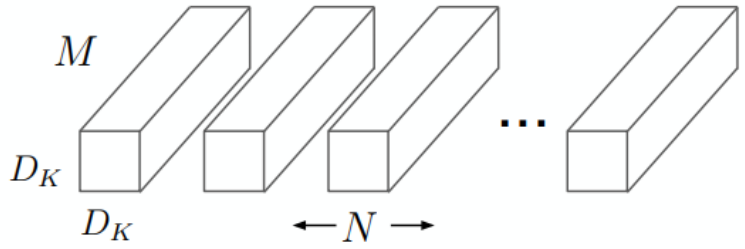
深度可分离卷积（Deepwise Separable Convolution）

接下来详细介绍深度可分离卷积的内部结构：对于标准卷积（如图29（a）所示），假定输入的特征图大小为$D_F \times D_F \times \mathrm{M}$，输出的特征图大小为$D_F \times D_F \times N$，其中$D_F$为输入、输出特征图的宽度和高度，$M$、$N$分别代表输入的通道数与输出的通道数（$N$也表示卷积核的个数）。若卷积核的大小为$D_K \times D_K \times M$（标准卷积的卷积核作用在$M$个输入通道上），对于施加padding操作、步长为1的标准卷积，输出特征图计算公式如下：

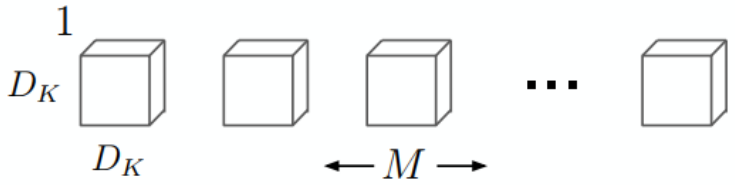
$$G_{k,l,n} = \sum_{i,j,m} K_{i,j,m,n} \cdot F_{k+i-1,l+j-1,m}$$

其中F代表输入的特征图、G代表输出的特征图、K代表卷积核，标准卷积所需的计算量为$D_K \times D_K \times M \times N \times D_F \times D_F$。

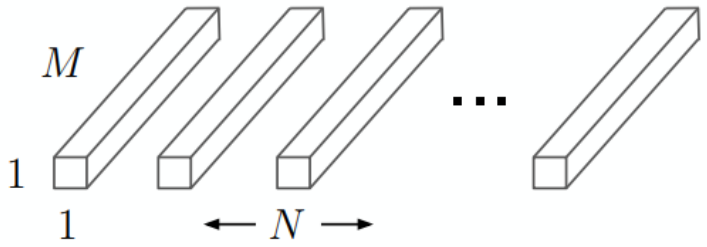
深度可分离卷积（Deepwise Separable Convolution）



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

图 29: 标准卷积（a）、深度卷积（b）、逐点卷积（c）示意图

深度可分离卷积（Deepwise Separable Convolution）

对于深度卷积（如图29（b）所示），假定输入的特征图大小为$D_F \times D_F \times \mathrm{M}$。此时有$M$个卷积核，每个卷积核的大小均为$D_K \times D_K \times 1$。对输入的特征图的$M$个通道分别做卷积，此时每个卷积核作用的通道数为1。对于第m个通道（$m \in \{1, 2, \ldots, M\}$），深度卷积输出的特征图计算公式如下：

$$\hat{G}_{k,l,m} = \sum_{i,j} \hat{K}_{i,j,m} \cdot F_{k+i-1,l+j-1,m}$$

此处F表示输入的特征图，$\hat{G}$表示输出的特征图，$\hat{K}$表示卷积核。深度卷积所需的计算量为$D_K \times D_K \times M \times D_F \times D_F$。与标准卷积相比，深度卷积更有效、所需的计算量更小，但是深度卷积只起到了过滤输入特征的作用，并没有将过滤后的特征结合组成新特征。因此，我们需要逐点卷积操作来结合深度卷积输出的特征，产生新的特征表达。

深度可分离卷积（Deepwise Separable Convolution）

对逐点卷积（如图29（c）所示）而言，它的输入是深度卷积所输出的特征图。逐点卷积共有N个卷积核，大小均为$1 \times 1 \times M$，每一个卷积核作用于M个通道。逐点卷积的计算量为$M \times N \times D_F \times D_F$。

因此，深度可分离卷积的总计算量为深度卷积的计算量加逐点卷积的计算量，等于$D_K \times D_K \times M \times D_F \times D_F + M \times N \times D_F \times D_F$。深度可分离卷积的总计算量与标准卷积的计算量的比值为$\frac{1}{N} + \frac{1}{D_K^2}$。当$D_K = 3$，即使用$3 \times 3$大小的卷积核时，深度可分离卷积的计算量大约能减少为标准卷积计算量的$\frac{1}{9}$。

深度可分离卷积（Deepwise Separable Convolution）

深度可分离卷积在应用中会包含batch normalization操作，并采用ReLU函数进行激活。其基本结构如图30所示：

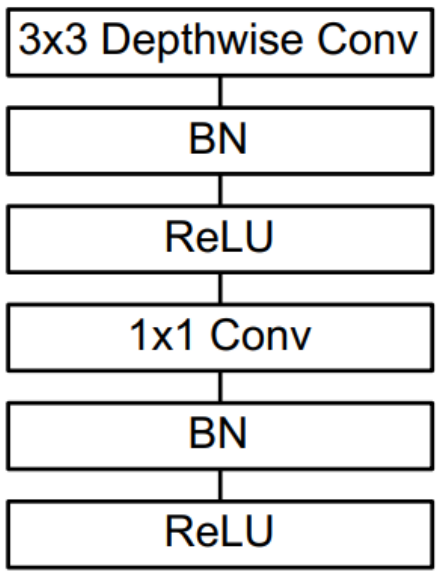


图 30: 深度可分离卷积结构图

可以看到，深度可分离卷积包含深度卷积层与逐点卷积层，每一层之后紧接着进行batch normalization操作，再通过ReLU函数。

MobileNet结构

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

图 31: MobileNet结构表

由图31可以看到，整个MobileNet网络第一层为标准卷积层，使用32个卷积核，卷积核大小为$3 \times 3 \times 3$，后续的层由多个深度可分离卷积模块组成，最后的输出经过平均池化操作以后输入到全连接层，再通过Softmax函数得到对应的分类概率。

超参数：宽度因子与分辨率因子

我们还可以设置宽度因子α（Width Multiplier）、分辨率因子（Resolution Multiplier）这两个超参数来调整MobileNet的模型大小。

宽度因子$\alpha(0 < \alpha \leq 1)$可以将输入和输出通道数由$M, N$对应调整为$\alpha M, \alpha N$。此时，模型的参数量将会下降，同时深度可分离卷积的计算量将变为$D_K \times D_K \times \alpha M \times D_F \times D_F + \alpha M \times \alpha N \times D_F \times D_F$，因为主要计算量集中在后面一项，所以整个深度可分离卷积的计算量将变为原来的α倍。

分辨率因子$\rho(0 < \rho \leq 1)$能够控制输入图像的分辨率，能够按比例ρ降低特征图的大小。假设原来输入的特征图大小为$D_F \times D_F$，设置分辨率因子之后特征图大小变为$\rho D_F \times \rho D_F$。需要注意的是，仅设置分辨率因子并不能影响参数的数目，只能减少计算量。当我们设置宽度因子α与分辨率因子ρ以后，深度可分离卷积的计算量将变为$D_K \times D_K \times \alpha M \times \rho D_F \times \rho D_F + \alpha M \times \alpha N \times \rho D_F \times \rho D_F$。

单隐藏层BP神经网络

图32、图33分别展示了选择不同的α和ρ时模型的预测精度、计算量、参数个数。

Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
0.75 MobileNet-224	68.4%	325	2.6
0.5 MobileNet-224	63.7%	149	1.3
0.25 MobileNet-224	50.6%	41	0.5

图 32: 宽度因子选择与模型预测精度、计算量、参数量的关系

Resolution	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
1.0 MobileNet-192	69.1%	418	4.2
1.0 MobileNet-160	67.2%	290	4.2
1.0 MobileNet-128	64.4%	186	4.2

图 33: 分辨率因子选择与模型预测精度、计算量、参数量的关系

MobileNet与其他网络的比较

如图34所示，在ImageNet数据集上，MobileNet与GoogleNet相比模型预测精度更高，但是MobileNet所需的计算量、参数个数均小于GoogleNet。与VGG16相比，MobileNet虽然预测精度低了0.9%，但是MobileNet模型计算量、参数个数均远远小于VGG16。

如图35所示，当设置宽度因子$\alpha = 0.5$与分辨率因子$\rho = \frac{160}{224}$之后，可以发现在ImageNet数据集上MobileNet的预测精度高于Squeezenet和AlexNet，并且计算量也小于这两个网络，在参数量上，MobileNet与Squeezenet相接近，但是远远小于AlexNet。因此，在ImageNet数据集上，MobileNet的表现更加优异。

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

图 34: MobileNet与其他模型的比较

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.50 MobileNet-160	60.2%	76	1.32
Squeezenet	57.5%	1700	1.25
AlexNet	57.2%	720	60

图 35: 小型MobileNet与其他模型的比较

ShuffleNet

ShuffleNet是由旷视(Face++)研究院提出的一种高效卷积模型神经网络结构，它在保持较高的识别精度的同时，可大幅降低模型计算复杂度。ShuffleNet是针对移动端低功耗设备设计的。ShuffleNet网络结构沿袭了稀疏连接的设计理念，在分析了目前的网络结构后，ShuffleNet作者发现将卷积核拆分成逐通道卷积与逐点卷积的方式虽然能使计算复杂度有所下降，但是拆分所产生的逐点卷积计算量却占据了大部分计算量，例如MobileNet模型中逐点卷积占据了95%的计算量和75%的参数，成为新的瓶颈。
为了更进一步地优化计算量以提升模型的速度，ShuffleNet使用分组逐点卷积(group pointwise convolution)来代替逐点卷积，将输入特征图分为若干组，卷积运算的输入限制在每个组内，各组独立进行卷积操作，如图36 (a)所示。分组逐点卷积的方式虽然减少了计算量，但是多层堆叠后，特征图中的信息被分割在各个组内，各组间缺少信息交流会影响到模型的特征提取能力。因此需要引入组间信息交换的机制，如图36（b）所示，在下一卷积中，每个卷积核需要来自不同组的特征作为输入，这种信息交换的种实现机制就是通道重排(channel shuffle)，如图36（c）所示，通道重排的梯度传播类似池化层，梯度会分配到原来的位置，所以可以把通道重排层嵌入到网络中直接训练。

ShuffleNet

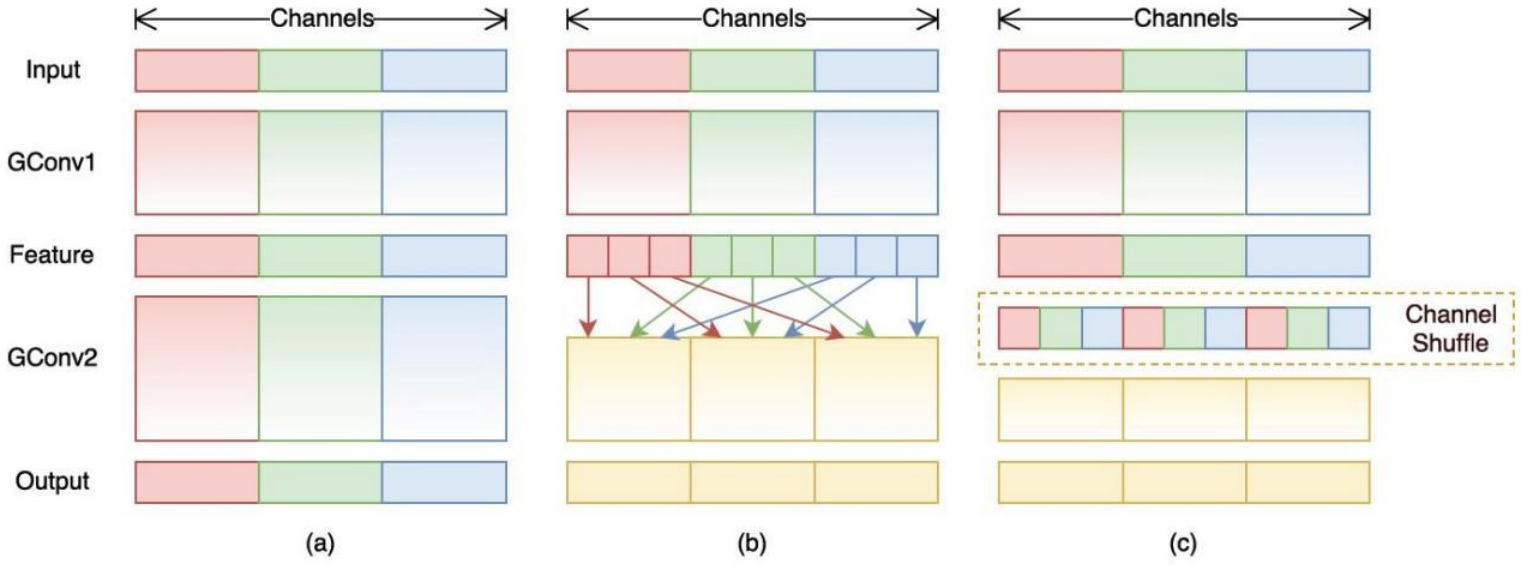


图 36: 分组逐点卷积与通道重排

ShuffleNet

Suflelet的基本模块在设计时继承了深度残差网络(ResNet)的设计思想，基于分组逐点卷积和通道重排操作设计了两种模块，图37(a)和图37(b)表明了将一个残差模块改造成Shuflele模块的过程。图37(a)中为了提高计算效率，使用逐通道卷积(DWConv)替换普通卷积，以便降低提取空间特征的卷积操作的复杂度。图37(b)在图37(a)的基础上将前后两个卷积使用分组逐点卷积代替(GConv)，并嵌入通道重排层以降低卷积运算的跨通道计算量，得到ShuffleNet模块。为了更高效地进行降采样，以同样的思想设计了ShuffleNet降采样模块，见图37(c)。

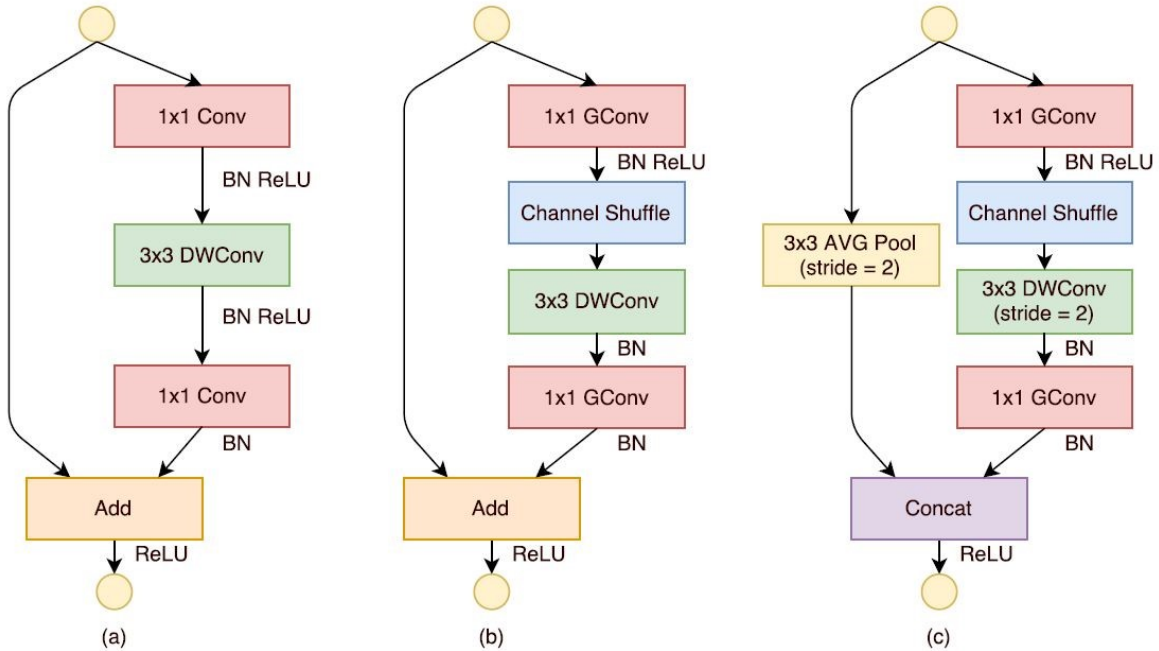


图 37: ShuffleNet的结构单元

ShuffleNet

基于ShuffleNet模块可构建出ShuffeNet网络，网络的配置如图38所示。ShuffleNet主要由16个ShuffleNet模块组成。这些ShuffleNet模块分为三个阶段，每个阶段包含一个ShuffleNet下采样模块和若干ShuffleNet模块，与VGGNet类似，每个阶段特征图的空间尺寸缩减为原来的一半，通道数变为原来的两倍。在ShuffleNet的分组逐点卷积中，不同的分组数会对模块的准确率和计算量有影响。分组数就是ShuffleNet的超参数。分组越多，模型的计算量就越小。当总计算量一定时，较大的分组数可以允许较多的通道数，有利于网络编码更多的信息，提升模型的识别能力。在实际使用中，可根据不同的计算需要，适当调整分组，放缩各层通道数以定制计算复杂度可承受的模型。

Layer	Output size	KSize	Stride	Repeat	Output channels (g groups)				
					$g=1$	$g=2$	$g=3$	$g=4$	$g=8$
Image	224×224				3	3	3	3	3
Conv1	112×112	3×3	2	1	24	24	24	24	24
MaxPool	56×56	3×3	2						
Stage2	28×28		2	1	144	200	240	272	384
	28×28		1	3	144	200	240	272	384
Stage3	14×14		2	1	288	400	480	544	768
	14×14		1	7	288	400	480	544	768
Stage4	7×7		2	1	576	800	960	1088	1536
	7×7		1	3	576	800	960	1088	1536
GlobalPool	1×1	7×7							
FC					1000	1000	1000	1000	1000
Complexity					143M	140M	137M	133M	137M

图 38: ShuffleNet的网络配置

习题

1. same卷积和valid卷积有什么区别？
2. 卷积和池化的联系和区别是什么？
3. 请简要阐述卷积神经网络的发展历程，对于不同时间段的代表性网络，说明它们各自的优缺点。
4. 请简要阐述ResNet的残差单元结构，并说明残差单元发挥的作用。
5. 请简要阐述DenseNet中Dense Block的结构，它与ResNet的残差单元的区别与联系是什么？
6. 考虑keras的数据库模块中的MNIST数据集
   （1）仿照LeNet-5网络结构搭建简单的卷积神经网络模型
   （2）通过训练模型，实现手写数字识别
   （3）进行模拟预测