学号 20171004186

密级

学习笔记

典型的分类网络

院（系）名　称：　计算机学院

专　业　名　称：　空间信息与数字技术

学　生　姓　名：　吴朝明

指　导　教　师：　xxx　教授

二〇二一年三月

郑 重 声 明

本人呈交的学位论文，是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确的方式标明。本学位论文的知识产权归属于培养单位。

本人签名：_____ 日期：_____

目　　录

1　典型神经网络 …… 1

1.1　AlexNet …… 1

1.2　VGGNet …… 2

1.3　GoogLeNet …… 3

1.4　inception v2 v3 …… 9

1.5　inception v4 …… 12

1.6　ResNet …… 14

1.7　DenseNet …… 23

1.8　SENet …… 27

参考文献 …… 30

1 典型神经网络

1.1 AlexNet

Alex Krizhevsky 等人 [1] 提出的，并由此网络模型获得了 2012 年的 ImageNet ILSVRC 的冠军。AlexNet 采用 5 层卷积加 3 层全连接和一个 softmax 分类器的结构，使用两个 GPU 进行训练，最后在 ImageNet 数据集上取得了优秀的成绩。

ImageNet 数据集有 1500 万张带标签的高分辨率图像，共有 22000 个类别。ILSVRC 比赛使用 ImageNet 的子集，有 1000 个类别。在 ImageNet 数据集中，包含有各种分辨率的图像，AlexNet 模型却需要一个恒定的输入维数，下采样这些图像得到固定的分辨率 256×256。

AlexNet 整体结构比较简单，主要特点有：采用最大池化代替以往的平均池化，这是因为平均池化容易出现特征图模糊的现象；卷积层后采用 Relu 激活函数代替 Sigmoid 激活函数，用于缓解神经网络中的梯度消息问题；并且引入 Dropout 随机丢弃一部分神经元，防止训练过程中出现过拟合现象；反向传播过程中使用随机梯度下降算法用于优化网络训练过程，提高收敛速度。其网络结构如 1.1所示：

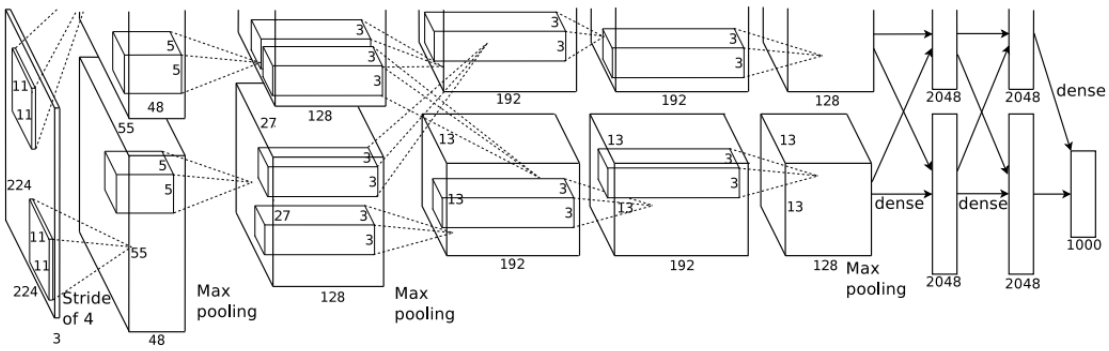


图 1.1 AlexNet 网络结构

从图中可以看出，ALexNet 包含有 8 层网络结构，其中前 5 层为卷积层，后 3 层为全连接层。作者将其在两个 GPU 上运行，卷积层中只有在第 3 层两个 GPU 中才有数据的交流，全连接层中两个 GPU 之间都有连接。相关的参数设置总结如下：

- 卷积层 1：输入 $224 \times 224 \times 3$，卷积核 $11 \times 11 \times 3$，卷积核个数 96，步长 4，padding 为 0，输出 $96 \times 55 \times 55$；
- 池化层：3×3，步长 2，最大池化，输出 $96 \times 27 \times 27$；

- 卷积层 2：输入 96×27×27，卷积核 5×5，卷积核个数 256，步长 1，padding 为 2，输出 256×13×13；
- 池化层：3×3，步长 2，最大池化，输出 256×13×13；
- 卷积层 3：输入 256×13×13，卷积核 3×3，卷积核个数 384，步长 1，padding 为 1，输出 384×13×13；
- 卷积层 4：输入 384×13×13，卷积核 3×3，卷积核个数 384，步长 1，padding 为 1，输出 384×13×13；
- 卷积层 5：输入 384×13×13，卷积核 3×3，卷积核个数 256，步长 1，padding 为 1，输出 256×13×13；
- 池化层：3×3，步长 2，最大池化，输出 256×6×6；
- 全连接层 1：共 4096 个神经元；
- 全连接层 2：共 4096 个神经元；
- 全连接层 3：共 1000 个神经元；

1.2 VGGNet

Simonyan 等人 [2] 于 2014 年提出的 VGG 网络结构在当年的 ILSVRC 比赛中定位和分类分别获得了第一名和第二名的优秀成绩。VGGNet 与 AlexNet 相似，都是由卷积层和全连接层构成，但 VGGNet 的性能以及使用程度都在 AlexNet 上有了较大的改进。在文章中，作者主要探索了卷积网络架构设计中网络深度对网络性能的影响，通过在网络层中使用非常小的 3×3 的卷积滤波器，保证了在添加卷积层时能够稳定地增加网络深度。

在原文中，作者提出了多种深度的网络结构，如 1.2所示：

VGGNet 由 5 层卷积层，3 层全连接层和 1 层 softmax 输出层组成，5 个卷积层的后面都接有最大池化层，激活函数采用 ReLU 函数。需要注意的是，在 VGGNet 网络结构中，并不是每一层卷积层后面都接有池化层。相较于 AlexNet 而言，VggNet 采用了更小的卷积核，以及更多的卷积子层。VGGNet 中卷积核大小全为 3×3，作者认为两个 3×3 卷积核堆积有 5×5 的有效感受野，三个这样的层则具有 7×7 的有效感受野，这样的话结合了三个非线性修正层，使得决策函数更具有判别性，同时能够减少参数量。而在测试阶段，将 3 层全连接层改为卷积层，使得对测试时输入的图像尺寸没有要求。该文中通过多组实验对比说明了 VGGNet 在大部分数

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					

图 1.2　VGG 网络结构

据上的性能较好，同时也证明了 AlexNet 中 LRN 操作对网络性能没有影响。

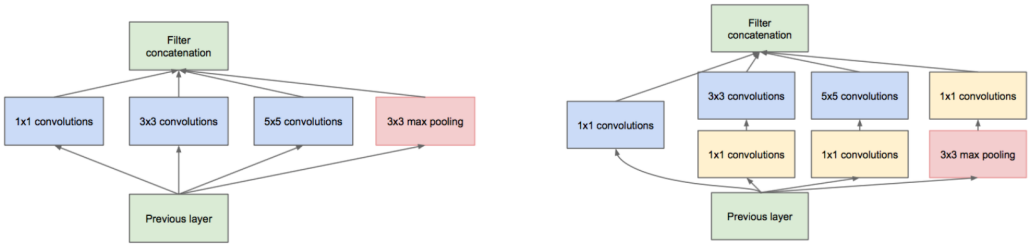
1.3　GoogLeNet

Szegedy 等人 [3] 提出的 GoogLeNet(inception v1) 在 2014 年的 ILSVRC 比赛中获得了第一名的成绩。相较于 VGGNet 继承了 AlexNet 网络结构并进行加深和使用小卷积核而言，GoogLeNet 则是尝试了新的网络结构，尽管深度比 AlexNet 和 VGGNet 都更多，有 22 层，但参数只有 500 万个，比两者都要小得多，特点在于采用了 Inception 模块。

在以往，提升神经网络的性能最常见的方法就是提高网络深度 (网络层次的

数量) 和宽度 (神经元的数量)，但这样也会出现很多问题。参数过多的话，训练集如果数量不够多就容易出现过拟合的情况；网络越深，宽度越大，计算量也会越大；网络越深，容易出现梯度弥散问题，难以优化模型。为此，Google 团队提出了 Inception 网络结构。

Inception 架构的主要想法在于如何近似卷积神经网络中的最优局部稀疏结构并用其他的密集组件进行覆盖。为此，首先提出了最初的基本结构 1.3a：



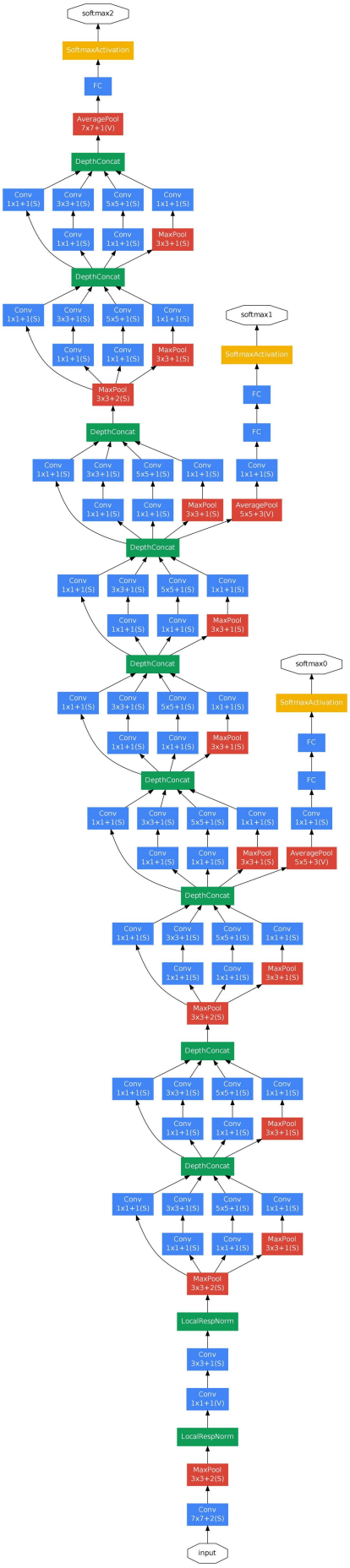
(a) Inception module naive version　　(b) Inception module with dimensionality reduction

图 1.3　GoogLeNet Inception 架构

最初的 Inception 架构含有 4 个分支，前 3 个分支进行卷积之后得到的特征图大小是一致的，于是在进行卷积之后就可以堆叠起来作为下一层的输入。其中的卷积核大小分别为 1×1，3×3，5×5，这样的设置更多的是为了方便而并非是必须，第 4 个分支是池化操作。但是这种原始结构的缺点在于 5×5 的卷积核计算量太大，为此，在 3×3，5×5 的卷积层前，3×3 的最大池化层后加上 1×1 的卷积降维，即 Inception v1 的网络架构，如图 1.3b，1×1 的卷积核的主要目的减小维度和修正线性激活。

GoogLeNet 的具体网络结构如图 1.4，网络的具体描述如下：

1. 输入图像 $224\times224\times3$，预处理是图像每个像素减去均值 (零均值化)；
2. 卷积层 1，7×7，stride：2，padding：3，64 通道，输出 $112\times112\times64$，卷积后 ReLU 操作；
3. 池化层，最大池化，3×3，stride：2，输出 $56\times56\times64$，后接 ReLU 操作；
4. 卷积层 2，3×3，stride：1，padding：1，192 通道，输出 $56\times56\times192$，卷积后 ReLU 操作；
5. 池化层，最大池化，3×3，stride：2，输出 $28\times28\times192$，后接 ReLU 操作；
6. • inception(3a)



5

图 1.4　GoogLeNet 网络结构

- 1×1 卷积核，64 通道，输出 $28\times28\times64$，后接 ReLU；
- 1×1 卷积核，96 通道，得到 $28\times28\times96$，后接 ReLU，3×3 卷积核，padding 为 1，通道 128，输出到 $28\times28\times128$；
- 1×1 卷积核，16 通道，得到 $28\times28\times16$，后接 ReLU，5×5 卷积核，padding 为 2，通道 32，输出到 $28\times28\times32$；
- 最大池化，3×3 卷积核，padding 为 1，通道 192，得到 $28\times28\times192$，1×1 卷积核，32 通道，得到 $28\times28\times32$；
- 四个部分连接，$64+128+32+32=256$，得到 $28\times28\times256$；

7. - inception(3b)
   - 1×1 卷积核，128 通道，输出 $28\times28\times128$，后接 ReLU；
   - 1×1 卷积核，128 通道，得到 $28\times28\times128$，后接 ReLU，3×3 卷积核，padding 为 1，通道 192，输出到 $28\times28\times192$；
   - 1×1 卷积核，32 通道，得到 $28\times28\times32$，后接 ReLU，5×5 卷积核，padding 为 2，通道 96，输出到 $28\times28\times96$；
   - 最大池化，3×3 卷积核，padding 为 1，通道 256，得到 $28\times28\times256$，1×1 卷积核，64 通道，得到 $28\times28\times64$；
   - 四个部分连接，$128+192+96+64=480$，得到 $28\times28\times480$；
8. 池化层，最大池化，3×3，stride：2，padding：1，输出 $14\times14\times480$，后接 ReLU 操作；
9. - inception(4a)
   - 1×1 卷积核，192 通道，输出 $14\times14\times192$，后接 ReLU；
   - 1×1 卷积核，96 通道，得到 $14\times14\times96$，后接 ReLU，3×3 卷积核，padding 为 1，通道 208，输出到 $14\times14\times208$；
   - 1×1 卷积核，16 通道，得到 $14\times14\times16$，后接 ReLU，5×5 卷积核，padding 为 2，通道 48，输出到 $14\times14\times48$；
   - 最大池化，3×3 卷积核，padding 为 1，通道 480，得到 $14\times14\times480$，1×1 卷积核，64 通道，得到 $14\times14\times64$；
   - 四个部分连接，$192+208+48+64=512$，得到 $14\times14\times512$；
10. - inception(4b)
    - 1×1 卷积核，160 通道，输出 $14\times14\times160$，后接 ReLU；

- 1×1 卷积核，112 通道，得到 $14\times14\times112$，后接 ReLU，3×3 卷积核，padding 为 1，通道 224，输出到 $14\times14\times224$；
- 1×1 卷积核，24 通道，得到 $14\times14\times24$，后接 ReLU，5×5 卷积核，padding 为 2，通道 64，输出到 $14\times14\times64$；
- 最大池化，3×3 卷积核，padding 为 1，通道 512，得到 $14\times14\times512$，1×1 卷积核，64 通道，得到 $14\times14\times64$；
- 四个部分连接，$160+224+64+64=512$，得到 $14\times14\times512$；

11. • inception(4c)
    - 1×1 卷积核，128 通道，输出 $14\times14\times128$，后接 ReLU；
    - 1×1 卷积核，128 通道，得到 $14\times14\times128$，后接 ReLU，3×3 卷积核，padding 为 1，通道 256，输出到 $14\times14\times256$；
    - 1×1 卷积核，24 通道，得到 $14\times14\times24$，后接 ReLU，5×5 卷积核，padding 为 2，通道 64，输出到 $14\times14\times64$；
    - 最大池化，3×3 卷积核，padding 为 1，通道 512，得到 $14\times14\times512$，1×1 卷积核，64 通道，得到 $14\times14\times64$；
    - 四个部分连接，$128+256+64+64=512$，得到 $14\times14\times512$；
12. • inception(4d)
    - 1×1 卷积核，112 通道，输出 $14\times14\times112$，后接 ReLU；
    - 1×1 卷积核，144 通道，得到 $14\times14\times144$，后接 ReLU，3×3 卷积核，padding 为 1，通道 288，输出到 $14\times14\times288$；
    - 1×1 卷积核，32 通道，得到 $14\times14\times32$，后接 ReLU，5×5 卷积核，padding 为 2，通道 64，输出到 $14\times14\times64$；
    - 最大池化，3×3 卷积核，padding 为 1，通道 512，得到 $14\times14\times512$，1×1 卷积核，64 通道，得到 $14\times14\times64$；
    - 四个部分连接，$112+288+64+64=528$，得到 $14\times14\times528$；
13. • inception(4e)
    - 1×1 卷积核，256 通道，输出 $14\times14\times256$，后接 ReLU；
    - 1×1 卷积核，160 通道，得到 $14\times14\times160$，后接 ReLU，3×3 卷积核，padding 为 1，通道 320，输出到 $14\times14\times320$；
    - 1×1 卷积核，32 通道，得到 $14\times14\times32$，后接 ReLU，5×5 卷积核，

padding 为 2，通道 128，输出到 14×14×128；

- 最大池化，3×3 卷积核，padding 为 1，通道 528，得到 14×14×528，1×1 卷积核，128 通道，得到 14×14×128；
- 四个部分连接，256＋320＋128＋128＝832，得到 14×14×832；

14. 池化层，最大池化，3×3，stride：2，padding：1，输出 7×7×832，后接 ReLU 操作；
15.
    - inception(5a)
    - 1×1 卷积核，256 通道，输出 7×7×256，后接 ReLU；
    - 1×1 卷积核，160 通道，得到 7×7×160，后接 ReLU，3×3 卷积核，padding 为 1，通道 320，输出到 7×7×320；
    - 1×1 卷积核，32 通道，得到 7×7×32，后接 ReLU，5×5 卷积核，padding 为 2，通道 128，输出到 7×7×128；
    - 最大池化，3×3 卷积核，padding 为 1，通道 832，得到 7×7×832，1×1 卷积核，128 通道，得到 7×7×128；
    - 四个部分连接，256＋320＋128＋128＝832，得到 7×7×832；
16.
    - inception(5b)
    - 1×1 卷积核，384 通道，输出 7×7×384，后接 ReLU；
    - 1×1 卷积核，192 通道，得到 7×7×192，后接 ReLU，3×3 卷积核，padding 为 1，通道 384，输出到 7×7×384；
    - 1×1 卷积核，48 通道，得到 7×7×48，后接 ReLU，5×5 卷积核，padding 为 2，通道 128，输出到 7×7×128；
    - 最大池化，3×3 卷积核，padding 为 1，通道 832，得到 7×7×832，1×1 卷积核，128 通道，得到 7×7×128；
    - 四个部分连接，384＋384＋128＋128＝1024，得到 7×7×1024；
17. 池化层，平均池化，7×7，stride：1，输出 1×1×1024，后接 ReLU 操作；
18. dropout(40%)，输出 1×1×1024；
19. linear，输入 1×1024，输出 1×1000；
20. softmax，输出 1×1000；

网络的相关参数设置如下 1.5所示，其中 #3×3 *reduce* 和 #5×5 *reduce* 表示在卷积之前，进行降维使用的 1×1 的滤波器数量。

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

图 1.5　GoogLeNet 网络参数设置

1.4　inception v2 v3

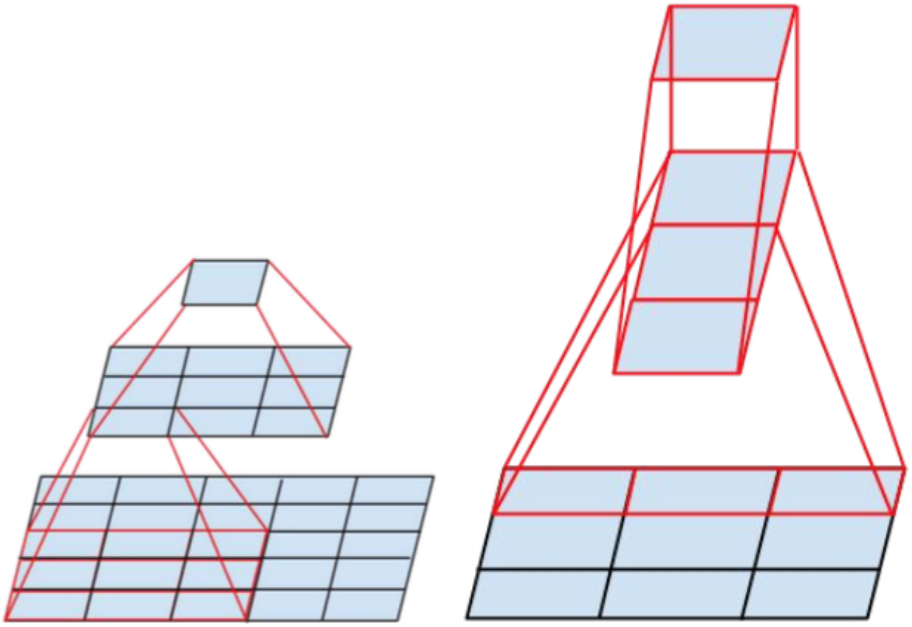
在提出了 Inception v1 结构之后，Google 团队在 [4] 中提到训练深度神经网络的复杂性在于，由于前面的层的参数在训练过程中会发生变化，这就导致了后面的层是输入的分布在训练过程中也会发生改变；并且由于网络层数的加深，网络参数细微的变化会被逐渐放大；较低的学习率和仔细地参数初始化也导致了模型的训练速度变慢，同时使得具有饱和非线性的模型训练起来十分困难，这种现象被称为内部协变量转移。为此，该文章中又提出了批标准化 BN 来解决该问题。将标准化做成模型架构的一部分，并为每个批量小数据做标准化，可以在不用太注意学习率初始化的同时提高学习率，同时作为一个正则化项，使得网络可以去除 Dropout。应用于其他的图像分类模型之后，在达到相同精度的情况下，能够缩小 14 倍的训练步骤，并以显著的差距击败了原始模型。

2016 年，Szegedy 等人 [5] 针对计算效率限制了卷积网络具体场景上的应用的问题，探索了增大网络的方法，通过适当的卷积和正则化来尽可能提高计算效率。

Inception 架构的复杂性使得难以对网络进行更改，单纯地方法架构会导致计算效率的降低，为此需要使用一些具有卷积网络、具有各种架构选择和基于大规

模实验的设计原则。在网络的前面尽量避免表征瓶颈；更高维度的表示在网络中更容易局部处理；空间聚合可以在较低维度嵌入上完成，而不会在表示能力上造成许多损失；最后是需要平衡每个阶段的滤波器数量和网络深度以使网络达到较好性能。

在文章中，作者提出了一些新的想法：

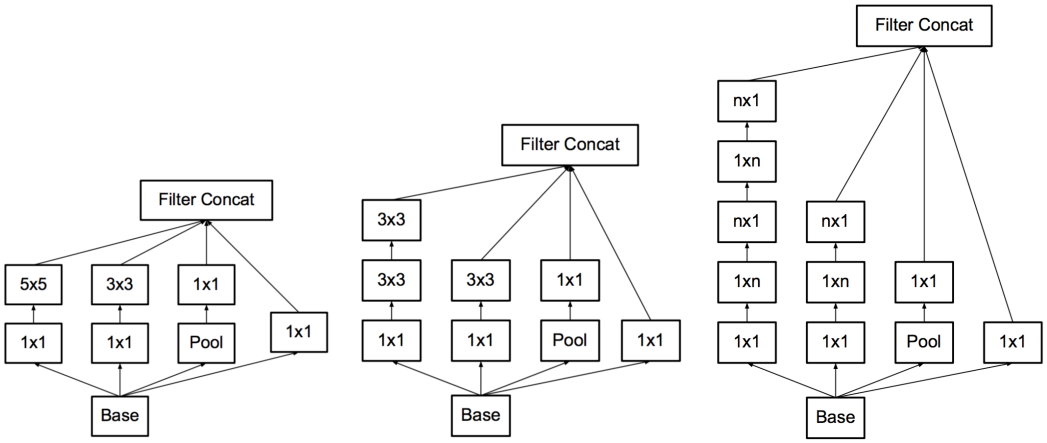


(a) Inception v2 3x3 convolutions　(b) Inception v2 3x1 convolutions

图 1.6　Inception v2 卷积核替代

卷积分解，如图 1.6a一个 5×5 的卷积可由两个 3×3 的卷积核进行替代，在保证了有 5×5 的卷积核的感受野的同时，在一定程度上减少了参数量。因此，大于 3×3 的滤波器并不总是有用的，因为总可以用 3×3 的卷积核来简化。更进一步想，是否通过分解成更小的卷积核例如 2×2 的卷积。但是，论文中提到通过非对称卷积即 $n\times1$ 甚至可以做出比 2×2 卷积更好的效果，3×1 的卷积后面接 1×3 的卷积 1.6b相当于以 3×3 的卷积的感受野滑动两层网络。因此，改进的两种新的 Inception 架构如图 1.7，进一步可改进为 1.8。

有效减小网格尺寸，作者对 Inception 进行了改进，假设开始有一个带有 k 个滤波器的 d×d 网格，如果我们想要达到一个带有 2k 个滤波器的 $\frac{d}{2}\times\frac{d}{2}$ 网格，现有以下两种方案 1.9，左边是转换为带有卷积的池化，将计算成本降为原来的 $\frac{1}{4}$，但是这样将会导致网络产生瓶颈；右边则是先采用 1×1 卷积进行升维，再进行池化



(a) Inception v1 module 5x5 convolutions (b) Inception v2 module 3x3 convolutions (c) Inception v3 module 3x1 convolutions

图 1.7　Inception 改进架构

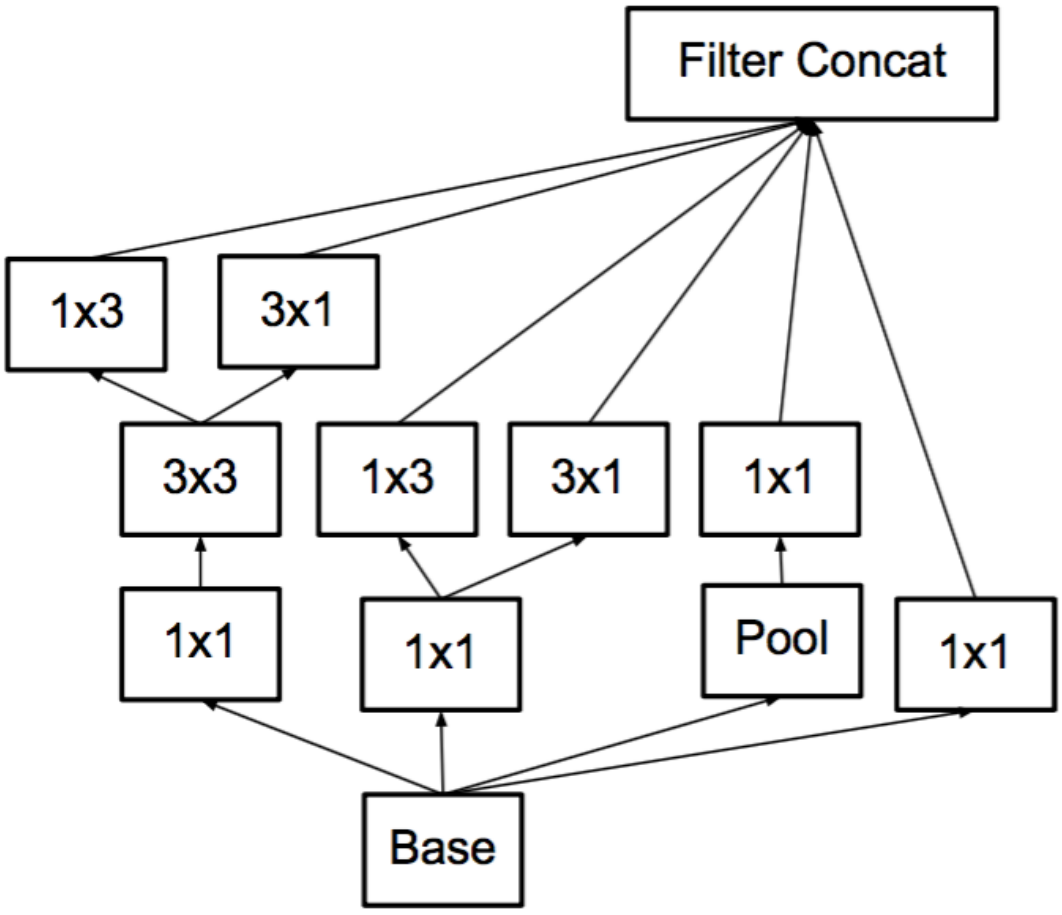


图 1.8　Inception module 改进

缩小尺寸，但这样无疑会导致计算量巨大。为此，产生了另一种变体 1.10，使用两个平行的步长为 2 的块，改进后的 Inception 模块具有缩小特征图尺寸的功能，左图是具体的操作方式，右图从特征图尺寸角度说明。

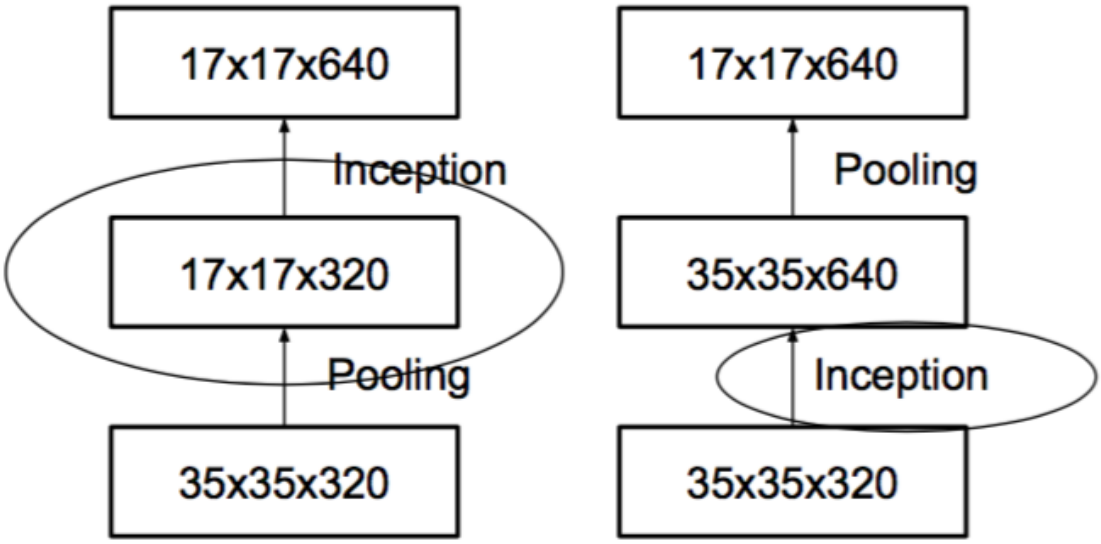


图 1.9　原始减小网格尺寸方法

Inception v2 的结构如 1.11，图中的 figure5 对应 1.7b，figure6 对应 1.7c，figure7 对应 1.8

作者等人在 k=1000 类的 ImageNet 比赛中进行了模型的测试，结果如 1.12，在 Inception-v2 行之后，变化是累积的，每一行都包含除了前面的变化之外的新变化。

Inception v3 的结构与 Inception v2 的结构类似，只是将 1.12中 Inception-v2 的所有变化都包含了。

1.5　inception v4

在 ResNet 提出之后，Szegedy 等人 [6] 考虑到残差模块对于较深的网络具有较大的意义，而 Inception 网络普遍比较深，于是提出了是否可以将残差网络与 Inception module 结合起来，结果证实了可以显著提高网络的性能。

Inception v4 并没有用到残差网络的思想，而是在 Inception v2 和 v3 的基础上进行了简化得到的，网络结构如 1.13。

其中包含有：stem 结构 1.14：

3 个 Inception 基本模块 1.15：

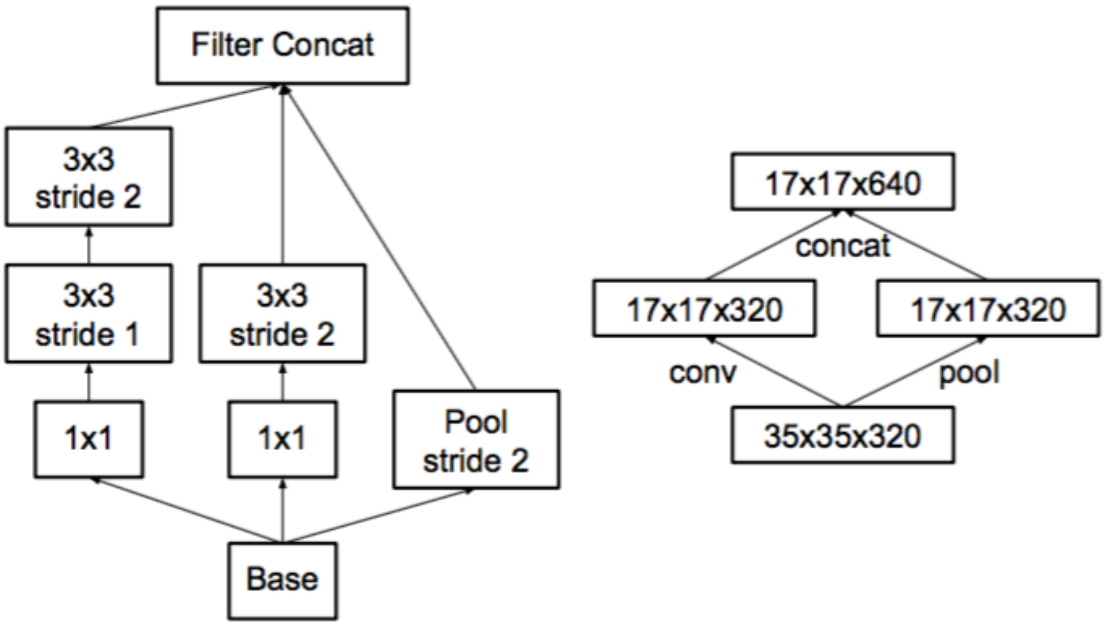


图 1.10　Inception 减小网格尺寸方法

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

图 1.11　Inception v2 结构

Network	Top-1 Error	Top-5 Error	Cost Bn Ops
GoogLeNet [20]	29%	9.2%	1.5
BN-GoogLeNet	26.8%	-	**1.5**
BN-Inception [7]	25.2%	7.8	2.0
Inception-v2	23.4%	-	3.8
Inception-v2 RMSProp	23.1%	6.3	3.8
Inception-v2 Label Smoothing	22.8%	6.1	3.8
Inception-v2 Factorized 7×7	21.6%	5.8	4.8
Inception-v2 BN-auxiliary	**21.2%**	**5.6%**	4.8

图 1.12　Inception 测试结果

2 个过渡模块 1.16：

在 Inception v4 的同篇文章中，作者等人将残差网络的思想与 Inception 结构结合起来，提出了 Inception-ResNet-v1 和 Inception-ResNet-v2 的网络，两者的网络结构相同 1.17，不同是中间使用的组件不同。

Inception-ResNet-v1 使用 1.18的 stem 结构， 1.19a、 1.19b和 1.19c的基本结构，1.16a和 1.19d的过渡模块；

Inception-ResNet-v2 使用 1.14的 stem 结构， 1.20a、 1.20b和 1.20c的基本结构，1.16a和 1.20d的过渡模块。

1.6　ResNet

He 等人 [7] 提出的 ResNet 网络取得了 2015 年的 ILSVRC 冠军。

ResNet 主要针对神经网络训练过程中的退化问题，文章中提到对浅层神经网络通过堆叠层的数量，随着模型复杂度的提高，可以提高训练的准确率。随着神经网络层数的增加，梯度消失或梯度爆炸是很常见的问题，但是这些都已经通过标准初始化或 BN 层的引入得到很好地解决。但是，当网络层数很深时，却存在了退化问题：随着网络深度的增加，准确率达到饱和，然后迅速下降，如图 1.21所示。从图中可以看出，当深度达到某一值后，更深的网络具有更高的训练误差和测试误差。为此，ResNet 网络从改变模型结构着手，使得模型更易于优化。

何为残差学习，假设 H(x) 作为几个堆叠层 (不必要是整个网络) 要拟合的基础映射，其能够拟合的映射是 F(x)，x 表示这些层中第一层的输入。为此，可以让这

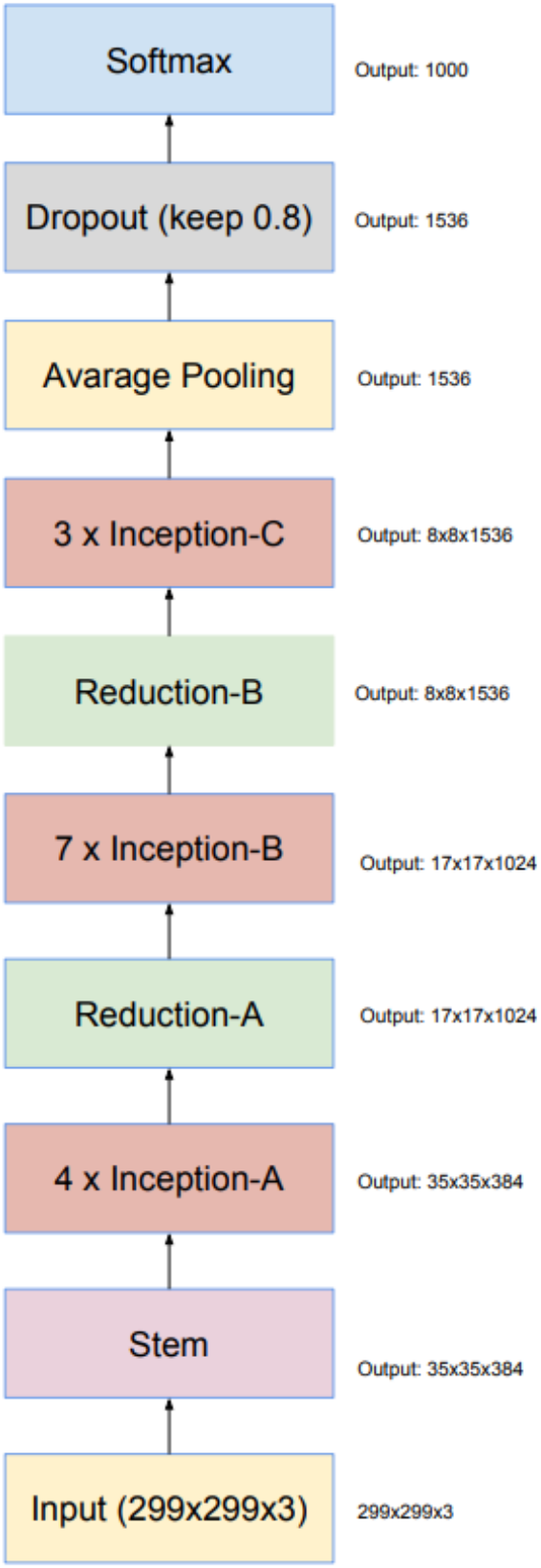


图 1.13　Inception v4

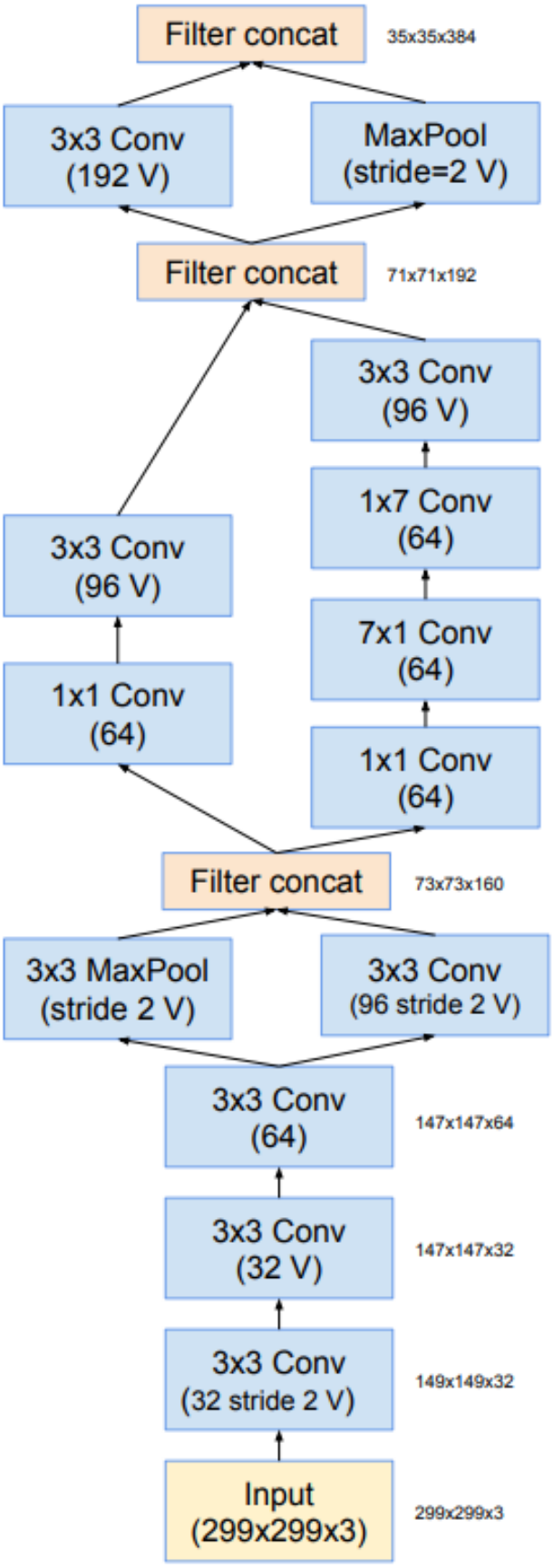
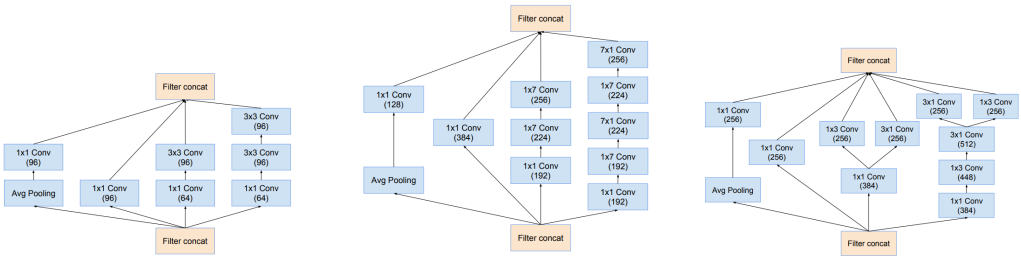
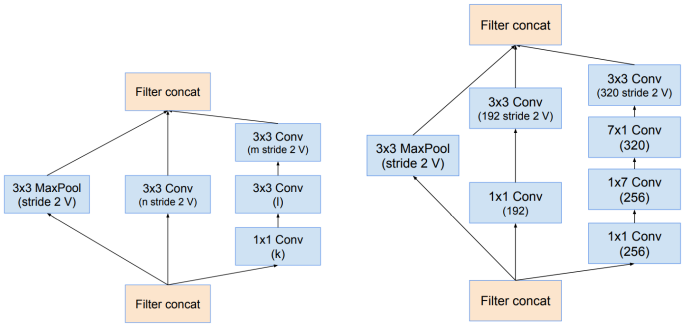


图 1.14　Inception v4 stem



(a) Inception v4 module 35x35 grid (b) Inception v4 module 17x17 grid (c) Inception v4 module 8x8 grid

图 1.15　Inception v4 基本模块



(a) Inception v4 module 35to17 (b) Inception v4 module 17to8

图 1.16　Inception v4 过渡模块

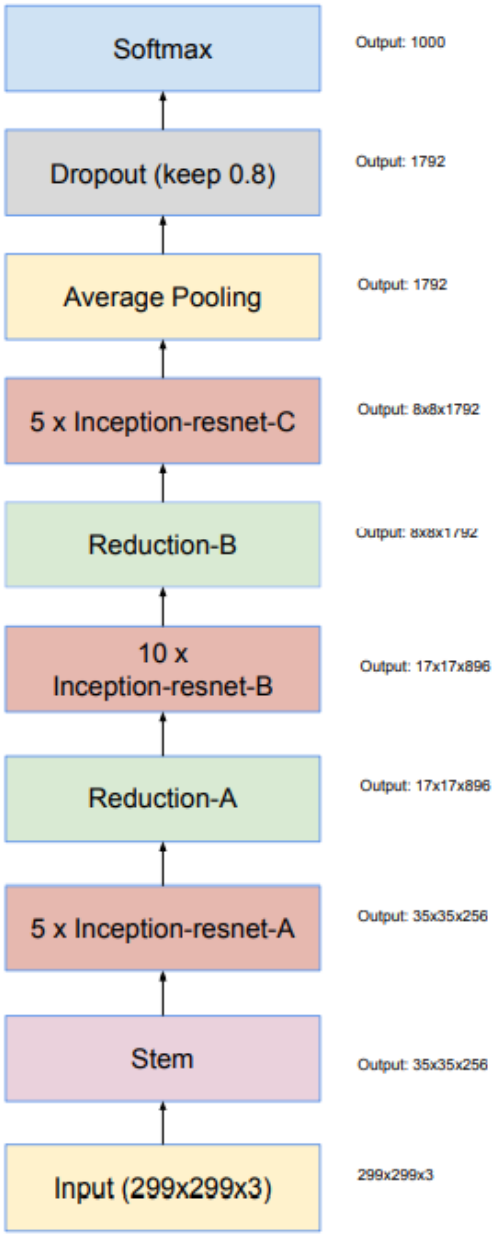


图 1.17　Inception ResNet 网络结构

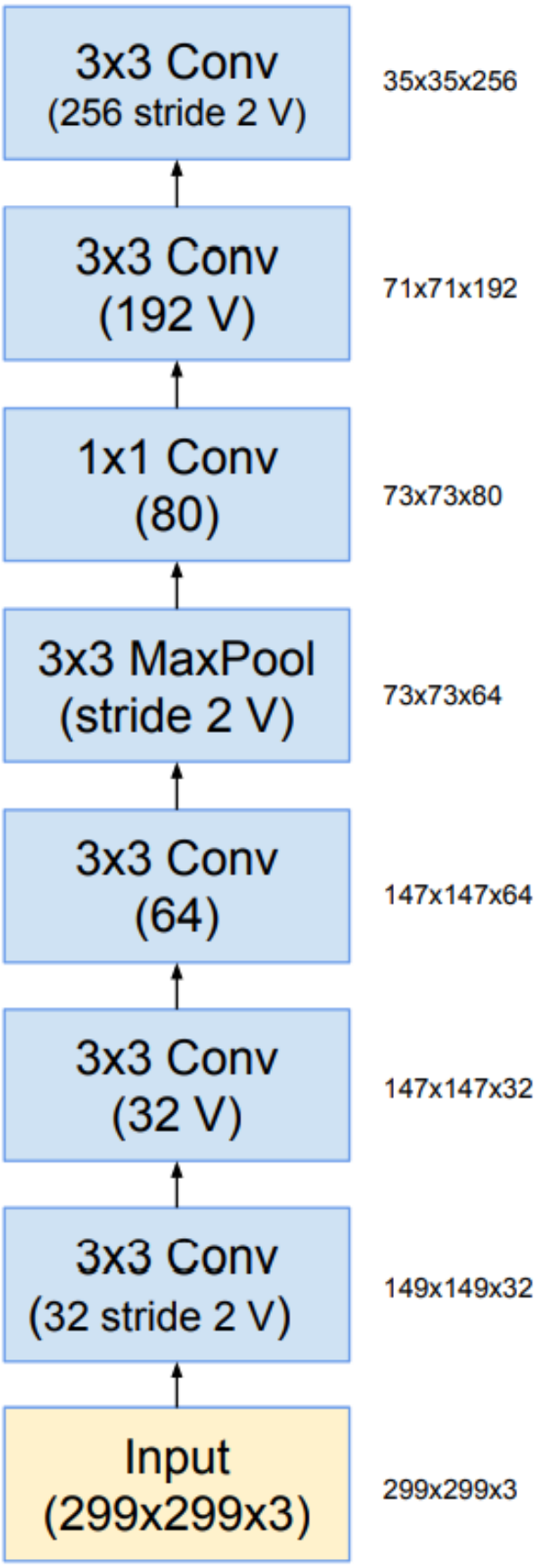
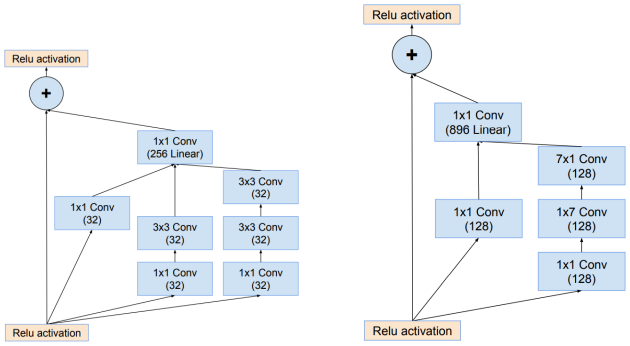
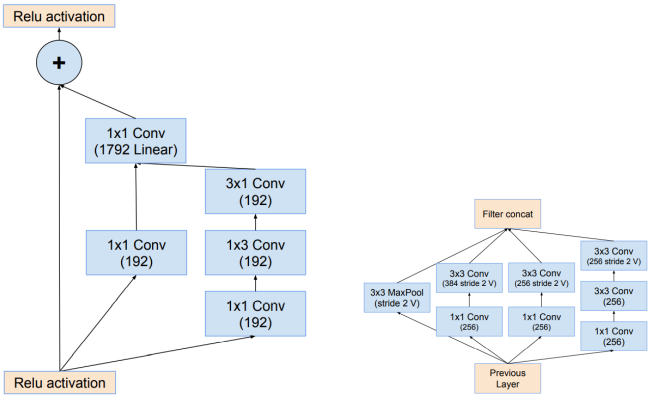


图 1.18　Inception ResNet v1 stem 结构

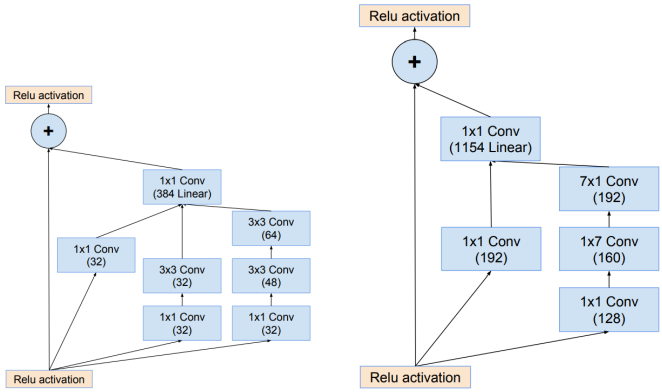


(a) Inception-ResNet-v1 module 35x35 grid (b) Inception-ResNet-v1 module 17x17 grid

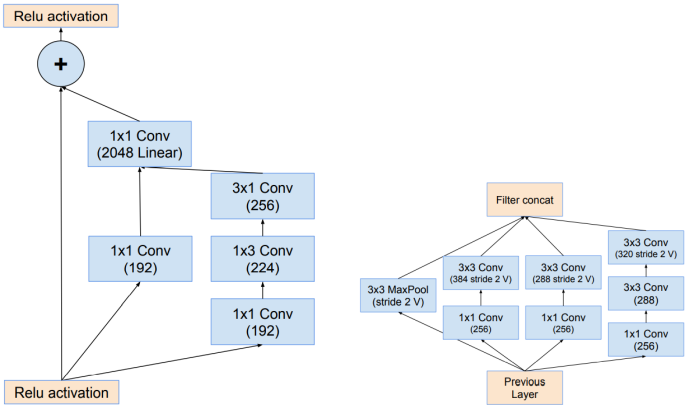


(c) Inception-ResNet-v1 module 8x8 grid (d) Inception-ResNet-v1 module 17to8

图 1.19 Inception-ResNet-v1 模块



(a) Inception-ResNet-v2 module 35x35 grid (b) Inception-ResNet-v2 module 17x17 grid



(c) Inception-ResNet-v2 module 8x8 grid (d) Inception-ResNet-v2 module 17to8

图 1.20 Inception-ResNet-v2 模块

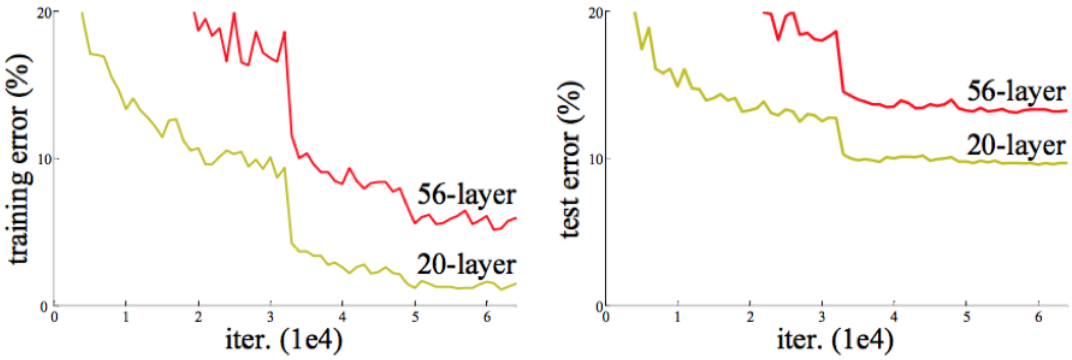


图 1.21 训练结果与层数关系

些层近似 $F(x) := H(x) - x$，而不是期望近似 H(x)，因此，原始函数变为 $F(x) + x$。

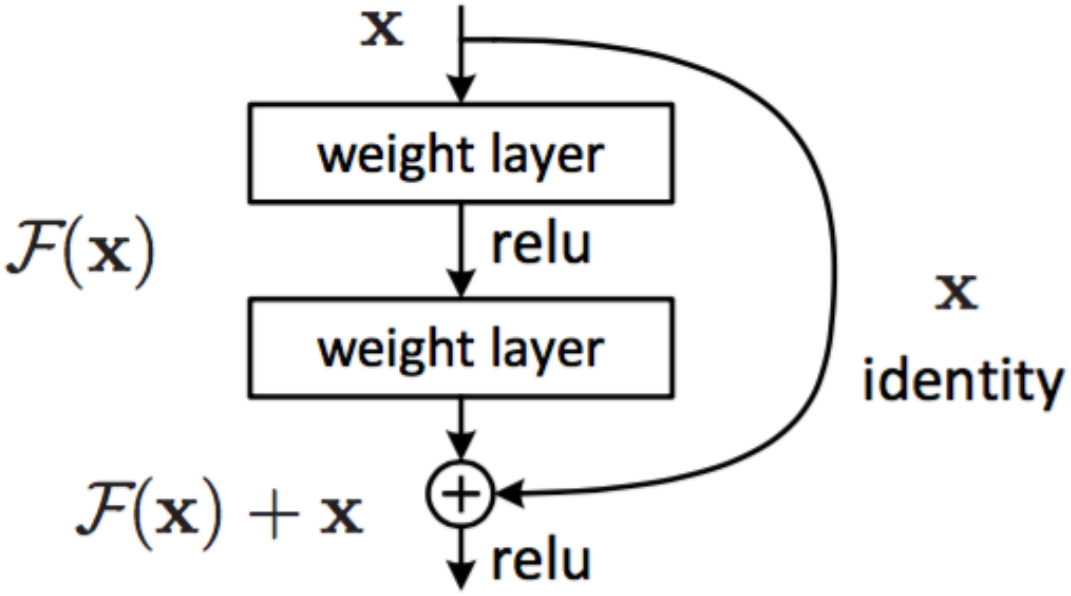


图 1.22　ResNet block 设计

对于这样的堆叠层的设计，被称为 block，如图 1.22，用于计算 $F(x) + x$。block 中包含有两个路径方向，一是计算残差 F(x)，另一种是计算 x，恒等映射。最后将 F(x) 与 x 进行累加，但是需要注意的是两者尺寸必须相同。关于 F(x) 的计算，则是 1.23中，一种是两个 3×3 的卷据构成，另一种是通过 1×1 的卷积先进行降维后升维。x 的变化则依据 F(x) 的不同有所变化，一种是输入 x 作为输出，另一种是通过 1×1 的卷积进行维度的改变，以保证 x 和 F(x) 的尺寸一致。至于不同 block 之间的连接，$F(x) + x$ 进行 ReLU 之后作为下一个 block 的输入。

ResNet 网络是通过多个 block 进行堆叠形成的，原文中 ResNet34 与 34 层简单网络 plain net、VGG 网络的对比图如图 1.24。网络参数设置如图 1.25，ResNet 的主要特点有：

1. ResNet 结构是通过堆叠 block 形成的，与简单网络相比，ResNet 中的 block 多了分支；
2. ResNet 中的下采样是通过 2×2 的卷积层实现；
3. ResNet 在图中虚线处进行下采样，并升维，即 *conv*3_1，*conv*4_1，*conv*5_1 处；
4. 最后采用平均池化，没有全连接层；

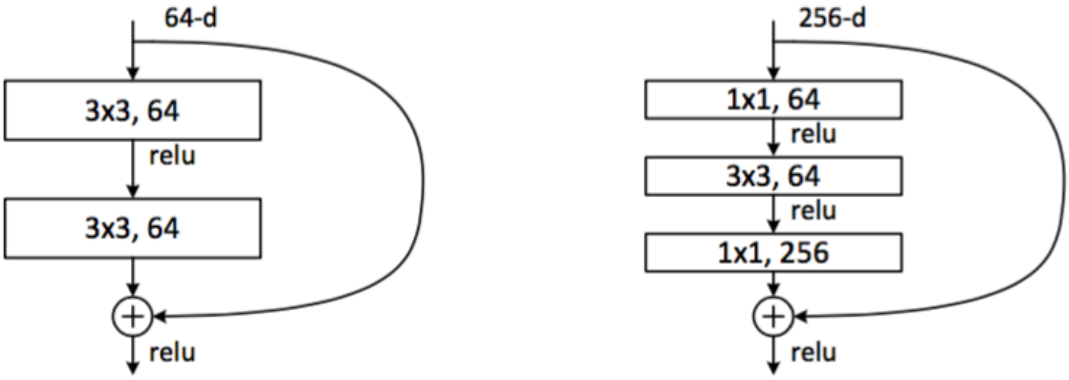


图 1.23　ResNet block F(x) 计算

5. 每个卷积层后都接有 BN；

正是由于 ResNet 使用 block 进行堆叠，可以通过对 block 的通道数量以及整个网络的 block 数量进行修改来调整网络的深度，而不用担心网络的“退化”问题。

1.7　DenseNet

目前已经提出的效果较好的网络结构中，要么是对网络深度进行加深，如 Vgg，ResNet 通过恒等映射设计了不同深度的网络结构，另一个方向则是对网络进行了加宽，如 GoogLeNet 中的 Inception 结构，而 Huang 等人于 2017 年提出的 DenseNet[8] 是在尽可能保证网络层与层之间的信息传递，使得特征图能够得到最大程度的利用，获得了 CVPR 的最佳论文奖。

无论是 ResNet 还是 Highway Networks，或者是 FractalNets，尽管在网络结构和训练过程中都一样，但都保持着从前面的层到后面的层的一个短路径。而作者等人为了保证网络中层与层之间拥有最大信息流，则是直接将网络中的所有图层相互连接，即在保证网络前馈的前提下，每个图层获取前面所有图层的输出作为输入，然后将本层的特征图贴到后面共同作为下一层的输入，示意图如 1.26：

与传统卷积神经网络不同的是，在卷积神经网络中，有 L 层，那么会存在 L 个连接；而在 DenseNet 中，则会有 $\frac{L(L+1)}{2}$ 个连接。ResNet 中，在将特征传给下一层之间会进行相加操作；而在 DenseNet 中，则是将这些特征图进行拼接，即第 l 层有 l 个输入，前面 l 层的特征图，本层的特征图则传递给之后的层。

DenseNet 的优点在于不用学习多余的特征图，所以参数较少。传统的前馈神

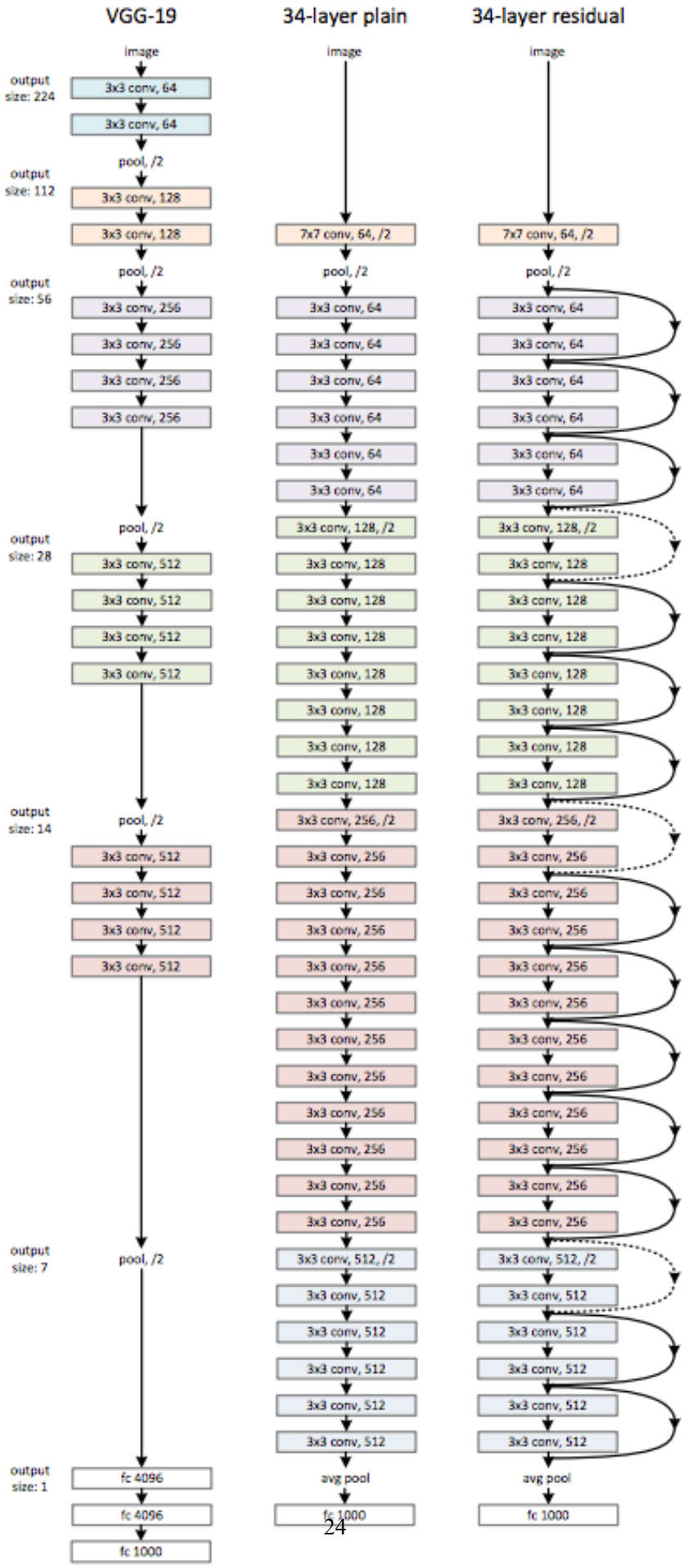


图 1.24　ResNet 网络结构对比图

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 1.25　ResNet 网络结构参数图

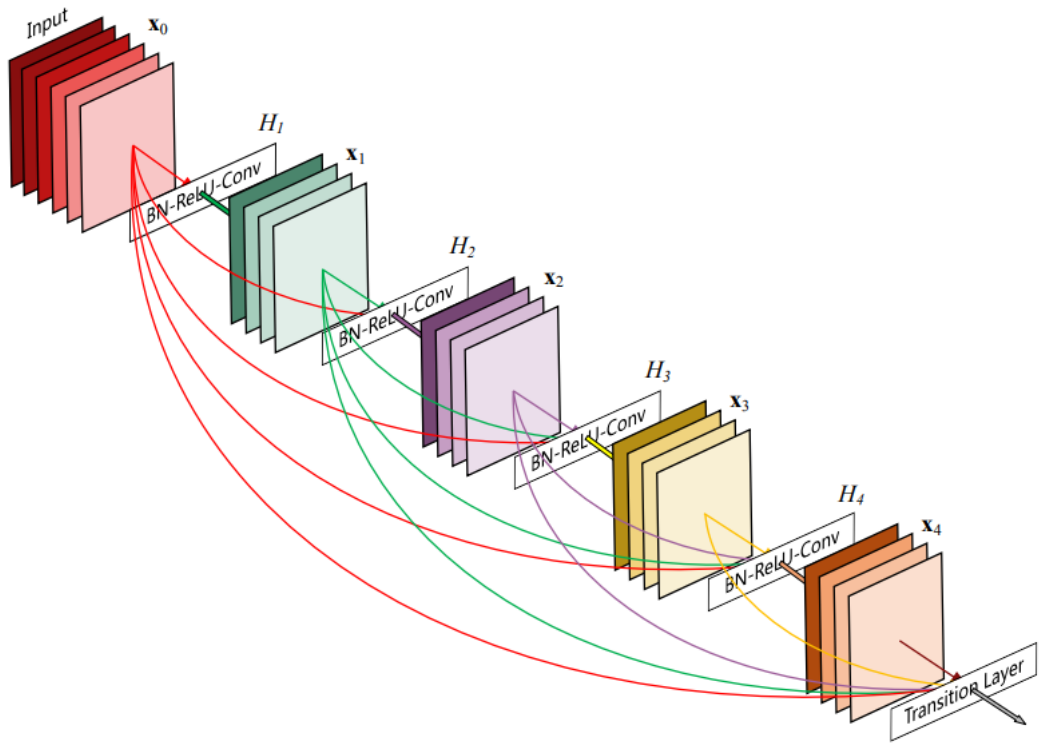


图 1.26　DenseNet 连接示意图

经网络中，可以看作每一层都从前一层读取状态，然后送入下一层中，每一层在改变状态的同时也将自己保存的信息送出去了。DenseNet 在信息传递时做了区分，信息在网络中得以保存。同时，DenseNet 的另一个优点在于在整个网络的训练过程中，信息和梯度的改进，从而更容易收敛；同时，这样的密集连接也能起到一个正则化的作用，能够有效防止过拟合。

一个典型的 DenseNet 结构如 1.27所示，图中包含有多个 dense block。

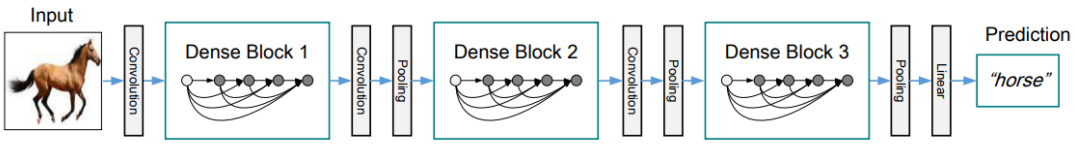


图 1.27　DenseNet 结构示意图

假设一张图片 x_0 在卷积网络中传播，网络有 L 层，每一层都实现了非线性转换函数 H_l，l 表示第 l 层，H_l 是包含了 BN、ReLU、池化和卷积的组合操作，在 DenseNet 中，H_l 是 BN、ReLU 和 3×3 卷积操作的组合，第 l 层的输入用 x_l 表示。

那么在传统的神经网络中，将 l-1 层的输出 x_{l-1} 作为第 l 层的输入，得到第 l 层的输出 x_l，可表示为 $x_l = H_l(x_{l-1})$。在 ResNet 中，添加了 skip-connection，可表示为 $x_l = H_l(x_{l-1}) + x_{l-1}$。在 DenseNet 中，则采用密集连接的方式，第 l 层将前面所有层的特征图 $[x_0, x_1, x_2, ..., x_{l-1}]$ 作为输入，可表示为 $x_l = H_l([x_0, x_1, x_2, ..., x_{l-1}])$。

Layers	Output Size	DenseNet-121($k=32$)	DenseNet-169($k=32$)	DenseNet-201($k=32$)	DenseNet-161($k=48$)
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48	[1 × 1 conv; 3 × 3 conv] × 36
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 24
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

图 1.28　DenseNet 网络配置

DenseNet 在 ImageNet 数据集上的配置 1.28所示，其中含有：

增长速率 k(Growth rate)，函数 H_l 产生 k 个特征图，那么第 l 层有 $k_0+k\times(l-1)$ 个输入特征图，k_0 表示输入层的通道数，k 就被称为增长速率，控制着每一层能有多少信息对整体网络的状态产生影响，实验证明较小的增长速率就能取得很好的效果。

Bottleneck layers，在 3×3 卷积层前面加上 1×1 卷积，可以减少输入的特征图数量，并提高计算效率。

Compression，如果一个 dense block 包含有 m 个特征图，那么就让之后的过渡层 (1×1 卷积层后接 2×2 平均池化层) 生成 θ_m 个输出特征图，$0<\theta\leq1$ 被称为压缩系数。

1.8 SENet

hu 等人 [9] 于 2017 年中提出的 SENet 获得了 2017 年的 ImageNet 挑战赛冠军。作者以另外一种方式思考现有的卷积神经网络实现方式，设计一种新型的结构取得了很好的效果。

传统的卷积神经网络是通过考虑增加网络深度或者宽度的方式从空间维度的角度以提升网络性能，而作者等人从特征通道的角度提出了 SENet，关键步骤是 squeeze 和 excitation 操作，采用自适应的方式对不同特征通道表达的信息进行学习，对有用的特征进行提升以便更好的训练，而抑制那些没有用或者用处不大的特征。SE 模块示意图如 1.29所示，对于一个输入特征通道数为 c_1 的输入 x，经过一系列操作之后特征通道数为 c_2。对于任意的一组变换 $F_{tr}:X\rightarrow U, X\in\mathbb{R}^{W'\times H'\times C'}, U\in\mathbb{R}^{W\times H\times C}$，可以是卷积或者一组卷积，之后构造一组 SE 模块来完成特征重新校准的操作。特征 U 首先经过 squeeze 操作在空间维度上对一个 $W\times H$ 的二维矩阵完成聚合特征映射产生特征通道描述符。该描述符能够表示特征通道响应的全局分布，使较低层能够获得网络全局感受野。之后 excitation 操作为每个特征通道分配一定的权重。然后特征 U 被重新加权作为 SE 模块的输出。

squeeze 操作，从空间维度的方面来进行特征压缩，将每一个二维的特征通道转换成一个实数，通过使用全局平均池化生成通道描述符。$z\in\mathbb{R}^C$ 是对特征 U 在空间维度 $W\times H$ 上进行收缩形成的，那么 z 中的第 c 个元素的计算形式为：

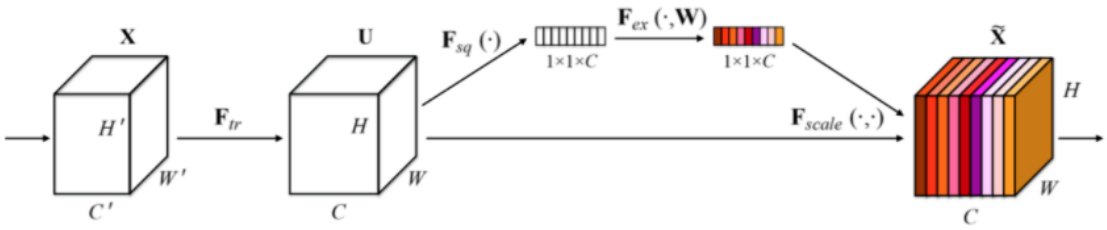


图 1.29　SE 模块示意图

$$z_c = F_{sq}(U_c) = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} u_c(i,j)$$

excitation 操作，采用了简单的类似于循环神经网络中的门机制，并使用 sigmoid 函数激活：

$$s = F_{ex}(z, W) = \sigma(g(z, W)) = \sigma(W_2 \delta(W_1 z))$$

δ 表示 ReLU 激活函数，降维层参数 $W_1 \in \mathbb{R}^{\frac{C}{r} \times C}$，降维比例是 r，一般设置为 16，升维层参数 $W_2 \in \mathbb{R}^{C \times \frac{C}{r}}$。SE 模块的最终输出通过重新调节带有激活的变换输出 U 得到：

$$\tilde{X}_c = F_{scale}(U_c, s_c) = s_c \cdot U_c$$

$\tilde{X} = [\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_C]$ 和 $F_scale(U_c, s_c)$ 指的是特诊映射 $U_c \in \mathbb{R}^{W \times H}$ 和标量 s_c 之间的对应通道乘积。

SE 模块具有优秀的灵活性，可以嵌入到流行的网络架构如 Inception 和 ResNet 系列中，将 F_{tr} 看作一个 Inception 模块，可以构建一个 SE-Inception 网络 1.30：

采用全局池化作为 squeeze 操作，之后采用两个全连接层先降维再进行升维操作，这比单独使用一个全连接层具有了更多的非线性关系，可以更好的拟合通道间复杂的相关性，同时在很大程度上减少了参数量和计算量。之后采用 sigmoid 激活函数获得 0-1 之间归一化的权重，最后通过 Scale 操作将归一化的权重加权到每个通道的特征上。

此外，SE 模块还可以嵌入到使用 skip-connections 的网络中，构建 SE-ResNet 网络 1.31，过程和 SE-Inception 基本一致。

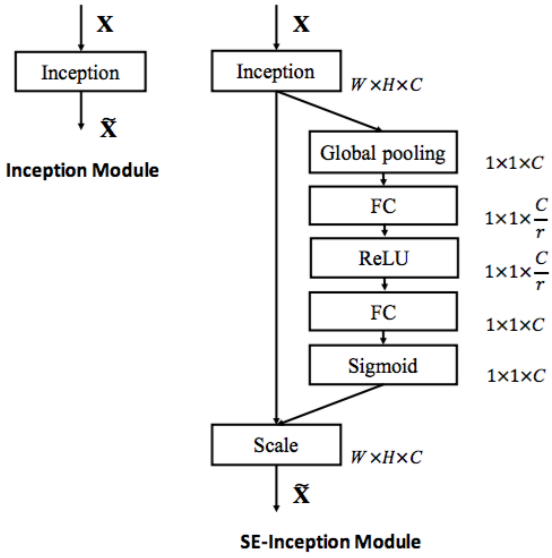


图 1.30　SE-Inception 示意图

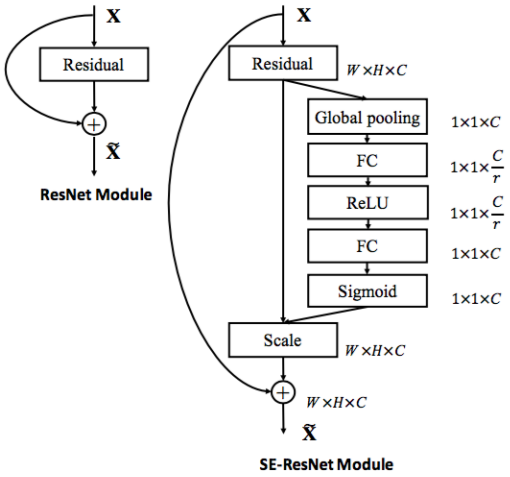


图 1.31　SE-ResNet 示意图

参考文献

[1] HINTON G E, KRIZHEVSKY A, SUTSKEVER I. Imagenet classification with deep convolutional neural networks[J]. Advances in neural information processing systems, 2012, 25 : 1106 – 1114.

[2] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[3] SZEGEDY C, LIU W, JIA Y, et al. Going deeper with convolutions[A]. Proceedings of the IEEE conference on computer vision and pattern recognition[C], 2015 : 1 – 9.

[4] IOFFE S, SZEGEDY C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[J]. arXiv preprint arXiv:1502.03167, 2015.

[5] SZEGEDY C, VANHOUCKE V, IOFFE S, et al. Rethinking the inception architecture for computer vision[A]. Proceedings of the IEEE conference on computer vision and pattern recognition[C], 2016 : 2818 – 2826.

[6] SZEGEDY C, IOFFE S, VANHOUCKE V, et al. Inception-v4, inception-resnet and the impact of residual connections on learning[A]. Proceedings of the AAAI Conference on Artificial Intelligence : Vol 31[C], 2017.

[7] HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition[A]. Proceedings of the IEEE conference on computer vision and pattern recognition[C], 2016 : 770 – 778.

[8] HUANG G, LIU Z, VAN DER MAATEN L, et al. Densely connected convolutional networks[A]. Proceedings of the IEEE conference on computer vision and pattern recognition[C], 2017 : 4700 – 4708.

[9] HU J, SHEN L, SUN G. Squeeze-and-excitation networks[A]. Proceedings of the IEEE conference on computer vision and pattern recognition[C], 2018 : 7132 – 7141.