《机器学习基础》

卷积神经网络

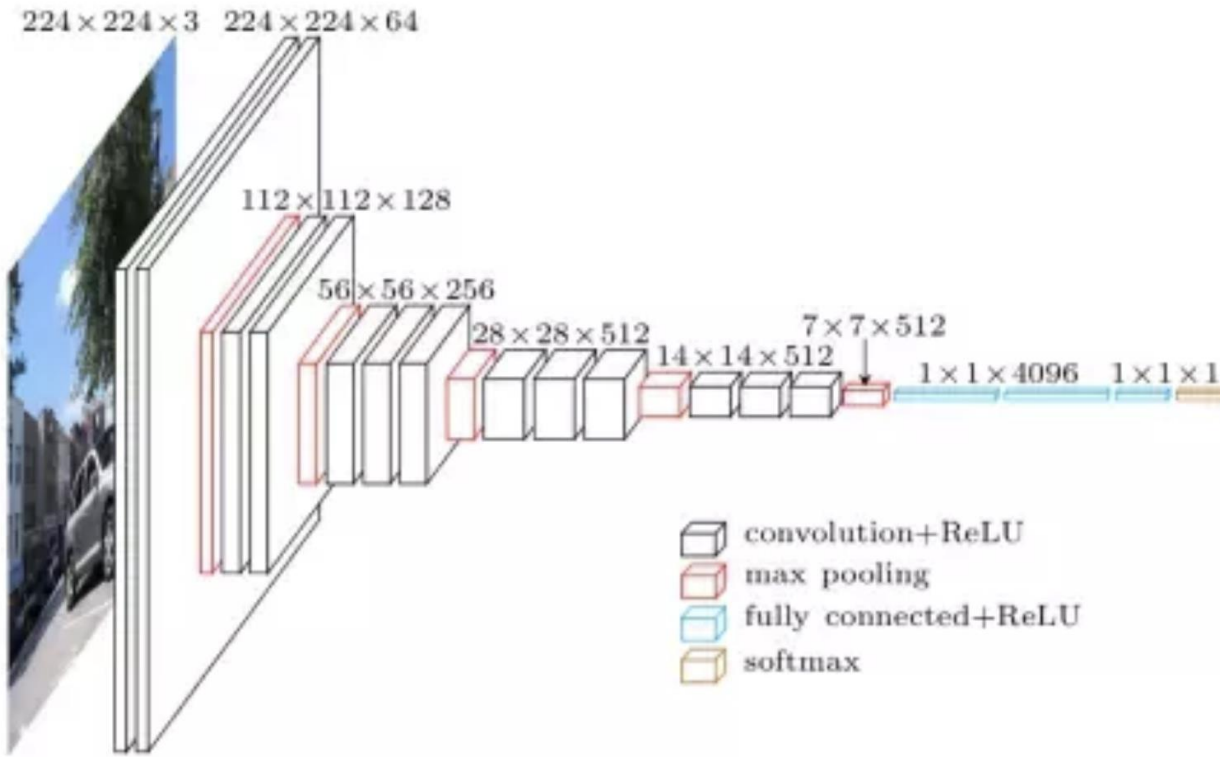
常见的卷积神经网络模型

- LeNet-5
- AlexNet-8
- VGG
- ResNet
- InceptionNet (GoogLeNet)

卷积神经网络参数选择的几个问题

- 1. 卷积神经网络的层数
- 2. 卷积核的大小
- 3. 学习效率和学习准确度

VGG

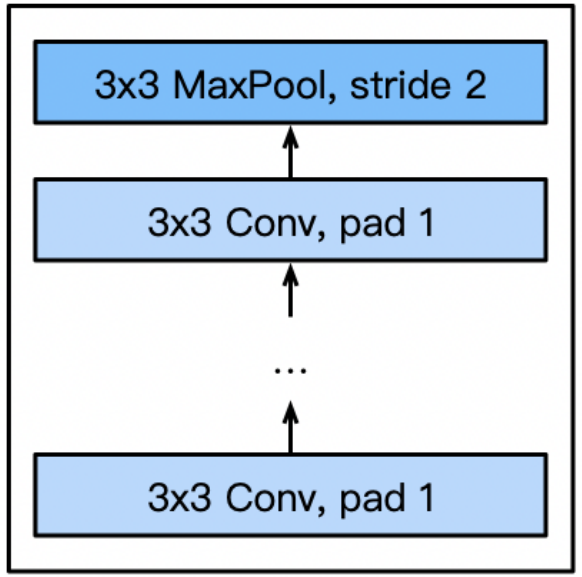


ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	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					

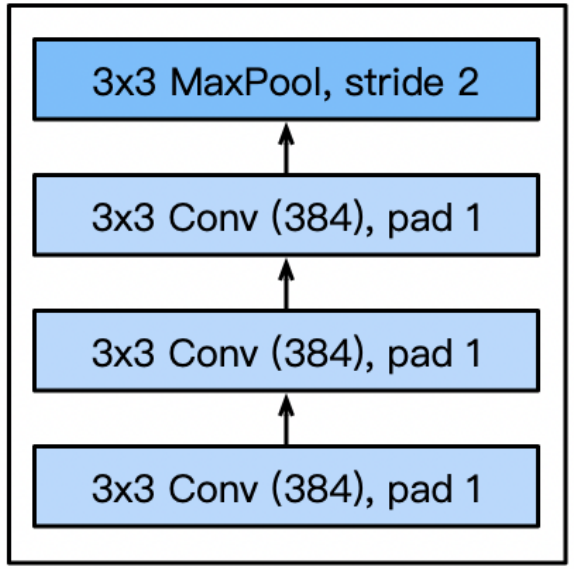
VGG 块

- 更深还是更宽?
  - 5x5 卷积
  - 3x3 卷积（更多）
  - 更深和更窄更好
- VGG 块
  - 3x3 卷积（填充=1）（n层，m个通道）
  - 2x2 最大池化层（步幅=2）

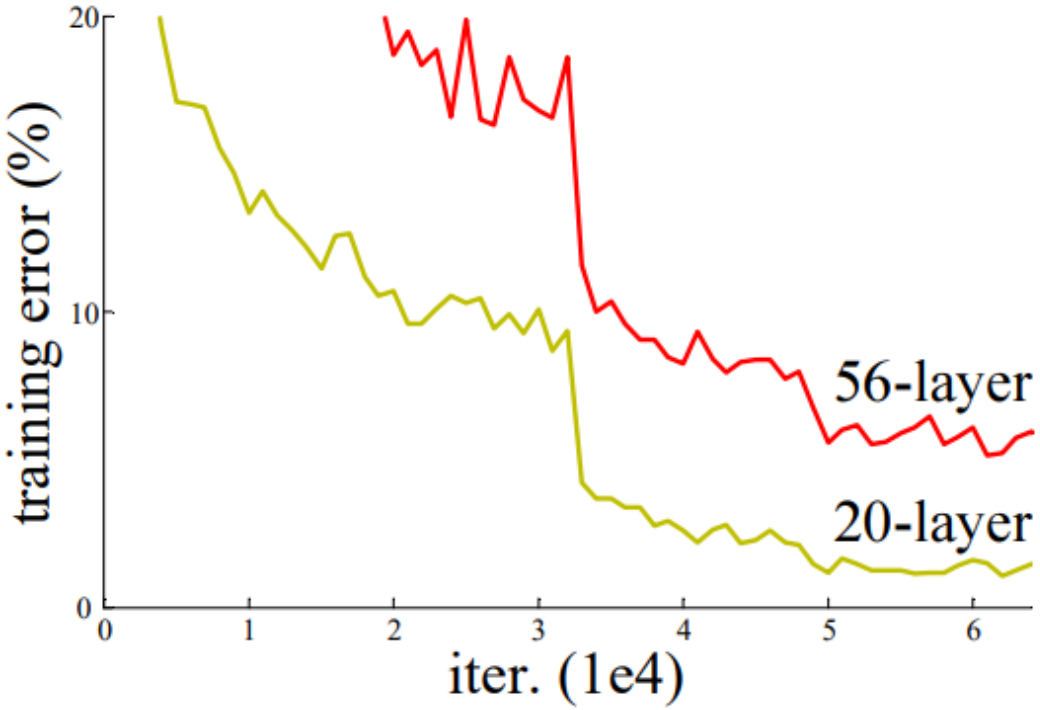
VGG 块

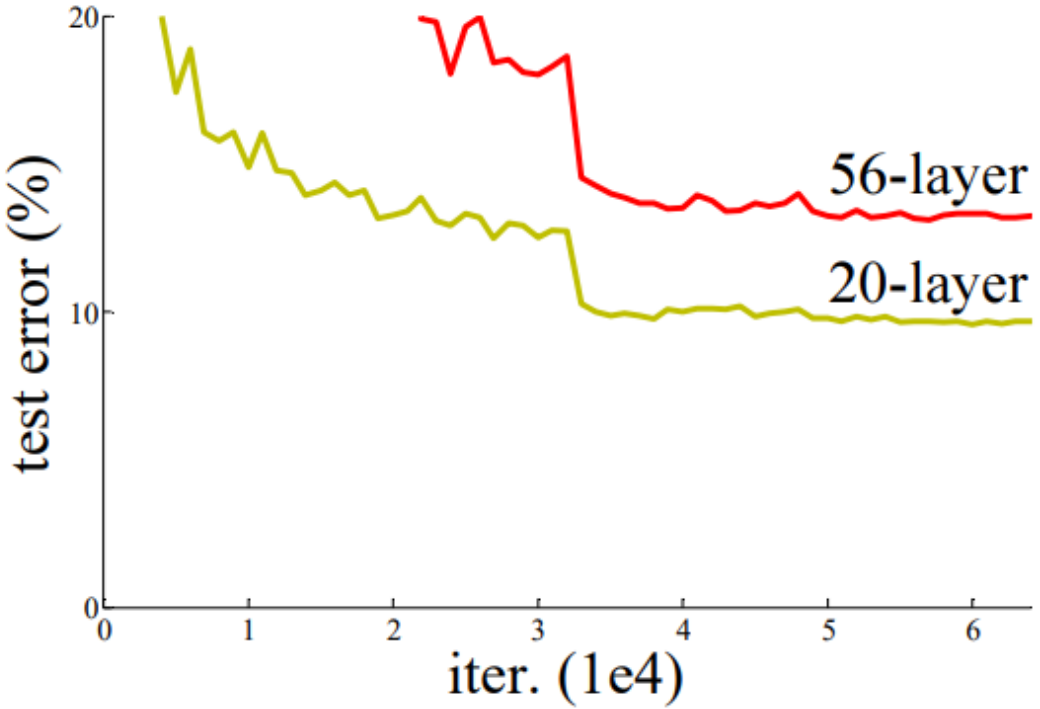


AlexNet 一部分



VGG


20
10
0
training error (%)
0
1
2
3
4
5
6
iter. (1e4)
56-layer
20-layer


20
10
0
test error (%)
0
1
2
3
4
5
6
iter. (1e4)
56-layer
20-layer

残差网络

- 是一种深度卷积神经网络（CNN）架构，由微软研究院提出，2015年在ImageNet竞赛中获得了冠军。
  - 假设在一个深度网络中，我们期望一个非线性单元（可以为一层或多层的卷积层）f(x,θ)去逼近一个目标函数为h(x)。
  - 将目标函数拆分成两部分：恒等函数和残差函数

$$h(\mathbf{x}) = \underbrace{\mathbf{x}}_{\text{恒等函数}} + \underbrace{\big(h(\mathbf{x}) - \mathbf{x}\big)}_{\text{残差函数}} \rightarrow f(\mathbf{x}, \theta)$$

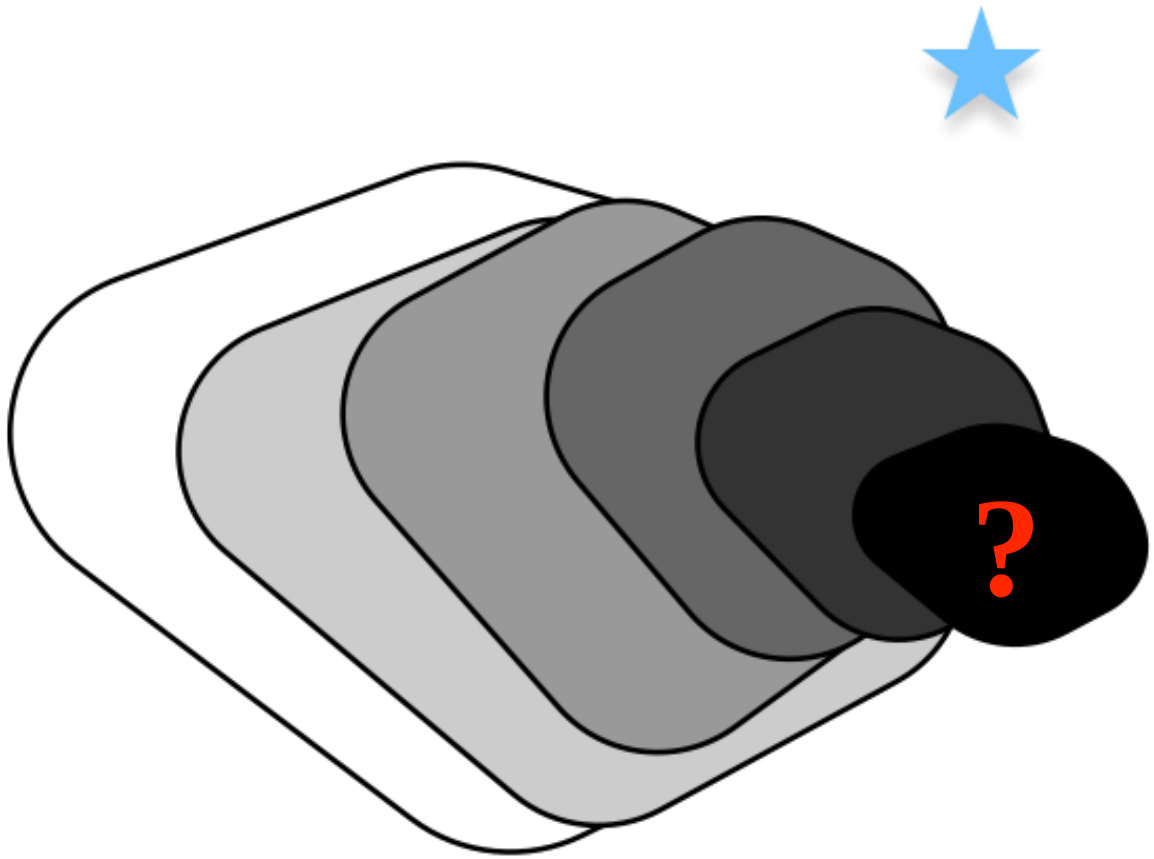
残差网络

- 是一种深度卷积神经网络（CNN）架构，由微软研究院提出，2015年在ImageNet竞赛中获得了冠军。
  - ResNet的核心思想是引入了“残差学习”来解决深度网络训练中的退化问题，即随着网络层数的增加，网络的性能反而下降的问题。
  - 在传统的深度网络中，如果网络层数增加到一定程度，网络将难以训练，并且准确率先是增加然后饱和，再增加层数甚至会导致准确率下降。这是因为，随着层数的增加，网络中的梯度在反向传播过程中会逐渐消失或爆炸，导致网络无法学习到有效的特征表示。
  - ResNet通过引入跳跃连接（skip connections）或称为shortcut connections来解决这个问题。这些连接允许网络中的信号绕过一些层直接传播，从而使得网络可以学习到恒等映射（identity mapping），即直接传递输入到输出而不经过任何变化。这样，即使网络非常深，也可以保证网络至少能够学习到一个有效的恒等映射，从而保证了网络性能不会随着层数增加而下降。

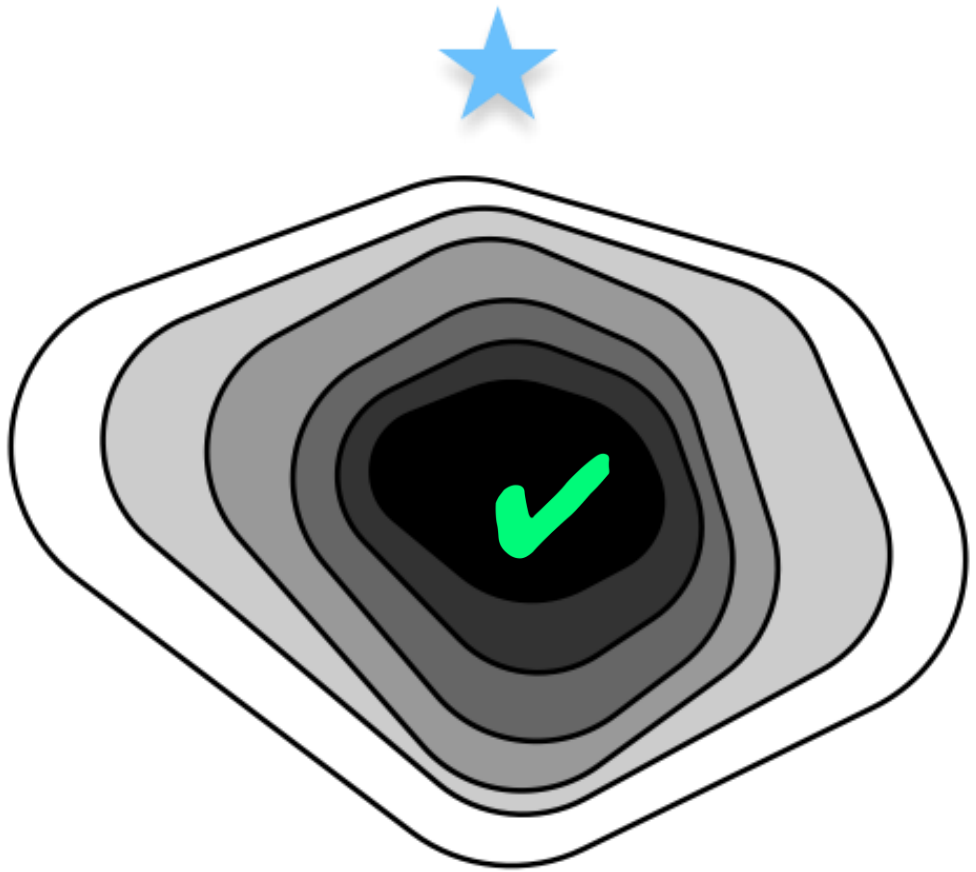
ResNet

- 2015 ILSVRC winner （152层）
  - 错误率：3.57%
  - 超深的网络结构(突破1000层)
  - 提出residual模块
  - 使用Batch Normalization加速训练(丢弃dropout)

添加层会提高准确性吗?

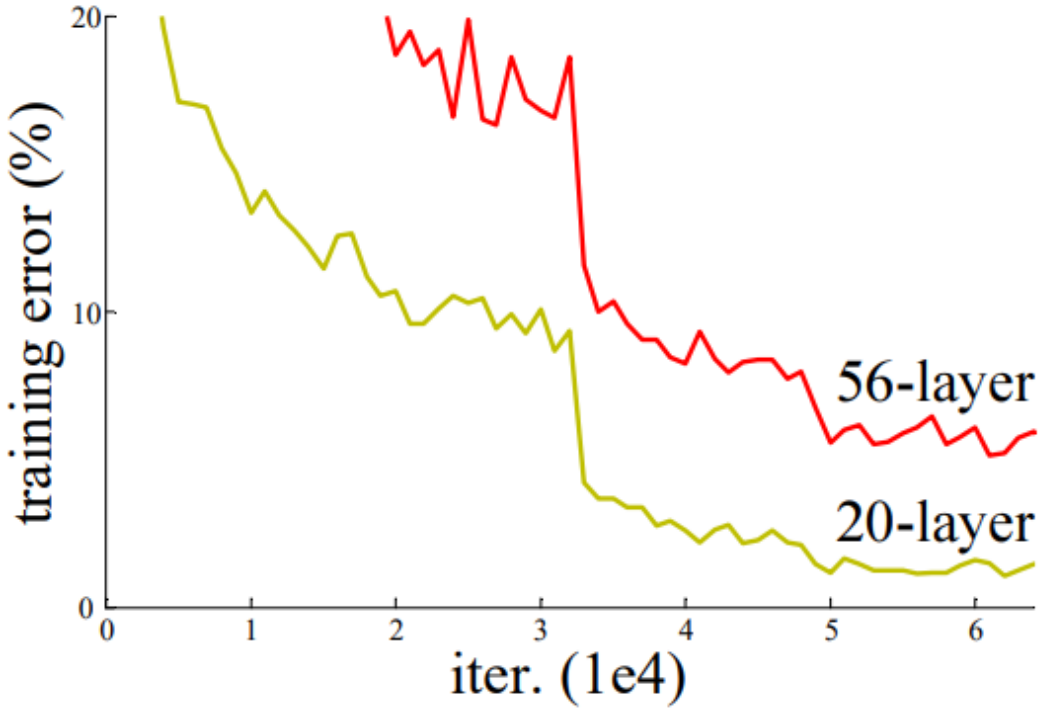


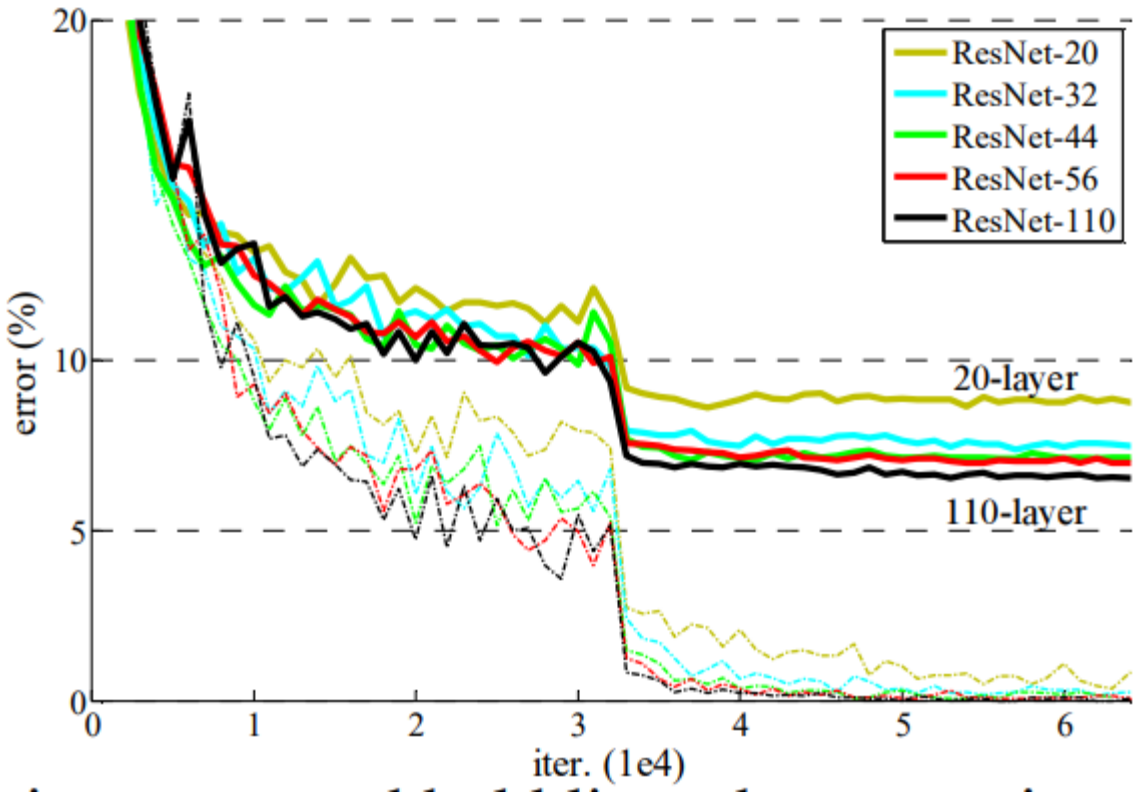
generic function classes



nested function classes

添加层会提高准确性吗?


training error (%)
iter. (1e4)
56-layer
20-layer

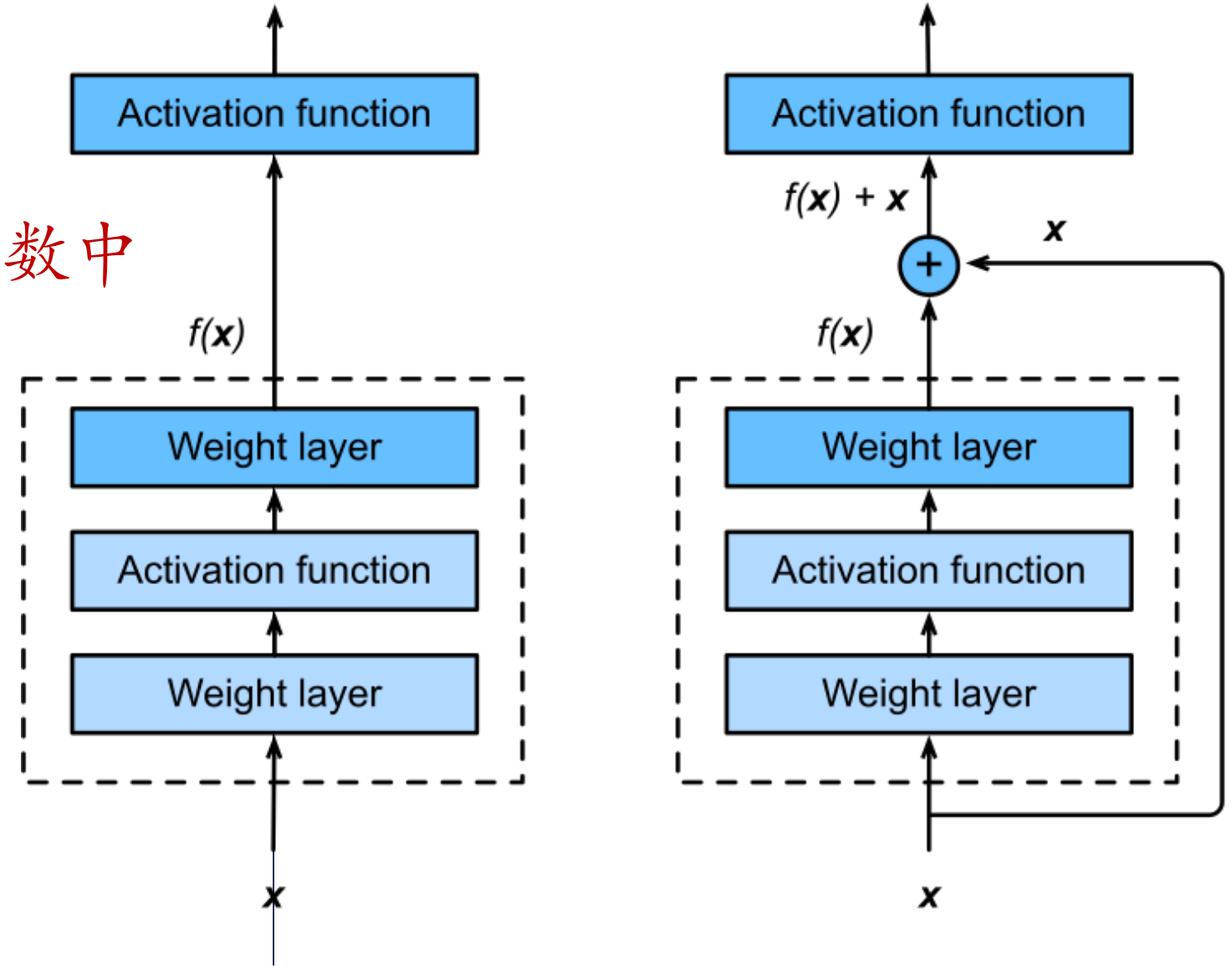

error (%)
iter. (1e4)
ResNet-20
ResNet-32
ResNet-44
ResNet-56
ResNet-110
20-layer
110-layer

残差网络（ResNet）

- 添加层会更改原特征类

- 我们想要“加”到原函数中“泰勒展开”的参数

$$f(x) = x + g(x)$$



1x1的卷积核

- 经卷积后的矩阵尺寸大小计算公式为：

$$\text{Out} = \frac{\text{Inp} - \text{W} + 2\text{P}}{S} + 1$$

-

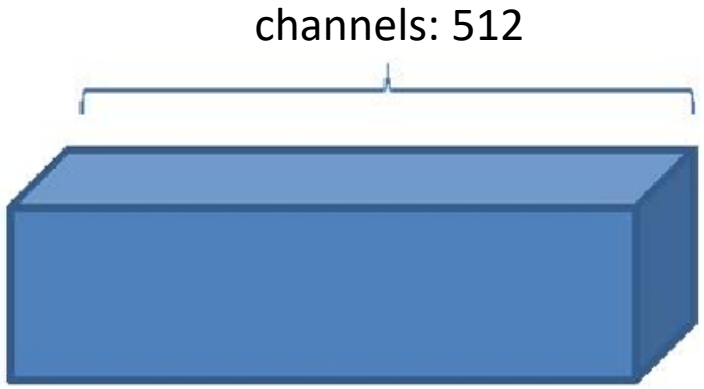
① 输入图片大小 Inp× Inp
② Filter大小 W× W
③ 步长 S
④ padding的像素数 P

1x1的卷积核

- Padding: Valid
  - $Out = \frac{Inp - W + 1}{S}$向上取整
  - ① 输入图片大小 Inp× Inp
    ② Filter大小 W× W
    ③ 步长 S
- Padding: Same
  - $Out = \frac{Inp}{S}$向上取整

1x1的卷积核

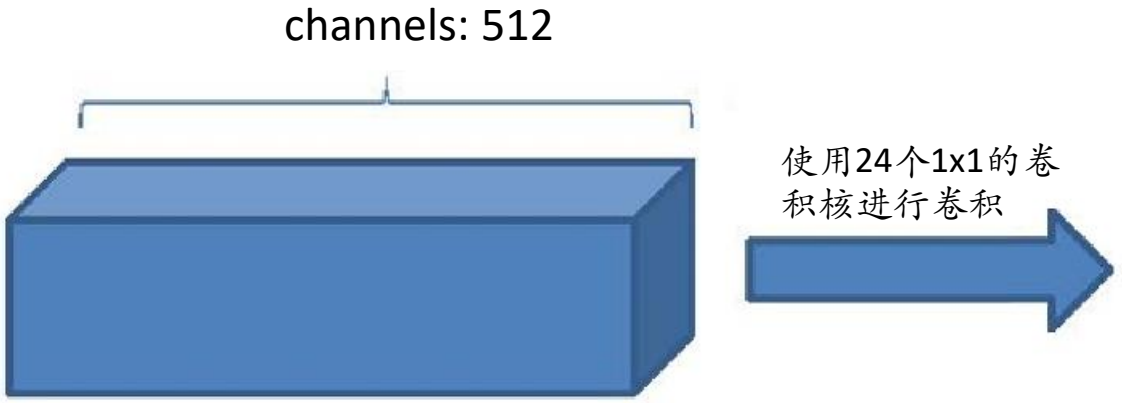
- GoogLeNet详解
  - 不使用1x1卷积核降维



使用64个5x5的卷积核进行卷积

$$5\times5\times512\times64=819\,200$$

  - 使用1x1卷积核降维

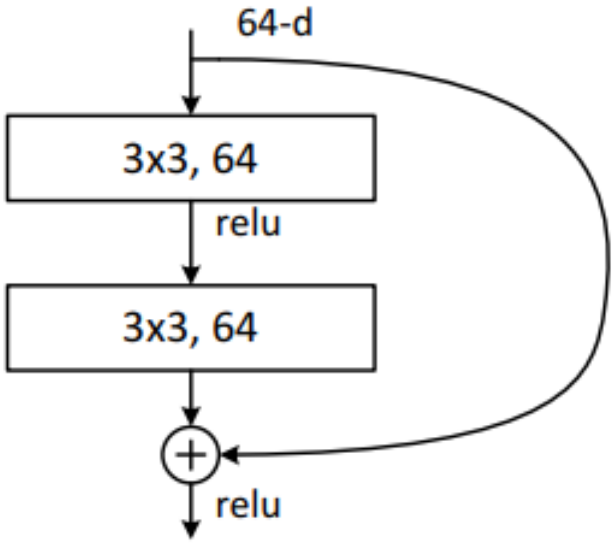


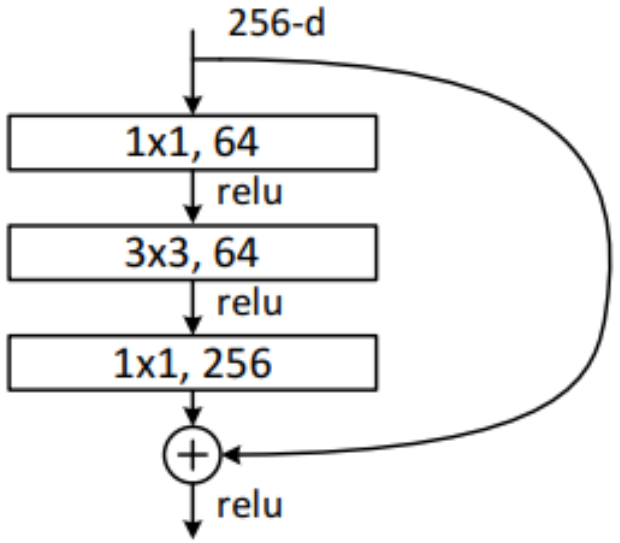
$$1\times1\times512\times24=12\,288$$

$$5\times5\times24\times64=38\,400$$

$$12\,288+38\,400=50\,688$$

残差块

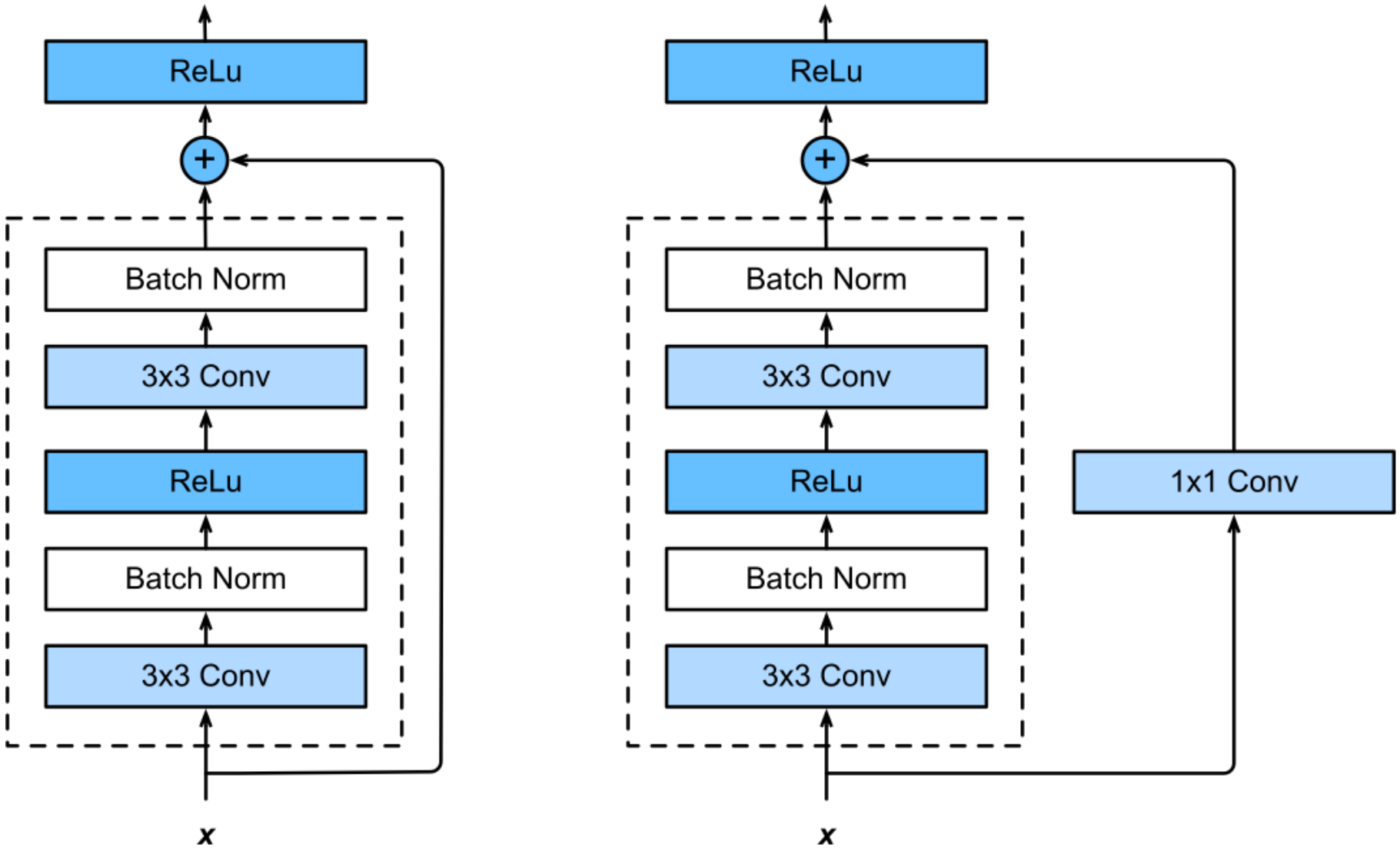




3x3x256x256+3x3x256x256=1,179,648

1x1x256x64+3x3x64x64+1x1x64x256=69,632

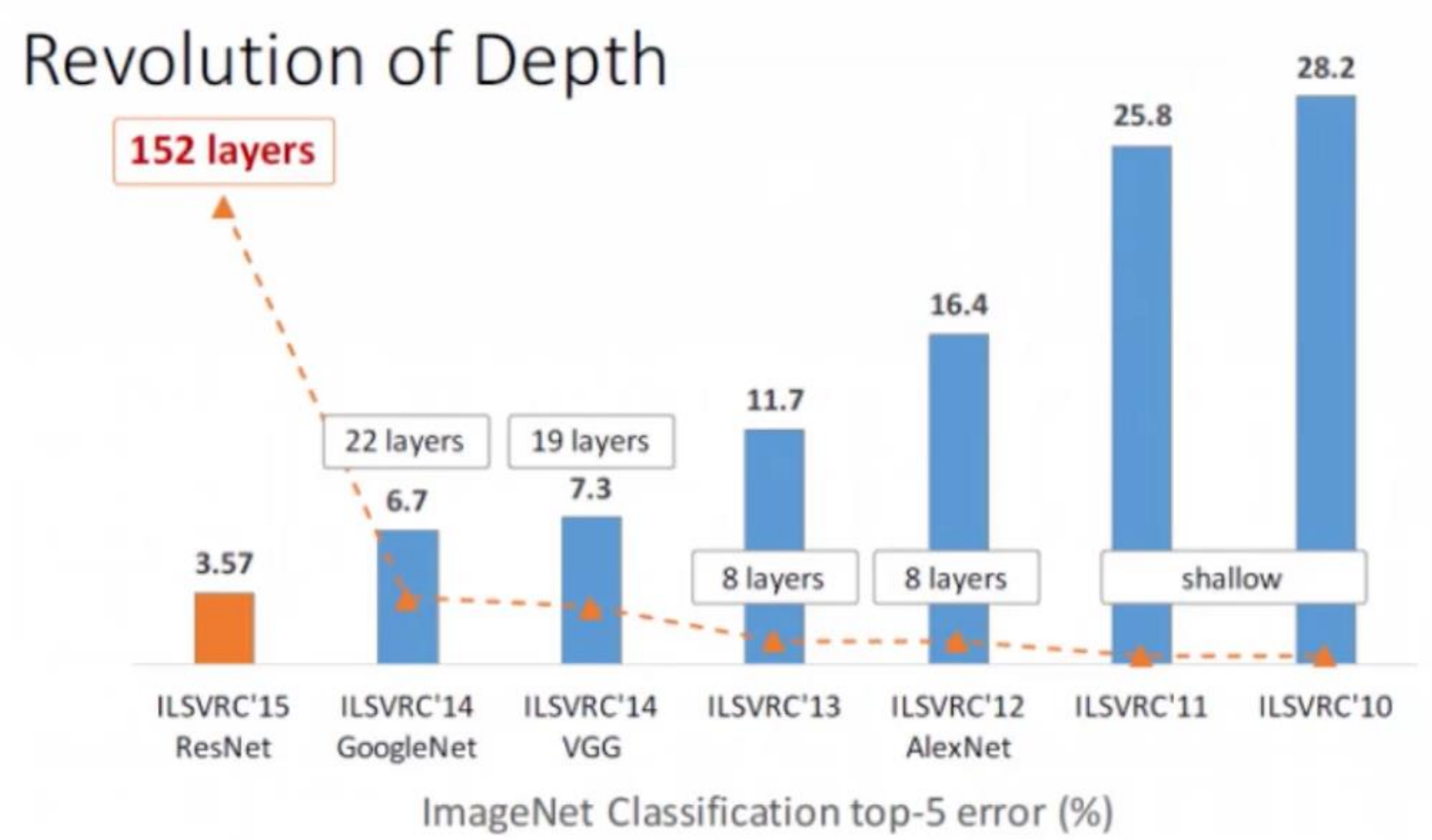
残差块


ReLu
Batch Norm
3x3 Conv
ReLu
Batch Norm
3x3 Conv
x
ReLu
Batch Norm
3x3 Conv
ReLu
Batch Norm
3x3 Conv
1x1 Conv
x

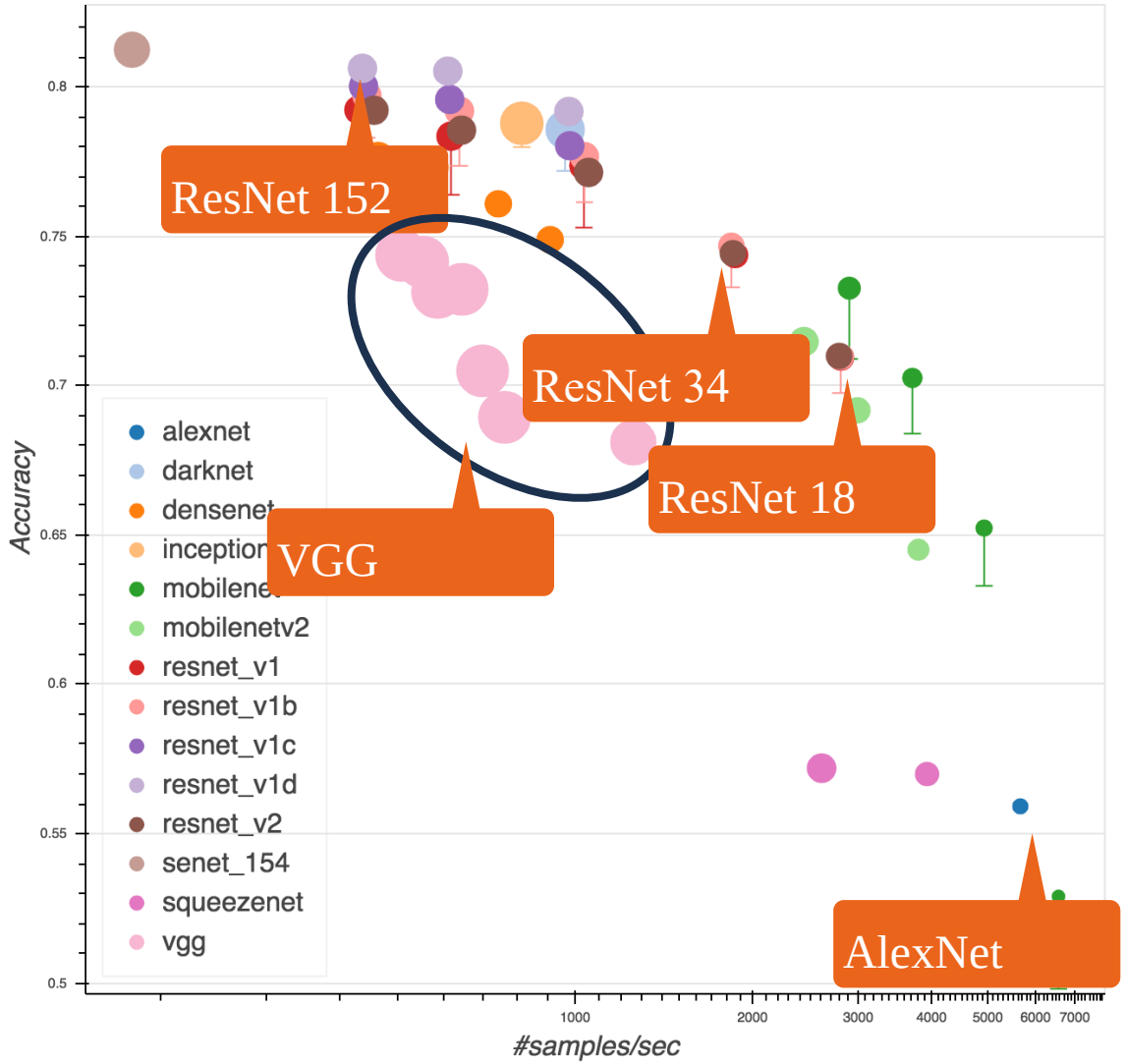
ResNet详解

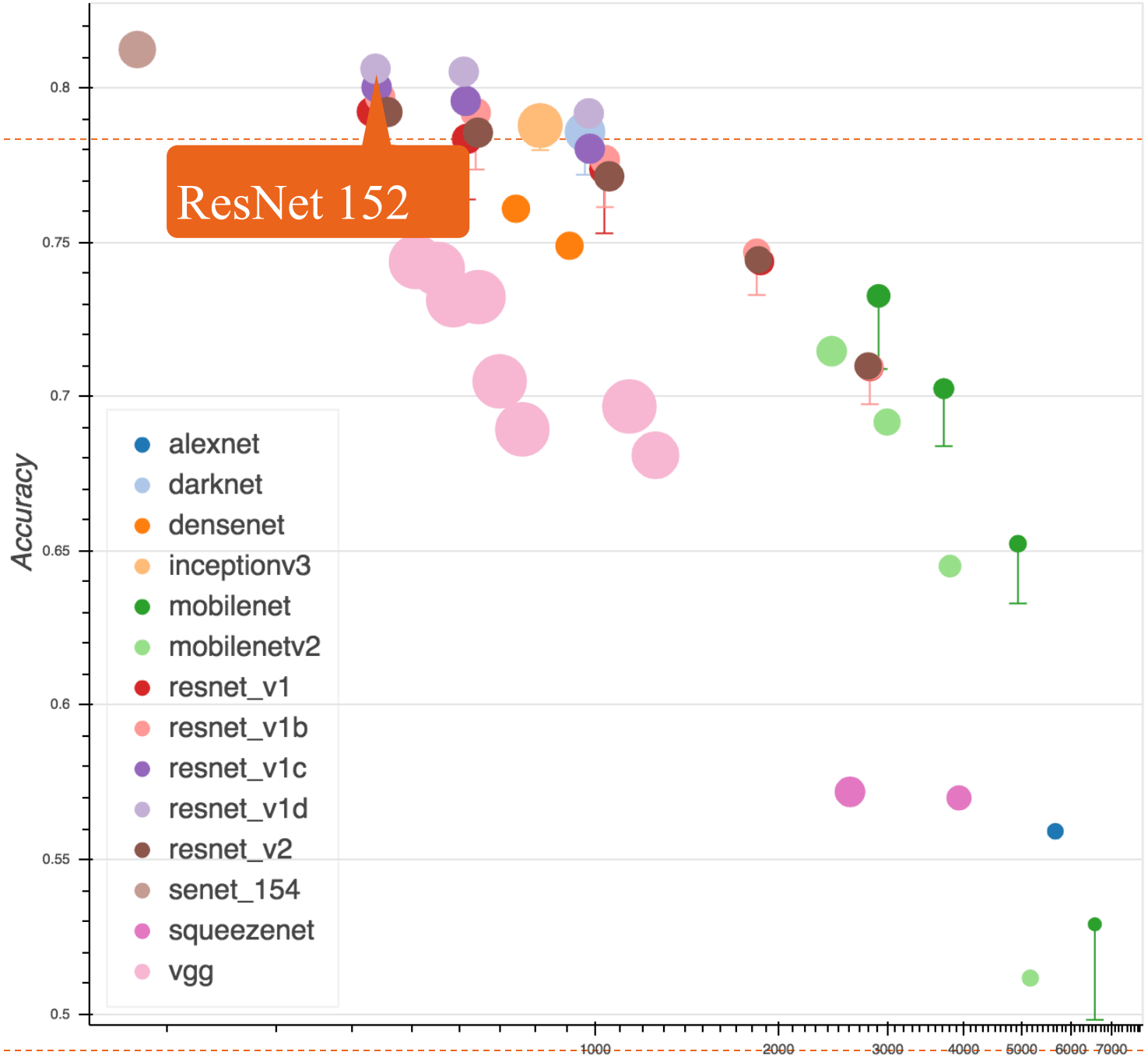
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
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

ResNet


Revolution of Depth
152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15 ResNet
ILSVRC'14 GoogleNet
ILSVRC'14 VGG
ILSVRC'13
ILSVRC'12 AlexNet
ILSVRC'11
ILSVRC'10
ImageNet Classification top-5 error (%)

ResNet


ResNet 152
ResNet 34
ResNet 18
VGG
AlexNet
alexnet
darknet
densenet
inception
mobilenet
mobilenetv2
resnet_v1
resnet_v1b
resnet_v1c
resnet_v1d
resnet_v2
senet_154
squeezenet
vgg
Accuracy
#samples/sec
0.8
0.75
0.7
0.65
0.6
0.55
0.5
1000
2000
3000
4000
5000
6000
7000



GluonCV 模型“动物园”

https://gluon-cv.mxnet.io/model_zoo/classification.html

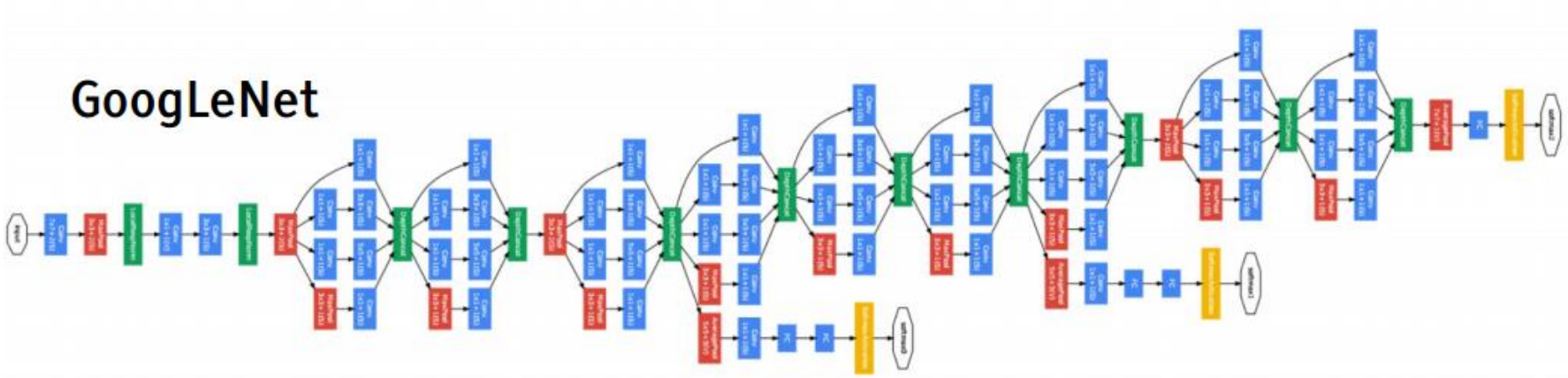
ResNet

- 残差块使得很深的网络更加容易训练
  - 甚至可以训练一千层的网络

- 残差网络对随后的深层神经网络设计产生了深远影响，无论是卷积类网络还是全连接类网络。

Inception网络

- 2014 ILSVRC winner （22层）
  - 参数：GoogLeNet：4M VS AlexNet：60M
  - 错误率：6.7%
  - Inception网络是由有多个inception模块和少量的汇聚层堆叠而成。



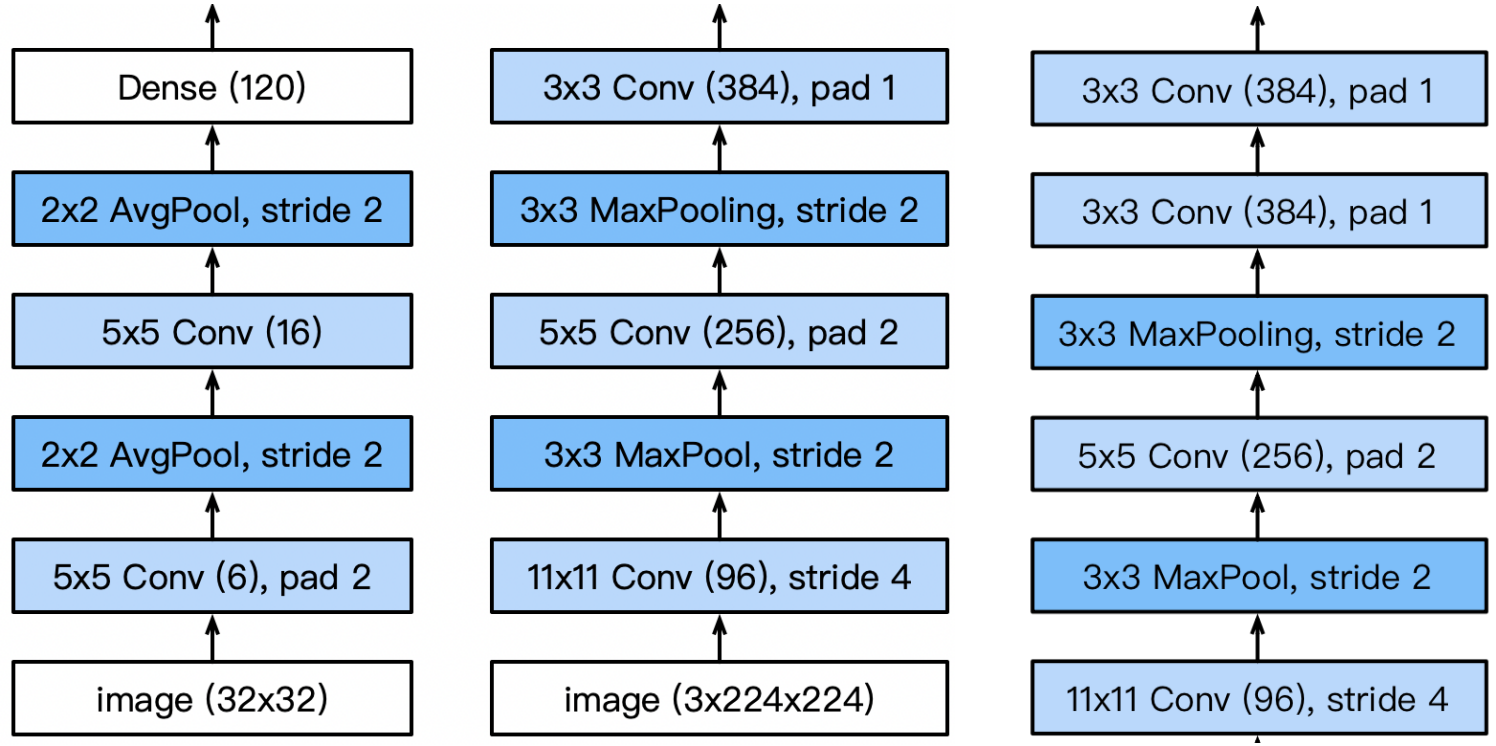
GoogLeNet

- 引入了Inception结构（融合不同尺度的特征信息）
- 使用1x1的卷积核进行降维以及映射处理
- 添加两个辅助分类器帮助训练
- 丢弃全连接层，使用平均池化层（大大减少模型参数）

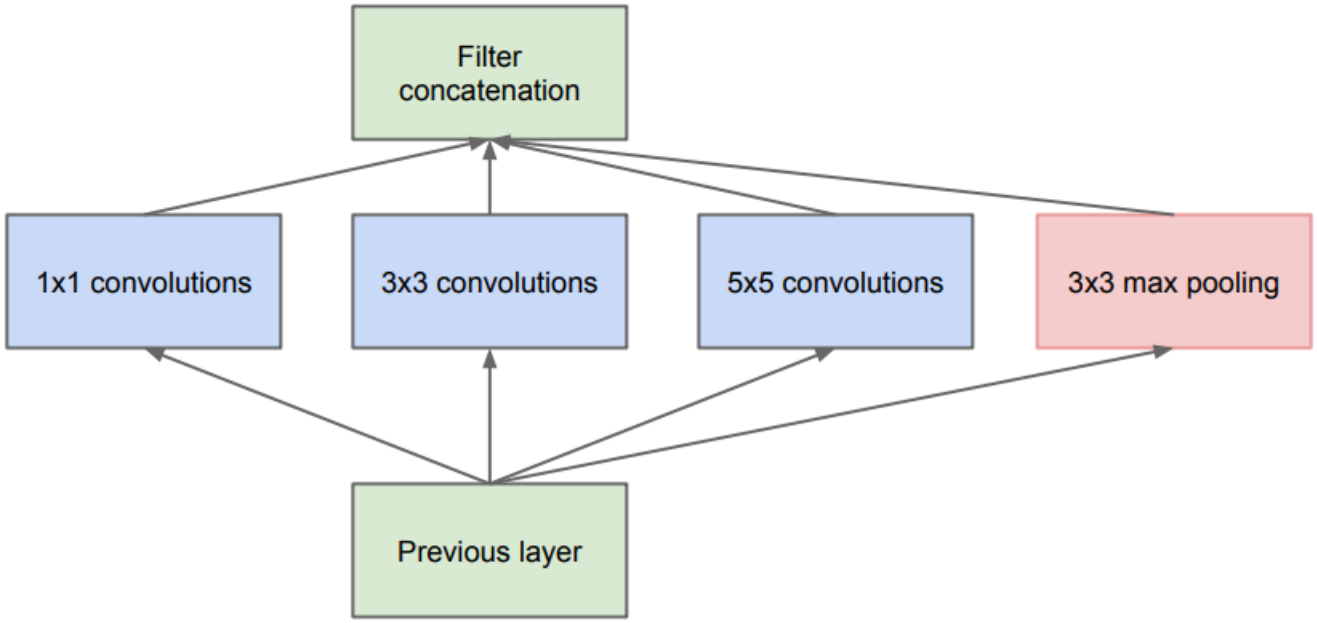
选最合适的卷积 …

1x1　3x3　5x5　最大池化

LeNet　AlexNet　VGG

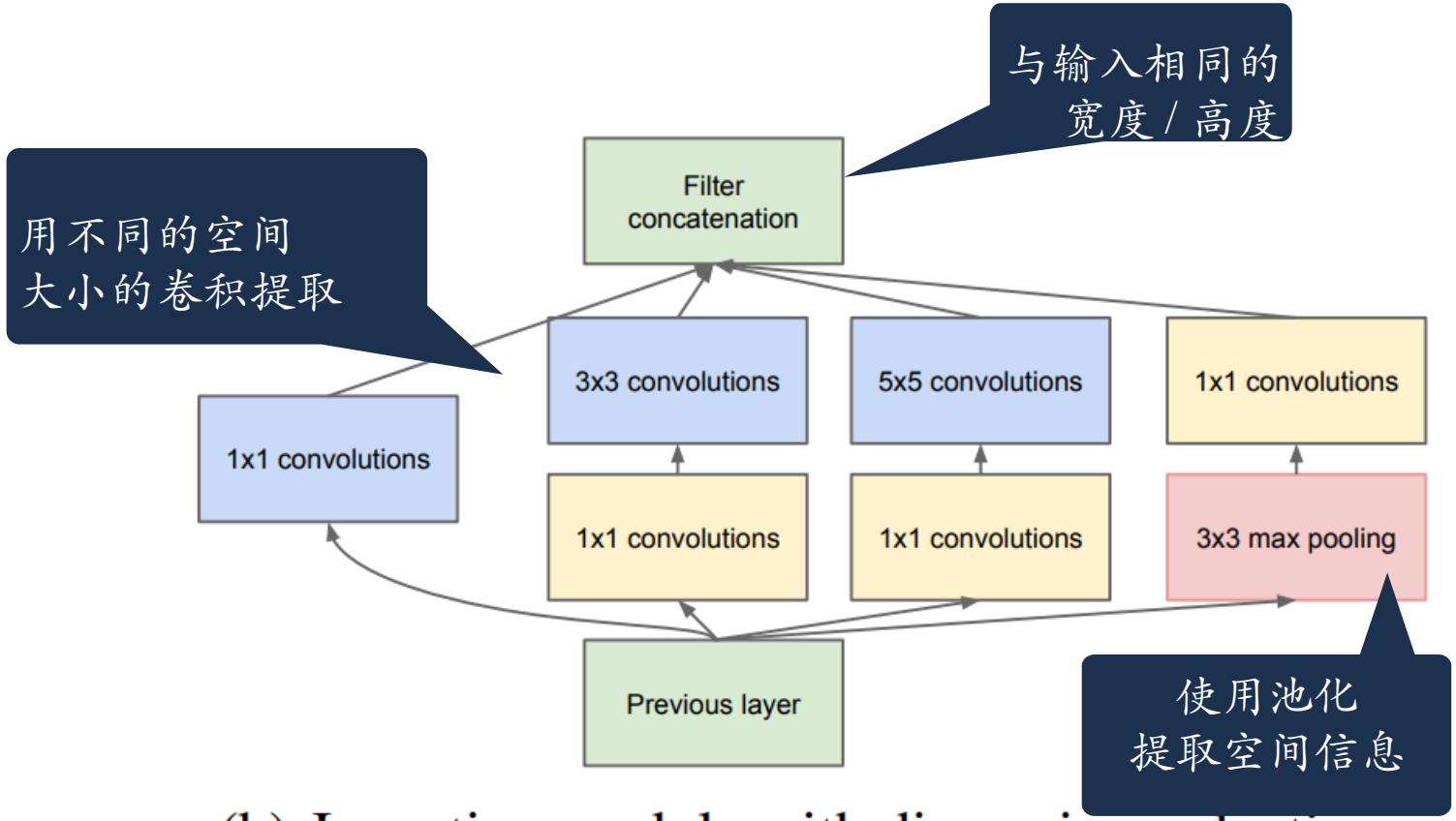


Inspection结构



(a) Inception module, naïve version

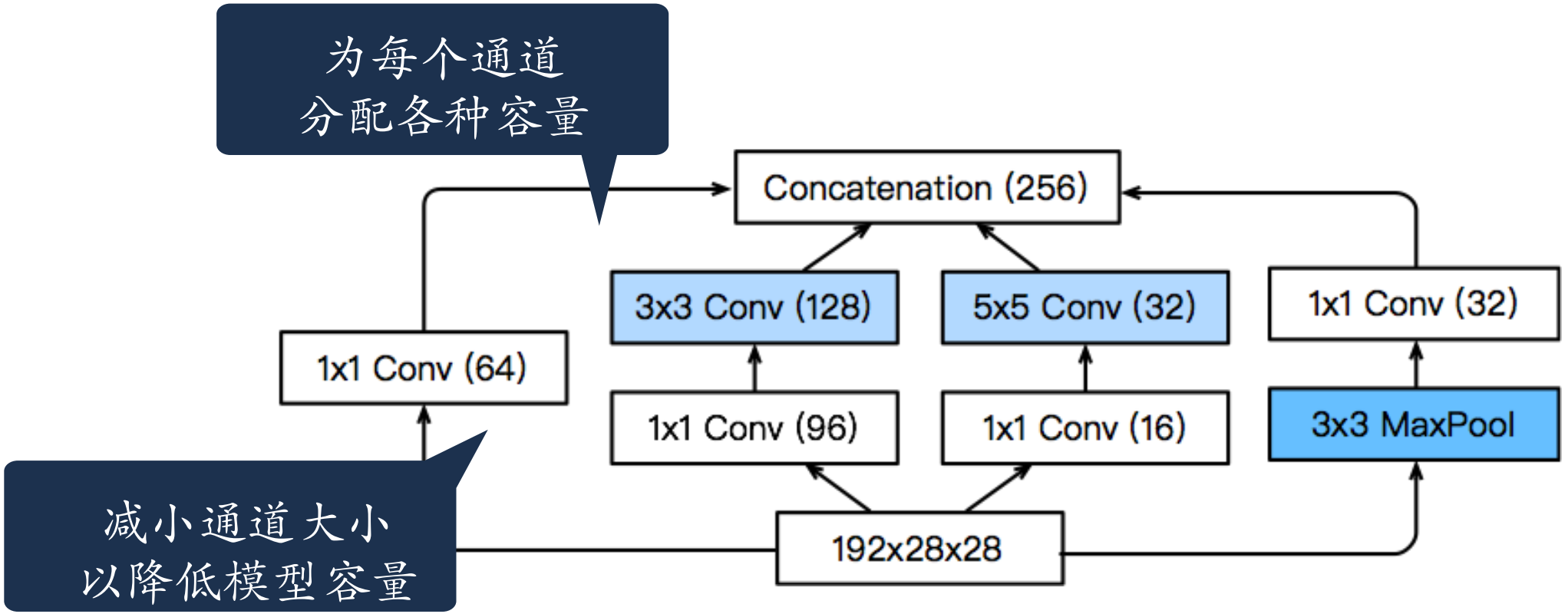
Inspection结构



(b) Inception module with dimension reductions

Inception 块

（第一个初始块）指定的通道大小

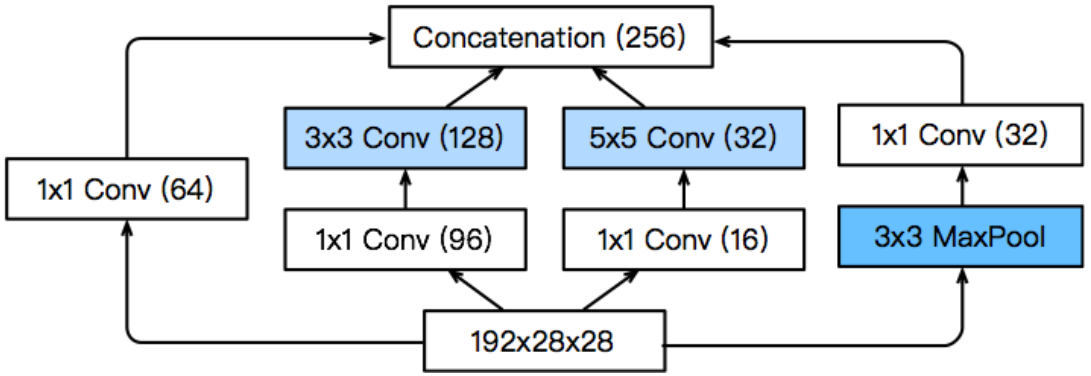


Inception 块

与单个3x3或5x5卷积层相比，初始块具有更少的参数和更低的计算复杂度

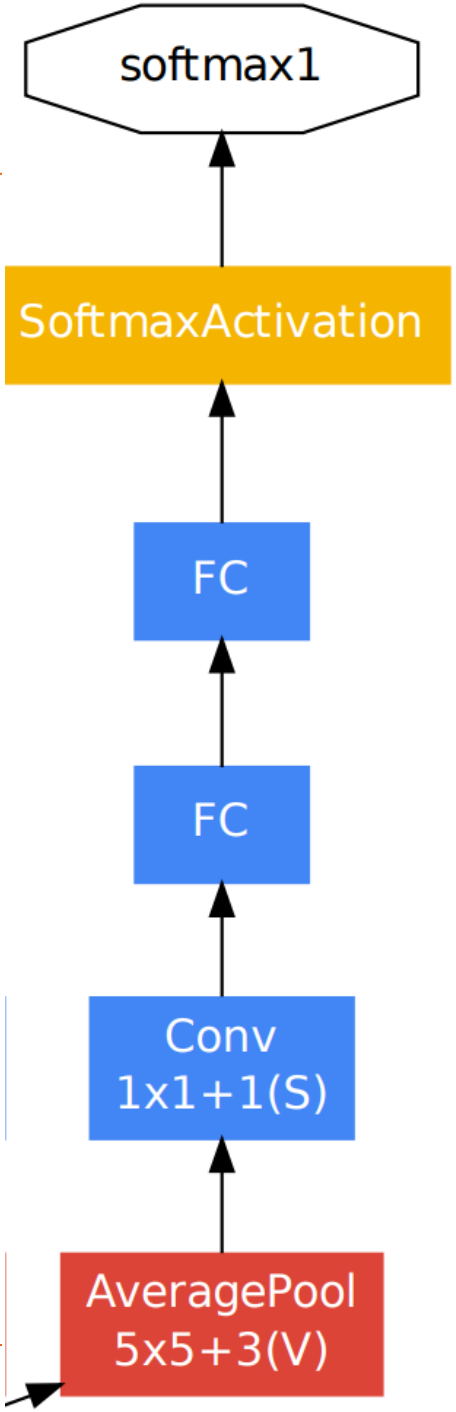
- 不同功能混合（多样的功能类）
- 卷积核计算高效（良好的泛化）

	# 参数	浮点运算 FLOPS
Inception	0.16 M	128 M
3x3 卷积	0.44 M	346 M
5x5 卷积	1.22 M	963 M

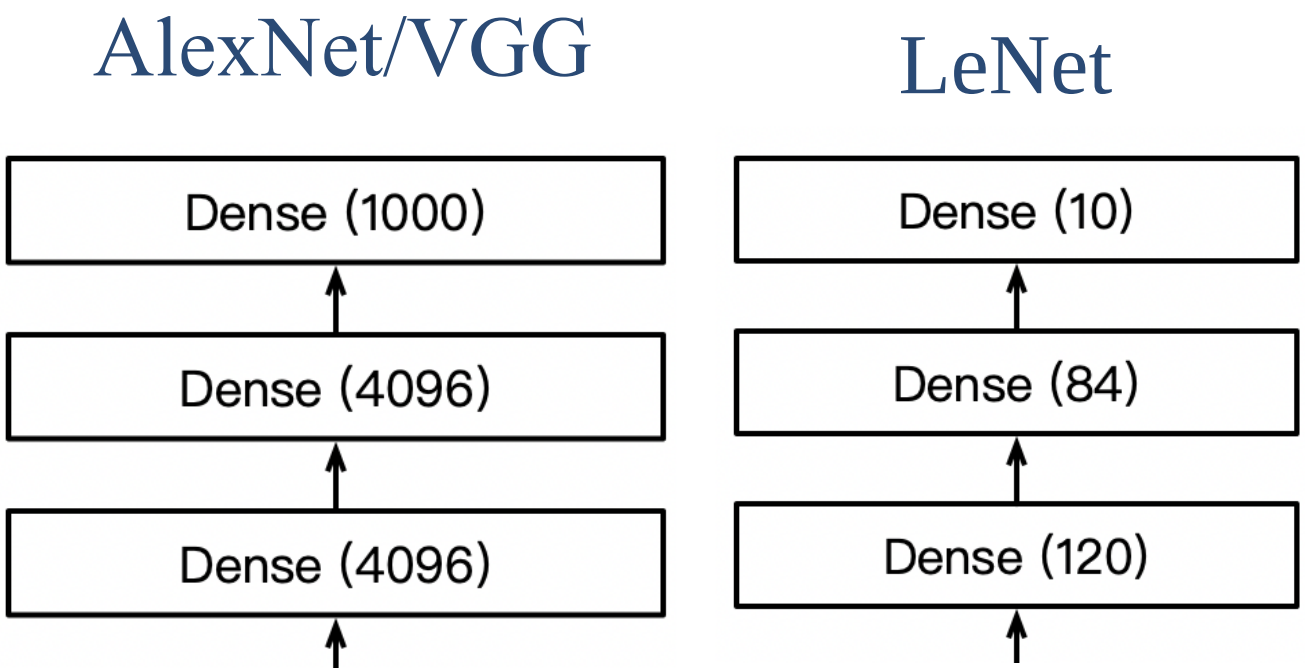


辅助分类器（Auxiliary Classifier）

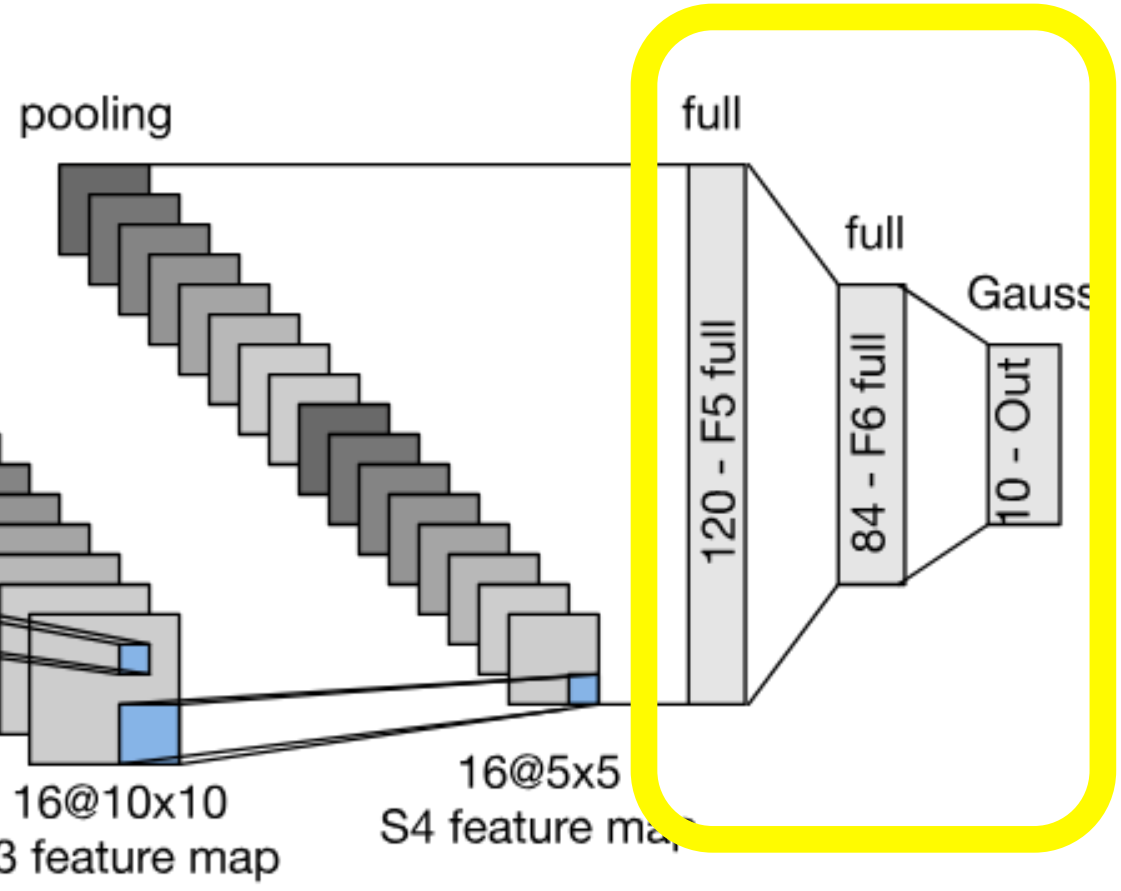
- 辅助分类器是一个附加网络结构，其具体结构如下：
  - 平均池化层：使用5×5的过滤器尺寸和步长为3，对于(4a)阶段产生一个4×4×512的输出，对于(4d)阶段产生一个4×4×528的输出。
  - 1×1卷积层：使用128个过滤器进行维度缩减，并使用修正线性激活函数（Rectified Linear Activation，即ReLU）。
  - 全连接层：包含1024个单元，同样使用修正线性激活函数。
  - 丢弃层（Dropout Layer）：以70%的概率丢弃输出，这是一种防止过拟合的正则化技术。
  - 线性层：使用softmax损失作为分类器，这个分类器预测与主分类器相同的1000个类别，但在推理（inference）时会被移除。



全连接架构


AlexNet/VGG
Dense (1000)
Dense (4096)
Dense (4096)
LeNet
Dense (10)
Dense (84)
Dense (120)

最后一层的诅咒



- 卷积层需要相对较少的参数

$$c_i \times c_o \times k^2$$

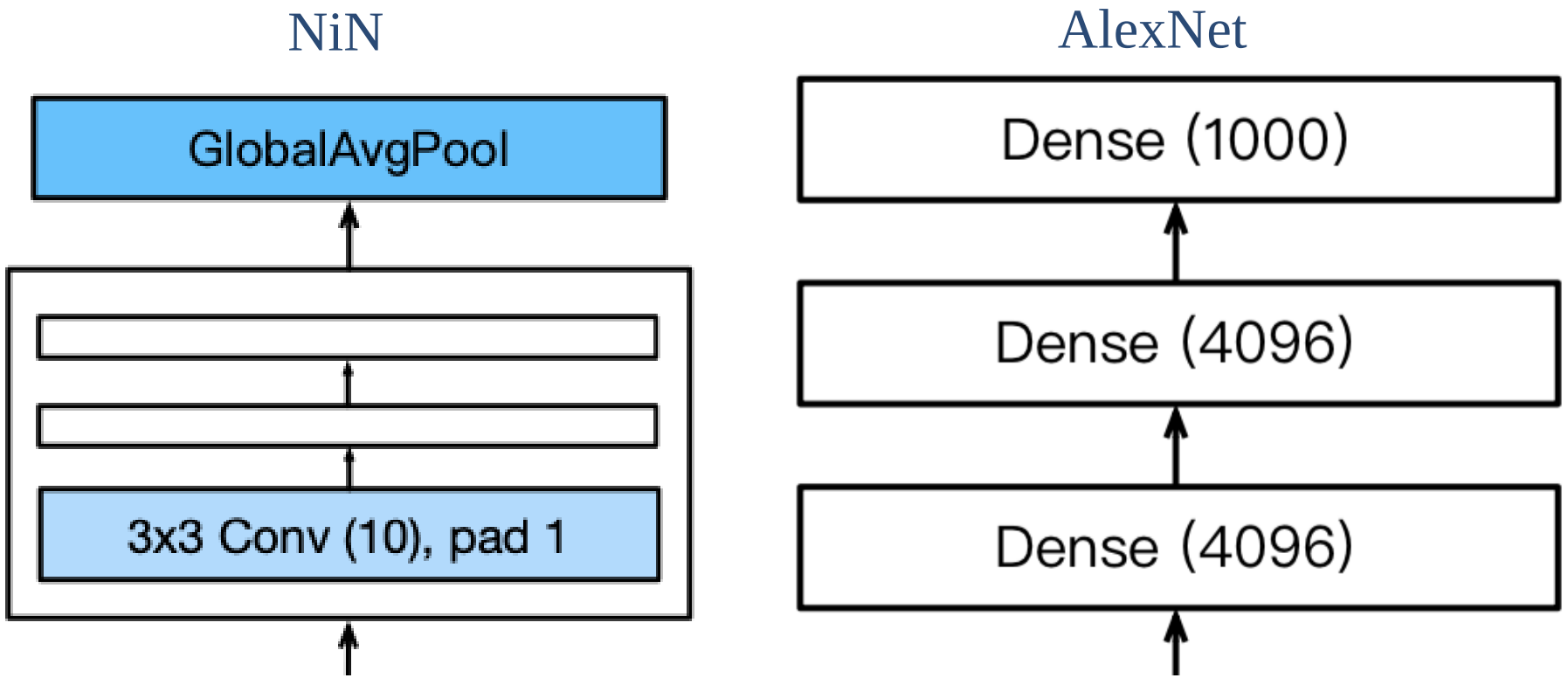
- 最后一层(稠密层)对于n个类的需 要许多参数

$$c \times m_w \times m_h \times n$$

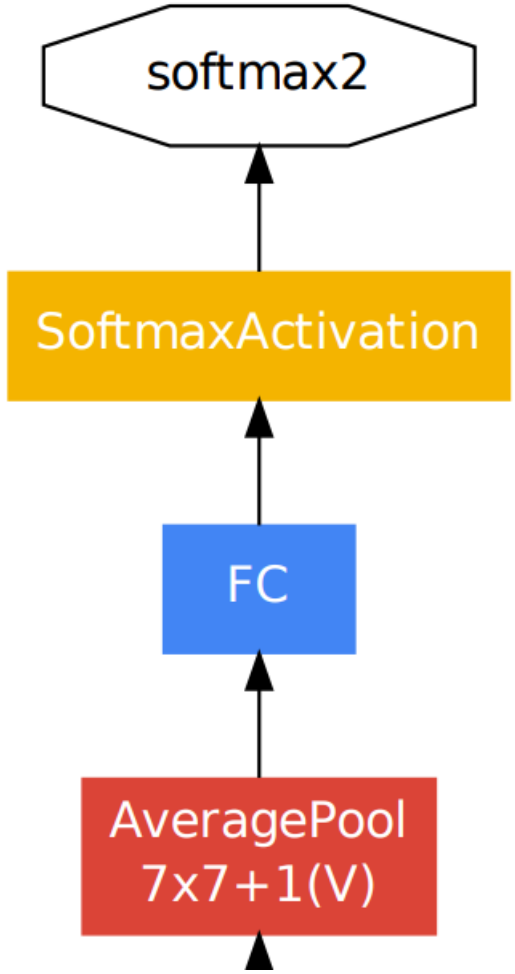
- LeNet 16x5x5x120 = 48k
- AlexNet 256x5x5x4096 = 26M
- VGG 512x7x7x4096 = 102M

NiN 最后一层

- 用 NiN 块替换了 AlexNet 的稠密层
- 输出：全局平均池化层

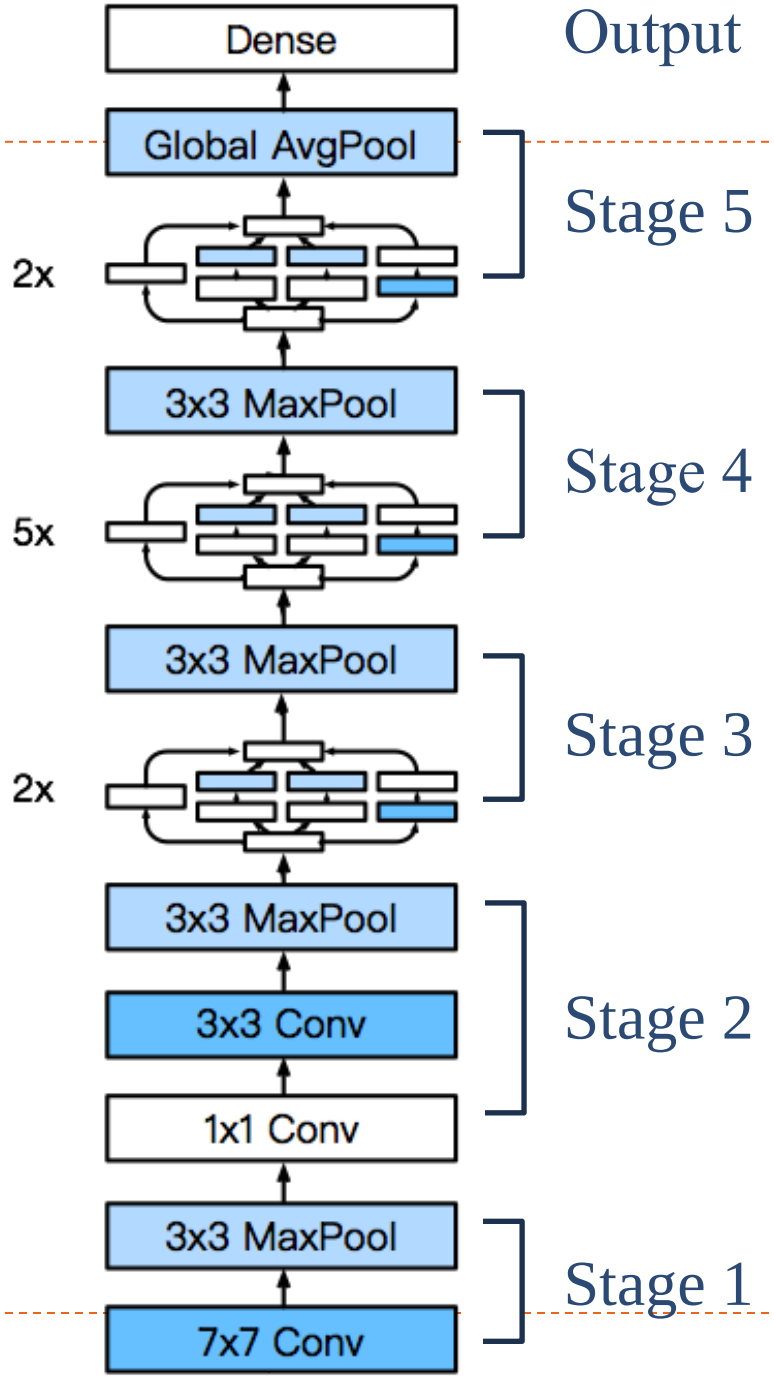


GoogLeNet



GoogLeNet

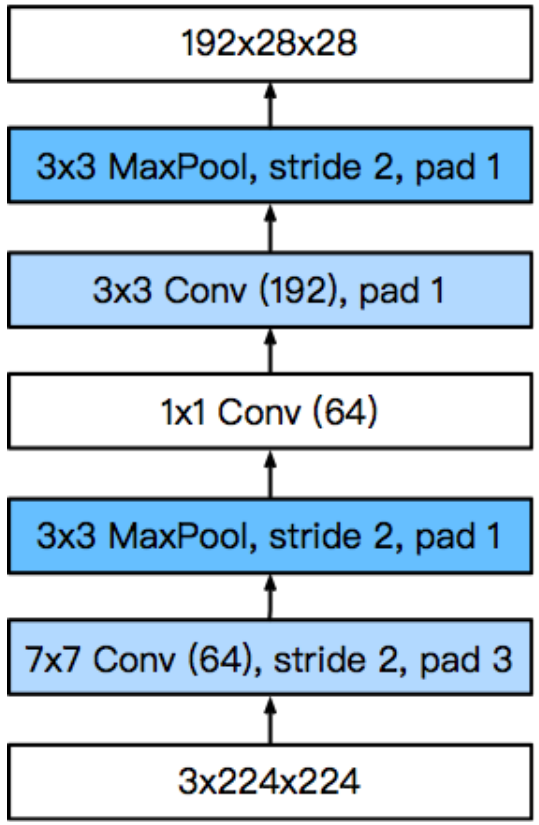
- 5 个阶段
- 9 个 Inception 块



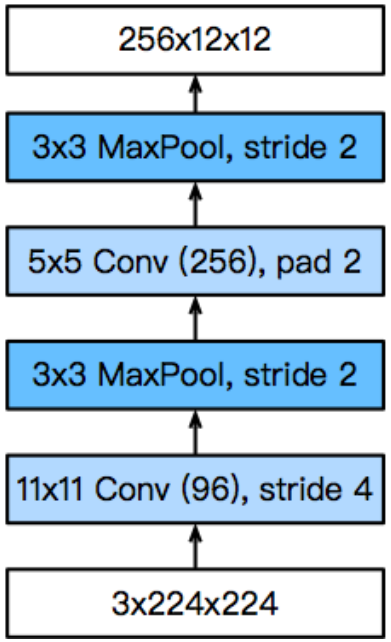
阶段 1 & 2

- 由于更多层:
  - 更小的内核
  - 更小的输出通道

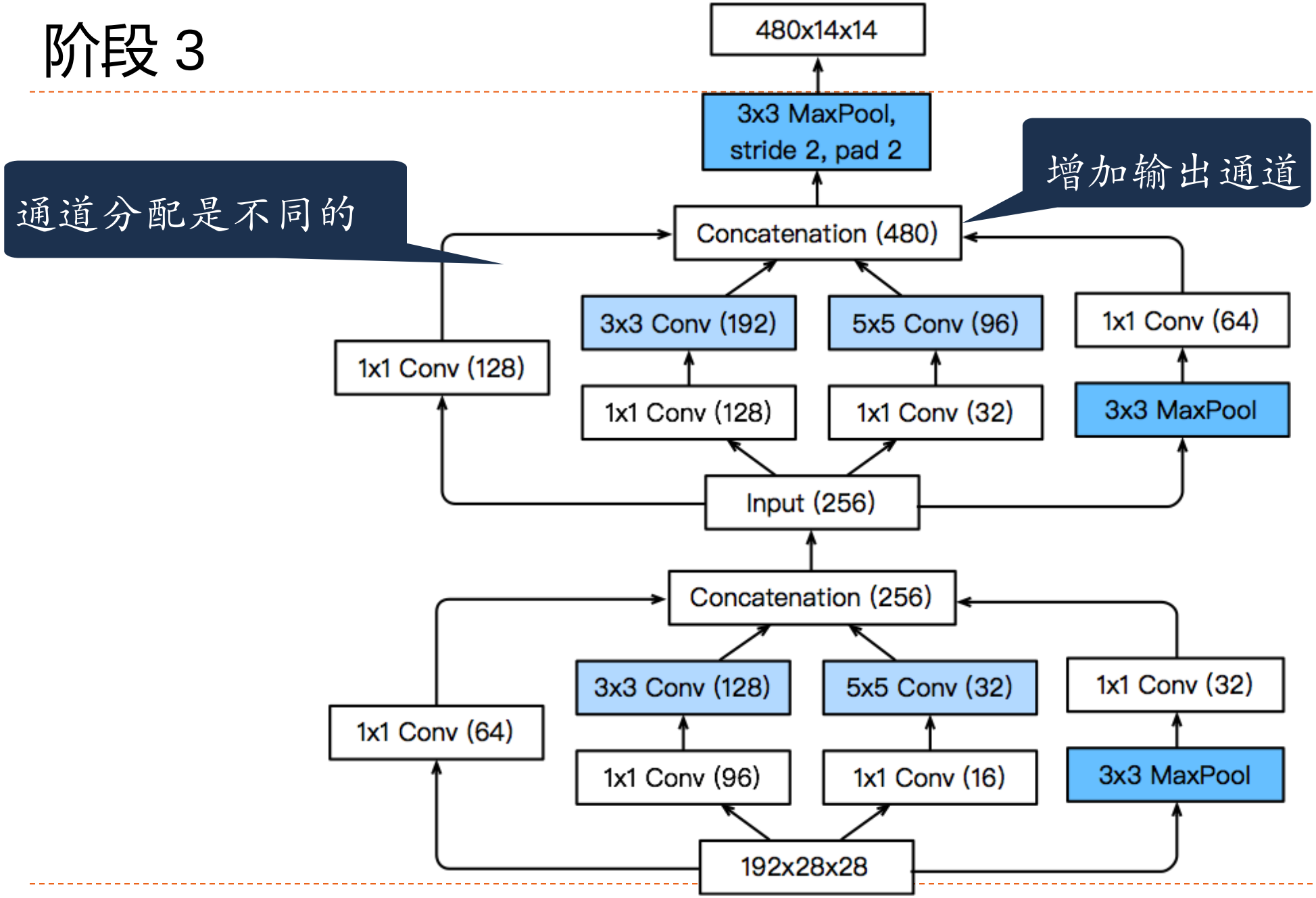
GoogLeNet



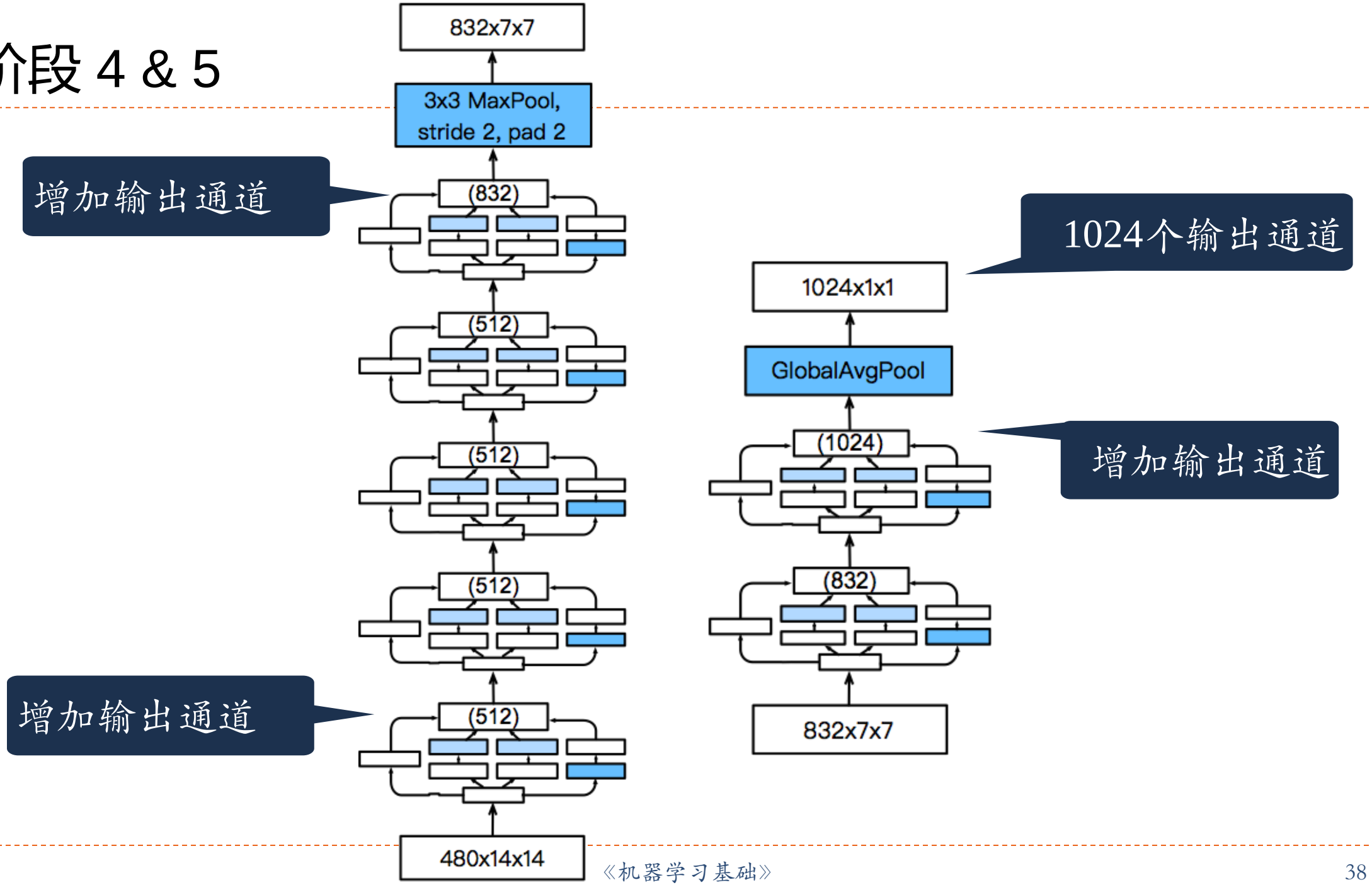
AlexNet



阶段 3


480x14x14
3x3 MaxPool, stride 2, pad 2
通道分配是不同的
增加输出通道
Concatenation (480)
3x3 Conv (192)
5x5 Conv (96)
1x1 Conv (64)
1x1 Conv (128)
1x1 Conv (128)
1x1 Conv (32)
3x3 MaxPool
Input (256)
Concatenation (256)
3x3 Conv (128)
5x5 Conv (32)
1x1 Conv (32)
1x1 Conv (64)
1x1 Conv (96)
1x1 Conv (16)
3x3 MaxPool
192x28x28

阶段 4 & 5

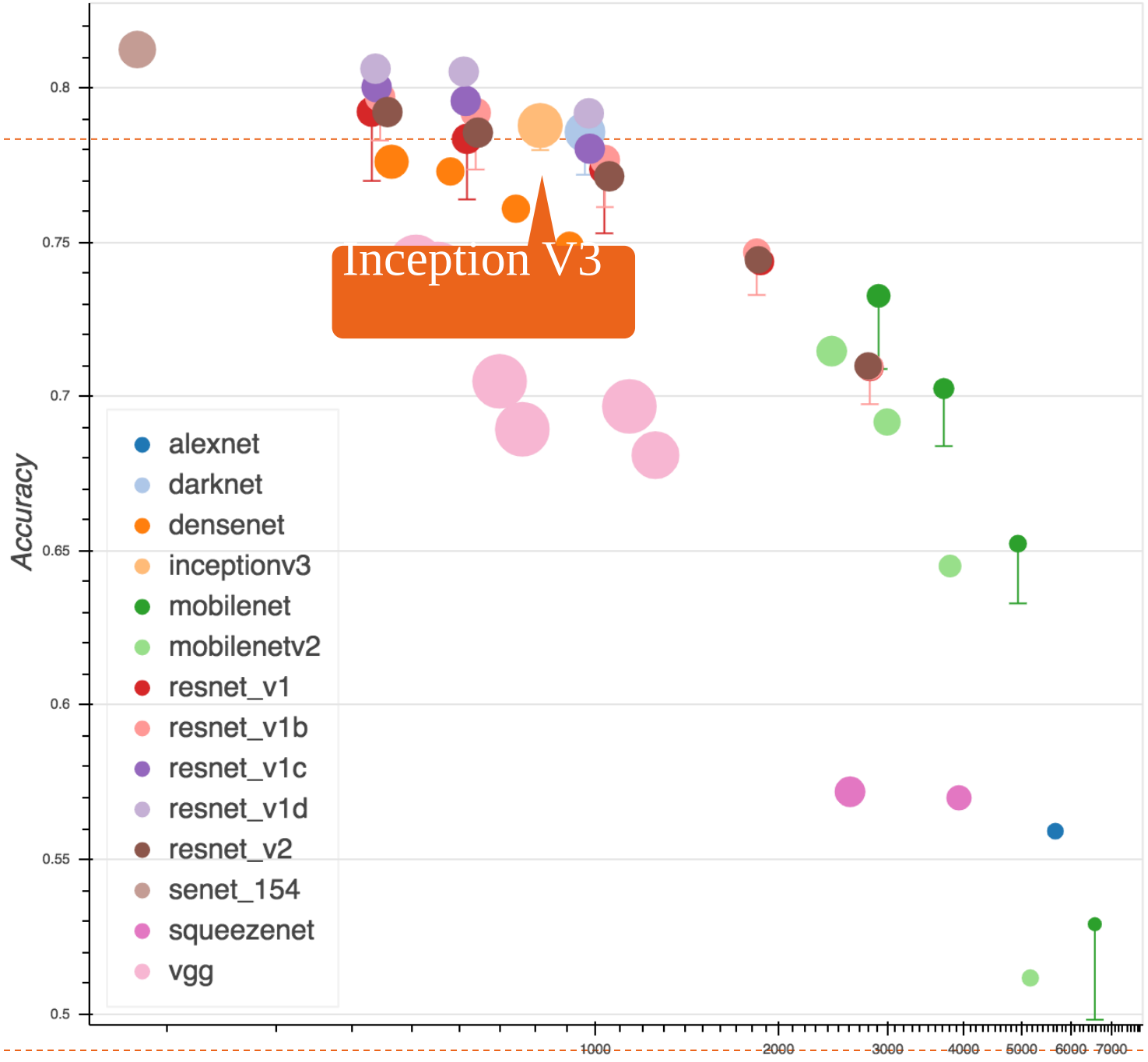

832x7x7
3x3 MaxPool, stride 2, pad 2
增加输出通道
(832)
(512)
(512)
(512)
(512)
增加输出通道
480x14x14
1024x1x1
1024个输出通道
GlobalAvgPool
(1024)
增加输出通道
(832)
832x7x7

GoogLeNet

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								

许多种类的 Inception 网络

- Inception-BN（v2） - 添加批量归一化
- Inception-V3 - 修改了初始块
  - 用多个 3x3 卷积替换 5x5
  - 用 1x7 和 7x1 卷积替换 5x5
  - 用 1x3 和 3x1 卷积替换 3x3
  - 通常用更深的堆
- Inception-V4 - 添加残差块连接



GluonCV 模型“动物园”

https://gluon-cv.mxnet.io/model_zoo/classification.html