



CNN

- Convolutional Neural Network



概要

- **LeNet** (第一个卷积神经网络)
- **AlexNet**
 - 升级版的 LeNet
 - ReLu 激活, 丢弃法
- **VGG**
 - 模块化的 AlexNet
 - 重复的 VGG 块

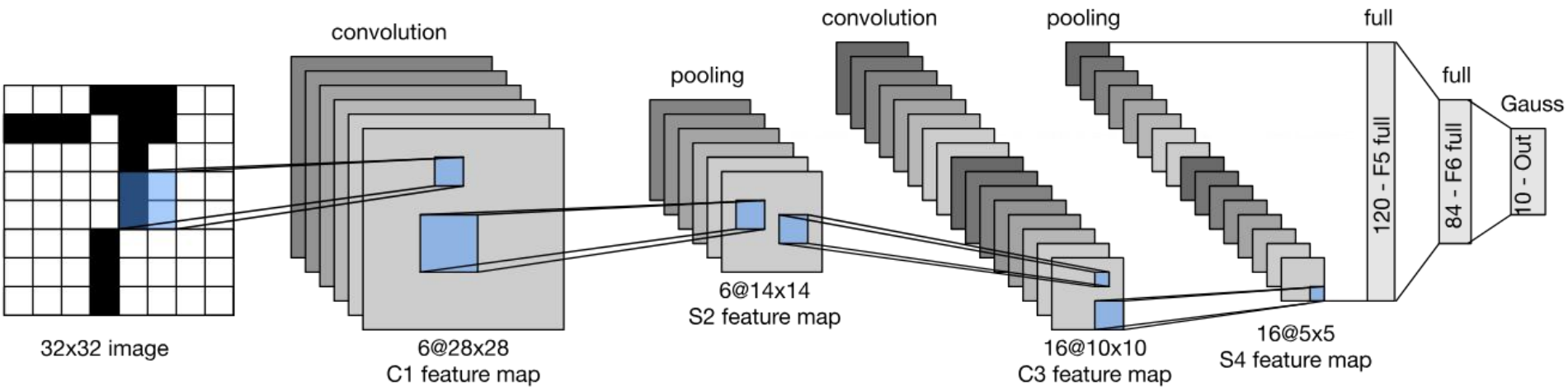
ResNet

泰勒展开式
残差网络

Hub

稠密连接网络
(DenseNet) ,
ShuffleNet, ...

LeNet 架构



手写的数字识别

Philip Marlowe
6381 Hollywood Blvd #615
Los Angeles, CA #90028

PORTLAND OR 970
16 JAN 2019 PM 1 L

Dave Fenwick
Letter, inc
509 Cascade Ave, Suite H
Hood River, OR 97031

97031206080

RENMIN UNIVERSITY OF CHINA
1937
中国人民大学

11725 715

March 10, 1990 16-24/83 1220

PAY TO THE ORDER OF Ballard-Wittman-Robb Chevrolet \$5000.00

Five thousand and 00/100 DOLLARS

WILSHIRE DOHENY OFFICE
WELLS FARGO BANK
NATIONAL ASSOCIATION
9101 WILSHIRE BOULEVARD
BEVERLY HILLS, CALIFORNIA 90211

deposit 1480 Chev. pickup for UGO.

Carroll O'Connor

⑆1⑆1220⑈0024⑆715 0635 111875⑈ ⑈0000500000⑈

MNIST

- 居中和缩放
- 50,000 个训练数据
- 10,000 个测试数据
- 图像大小 28×28
- 10 类





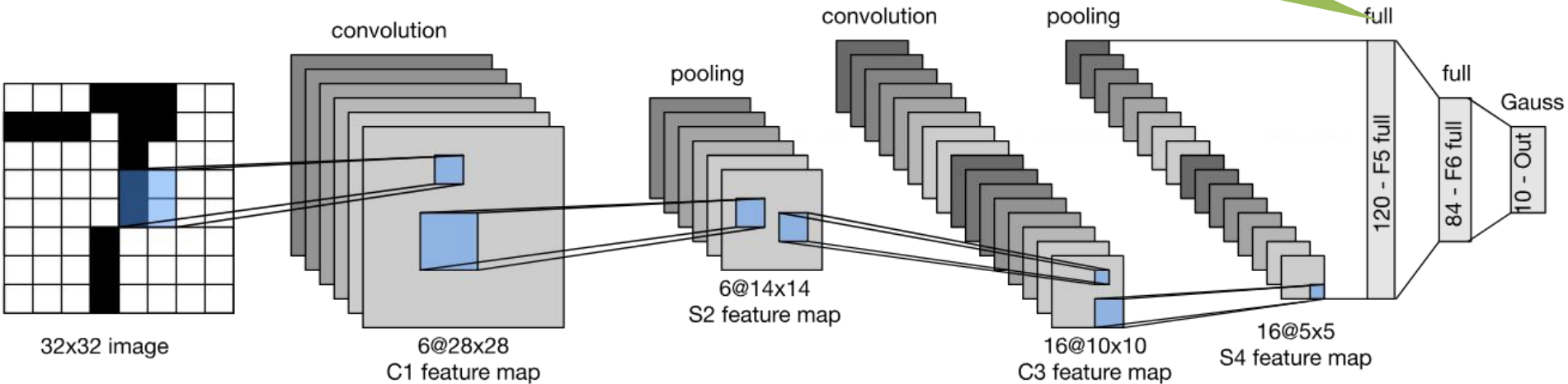
0
103



Y. LeCun, L. Bottou, Y.
Bengio, P. Haffner,
1998

Gradient-based
learning applied to
document
recognition

如果我们有很多输出，那开销就很大了

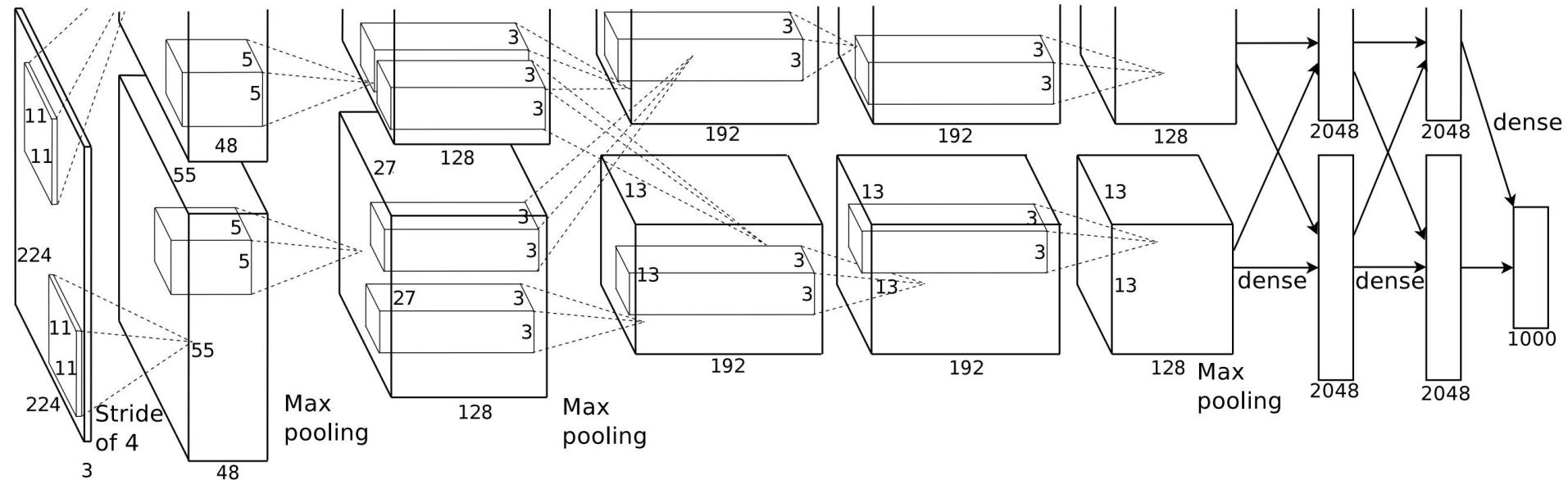




PyTorch中的 LeNet

- *Code...*

AlexNet



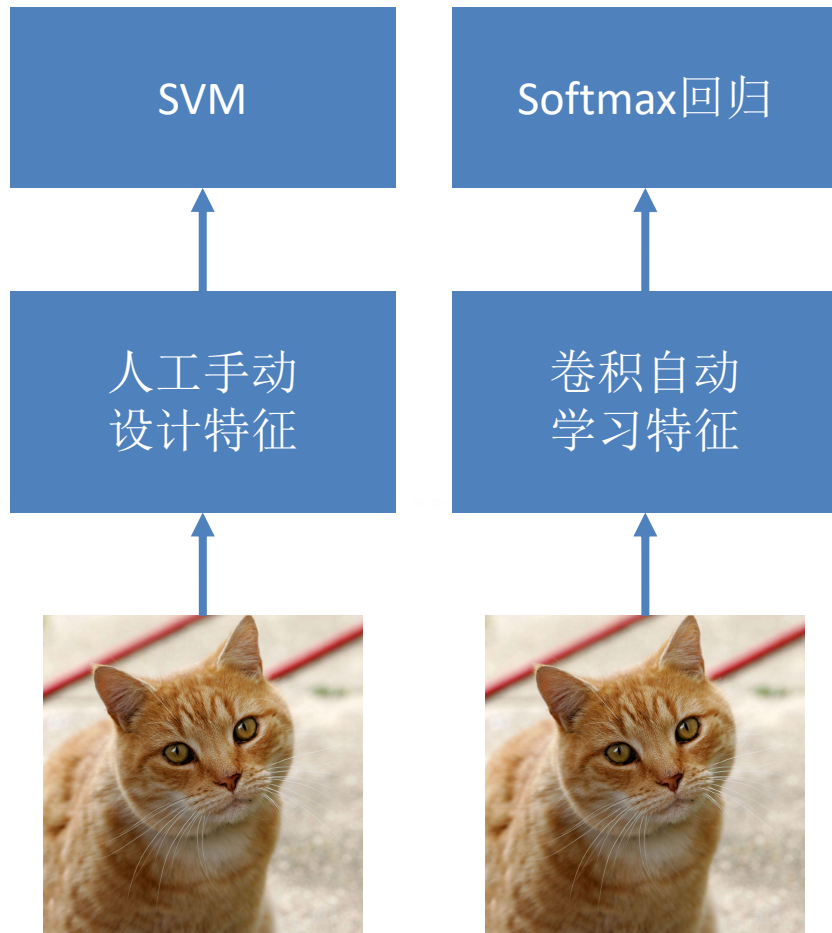
ImageNet 数据集 (2010)



图像	自然物体的彩色图像	手写数字的灰色图像
尺寸	469 x 387	28 x 28
# 样本数	1200万	6万
# 类别数	1,000	10

AlexNet

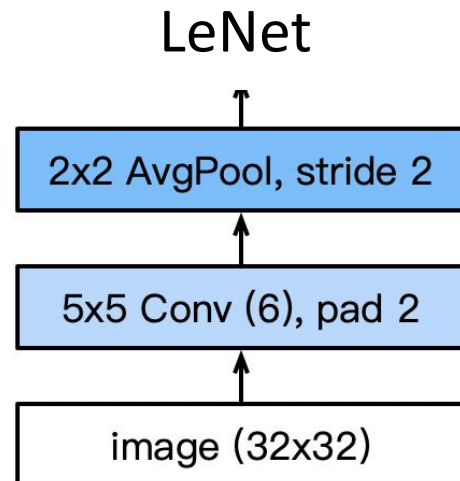
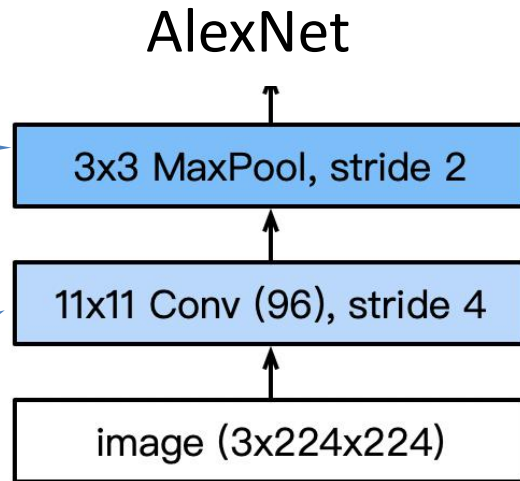
- AlexNet 在 2012 年赢得了ImageNet 竞赛
- 更深更大的 LeNet
- 主要修改
 - 丢弃法 (正则化)
 - ReLu 激活函数 (训练)
 - 最大池化法
- 计算机视觉的范式转变



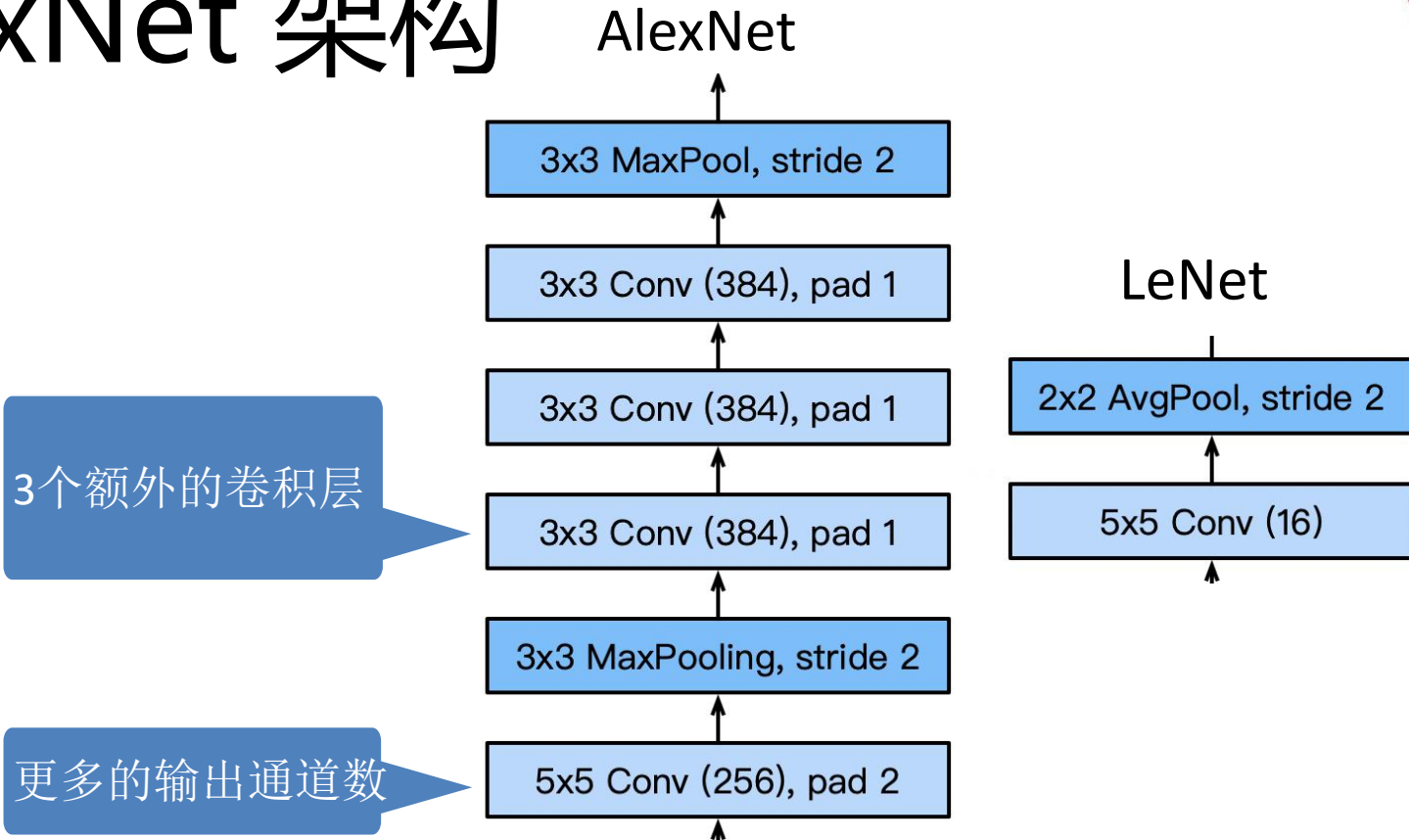
AlexNet 架构

池化窗口更大，更改为最大池化

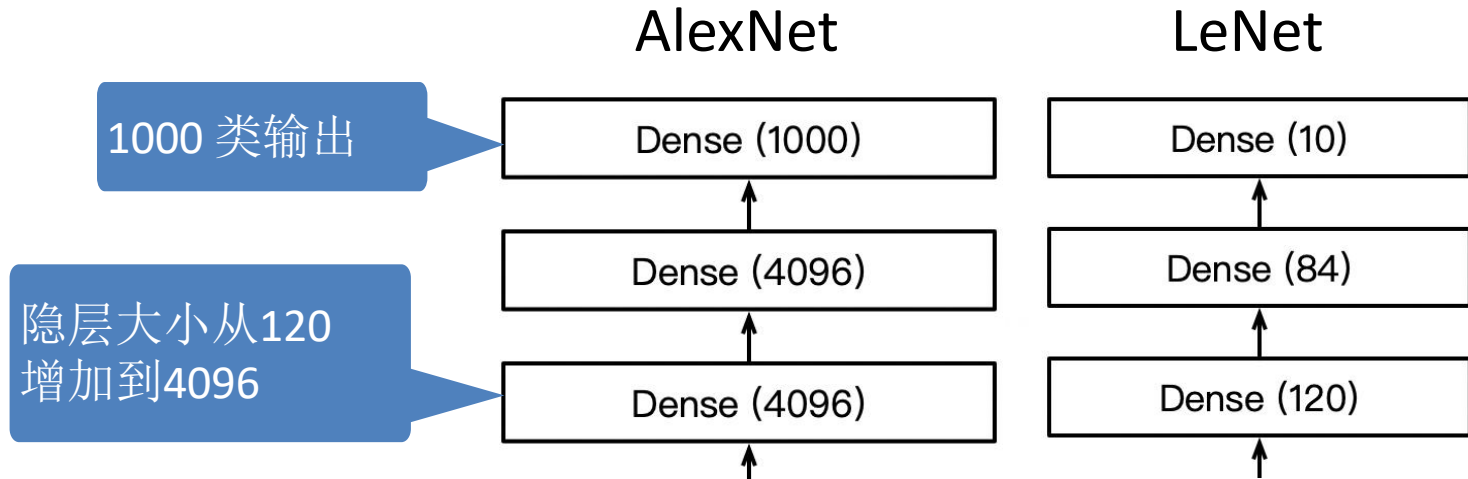
更大的卷积核，更大的步长，
图像尺寸增大，输出通道更多。



AlexNet 架构

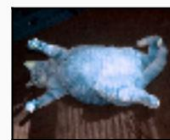
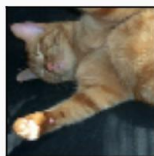
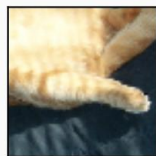
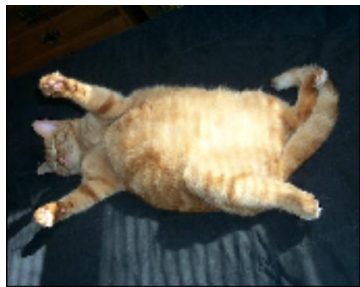


AlexNet 架构



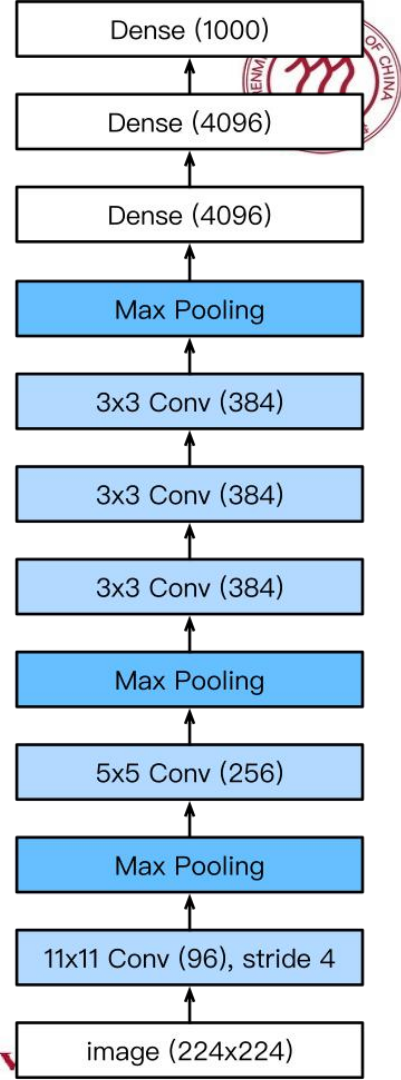
更多技巧

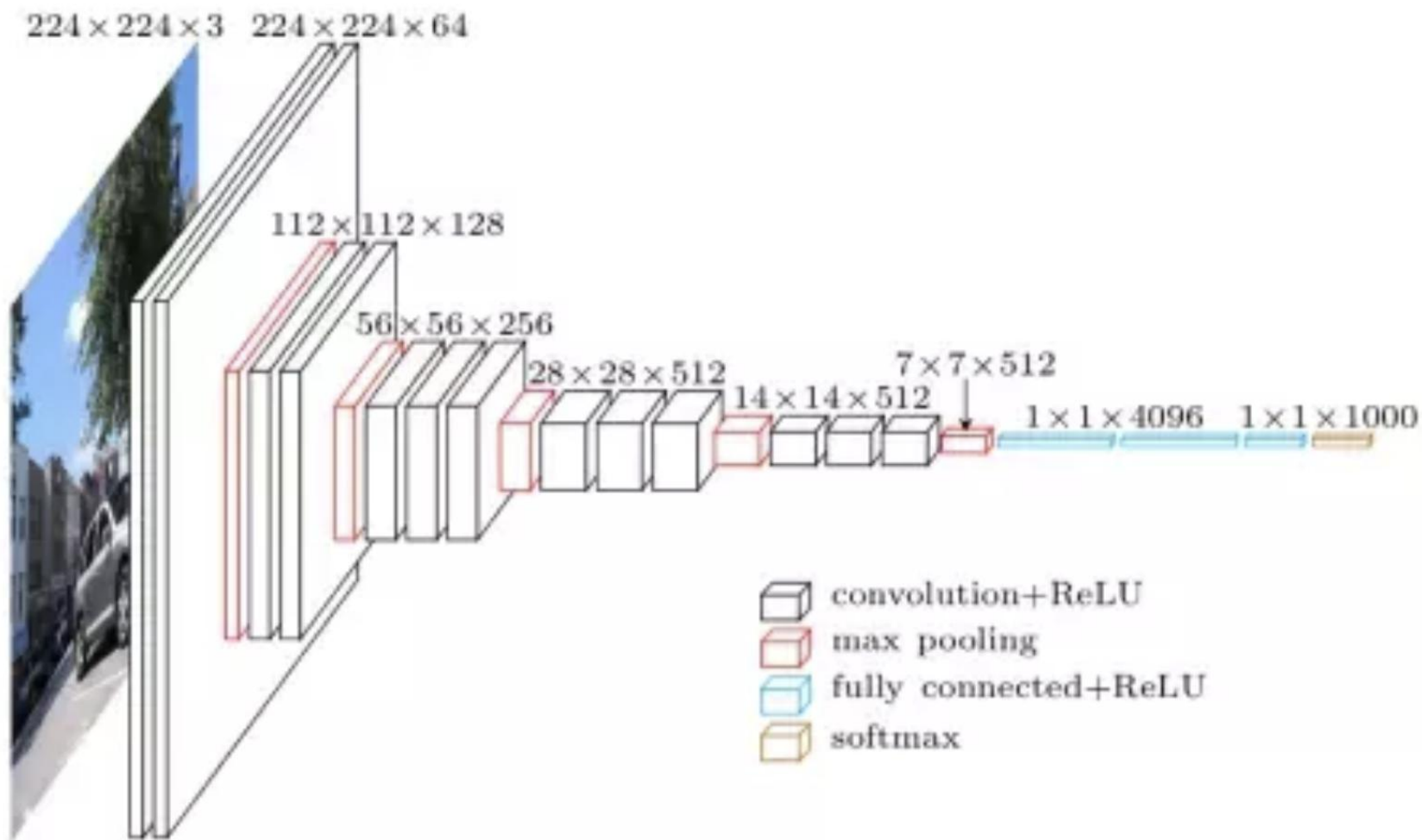
- 将激活函数从 sigmoid 更改为 ReLu (不再梯度消失)
- 在两个隐含层之后应用丢弃法 (更好的稳定性 / 正则化)
- 数据增强



复杂度

	# 参数量		浮点计算量 (FLOP)	
	AlexNet	LeNet	AlexNet	LeNet
卷积层1	35K	150	101M	1.2M
卷积层2	614K	2.4K	415M	2.4M
卷积层3-5	3M		445M	
稠密层1	26M	0.48M	26M	0.48M
稠密层2	16M	0.1M	16M	0.1M
总共	46M	0.6M	1G	4M
倍数增加	11x	1x	250x	1x





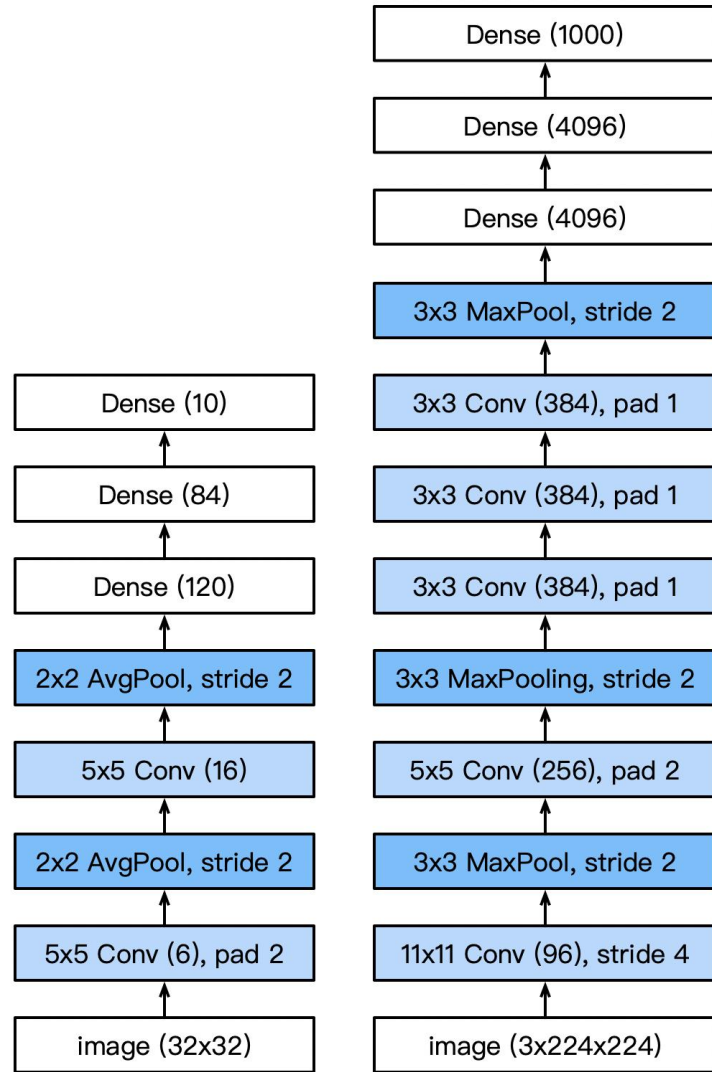


PyTorch中的 AlexNet

- *Code...*

VGG

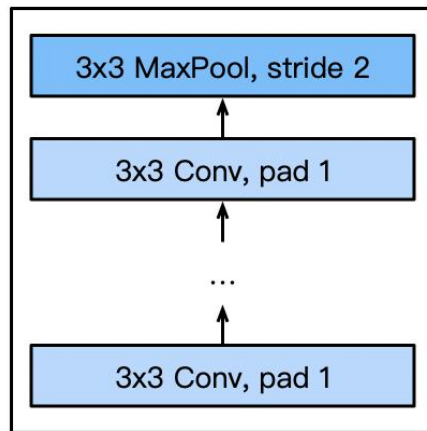
- AlexNet 比 LeNet 更深入更大，以获得更强性能
- 怎么更大更深？
 - 选项
 - 更多稠密层(开销太大)
 - 更多的卷积层
 - 分组成块



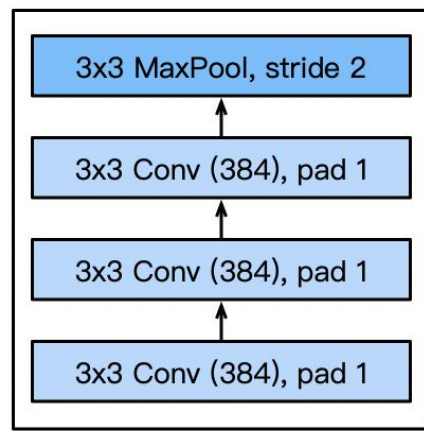
VGG 块

- 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

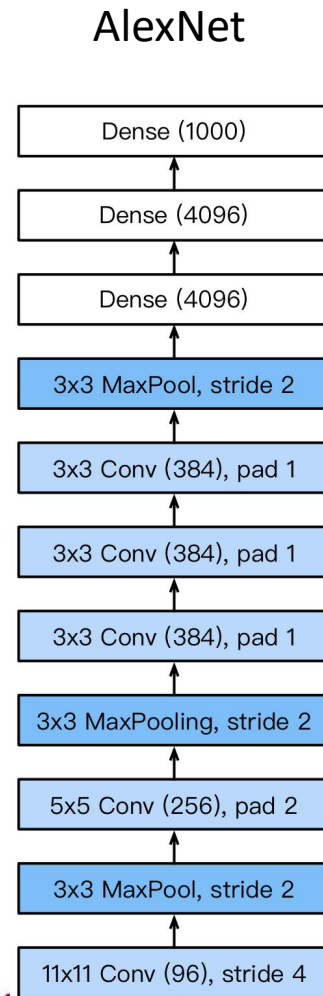
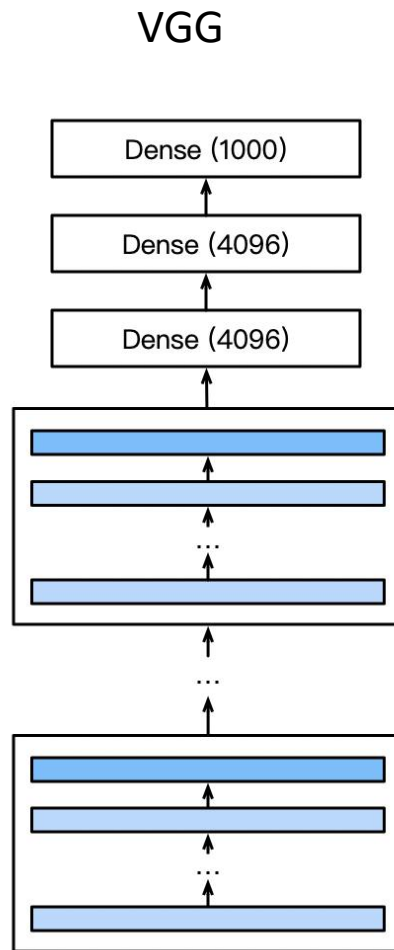


AlexNet 一部分



VGG 结构

- 多个VGG块后加稠密层
- 不同数目的重复VGG块, 可获得不同的架构, 例如 VGG-16, VGG-19,





发展进程

- LeNet (1995)
 - 2 卷积层 + 池化层
 - 2 隐含层
- AlexNet
 - 更大更深的 LeNet
 - ReLu 激活, 丢弃法, 预处理
- VGG
 - 更大更深的 AlexNet (重复的 VGG 块)



VGG 参数

sequential1 output shape: (1, 64, 112, 112)

sequential2 output shape: (1, 128, 56, 56)

sequential3 output shape: (1, 256, 28, 28)

sequential4 output shape: (1, 512, 14, 14)

sequential5 output shape: (1, 512, 7, 7)

dense0 output shape: (1, 4096)

dropout0 output shape: (1, 4096)

dense1 output shape: (1, 4096)

dropout1 output shape: (1, 4096)

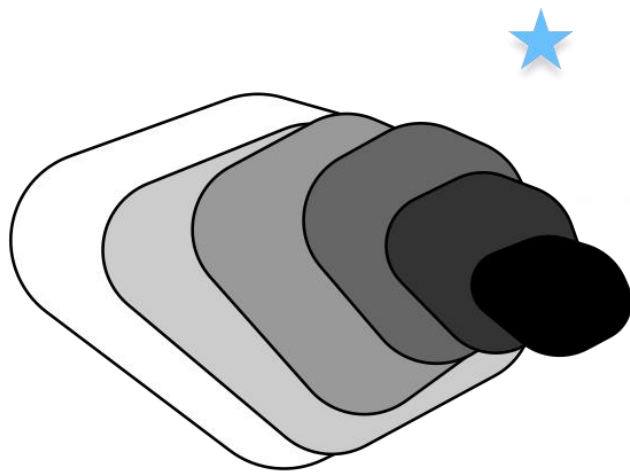
dense2 output shape: (1, 10)



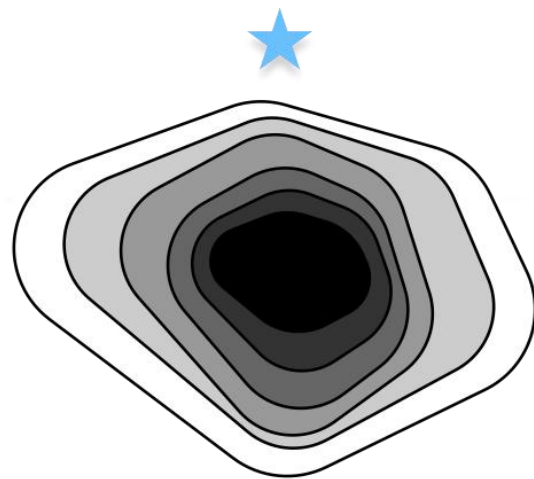
PyTorch中的 VGG

- *Code...*

残差网络 (ResNet)

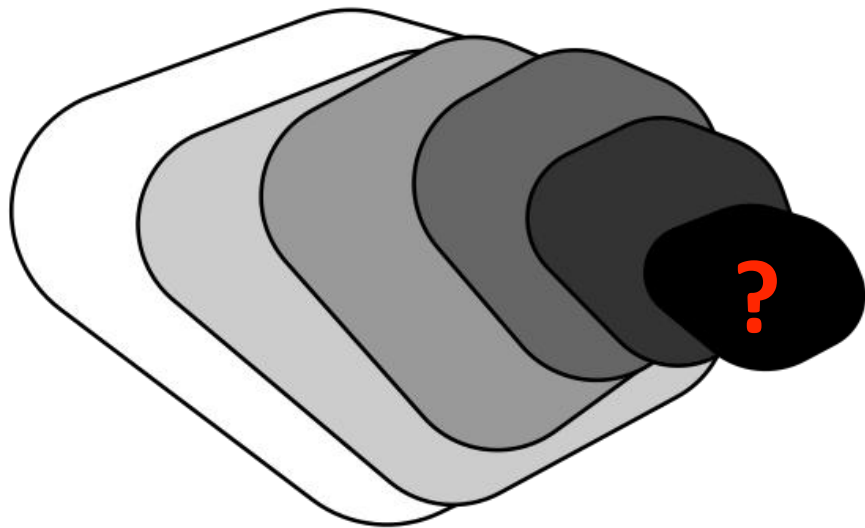


generic function classes

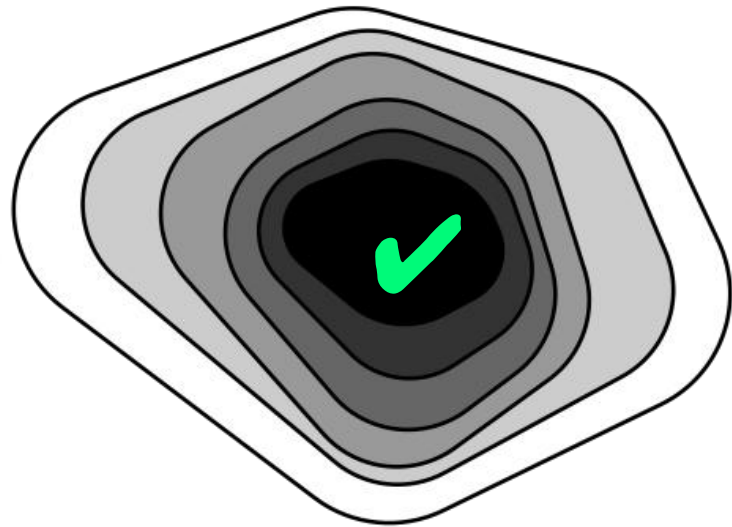


nested function classes

添加层会提高准确性吗？



generic function classes

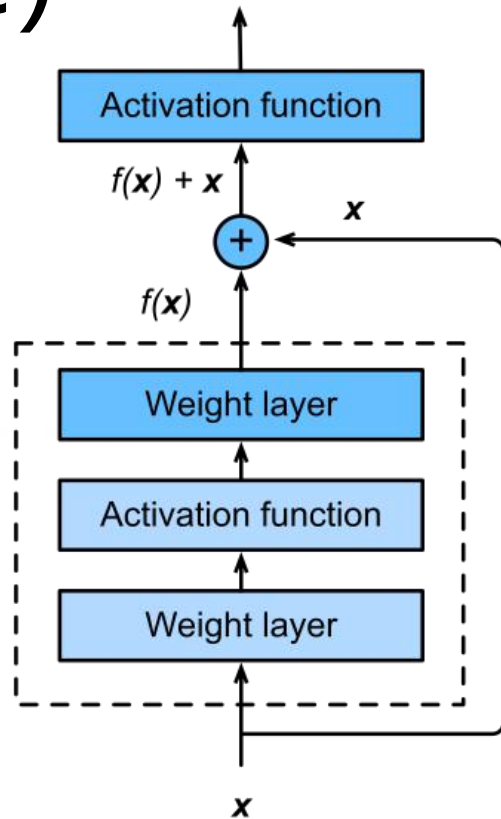
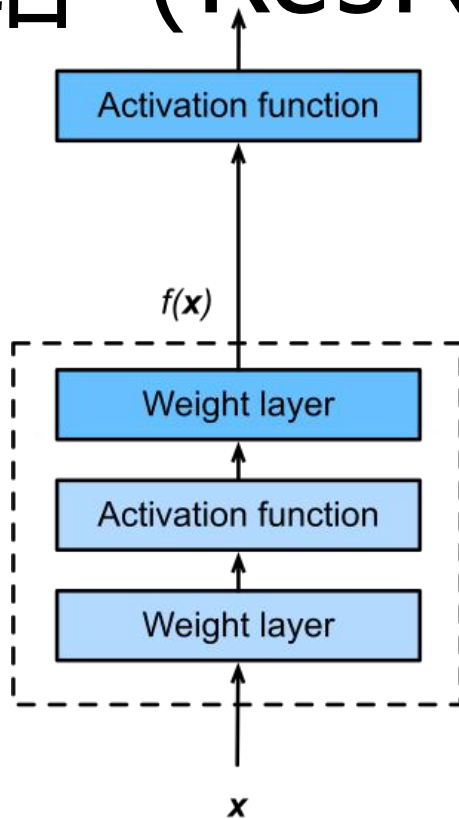


nested function classes

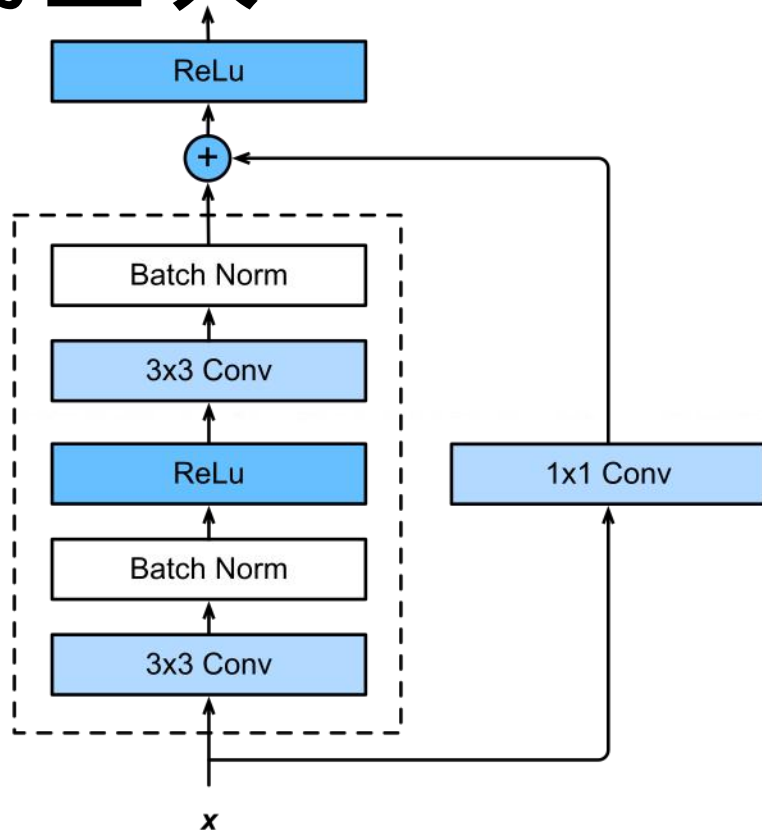
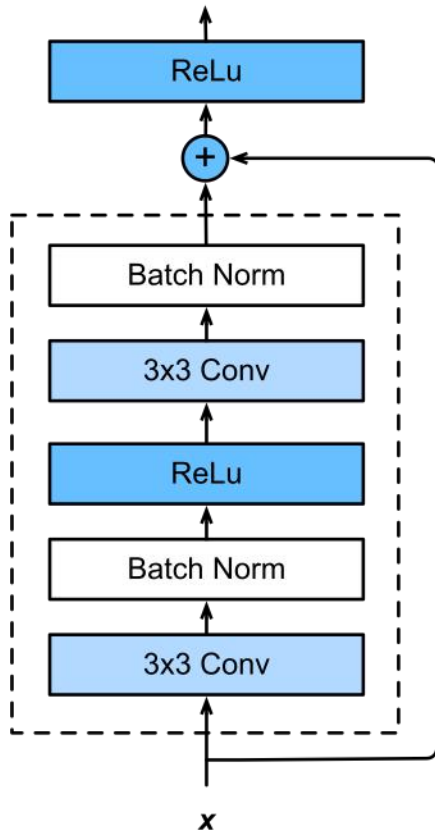
残差网络 (ResNet)

- 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

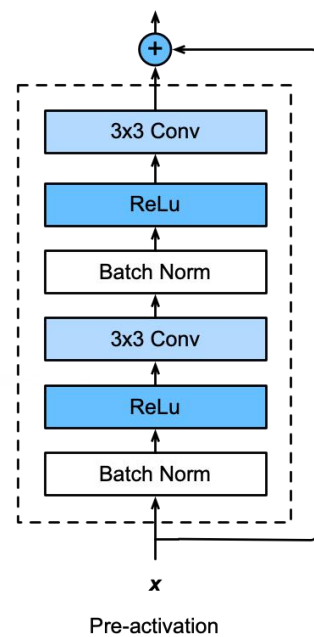
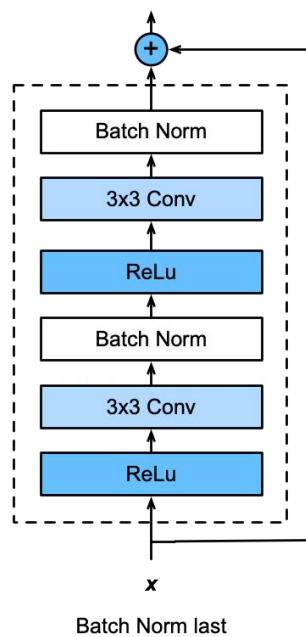
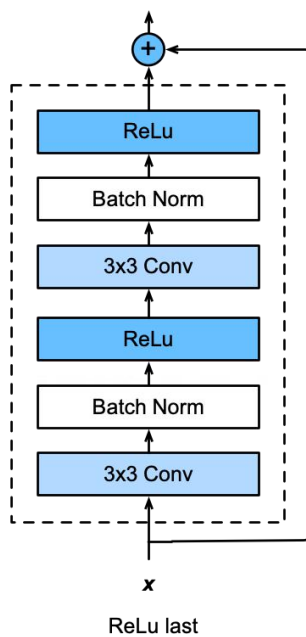
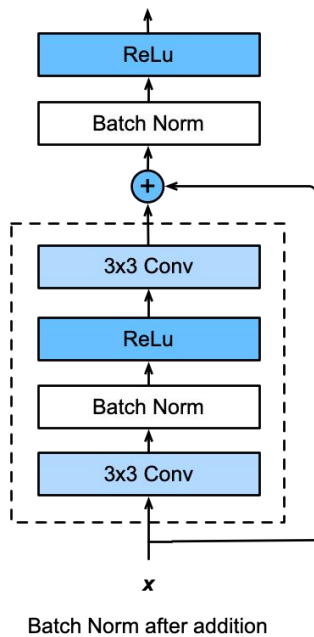
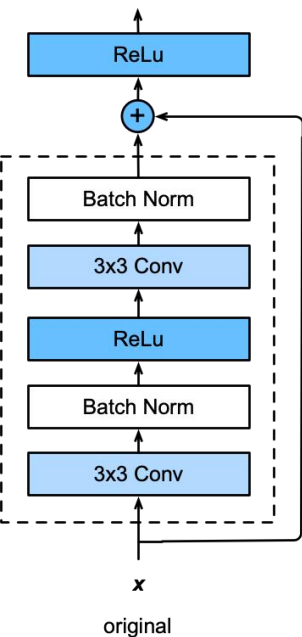
$$f(x) = x + g(x)$$



残差块

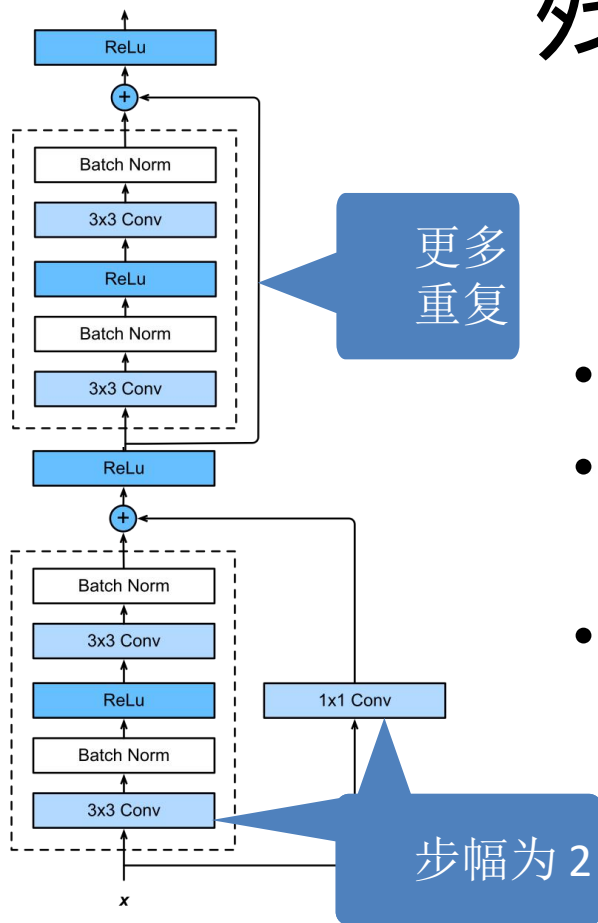


不同的残差块



尝试每一个排列

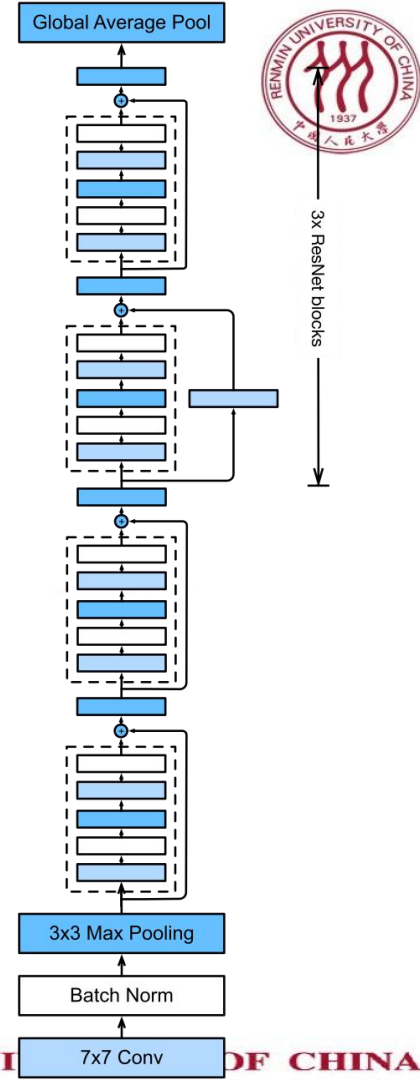
残差块



- 每个模块进行降采样
- 需要调整尺寸时，通过1x1卷积（步幅 = 2）
- 保证主路尺寸和旁路尺寸一致

残差网络 (ResNet)

- 相同块结构，类似VGG块结构
- 残差块连接可增加表现力
- 池化 / 步幅 - 减少维度
- 批量归一化 - 增加深度
-大规模训练.....





PyTorch中的 ResNet

- *Code...*



使用PyTorch Hub

```
model = torch.hub.load('pytorch/vision', 'resnet18', pretrained=True)
```

For Researchers | PyTorch

+

← → ×

pytorch.org/hub/research-models

Get Started

Ecosystem

Mobile

Blog

Tutorials

Docs

Resources

GitHub

Q

AllAudioGenerativeNlpScriptableVision

Q

Inception_v3

Also called GoogleNetv3, a famous ConvNet trained on Imagenet from 2015

MobileNet v2

Efficient networks optimized for speed and memory, with residual blocks

ProxylessNAS

Proxylessly specialize CNN architectures for different hardware platforms.

ResNet

Deep residual networks pre-trained on ImageNet

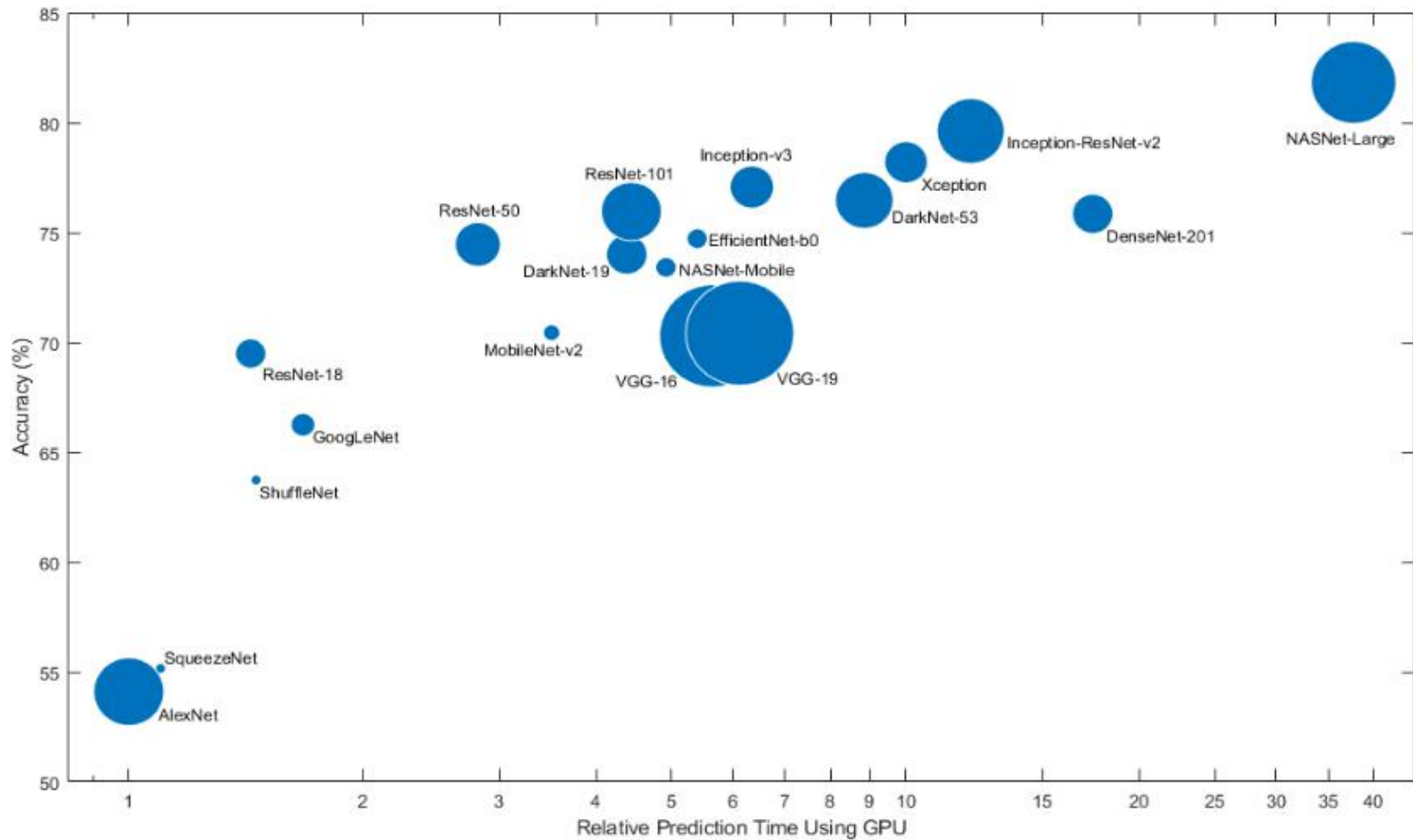
ResNext

Next generation ResNet, more efficient and accurate

ShuffleNet v2

An efficient ConvNet optimized for speed and memory,

https://pytorch.org/hub/pytorch_vision_mobilenet_v2/



pretrained networks properties



Network	Depth	Size	Parameters (Millions)	Image Input Size
squeezenet	18	5.2 MB	1.24	227-by-227
googlenet	22	27 MB	7.0	224-by-224
inceptionv3	48	89 MB	23.9	299-by-299
densenet201	201	77 MB	20.0	224-by-224
mobilenetv2	53	13 MB	3.5	224-by-224
resnet18	18	44 MB	11.7	224-by-224
resnet50	50	96 MB	25.6	224-by-224
resnet101	101	167 MB	44.6	224-by-224
xception	71	85 MB	22.9	299-by-299
inceptionresnetv2	164	209 MB	55.9	299-by-299
shufflenet	50	5.4 MB	1.4	224-by-224
nasnetmobile	*	20 MB	5.3	224-by-224
nasnetlarge	*	332 MB	88.9	331-by-331
darknet19	19	78 MB	20.8	256-by-256
darknet53	53	155 MB	41.6	256-by-256
efficientnetb0	82	20 MB	5.3	224-by-224
alexnet	8	227 MB	61.0	227-by-227
vgg16	16	515 MB	138	224-by-224
vgg19	19	535 MB	144	224-by-224