动手一天学深度学习

1 基础 · 2 卷积网络 · 3 计算 · 4 计算机视觉

深度学习实训营 2019

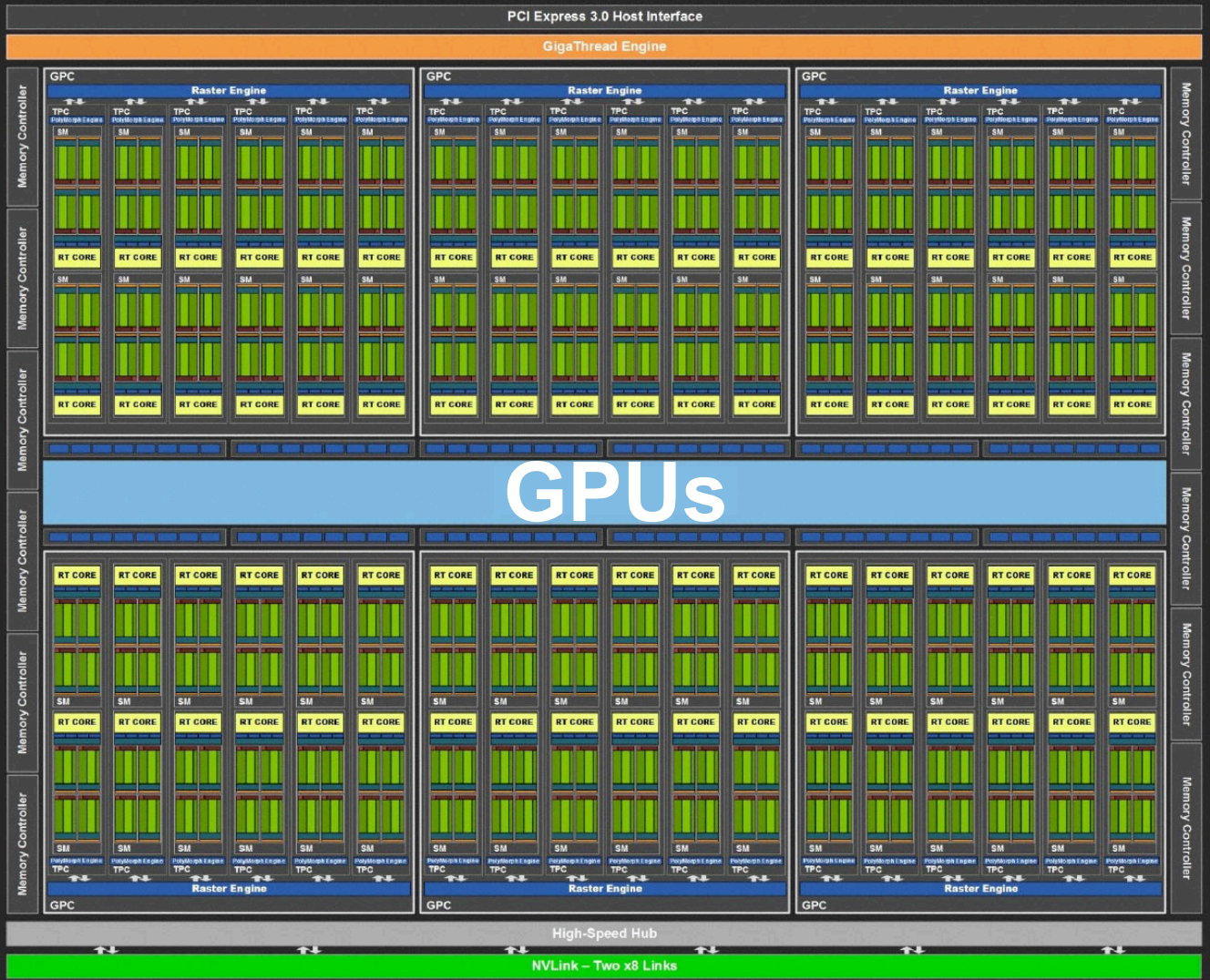
李沐

http://1day-zh.d2l.ai

aws

大纲

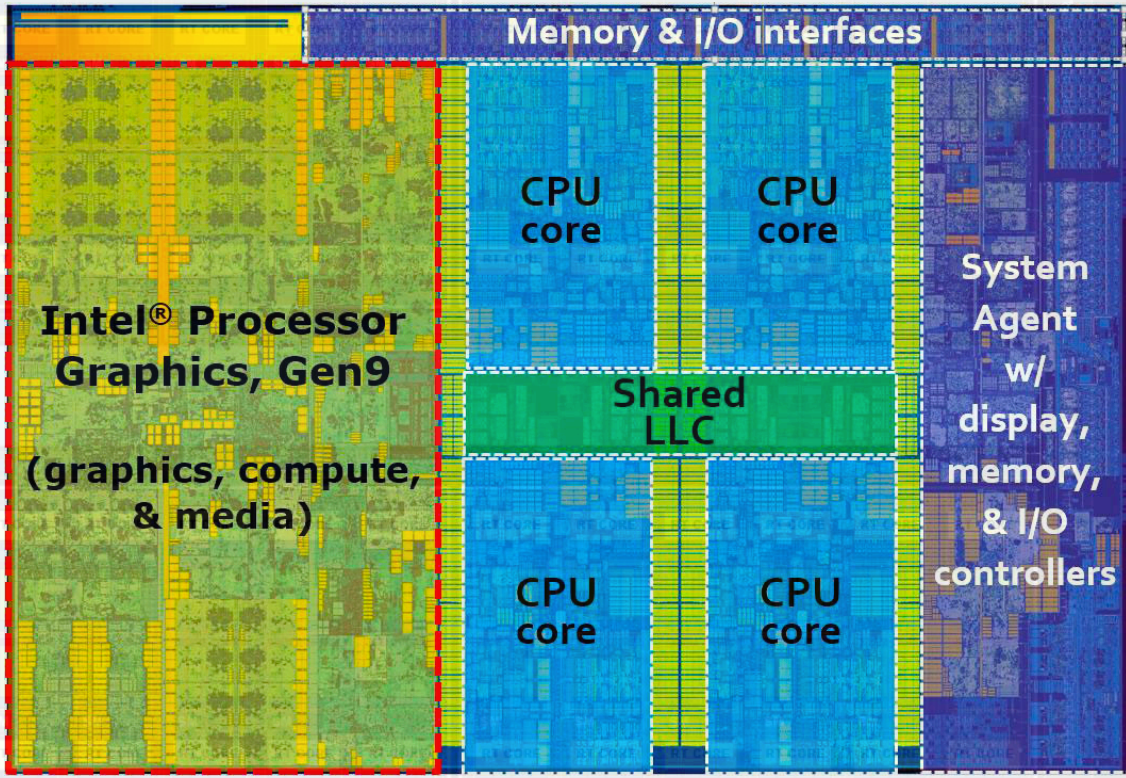
- GPUs
- 卷积，填充，步幅，池化
- 卷积神经网络 (LeNet)
- 深度卷积神经网络 (AlexNet)
- 使用重复元素 (VGG)
- 残差网络 (ResNet)



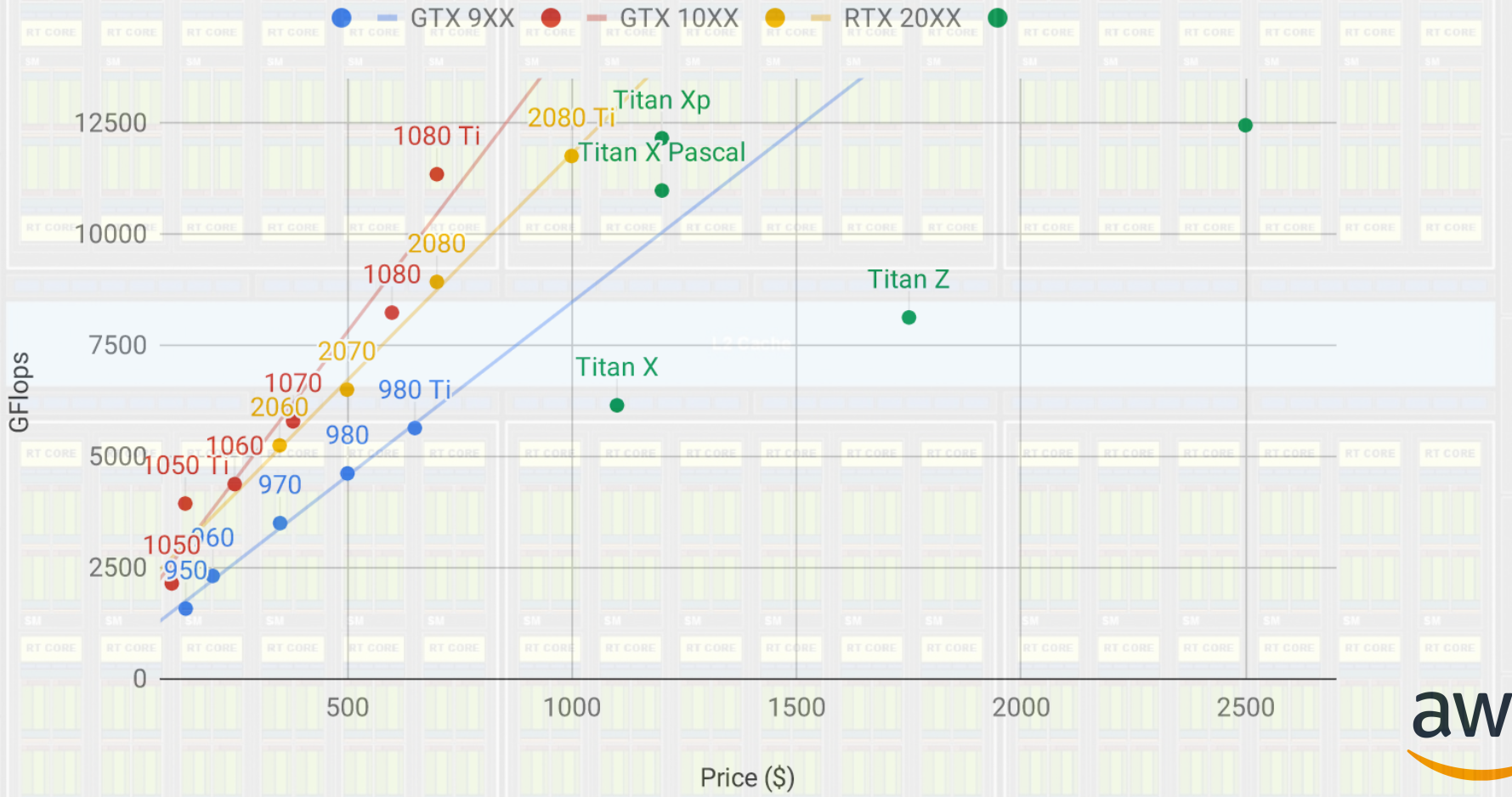
NVIDIA Turing TU102

Intel i7-6700K

- 4 物理核
- 每个核
  - 64KB L1 缓存
  - 256KB L2 缓存
- 共享的 8MB L3 缓存
- 30 GB/s 内存带宽

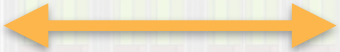


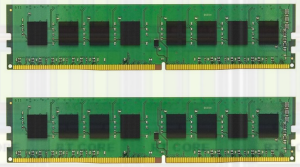
GPU 性能


GTX 9XX
GTX 10XX
RTX 20XX
12500
10000
7500
5000
2500
0
GFlops
500
1000
1500
2000
2500
Price ($)
Titan Xp
2080 Ti
1080 Ti
Titan X Pascal
2080
1080
Titan Z
2070
1070
Titan X
980 Ti
2060
980
1060
1050 Ti
970
1050
960
950

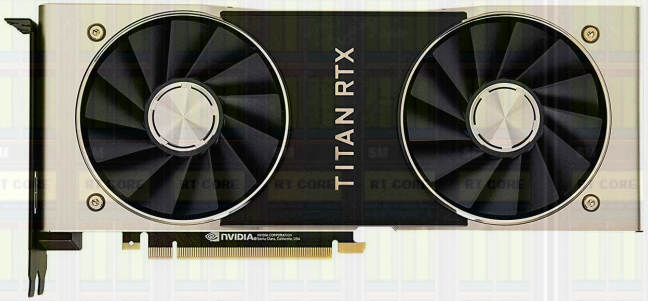
高端游戏、深度学习PC

Intel i7
0.15 TFLOPS





DDR4
32 GB

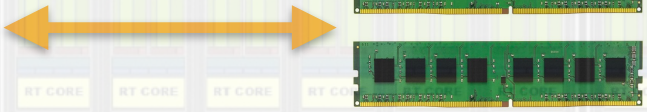


Nvidia Titan RTX
12 TFLOPS (130TF for FP16 TensorCores)
24 GB

http://1day-zh.gluon.ai

Highend Gaming / DeepLearning PC

Intel i7
0.15 TFLOPS



DDR4
32 GB

```
ctx = npx.cpu()
```

```
x.copyto(ctx)
```

Nvidia Titan RTX
12 TFLOPS (130TF for FP16 TensorCores)
24 GB

```
ctx = npx.gpu(0)
```

http://1day-zh.gluon.ai

aws

GPU Notebook

http://1day-zh.gluon.ai


从全连接网络到卷积网络

区分猫和狗图片

- 使用一个还可以的摄像头
- RGB 图片有 36M 元素
- 单100输出的隐藏层模型有 36 亿元素
- 超过了地球上所有的猫和狗 (9亿狗 + 6亿猫)

Dual

12MP

wide-angle and telephoto cameras

http://1day-zh.gluon.ai

aws

倒叙 — 单隐藏层网络

Output layer

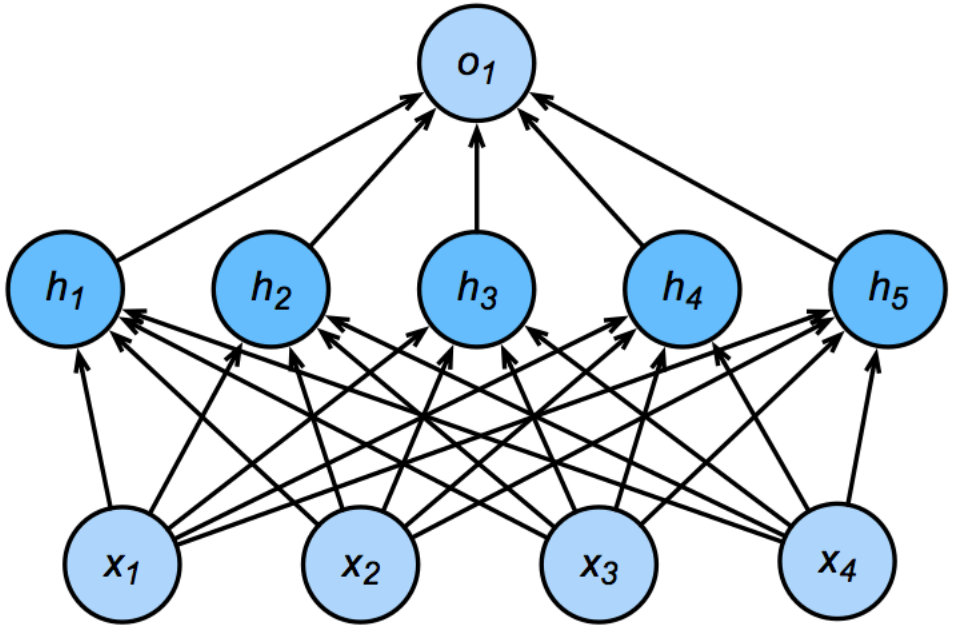
100 输出单元

Hidden layer

3.6B 参数 = 14GB

36M 特征

Input layer



$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

http://1day-zh.gluon.ai

aws

Waldo 在哪里?

两个原则

- 变化无关
- 本地性

http://1day-zh.gluon.ai

重新考虑全连接层

- 将输入和输出变形成矩阵 (宽，高)
- 将权重变形成 4-D 张量

$$h_{i,j} = \sum_{k,l} w_{i,j,k,l} x_{k,l} = \sum_{a,b} v_{i,j,a,b} x_{i+a,j+b}$$

这里 W 是 V 的元素重新排列：

$$v_{i,j,a,b} = w_{i,j,i+a,j+b}$$

想法1：变化无关

$$h_{i,j} = \sum_{a,b} v_{i,j,a,b} x_{i+a,j+b}$$

- 在 x 中的偏移会导致 h 中的偏移和数值变化
- v 不应该跟 (i,j) 相关联。修改 $v_{i,j,a,b} = v_{a,b}$

$$h_{i,j} = \sum_{a,b} v_{a,b} x_{i+a,j+b}$$

这是 2-D ~~卷积~~

互相关

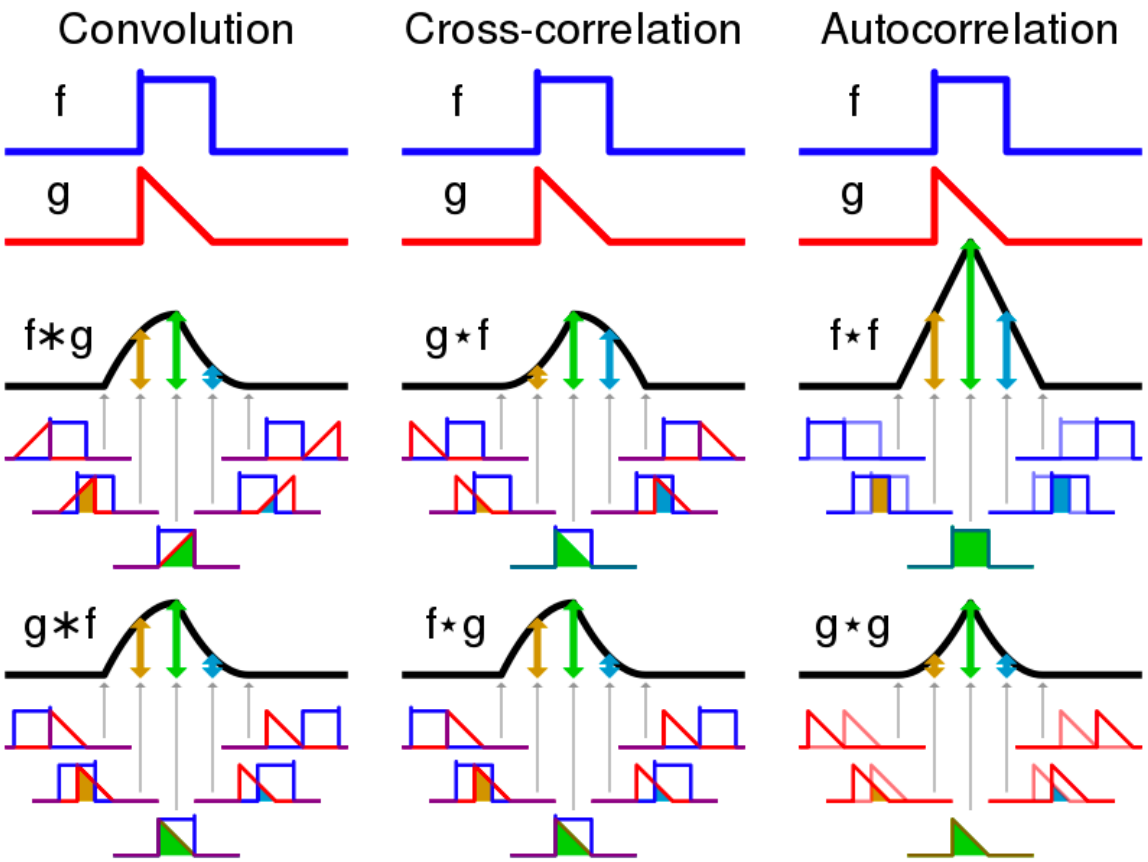
想法 2 — 本地性

$$h_{i,j} = \sum_{a,b} v_{a,b} x_{i+a,j+b}$$

- 我们不应该看离 x(i,j) 太远的来计算 h(i,j)
- 在区间 $|a|, |b| > \Delta$ 外的权重置零 $v_{a,b} = 0$

$$h_{i,j} = \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} v_{a,b} x_{i+a,j+b}$$

卷积层

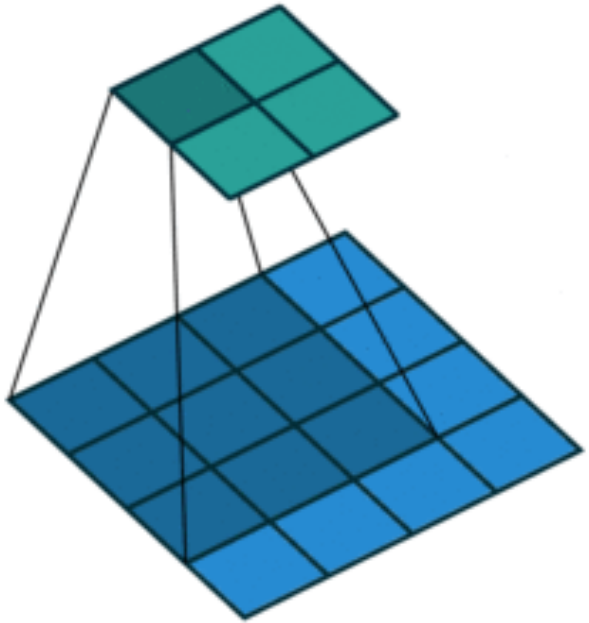


http://1day-zh.gluon.ai

2-D 互相关

Input			*	Kernel		=	Output	
0	1	2		0	1		19	25
3	4	5		2	3		37	43
6	7	8						

$$0\times0+1\times1+3\times2+4\times3=19,$$
$$1\times0+2\times1+4\times2+5\times3=25,$$
$$3\times0+4\times1+6\times2+7\times3=37,$$
$$4\times0+5\times1+7\times2+8\times3=43.$$

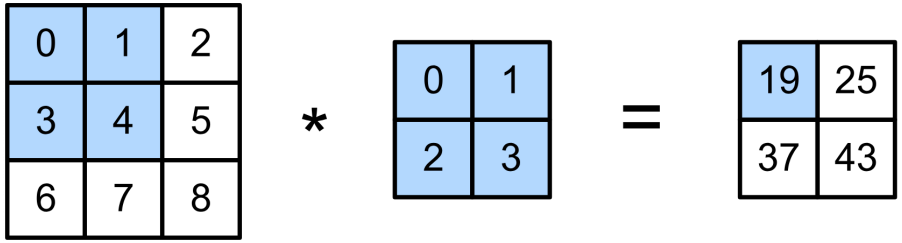


(vdumoulin@ Github)

http://1day-zh.gluon.ai

2-D 卷积层

- $\mathbf{X}: n_h \times n_w$：输入矩阵
- $\mathbf{W}: k_h \times k_w$：核矩阵
- b: scalar bias
- $\mathbf{Y}: (n_h - k_h + 1) \times (n_w - k_w + 1)$：输出矩阵



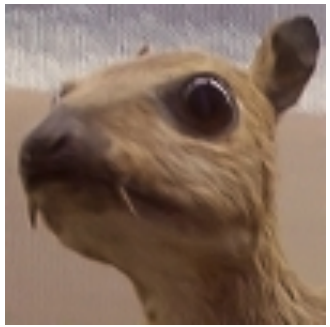
$$\mathbf{Y} = \mathbf{X} \star \mathbf{W} + b$$

- **W** 和 *b* 是可以学习的参数

例子

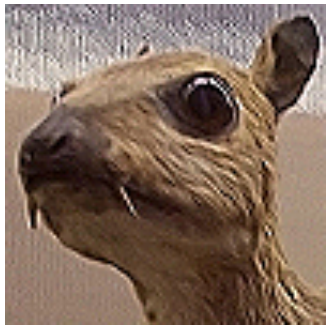
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

边缘检测



(wikipedia)

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



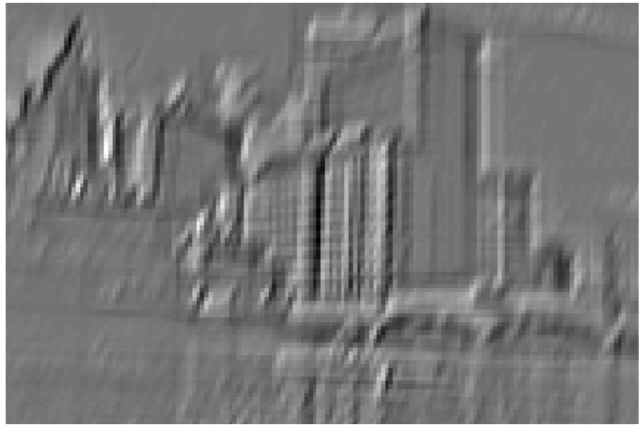
锐化

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

高斯模糊

http://1day-zh.gluon.ai

例子



(Rob Fergus)

http://1day-zh.gluon.ai

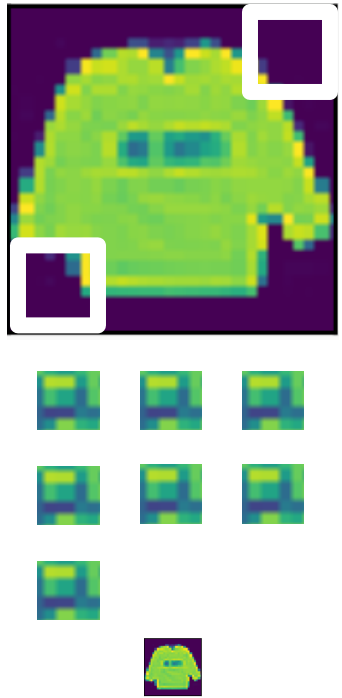
Convolutions Notebook

http://1day-zh.gluon.ai


填充和步幅

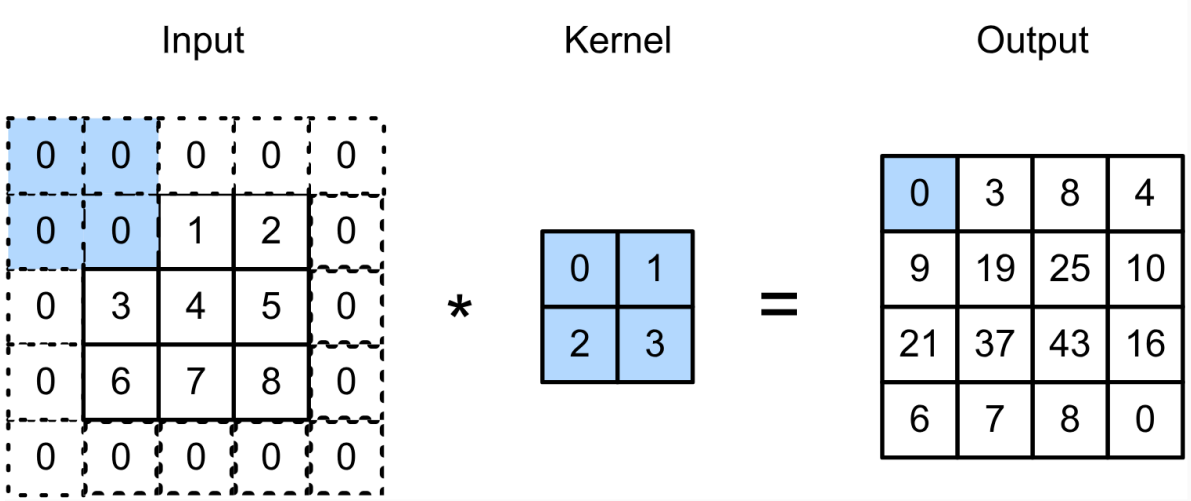
填充

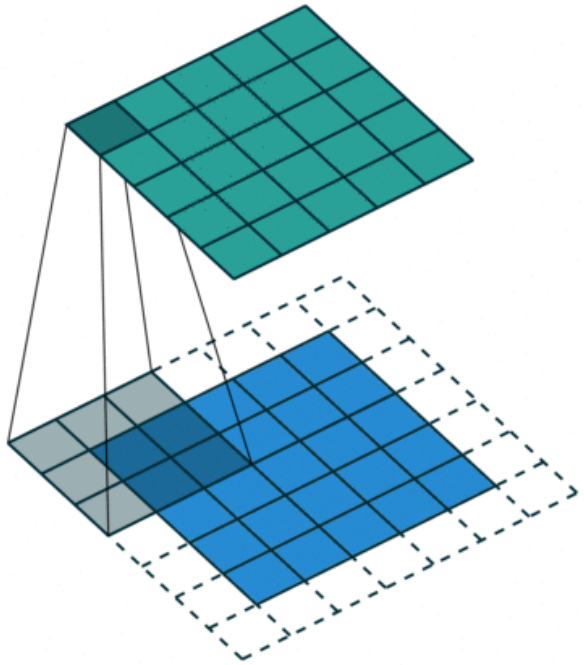
- 给定一张 32 x 32 输入图片
- 使用 5 x 5 核卷积
  - 使用一层：输出为 28 x 28
  - 使用七层：输出为 4 x 4
- 大尺寸的核导致形状大小锐减
  - 从 $n_h \times n_w$ 变成 $(n_h - k_h + 1) \times (n_w - k_w + 1)$



填充

在输入的四周填充行和列，通常填充值为0





$$0 \times 0 + 0 \times 1 + 0 \times 2 + 0 \times 3 = 0$$

http://1day-zh.gluon.ai

填充

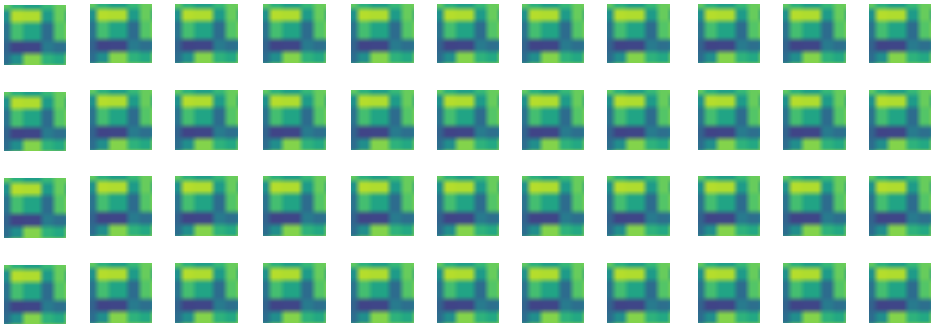
- 填充 p_h 行和 p_w 列，输出形状为

$$(n_h - k_h + p_h + 1) \times (n_w - k_w + p_w + 1)$$

- 通常使用 $p_h = k_h - 1$ 和 $p_w = k_w - 1$
  - 奇数 k_h：在两边填充 $p_h/2$
  - 偶数 k_h：上面填充 $\lceil p_h/2 \rceil$，下面 $\lfloor p_h/2 \rfloor$

步幅

- 填充可以让形状不变或者线性随层数减小
  - 使用 224 x 224 图片和 5 x 5 核卷积，需要 44 层来将输出变成 4 x 4，这样导致大量的计算

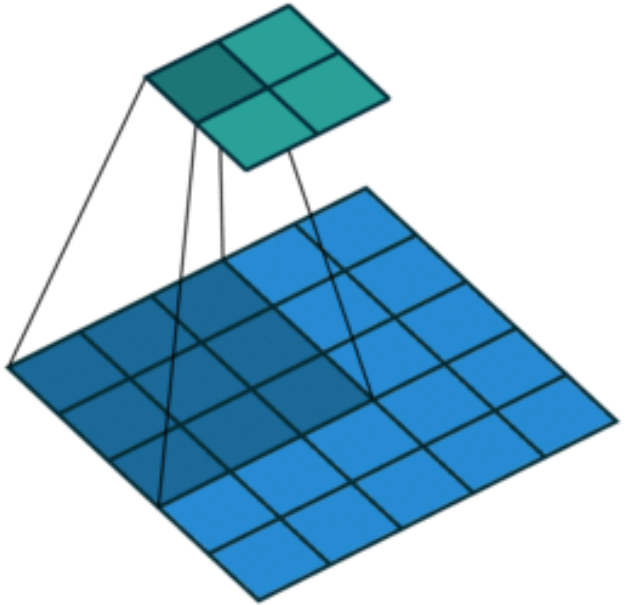


http://1day-zh.gluon.ai

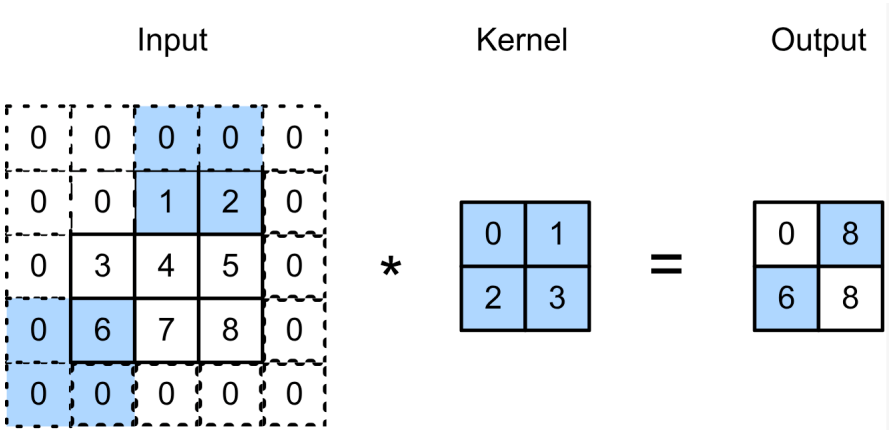
aws

步幅

- 步幅是每次滑动时越过的行、列数



高步幅3，宽步幅2



$$0\times0+0\times1+1\times2+2\times3=8$$
$$0\times0+6\times1+0\times2+0\times3=6$$

http://1day-zh.gluon.ai

步幅

- 给定高步幅 s_h 和宽步幅 s_w，输出的形状为

$$\lfloor (n_h - k_h + p_h + s_h)/s_h \rfloor \times \lfloor (n_w - k_w + p_w + s_w)/s_w \rfloor$$

- 使用 $p_h = k_h - 1$ 和 $p_w = k_w - 1$

$$\lfloor (n_h + s_h - 1)/s_h \rfloor \times \lfloor (n_w + s_w - 1)/s_w \rfloor$$

如果高和宽可以被步幅整除

$$(n_h/s_h) \times (n_w/s_w)$$


多输入和输出通道

多输入通道

- 彩色图片可能有 RGB 三个通道，转成灰度图片会损失信息

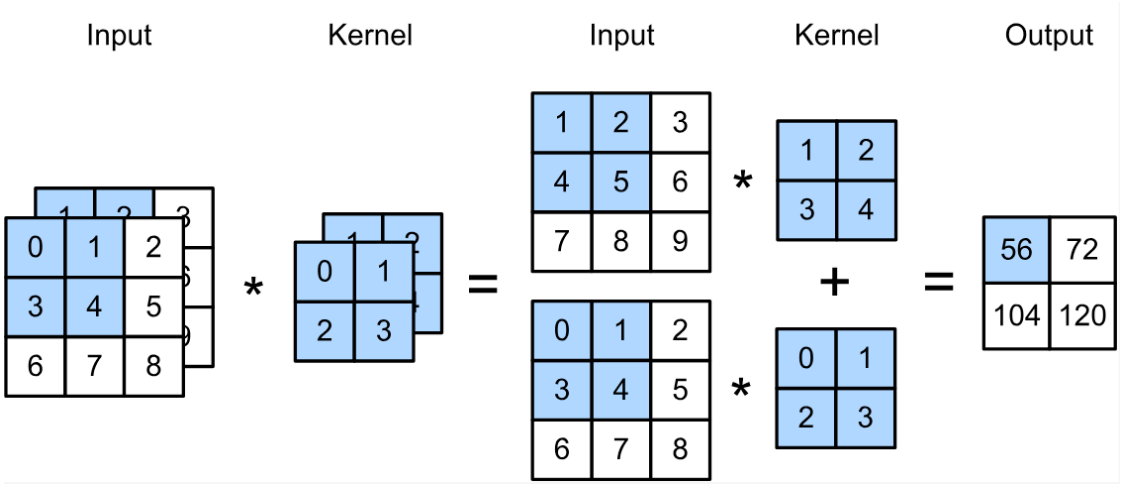
http://1day-zh.gluon.ai

多输入通道

- 彩色图片可能有 RGB 三个通道，转成灰度图片会损失信息

多输入通道

- 每个输出通道使用一个卷积核，然后对结果相加



$$(1\times1+2\times2+4\times3+5\times4)$$
$$+(0\times0+1\times1+3\times2+4\times3)$$
$$=56$$

http://1day-zh.gluon.ai

多输入通道

- 输入 $\mathbf{X}$：$c_i \times n_h \times n_w$
- 核 $\mathbf{W}$：$c_i \times k_h \times k_w$
- 输出 $\mathbf{Y}$：$m_h \times m_w$

$$\mathbf{Y} = \sum_{i=0}^{c_i} \mathbf{X}_{i,:,:} \star \mathbf{W}_{i,:,:}$$

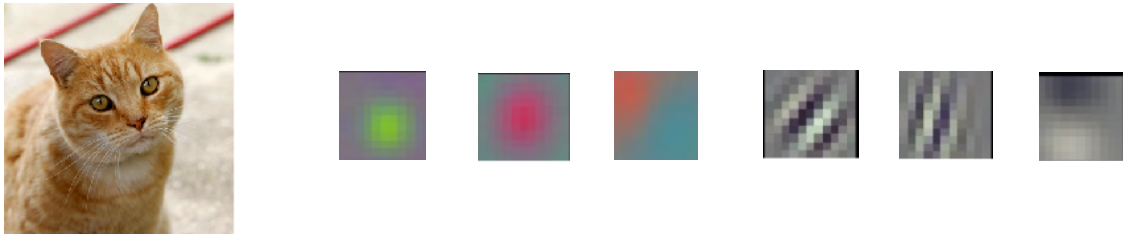
多输出通道

- 不论多少输入通道，输出通道总是1
- 我们可以有多个 3-D 核，每个核产生一个输出通道
- 输入 $\mathbf{X}: c_i \times n_h \times n_w$
- 核 $\mathbf{W}: c_o \times c_i \times k_h \times k_w$
- 输出 $\mathbf{Y}: c_o \times m_h \times m_w$

$$\mathbf{Y}_{i,:,:} = \mathbf{X} \star \mathbf{W}_{i,:,:,:} \quad \text{for } i = 1, \ldots, c_o$$

多输入、输出通道

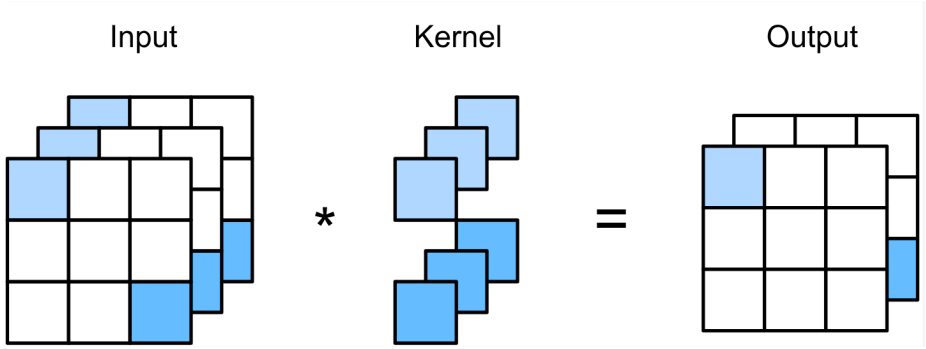
- 每个输出通道可能识别某个模式



- 讲多个输入通道模式合并来识别一个更高层的模式

1 x 1 卷积层

一个常用选择是 $k_h = k_w = 1$。它不识别空间信息，但混合通道



等价一个全连接层，它有 $n_h n_w \times c_i$ 输入，权重为 $c_o \times c_i$

2-D 卷积层总结 Layer Summary

- 输入 $\mathbf{X}: c_i \times n_h \times n_w$
- 核 $\mathbf{W}: c_o \times c_i \times k_h \times k_w$
- 偏移 $\mathbf{B}: c_o \times c_i$
- 输出 $\mathbf{Y}: c_o \times m_h \times m_w$

$$\mathbf{Y} = \mathbf{X} \star \mathbf{W} + \mathbf{B}$$

- 计算复杂度（浮点计算数 FLOP）$O(c_i c_o k_h k_w m_h m_w)$

  $c_i = c_o = 100, k_h = h_w = 5, m_h = m_w = 64$ -> 1GFLOP

- 10 层, 1M 样本: 10PF
  (CPU: 0.15 TF = 18h, GPU: 12 TF = 14min)

池化层

池化层

- 卷积对位置敏感
  - 检测垂直边

X

```
[[1. 1. 0. 0. 0.
 [1. 1. 0. 0. 0.
 [1. 1. 0. 0. 0.
 [1. 1. 0. 0. 0.
```

Y

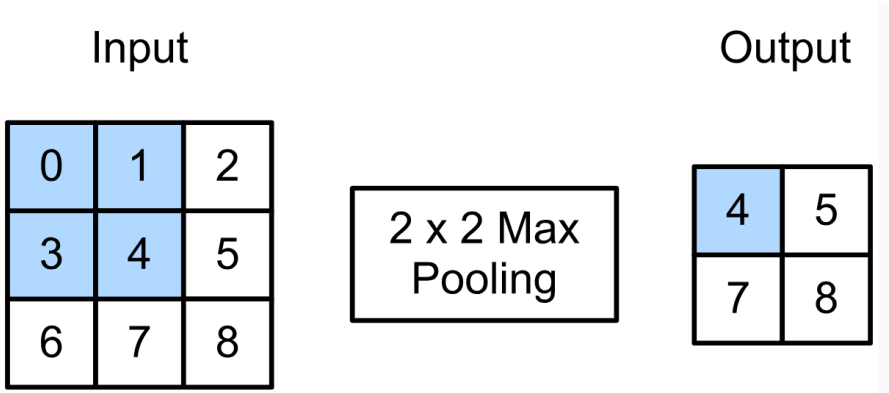
```
[[ 0.  1.  0.  0.
 [ 0.  1.  0.  0.
 [ 0.  1.  0.  0.
 [ 0.  1.  0.  0.
```

1 个像素偏移
导致输出为 0

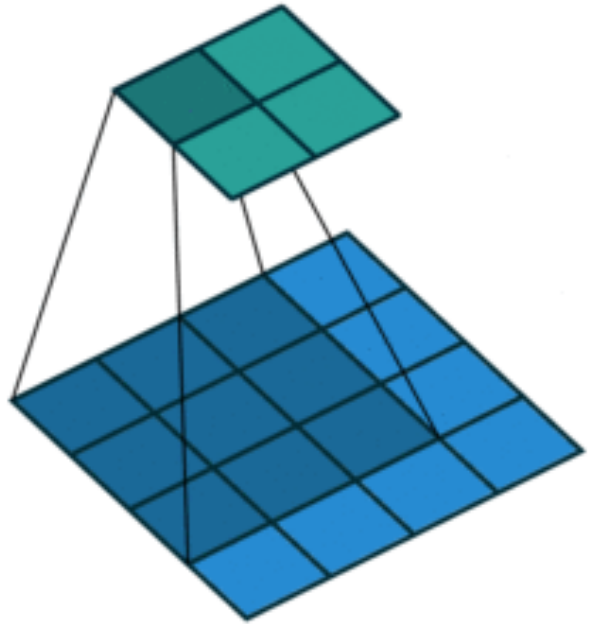
- 我们需要一定程度上变化无关
  - 光照，物体位置，大小，外观变化

2-D 最大池化层

- 返回在一个滑动窗里的最大值



$$\max(0,1,3,4) = 4$$



2-D 最大池化层

- 返回在一个滑动窗里的最大值

检测垂直边

```
[[1. 1. 0. 0. 0.
 [1. 1. 0. 0. 0.
 [1. 1. 0. 0. 0.
 [1. 1. 0. 0. 0.
```

卷积输出

```
[[ 0.  1.  0.  0.
 [ 0.  1.  0.  0.
 [ 0.  1.  0.  0.
 [ 0.  1.  0.  0.
```

2 x 2 最大池化层

```
[[ 1.  1.  1.  0.
 [ 1.  1.  1.  0.
 [ 1.  1.  1.  0.
 [ 1.  1.  1.  0.
```

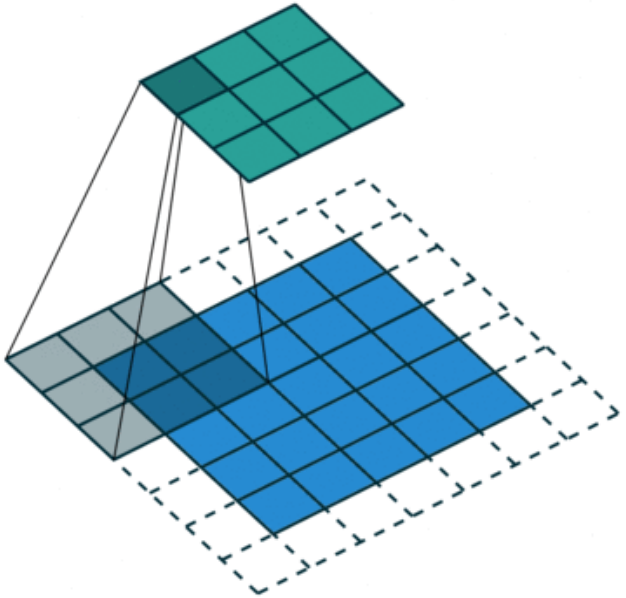
容忍一个像素偏移

http://1day-zh.gluon.ai

填充、步幅、和多通道

- 池化层的填充和步幅与卷积层类似
- 没有可学习的参数
- 对每个输入通道应用池化得到对应输出通道结果

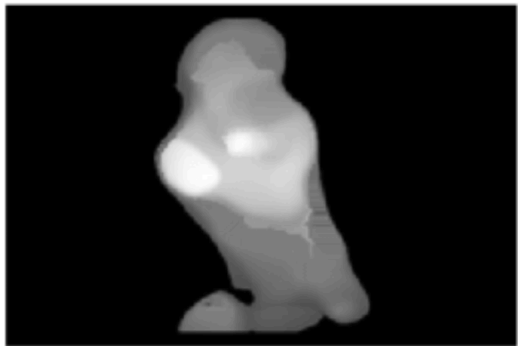
输出通道数 = 输入通道数



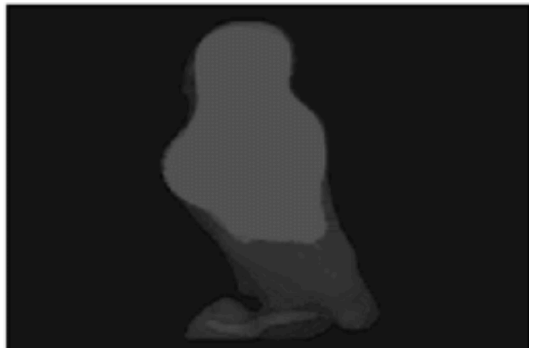
平均池化

- 最大池化：滑动窗里的最大信号值
- 平均池化：将 max 变成 mean
  - 窗口里的平均信号值

Max pooling



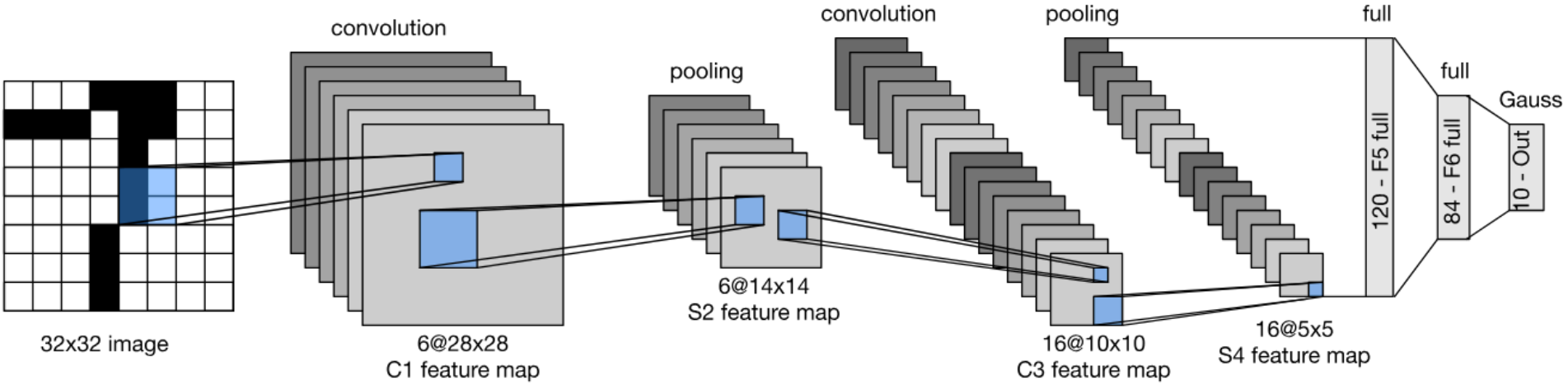
Average pooling



Pooling Notebook

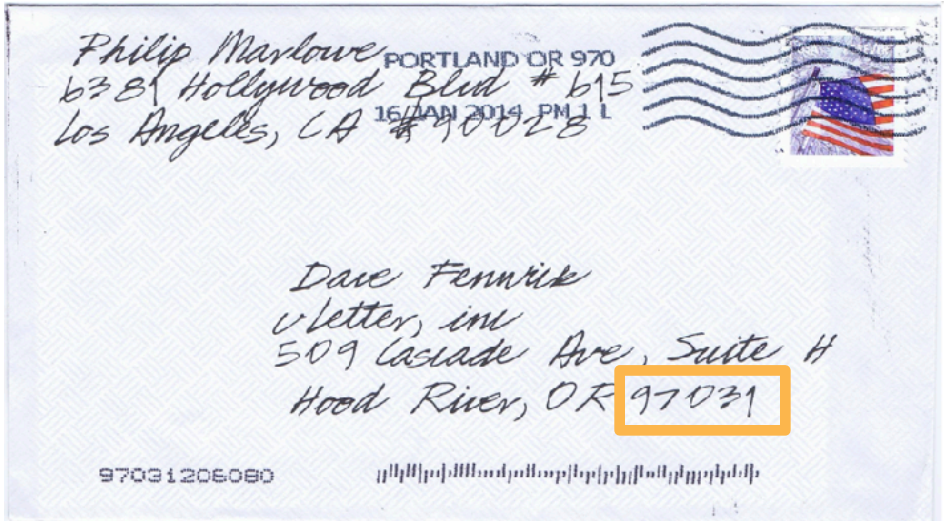
http://1day-zh.gluon.ai

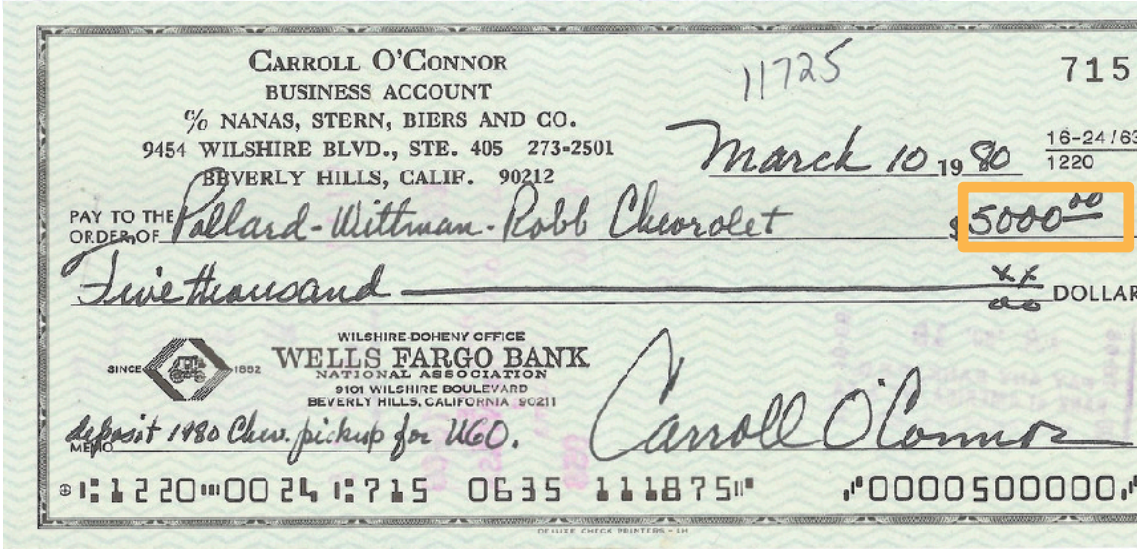
LeNet



http://1day-zh.gluon.ai

手写数字识别

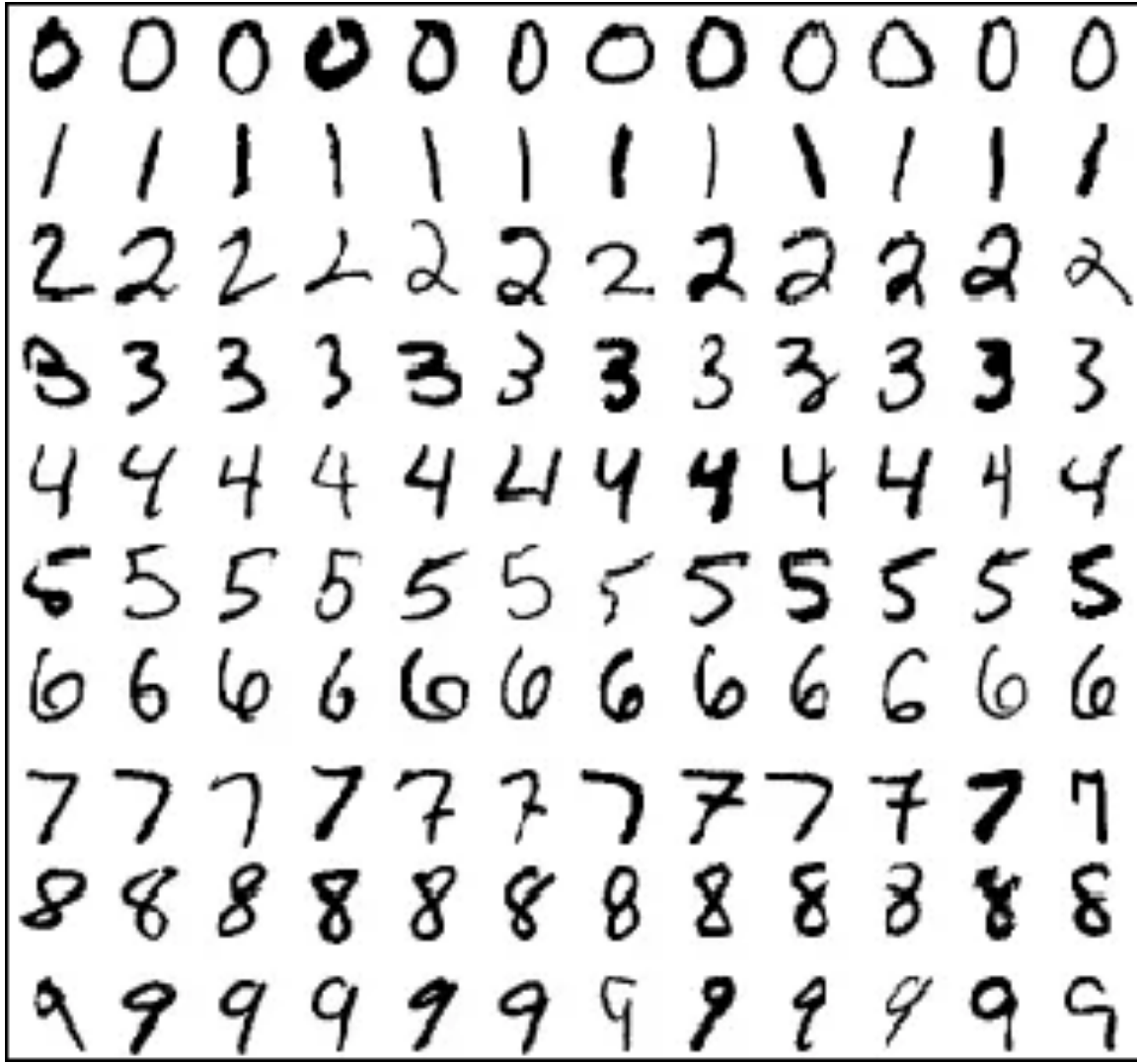




http://1day-zh.gluon.ai

MNIST

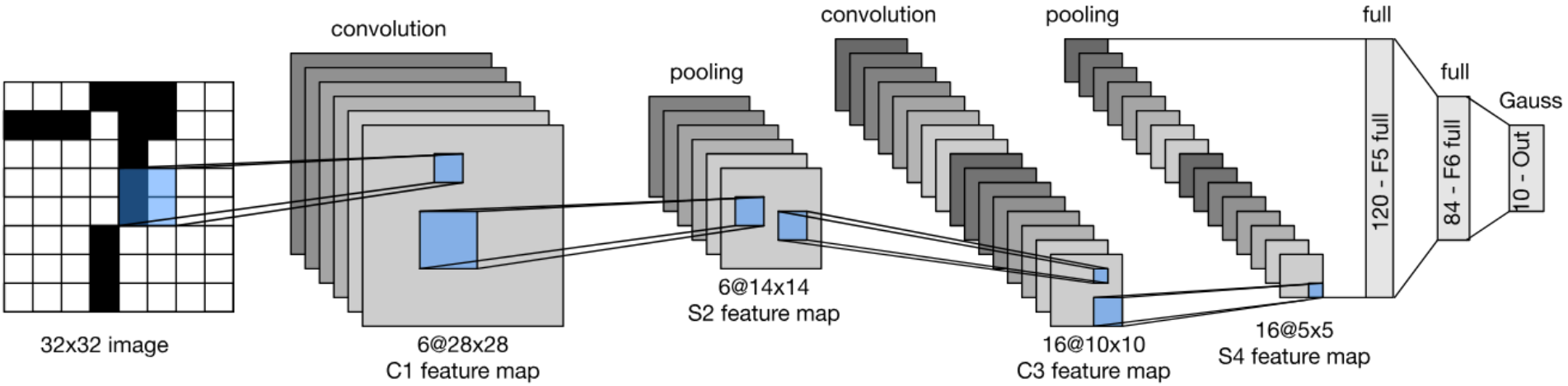
- 居中且剪裁了
- 50,000 训练样本
- 10,000 测试样本
- 28 x 28 灰度图片
- 10 类



http://1day-zh.gluon.ai

Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, 1998 Gradient-based learning applied to document recognition

aws

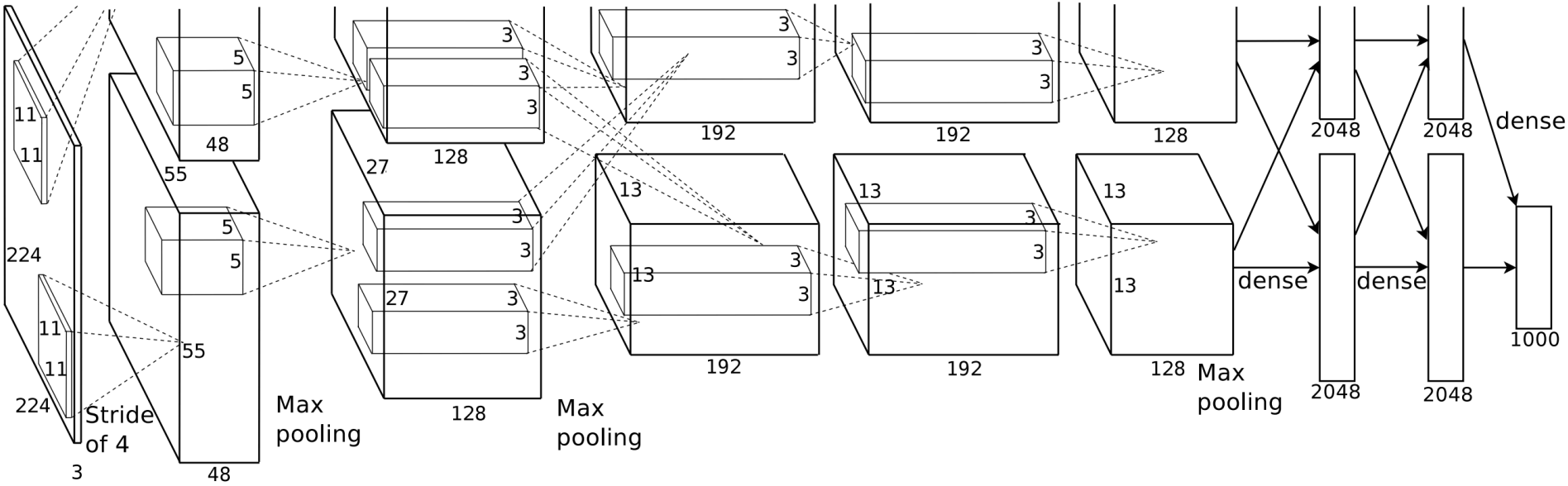


http://1day-zh.gluon.ai

LeNet Notebook

http://1day-zh.gluon.ai

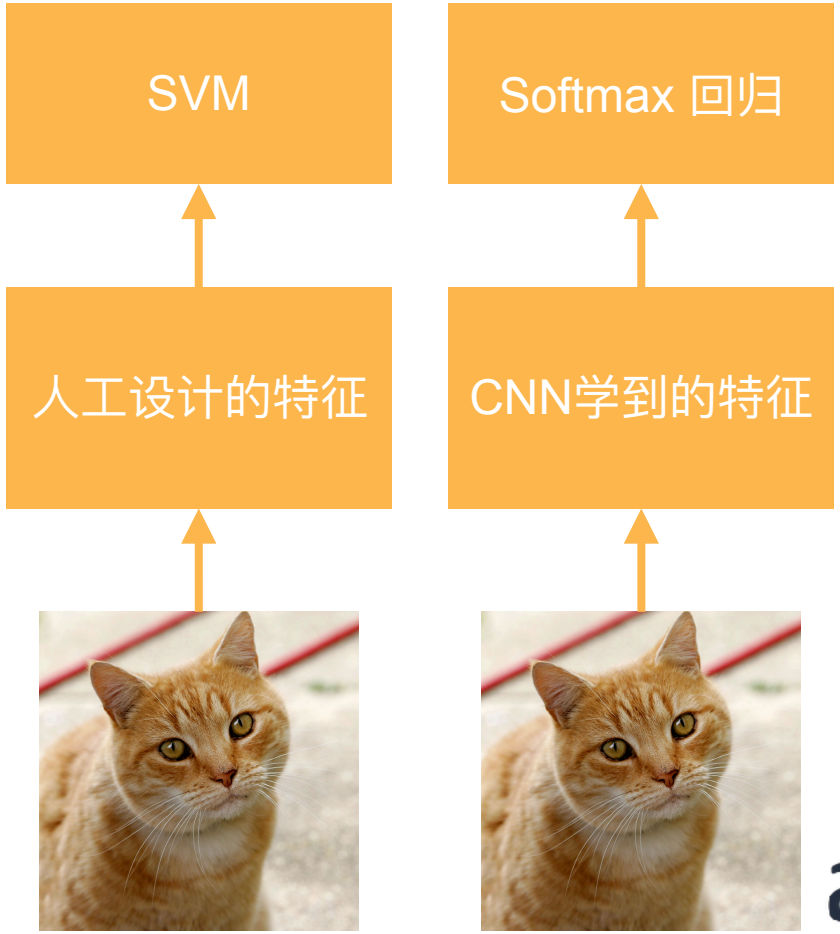
AlexNet



http://1day-zh.gluon.ai

AlexNet

- AlexNet 在 ImageNet 2012 竞赛中获得了冠军
- 更深更大的LeNet
- 关键改动
  - Dropout
  - ReLu
  - MaxPooling
- 观点上的改变

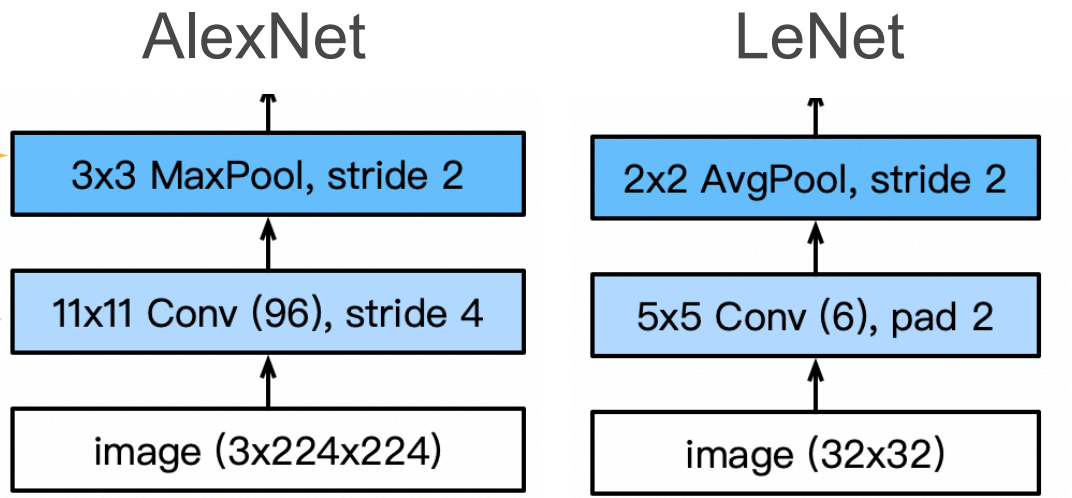


http://1day-zh.gluon.ai

AlexNet 架构

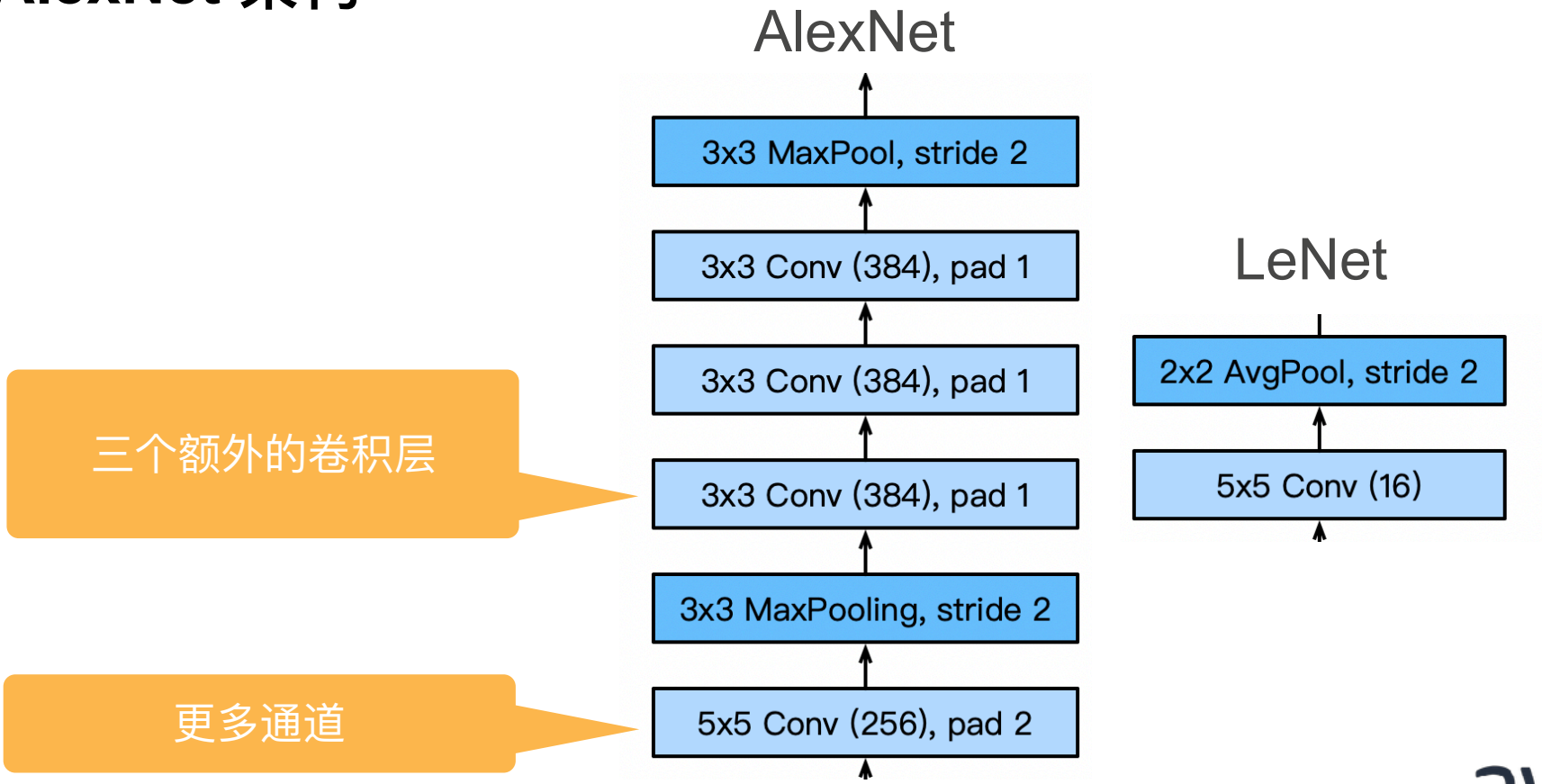
更大的池化窗口

更大的卷积窗和步幅，更多通道。因为输入更大了



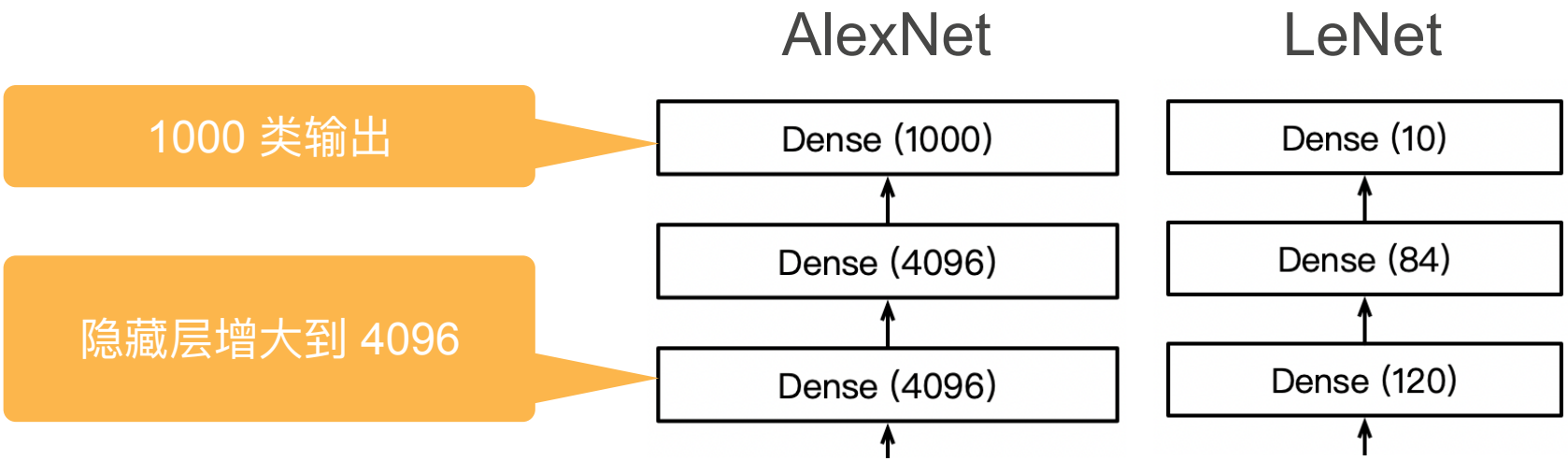
http://1day-zh.gluon.ai

AlexNet 架构



http://1day-zh.gluon.ai

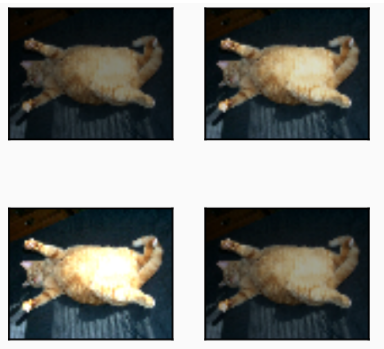
AlexNet 架构

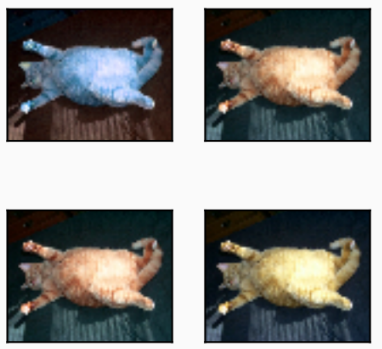


http://1day-zh.gluon.ai

更多技巧

- 激活函数从 sigmoid 变成 ReLu (没有梯度消失)
- 增加 dropout 层 (更鲁棒)
- 数据增强

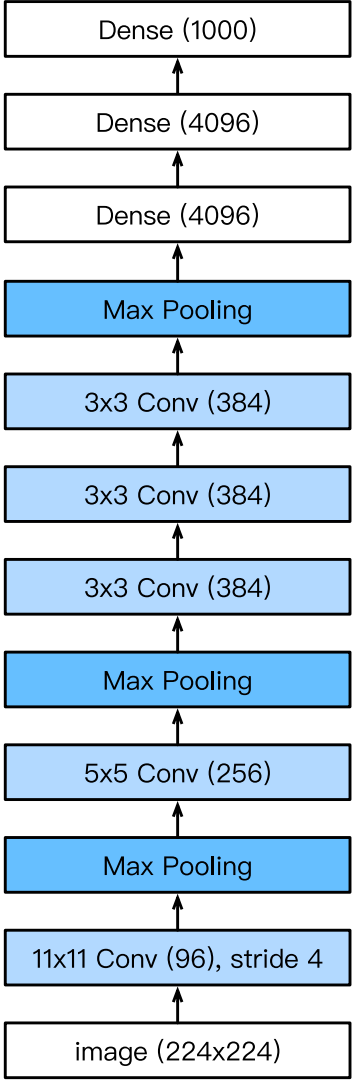




http://1day-zh.gluon.ai

复杂度

	参数数量		FLOP	
	AlexNet	LeNet	AlexNet	LeNet
Conv1	35K	150	101M	1.2M
Conv2	614K	2.4K	415M	2.4M
Conv3-5	3M		445M	
Dense1	26M	0.48M	26M	0.48M
Dense2	16M	0.1M	16M	0.1M
Total	46M	0.6M	1G	4M
Increase	11x	1x	250x	1x

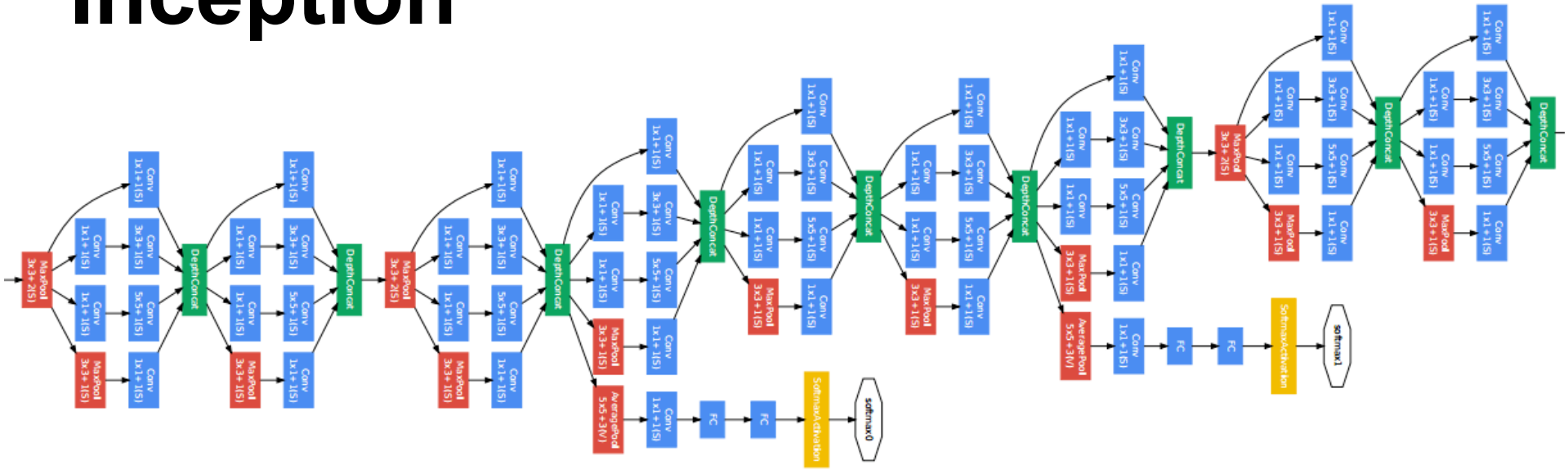


http://1day-zh.gluon.ai

AlexNet Notebook

http://1day-zh.gluon.ai

Inception

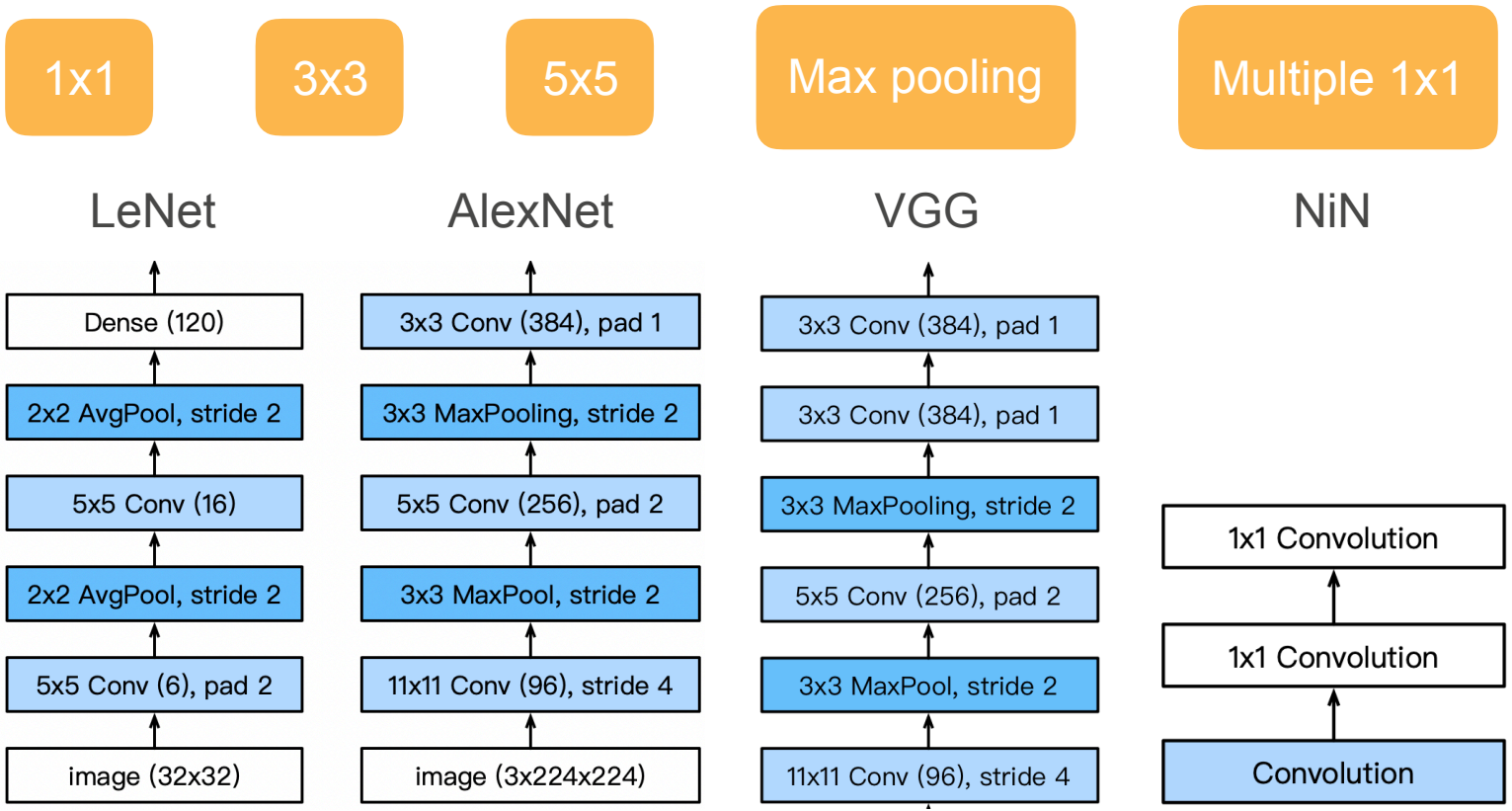


http://1day-zh.gluon.ai

aws

如何选取最好的卷积层

1x1 | 3x3 | 5x5 | Max pooling | Multiple 1x1



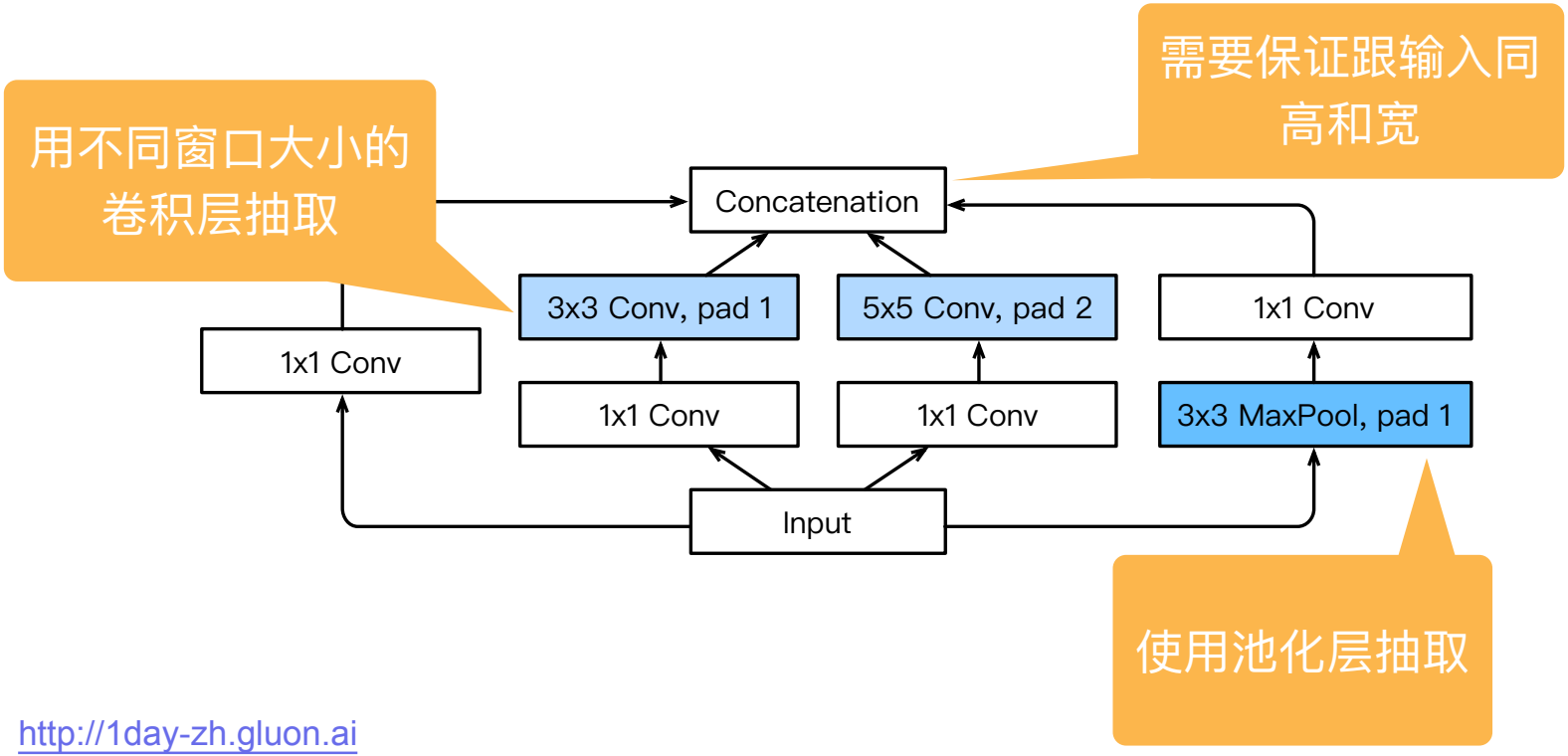
http://1day-zh.gluon.ai

只有别人做选择。我要全部

http://1day-zh.gluon.ai

Inception 块

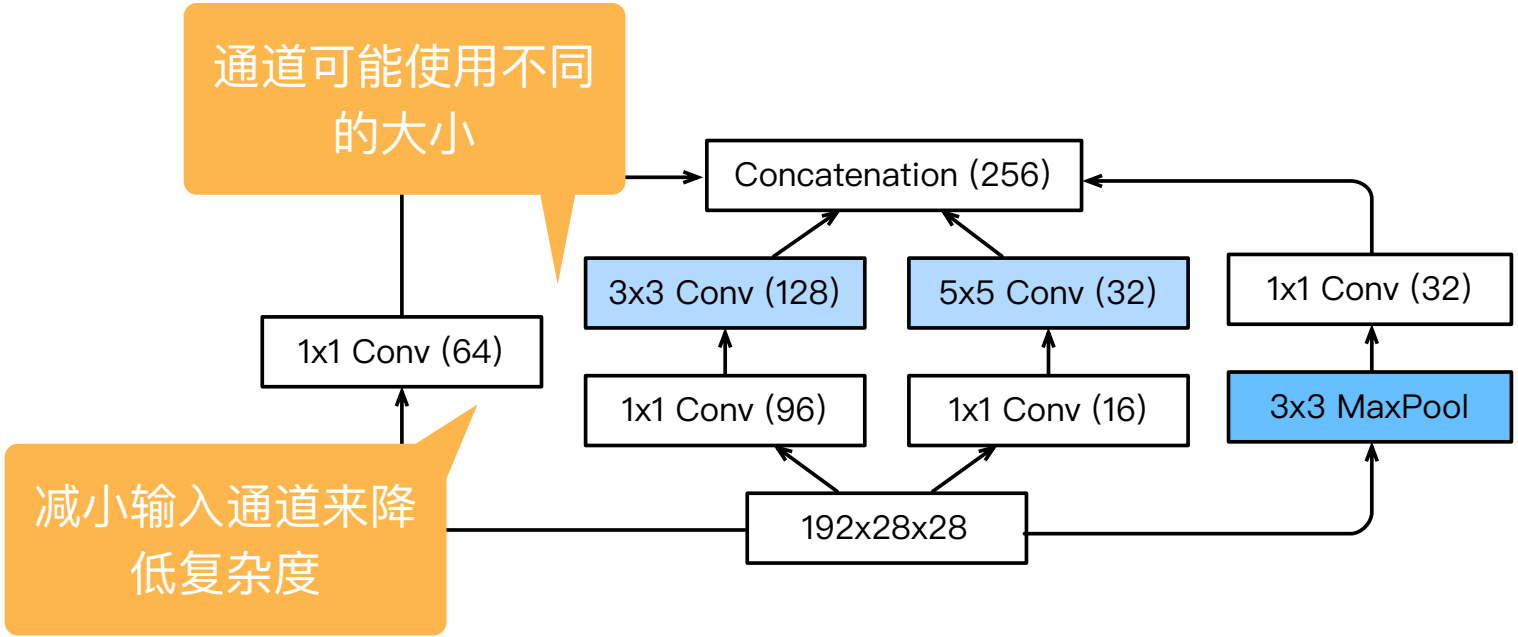
4 条通道分别抽取不同的方面信息，然后在通道维上合并



http://1day-zh.gluon.ai

Inception 块

第一个 Inception 块，以及每个通道的大小



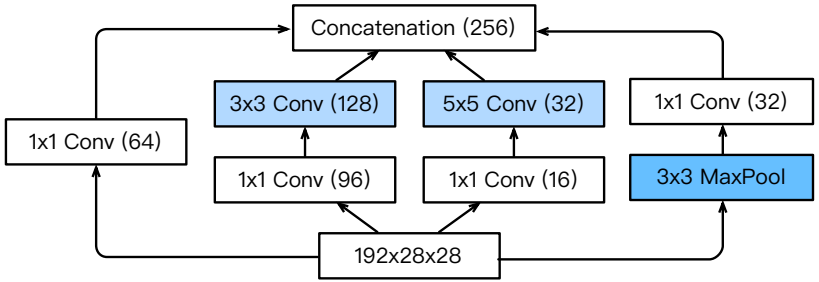
http://1day-zh.gluon.ai

Inception 块

Inception 块通常比单 3x3 或 5x5 卷积层设计有更少的可学习参数和计算量

- 混合不同的函数
- 内存和计算有效

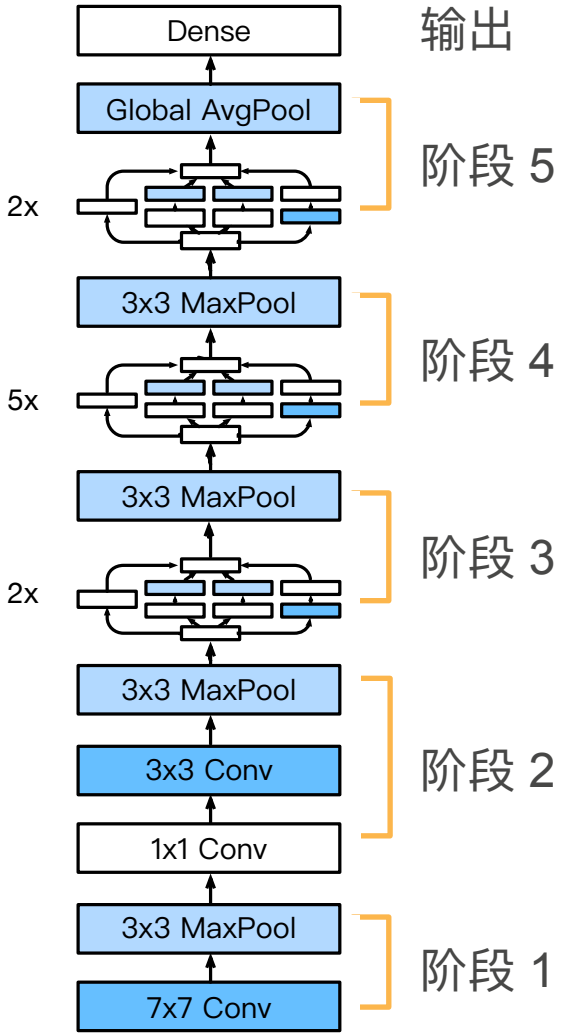
	#parameters	FLOPS
Inception	0.16 M	128 M
3x3 Conv	0.44 M	346 M
5x5 Conv	1.22 M	963 M



http://1day-zh.gluon.ai

GoogLeNet

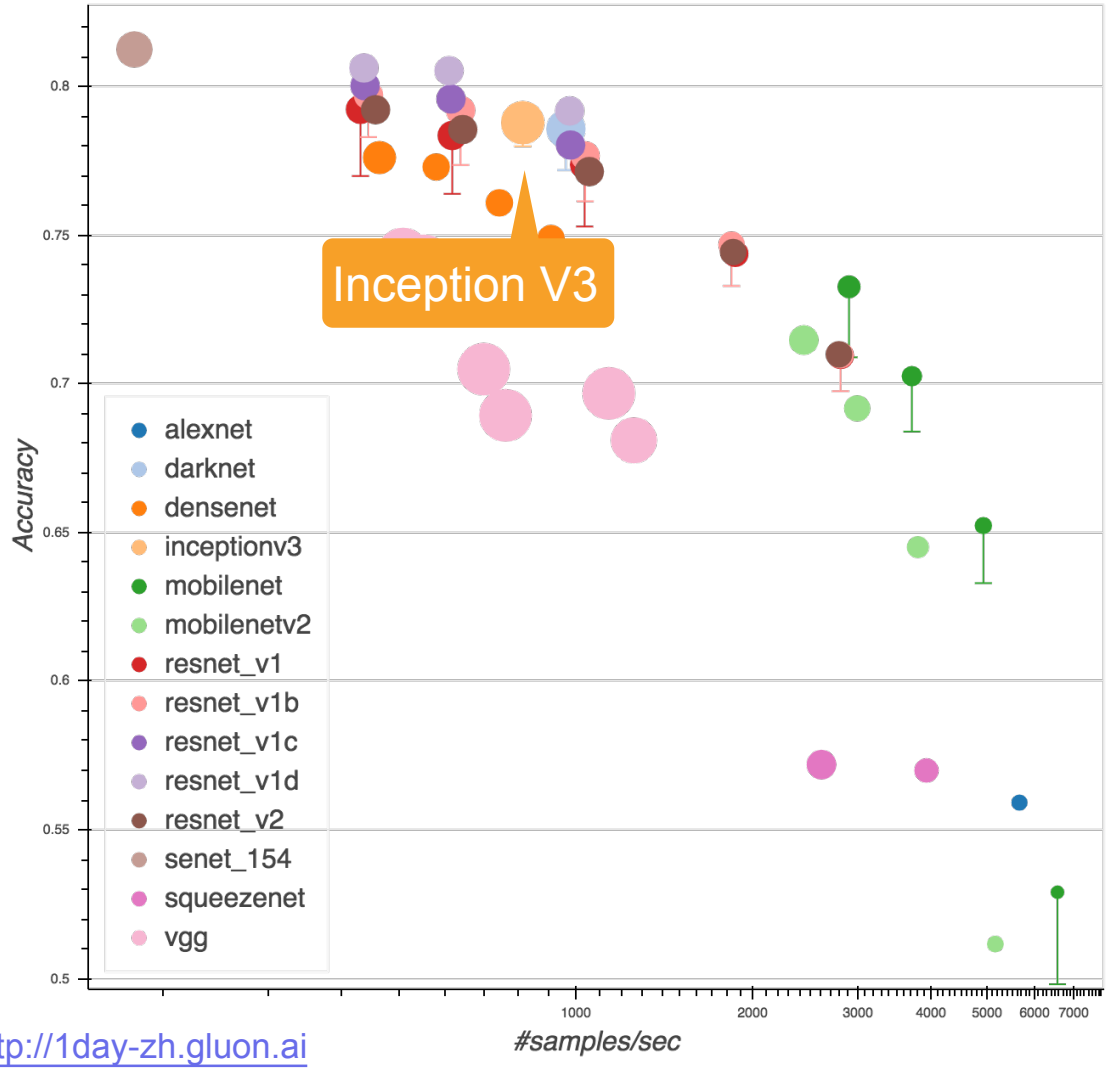
- 5 阶段，共 9 个 inception 块



http://1day-zh.gluon.ai

不同的 Inception 架构

- Inception-BN (v2) — 使用批量归一化
- Inception-V3 — 修改了 inception 块
  - 替换 5x5 成多个 3x3 卷积层
  - 替换 5x5 成 1x7 和 7x1 卷积层
  - 替换 3x3 成 1x3 和 3x1 卷积层
  - 更深
- Inception-V4 — 加了残差连接



GluonCV Model Zoo

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

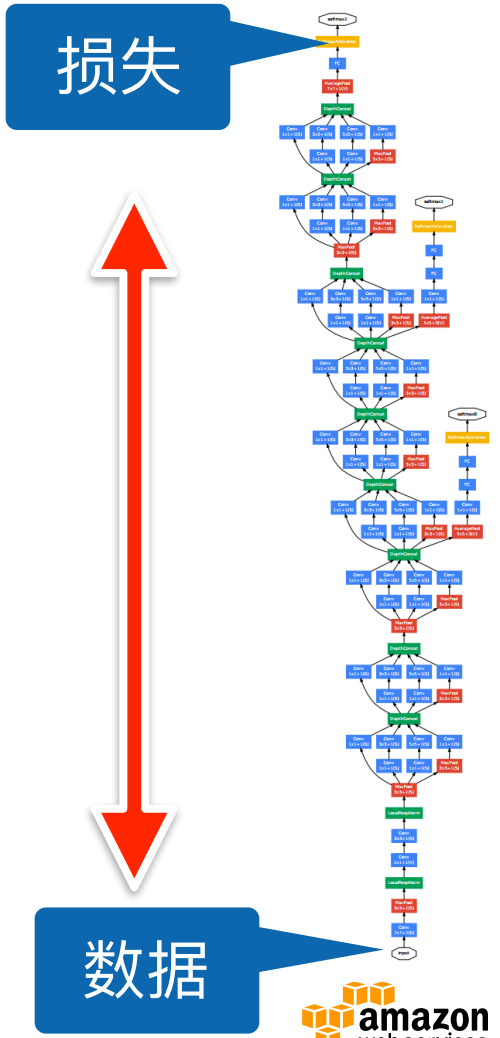
http://1day-zh.gluon.ai

aws

http://1day-zh.gluon.ai

批量归一化

- 损失通常在最后一层
  - 顶部层通常学习更快
- 数据是从底部进入
  - 底部层变化会改变所有层
  - 顶部层需要重新学习很多次
  - 收敛更慢
- 我们能不能在更新底部层的时候不要改变太多顶部层



http://1day-zh.gluon.ai

批量归一化

- 我们能不能在更新底部层的时候不要改变太多顶部层
- 固定均值和方差

$$\mu_B = \frac{1}{|B|}\sum_{i\in B} x_i \text{ and } \sigma_B^2 = \frac{1}{|B|}\sum_{i\in B}(x_i - \mu_B)^2 + \epsilon$$

然后分别调整

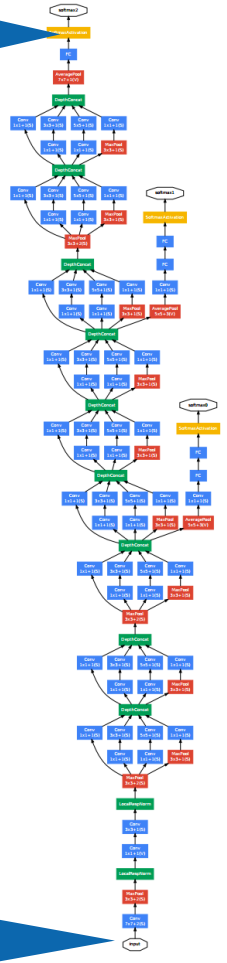
$$x_{i+1} = \gamma\frac{x_i - \mu_B}{\sigma_B} + \beta$$

均值

方差

损失

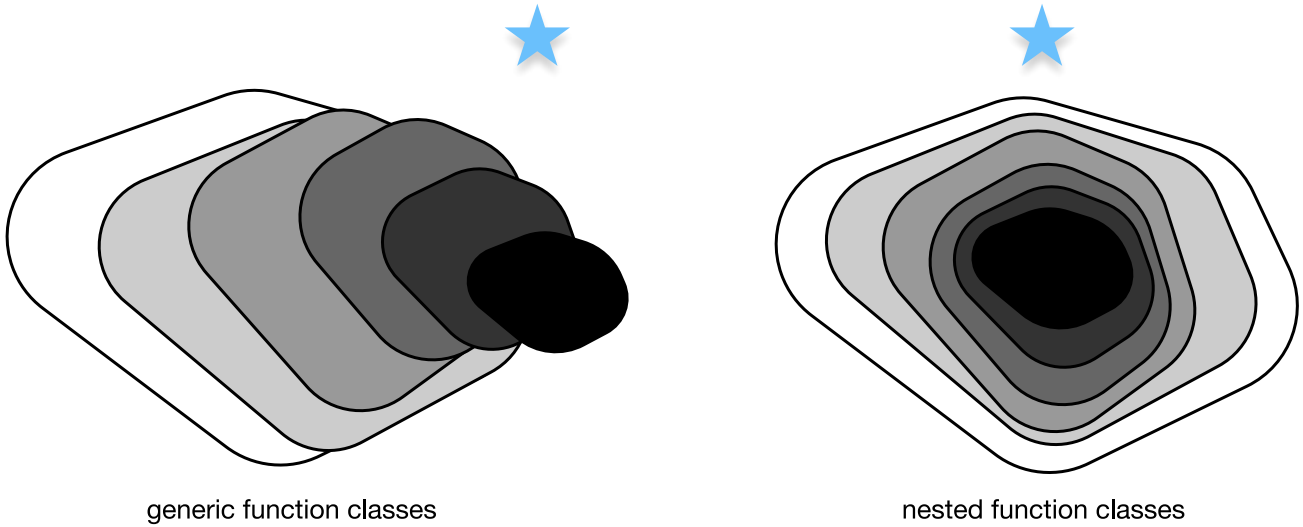
数据



http://1day-zh.gluon.ai

amazon web services

残差网络

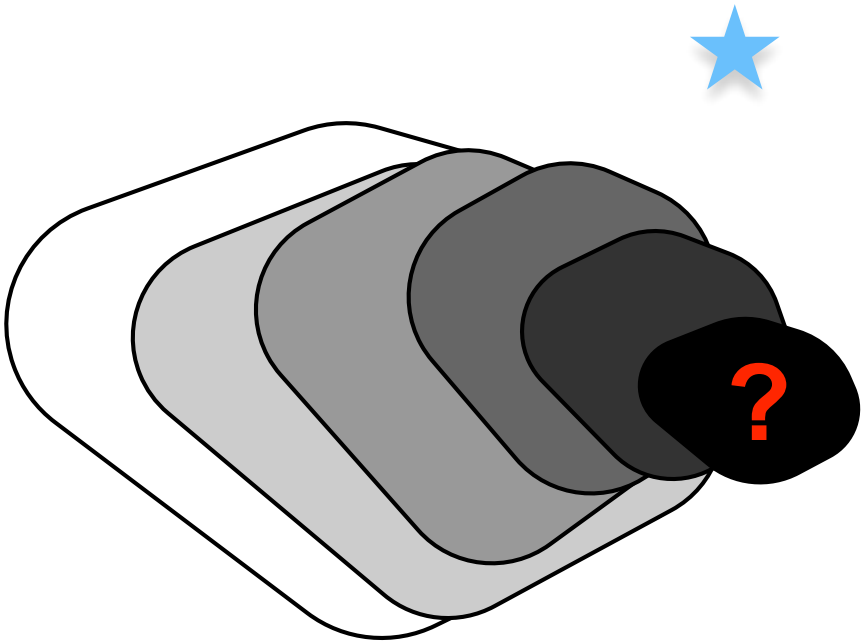


generic function classes

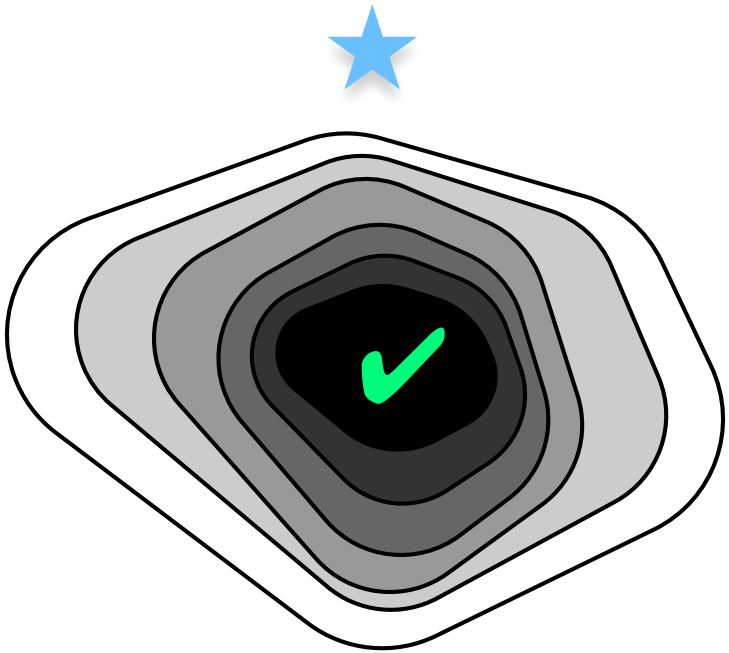
nested function classes

http://1day-zh.gluon.ai

加层总是能提升精度吗?



generic function classes



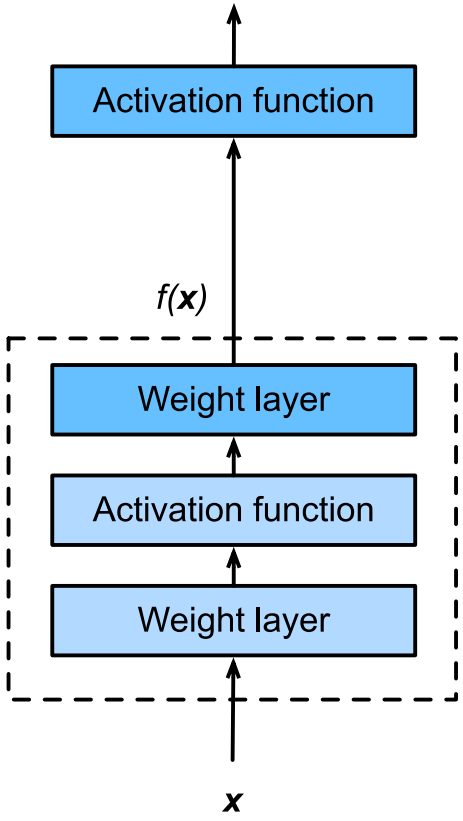
nested function classes

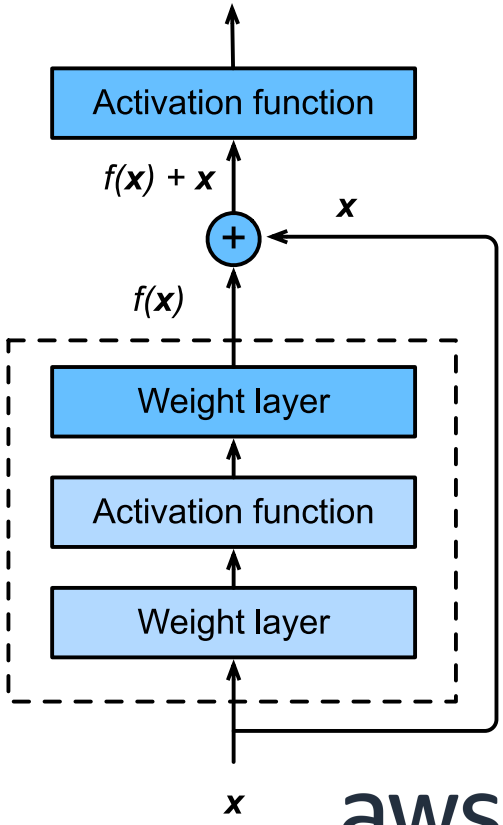
http://1day-zh.gluon.ai

残差网络

- 加一层会改变函数类
- 我们想加一个函数
- ‘Taylor 展开’ 风格

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

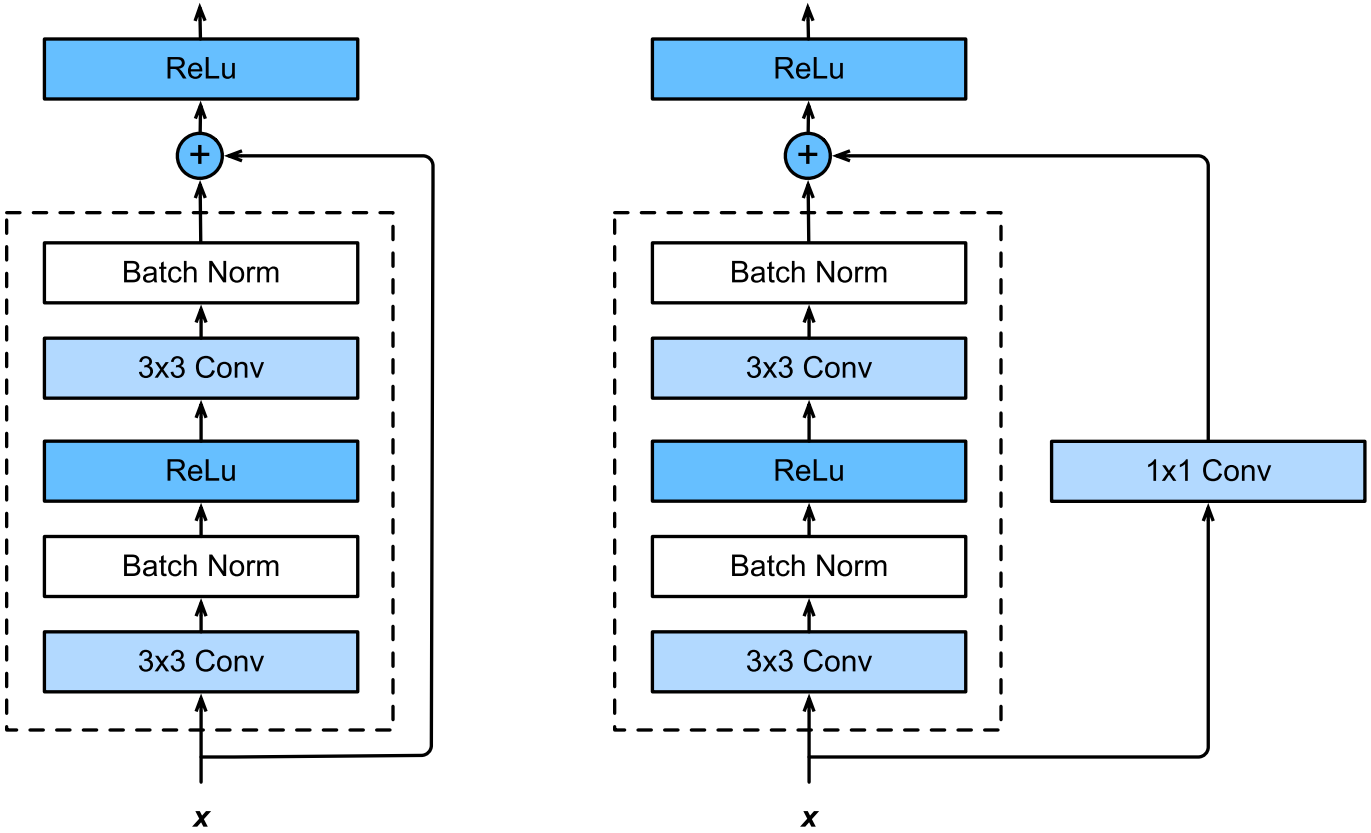




He et al., 2015

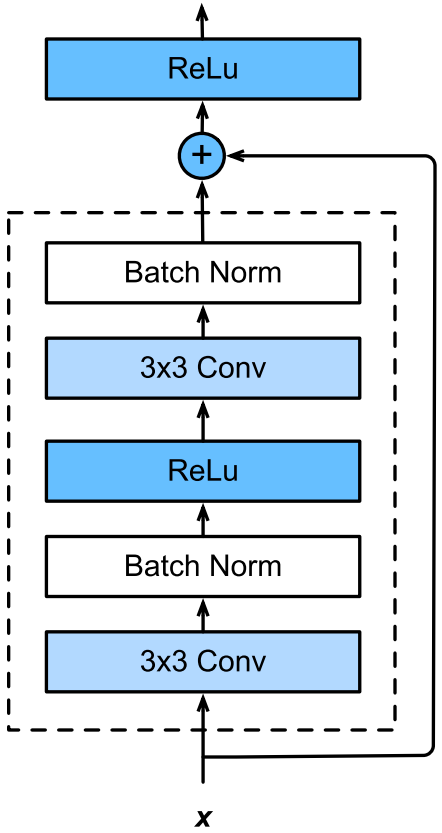
http://1day-zh.gluon.ai

残差块细节



http://1day-zh.gluon.ai

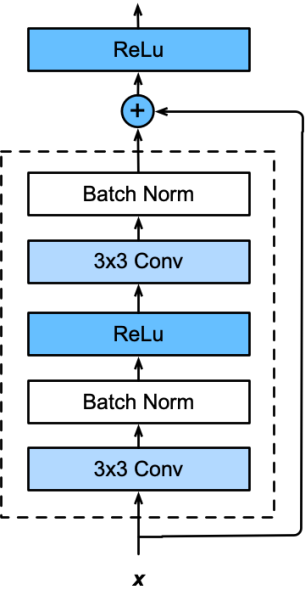
代码实现



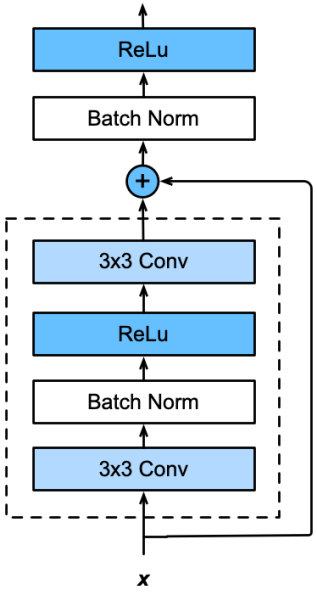

```
def forward(self, X):
  Y = npx.relu(self.bn1(self.conv1(X)))
  Y = self.bn2(self.conv2(Y))
  return npx.relu(Y + X)
```

http://1day-zh.gluon.ai

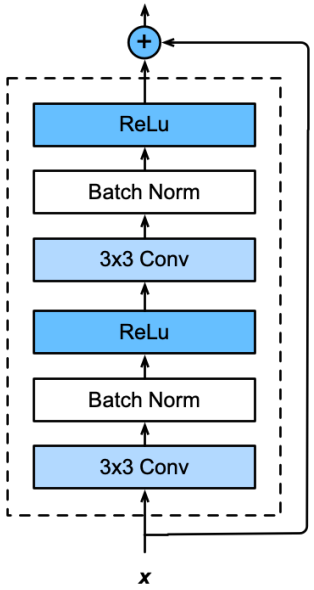
残差块有很多方式



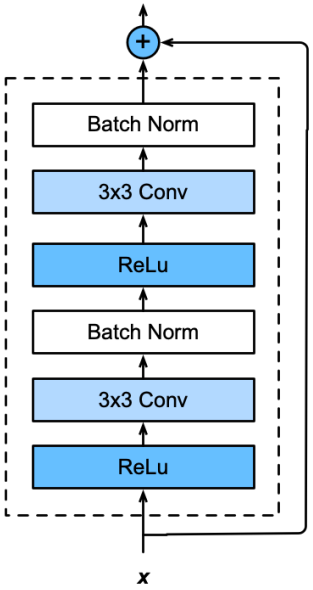
original



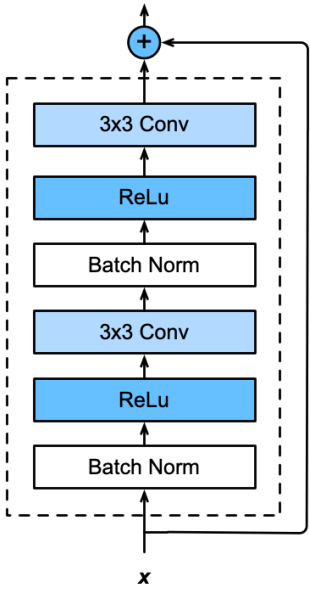
Batch Norm after addition



ReLu last



Batch Norm last



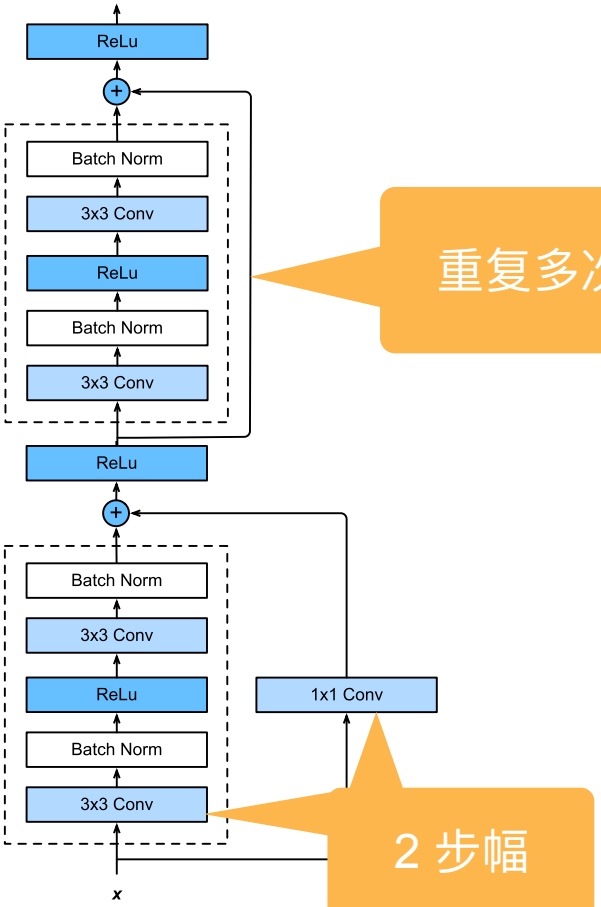
Pre-activation

使用各种不同排列

http://1day-zh.gluon.ai

aws

ResNet 块

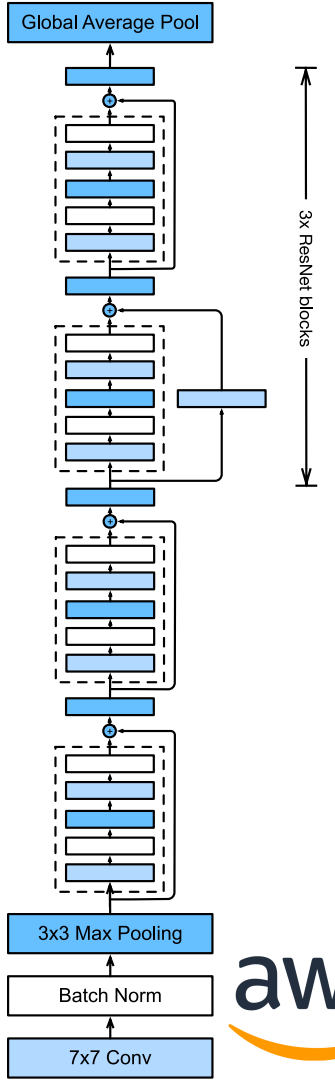


- 每块里面先减半高宽 (stride=2)
- 累加残差块

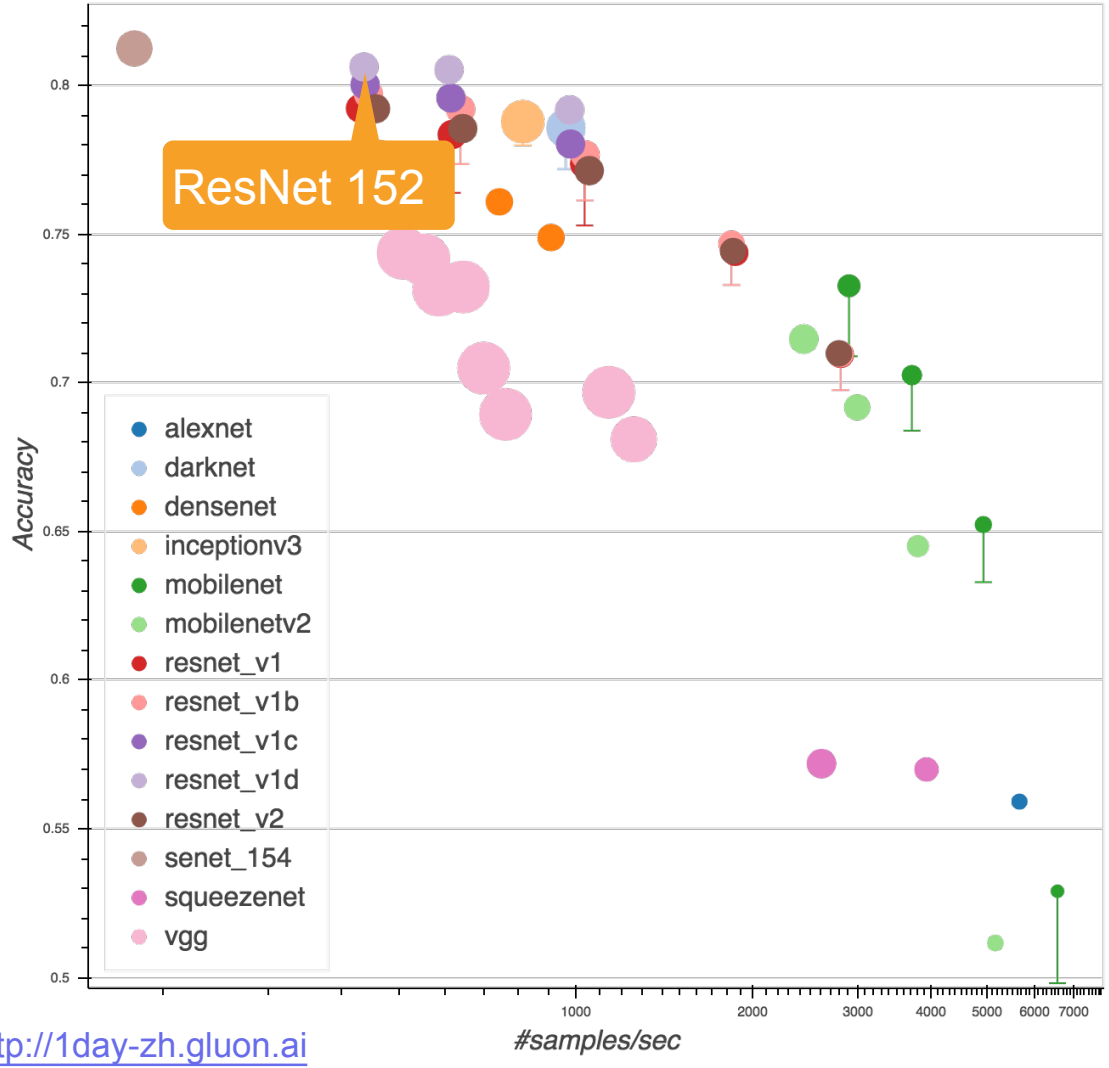
http://1day-zh.gluon.ai

ResNet 架构

- 跟 VGG 和 GoogleNet 一样的总体架构
- 通过残差连接来增加表达性
- 池化和步幅来降低复杂度
- 批量归一化来做正则



http://1day-zh.gluon.ai



GluonCV Model Zoo

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

http://1day-zh.gluon.ai

Jupyter Notebook

http://1day-zh.gluon.ai

总结

- GPUs
- 卷积，填充，步幅，池化
- 卷积神经网络 (LeNet)
- 深度卷积神经网络 (AlexNet)
- 使用重复元素 (VGG)
- 残差网络 (ResNet)