

CNN

神经网络处理图像时遇到的问题

解决问题

1. 卷积

原理:

性质:

公式:

卷积——处理多通道

多个卷积核

2. 激活函数

关于激活函数的选择

Summary

公式: 卷积核的参数

3. 池化

性质:

4. 全连接层

5. CNN结构

模型进化

AlexNet

结构:

trick:

其他细节

VGNet

Summary:

视野域:

网络结构:

训练trick:

ResNet

模型结构:

InceptionNet

背景:

结构:

MobileNet

结构:

CNN

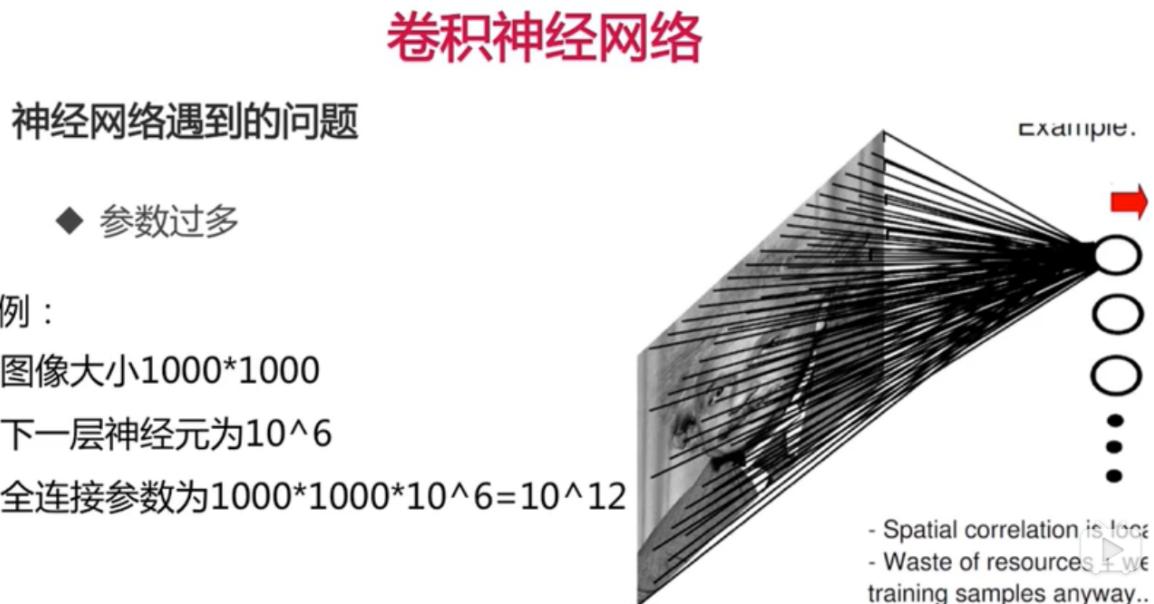
参考: <https://www.cnblogs.com/wj-1314/p/9593364.html>

为了图像问题做了一些特有操作。

- 卷积
- 激活函数
- 池化
- 全连接

神经网络处理图像时遇到的问题

- 参数量太多



- 模型参数太多，对训练集的表达能力就过强，可以理解为记住了这个训练集图像，从而导致过拟合即泛化能力太弱。
- 某些数据会导致模型容易收敛到较差的局部极值。
- 没有利用到位置信息

对于图像任务来说，关键部位的像素之间关系紧密，但不同关键部位之间的联系不紧密；对于全连接来说，相当于把所有位置的信息都要注意到，那么非关键位置的权重最后就会很小，但是训练这些位置要花费很长时间，太亏了。

- 网络层数限制

网络层数越多，其表达能力越强，但是通过梯度下降方法训练深度全连接神经网网络很困难，因为全连接神经网网络的梯度很难传递超过三层。我们不能得到一个很深的全连接NN，也就限制了它的表达能力。

解决问题

1. 卷积

原理：

- 不使用全连接，使用Locally Connected Neural Net局部连接，利用了图像的区域性。

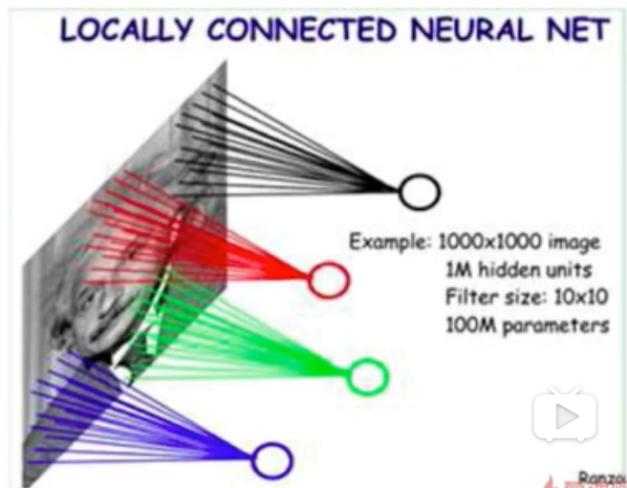
卷积神经网络

卷积——解决问题

◆ 局部连接

举例：

- 图像大小 1000×1000
- 下一层神经元为 10^6
- 局部连接范围为 10×10
- 全连接参数为 $10 \times 10 \times 10^6 = 10^8$



- 参数共享

不让神经元只学固定位置的特征，使用参数共享，即强制每个神经元使用同样的权值和偏置参数，则全连接参数：

$10 \times 10 = 10^2$ ，利用了图像特征与位置无关。

性质：

1. 输入图像、卷积核和输出之间的size关系：

输出size = 输入size - 卷积核size + 1；如图所示，输入图像是 5×5 ，卷积核 3×3 ，对应输出 3×3 。

卷积——每个位置进行计算

输入图像					卷积核			输出		
1	2	3	4	5	1	0	1	35	?	?
6	7	8	9	10	0	1	0	?	?	?
11	12	13	14	15	1	0	1	?	?	?
16	17	18	19	20						
21	22	23	24	25						

点积乘法： $\text{result} = 1 \times 1 + 1 \times 3 + 1 \times 7 + 1 \times 11 + 1 \times 13 = 35$

2. 计算方法：求数量积，如上图。

3. 步长：

卷积——步长为2

输入图像					卷积核			输出	
1	2	3	4	5					
6	7	8	9	10					
11	12	13	14	15					
16	17	18	19	20					
21	22	23	24	25					

点积乘法： $\text{result} = 1 \times 3 + 1 \times 5 + 1 \times 9 + 1 \times 13 + 1 \times 15 = 45$

但是发现，通过卷积核的方式，输出的图像都缩小了，所以我们要把原图像放大，即 padding

padding

卷积——padding使输出size不变

输入图像							卷积核			输出				
0	0	0	0	0	0	0	1	0	1	?	?	?	?	?
0	1	2	3	4	5	0	0	1	0	?	?	?	?	?
0	6	7	8	9	10	0	1	0	1	?	?	?	?	?
0	11	12	13	14	15	0	0	1	0	?	?	?	?	?
0	16	17	18	19	20	0	1	0	1	?	?	?	?	?
0	21	22	23	24	25	0								
0	0	0	0	0	0	0								

$$\text{输出size} = \text{输入size} - \text{卷积核size} + 1$$

公式：

$$1. \text{ 池化层公式: } N_f = \frac{N+2P-F}{S} + 1,$$

其中 N 是图像宽度， P 是 padding 圈数， F 是 filter 宽度， S 是步长。

2. 问题：卷积层，输入三通道输出 192 通道，卷积核大小 3×3 ，该卷积层多少个参数？

答：(三通道 \times (3×3)) \times 192 = 5184 个参数

即：卷积核面积 \times 输入通道数 \times 输出通道数 = 卷积层参数

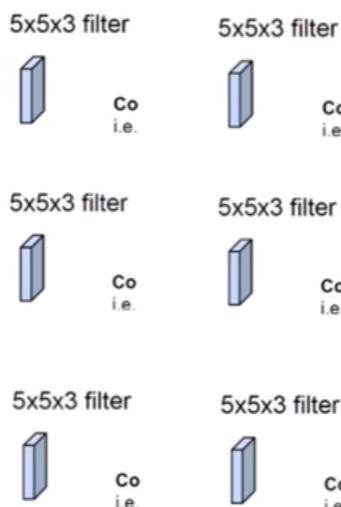
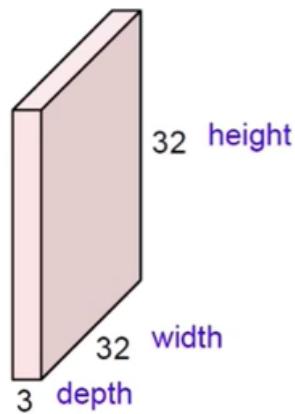
卷积——处理多通道

多通道图像用多通道卷积核，每个通道之间不共享参数。

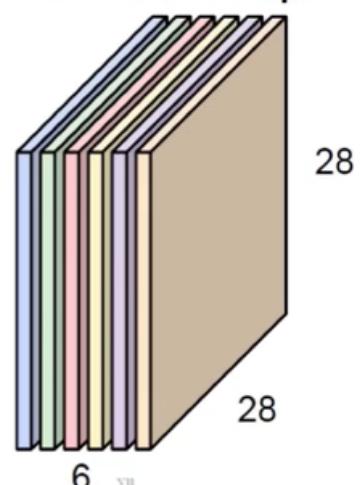
多个卷积核

卷积——多个卷积核

32x32x3 image



activation maps



? : 通道(channel)

多个卷积核提取不同的特征；

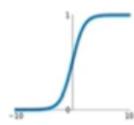
2. 激活函数

卷积神经网络

激活函数

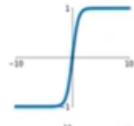
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



tanh

$$\tanh(x)$$



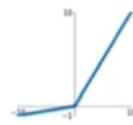
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

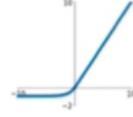


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



5	6	-2
-10	0	4
9	-1	7

ReLU激活

5	6	0
0	0	4
9	0	7

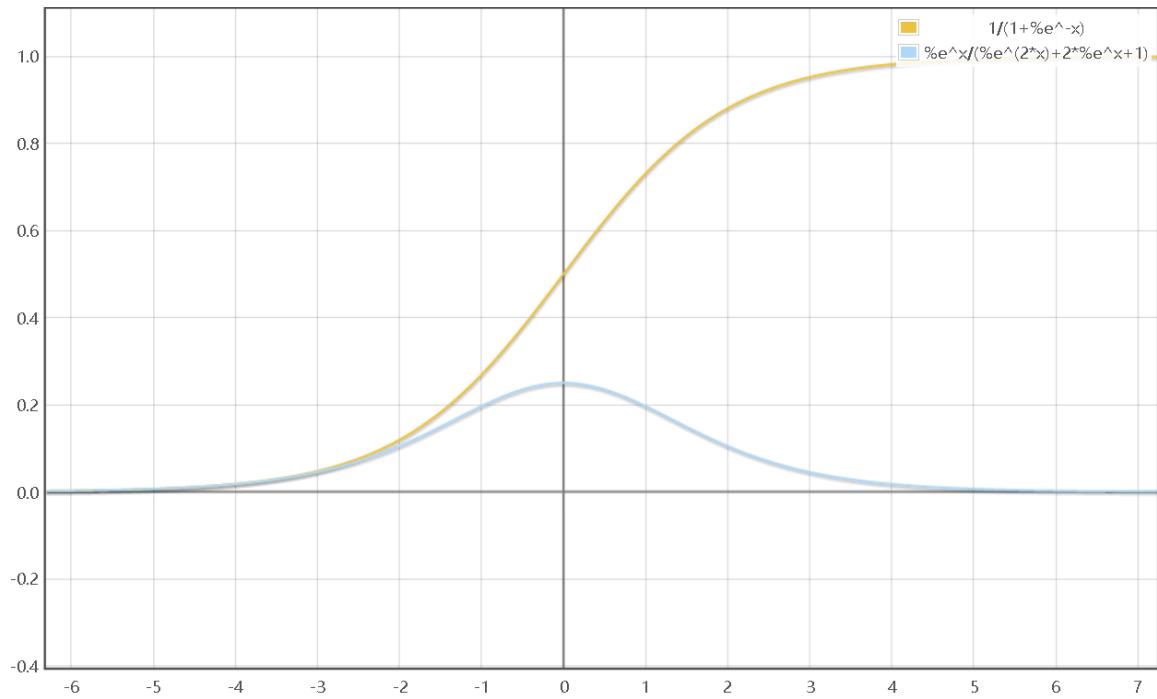
特点：单调性、非线性函数

单调性保证，输入和输出成正比。

为什么需要非线性激活函数，因为这两个线性函数的组合本身就是线性函数，所以除非你引入非线性，否则你无法计算更有用的函数，即使你的网络层数再多也不行；除非回归问题，可以在输出层用线性激活函数。总之，不能在隐藏层使用线性激活函数。

- Sigmoid

- 输入非常大或者非常小的时候梯度消失



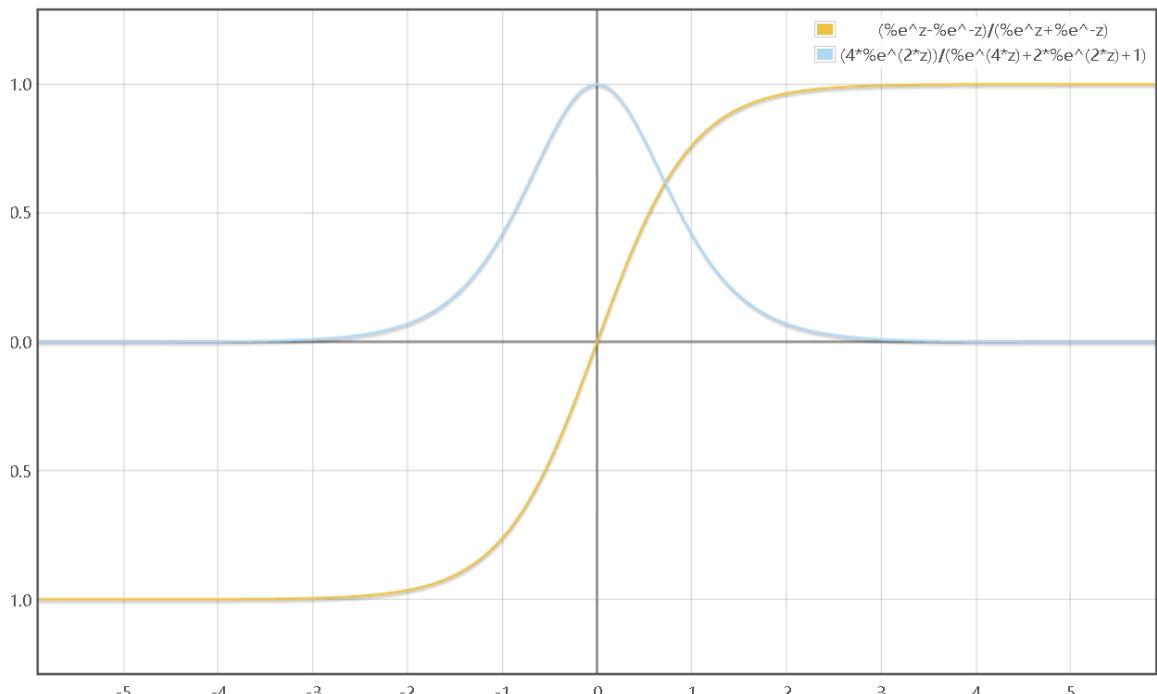
- 输出均值不为0

非0对于dl来说是不友好的，网络的学习和初始化都和均值为0有关系。

- 指数运算复杂
- 梯度消失

- tanh

- 依旧有梯度消失问题



- 运算依旧复杂（比sigmoid快）

- 均值为0——优点

- ReLU (Rectified Linear Unit)

- 梯度不会消失

- 计算量小，收敛速度快

缺点：

- 输出均值非0
- Dead ReLU

死的神经单元不会激活。TODO：查找论文

- Leaky ReLU/PReLU

$$y_i = \begin{cases} x_i & \text{if } x_i \geq 0 \\ \frac{x_i}{a_i} & \text{if } x_i < 0, \end{cases}$$

a_i 是 $(1, +\infty)$ 的随机数。

- 解决dead ReLU问题
- PReLU的 a_i 是随着数据变化的。TODO：查找论文

- RReLU

$$y_{ji} = \begin{cases} x_{ji} & \text{if } x_{ji} \geq 0 \\ a_{ji}x_{ji} & \text{if } x_{ji} < 0, \end{cases}$$

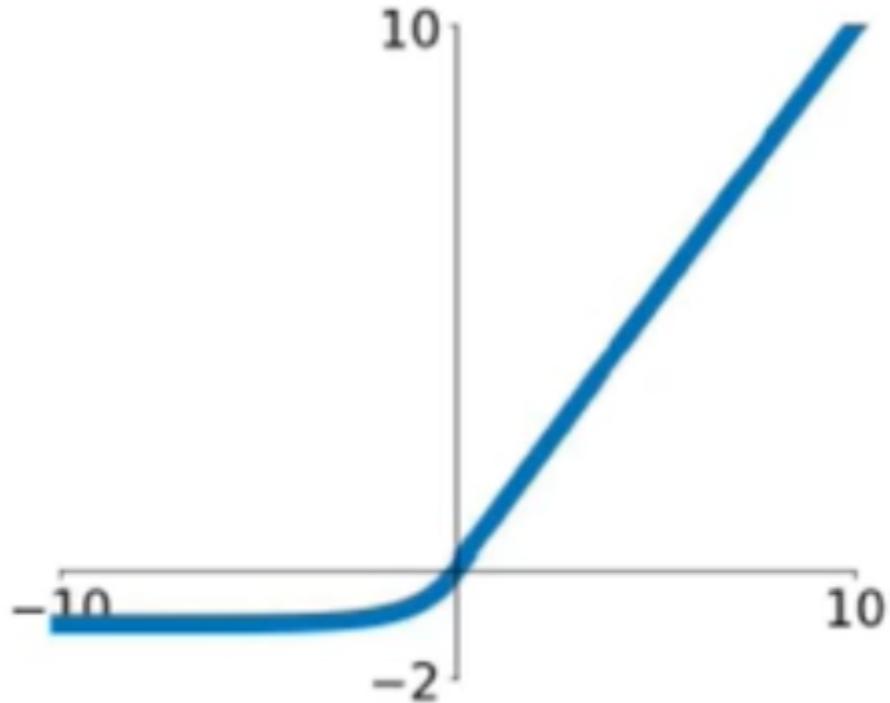
where

$$a_{ji} \sim U(l, u), l < u \text{ and } l, u \in [0, 1)$$

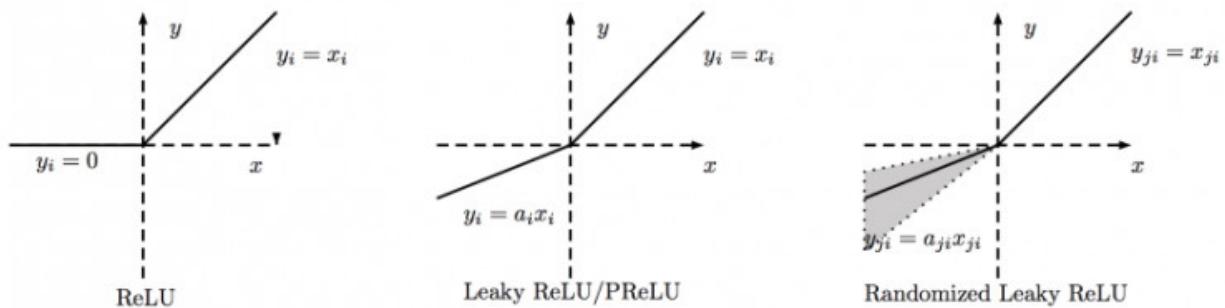
一图流，自己看公式。TODO：查找论文

- ELU

$$ELU : f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha(\exp(x) - 1) & \text{if } x \leq 0 \end{cases} \quad (1)$$



- 小于0的时候因为有指数函数exp运算，所以运算量还是较大
- 几种ReLU的比较



- maxout
- $$\max(w_1^T x + b_1, w_2^T x + b_2)$$
- ReLU的泛化，继承了各种ReLU的特点
 - 没有dead ReLU

缺点：

- 参数x2了

关于激活函数的选择

ReLU函数是一个通用的激活函数，目前在大多数情况下使用。但是，ReLU函数只能在隐藏层中使用。

用于分类器时，sigmoid函数及其组合通常效果更好。由于梯度消失问题，有时要避免使用sigmoid和tanh函数。

在神经网络层数较多的时候，最好使用ReLU函数，ReLU函数比较简单计算量少，而sigmoid和tanh函数计算量大很多。

在选择激活函数的时候可以先选用ReLU函数如果效果不理想可以尝试其他激活函数。

Summary

公式：卷积核的参数

- P=padding圈数(padding)
- S=步长(stride)
- 输出尺寸 = $(n - p)/s + 1$
 - n 是输入的大小
- 参数数目 = $kw \times kh \times Ci \times Co$
 - Ci 输入通道数
 - Co 输出通道数
 - Kw, Kh 卷积核长宽

3. 池化

- 最大值池化
取滑过区域的最大值
- 平均值池化

池化——平均值池化

输入图像				
1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Avg-pool操作
Stride=2
Kernel_size=2*2

-	-
-	-

4	6
14	16

性质：

- 使用时：一般不重叠、不补零
- 没有用于求导的参数
直接求的某个区域的最大值和平均值
- 步长、池化层大小
一般默认设为相等的，可以达到滑动不重叠的效果。

优点：

- 用于减少图像尺寸，从而减少计算量
- 一定程度解决了平移不变性

例如稍稍移动一副猫的图像，它仍然是一副猫的图像；

- 损失了空间位置精度，是计算量和精度的trade off

4. 全连接层

全连接，即上一层和下一层的神经元全部连接在一起。

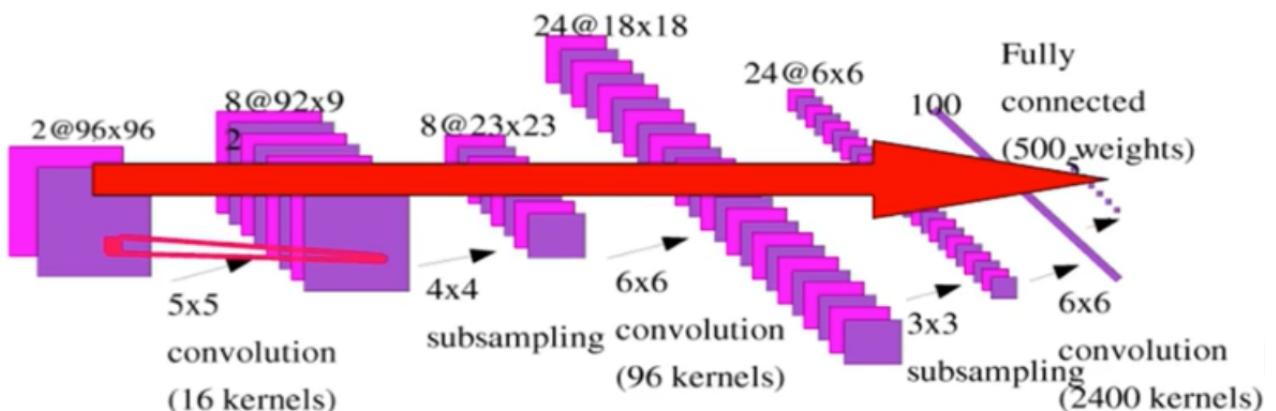
假如卷积层的输出要做全连接的话，会把二维、多通道、图像形式的卷积层展开，使得之后的输出变成一维的了，之后不能接卷积层和池化层。

- 即普通的层
- 相比于卷积层，参数数目大
- 参数数目 = $C_i \times C_o$
 - C_i 、 C_o 即输入和输出的通道数

5. CNN结构

卷积神经网络结构

◆ 卷积神经网络=卷积层+池化层+全连接层



因为输出用到了全连接层，不是图像形式输出的，所以去掉即可输出图像。

模型进化

- 更深更宽-AlexNet,VGGNet
- 不同的模型结构-VGG到InceptionNet/ResNet
- 优势组合 InceptionResNet

- NN自我学习：强化学习，不依靠人为设计网络结构
- 实用角度的：MobileNet

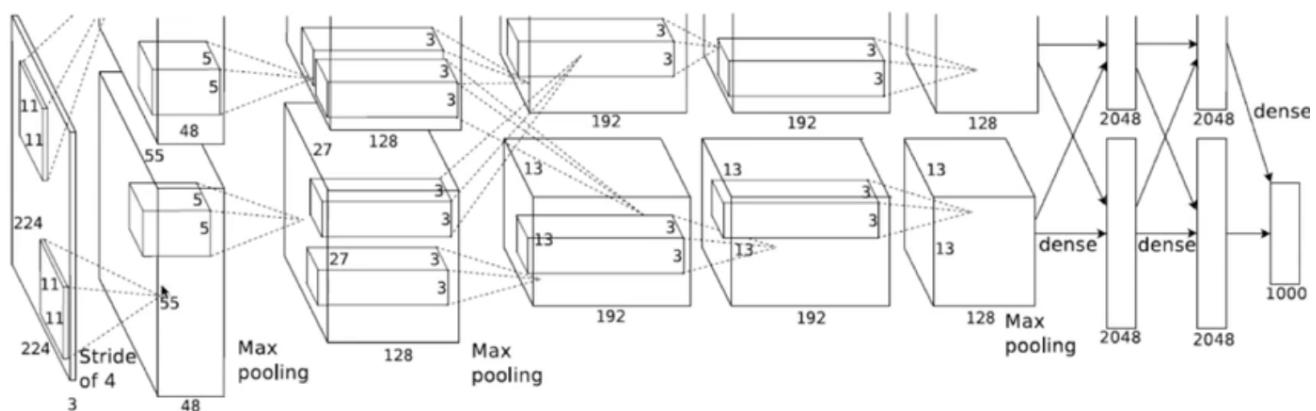
AlexNet

2012年推出，在ImageNet LSVRC-2010。

结构：

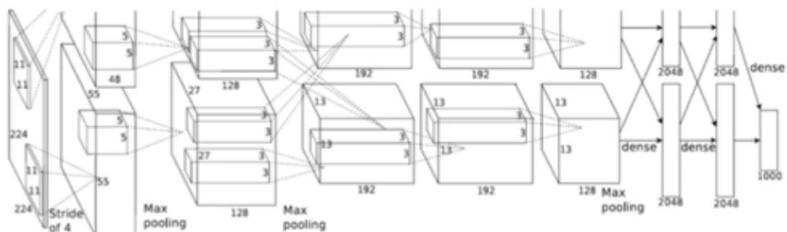
AlexNet

网络结构



AlexNet

网络结构



◆ 第一个卷积层

- ◆ 输入 $224 * 224$
- ◆ Stride = 4 , 卷积核 $11*11$
- ◆ 输出大小=(输入大小-卷积核+padding)/stride+1 = 55
- ◆ 参数数目= $3*(11*11)*96=35k$

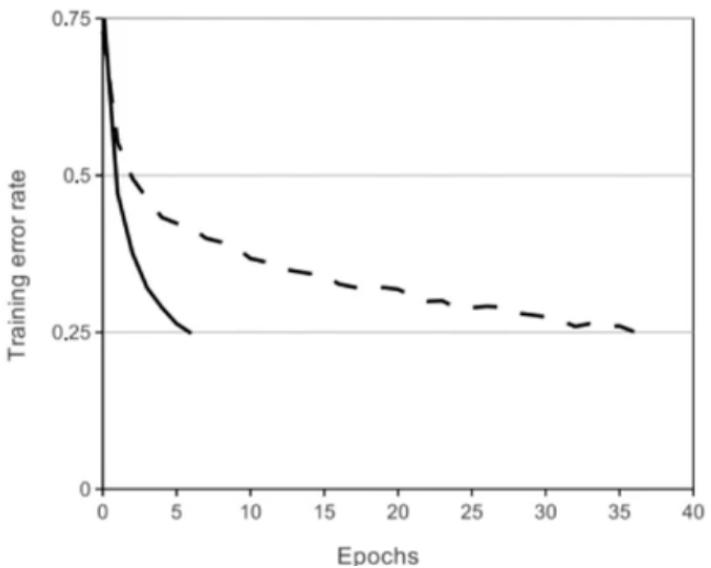
$(224-11+\text{padding}) / 4 + 1$, 因为 $213/4$ 不能整除, 可以加padding, 也可以减小图大小, 那么padding=3。

trick:

- 首次使用Relu函数，高效

网络结构

◆ 首次使用Relu



- 2个GPU并行
- 1、2、5个卷积层使用max pooling

其实也可以不用这个，可以改变步长，使得高效一点。

- 全连接层之间使用**dropout**

dropout: 随机把神经单元砍掉一半

为什么用在全连接层：全连接层参数占很多，容易过拟合，那么减少参数，以减少过拟合。

原理解释：

1. 组合解释

- 每次dropout之后的网络，都相当于一个子网络
- 最后的结果就是多个子网络结合

2. 动机解释

- 消除了神经单元之间组合记忆数据的依赖，增加了泛化能力

3. 数据解释

- 相当于增加数据，即数据增强，增加了泛化能力

其他细节

网络结构-其他细节

- ◆ 数据增强，图片随机采样
 - ◆ [256, 256] 采样 [224, 224]
- ◆ Dropout=0.5
- ◆ Batch size=128
- ◆ SGD momentum = 0.9
- ◆ Learning rate = 0.01, 过一定次数降低为1/10
- ◆ 7个CNN做ensemble： 18.2%->15.4%

VGGnet

参考：<https://www.jianshu.com/p/3510872fb186>

ImageNet Challenge2014

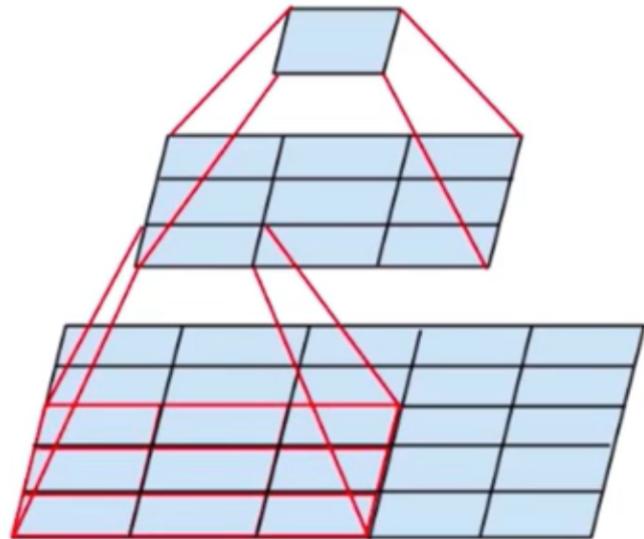
物体检测no.1，分类no.2

Summary:

- 更深
- 经验：多使用 3×3 的卷积层
 - 2个 3×3 的卷积层可以看作一层 5×5 的卷积层
 - 3个 3×3 的卷积层可以看作一层 7×7 的卷积层
- **1x1卷积层**可看作非线性变换
- 每经过一个pooling层，通道数目翻倍
 - 信息丢失没有那么多

视野域：

- 2个 3×3 =1个 5×5



- 2层比1层多一次非线性变换

- 卷积层参数降低28%

假设输入和输出通道数一样，那么：

$$\frac{2 \times 2 - 5 \times 5}{5 \times 5} = 0.28 = 28\%$$

- **1x1卷积核**

又称Network in Network

降维，可以加多层的情况下不损失信息。

网络结构：

从11-19层

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 LRN	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 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
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
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
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: Number of parameters (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

训练trick：

- 逐步训练：先训练A-再训练A-LRN、B等
- 图像尺度缩放：
 - 不同的图像尺度训练多个模型，然后做ensemble
 - 使用随机尺度的图像训练模型

ResNet

ILSVRC 2015, 分类比赛的no1

参考：<https://www.jianshu.com/p/93990a641066>

上章说到VGG在19层是最佳状态，为了可以让nn更加deep，我们假设：深层网络更难优化，而非深层网络学不到东西，所以：

- 深层网络至少可以和浅层网络持平
- 在deep nn里，加一层恒等变换 $y=x$ ，误差相较deep nn不会增加

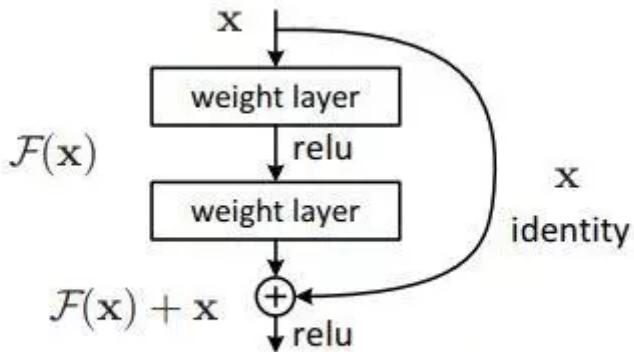


Figure 2. Residual learning: a building block.

模型结构：

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				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	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

大体结构是：

- 一个普通的卷积层， $7 \times 7, 64, \text{stride } 2$
- 再经过一个 3×3 的maxpooling,stride 2
- 残差结构
- 无fc，经过softmax输出1000维向量。

为什么没有fc：

对于一个更深的nn来说，卷积层已经有很多参数了，能学到一定的模式，加入fc可能会导致过拟合，所以去掉了fc。

- 残差结构因为恒等变换的存在，使得nn需要学习的知识变少，容易学习。
- 残差结构使得每一层的数据分布接近，容易学习。

InceptionNet

参考: <https://www.jianshu.com/p/57cccc799277>

<https://www.jianshu.com/p/d6ca52105cb5>

<https://www.jianshu.com/p/006248a3fd7f>

即GoogleNet; ILSVRC 2014, no1;

在此之前经典的CNN模型像LeNet/Alexnet/VGG等无不是一个模子即使用Conv/Pool/Normalization/Activation等层来不断累积而成。模型对数据集概率分布的表达能力则往往通过单纯增加模型的深度（层数）或宽度（层的channels数）来提高（当然这也亦是当下深度学习领域的共识）。但这样进行网络设计一般会带来大量的计算开销，因为每一层channels数目的增加都会随着层深而指数级增加，这大大地限制了模型的实际应用。

背景:

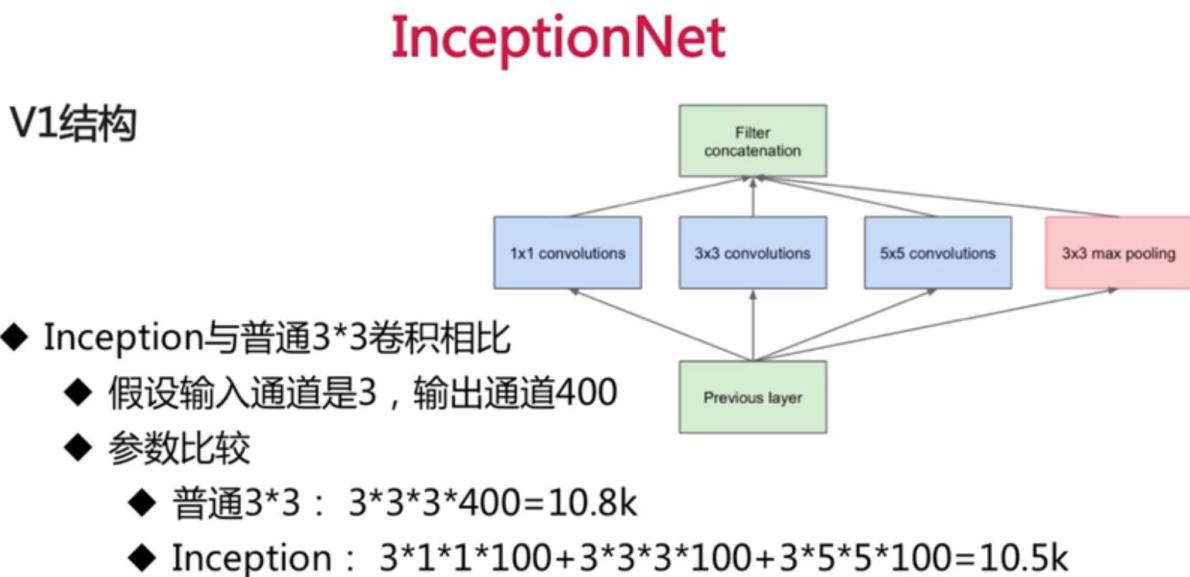
此时deepnn遇到的问题:

- 更深的网络容易过拟合
- 更深的网络有更大的计算量
 - 稀疏网络虽然减少了参数但没有减少计算量且当下用于CPU/GPU上的一些高效加速库多是在密集计算上进行优化的，同时也会带来较多的cache miss及内存地址非连续搜索的overhead.

最终,GoogleNet团队选择了中间路线,即使用密集计算子结构组合成的系数模块来用于特征提取及表达.

结构:

- v1



分组卷积：

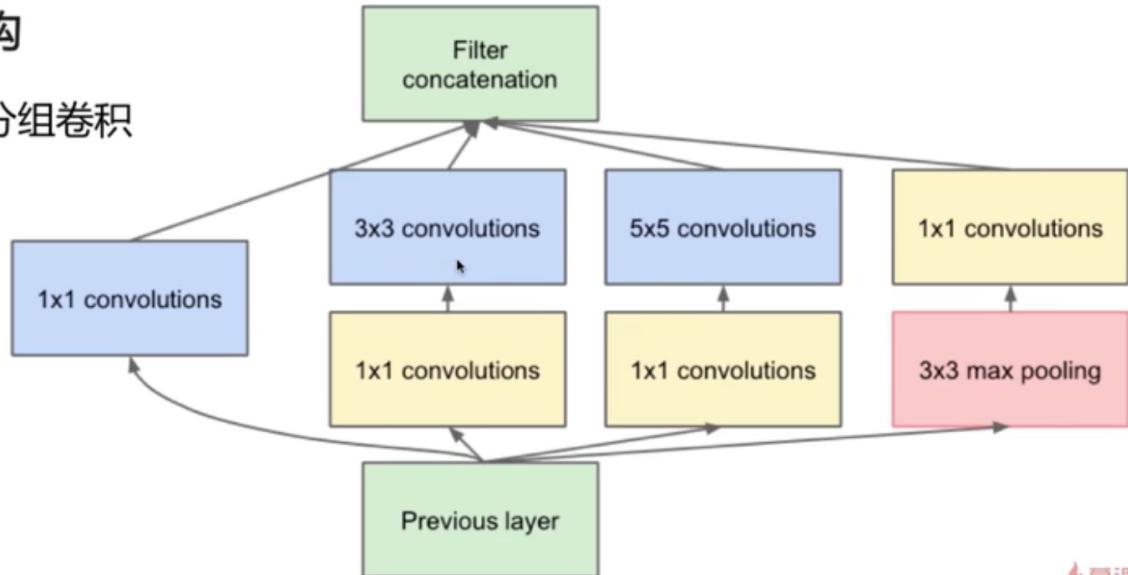
- 把卷积核分组处理，一层可以使用多种卷积核，可以取到不同的featureMap；
- 不同组之间特征不共享，减少计算量。

计算量的计算：

对上图优化：

V1结构

◆ 分组卷积



增加1x1卷积核,对Previous Layer降维.

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								

Table 1: GoogLeNet incarnation of the Inception architecture

- V2

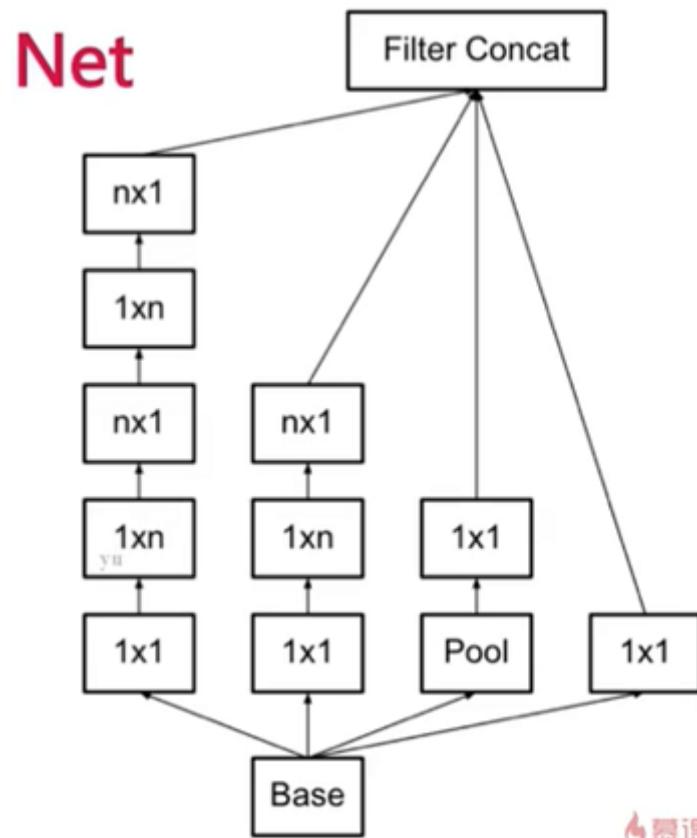
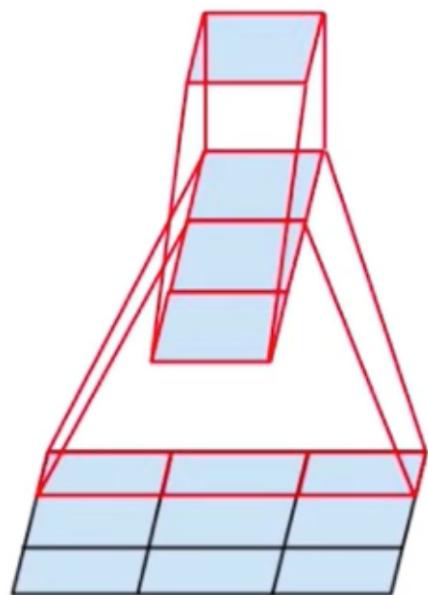
3x3换5x5,类似VGG

- V3

$$3 \times 3 = 3 \times 1 + 1 \times 3$$

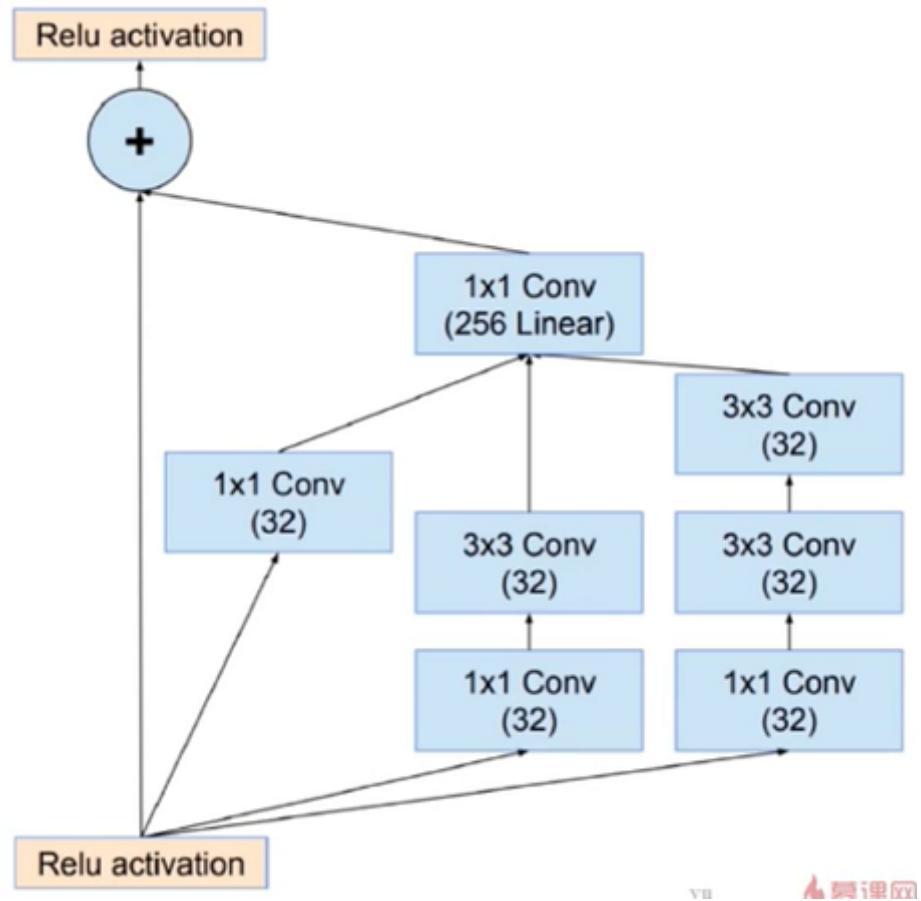
V3结构

- ◆ 3x3不是最小卷积
 - ◆ $3 \times 3 = 1 \times 3$ 和 3×1
 - ◆ 参数降低33%



🔥 热课

- V4



viii

课件

MobileNet

参考: <https://www.jianshu.com/p/7f77faf1776d>

结构:

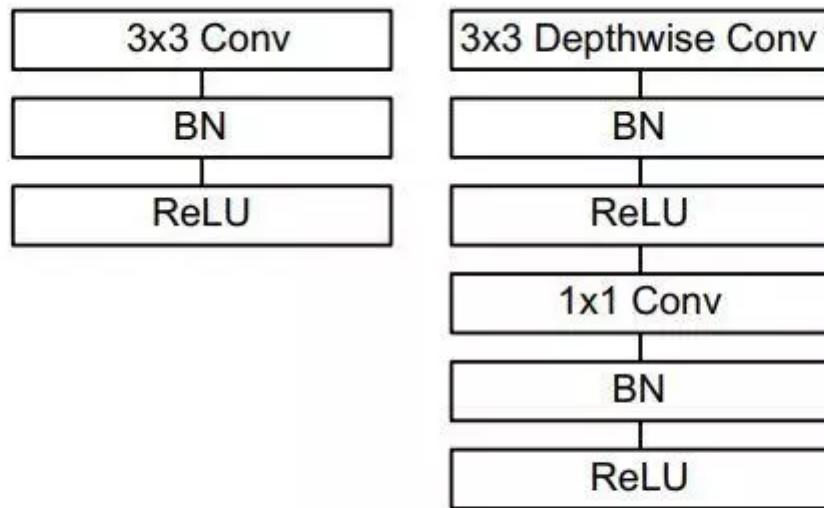


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

其中,有BN批归一化,和Depthwise Conv.

- Depthwise Conv

模型结构

- ◆ 引入深度可分离卷积
 - ◆ 回顾Inception
 - ◆ 分到极致

