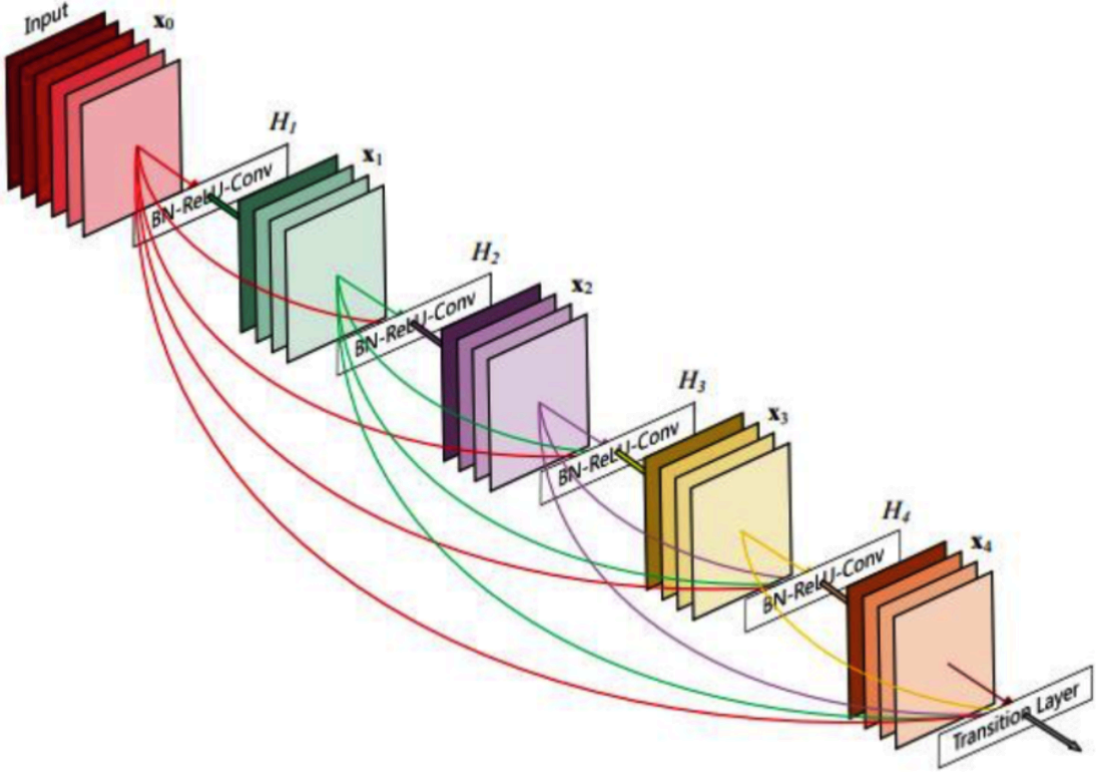
开课吧
kaikeba.com

CNN Arch 2

Allen . Huang

开课吧
kaikeba.com

DenseNet



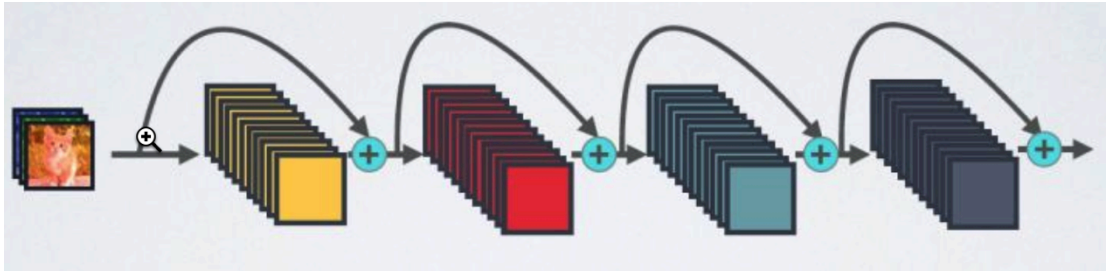
开课吧
kaikeba.com

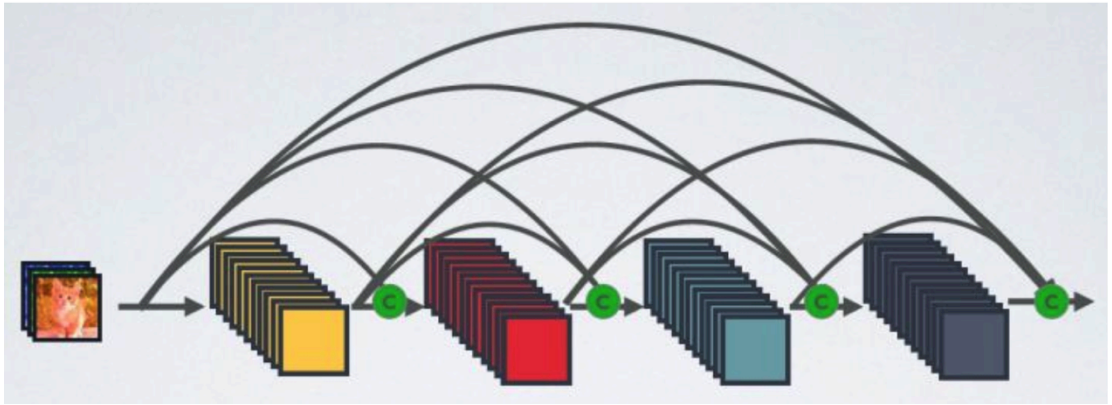
背景

作为CVPR2017年的Best Paper, DenseNet脱离了加深网络层数(ResNet)和加宽网络结构(Inception)来提升网络性能的定式思维,从特征的角度考虑,通过特征重用和旁路(Bypass)设置,既大幅度减少了网络的参数量,又在一定程度上缓解了gradient vanishing问题的产生.结合信息流和特征复用的假设,DenseNet当之无愧成为2017年计算机视觉顶会的年度最佳论文.

DenseNet

相比ResNet，DenseNet提出了一个更激进的密集连接机制：即互相连接所有的层，具体来说就是每个层都会接受其前面所有层作为其额外的输入





开课吧
kaikeba.com

DenseNet的优点

DenseNet作为另一种拥有较深层数的卷积神经网络,具有如下优点:

(1) 相比ResNet拥有更少的参数数量.

(2) 旁路加强了特征的重用.

(3) 网络更易于训练,并具有一定的正则效果.

(4) 缓解了gradient vanishing和model degradation的问题.

开课吧
kaikeba.com

ResNet与DenseNet的运算对比

·ResNet中，每个层与前面的某层（一般是2~3层）短路连接在一起，连接方式是通过**元素级相加**

·在DenseNet中，每个层都会与前面所有层在channel维度上**叠加（concat）在一起**，并作为下一层的输入

传统网络输出为： $x_l = H_l(x_{l-1})$

而对于ResNet，增加了来自上一层输入的identity函数： $x_l = H_l(x_{l-1}) + x_{l-1}$

在DenseNet中，会连接前面所有层作为输入： $x_l = H_l([x_0, x_1, \ldots, x_{l-1}])$

其中，上面的 H代表是非线性转化函数（non-liear transformation），它是一个组合操作，其可能包括一系列的BN(Batch Normalization)，ReLU，Pooling及Conv操作。注意这里 层与 层之间可能实际上包含多个卷积层。

开课吧
kaikeba.com

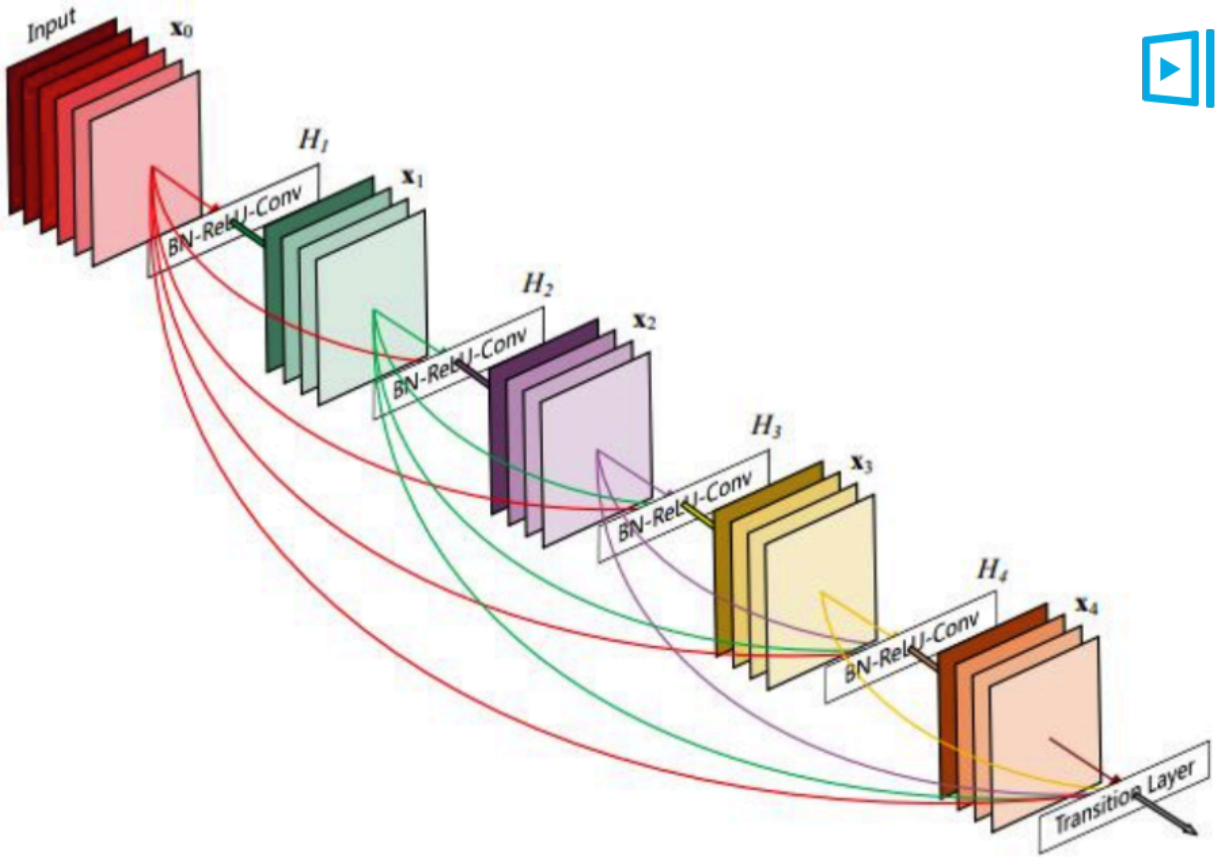


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

开课吧
kaikeba.com

由于在DenseNet中需要对不同层的feature map进行cat操作,所以需要不同层的feature map保持相同的feature size,这就限制了网络中Down sampling的实现.为了使用Down sampling,作者将DenseNet分为多个Denseblock,和TransitionLayer如下图所示

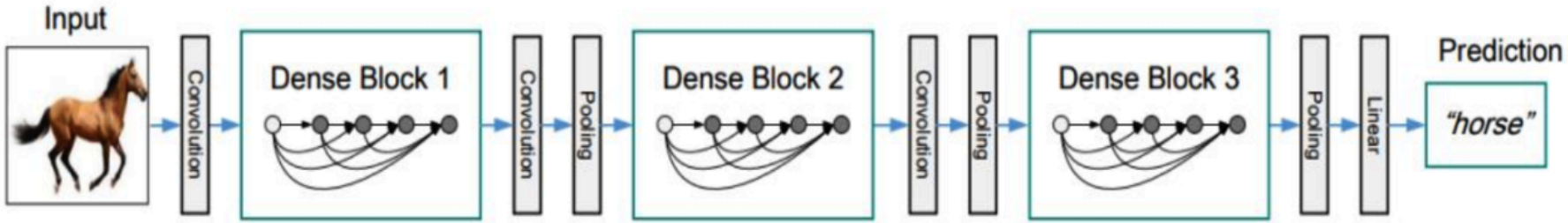


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

在同一个Denseblock中要求feature size保持相同大小,在不同Denseblock之间设置transition layers实现Down sampling, 在作者的实验中transition layer由BN + Conv(1×1) +2×2 average-pooling组成.其中Conv1x1可以调节输出的数量，2x2pooling改变长宽

增长率-GrowthRate

在Denseblock中,假设每一个非线性变换H的输出为K个feature map，那么第i层网络的输入便为$K_0+(i-1)\times K$，这里我们可以看到DenseNet和现有网络的一个主要的不同点:DenseNet可以接受较少的特征图数量作为网络层的输出

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
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] × 64
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] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

开课吧
kaikeba.com

BottleNeck Layer

虽然DenseNet接受**较少的k**,也就是feature map的数量作为输出,但由于不同层feature map之间由cat操作组合在一起,最终仍然会是feature map的channel较大而成为网络的负担.

作者在这里使用**1×1 Conv(Bottleneck)**作为特征降维的方法来降低channel数量,以提高计算效率.经过改善后的非线性变换变为**BN-ReLU-Conv(1×1)-BN-ReLU-Conv(3×3)**,使用Bottleneck layers的DenseNet被作者称为DenseNet-B.在实验中通过1x1卷积核生成4k个featuremap，这里k代表了之前的增长率

开课吧
kaikeba.com

压缩-Compression

Transition Layer用来进行DenseBlock之间的Pooling操作

θ: 代表压缩率

上层DenseBlock的输出的featuremap层数是M

下层DenseBlock的输入是 M`=⌊θM⌋

作者将使用compression且θ=0.5的DenseNet命名为DenseNet-C,将使用Bottleneck和compression且θ=0.5的DenseNet命名为DenseNet-BC

开课吧
kaikeba.com

操作细节

·DenseNet都有四个dense block，每一个block具有不同的层数

·对于3x3的卷积层，使用一个像素的零填充来保证特征图尺寸不变

·在两个dense block之间的过渡层中，2x2的平均池化层之前进行1x1的卷积

·在最后一个dense block之后，使用全局平均池化和softmax分类器。

开课吧
kaikeba.com

DenseNet的实验表现

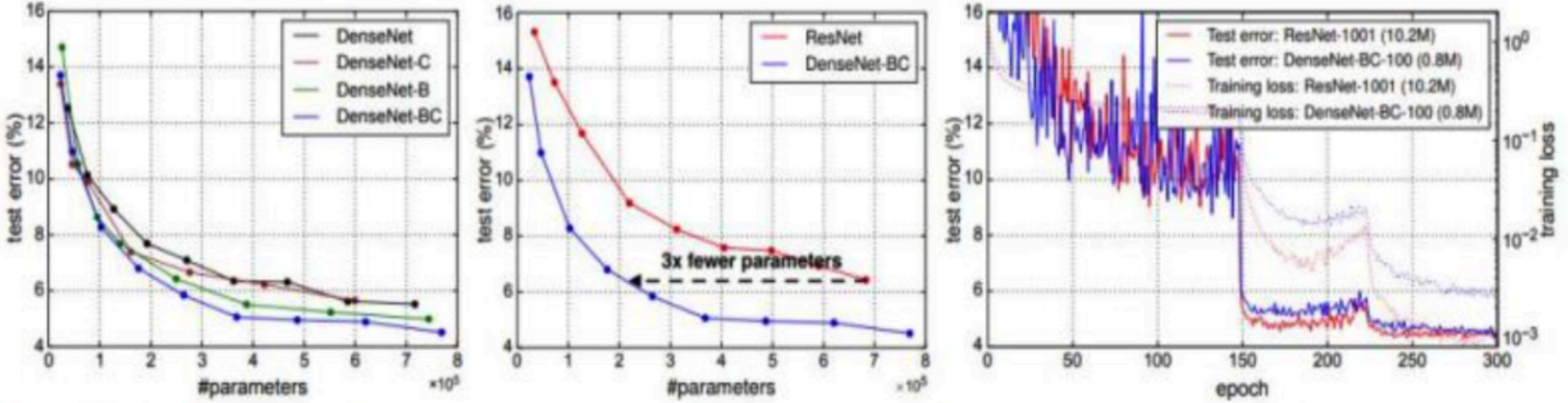


Figure 4: *Left:* Comparison of the parameter efficiency on C10+ between DenseNet variations. *Middle:* Comparison of the parameter efficiency between DenseNet-BC and (pre-activation) ResNets. DenseNet-BC requires about 1/3 of the parameters as ResNet to achieve comparable accuracy. *Right:* Training and testing curves of the 1001-layer pre-activation ResNet [12] with more than 10M parameters and a 100-layer DenseNet with only 0.8M parameters.

开课吧
kaikeba.com

结果

Method	Depth	Params	C10	C10+	C100	C100+	SVHN
Network in Network [22]	-	-	10.41	8.81	35.68	-	2.35
All-CNN [31]	-	-	9.08	7.25	-	33.71	-
Deeply Supervised Net [20]	-	-	9.69	7.97	-	34.57	1.92
Highway Network [33]	-	-	-	7.72	-	32.39	-
FractalNet [17]	21	38.6M	10.18	5.22	35.34	23.30	2.01
with Dropout/Drop-path	21	38.6M	7.33	4.60	28.20	23.73	1.87
ResNet [11]	110	1.7M	-	6.61	-	-	-
ResNet (reported by [13])	110	1.7M	13.63	6.41	44.74	27.22	2.01
ResNet with Stochastic Depth [13]	110	1.7M	11.66	5.23	37.80	24.58	1.75
	1202	10.2M	-	4.91	-	-	-
Wide ResNet [41]	16	11.0M	-	4.81	-	22.07	-
	28	36.5M	-	4.17	-	20.50	-
with Dropout	16	2.7M	-	-	-	-	1.64
ResNet (pre-activation) [12]	164	1.7M	11.26*	5.46	35.58*	24.33	-
	1001	10.2M	10.56*	4.62	33.47*	22.71	-
DenseNet ($k = 12$)	40	1.0M	**7.00**	5.24	**27.55**	24.42	1.79
DenseNet ($k = 12$)	100	7.0M	**5.77**	**4.10**	**23.79**	**20.20**	1.67
DenseNet ($k = 24$)	100	27.2M	**5.83**	**3.74**	**23.42**	**19.25**	**1.59**
DenseNet-BC ($k = 12$)	100	0.8M	**5.92**	4.51	**24.15**	22.27	1.76
DenseNet-BC ($k = 24$)	250	15.3M	**5.19**	**3.62**	**19.64**	**17.60**	1.74
DenseNet-BC ($k = 40$)	190	25.6M	-	**3.46**	-	**17.18**	-

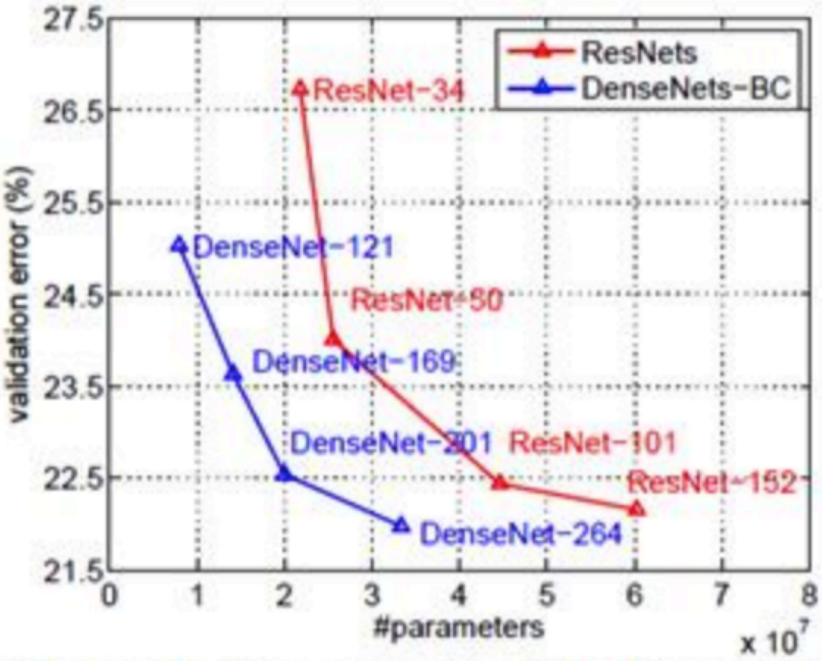
Table 2: Error rates (%) on CIFAR and SVHN datasets. k denotes network's growth rate. Results that surpass all competing methods are **bold** and the overall best results are **blue**. "+" indicates standard data augmentation (translation and/or mirroring). $*$ indicates results run by ourselves. All the results of DenseNets without data augmentation (C10, C100, SVHN) are obtained using Dropout. DenseNets achieve lower error rates while using fewer parameters than ResNet. Without data augmentation, DenseNet performs better by a large margin.

开课吧
kaikeba.com

ImageNet实验结果

Model	top-1	top-5
DenseNet-121	25.02 / 23.61	7.71 / 6.66
DenseNet-169	23.80 / 22.08	6.85 / 5.92
DenseNet-201	22.58 / 21.46	6.34 / 5.54
DenseNet-264	22.15 / 20.80	6.12 / 5.29

Table 3: The top-1 and top-5 error rates on the ImageNet validation set, with single-crop / 10-crop testing.



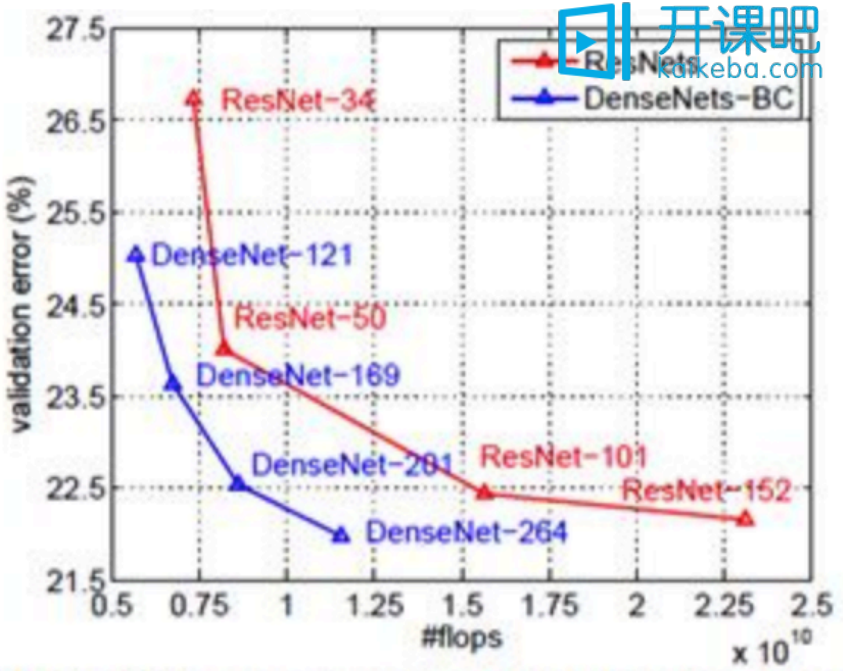


Figure 3: Comparison of the DenseNets and ResNets top-1 error rates (single-crop testing) on the ImageNet validation dataset as a function of learned parameters (*left*) and FLOPs during test-time (*right*).

模型简化性

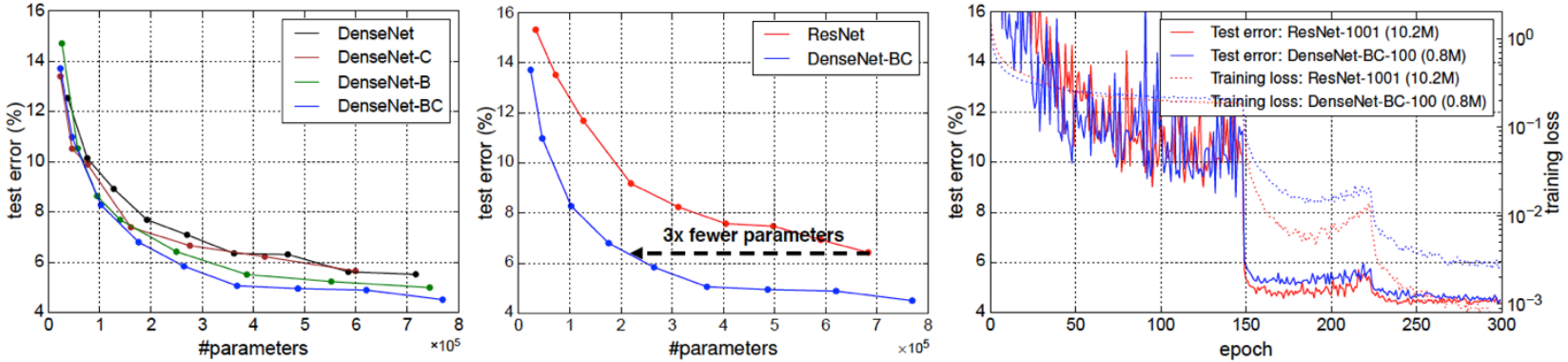


Figure 4: *Left:* Comparison of the parameter efficiency on C10+ between DenseNet variations. *Middle:* Comparison of the parameter efficiency between DenseNet-BC and (pre-activation) ResNets. DenseNet-BC requires about 1/3 of the parameters as ResNet to achieve comparable accuracy. *Right:* Training and testing curves of the 1001-layer pre-activation ResNet [12] with more than 10M parameters and a 100-layer DenseNet with only 0.8M parameters.

隐含深度监督

解释DenseNet为何拥有如此高性能的另一个原因是网络中的每一层不仅接受了原始网络中来自loss的监督,同时由于存在多个bypass与shortcut,网络的监督是多样的.Deep supervision的优势同样在deeply-supervised nets (DSN)中也被证实.(DSN中每一个Hidden layer都有一个分类器,强迫其学习一些有区分度的特征).与DSN不同的是,DenseNet拥有单一的loss function, 模型构造和梯度计算更加简易.

深层实验

DenseNets允许每一层获得之前所有层（尽管一些是通过过渡层）的特征图。我们做了一个实验来判断是否训练的网络可以重复利用这个机会。我们首先在C10+数据上训练了 的DenseNet。对于每个block的每个卷积层 ，我们计算其与 层连接的平均权重。三个dense block的热度图如图5所示

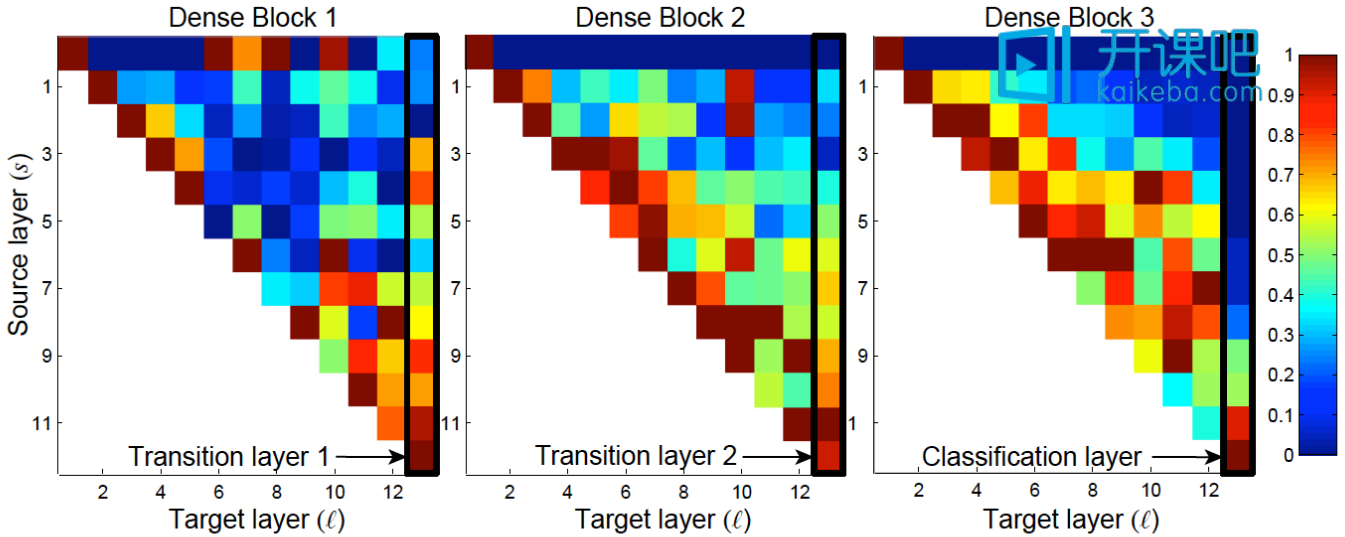


Figure 5: The average absolute filter weights of convolutional layers in a trained DenseNet. The color of pixel (s, ℓ) encodes the average $L1$ norm (normalized by number of input feature-maps) of the weights connecting convolutional layer s to ℓ within a dense block. Three columns highlighted by black rectangles correspond to two transition layers and the classification layer. The first row encodes weights connected to the input layer of the dense block.

开课吧
kaikeba.com

从上图中可以得到结论如下：

a) 一些较早层提取出的特征仍可能被较深层直接使用

b) 即使是Transition layer也会使用到之前Denseblock中所有层的特征

c) 第2-3个Denseblock中的层对之前Transition layer利用率很低,说明transition layer输出大量冗余特征.这也为DenseNet-BC提供了证据支持,既Compression的必要性.

d)最后的分类层虽然使用了之前Denseblock中的多层信息,但更偏向于使用最后几个featuremap的特征,说明在网络的最后几层,某些high-level的特征可能被产生.

开课吧
kaikeba.com

结论

我们提出了一个新的卷积网络结构，称之为稠密卷积网络（DenseNet）。它将两个相同特征图尺寸的任意层进行连接。这样我们就可以很自然的设计上百层的网络，还不会出现优化困难的问题。在我们的实验中，随着参数量的增加，DenseNets的准确率也随之提高，而且也没有出现较差表现或过拟合的现象。通过超参数的调整，该结构在很多比赛的数据上都获得了不错的结果。此外，DenseNets有更少的参数和计算量。因为我们只是在实验中调整了对于残差网络的超参数，所以我们相信通过调整更多的超参数和学习率，DenseNets的准确率还会有更大的提升。

遵循这个简单的连接规则，DenseNets可以很自然的将自身映射（identity mappings）、深度监督（deep supervision）和深度多样化（diversified depth）结合在一起。根据我们的实验来看，该结构通过对网络特征的重复利用，可以学习到更简单、准确率更高的模型。由于简化了内部表征和降低了特征冗余，DenseNets可能是目前计算机视觉领域中在卷积网络方面非常不错的特征提取器。在以后的工作中我们计划研究DenseNets下的特征迁移工作。

ResNext

综述：传统的要提高模型的准确率，都是加深或加宽网络，但是随着超参数数量的增加（比如channels数，filter size等等），网络设计的难度和计算开销也会增加。因此本文提出的 ResNeXt 结构可以在不增加参数复杂度的前提下提高准确率，同时还减少了超参数的数量

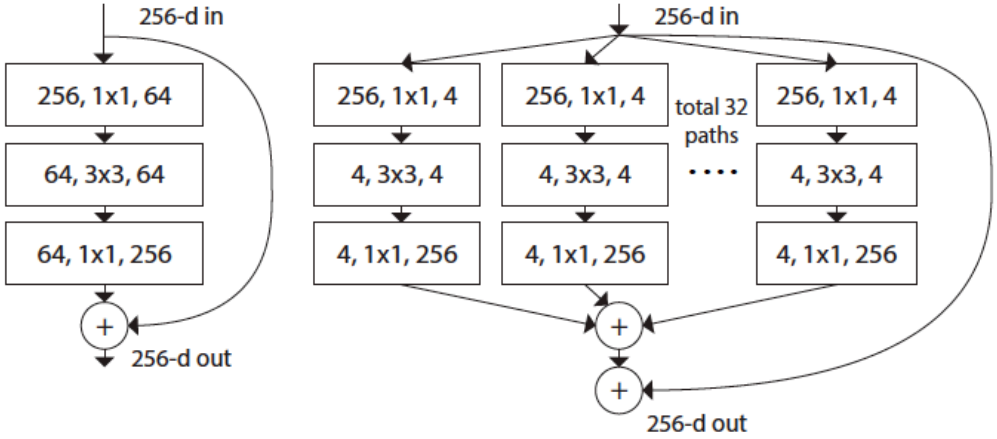
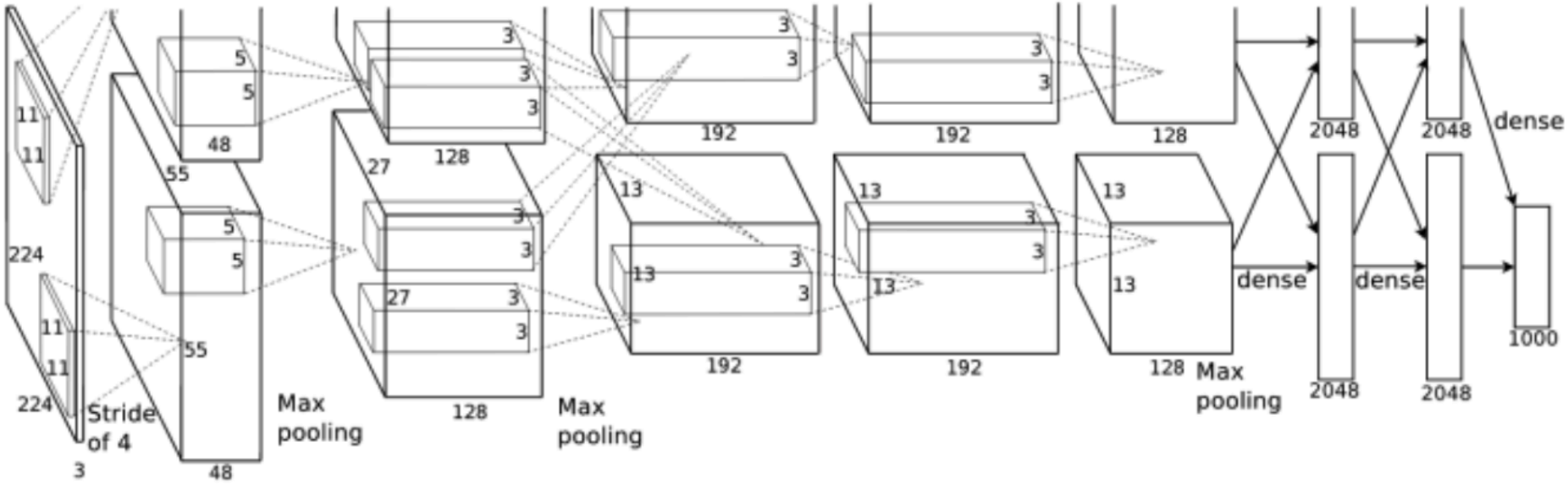


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

开课吧
kaikeba.com

Group Convolution

最早出现于AlexNet中，无法在单块GPU上运行



开课吧
kaikeba.com

分组卷积

·上一层的输出feature map有N个，即通道数channel=N

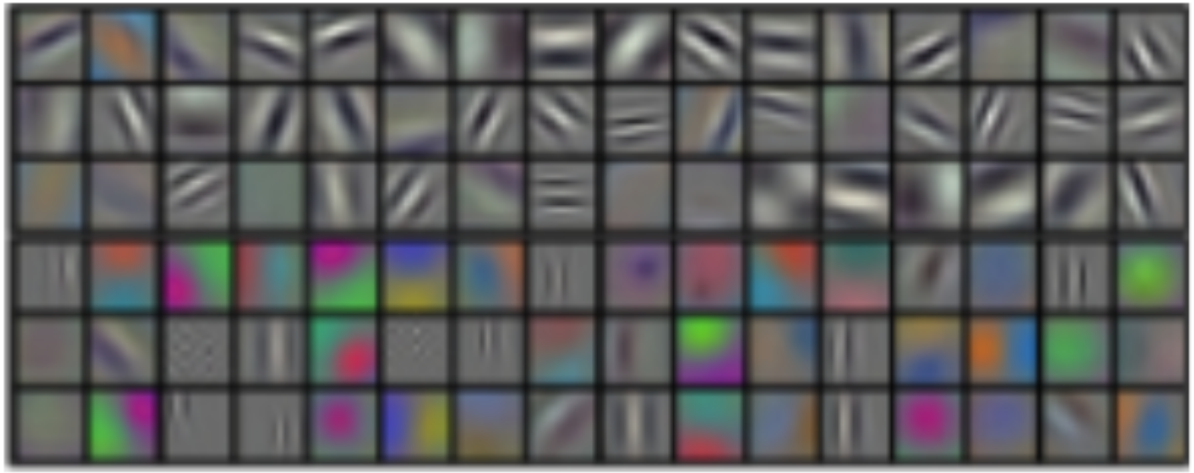
·卷积的群数目M

·每一个group对应N/M个channel，与之独立连接

·各个group卷积完成后将输出叠在一起（concatenate），作为这一层的输出channel

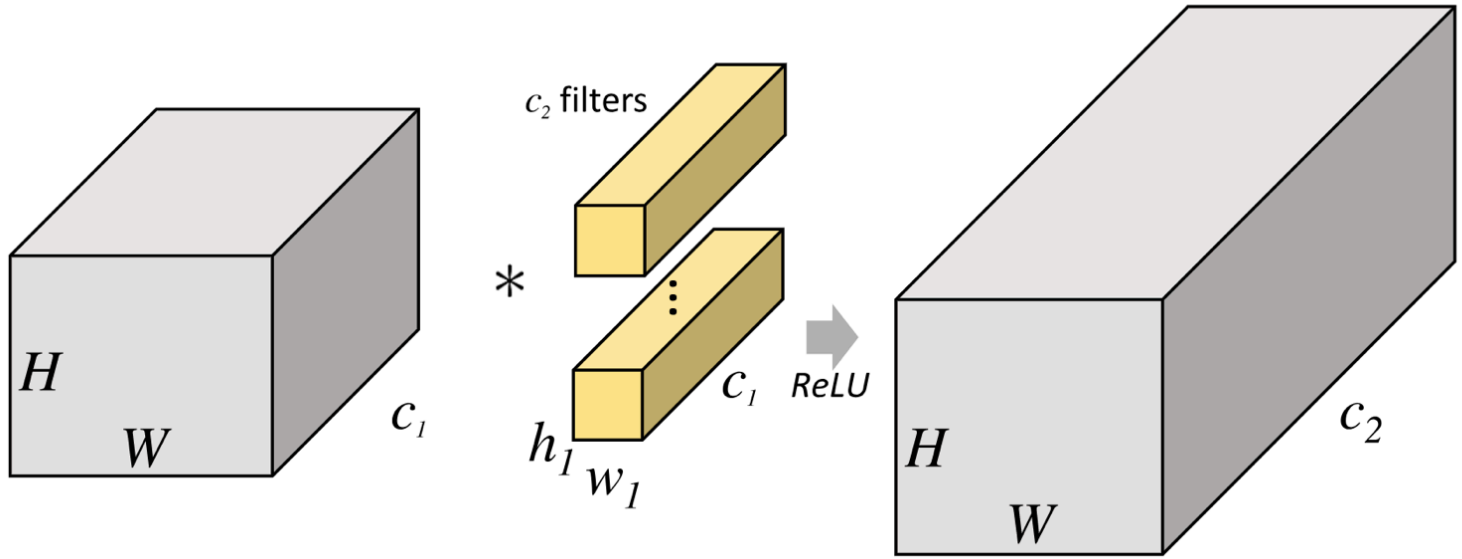
开课吧
kaikeba.com

分组卷积的上下两部分的FeatureMap



传统的卷积运算

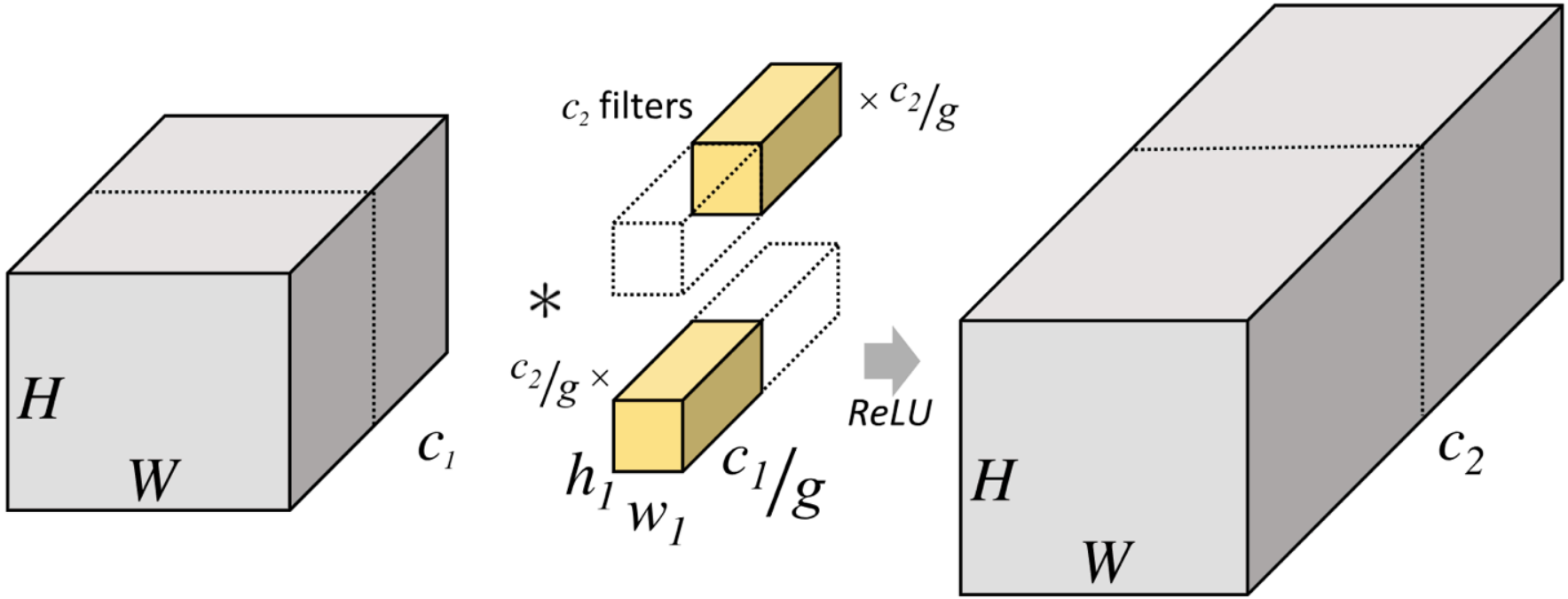
(H-h1+1)*(W-w1+1)*h1*w1*c1*c2次运算



开课吧
kaikeba.com

Group卷积的运算量

运算量，参数量是多少?



分组卷积的优缺点

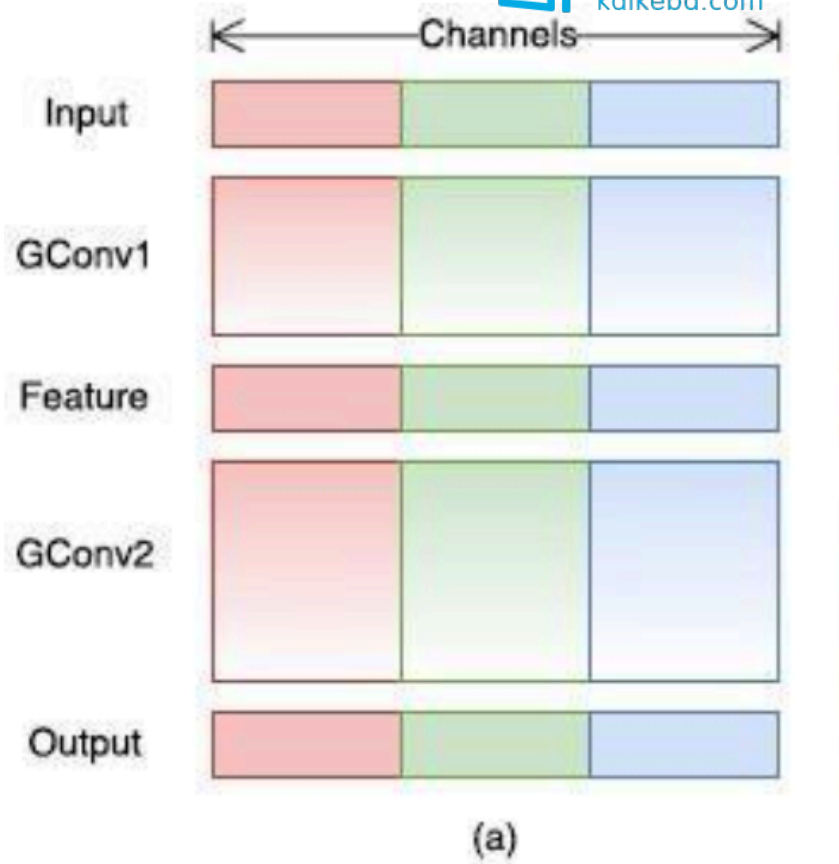
优点：

并行化、降低运算量、降低参数、Kernel对角化

缺点：

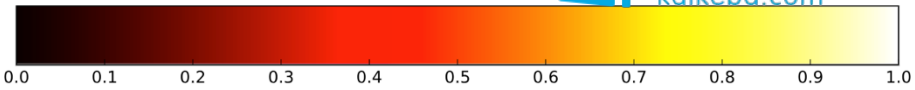
某个输出channel仅仅来自输入channel的一小部分，

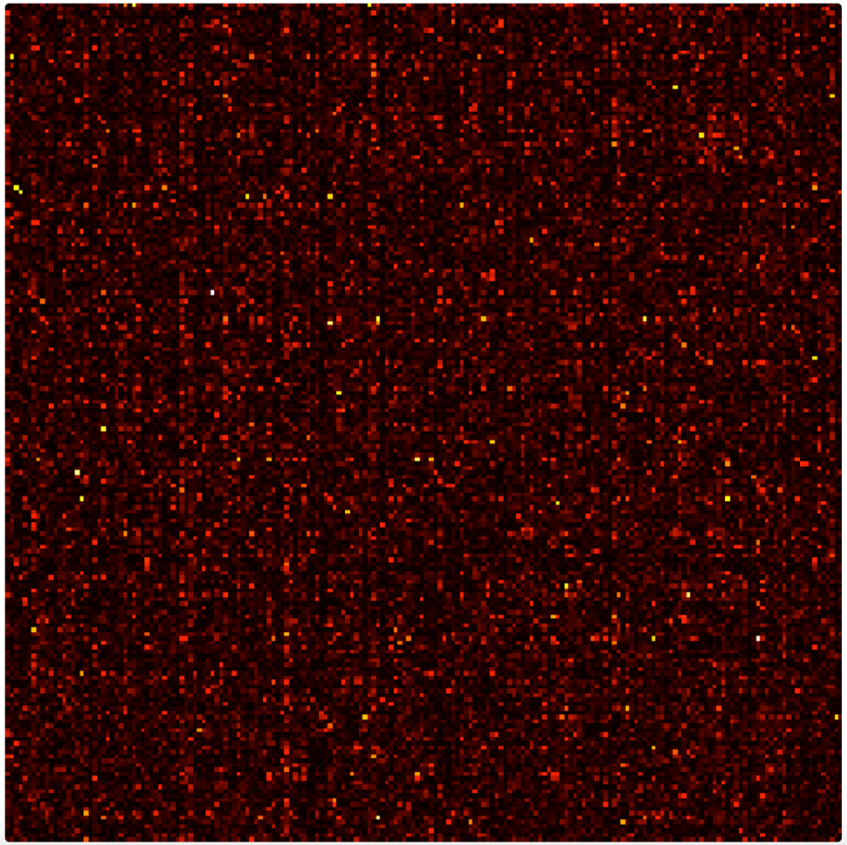
学出来的特征会非常局限

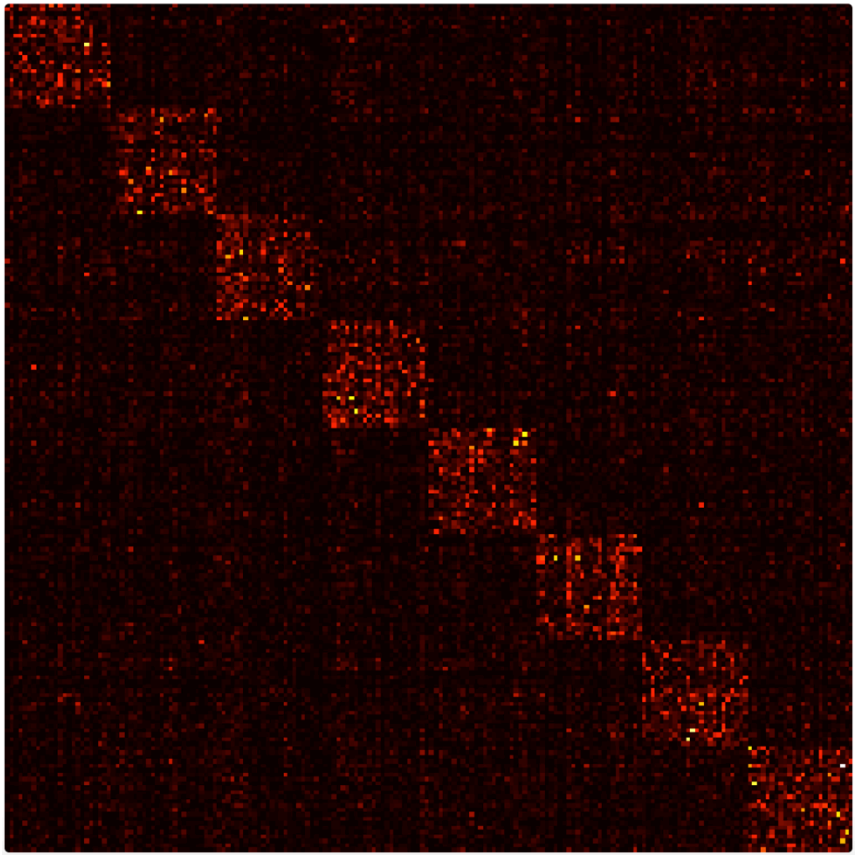


关于groupfilter的对角化

开课吧
kaikeba.com







开课吧
kaikeba.com

Group Conv

对于待搜索函数空间进行了限制

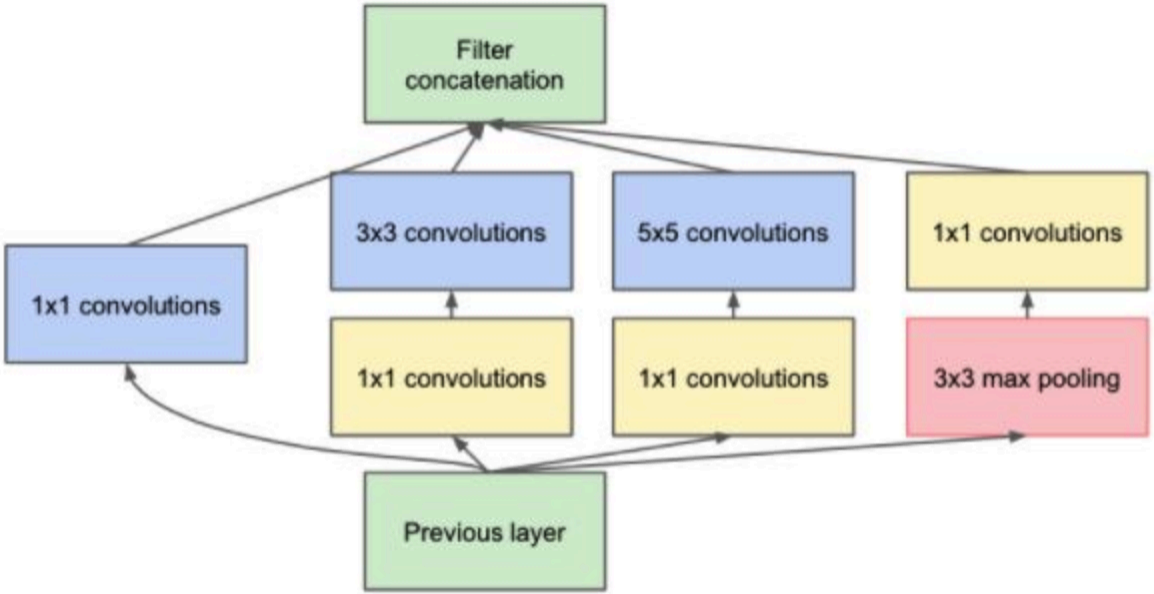
因此很难过拟合

缺点也是对于搜索函数空间进行了限制，

因此会忽略掉很多的minima，这些minima可能表现比起现有的要好得多

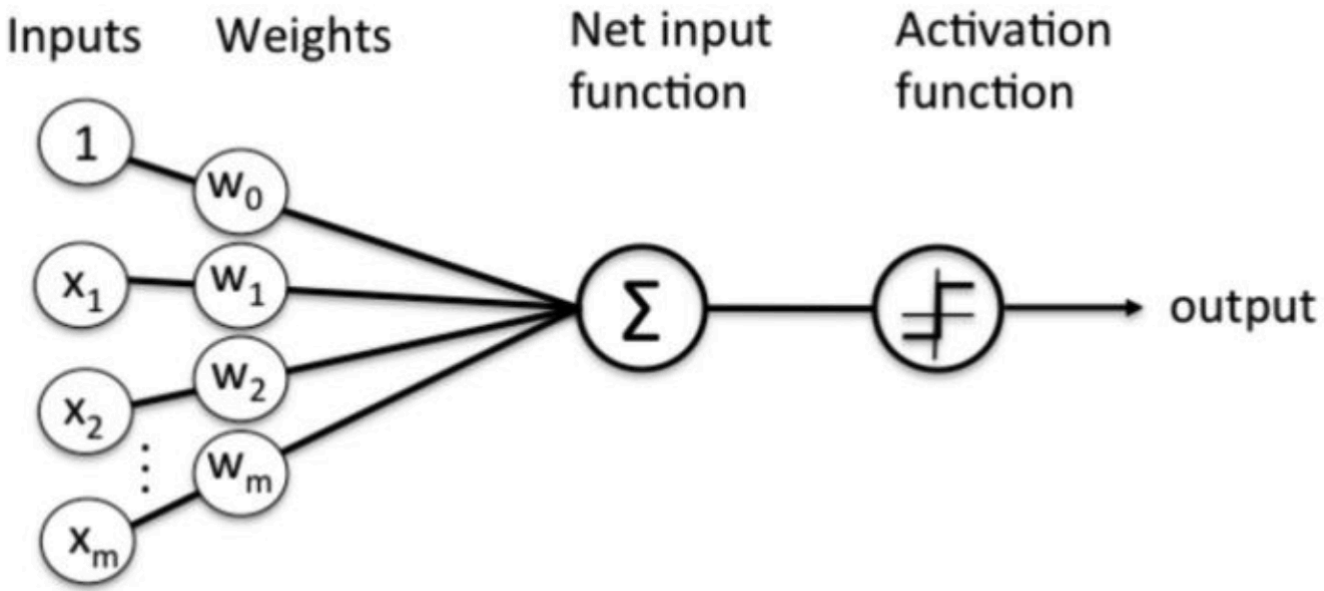
开课吧
kaikeba.com

Split-Transform-Merge



开课吧
kaikeba.com

Split-Transform-Merge



基数Cardinality

$$\mathcal{F}(\mathbf{x}) = \sum_{i=1}^{C} \mathcal{T}_i(\mathbf{x})$$

$$\mathbf{y} = \mathbf{x} + \sum_{i=1}^{C} \mathcal{T}_i(\mathbf{x})$$

上面的变换T可以是任意形式，一共有C个独立的变换，作者将C称之为基数，并且指出，基数C对于结果的影响比宽度和深度更加重要。

开课吧
kaikeba.com

基数

左边是ResNet的结构，右边是ResNext的基本结构

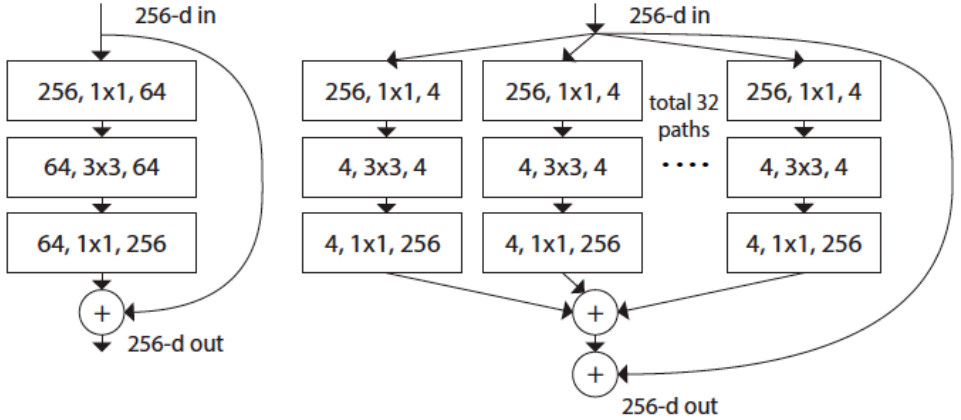


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

开课吧
kaikeba.com

基数

Our method indicates that *cardinality* (the size of the set of transformations) is a concrete, measurable dimension that is of central importance, in addition to the dimensions of width and depth. Experiments demonstrate that *increasing cardinality is a more effective way of gaining accuracy than going deeper or wider*, especially when depth and width starts to give diminishing returns for existing models.

注意C=32事实上是将输入的Channel均分成C份，并行的进行处理，之后又合并

stage	output	ResNet-50	ResNeXt-50 (32×4d)
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix}\times4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix}\times6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		$\mathbf{25.5}\times10^6$	$\mathbf{25.0}\times10^6$
FLOPs		$\mathbf{4.1}\times10^9$	$\mathbf{4.2}\times10^9$

Table 1. (**Left**) ResNet-50. (**Right**) ResNeXt-50 with a 32×4d template (using the reformulation in Fig. 3(c)). Inside the brackets are the shape of a residual block, and outside the brackets is the number of stacked blocks on a stage. "C=32" suggests grouped convolutions [24] with 32 groups. *The numbers of parameters and FLOPs are similar between these two models.*

变种

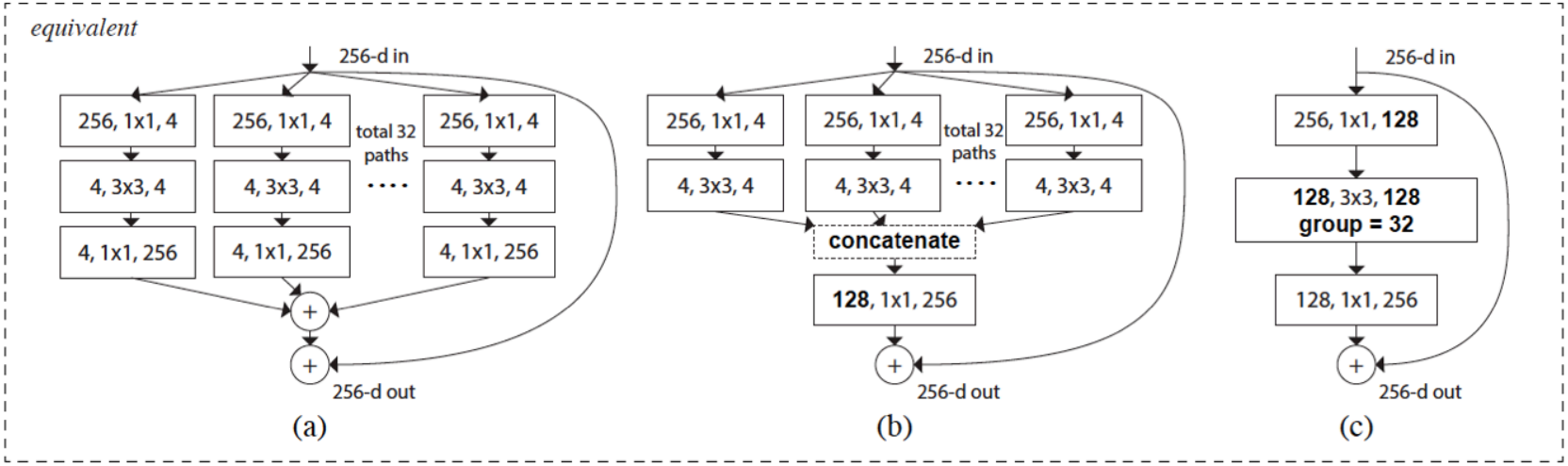


Figure 3. Equivalent building blocks of ResNeXt. **(a)**: Aggregated residual transformations, the same as Fig. 1 right. **(b)**: A block equivalent to (a), implemented as early concatenation. **(c)**: A block equivalent to (a,b), implemented as grouped convolutions [24]. Notations in **bold** text highlight the reformulation changes. A layer is denoted as (# input channels, filter size, # output channels).

实验

开课吧
kaikeba.com

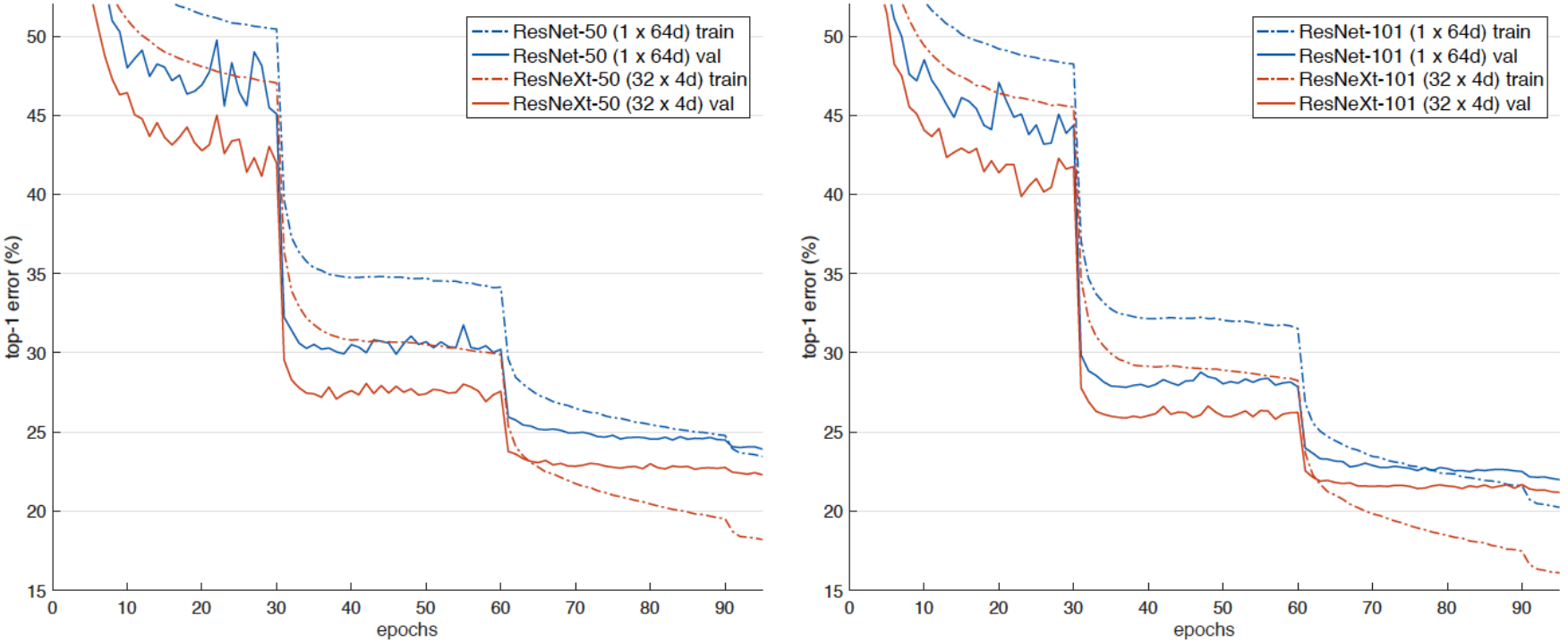


Figure 5. Training curves on ImageNet-1K. (**Left**): ResNet/ResNeXt-50 with preserved complexity (∼4.1 billion FLOPs, ∼25 million parameters); (**Right**): ResNet/ResNeXt-101 with preserved complexity (∼7.8 billion FLOPs, ∼44 million parameters).

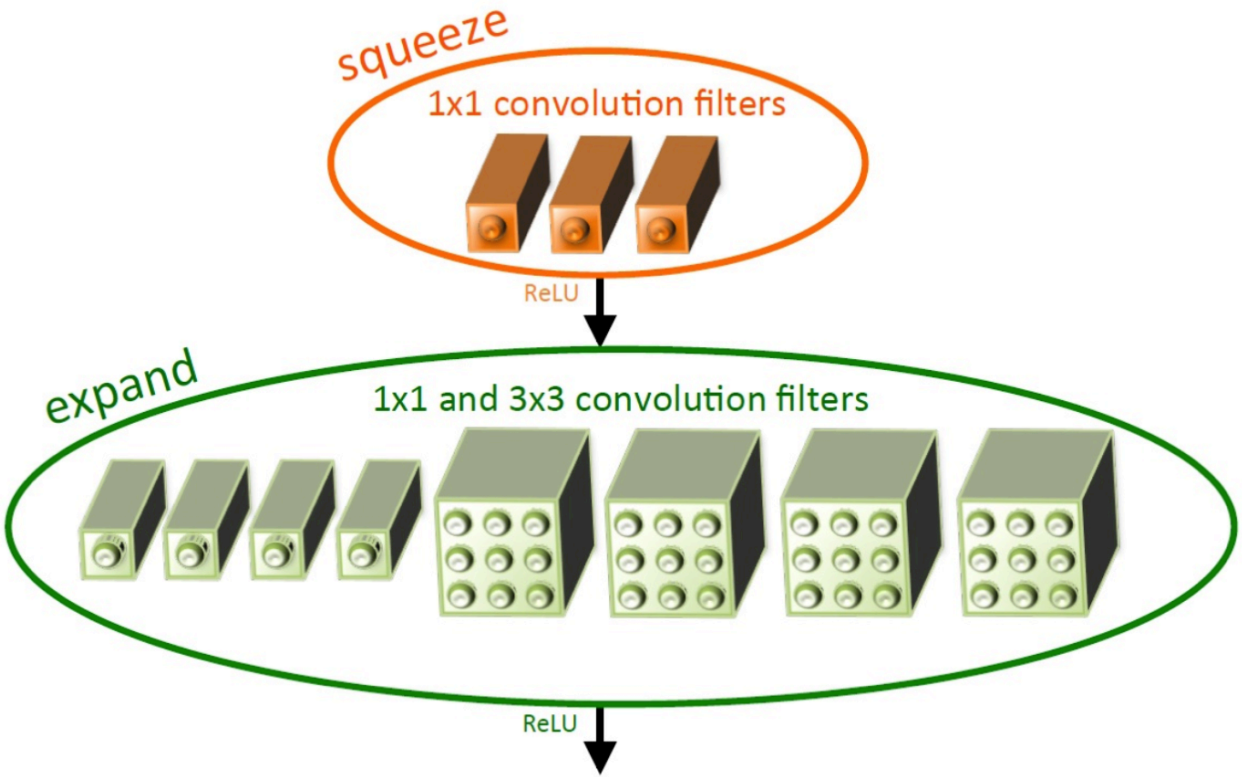
开课吧
kaikeba.com

	setting	top-1 error (%)
ResNet-50	1 × 64d	23.9
ResNeXt-50	2 × 40d	23.0
ResNeXt-50	4 × 24d	22.6
ResNeXt-50	8 × 14d	22.3
ResNeXt-50	32 × 4d	**22.2**
ResNet-101	1 × 64d	22.0
ResNeXt-101	2 × 40d	21.7
ResNeXt-101	4 × 24d	21.4
ResNeXt-101	8 × 14d	21.3
ResNeXt-101	32 × 4d	**21.2**

Table 3. Ablation experiments on ImageNet-1K. (**Top**): ResNet-50 with preserved complexity (∼4.1 billion FLOPs); (**Bottom**): ResNet-101 with preserved complexity (∼7.8 billion FLOPs). The error rate is evaluated on the single crop of 224×224 pixels.

开课吧
kaikeba.com

SqueezeNet



开课吧
kaikeba.com

SqueezeNet

提高运算效率可以从2方面入手：

·减少可学习参数的数量

·减少整个网络的计算量

开课吧
kaikeba.com

SqueezeNet

·减少模型训练和测试时候的计算量，单个step的速度更快；

·减小模型文件的大小，更利于模型的保存和传输；

·可学习参数更少，网络占用的显存更小。

开课吧
kaikeba.com

压缩策略

1. 将3x3卷积替换成1x1卷积：通过这一步，一个卷积操作的参数数量减少了9倍
2. 减少3x3卷积的通道数
3. 将降采样后置：作者认为较大的Feature Map含有更多的信息，因此将降采样往分类层移动。然而这样的操作虽然会提升网络的精度，但是它有一个非常严重的缺点：即会增加网络的计算量。

开课吧
kaikeba.com

Fire Module

一个Fire模块由Squeeze部分和Expand部分组成

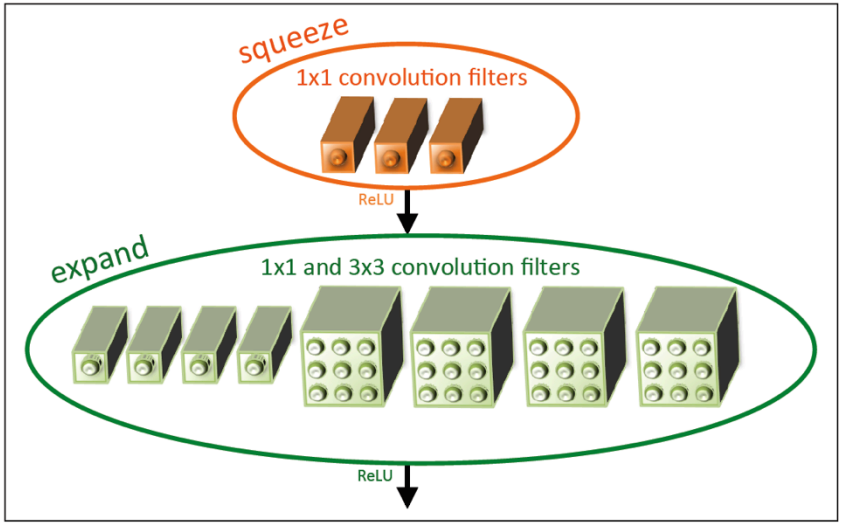


Figure 1: Microarchitectural view: Organization of convolution filters in the **Fire module**. In this example, $s_{1x1} = 3$, $e_{1x1} = 4$, and $e_{3x3} = 4$. We illustrate the convolution filters but not the activations.

开课吧
kaikeba.com

Fire Module

Squeeze部分是一组连续的1x1卷积组成，Expand部分则是由一组连续的1x1卷积和一组连续的3x3卷积cancatnate组成

Squeeze部分的1x1卷积的输出通道数是S1x1

Expand部分的3x3卷积的输出通道数是E3x3

Expand部分的1x1卷积的输出通道数是E1x1

首先输入通过Squeeze部分进行压缩，之后通过Expand部分进行分别的处理跟合并

作者建议 S1x1=E3x3/4=E1x1/4

Fire Module

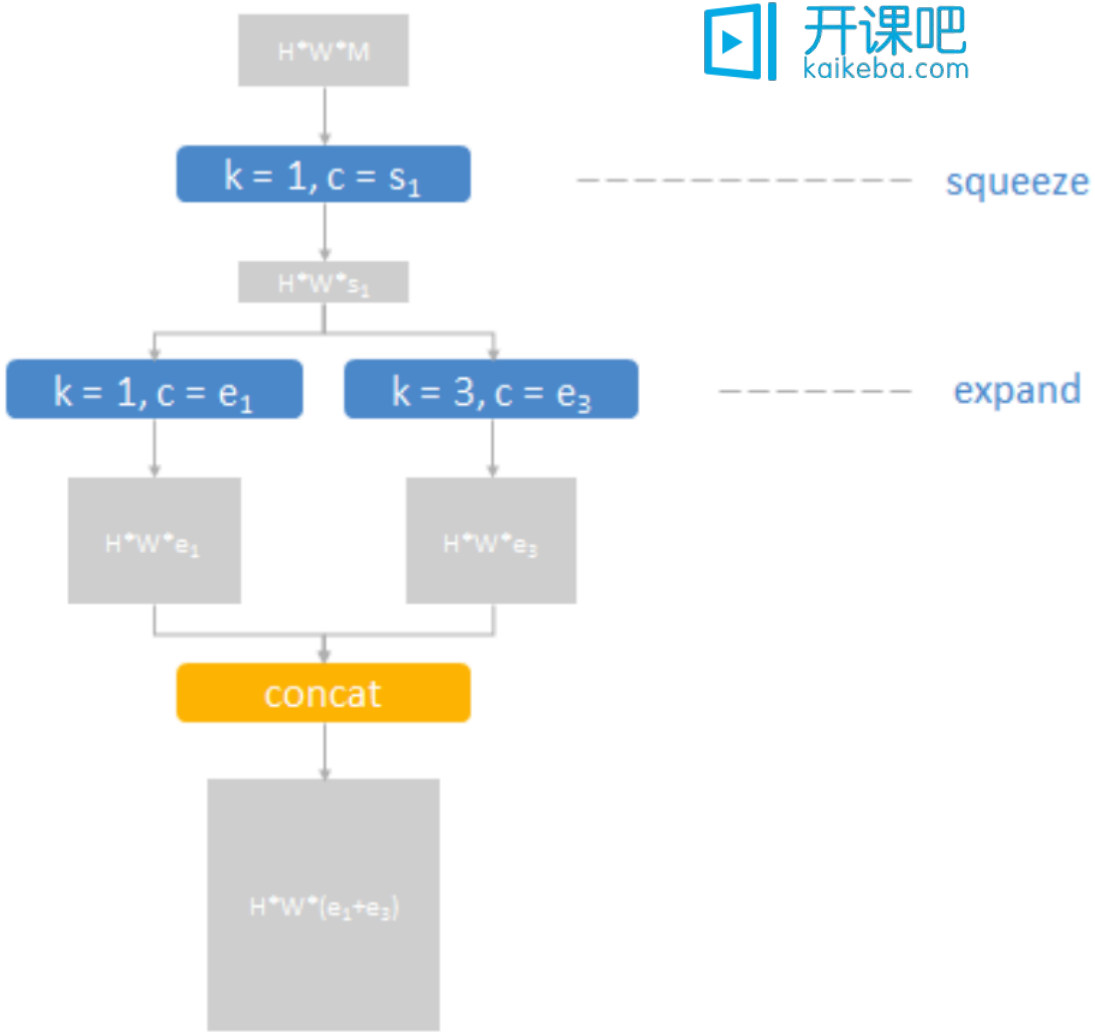
H*W*(e1+e2)*3*3*M

==========>

H*W*s1*1*1*M

+H*W*e1*1*1*s1

+H*W*e3*3*3*s1



开课吧 kaikeba.com

Fire Module

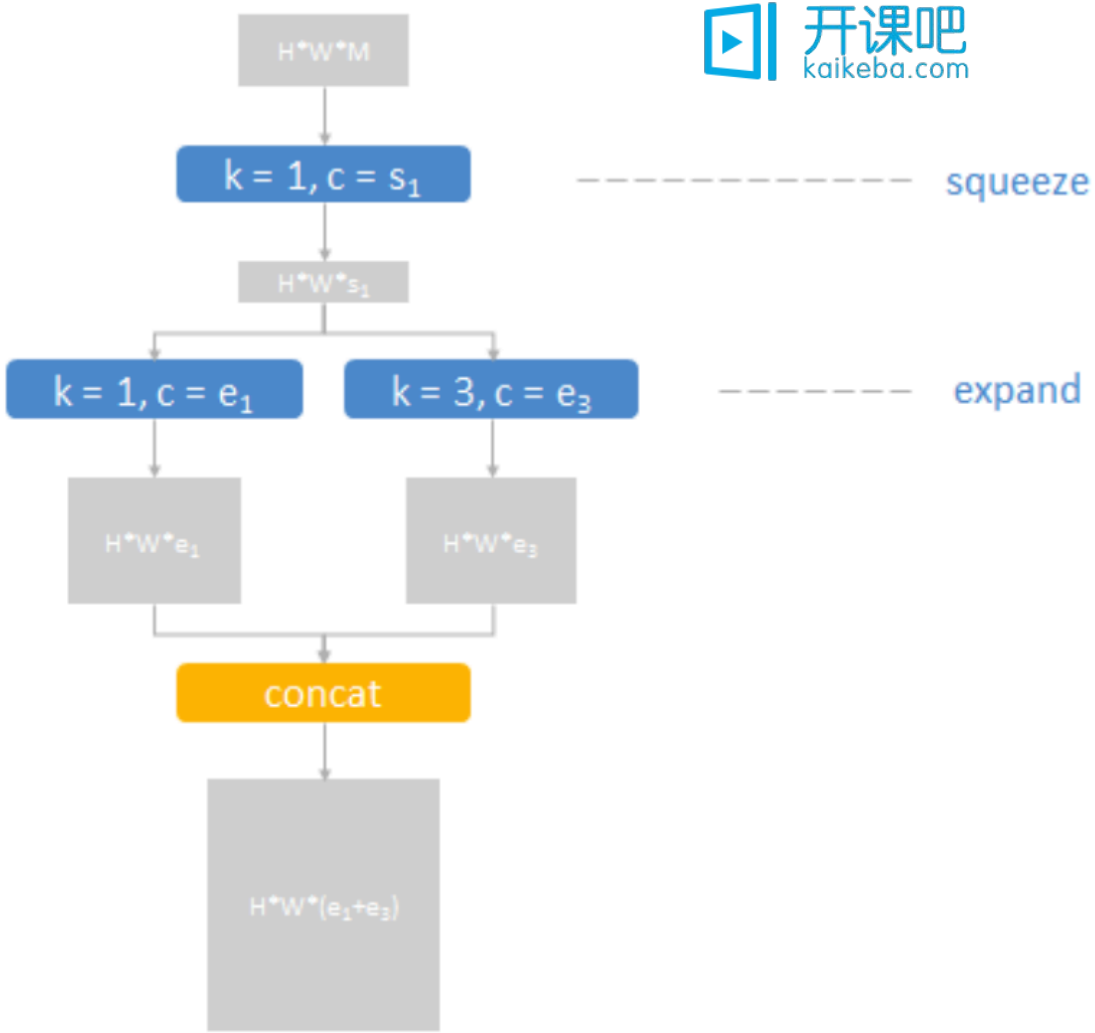
(e1+e2)*3*3*M

==========>

s1*1*1*M

+e1*1*1*s1

+e3*3*3*s1

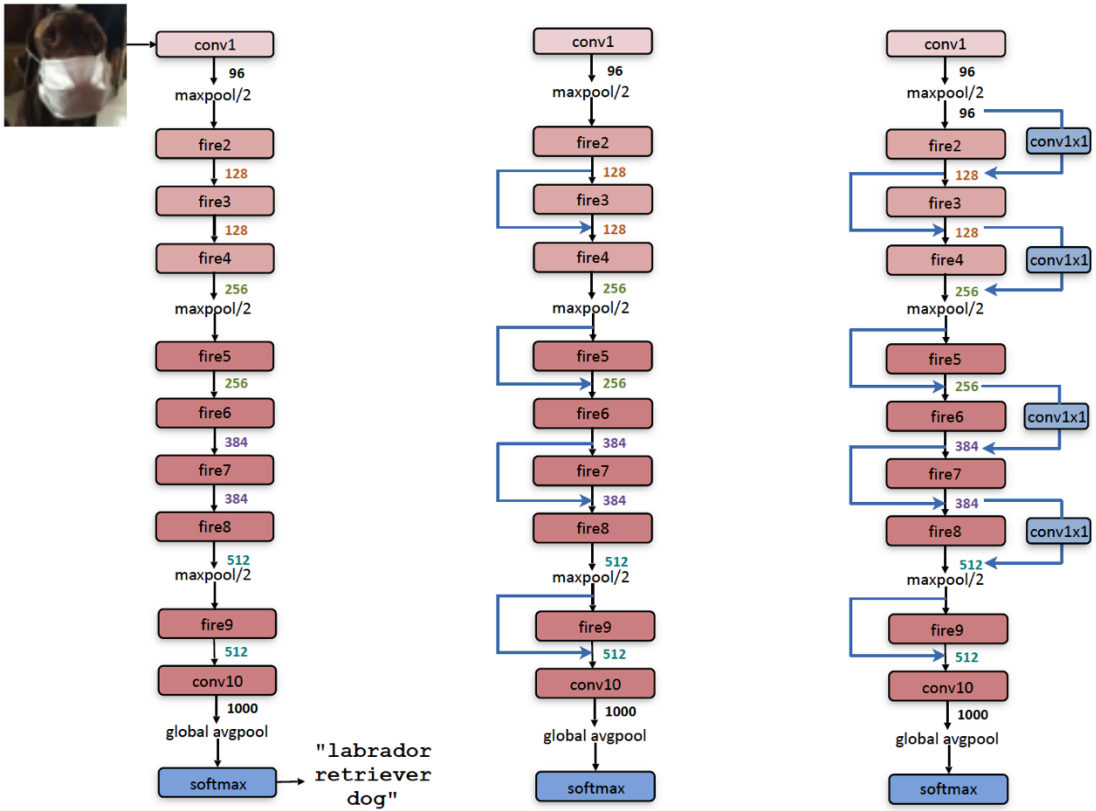


开课吧
kaikeba.com

Squeeze的多种形式

右图是SqueezeNet的几个实现，左侧是不加short-cut的SqueezeNet，中间是加了short-cut的，右侧是复杂的short-cut。

1. 激活函数默认都使用ReLU
2. fire9之后接了一个rate为0.5的dropout
3. 使用same卷积(padding)



深层压缩

论文作者将SqueezeNet与AlexNet在ImageNet上做了对比，值得注意的是，不仅对比了基础模型之间的差异，还对比了模型压缩的性能，其中模型压缩主要采用的技术有SVD，网络剪枝（network pruning）和量化（quantization）

layer name/type	output size	filter size / stride (if not a fire layer)	depth	s_{1x1} (#1x1 squeeze)	e_{1x1} (#1x1 expand)	e_{3x3} (#3x3 expand)	s_{1x1} sparsity	e_{1x1} sparsity	e_{3x3} sparsity	# bits	#parameter before pruning	#parameter after pruning
input image	224x224x3										-	-
conv1	111x111x96	7x7/2 (x96)	1				100% (7x7)			6bit	14,208	14,208
maxpool1	55x55x96	3x3/2	0									
fire2	55x55x128		2	16	64	64	100%	100%	**33%**	6bit	11,920	5,746
fire3	55x55x128		2	16	64	64	100%	100%	**33%**	6bit	12,432	6,258
fire4	55x55x256		2	32	128	128	100%	100%	**33%**	6bit	45,344	20,646
maxpool4	27x27x256	3x3/2	0									
fire5	27x27x256		2	32	128	128	100%	100%	**33%**	6bit	49,440	24,742
fire6	27x27x384		2	48	192	192	100%	**50%**	**33%**	6bit	104,880	44,700
fire7	27x27x384		2	48	192	192	**50%**	100%	**33%**	6bit	111,024	46,236
fire8	27x27x512		2	64	256	256	100%	**50%**	**33%**	6bit	188,992	77,581
maxpool8	13x12x512	3x3/2	0									
fire9	13x13x512		2	64	256	256	**50%**	100%	**30%**	6bit	197,184	77,581
conv10	13x13x1000	1x1/1 (x1000)	1				**20%** (3x3)			6bit	513,000	103,400
avgpool10	1x1x1000	13x13/1	0									
	activations	parameters					compression info				1,248,424 (total)	**421,098** (total)

开课吧 kaikeba.com

SqueezeNet的压缩能力

CNN architecture	Compression Approach	Data Type	Original → Compressed Model Size	Reduction in Model Size vs. AlexNet	Top-1 ImageNet Accuracy	Top-5 ImageNet Accuracy
AlexNet	None (baseline)	32 bit	240MB	1x	57.2%	80.3%
AlexNet	SVD (Denton et al., 2014)	32 bit	240MB → 48MB	5x	56.0%	79.4%
AlexNet	Network Pruning (Han et al., 2015b)	32 bit	240MB → 27MB	9x	57.2%	80.3%
AlexNet	Deep Compression (Han et al., 2015a)	5-8 bit	240MB → 6.9MB	35x	57.2%	80.3%
SqueezeNet (ours)	None	32 bit	4.8MB	**50x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	8 bit	4.8MB → 0.66MB	**363x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	6 bit	4.8MB → 0.47MB	**510x**	57.5%	80.3%

开课吧
kaikeba.com

总结-优点

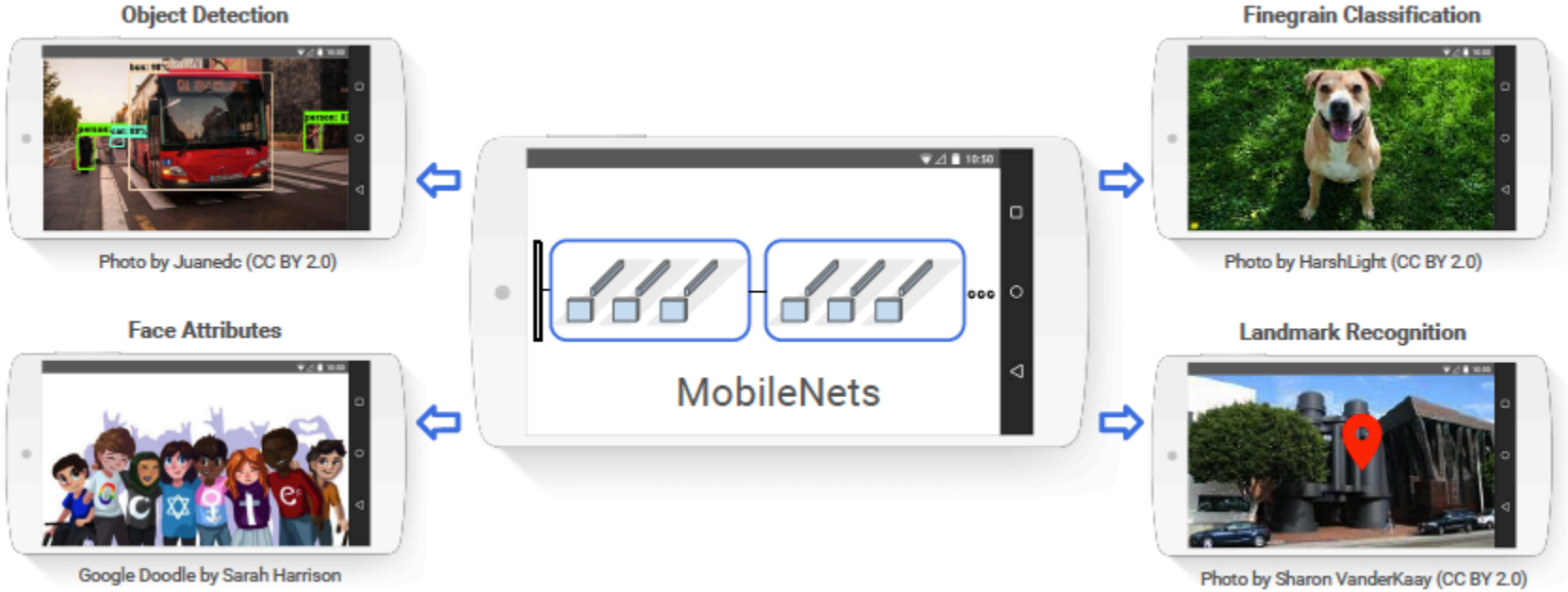
SqueezeNet的压缩策略是依靠将3x3卷积替换成两种类型的1x1卷积+(1x1+3x3)来达到的，其参数数量是等性能的AlexNet的2.14%。从参数数量上来看，SqueezeNet的目的达到了。SqueezeNet的最大贡献在于其开拓了模型压缩这一方向，之后的一系列文章也就此打开

开课吧
kaikeba.com

总结-缺点

1. 论文50倍压缩略显夸张，虽然纸面上是减少了50倍的参数，但是问题的主要症结在于AlexNet本身全连接节点过于庞大，50倍参数的减少和SqueezeNet的设计并没有关系，考虑去掉全连接之后3倍参数的减少更为合适。
2. SqueezeNet得到的模型是5MB左右，0.5MB的模型还要得益于Deep Compression。

MobileNet

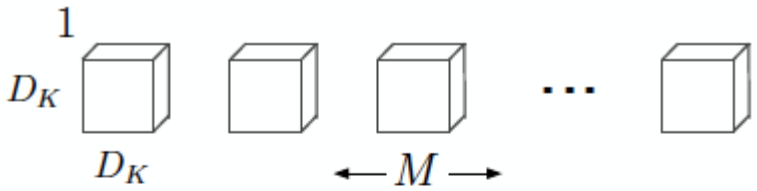

Object Detection
Photo by Juanedc (CC BY 2.0)
Face Attributes
Google Doodle by Sarah Harrison
MobileNets
Finegrain Classification
Photo by HarshLight (CC BY 2.0)
Landmark Recognition
Photo by Sharon VanderKaay (CC BY 2.0)

开课吧
kaikeba.com

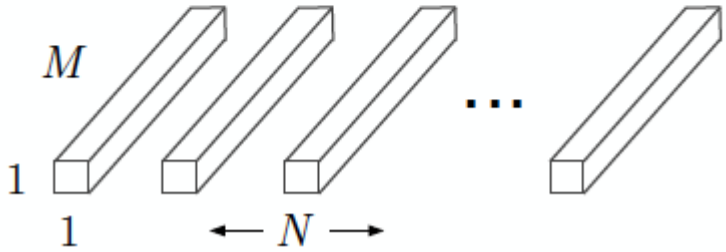
MobileNet v1是小模型中的一个代表作，它是一种基于流水线结构，使用深度级可分离卷积构建的轻量级神经网络，并通过两个超参数的引入使得开发人员可以基于自己的应用和资源限制选择合适的模型

Depthwidth Seprereable Convnet

深度可分离卷积是将标准卷积拆分为了两个操作：深度卷积(depthwise convolution) 和 逐点卷积(pointwise convolution)

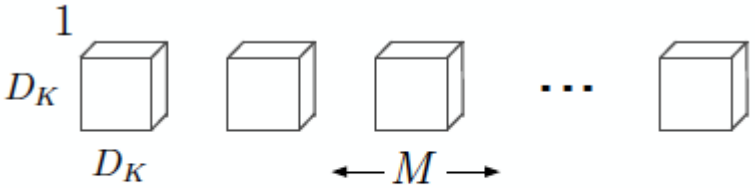


(b) Depthwise Convolutional Filters

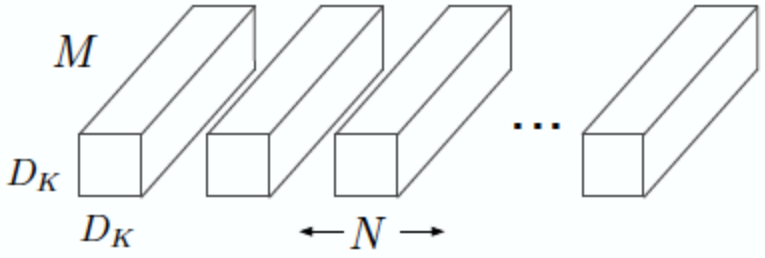


深度卷积

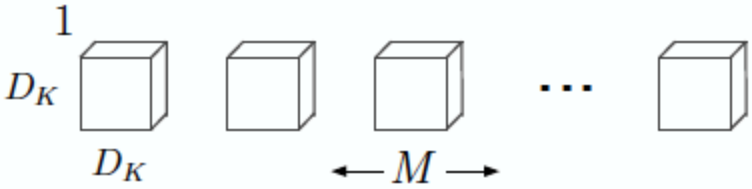
Group=Channel的
分组卷积网络



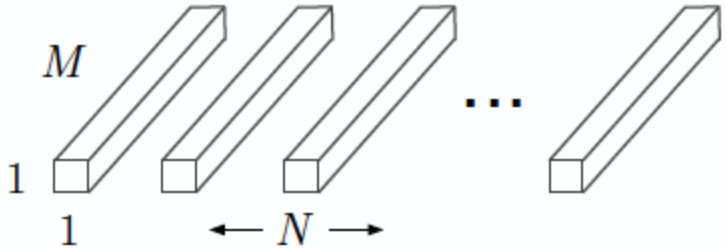
(b) Depthwise Convolutional Filters



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



假设输入是Df*Df*M,输出是Df*Df*N

卷积核Dk*Dk

a是标准的卷积神经网络

问：计算量是多少?

---

b是深度卷积网络和逐点卷积网络

问计算量是多少?

开课吧 kaikeba.com

开课吧
kaikeba.com

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F$$

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$$

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F}$$

$$= \frac{1}{N} + \frac{1}{D_K^2}$$

N通常很大的情况下，1/N会很小，如果Dk是3x3的卷积网络，那么会压缩9倍左右

开课吧
kaikeba.com

实际网络结构

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

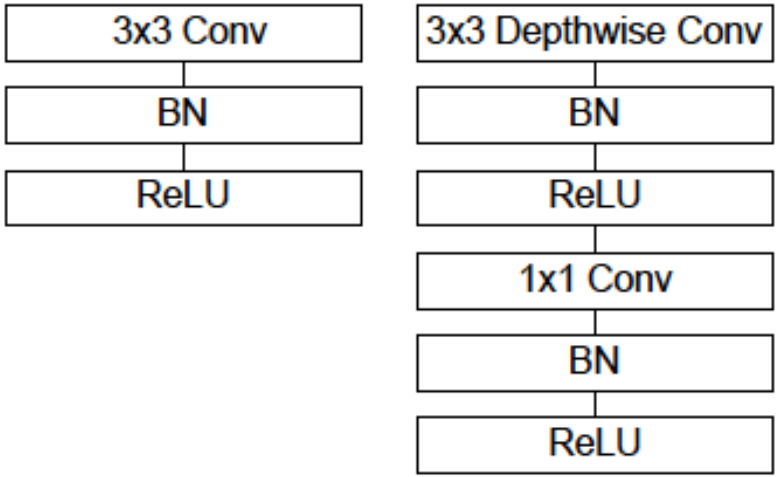


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

开课吧
kaikeba.com

资源占用

Table 2. Resource Per Layer Type

Type	Mult-Adds	Parameters
Conv 1×1	94.86%	74.59%
Conv DW 3×3	3.06%	1.06%
Conv 3×3	1.19%	0.02%
Fully Connected	0.18%	24.33%

开课吧
kaikeba.com

宽度因子

但是很多时候在特定应用下还是需要更小更快的模型，为此引入了宽度因子 alpha（Width Mutiplier）在每一层对网络的输入输出通道数进行缩减，输入通道数由 M 到 alpha*M，输出通道数由 N 到 alpha*N，变换后的计算量变为：

$$D_K \cdot D_K \cdot \alpha M \cdot D_F \cdot D_F + \alpha M \cdot \alpha N \cdot D_F \cdot D_F$$

通常alpha在(0, 1]之间，比较典型的值有 1, 0.75, 0.5 和 0.25。计算量和参数数量减少程度与未使用宽度因子之前提高了1/alpha**2倍。

开课吧
kaikeba.com

分辨率因子

分辨率因子ρ(resolution multiplier)，用于控制输入和内部层表示，即用分辨率因子控制输入的分辨率。

$$D_K \cdot D_K \cdot \alpha M \cdot \rho D_F \cdot \rho D_F + \alpha M \cdot \alpha N \cdot \rho D_F \cdot \rho D_F$$

分辨率因子是通过设置input的分辨率来实现的

开课吧
kaikeba.com

- 使用宽度因子和分辨率因子后的压缩比例
- 为什么?

Layer/Modification	Million Mult-Adds	Million Parameters
Convolution	462	2.36
Depthwise Separable Conv	52.3	0.27
$\alpha = 0.75$	29.6	0.15
$\rho = 0.714$	15.1	0.15

一些实验

Table 4. Depthwise Separable vs Full Convolution MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

Table 5. Narrow vs Shallow MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.75 MobileNet	68.4%	325	2.6
Shallow MobileNet	65.3%	307	2.9

Table 5. Narrow vs Shallow MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.75 MobileNet	68.4%	325	2.6
Shallow MobileNet	65.3%	307	2.9

Table 6. MobileNet Width Multiplier

Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
0.75 MobileNet-224	68.4%	325	2.6
0.5 MobileNet-224	63.7%	149	1.3
0.25 MobileNet-224	50.6%	41	0.5

Table 7. MobileNet Resolution

Resolution	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
1.0 MobileNet-192	69.1%	418	4.2
1.0 MobileNet-160	67.2%	290	4.2
1.0 MobileNet-128	64.4%	186	4.2

开课吧 kaikeba.com

开课吧
kaikeba.com

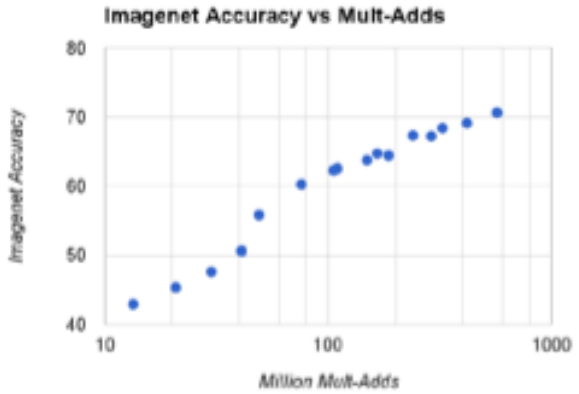


Figure 4. This figure shows the trade off between computation (Mult-Adds) and accuracy on the ImageNet benchmark. Note the log linear dependence between accuracy and computation.

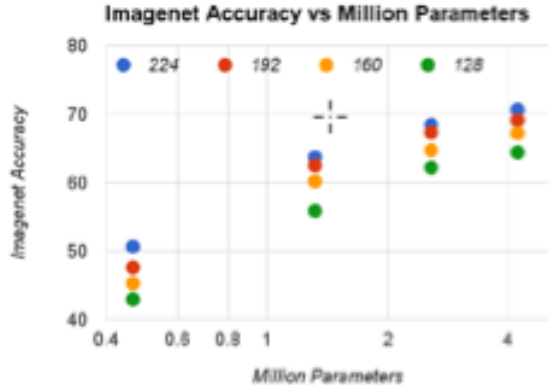


Figure 5. This figure shows the trade off between the number of parameters and accuracy on the ImageNet benchmark. The colors encode input resolutions. The number of parameters do not vary based on the input resolution.

开课吧
kaikeba.com

MobileNet在知识蒸馏中的应用

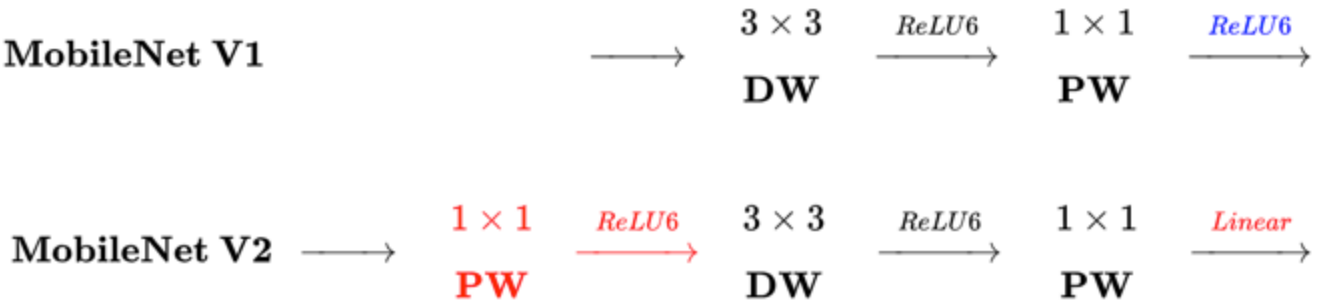
Width Multiplier / Resolution	Mean AP	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	88.7%	568	3.2
0.5 MobileNet-224	88.1%	149	0.8
0.25 MobileNet-224	87.2%	45	0.2
1.0 MobileNet-128	88.1%	185	3.2
0.5 MobileNet-128	87.7%	48	0.8
0.25 MobileNet-128	86.4%	15	0.2
Baseline	86.9%	1600	7.5

开课吧
kaikeba.com

MobileNet-V2

Inverted Residules and BottleNeck

MobileNet中V1到V2的进化



开课吧
kaikeba.com

相同点

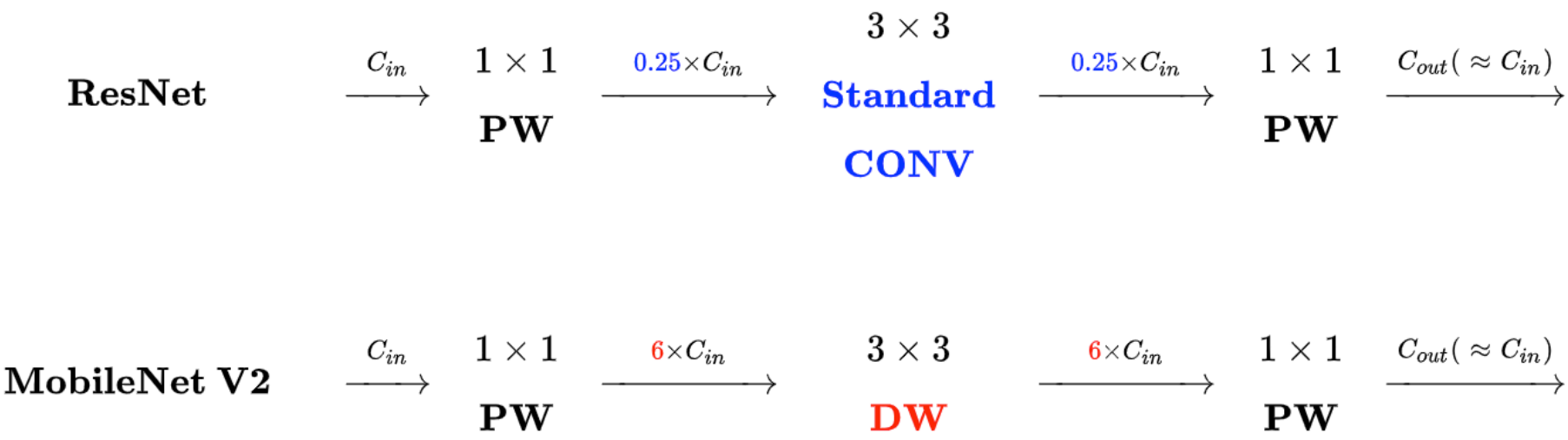
$$\text{Complexity} \quad \frac{\textbf{Depth-wise Separable CONV}}{\textbf{Standard CONV}} = \frac{1}{K^2} + \frac{1}{C_{out}} \sim \frac{1}{K^2}$$

不同点-Linear BottleNeck

- V2 在 DW 卷积之前新加了一个 PW 卷积

- V2 去掉了第二个 PW 的激活函数

对比ResNet与MobileNet V2



开课吧
kaikeba.com

相同点

- MobileNet V2 借鉴 ResNet，都采用了1x1->3x3->1x1 的模式

- MobileNet V2 借鉴 ResNet，同样使用 Shortcut 将输出与输入相加

开课吧
kaikeba.com

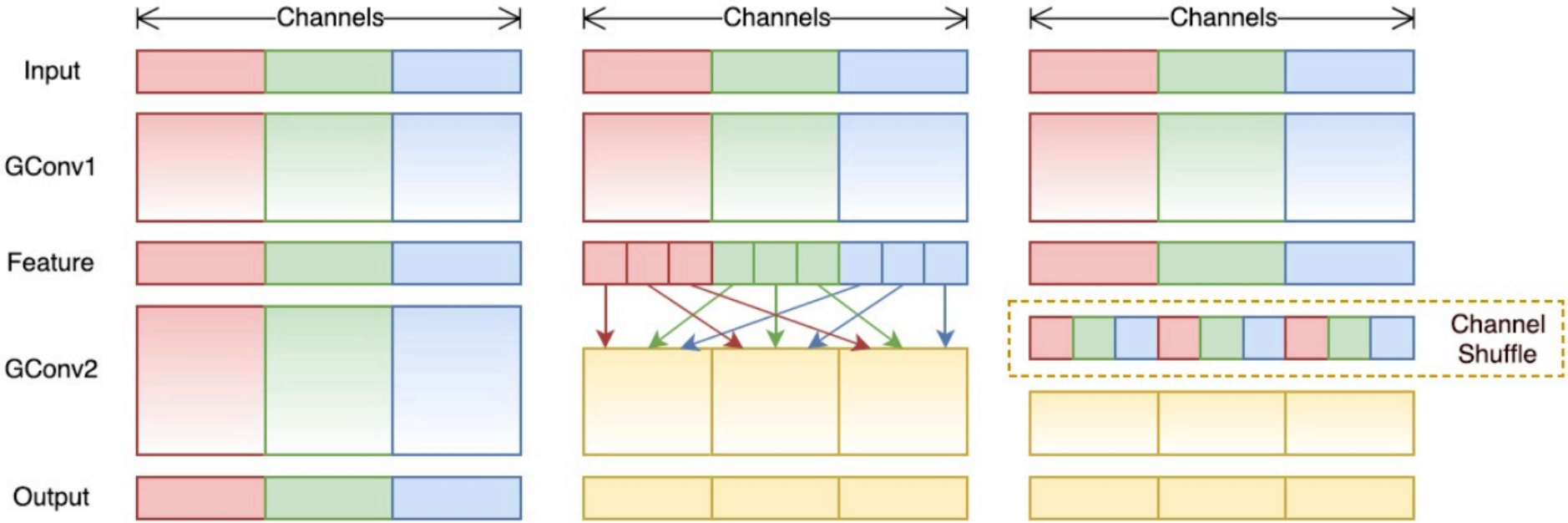
不同点--Invert Residual Block

- ResNet 使用 标准卷积 提特征，MobileNet 始终使用 DW卷积 提特征

- ResNet 先降维 (0.25倍)、卷积、再升维，而 MobileNet V2 则是 先升维 (6倍)、卷积、再降维。

开课吧
kaikeba.com

ShuffleNet



开课吧
kaikeba.com

核心设计思路

对不同的channels进行shuffle来解决group convolution带来的弊端

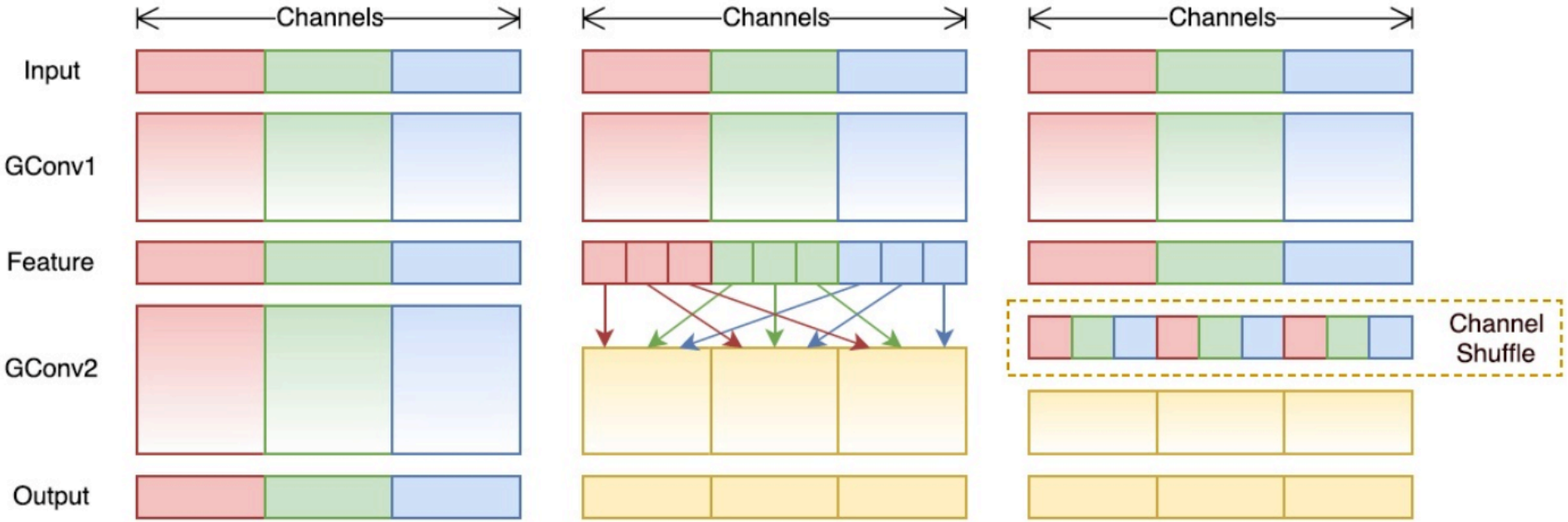
深度可分离卷积DWConv的缺陷：Dense Pointwise Convolution

ResNext中90%以上是1x1卷积操作

Group卷积的缺陷：组与组之间的信息流难以传递

开课吧
kaikeba.com

转置重组



转置重组

假定将输入层分为g组，总通道数为g*n，首先你将通道那个维度拆分为(g,n)两个维度，然后将这两个维度转置变成 (n*g)，最后重新reshape成一个维度

网络重组

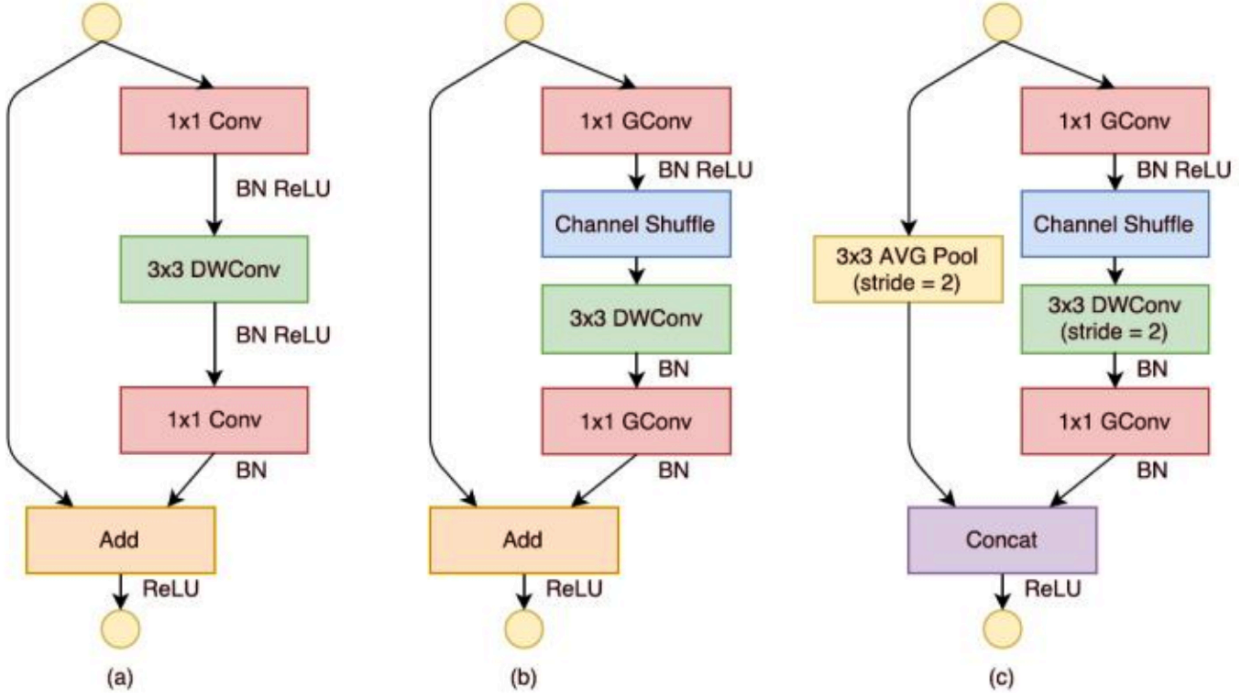


图2 ShuffleNet的基本单元

开课吧
kaikeba.com

网络结构

Layer	Output size	KSize	Stride	Repeat	Output channels (g groups)				
					$g = 1$	$g = 2$	$g = 3$	$g = 4$	$g = 8$
Image	224×224				3	3	3	3	3
Conv1	112×112	3×3	2	1	24	24	24	24	24
MaxPool	56×56	3×3	2						
Stage2	28×28		2	1	144	200	240	272	384
	28×28		1	3	144	200	240	272	384
Stage3	14×14		2	1	288	400	480	544	768
	14×14		1	7	288	400	480	544	768
Stage4	7×7		2	1	576	800	960	1088	1536
	7×7		1	3	576	800	960	1088	1536
GlobalPool	1×1	7×7							
FC					1000	1000	1000	1000	1000
Complexity					143M	140M	137M	133M	137M

开课吧
kaikeba.com

模型效果

Model	Complexity (MFLOPs)	Classification error (%)				
		$g = 1$	$g = 2$	$g = 3$	$g = 4$	$g = 8$
ShuffleNet 1×	140	33.6	32.7	32.6	32.8	**32.4**
ShuffleNet 0.5×	38	45.1	44.4	43.2	**41.6**	42.3
ShuffleNet 0.25×	13	57.1	56.8	55.0	54.2	**52.7**

开课吧
kaikeba.com

Model	Cls err. (%, no shuffle)	Cls err. (%, shuffle)	Δ err. (%)
ShuffleNet 1x ($g = 3$)	34.5	**32.6**	1.9
ShuffleNet 1x ($g = 8$)	37.6	**32.4**	5.2
ShuffleNet 0.5x ($g = 3$)	45.7	**43.2**	2.5
ShuffleNet 0.5x ($g = 8$)	48.1	**42.3**	5.8
ShuffleNet 0.25x ($g = 3$)	56.3	**55.0**	1.3
ShuffleNet 0.25x ($g = 8$)	56.5	**52.7**	3.8

开课吧
kaikeba.com

Model	Complexity (MFLOPs)	Cls err. (%)	Δ err. (%)
1.0 MobileNet-224	569	29.4	-
ShuffleNet 2× ($g = 3$)	524	**26.3**	3.1
ShuffleNet 2× (with *SE*[13], $g = 3$)	527	**24.7**	4.7
0.75 MobileNet-224	325	31.6	-
ShuffleNet 1.5× ($g = 3$)	292	**28.5**	3.1
0.5 MobileNet-224	149	36.3	-
ShuffleNet 1× ($g = 8$)	140	**32.4**	3.9
0.25 MobileNet-224	41	49.4	-
ShuffleNet 0.5× ($g = 4$)	38	**41.6**	7.8
ShuffleNet 0.5× (shallow, $g = 3$)	40	42.8	6.6