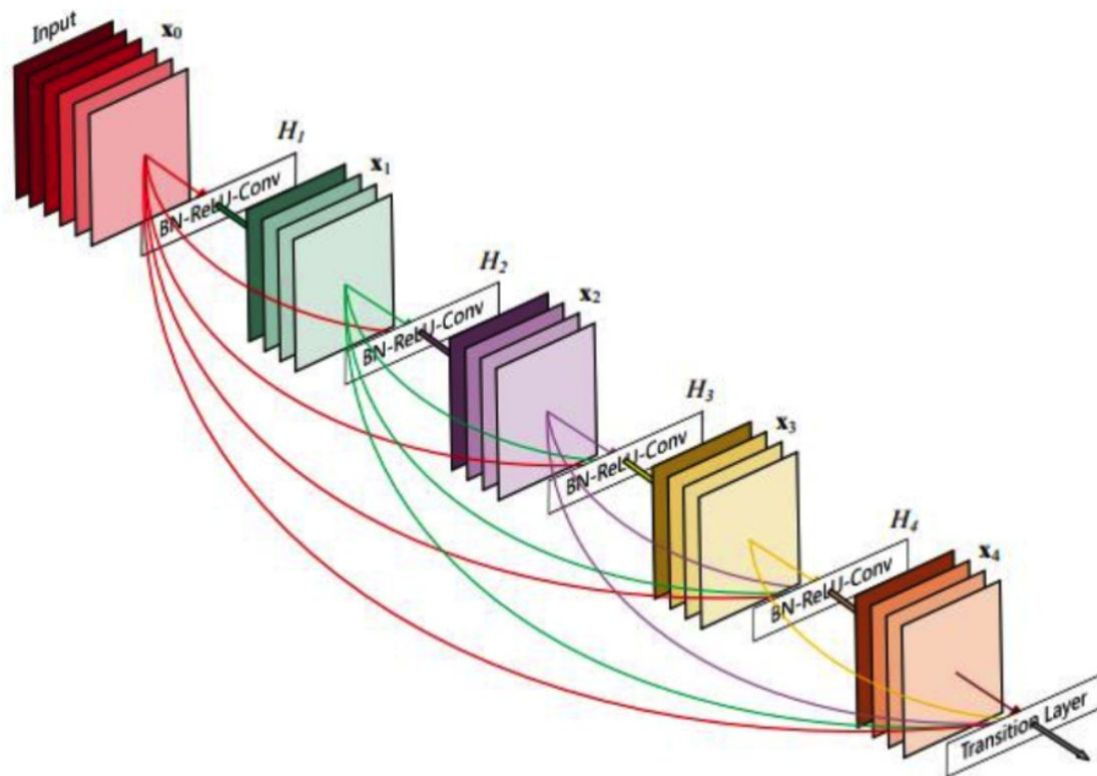


# CNN Arch 2

Allen . Huang

# DenseNet

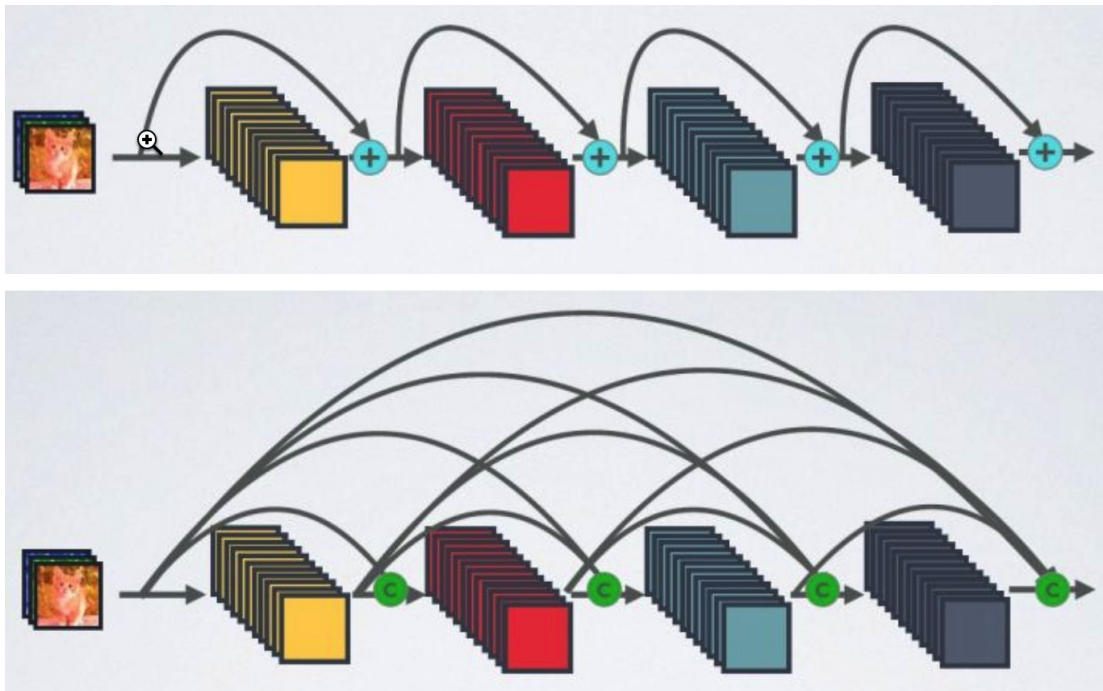


# 背景

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



# 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.

# ResNet与DenseNet的运算对比

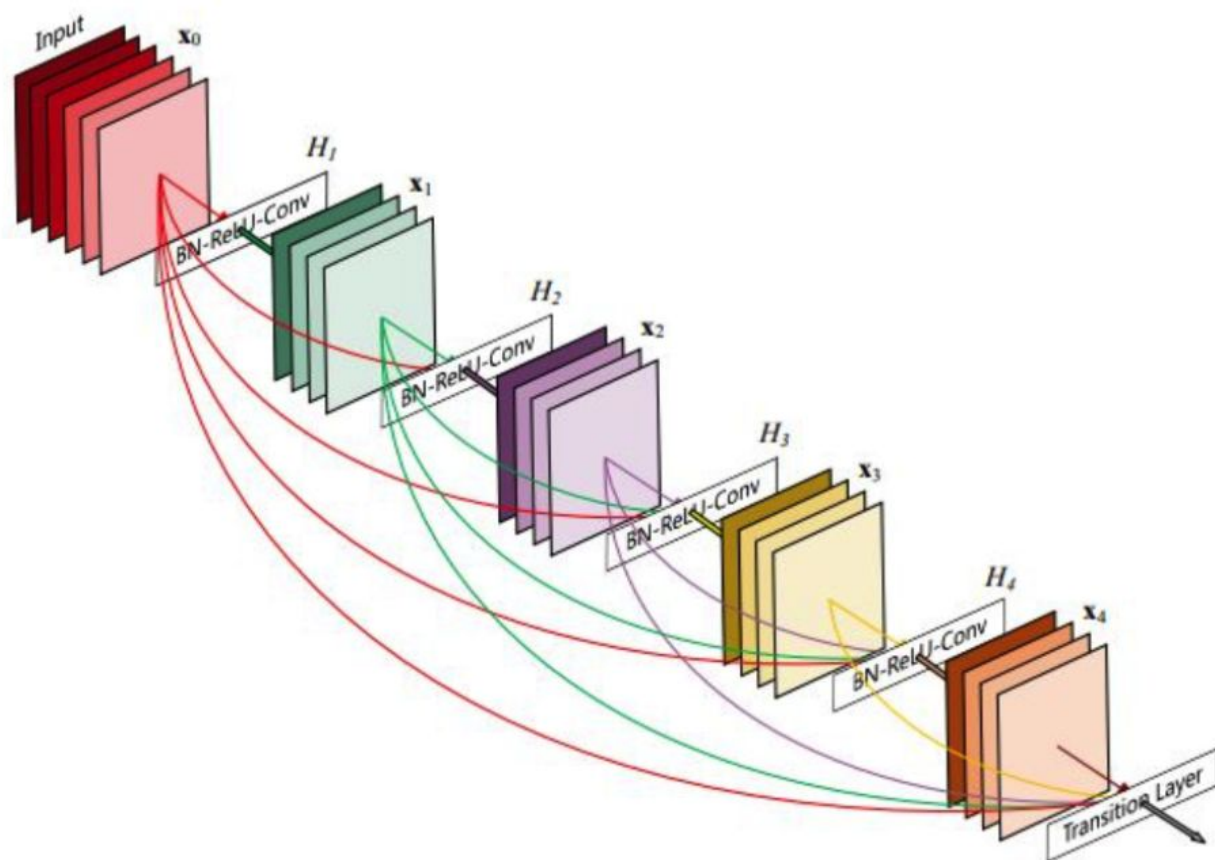
- ResNet中, 每个层与前面的某层(一般是2~3层)短路连接在一起, 连接方式是通过**元素级相加**
- 在DenseNet中, 每个层都会与前面所有层在channel维度上**叠加(concat)在一起**, 并作为下一层的输入

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:  $x_l = H_l(x_{l-1}) + x_{l-1}$

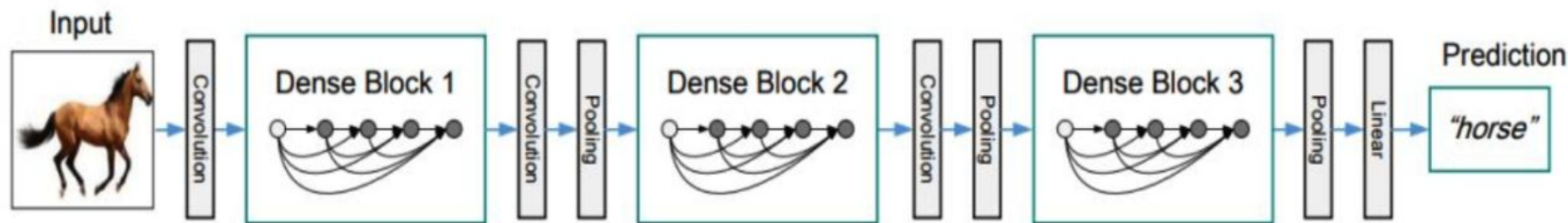
在DenseNet中, 会连接前面所有层作为输入:  $x_l = H_l([x_0, x_1, \dots, x_{l-1}])$

其中, 上面的 H代表是非线性转化函数(non-linear 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。



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

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	$112 \times 112$	$7 \times 7$ conv, stride 2			
Pooling	$56 \times 56$	$3 \times 3$ max pool, stride 2			
Dense Block (1)	$56 \times 56$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	$56 \times 56$	$1 \times 1$ conv			
	$28 \times 28$	$2 \times 2$ average pool, stride 2			
Dense Block (2)	$28 \times 28$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	$28 \times 28$	$1 \times 1$ conv			
	$14 \times 14$	$2 \times 2$ average pool, stride 2			
Dense Block (3)	$14 \times 14$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	$14 \times 14$	$1 \times 1$ conv			
	$7 \times 7$	$2 \times 2$ average pool, stride 2			
Dense Block (4)	$7 \times 7$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	$1 \times 1$	$7 \times 7$ global average pool			
		1000D fully-connected, softmax			

# 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

# 压缩-Compression

Transition Layer用来进行DenseBlock之间的Pooling操作

$\theta$ : 代表压缩率

上层DenseBlock的输出的featuremap层数是M

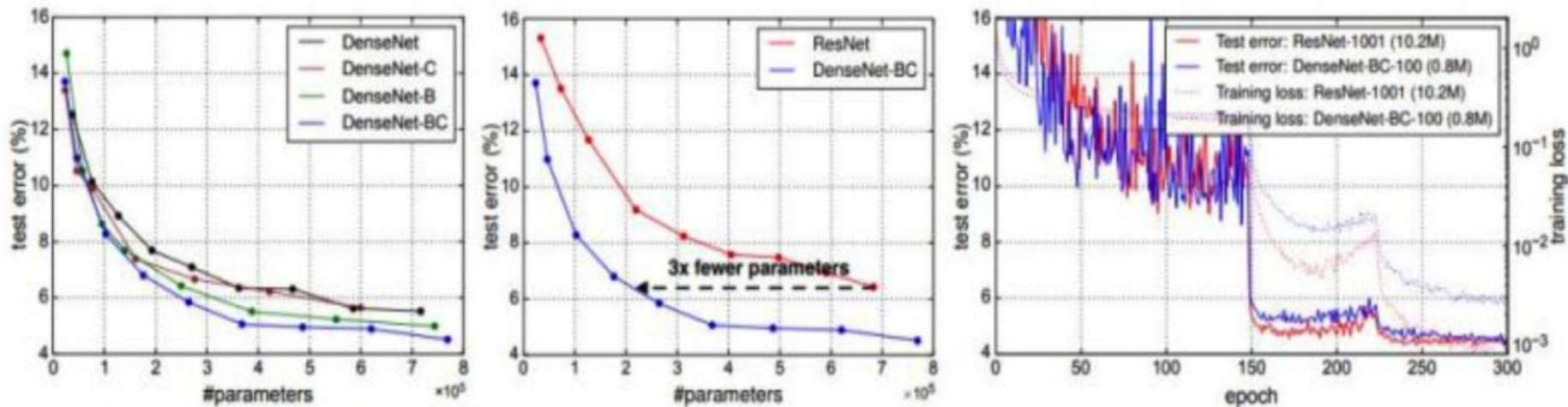
下层DenseBlock的输入是  $M' = \lfloor \theta M \rfloor$

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

# 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。

# 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.

# 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	<b>7.00</b>	5.24	<b>27.55</b>	24.42	1.79
DenseNet ( $k = 12$ )	100	7.0M	<b>5.77</b>	<b>4.10</b>	<b>23.79</b>	<b>20.20</b>	1.67
DenseNet ( $k = 24$ )	100	27.2M	<b>5.83</b>	<b>3.74</b>	<b>23.42</b>	<b>19.25</b>	<b>1.59</b>
DenseNet-BC ( $k = 12$ )	100	0.8M	<b>5.92</b>	4.51	<b>24.15</b>	22.27	1.76
DenseNet-BC ( $k = 24$ )	250	15.3M	<b>5.19</b>	<b>3.62</b>	<b>19.64</b>	<b>17.60</b>	1.74
DenseNet-BC ( $k = 40$ )	190	25.6M	-	<b>3.46</b>	-	<b>17.18</b>	-

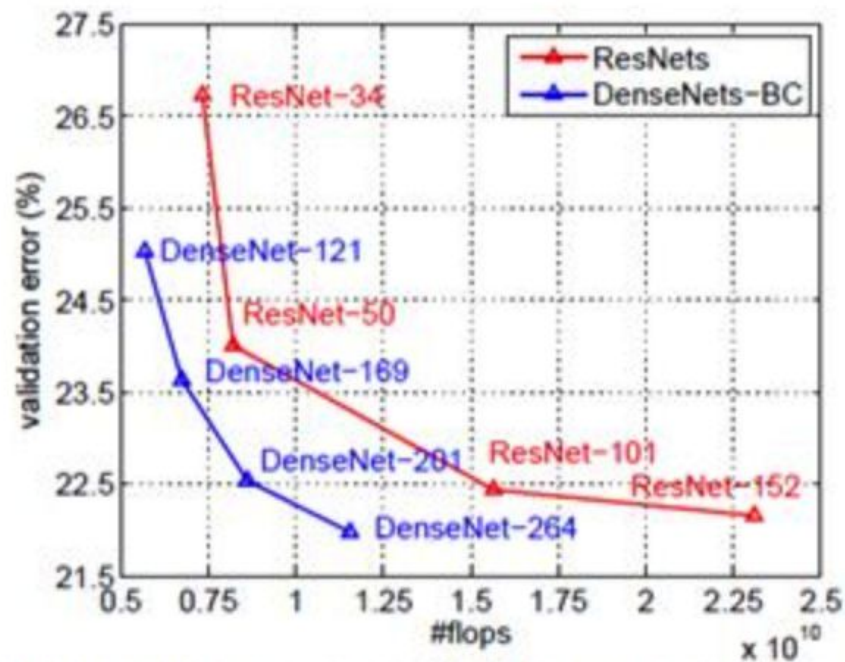
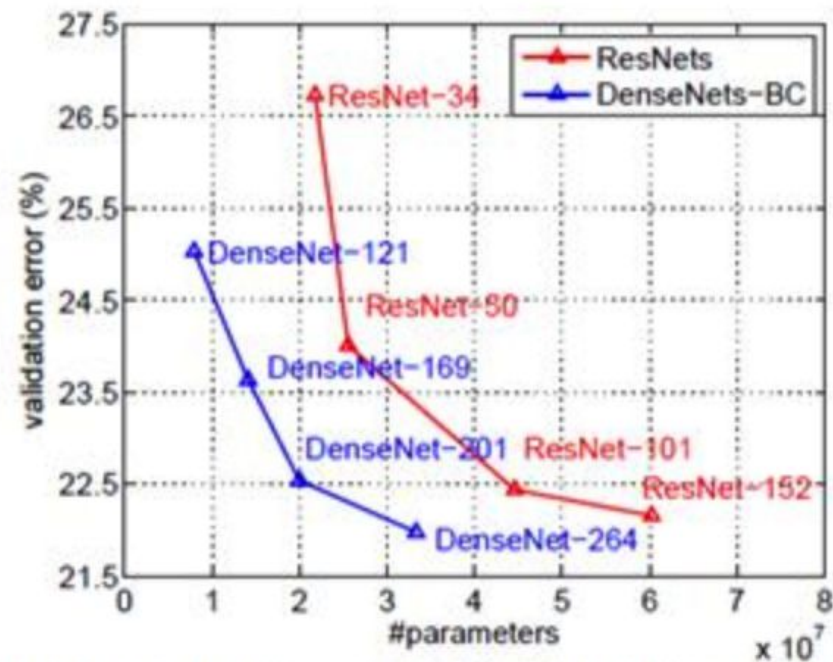
**Table 2:** Error rates (%) on CIFAR and SVHN datasets.  $k$  denotes network's growth rate. Results that surpass all competing methods are **bold** and the overall best results are **blue**. "+" indicates standard data augmentation (translation and/or mirroring). \* indicates results run by ourselves. 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.

# 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.

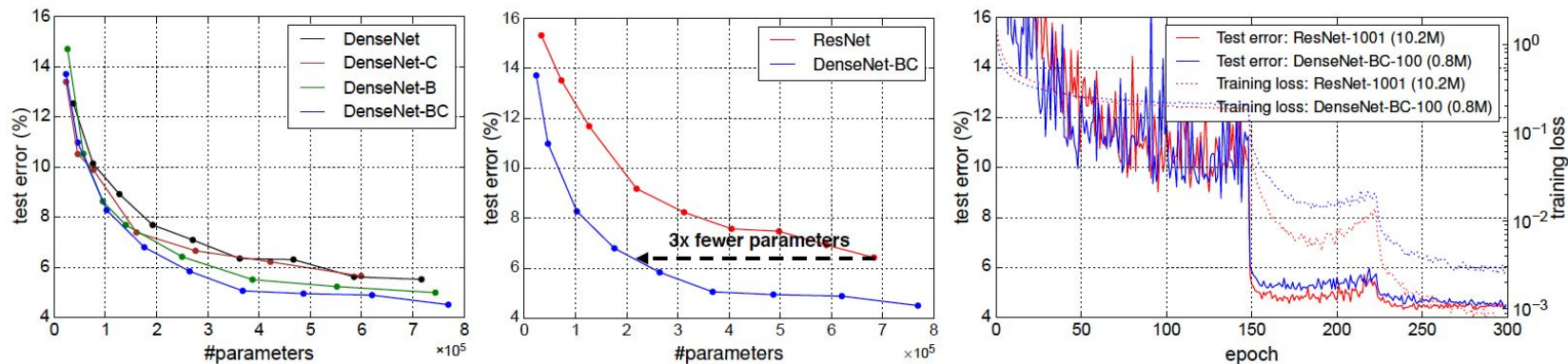




**Figure 3:** Comparison of the DenseNets and ResNets top-1 error rates (single-crop testing) on the ImageNet validation dataset as a function of learned parameters (*left*) and FLOPs during test-time (*right*).



# 模型简化性



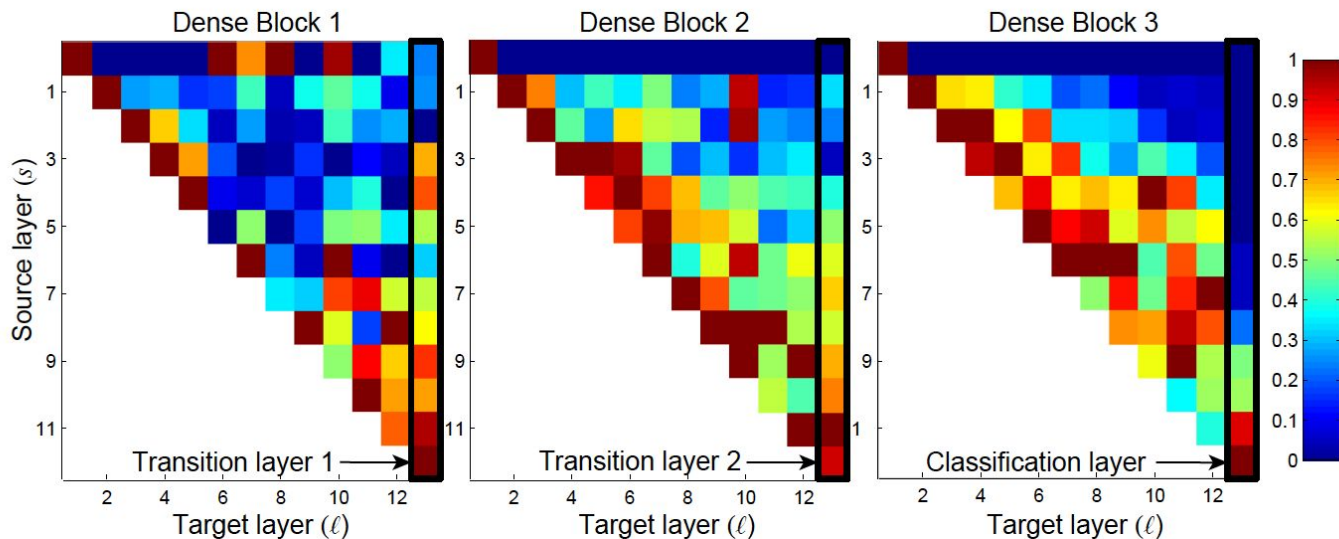
**Figure 4:** *Left:* Comparison of the parameter efficiency on C10+ between DenseNet variations. *Middle:* Comparison of the parameter efficiency between DenseNet-BC and (pre-activation) ResNets. DenseNet-BC requires about 1/3 of the parameters as ResNet to achieve comparable accuracy. *Right:* Training 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  $(s, \ell)$  encodes the average  $L1$  norm (normalized by number of input feature-maps) of the weights connecting convolutional layer  $s$  to  $\ell$  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.

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.

# 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

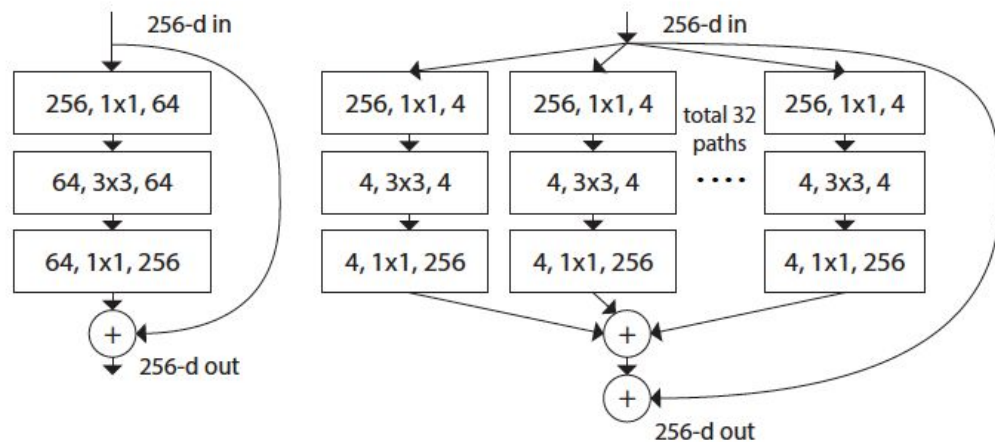
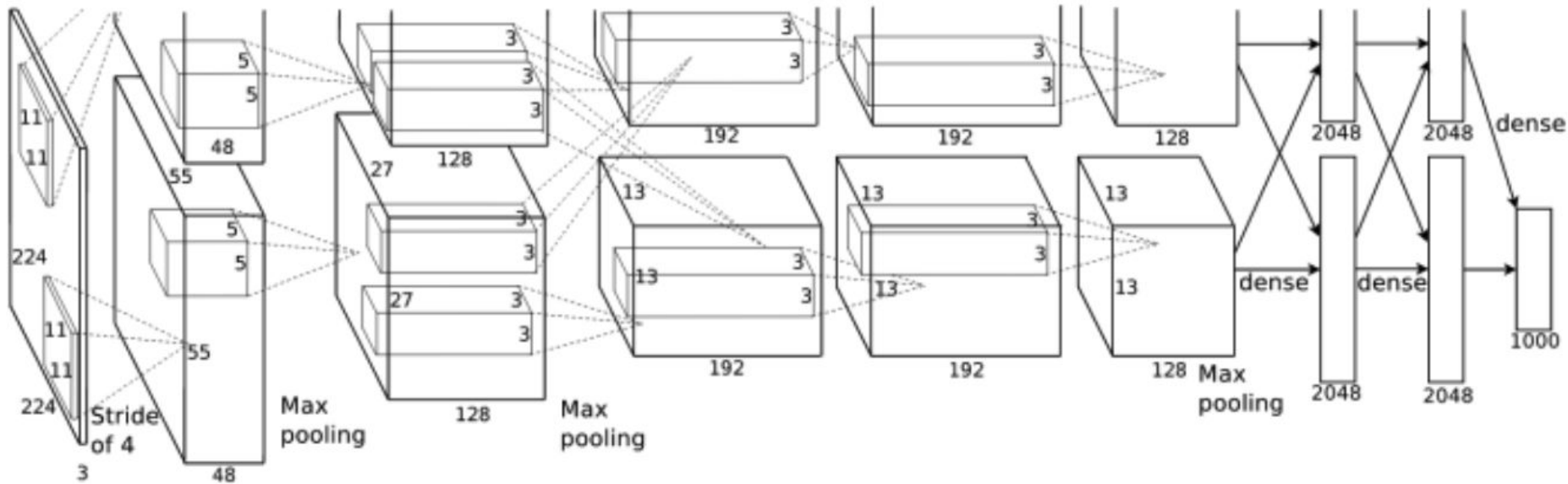


Figure 1. **Left:** A block of ResNet [14]. **Right:** A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

# Group Convolution

最早出现于AlexNet中，无法在单块GPU上运行

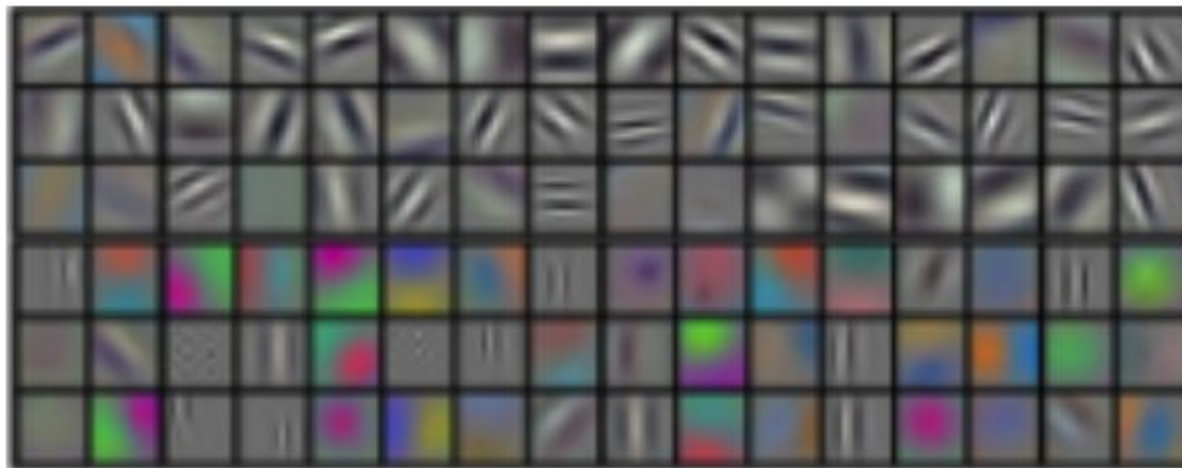


# 分组卷积

- 上一层的输出feature map有 $N$ 个, 即通道数 $\text{channel}=N$
- 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

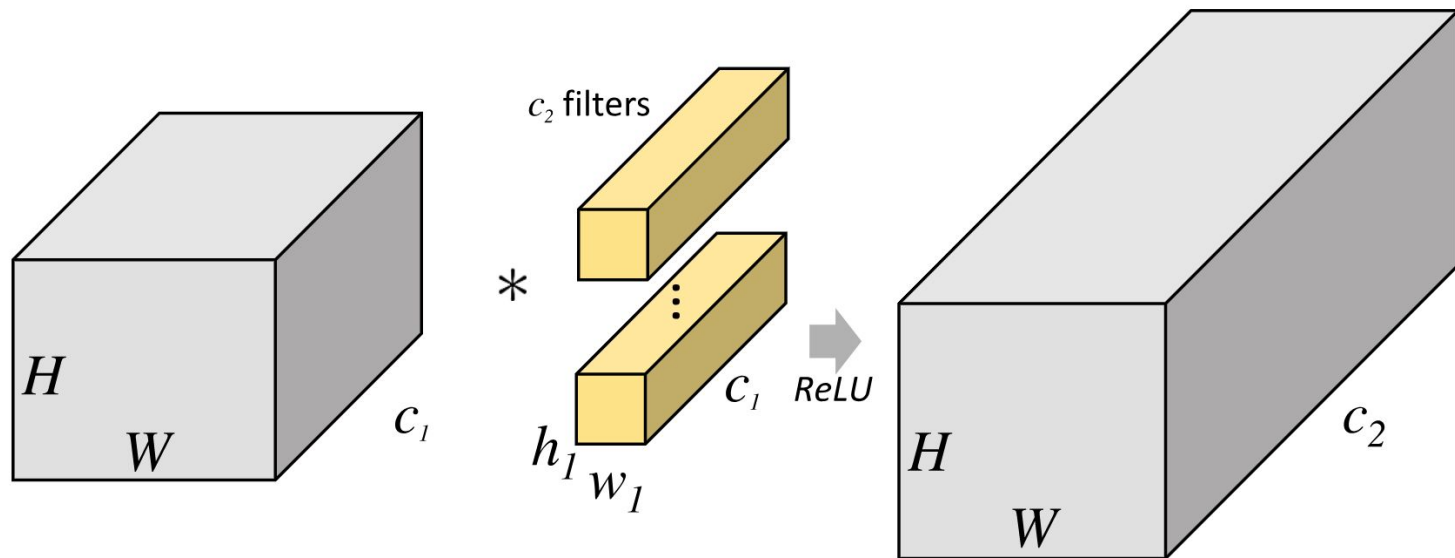


## 分组卷积的上下两部分的FeatureMap



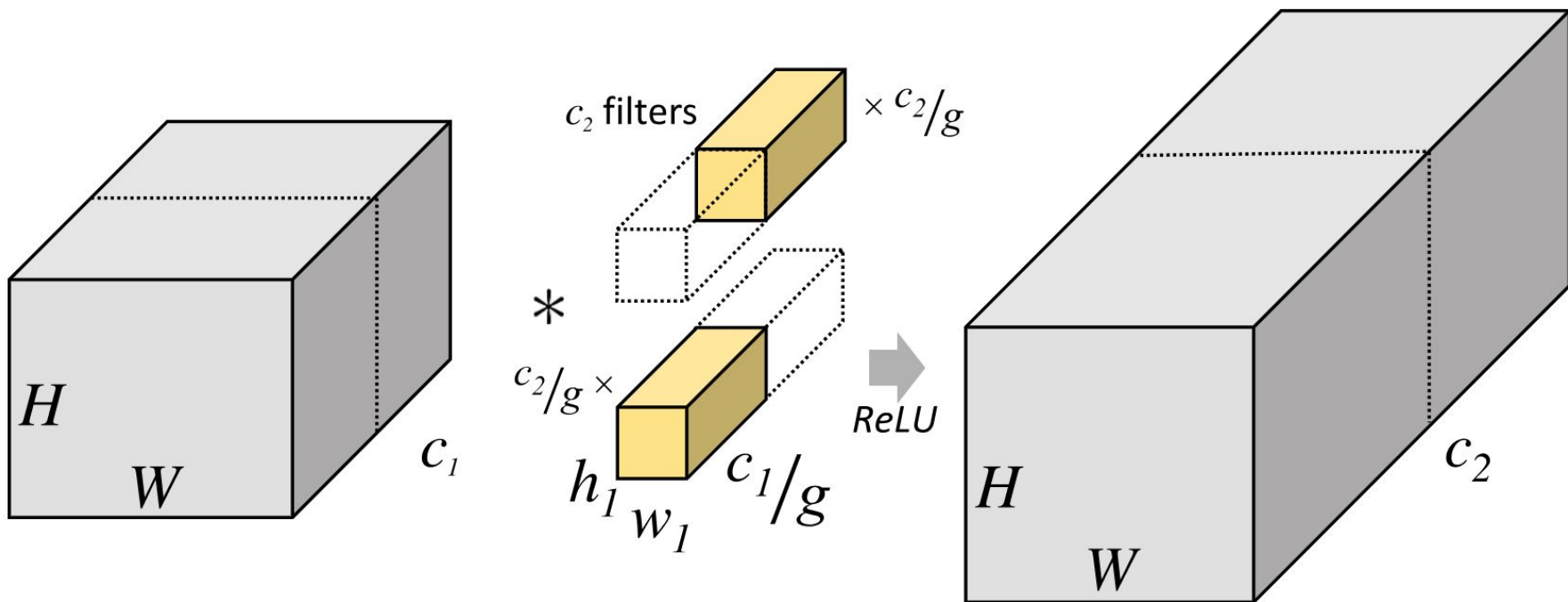
# 传统的卷积运算

$(H-h_1+1)*(W-w_1+1)*h_1*w_1*c_1*c_2$ 次运算



# Group卷积的运算量

运算量, 参数量是多少?



# 分组卷积的优缺点

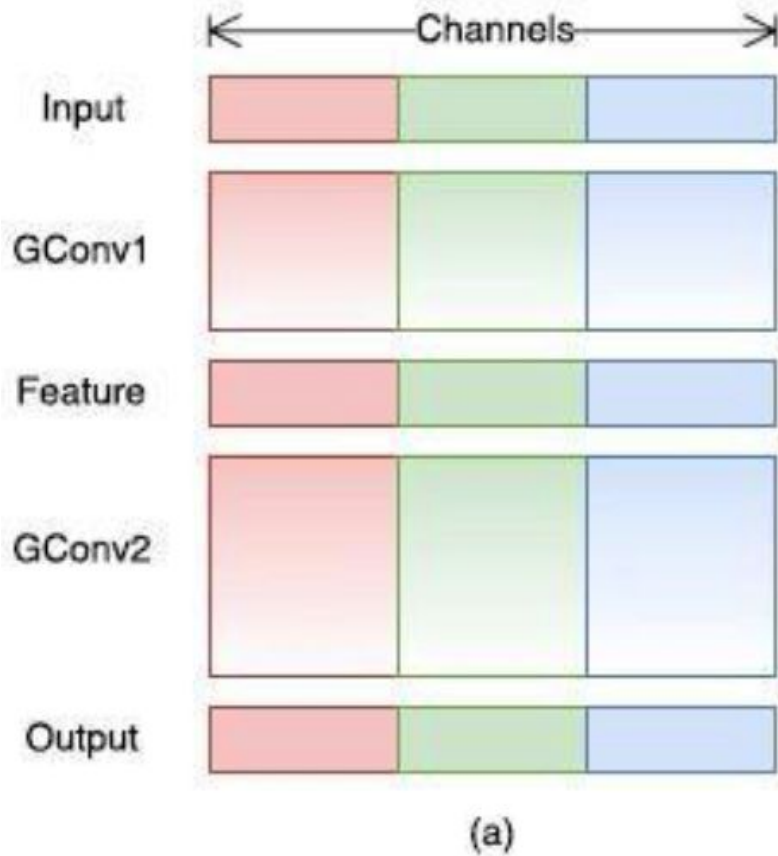
优点：

并行化、降低运算量、降低参数、Kernel对角化

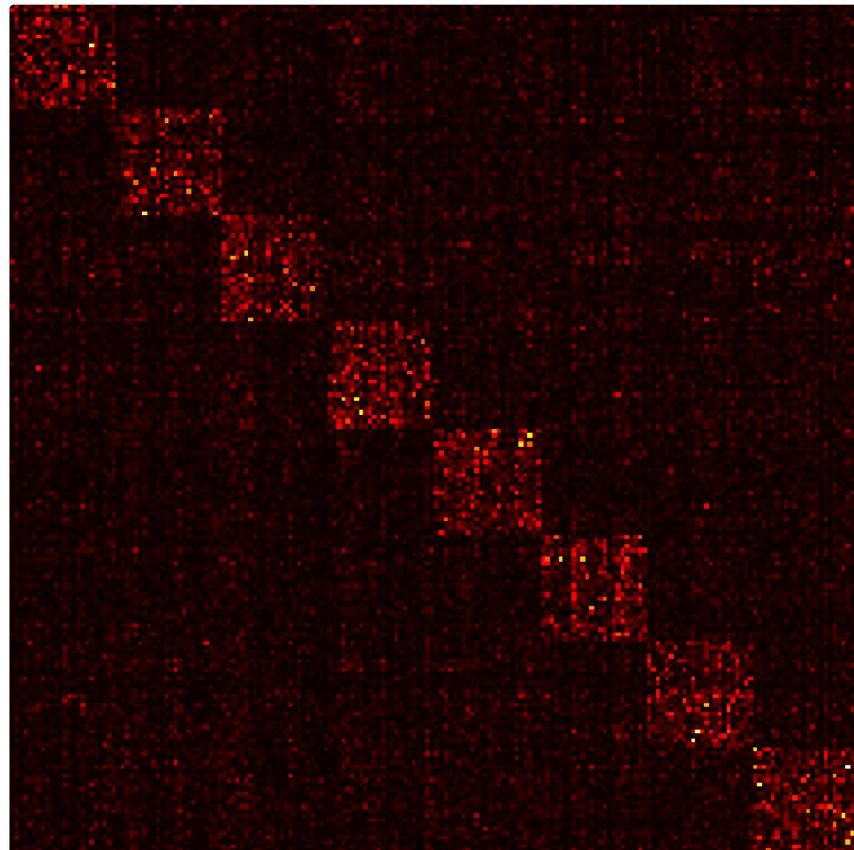
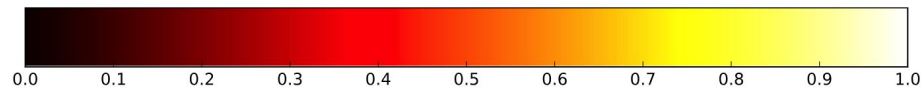
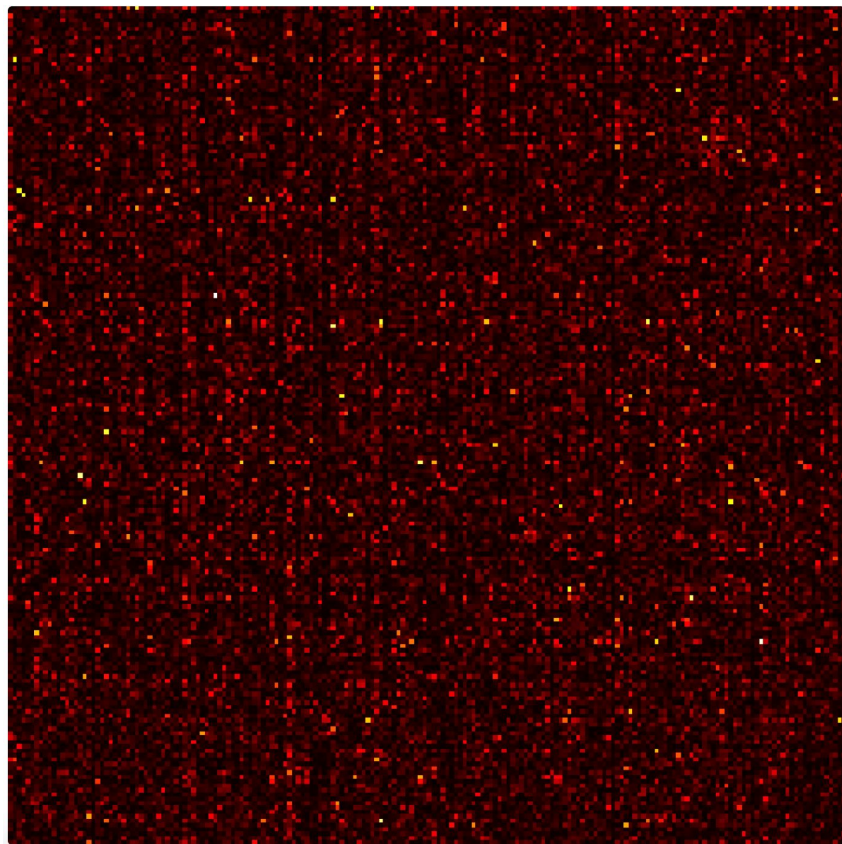
缺点：

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



# Group Conv

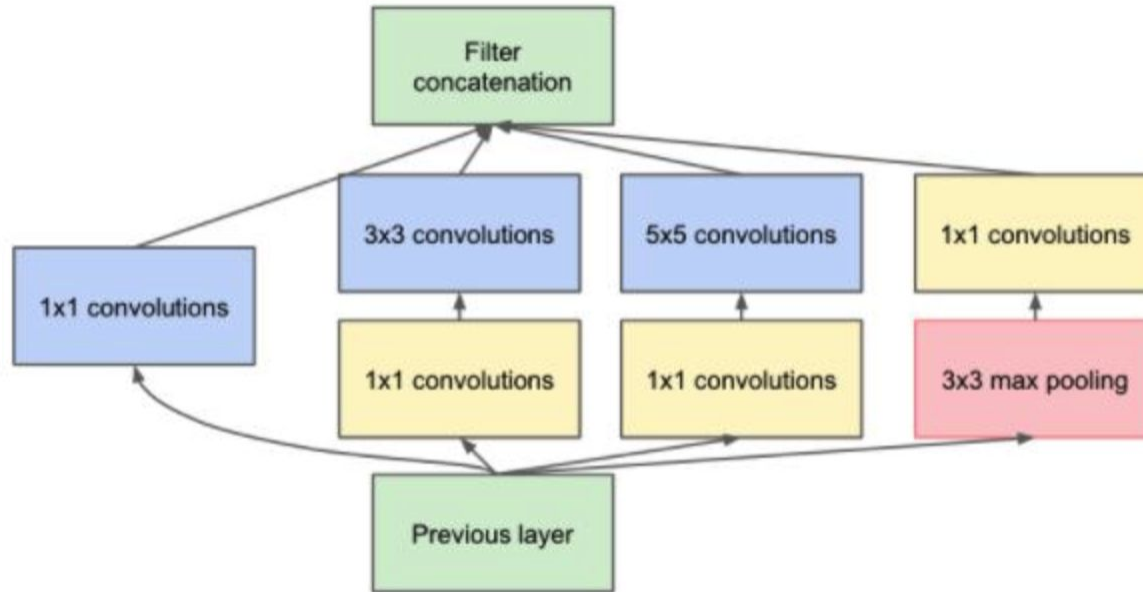
对于待搜索函数空间进行了限制

因此很难过拟合

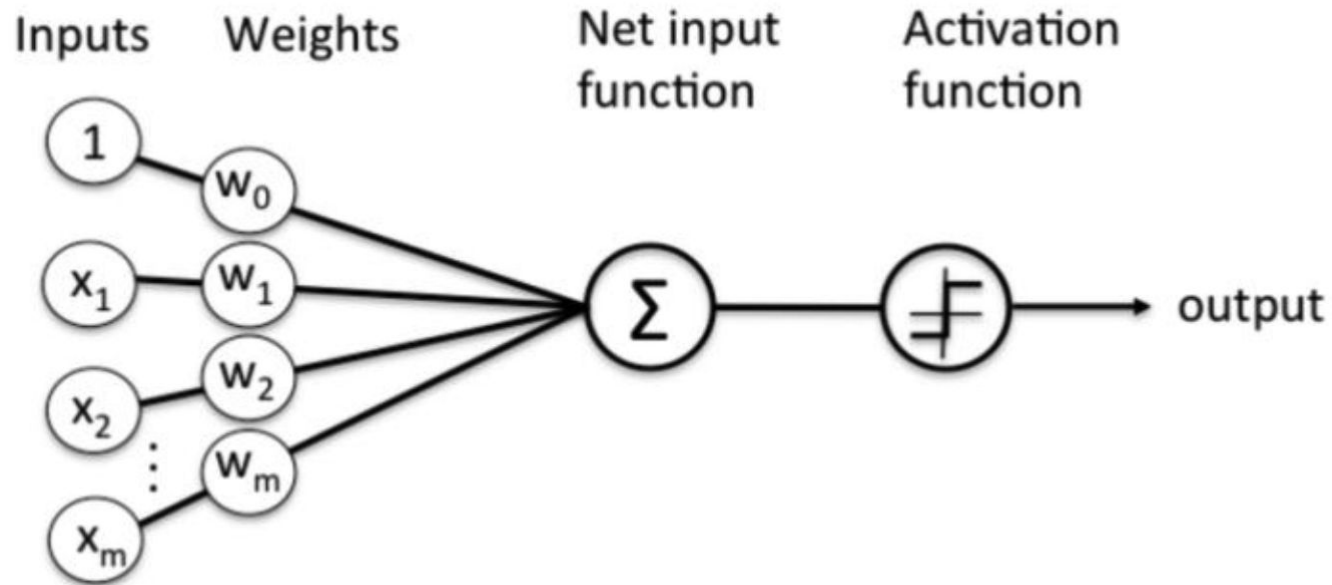
缺点也是对于搜索函数空间进行了限制,

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

# Split-Transform-Merge



# 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。

# 基数

左边是ResNet的结构, 右边是ResNext的基本结构

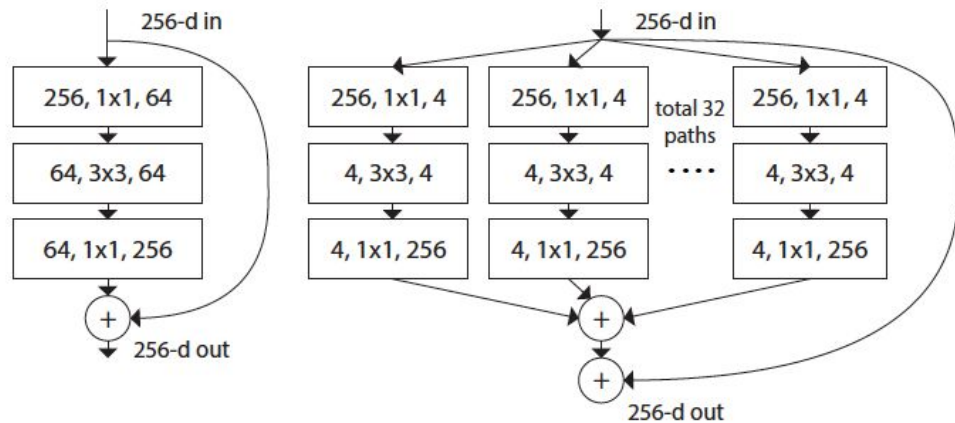


Figure 1. **Left:** A block of ResNet [14]. **Right:** A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

## 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
		$\begin{bmatrix} 1\times 1, 64 \\ 3\times 3, 64 \\ 1\times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times 1, 128 \\ 3\times 3, 128, C=32 \\ 1\times 1, 256 \end{bmatrix} \times 3$
conv3	28×28	$\begin{bmatrix} 1\times 1, 128 \\ 3\times 3, 128 \\ 1\times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times 1, 256 \\ 3\times 3, 256, C=32 \\ 1\times 1, 512 \end{bmatrix} \times 4$
conv4	14×14	$\begin{bmatrix} 1\times 1, 256 \\ 3\times 3, 256 \\ 1\times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times 1, 512 \\ 3\times 3, 512, C=32 \\ 1\times 1, 1024 \end{bmatrix} \times 6$
conv5	7×7	$\begin{bmatrix} 1\times 1, 512 \\ 3\times 3, 512 \\ 1\times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times 1, 1024 \\ 3\times 3, 1024, C=32 \\ 1\times 1, 2048 \end{bmatrix} \times 3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		<b>25.5</b> ×10 <sup>6</sup>	<b>25.0</b> ×10 <sup>6</sup>
FLOPs		<b>4.1</b> ×10 <sup>9</sup>	<b>4.2</b> ×10 <sup>9</sup>

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

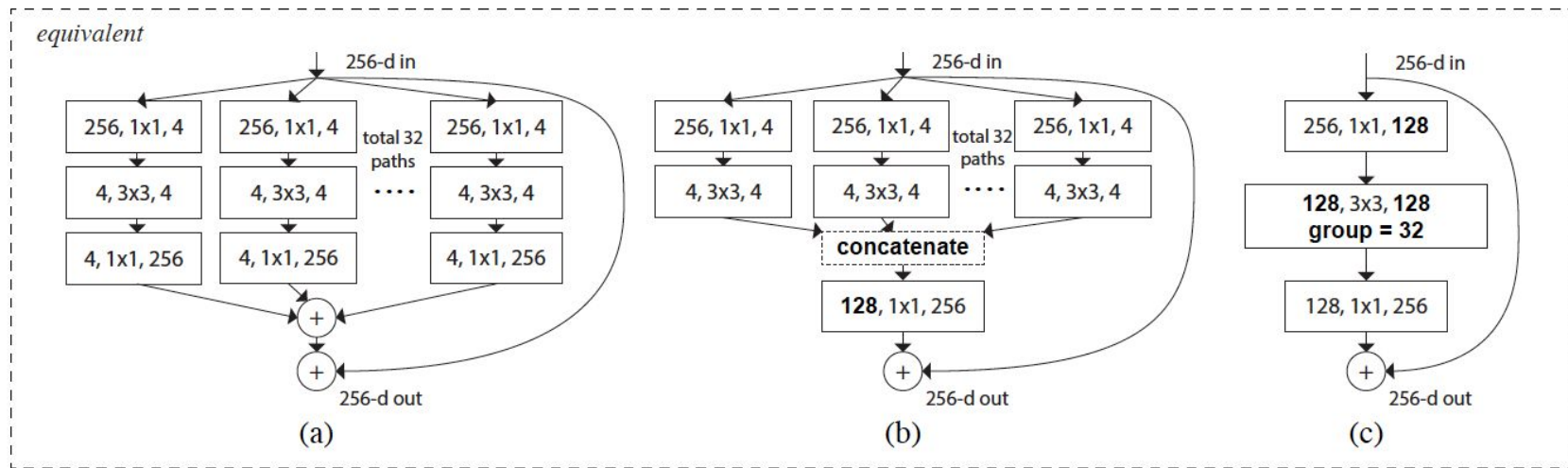


Figure 3. Equivalent building blocks of ResNeXt. **(a)**: Aggregated residual transformations, the same as Fig. 1 right. **(b)**: A block equivalent to (a), implemented as early concatenation. **(c)**: A block equivalent to (a,b), implemented as grouped convolutions [24]. Notations in **bold** text highlight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

# 实验

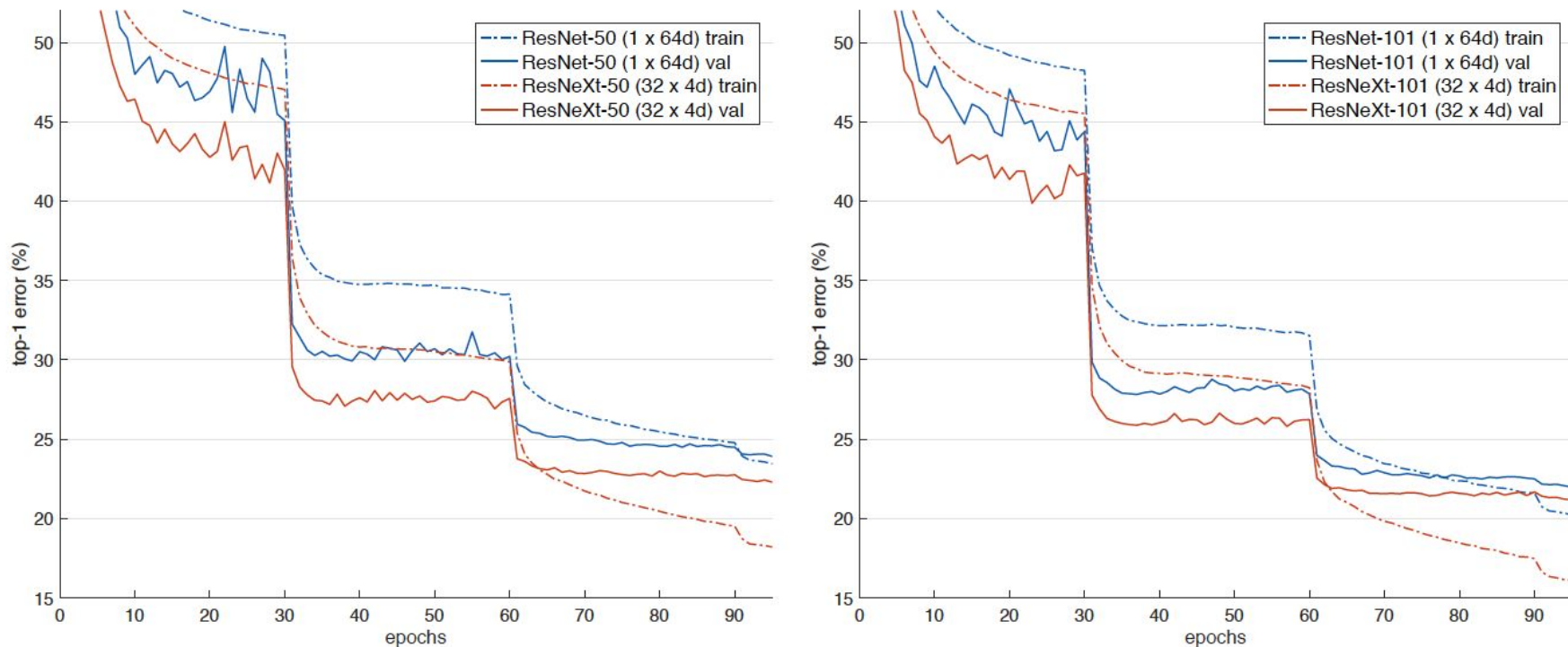


Figure 5. Training curves on ImageNet-1K. **(Left)**: ResNet/ResNeXt-50 with preserved complexity ( $\sim 4.1$  billion FLOPs,  $\sim 25$  million parameters); **(Right)**: ResNet/ResNeXt-101 with preserved complexity ( $\sim 7.8$  billion FLOPs,  $\sim 44$  million parameters).



	setting	top-1 error (%)
ResNet-50	$1 \times 64d$	23.9
ResNeXt-50	$2 \times 40d$	23.0
ResNeXt-50	$4 \times 24d$	22.6
ResNeXt-50	$8 \times 14d$	22.3
ResNeXt-50	$32 \times 4d$	<b>22.2</b>
ResNet-101	$1 \times 64d$	22.0
ResNeXt-101	$2 \times 40d$	21.7
ResNeXt-101	$4 \times 24d$	21.4
ResNeXt-101	$8 \times 14d$	21.3
ResNeXt-101	$32 \times 4d$	<b>21.2</b>

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