一 引言

自2012年AlexNet以来，卷积神经网络（简称CNN）在图像分类、图像分割、目标检测等领域获得广泛应用。随着性能的要求越来越高，AlexNet已经无法满足大家的需求，于是乎各路大牛纷纷提出性能更优越的CNN网络，如VGG，GoogLeNet，ResNet，DenseNet等。由于神经网络的性质，为了获得更好的性能，网络层数不断增加，从7层AlexNet到16层VGG，再从16层VGG到GoogLeNet的22层，再从22层到152层ResNet，更有上千层的ResNet和DenseNet。虽然网络性能得到了提高，但随之而来的就是效率问题。

效率问题主要是**模型的存储问题**和**模型进行预测的速度问题**（以下简称速度问题）

第一，存储问题。数百层网络有着大量的权值参数，保存大量权值参数对设备的内存要求很高；

第二，速度问题。在实际应用中，往往是毫秒级别，为了达到实际应用标准，要么提高处理器性能（看英特尔的提高速度就知道了，这点暂时不指望），要么就减少计算量。

只有解决CNN效率问题，才能让CNN走出实验室，更广泛的应用于移动端。对于效率问题，通常的方法是进行模型压缩（Model Compression），即在已经训练好的模型上进行压缩，使得网络携带更少的网络参数，从而解决内存问题，同时可以解决速度问题。

相比于在已经训练好的模型上进行处理，轻量化模型模型设计则是另辟蹊径。轻量化模型模型设计，主要思想在于设计更高效的“网络计算方式”（主要针对卷

积方式），从而使网络参数减少的同时，不损失网络性能。

本文就近年提出的四个轻量化模型进行学习和对比，四个模型分别是：SqueezeNet, MobileNet, ShuffleNet, Xception

（PS： 以上四种模型都不是模型压缩方法！！）

以下是4个模型的作者团队及发表时间

网络	最早公开日期	发表情况	作者团队	
SqueezeNet	2016.02	ICLR-2017	伯克利&斯坦福	https://a
MobileNet	2016.04	CVPR-2017	Google	https://a
ShuffleNet	2016.06	N/A	Face++	https://a
Xception	2016.10	CVPR-2017	Google	https://a

其中ShuffleNet论文中引用了SqueezeNet、Xception、MobileNet；Xception 论文中引用了MobileNet

二 轻量化模型

由于这四种轻量化模型仅是在卷积方式上提出创新，因此本文仅对轻量化模型的创新点进行详细描述，对模型实验以及实现的细节感兴趣的朋友，请到论文中详细阅读。

2.1 SqueezeNet

SqueezeNet由伯克利&斯坦福的研究人员合作发表于ICLR-2017，论文标题：

《SQUEEZENET: ALEXNET-LEVEL ACCURACY WITH 50X FEWER PARAMETERS AND <0.5MB MODEL SIZE》

（http://blog.csdn.net/u011995719/article/details/78908755）

命名

从名字SqueezeNet就知道，本文的新意是squeeze，squeeze在SqueezeNet中表示一个squeeze层，该层采用1*1卷积核对上一层feature map进行卷积，主要目的是减少feature map的维数（维数即通道数，就是一个立方体的feature map，切成一片一片的，一共有几片）。

创新点

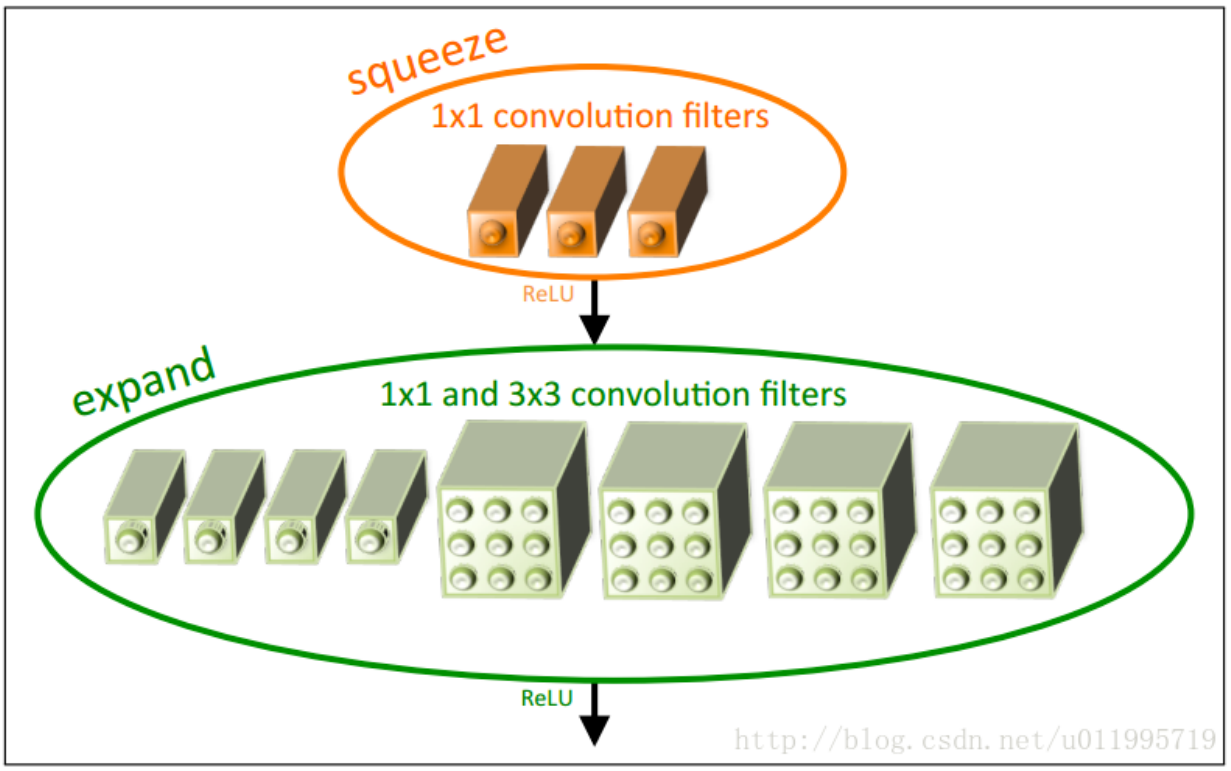
1. 采用不同于传统的卷积方式，提出fire module；fire module包含两部分：squeeze层+expand层

创新点与inception系列的思想非常接近！首先squeeze层，就是1*1卷积，其卷积核数要少于上一层feature map数，这个操作从inception系列开始就有了，并美其名曰压缩，个人觉得“压缩”更为妥当。

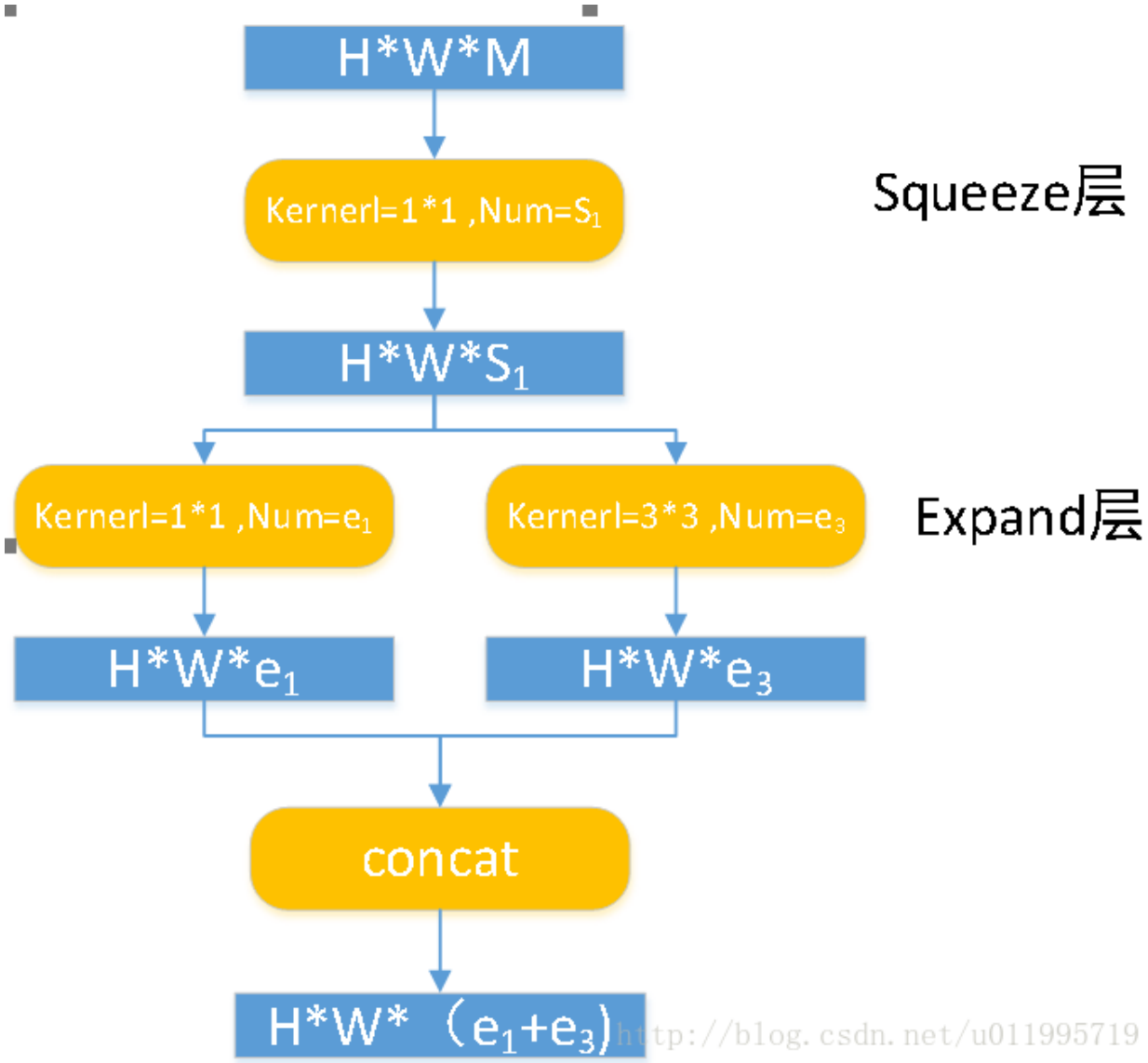
Expand层分别用1*1 和3*3 卷积，然后concat，这个操作再inception系列里面也有。

————————————————————分割线————————————————————

SqueezeNet的核心在于Fire module，Fire module 由两层构成，分别是squeeze层+expand层，如下图1所示，squeeze层是一个1*1卷积核的卷积层，expand层是1*1 和3*3卷积核的卷积层，expand层中，把1*1 和3*3 得到的feature map 进行concat。



具体操作情况如下图所示：



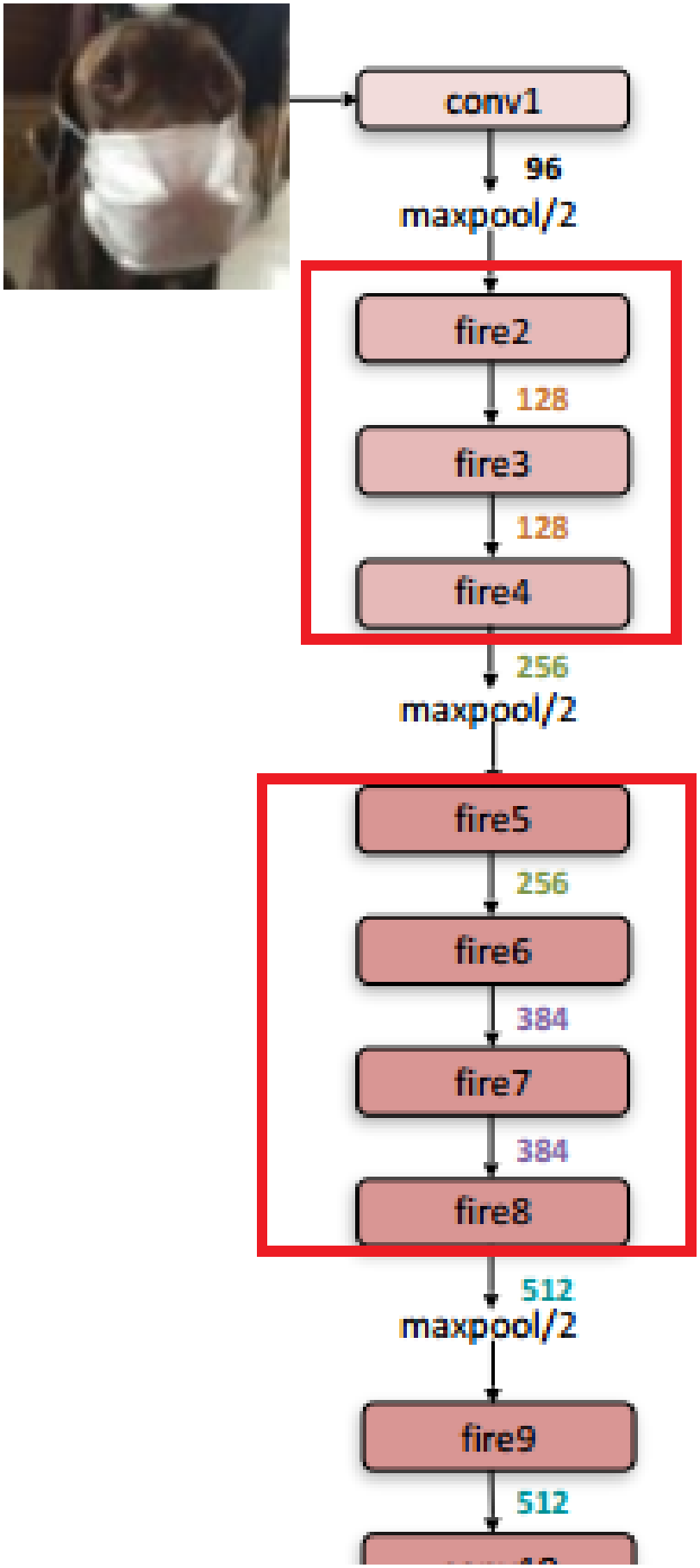
Fire module输入的feature map为H*W*M的，输出的feature map为H*M*(e1+e3)，可以看到feature map的分辨率是不变的，变的仅是维数，也就是通道数，这一点和VGG的思想一致。

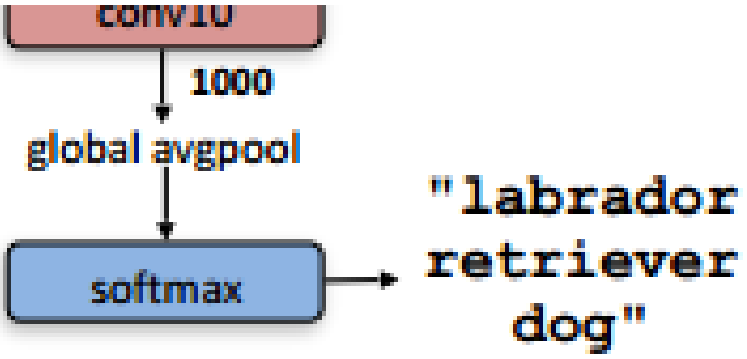
首先，H*W*M的feature map经过Squeeze层，得到S1个feature map，这里的S1均是小于M的，以达到“压缩”的目的，详细思想可参考Google的Inception系列。

其次，H*W*S1的特征图输入到Expand层，分别经过1*1卷积层和3*3卷积层进行卷积，再将结果进行concat，得到Fire module的输出，为 H*M*(e1+e3)的feature map。

fire模块有三个可调参数：S1，e1，e3，分别代表卷积核的个数，同时也表示对应输出feature map的维数，在本文提出的SqueezeNet结构中，e1=e3=4s1 。

讲完SqueezeNet的核心——Fire module，看看SqueezeNet的网络结构，如下图所示：





网络结构设计思想，同样与VGG的类似，堆叠的使用卷积操作，只不过这里堆叠的使用本文提出的Fire module(图中用红框部分)

看看Squezeenet的参数数量以及性能：

Table 2: Comparing SqueezeNet to model compression approaches. By *model size*, we mean the number of bytes required to store all of the parameters in the trained model.

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%

在这里可以看到，论文题目中提到的小于0.5M，是采用了Deep Compression进行模型压缩之后的结果！！

看了上图再回头看一看论文题目：

SqueezeNet ： AlexNet-level accuracy with 50x fewer parameters and <0.5MB

标！题！党！ SqueezeNet < 0.5MB, 这个是用了别的模型压缩技术获得的，很容易让人误以为SqueezeNet可以压缩模型！！

SqueezeNet小结：

1 Fire module 与GoogLeNet思想类似，采用1*1卷积对feature map的维数进行“压缩”，从而达到减少权值参数的目的；

2 采用与VGG类似的思想——堆叠的使用卷积，这里堆叠的使用Fire module

SqueezeNet与GoogLeNet和VGG的关系很大！

2.2 MobileNet

MobileNet 由Google团队提出，发表于CVPR-2017，论文标题：

《MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications》
(http://blog.csdn.net/u011995719/article/details/78850275)

命名

MobileNet的命名是从它的应用场景考虑的，顾名思义就是能够在移动端使用的网络模型。

创新点

1. 采用名为depth-wise separable convolution 的卷积方式代替传统卷积方式，以达到减少网络权值参数的目的。

通过采用depth-wise convolution的卷积方式，达到：1.减少参数数量 2.提升运算速度。（这两点是要区别开的，参数少的不一定运算速度快！）

depth-wise convolution不是MobileNet提出来的，也是借鉴，文中给的参考文献是 2014年的博士论文——《L. Sifre. Rigid-motion scattering for image classification. hD thesis, Ph. D. thesis, 2014》

depth-wise convolution 和 group convolution其实是一样的，都是一个卷积核负责一部分feature map，每个feature map只被一个卷积核卷积。

————————————————- 分割线————————————————

MobileNets精华在于卷积方式——depth-wise separable convolution；采用depth-wise separable convolution，会涉及两个超参：Width Multiplier和Resolution Multiplier这两个超参只是方便于设置要网络要设计为多小，方便于量化模型大小。

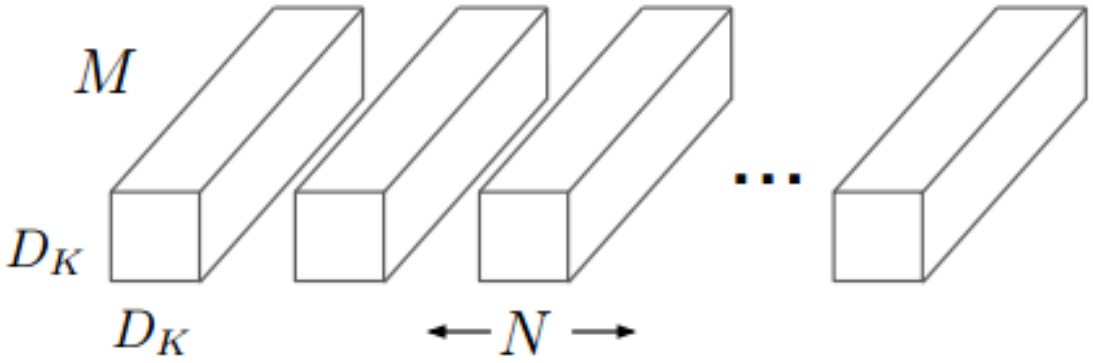
depth-wise convolution是将标准卷积分成两步：第一步 Depthwise convolution,即逐通道的卷积，**一个卷积核负责一个通道，一个通道只被一个卷积核“滤波”；**

第二步，Pointwise convolution，将depthwise convolution得到的feature map再“串”起来 ，注意这个“串”是很重要的。“串”作何解？为什么还需要pointwise convolution？作者说：However it only filters input channels, it does not combine them to create new features. Soan additional layer that computes a linear combination ofthe output of depthwise convolution via 1 × 1 convolutionis needed in order to generate these new features。

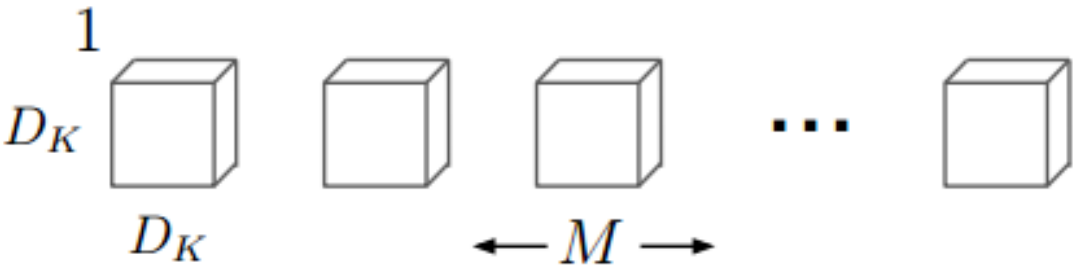
首先要承认一点：输出的每一个feature map要包含输入层所有feature map的信息。仅采用depthwise-convolution，是没办法做到这点,因此需要pointwise convolution的辅助。

“输出的每一个feature map要包含输入层所有feature map的信息” 这个是所有采用depth-wise convolution操作的网络都要去解决的问题，ShuffleNet中的命名就和这个有关！详细请看2.3

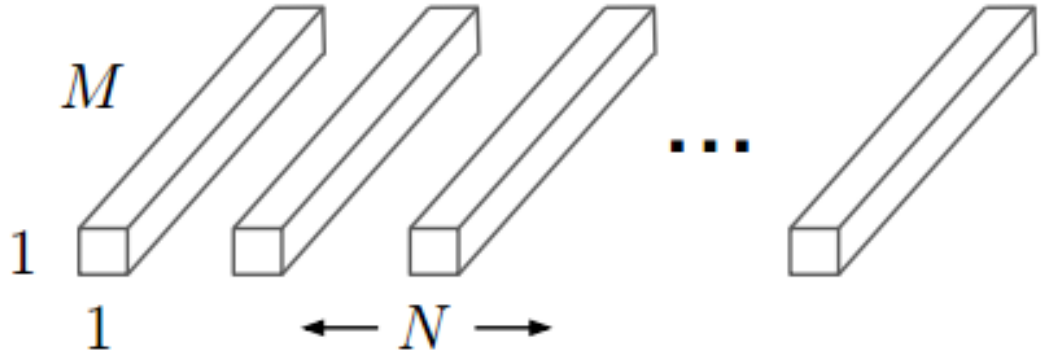
Standard convolution、depthwise convolution和pointwiseconvolution示意图如下：



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the con-

其中输入的feature map有M个，输出的feature map有N个。

Standard convolution呢，是采用N个大小为DK*DK的卷积核进行操作（注意卷积核大小是DK*DK, DK*DK*M是具体运算时候的大小！）

而depthwise convolution + pointwise convolution需要的卷积核呢？

Depthwise convolution ：一个卷积核负责一个通道，一个通道只被一个卷积核卷积；则这里有M个DK*DK的卷积核；

Pointwise convolution：为了达到输出N个feature map的操作，所以采用N个1*1的卷积核进行卷积，这里的卷积方式和传统的卷积方式是一样的，只不过采用了1*1的卷积核；其目的就是让新的每一个feature map包含有上一层各个feature map的信息！在此理解为将depthwise convolution的输出进行“串”起来。

————————————————- 分割线————————————————

下面举例讲解 Standard convolution、depthwise convolution和pointwise convolution 。

假设输入的feature map 是两个5*5的，即5*5*2；输出feature map数量为3，大小是3*3（因为这里采用3*3卷积核）即3*3*3。

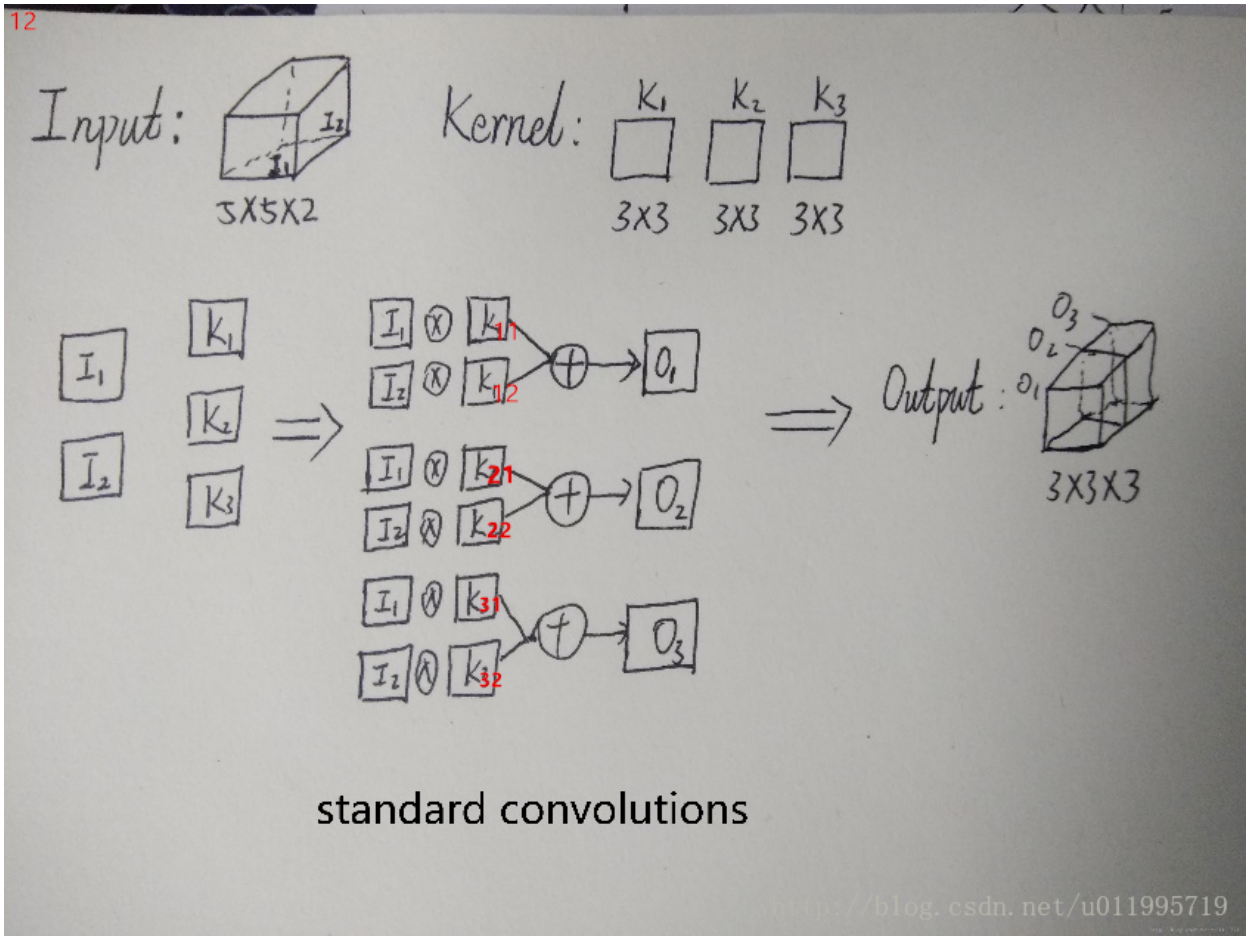
标准卷积，是将一个卷积核（3*3）复制M份（M=2），是让二维的卷积核（面包片）拓展到与输入feature map一样的面包块形状。例如，我们设置3个 3*3的卷积核，如下图Kernel所示，但是在实际计算当中，卷积核并不是3*3*3这么多，而是3*3*2*3 （ w*h*c_in*c_out） 。也就是上面所说的把二维的卷积核拓展到与feature map一样的面包块形状，如下图的K1 扩展成 K11,K12 。

（注：不是复制M份，因为每个二维卷积核的参数是不一样的，因此不是复制！感谢DSQ_17这位朋友指出错误。）

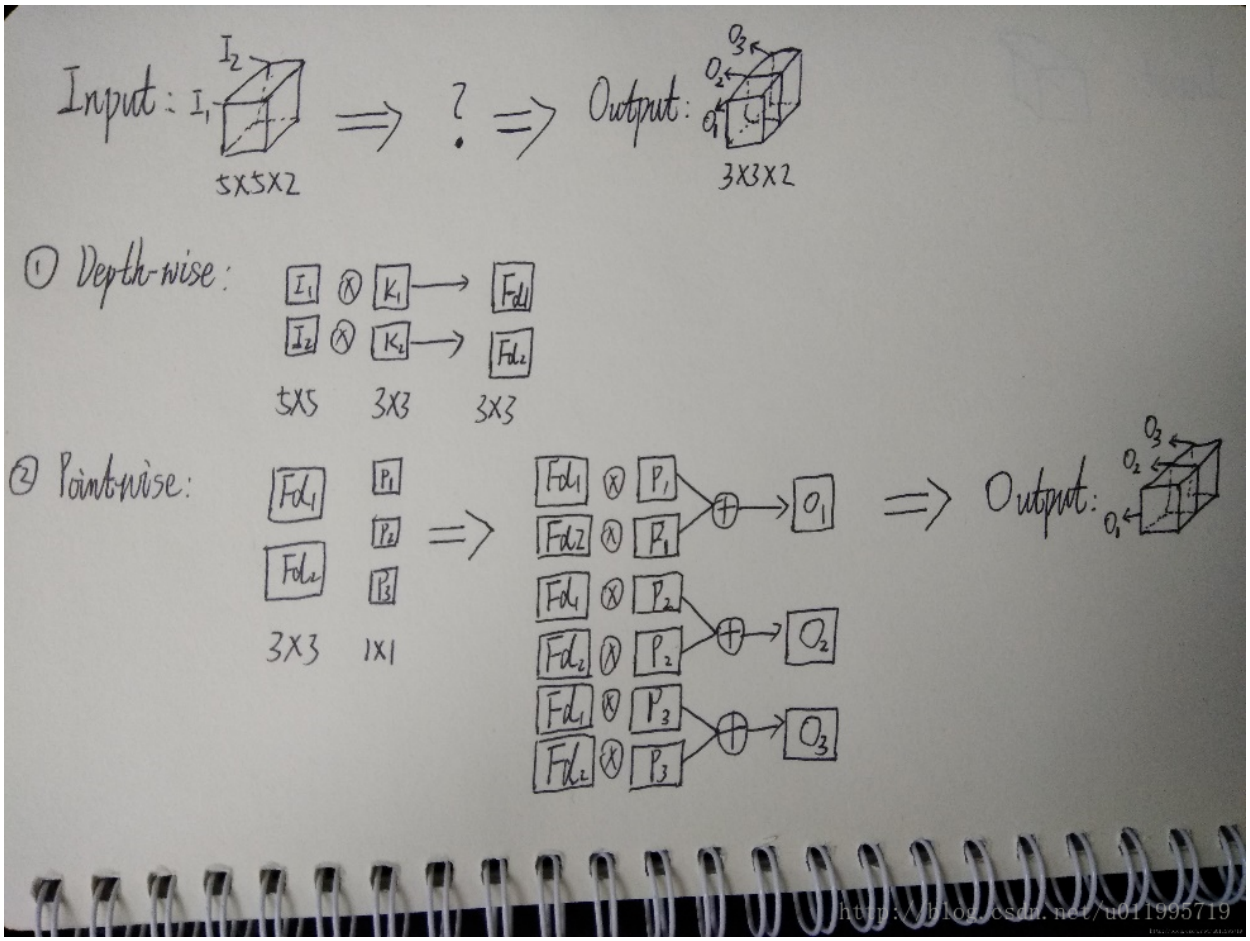
2018年3月15日补充：实际上，卷积核实际的尺寸应该是 w*h*c_in*c_out。往往，我们忽略掉c_in这个数，在设置卷积核数量时，也不会涉及到这个参数，但是在计算过程中是不能忽略的。

其中，w*h就是通常我们所说的卷积核大小，例如3*3，5*5，7*7等；c_out是平时我们讲的卷积核个数，例如该卷积层设置了64个卷积核，则c_out = 64；而c_in则是等于上一层的feature map的数量。

Standard过程如下图，X表示卷积，+表示对应像素点相加，可以看到对于O1来说，其与输入的每一个feature map都“发生关系”，包含输入的各个feature map的信息。



Depth-wise 过程如下图，而Depthwise并没有，可以看到depthwise convolution 得出的两个feature map——fd1 和fd2分别只与i1和i2 “发生关系” ，这就导致违背上面所承认的观点“输出的每一个feature map要包含输入层所有feature map的信息”，因而要引入pointwise convolution



那么计算量减少了多少呢？通过如下公式计算：

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

其中DK为标准卷积核大小，M是输入feature map通道数，DF为输入feature map大小，N是输出feature map大小。本例中，DK=3，M=2，DF=5，N=3 ， 参数的减少量主要就与卷积核大小DK有关。在本文MobileNet的卷积核采用DK=3，则大约减少了8~9倍计算量。

看看MobileNet的网络结构，MobileNet共28层，可以发现这里下采样的方式没有采用池化层，而是利用depth-wise convolution的时候将步长设置为2，达到下采样的目的。

Table 1. MobileNet Body Architecture

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

1.0 MobileNet-224 与GoogLeNet及VGG-16的对比：

Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

可以发现，相较于GoogLeNet，虽然参数差不多，都是一个量级的，但是在运算量上却小于GoogLeNet一个量级，这就得益于 depth-wise convolution ！

MobileNet小结：

1. 核心思想是采用depth-wise convolution操作，在相同的权值参数数量的情况下，相较于standard convolution操作，可以减少数倍的计算量，从而达到提升网络运算速度的目的。

    1. depth-wise convolution的思想非首创，借鉴于 2014年一篇博士论文：《L. Sifre. Rigid-motion scattering for image classification. hD

thesis, Ph. D. thesis, 2014》

2. 采用depth-wise convolution 会有一个问题，就是导致“信息流通不畅”，即输出的feature map仅包含输入的feature map的一部分，在这里，MobileNet采用了point-wise convolution解决这个问题。在后来，ShuffleNet采用同样的思想对网络进行改进，只不过把point-wise convolution换成了 channel shuffle，然后给网络美其名曰 ShuffleNet~ 欲知后事如何，请看 2.3 ShuffleNet

2.3 ShuffleNet

ShuffleNet 是Face++团队提出的，晚于MobileNet两个月在arXiv上公开。论文标题：

《ShuffleNet： An Extremely Efficient Convolutional Neural Network for Mobile Devices 》

命名

一看名字ShuffleNet，就知道shuffle是本文的重点，那么shuffle是什么？为什么要进行shuffle？

shuffle具体来说是channel shuffle，是将各部分的feature map的channel进行有序的打乱，构成新的feature map，以解决group convolution带来的“信息流通不畅”问题。（MobileNet是用point-wise convolution解决的这个问题）

因此可知道shuffle不是什么网络都需要用的，是有一个前提，就是采用了group convolution，才有可能需要shuffle！！ 为什么说是有可能呢？因为可以用point-wise convolution 来解决这个问题。

创新点

1. 利用group convolution 和 channel shuffle 这两个操作来设计卷积神经网络模型，以减少模型使用的参数数量。

group convolutiosn非原创，而channel shuffle是原创。 channel shuffle因group convolution 而起，正如论文中3.1标题： . Channel Shuffle for Group Convolution;

采用group convolution 会导致信息流通不当，因此提出channel shuffle，所以channel shuffle是有前提的，使用的话要注意！

对比一下MobileNet，采用shuffle替换掉1*1卷积(注意！是1*1 Conv，也就是point-wise convolution；特别注意，point-wise convolution和 1*1 GConv是不同的)，这样可以减少权值参数，而且是减少大量权值参数 ，因为在MobileNet中，1*1卷积层有较多的卷积核，并且计算量巨大，MobileNet每层的参数量和运算量如下图所示：

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%

————————————————- 分割线————————————————-

ShuffleNet的创新点在于利用了group convolution 和 channel shuffle，那么有必要看看group convolution 和channel shuffle

Group convolution

Group convolution 自Alexnet就有，当时因为硬件限制而采用分组卷积；之后在2016年的ResNeXt中，表明采用group convolution可获得高效的网络；再有Xception和MobileNet均采用depthwise convolution, 这些都是最近出来的一系

列轻量化网络模型。depth-wise convolution具体操作可见2.2 MobileNet里边有简介

如下图(a)所示，为了提升模型效率，采用group convolution，但会有一个副作用，即：“outputs from a certain channel are only derived from a small fraction of input channels.”

于是采用channel shuffle来改善各组间“信息流通不畅”问题，如下图(b)所示。

具体方法为：把各组的channel平均分为g（下图g=3）份，然后依次序的重新构成feature map

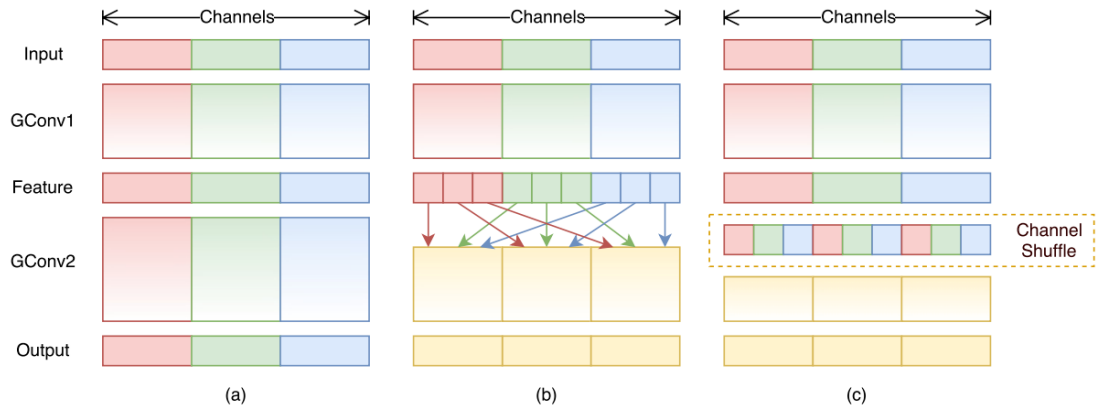


Figure 1. Channel shuffle with two stacked group convolutions. GConv stands for group convolution. a) two stacked convolution layers with the same number of groups. Each output channel only relates to the input channels within the group. No cross talk; b) input and output channels are fully related when GConv2 takes data from different groups after GConv1; c) an equivalent implementation to b) using channel shuffle.

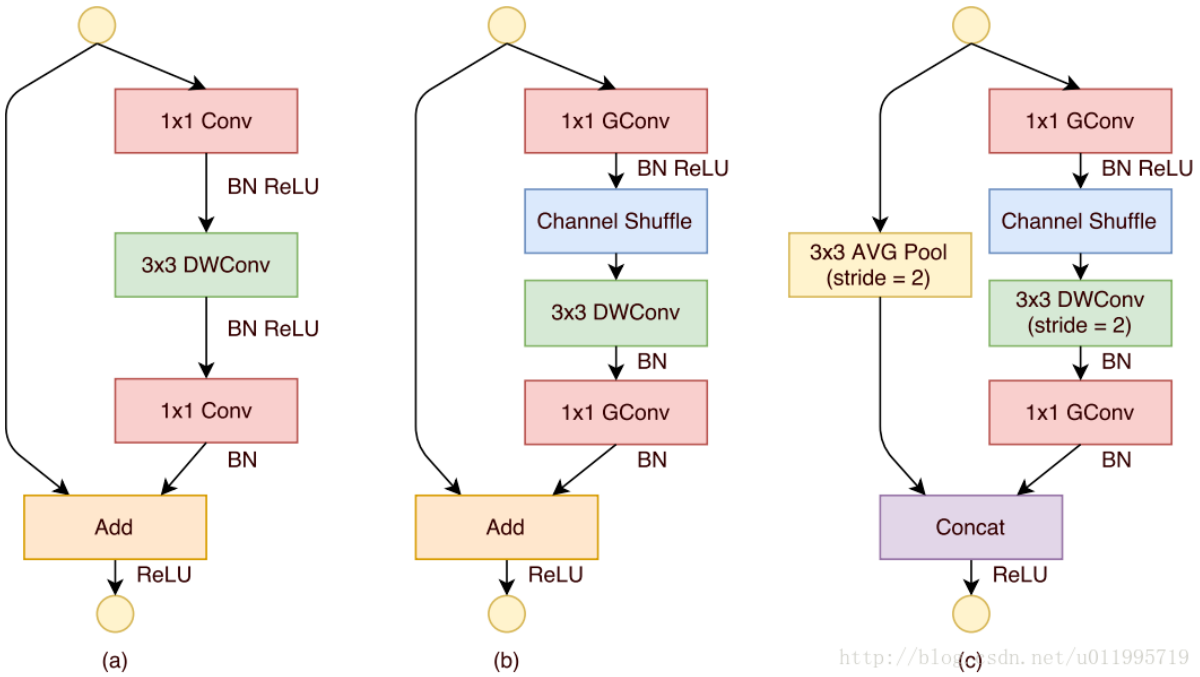
Channel shuffle 的操作非常简单，接下来看看ShuffleNet，ShuffleNet借鉴了Resnet的思想，从基本的resnet 的bottleneck unit 逐步演变得到 ShuffleNet的bottleneck unit，然后堆叠的使用ShuffleNet bottleneck unit获得ShuffleNet；

下图展示了ShuffleNet unit的演化过程

图(a)：是一个带有depthwise convolution的bottleneck unit；

图(b)：作者在(a)的基础上进行变化，对1*1 conv 换成 1*1 Gconv，并在第一个1*1 Gconv之后增加一个channel shuffle 操作；

图(c)： 在旁路增加了AVG pool，目的是为了减小feature map的分辨率；因为分辨率小了，于是乎最后不采用Add，而是concat，从而“弥补”了分辨率减小而带来的信息损失。



文中提到两次，对于小型网络，多多使用通道，会比较好。

“this is critical for small networks, as tiny networks usually have an insufficient number of channels to process the information”

所以，以后若涉及小型网络，可考虑如何提升通道使用效率

至于实验比较，并没有给出模型参数量的大小比较，而是采用了Complexity (MFLOPs)指标，在相同的Complexity (MFLOPs)下，比较ShuffleNet和各个网络，还专门和MobileNet进行对比，由于ShuffleNet相较于MobileNet少了1*1 Conv(注意！少了1*1 Conv，也就是point-wise convolution)，所以效率大大提高了嘛，贴个对比图随意感受一下好了

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

Table 5. ShuffleNet vs. MobileNet [12] on ImageNet Classification

ShuffleNet小结：

1.与MobileNet一样采用了depth-wise convolution，但是针对 depth-wise convolution带来的副作用——“信息流通不畅”，ShuffleNet采用了一个channel shuffle 操作来解决。

1. 在网络拓扑方面，ShuffleNet采用的是resnet的思想，而mobielnet采用的是VGG的思想，2.1 SqueezeNet也是采用VGG的堆叠思想

2.4 Xception

Xception并不是真正意义上的轻量化模型，只是其借鉴depth-wise convolution，而depth-wise convolution又是上述几个轻量化模型的关键点，所以在此一并介绍，其思想非常值得借鉴。

Xception是Google提出的，arXiv 的V1 于2016年10月公开。论文标题：

《Xception: Deep Learning with Depthwise Separable Convolutions 》

命名

Xception是基于Inception-V3的，而X表示Extreme，为什么是Extreme呢？因为Xception做了一个加强的假设，这个假设就是：

we make the following hypothesis: that the mapping of cross-channels correlations and spatial correlations in the feature maps of

convolutional neural networks can be entirely decoupled

创新点

1. 借鉴（非采用）depth-wise convolution 改进Inception V3

既然是改进了Inception v3，那就得提一提关于inception的一下假设（思想）了。

“the fundamental hypothesis behind Inception is that cross-channel correlations and spatial correlations are sufficiently decoupled that it is preferable not to map them jointly”

简单理解就是说，卷积的时候要将通道的卷积与空间的卷积进行分离，这样会比较好。（没有理论证明，只有实验证明，就当它是定理，接受就好了，现在大多数神经网络的论文都这样。

————————————————分割线————————————————

既然是在Inception V3上进行改进的，那么Xception是如何一步一步的从Inception V3演变而来。

下图1 是Inception module，图2是作者简化了的 inception module（就是只保留1*1的那条“路”，如果带着avg pool，后面怎么进一步假设嘛~~~）

Figure 1. A canonical Inception module (Inception V3).

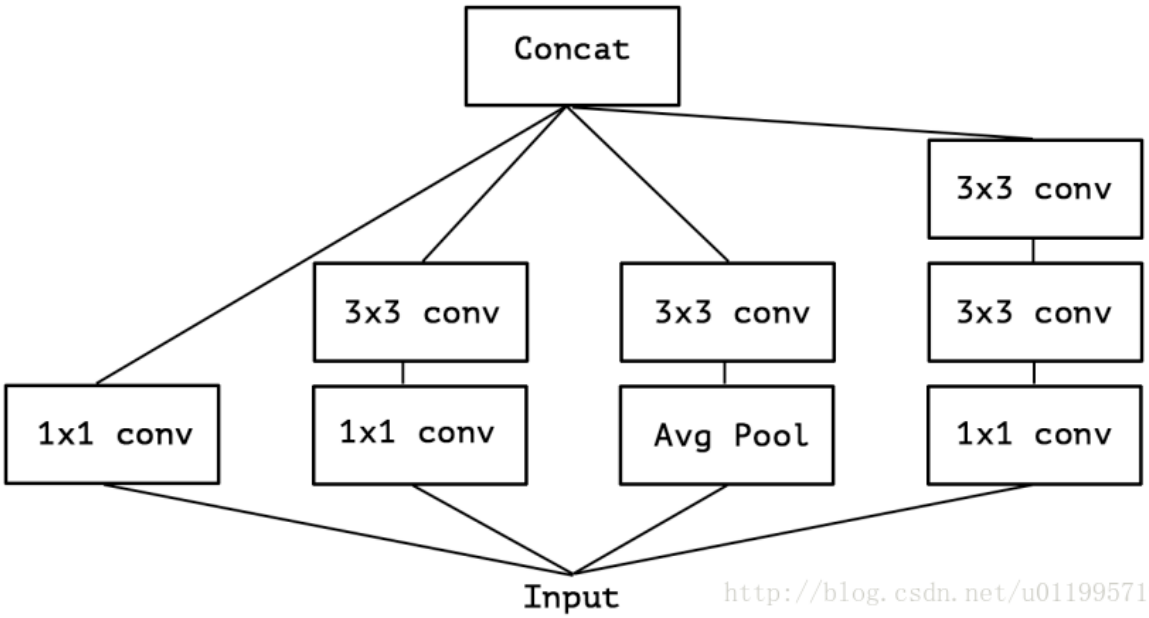
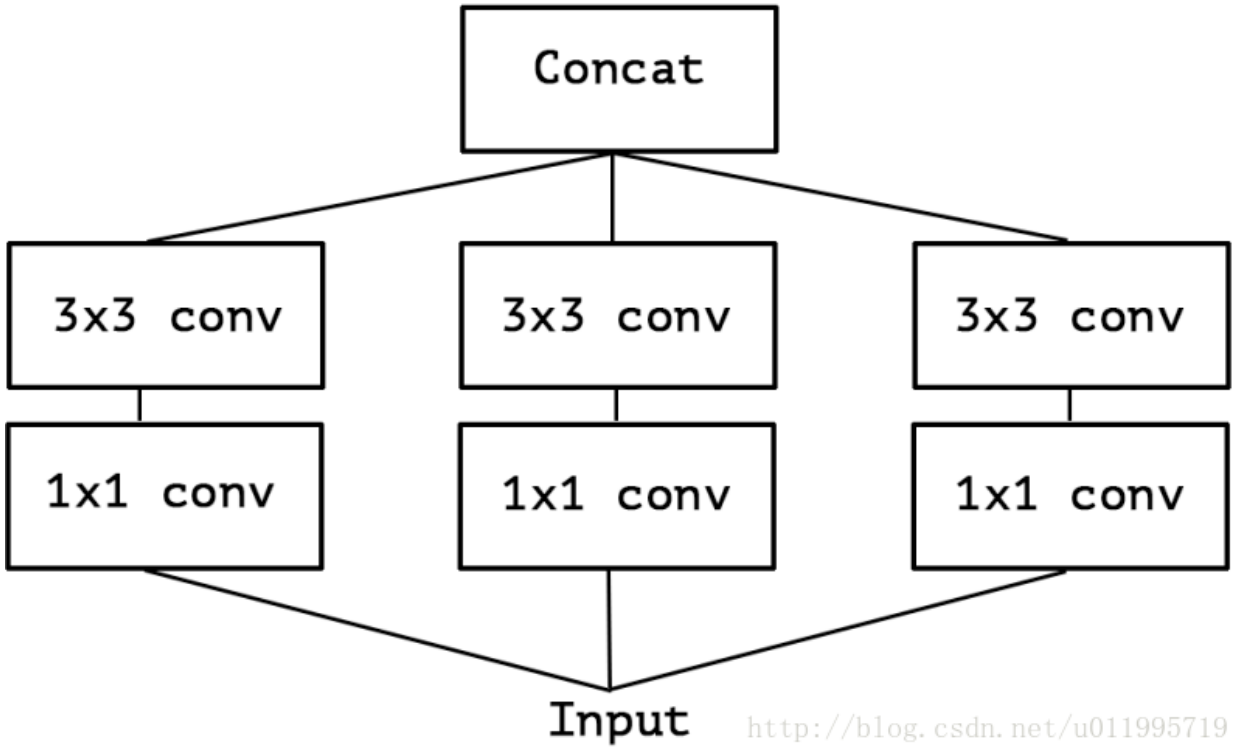


Figure 2. A simplified Inception module.



假设出一个简化版inception module之后，再进一步假设，把第一部分的3个1*1卷积核统一起来，变成一个1*1的，后面的3个3*3的分别“负责”一部分通道，如图3所示； 最后提出“extreme” version of an Inception ，module

Xception登场，， 先用1*1卷积核对各通道之间（cross-channel）进行卷积，如图4所示，

Figure 3. A strictly equivalent reformulation of the simplified Inception module.

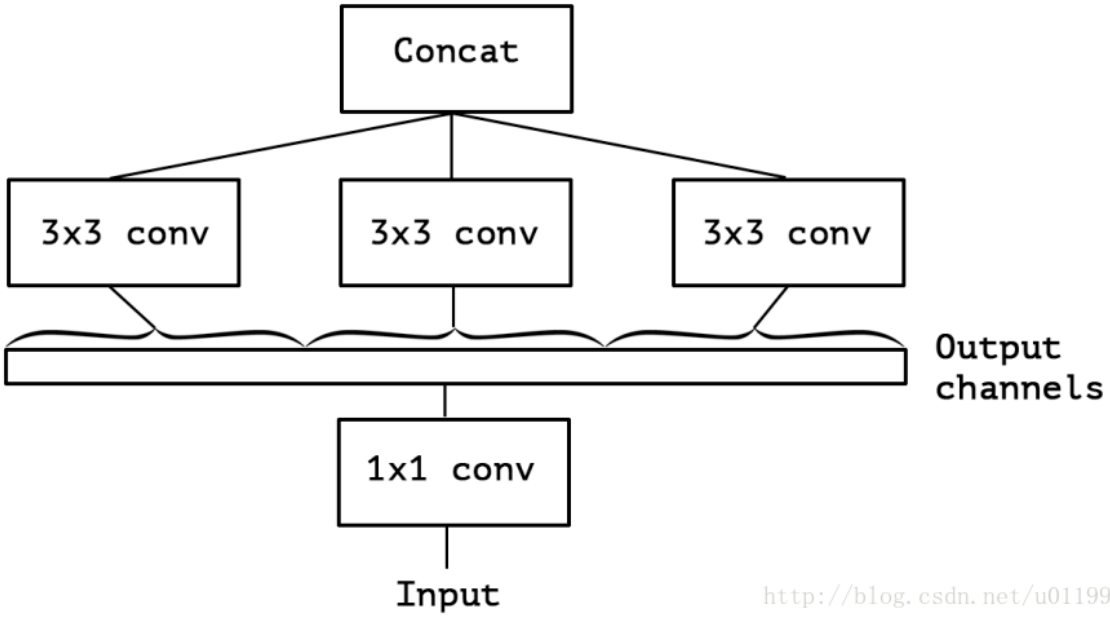
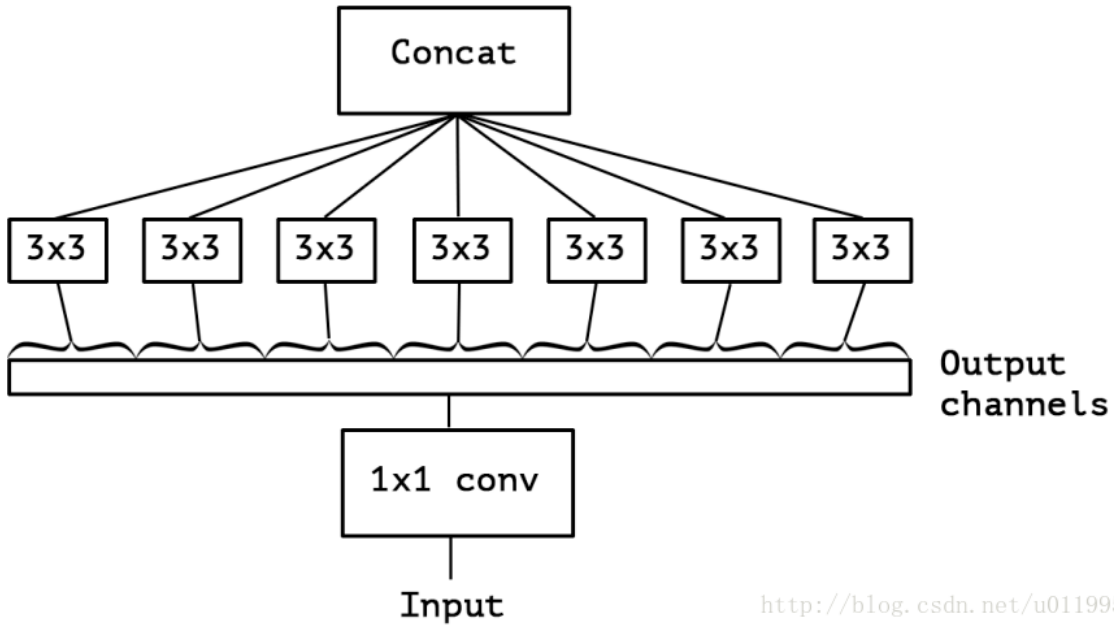


Figure 4. An “extreme” version of our Inception module, with one spatial convolution per output channel of the 1x1 convolution.

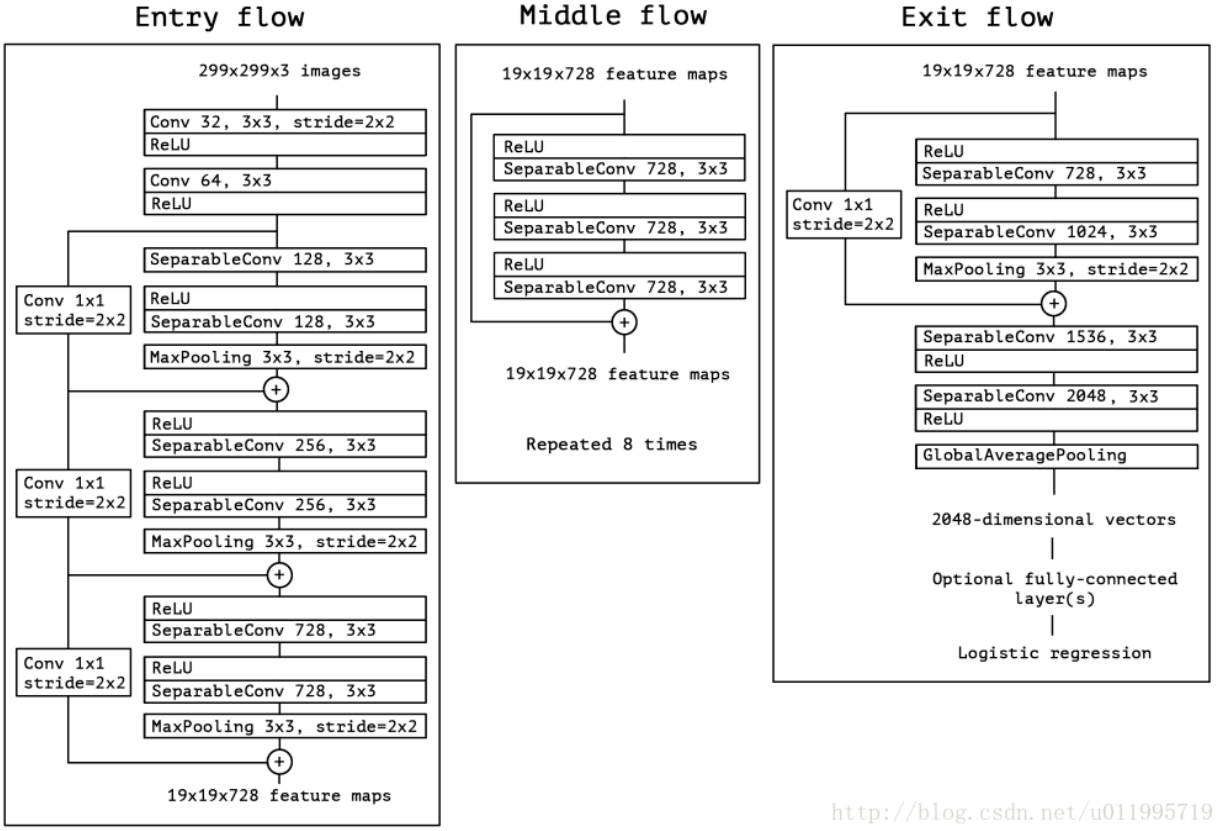


作者说了，这种卷积方式和depth-wise convolution 几乎一样。Depth-wise convolution 较早用于网络设计是来自：Rigid-Motion Scatteringfor Image Classification，但是具体是哪一年提出，不得而知；至少2012年就有相关研究，再比如说AlexNet，由于内存原因，AlexNet分成两组卷积 ；想深入了解Depth-wise convolution的可以查阅本论文2.Prior work，里面有详细介绍。

Xception是借鉴Rigid-Motion Scatteringfor Image Classification 的Depth-wise convolution，是因为Xception与原版的Depth-wise convolution有两个不同之处

第一个：原版Depth-wise convolution，先逐通道卷积，再1*1卷积；而Xception是反过来，先1*1卷积，再逐通道卷积；

第二个：原版Depth-wise convolution的两个卷积之间是不带激活函数的，而Xception在经过1*1卷积之后会带上一个Relu的非线性激活函数；



Xception 结构如上图所示，共计36层分为Entry flow; Middle flow; Exit flow;

Entry flow 包含 8个conv；Middle flow 包含 3*8 =24个conv；Exit flow包含4个conv，所以Xception共计36层

文中Xception实验部分是非常详细的，实现细节可参见论文。

Table 1. Classification performance comparison on ImageNet (single crop, single model). VGG-16 and ResNet-152 numbers are only included as a reminder. The version of Inception V3 being benchmarked does not include the auxiliary tower.

	Top-1 accuracy	Top-5 accuracy
VGG-16	0.715	0.901
ResNet-152	0.770	0.933
Inception V3	0.782	0.941
Xception	**0.790**	**0.945**

Xception小结：

Xception是基于Inception-V3，并结合了depth-wise convolution，这样做的好处是提高网络效率，以及在同等参数量的情况下，在大规模数据集上，效果要优于Inception-V3。这也提供了另外一种“轻量化”的思路：在硬件资源给定的情况下，尽可能的增加网络效率和性能，也可以理解为充分利用硬件资源。

三 网络对比

本文简单介绍了四个轻量化网络模型，分别是SqueezeNet、 MobileNet、ShuffleNet和Xception，前三个是真正意义上的轻量化网络，而Xception是为提升网络效率，在同等参数数量条件下获得更高的性能。

在此列出表格，对比四种网络是如何达到网络轻量化的。

网络	实现轻量化技巧
SqueezeNet	1*1卷积核“压缩”feature map数量
MobileNet	Depth-wise convolution
ShuffleNet	Depth-wise convolution
Xception	修改的Depth-wise convolution

这么一看就发现，**轻量化主要得益于depth-wise convolution**，因此大家可以考虑采用depth-wise convolution 来设计自己的轻量化网络， 但是要注意“信息流通不畅问题”

解决“信息流通不畅”的问题，MobileNet采用了point-wise convolution，ShuffleNet采用的是channel shuffle。MobileNet相较于ShuffleNet使用了更多的卷积，计算量和参数量上是劣势，但是增加了非线性层数，理论上特征更抽象，更高级了；ShuffleNet则省去point-wise convolution，采用channel shuffle，简单明了，省去卷积步骤，减少了参数量。

学习了几个轻量化网络的设计思想，可以看到，并没有突破性的进展，都是借鉴或直接使用前几年的研究成果。希望广大研究人员可以设计出更实在的轻量化网络。

最后讲一下读后感，也可以认为是发(Shui)论文的idea：

1.继续采用depth-wise convolution，主要设计一个方法解决“信息流通不畅”问题， 然后冠以美名XX-Net。（看看ShuffleNet就是）

2.针对depth-wise convolution作文章，卷积方式不是千奇百怪么？各种卷积方式可参考Github（https://github.com/vdumoulin/conv_arithmetic），挑一个或者几个，结合起来，只要参数量少，实验效果好，就可以发(Shui)论文。

3.接着第2，如果设计出来一个新的卷积方式，如果也存在一些“副作用”，再想一个方法解决这个副作用，再美其名曰XX-Net。就是自己“挖”个坑，自己再填上去。

以上纯属读后感，写得比较“随意”，如果有什么地方描述有误，请大家指出来，欢迎大家前来讨论~

转载请注明出处：

http://blog.csdn.net/u011995719/article/details/79100582