论文：ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices

论文链接：https://arxiv.org/abs/1707.01083

算法详解：

ShuffleNet是Face++的一篇关于降低深度网络计算量的论文，号称是可以在移动设备上运行的深度网络。这篇文章可以和MobileNet、Xception和ResNeXt结合来看，因为有类似的思想。卷积的group操作从AlexNet就已经有了，当时主要是解决模型在双GPU上的训练。ResNeXt借鉴了这种group操作改进了原本的ResNet。MobileNet则是采用了depthwise separable convolution代替传统的卷积操作，在几乎不影响准确率的前提下大大降低计算量，具体可以参考MobileNets-深度学习模型的加速。Xception主要也是采用depthwise separable convolution改进Inception v3的结构。

该文章主要采用channel shuffle、pointwise group convolutions和depthwise separable convolution来修改原来的ResNet单元，接下来依次讲解。

channel shuffle的思想可以看下面的Figure 1。这就要先从group操作说起，一般卷积操作中比如输入feature map的数量是N，该卷积层的filter数量是M，那么M个filter中的每一个filter都要和N个feature map的某个区域做卷积，然后相加作为一个卷积的结果。假设你引入group操作，设group为g，那么N个输入feature map就被分成g个group，M个filter就被分成g个group，然后在做卷积操作的时候，第一个group的M/g个filter中的每一个都和第一个group的N/g个输入feature map做卷积得到结果，第二个group同理，直到最后一个group，如Figure1（a）。不同的颜色代表不同的group，图中有三个group。这种操作可以大大减少计算量，因为你每个filter不再是和输入的全部feature map做卷积，而是和一个group的feature map做卷积。但是如果多个group操作叠加在一起，如Figure1（a）的两个卷积层都有group操作，显然就会产生边界效应，什么意思呢？就是某个输出channel仅仅来自输入channel的一小部分。这样肯定是不行的的，学出来的特征会非常局限。于是就有了channel shuffle来解决这个问题，先看Figure1（b），在进行GConv2之前，对其输入feature map做一个分配，也就是每个group分成几个subgroup，然后将不同group的subgroup作为GConv2的一个group的输入，使得GConv2的每一个group都能卷积输入的所有group的feature map，这和Figure1（c）的channel shuffle的思想是一样的。

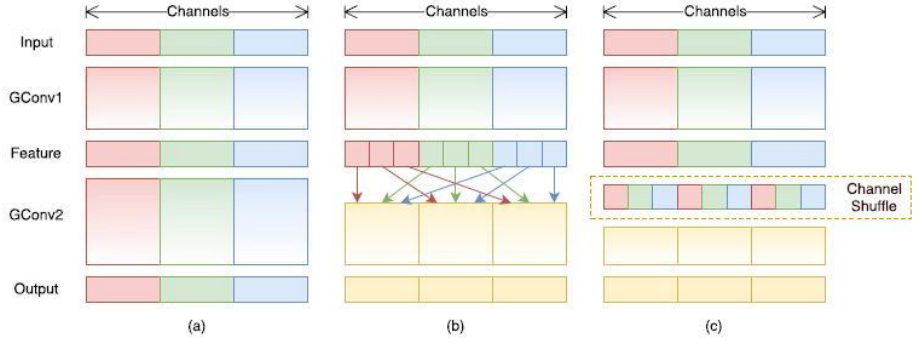


Figure 1: Channel shuffle with two stacked group convolutions. GConv stands for group convolution. a) two stacked convolution layers with the same number of groups. Each output channel only relates to the input channels within the group. No cross talk; b) input and output channels are fully related when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

pointwise group convolutions，其实就是带group的卷积核为1*1的卷积，也就是说pointwise convolution是卷积核为1*1的卷积。在ResNeXt中主要是对3*3的卷积做group操作，但是在ShuffleNet

中，作者是对1*1的卷积做group的操作，因为作者认为1*1的卷积操作的计算量不可忽视。可以看Figure2（b）中的第一个1*1卷积是GConv，表示group convolution。Figure2（a）是ResNet中的bottleneck unit，不过将原来的3*3 Conv改成3*3 DWConv，作者的ShuffleNet主要也是在这基础上做改动。首先用带group的1*1卷积代替原来的1*1卷积，同时跟一个channel shuffle操作，这个前面也介绍过了。然后是3*3 DWConv表示depthwise separable convolution。depthwise separable convolution可以参考MobileNet，下面贴出depthwise separable convolution的示意图。
Figure2（c）添加了一个Average pooling和设置了stride=2，另外原来Resnet最后是一个Add操作，也就是元素值相加，而在（c）中是采用concat的操作，也就是按channel合并，类似googleNet的Inception操作。

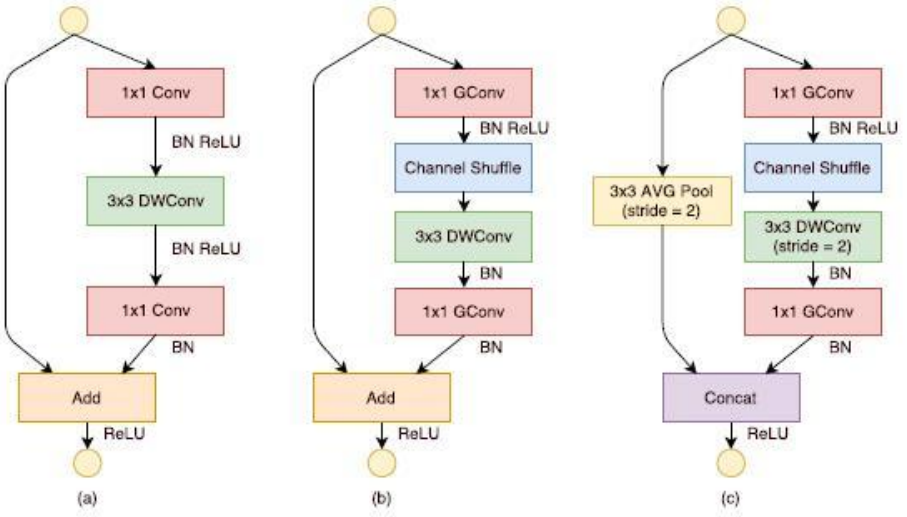
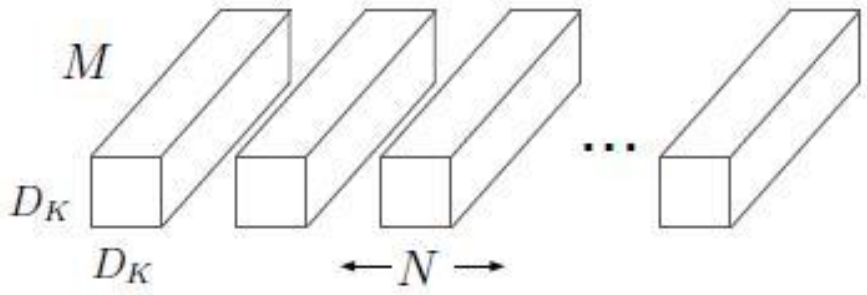
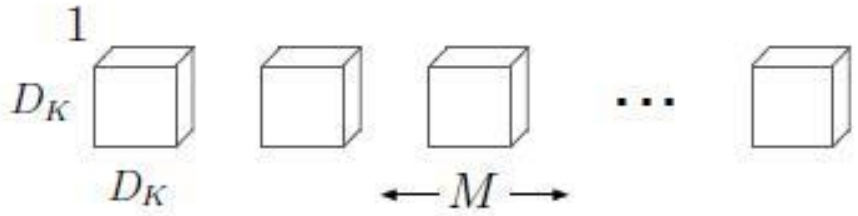


Figure 2: ShuffleNet Units. a) bottleneck unit [9] with depthwise convolution (DWConv) [3, 12]; b) ShuffleNet unit with pointwise group convolution (GConv) and channel shuffle; c) ShuffleNet unit with stride = 2.

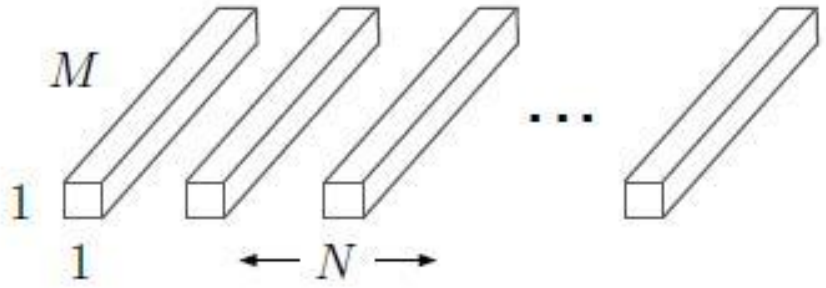
下图就是depthwise separable convolution的示意图，其实就是将传统的卷积操作分成两步，假设原来是3*3的卷积，那么depthwise separable convolution就是先用M个3*3卷积核一对一卷积输入的M个feature map，不求和，生成M个结果，然后用N个1*1的卷积核正常卷积前面生成的M个结果，求和，最后得到N个结果。具体可以看另一篇博文：MobileNets-深度学习模型的加速。



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Table 1是ShuffleNet的结构表，基本上和ResNet是一样的，也是分成几个stage（ResNet中有4个stage，这里只有3个），**然后在每个stage中用ShuffleNet unit代替原来的Residual block，这也就是ShuffleNet算法的核心。**这个表是在限定complexity的情况下，通过改变group（g）的数量来改变output channel的数量，更多的output channel一般而言可以提取更多的特征。

Table 1: ShuffleNet architecture

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[1]	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[2]					143M	140M	137M	133M	137M

实验结果：

Table2表示不同大小的ShuffleNet在不同group数量情况下的分类准确率比较。ShuffleNet s*表示将ShuffleNet 1*的filter个数变成s倍。arch2表示将原来网络结构中的Stage3的两个uint移除，同时在保持复杂度的前提下widen each feature map。Table2的一个重要结论是group个数的线性增长并不会带来分类准确率的线性增长。但是发现ShuffleNet对于小的网络效果更明显，因为一般小的网络的channel个数都不多，在限定计算资源的前提下，ShuffleNet可以使用更多的feature map。

Table 2: Classification error vs. number of groups g (*smaller number represents better performance*)

Model	Complexity	Classification error (%)				
	(MFLOPs)	$g=1$	$g=2$	$g=3$	$g=4$	$g=8$
ShuffleNet $1\times$	140	35.1	34.2	**34.1**	34.3	34.7
ShuffleNet $0.5\times$	38	46.1	45.1	44.4	**43.7**	43.8
ShuffleNet $0.25\times$	13	56.7	56.3	55.6	54.5	**53.7**
ShuffleNet $0.5\times$ (arch2)	40	45.7	44.3	43.8	43.2	**42.7**
ShuffleNet $0.25\times$ (arch2)	13	56.5	55.3	55.5	54.3	**53.3**

Table3表示channel shuffle的重要性。

Table 3: ShuffleNet with/without channel shuffle (*smaller number represents better performance*)

Model	Cls err. (%, no shuffle)	Cls err. (%, shuffle)	Δ err. (%)
ShuffleNet 1x ($g=3$)	36.4	**34.1**	2.3
ShuffleNet 0.5x ($g=3$)	46.1	**44.4**	1.7
ShuffleNet 0.25x ($g=3$)	56.5	**55.6**	0.9
ShuffleNet 0.25x (arch2, $g=3$)	56.6	**55.5**	1.1
ShuffleNet 0.5x (arch2, $g=8$)	46.2	**42.7**	3.5
ShuffleNet 0.25x (arch2, $g=8$)	57.3	**53.3**	4.0

Table4是几个流行的分类网络的分类准确率对比。Table5是ShuffleNet和MobileNet的对比，效果还可以。

Table 4: Classification error vs. various structures (%, *smaller number represents better performance*)

Complexity (MFLOPs)	VGG-like[4]	ResNet	Xception-like	ResNeXt	ShuffleNet (ours)
140	56.0	38.7	35.1	34.3	**34.1** ($1\times$, $g=3$)
38	-	48.9	46.1	46.3	**43.7** ($0.5\times$, $g=4$)
13	-	61.6	56.7	59.2	**53.7** ($0.25\times$, $g=8$)
40 (arch2)	-	48.5	45.7	47.2	**42.7** ($0.5\times$, $g=8$)
13 (arch2)	-	61.3	56.5	61.0	**53.3** ($0.25\times$, $g=8$)

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

Model	Complexity (MFLOPs)	Cls err. (%)	Δ err. (%)
1.0 MobileNet-224	569	29.4	-
ShuffleNet $2\times$ ($g=3$)	524	**29.1**	0.3
0.75 MobileNet-224	325	31.6	-
ShuffleNet $1.5\times$ ($g=3$)	292	**31.0**	0.6
0.5 MobileNet-224	149	36.3	-
ShuffleNet $1\times$ ($g=3$)	140	**34.1**	2.2
0.25 MobileNet-224	41	49.4	-
ShuffleNet $0.5\times$ (arch2, $g=8$)	40	**42.7**	6.7
ShuffleNet $0.5\times$ (shallow, $g=3$)	40	45.2	4.2

总结：

ShuffleNet的核心就是用pointwise group convolution，channel shuffle和depthwise separable convolution代替ResNet block的相应层构成了ShuffleNet uint，达到了减少计算量和提高准确率的目的。channel shuffle解决了多个group convolution叠加出现的边界效应，pointwise group convolution和depthwise separable convolution主要减少了计算量。