ShuffleNet-v1

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

创新点

(1) 受 ResNeXt 的启发，提出使用分组逐点卷积（group pointwise convolution）来代替分组卷积，减少运算复杂度

(2) 分组逐点卷积会导致组之间的信息无法交流，故引入组间信息交换的机制即shuffle层，可以很方便地实现这一机制；并且由于通道重排操作是可导的，因此可以嵌在网络结构中实现端到端的学习。

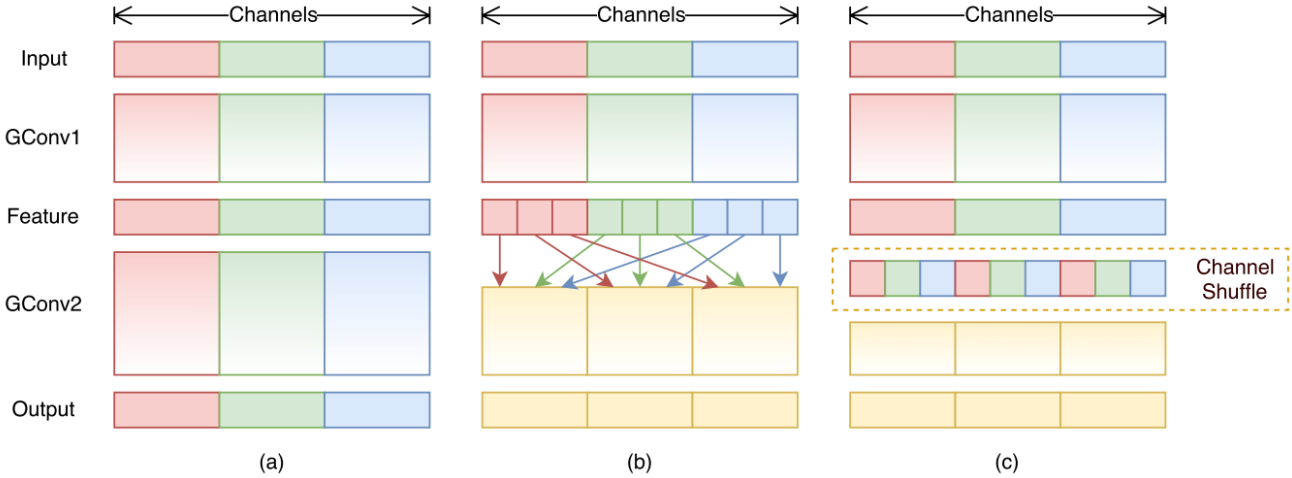
(3) 堆叠以上shuffle结构，构造了一个shuffleNet网络，实验表明，和目前的移动端网络相比，不仅减少了计算复杂度，而且精度有所提升。

核心思想

现代卷积神经网络的绝大多数计算量集中在卷积操作上，因此高效的卷积层设计是减少网络复杂度的关键。其中，稀疏连接（sparse connection）是提高卷积运算效率的有效途径，当前不少优秀的卷积模型均沿用了这一思路。例如Xception引入了"深度可分离卷积"的概念，将普通的卷积运算拆分成逐通道卷积（depthwise convolution）和逐点卷积（pointwise convolution）两步进行，有效地减少了计算量和参数量；Facebook 的ResNeXt则首先使用逐点卷积减少输入特征的通道数(其实就是1x1卷积)，再利用计算量较小的分组卷积（group convolution）结构取代原有的卷积运算，同样可以减少整体的计算复杂度。

ShuffleNet 网络结构同样沿袭了稀疏连接的设计理念。通过分析 Xception 和 ResNeXt 模型，发现这两种结构通过卷积核拆分虽然计算复杂度均较原始卷积运算有所下降，然而拆分所产生的逐点卷积(1x1卷积)计算量却相当可观，成为了新的瓶颈。例如对于 ResNeXt 模型逐点卷积占据了 93.4% 的运算复杂度(在mobilenet-v1论文中也有说明)。可见，为了进一步提升模型的速度，就必须寻求更为高效的结构来取代逐点卷积。

受 ResNeXt 的启发，作者提出使用分组逐点卷积（group pointwise convolution）来代替原来的结构。通过将卷积运算的输入限制在每个组内，模型的计算量取得了显著的下降。然而这样做也带来了明显的问题：在多层逐点卷积堆叠时，模型的信息流被分割在各个组内，组与组之间没有信息交换，这将可能影响到模型的表示能力和识别精度。因此，在使用分组逐点卷积的同时，需要引入组间信息交换的机制。也就是说，对于第二层卷积而言，每个卷积核需要同时接收各组的特征作为输入。作者指出，**通过引入"通道重排"（channel shuffle）可以很方便地实现这一机制；并且由于通道重排操作是可导的，因此可以嵌在网络结构中实现端到端的学习。**

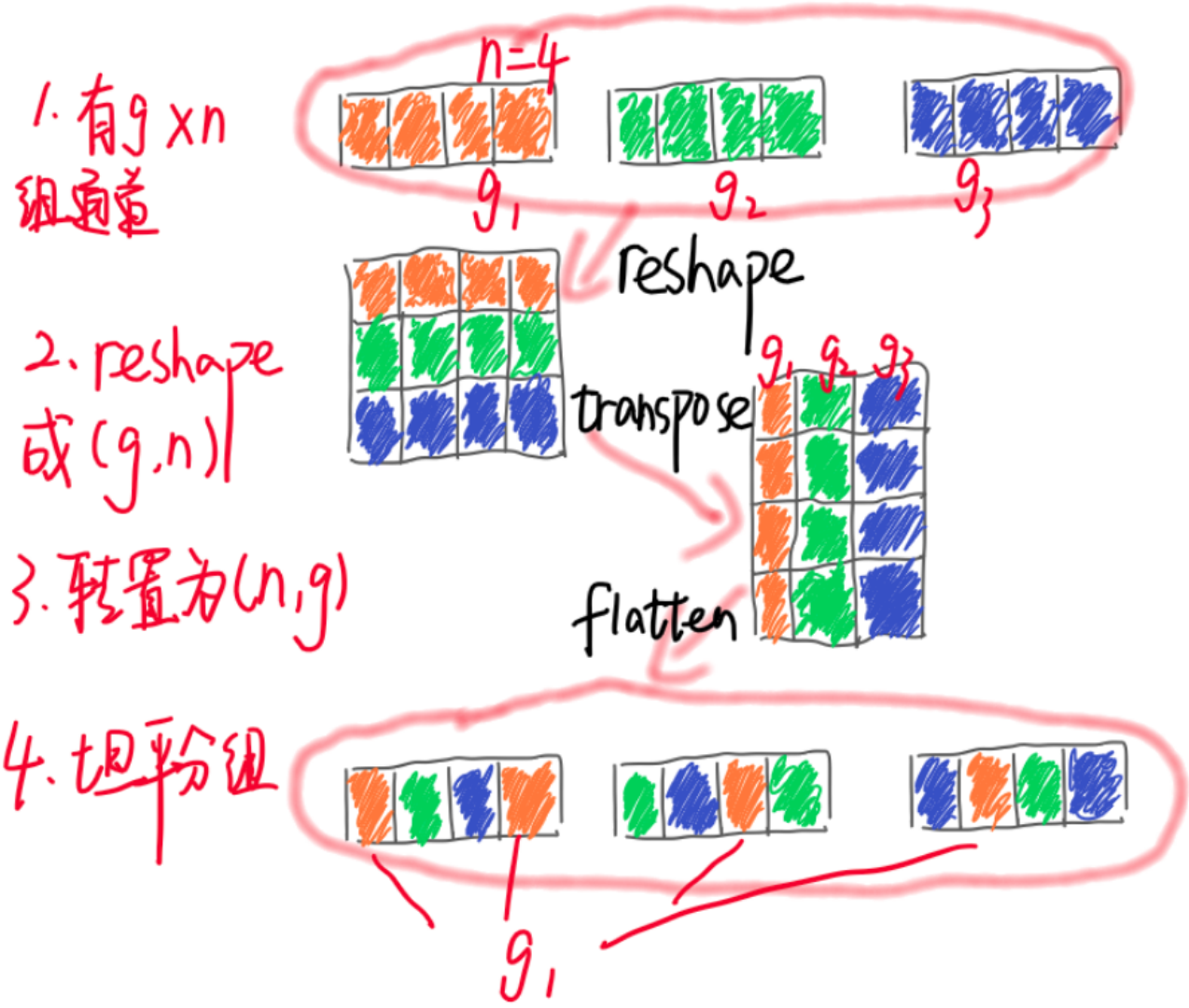


如上图所示，(a) 用了2个分组卷积，组和组之间没有信息流通，这可能影响到模型的表示能力和识别精度；(b) 在第一个组卷积后，将通道打乱实现组和组之间的信息流通；(c)是作者提出的channel shuffle，作用和(b)等价，但是实现上更加高效。

对于这个shuffle操作，有一个有效高雅(efficiently and elegantly)的实现:

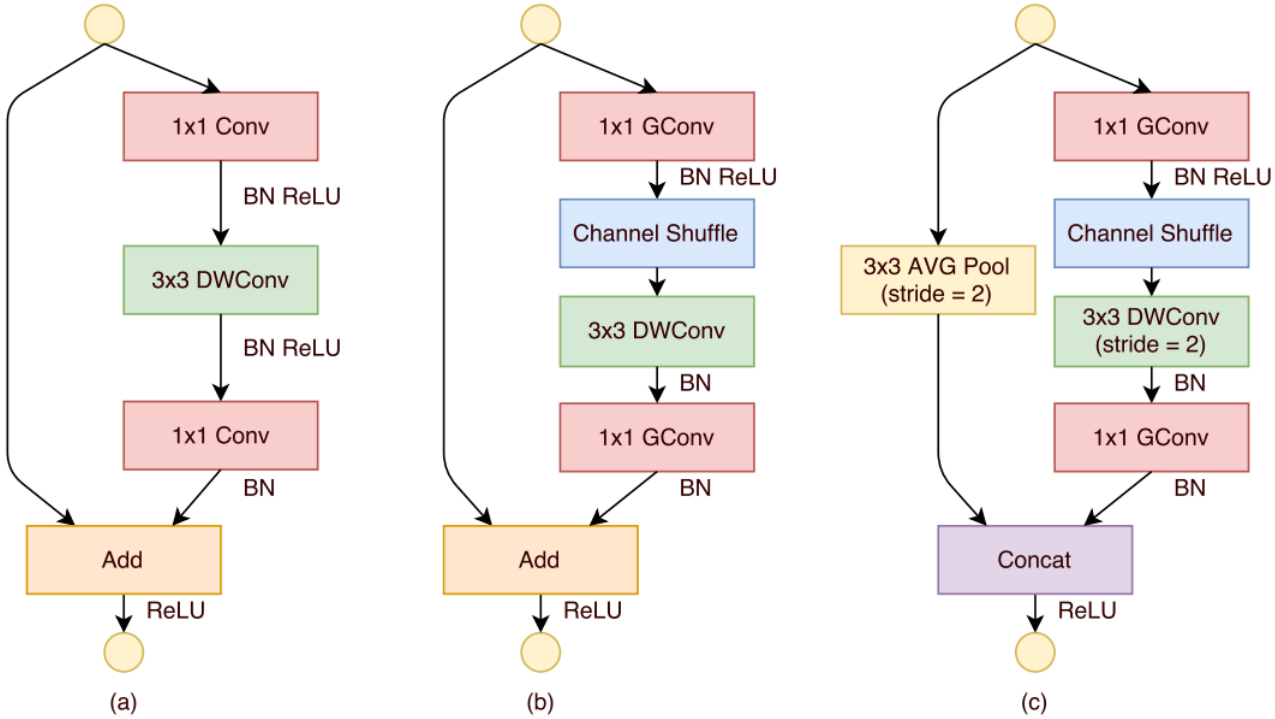
对于一个卷积层分为g组，

- 1.有g×n个输出通道
- 2.reshape为(g,n)
- 3.再转置为(n,g)
- 4.平坦化,再分回g组作为下一层的输入



这样操作有点在于是可微的，模型可以保持end-to-end训练。

在实际过程中我们构建了一个ShuffleNet unit，便于构建实际模型，如下图所示：



(a)是带深度方法卷积的bottleneck单元；(b)是设计的stride=1时候的shuffle单元；(c)是设计的stride=2时候的shuffle单元，在该单元中，使用concat来使得通道加倍，导致信息不缺失。

网络模型

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

上图为shufflenet网络结构，主要包括3个阶段：

- 每个阶段的第一个block的步长为2，下一阶段的通道翻倍
- 每个阶段内的除步长其他超参数保持不变
- 每个ShuffleNet unit的bottleneck通道数为输出的1/4(通道缩减 和ResNet设置一致)

借助ShuffleNet结构单元，作者构建了完整的ShuffeNet网络模型。它主要由16个ShuffleNet结构单元堆叠而成，分属网络的三个阶段，每经过一个阶段特征图的空间尺寸减半，而通道数翻倍。整个模型的总计算量约为140MFLOPs(每秒百万个浮点加乘操作)。通过简单地将各层通道数进行放缩，可以得到其他任意复杂度的模型。

由于**stage2的输入通道比较少，故该阶段的第一个逐点卷积层中不适用分组卷积操作**。其中g表示分组数，参数g控制逐点卷积的连接稀疏性(即分组数)。宁外可以发现，当卷积运算的分组数越多，模型的计算量就越低；这就意味着当总计算量一定时，较大的分组数可以允许设置较多的通道数，作者认为这将有利于网络编码更多的信息，提升模型的识别能力。

定制模型需要满足指定的预算，**我们可以简单的使用放缩因子s控制通道数，ShuffleNets×即表示通道数放缩到s倍**

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**

如上表所示，当计算量一定情况下，增加分组数意味着要增加通道数，可以带来更好的性能。但是并不是绝对的，因为在0.5x的shufflenet中，当g=8时候性能没有g=4的时候好，可能是分组太多，导致每个组卷积的输入特征通道过少。但是在0.25x的shufflenet中，是g=8时候最好的，这说明在特别小的网络中，应该要把分组数增加。

需要注意的是:虽然深度卷积可以减少计算量和参数量，但在低功耗设备上，与密集的操作相比，计算/存储访问的效率更差。故在ShuffleNet上我们只在bottleneck上使用深度卷积，尽可能的减少开销。

实验结果

实验在ImageNet的分类集上做评估，大多数遵循ResNeXt的设置，除了两点：

- 权重衰减从1e-4降低到了4e-5
- 数据增强使用较少的aggressive scale 增强

这样做的原因是小型网络在训练过程通常会遇到欠拟合而不是过拟合问题。

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

上表可以看出，使用了shuffle通道可以有效的减低错误率，且在小网络，分组组增加的情况下提升最明显。

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

通过对比，可以看出shufflenet性能远远好于Mobilenet

在Faster-RCNN框架下，和1.0 MobileNet-224网络复杂度可比的 ShuffleNet 2x，在600分辨率的图上的mAP达到24.5%，而MobileNet为19.8%，表明网络在检测任务上良好的泛化能力 。

其实可以看出shufflenetv1好于mobilenetv1，原因是：

(1) 引入了残差结构

(2) 用了很多分组卷积，导致在同样FLOPS情况下，通道数可以多一些，模型复杂度就会比mobilenet高

代码实现

```
def channel_shuffle(x, groups):
    batchsize, num_channels, height, width = x.data.size()

    channels_per_group = num_channels // groups

    # reshape
    x = x.view(batchsize, groups,
        channels_per_group, height, width)
    x = torch.transpose(x, 1, 2).contiguous()

    # flatten
    x = x.view(batchsize, -1, height, width)

    return x
```

参考文献

[1] https://blog.csdn.net/u011974639/article/details/79200559

[2] https://www.sohu.com/a/156321743_418390