ShuffleNet:2018

An Extremely Efficient Convolutional Neural Network for Mobile Devices

论文出发点或背景

现在许多CNNs模型的发展方向是更大更深，这让深度网络模型难以运行在移动设备上，针对这一问题，许多工作的重点放在对现有预训练模型的修剪、压缩或使用低精度数据表示。

Xception和ResNeXt虽然取得了SOTA，但是因为大量的1×1卷积耗费很多计算资源，导致在小型网络模型中效率较低，论文提出了逐点群卷积(pointwise group convolution)帮助降低计算复杂度，为了解决逐点卷积的副作用，提出了通道混洗来帮助信息流通

论文创新思路

ShuffleNet的工作是推广分组卷积和深度可分离卷积。ShuffleNet的工作专注于设计更好的模型，直接提高性能，而不是加速或转换现有模型。

论文方法介绍

1.针对分组卷积的通道混洗操作

逐点卷积要很大的计算复杂度。在小型网络中，昂贵的逐点卷积造成有限的通道之间充满约束，这会显著的损失精度。为了解决这个问题，一个直接的方法是应用通道稀疏连接，例如组卷积

然而，如果多个组卷积堆叠在一起，会有一个副作用： 某个通道输出仅从一小部分输入通道中导出，如下图所示，这样的属性降低了通道组之间的信息流通，降低了信息

表示能力。

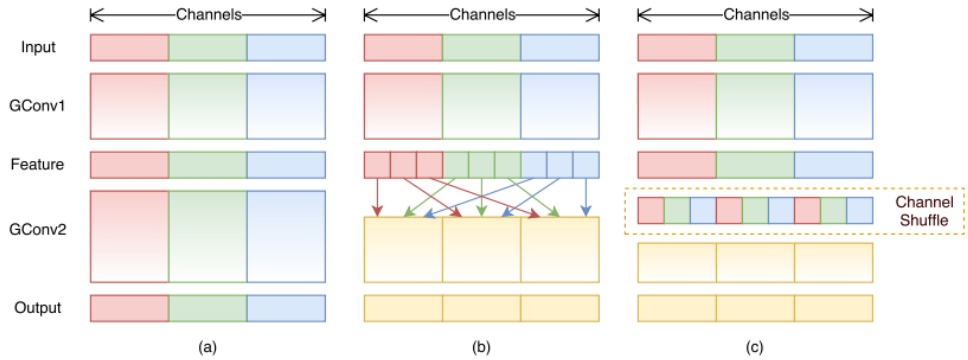

Figure 1. Channel shuffle with two stacked group convolutions. GConv stands for group convolution. a) two stacked convolution layers with the same number of groups. Each output channel only relates to the input channels within the group. No cross talk; b) input and output channels are fully related when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

如果我们允许组卷积能够得到不同组的输入数据，即上图(b)所示效果，那么输入和输出通道会是全关联的。具体来说，对于上一层输出的得到的feature map，我们可先将每个组的通道分为几个子组，然后向下一层的每个组中提供不同的子组。
对于这个混洗操作，有一个有效高雅的实现:

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.flatten，再分回g组作为下一层的输入

即使两个卷积的组数不同，该操作仍然有效。此外，通道混洗也是可微的，模型可以保持end-to-end训练。

2.shufflenet unit

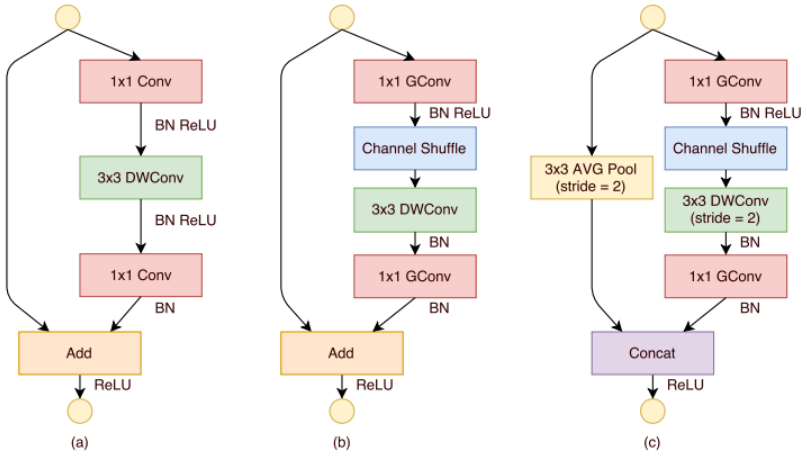

Figure 2. ShuffleNet Units. a) bottleneck unit [9] with depthwise convolution (DWConv) [3, 12]; b) ShuffleNet unit with pointwise group convolution (GConv) and channel shuffle; c) ShuffleNet unit with stride = 2.

我们从图2（a）中的瓶颈单元的设计原理开始，这是一个残差模块。对于主分支部分，我们可将其中标准卷积3×3拆分成深度分离卷积。我们将第一个1×1卷积替换为逐点组卷积，再作通道混洗(即(b))。

图(b)即ShuffleNet unit，主分支最后的1×1Conv改为1×1GConv，第二个逐点群卷积的目的是为了适配和恒等映射做通道融合。配合BN层和ReLU激活函数构成基本单元.

图(c)即是做降采样的ShuffleNet unit，这主要做了两点修改：

- 在辅分支加入步长为2的3×3平均池化
- 原本做元素相加的操作转为了通道级联，这扩大了通道维度，增加的计算成本却很少

给定一个计算限制，ShuffleNet可以使用更宽的特征映射。我们发现这对小型网络很重要，因为小型网络没有足够的通道传递信息。

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.

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

主要分为三个stage：

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

实际效果

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

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

注意组卷积可允许获得更多通道的信息，我们假设性能的收益源于更宽的特征映射，这帮助我们编码更多信息。并且，较小的模型的特征映射通道更少，这意味着能多的从特征映射上获取收益。

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

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

Model	Cls err. (%)	Complexity (MFLOPs)
VGG-16 [31]	28.5	15300
ShuffleNet 2× ($g = 3$)	26.3	**524**
GoogleNet [34]*	31.3	1500
ShuffleNet 1× ($g = 8$)	32.4	**140**
AlexNet [22]	42.8	720
SqueezeNet [14]	42.5	833
ShuffleNet 0.5× ($g = 4$)	41.6	**38**

Table 6. Complexity comparison. **Implemented by BVLC (https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet)*

个人理解

ShuffleNet的创新的点主要有：

- 利用分组卷积降低了普通卷积的计算量
- 利用channel shuffle增加了不同通道间的交互能力

但是这种channel shuffle只是特定通道之间的一种shuffle排列，虽然加强了不同通道之间的交互能力，但可能不是最优解