这篇文章介绍了一种新的网络结构 ShuffleNet 来进行加速运算，这种新结构有两个方面的创新：pointwise group convolution and channel shuffle。

1.pointwise group convolution 就是在 kernel size 为 1*1 的卷积层中也进行 group 操作，这样可以减少计算量，但是引起的问题就是每一组的输出只与改组的输入有关，与别的 group 无关，这样训练的效果并不好。
However, if multiple group convolutions stack together, there is one side effect: outputs from a certain channel are only derived from a small fraction of input channels. Fig 1 (a) illustrates a situation of
two stacked group convolution layers. It is clear that outputs from a certain group only relate to the inputs within the group. This property blocks information flow between channel groups and weakens representation.

因此，需要对每个 group 里面的 channel 进行 shuffle，如(b)所示，将每个 group 分为几个 subgroup，在进行 group 卷积时，输入就是不同 subgroup 的组合，这样做可以有效地利用其他 group 的信息。

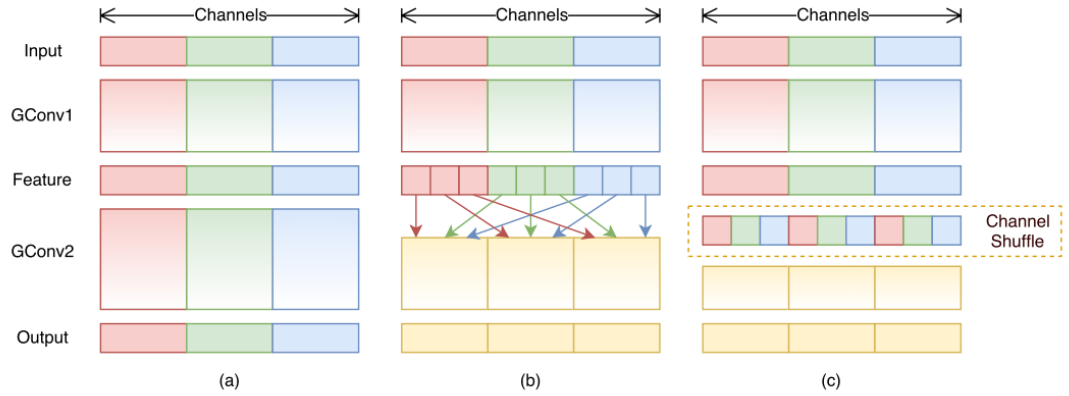

Figure 1: Channel shuffle with two stacked group convolutions. GConv stands for group convolution. a) two stacked convolution layers with the same number of groups. Each output channel only relates to the input channels within the group. No cross talk; b) input and output channels are fully related when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

2.ShuffleNet Units

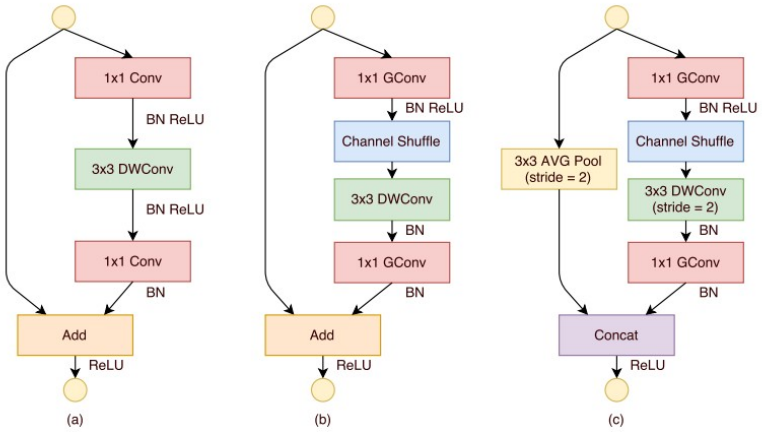

Figure 2: ShuffleNet Units. a) bottleneck unit [9] with depthwise convolution (DWConv) [3, 12]; b) ShuffleNet unit with pointwise group convolution (GConv) and channel shuffle; c) ShuffleNet unit with stride = 2.

网络结构：

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:

(1) 可以看到，group 操作要比不做 group 要好，特别是对于小网络来说，原因可能是在同等的计算量条件下，group 大的对应的 feature map 输入比较多。

Note that group convolution allows more feature map channels for a given complexity constraint, so we hypothesize that the performance gain comes from wider feature maps which help to encode more information.

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

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

为了验证上述猜测，文章用了新的结构 arch2
we remove two units in Stage3 and widen each feature map to maintain the overall complexity. Results of the new architecture (named "arch2")in Table 2. It is clear that the newly designed models are consistently better than the counterparts;

(2) shuffle VS no shuffle

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

(3) 与 MobileNet 的对比

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× ($g=3$)	524	**29.1**	0.3
0.75 MobileNet-224	325	31.6	-
ShuffleNet 1.5× ($g=3$)	292	**31.0**	0.6
0.5 MobileNet-224	149	36.3	-
ShuffleNet 1× ($g=3$)	140	**34.1**	2.2
0.25 MobileNet-224	41	49.4	-
ShuffleNet 0.5× (arch2, $g=8$)	40	**42.7**	6.7
ShuffleNet 0.5× (shallow, $g=3$)	40	45.2	4.2

(4)与其它网络的对比

Table 6: Complexity comparison

Model	Cls err. (%)	Complexity (MFLOPs)
VGG-16 [27]	28.5	15300
ShuffleNet 2× ($g = 3$)	29.1	**524**
PVANET [18] (*our impl.*)	35.3	557
ShuffleNet 1× ($g = 3$)	34.1	**140**
AlexNet [19]	42.8	720
SqueezeNet [13]	42.5	833
ShuffleNet 0.5× (arch2, $g = 8$)	42.7	**40**

(5)在 ARM 上的测试结果

Table 8: Actual inference time on mobile device (*smaller number represents better performance*)

Model	Cls err. (%)	FLOPs	224×224	480×640	720×1280
ShuffleNet 0.5× (arch2, $g = 3$)	43.8	40M	15.2ms	87.4ms	260.1ms
ShuffleNet 1× ($g = 3$)	34.1	140M	37.8ms	222.2ms	684.5ms
AlexNet [19]	42.8	720M	184.0ms	1156.7ms	3633.9ms

与 AlexNet 比较，理论上有 18x 的加速，实际运用中有 13x 的加速。