[논문 리뷰] ShuffleNet [2018]

🌟 해당 논문은 **제한된 Computing Power**를 위해 설계된 **ShuffleNet을 설명**합니다.

Introduction

ShuffleNet은 **pointwise group convolution**과 **channel shuffle 두 개의 operations으로 설계되어, 연산량은 상당히 줄이되**, **Accuracy는 유지**하였습니다.

논문에서는 지정된 **Computing 범위**에서 **효율적인 Architecture를 찾는 것에 목표**를 두었습니다.
Computing 계산을 줄이기 위해 1x1 Convolutions을 사용하기를 제안합니다.

ShuffleNet은 **더 많은 feature map channels를 허용하여 더 많은 정보를 획득**합니다. 또한, **경량화되어 있을 때도, 좋은 성능**을 냅니다.

Related Work

Efficient Model Designs

GoogLeNet, SqueezeNet, ResNet, SENet의 핵심부분과 관련하여 설명합니다.
또한 **NASNet과의 비교**를 함으로써, 언급합니다.

Group Convolution

Group Convolution개념은 **AlexNet에서 처음으로 언급**되었습니다.
ResNeXt는 Group Congolution개념이 매우 효율적이라는 것을 증명해왔으며,
Xception에서 제안된 Depthwise separable를 사용합니다.

MobileNet은 Depthwise separable을 활용하여, 경량화된 모델로 SOTA를 달성하였습니다.

Channel Shuffle Operation

이전의 Channel Shuffle과 관련된 연구를 언급하며, 성과가 없었다는 것을 설명하고 있습니다.

Channel shuffle for Group Convolutions

경량화된 모델에서는 **Expensive Pointwise Convolution를 사용**하게 되면, **제한된 Complexity에서 channel의 개수가 제한**됩니다. 그렇게 되면, **Accuracy에 상당한 피해를 초래**합니다.

이러한 문제를 해결하기 위하여, **Group convolutions을 활용**하여, **1x1 layer로 channel sparse connection을 사용**합니다.

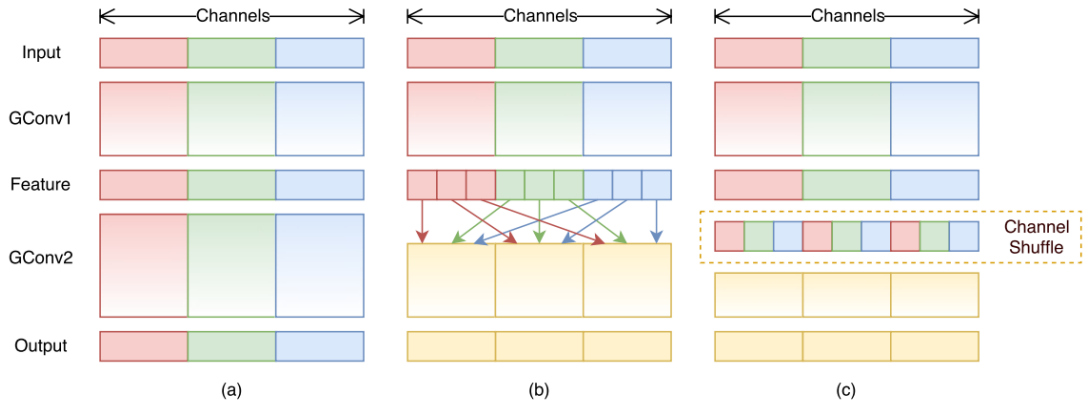

Figure 1. Channel shuffle with two stacked group convolutions. GConv stands for group convolution. a) two stacked convolution layers with the same number of groups. Each output channel only relates to the input channels within the group. No cross talk; b) input and output channels are fully related when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

Channel shuffle은 1x1 Conv layer에서 m개의 채널을 G개의 Group으로 분할하여, 하나의 그룹에 n개의 채널을 포함한다고 가정합니다.

> 💡 **1x1 Conv는 g x n 개의 채널을 출력하고, 이것을 transposing 한 후, flattening하여 다음 layer의 input으로 전달하는 방식으로 이루어집니다.**

ShuffleNet Unit

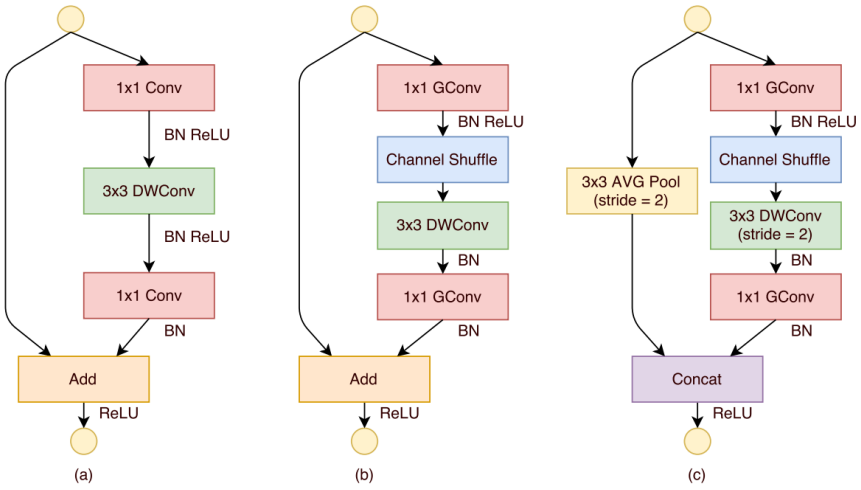

Figure 2. ShuffleNet Units. a) bottleneck unit [9] with depthwise convolution (DWConv) [3, 12]; b) ShuffleNet unit with pointwise group convolution (GConv) and channel shuffle; c) ShuffleNet unit with stride = 2.

(a)는 **Depthwise convolution을 사용하는 BottleNeck 구조**입니다.

(b), (c)는 **ShuffleNet에서 사용하는 unit**입니다.

(c)는 **Down Sampling을 할 때 사용되며, 3x3 DWConv에서 stride = 2를 사용**합니다.
일반적으로 **Feature map의 크기를 줄이는 경우, Channel의 수를 2배로 늘려줍니다.**

이후 **Conv 연산으로 채널 수를 확장하지 않고**, short cut과 concatentation을 하여 채널 수를 확장합니다.
short connection에서도 입력값의 크기를 감소하기 위해, stride = 2 3x3 AVG Pool을 사용합니다.

Pointwise group convolution과 channel shuffle 덕분에, ShuffleNet unit의 구조는 효율적입니다.

Network Architecture

✨**ShuffleNet은 3가지의 ShuffleNet unit이 쌓여 구성되어 있습니다.**

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

Table 1. ShuffleNet architecture. The complexity is evaluated with FLOPs, i.e. the number of floating-point multiplication-adds. Note that for Stage 2, we do not apply group convolution on the first pointwise layer because the number of input channels is relatively small.

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*)

Experiments

Table 4. Classification error vs. various structures (%, *smaller number represents better performance*). We do not report VGG-like structure on smaller networks because the accuracy is significantly worse.

Model	Complexity (MFLOPs)	Cls err. (%)	Δ err. (%)
1.0 MobileNet-224	569	29.4	-
ShuffleNet $2\times$ ($g = 3$)	524	**26.3**	3.1
ShuffleNet $2\times$ (with *SE*[13], $g = 3$)	527	**24.7**	4.7
0.75 MobileNet-224	325	31.6	-
ShuffleNet $1.5\times$ ($g = 3$)	292	**28.5**	3.1
0.5 MobileNet-224	149	36.3	-
ShuffleNet $1\times$ ($g = 8$)	140	**32.4**	3.9
0.25 MobileNet-224	41	49.4	-
ShuffleNet $0.5\times$ ($g = 4$)	38	**41.6**	7.8
ShuffleNet $0.5\times$ (shallow, $g = 3$)	40	42.8	6.6

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

Model	Cls err. (%)	Complexity (MFLOPs)
VGG-16 [30]	28.5	15300
ShuffleNet 2× ($g = 3$)	26.3	**524**
GoogleNet [33]*	31.3	1500
ShuffleNet 1× ($g = 8$)	32.4	**140**
AlexNet [21]	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)*

Model	mAP [.5, .95] (300× image)	mAP [.5, .95] (600× image)
ShuffleNet 2× ($g = 3$)	**18.7%**	**25.0%**
ShuffleNet 1× ($g = 3$)	14.5%	19.8%
1.0 MobileNet-224 [12]	16.4%	19.8%
1.0 MobileNet-224 (our impl.)	14.9%	19.3%

Table 7. Object detection results on MS COCO (*larger numbers represents better performance*). For MobileNets we compare two results: 1) COCO detection scores reported by [12]; 2) finetuning from our reimplemented MobileNets, whose training and finetuning settings are exactly the same as that for ShuffleNets.

Model	Cls err. (%)	FLOPs	224 × 224	480 × 640	720 × 1280
ShuffleNet 0.5× ($g = 3$)	43.2	38M	15.2ms	87.4ms	260.1ms
ShuffleNet 1× ($g = 3$)	32.6	140M	37.8ms	222.2ms	684.5ms
ShuffleNet 2× ($g = 3$)	26.3	524M	108.8ms	617.0ms	1857.6ms
AlexNet [21]	42.8	720M	184.0ms	1156.7ms	3633.9ms
1.0 MobileNet-224 [12]	29.4	569M	110.0ms	612.0ms	1879.2ms

Table 8. Actual inference time on mobile device (*smaller number represents better performance*). The platform is based on a single Qualcomm Snapdragon 820 processor. All results are evaluated with **single thread**.

Reference

ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices

We introduce an extremely computation-efficient CNN architecture named ShuffleNet, which is designed specially for mobile devices with very limited computing power (e.g., 10-150 MFLOPs). The new architecture utilizes two new operations, pointwise group convolution and channel shuffle, to

https://arxiv.org/abs/1707.01083

[논문 읽기] ShuffleNet(2018) 리뷰, An Extremely Efficient Convolutional Neural Network for Mobile Devices

안녕하세요! 이번에 읽어볼 논문은 ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices 입니다. ShuffleNet은 경량화에 집중한 모델입니다. 또 다른 경량화 모델인 MobileNetV1을 능가합니다. 그리고 AlexNet과 비교하여 동일한 정확도로 13배나 더 빠릅니다. MobileNetV1와 Xception에서 연산량을 줄이기 위해 사용하는 Depthwise separable

https://deep-learning-study.tistory.com/544