ShuffleNet

16기 최 리

2021.04.08

CONTENTS

01

Introduction

Introduction

- pointwise group convolution & channel shuffle
- 정확도 유지, 계산 비용 감소
- classification & object detection 좋은 성능
- MobileNet보다 낮은 오류, AlexNet보다 13배 빠른 속도

Introduction

- 제한된 계산에서 최고의 정확도 추구
- 모바일 플랫폼에 중점
- 원하는 계산 범위에서 효율적인 standard architecture를 찾는 것이 목표!

효율을 높이기 위해 pointwise group convolution을 사용,
부작용을 극복하기 위해 channel shuffle

- ShuffleNet은 많은 feature map channel을 사용하기 때문에 더 많은 정보를 알 수 있고 작은 네트워크에서도 좋은 성능!

02

Related Work

01. Efficient Model Designs
02. Group Convolution
03. Channel Shuffle Operation
04. Model Acceleration

Related Works

- Efficient Model Designs
- Group Convolution
- Channel Shuffle Operation
- Model Acceleration

03

Approach

01. Channel Shuffle for Group Convolutions
02. ShuffleNet Unit
03. Network Architecture

Channel Shuffle

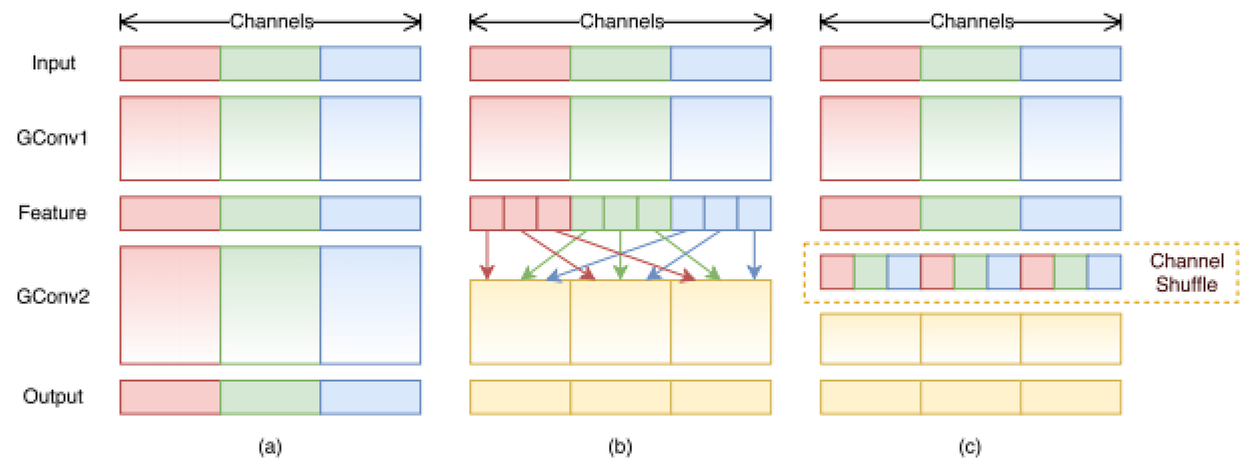

Figure 1. Channel shuffle with two stacked group convolutions. GConv stands for group convolution. a) two stacked convolution layers with the same number of groups. Each output channel only relates to the input channels within the group. No cross talk; b) input and output channels are fully related when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

Channel shuffle을 이용해 multiple group conv layer로 더 강력한 구조 구축

(a) 2개의 stack conv layer, input을 채널별 그룹으로 나눠서 conv 수행
-> 특정 그룹의 output은 해당 그룹 내의 input에만 관련

(b), (c) 각 그룹의 feature map을 서브그룹으로 나눠 channel shuffle 수행.
-> 모든 그룹이 서로 관계가 있음. Input과 채널이 잘 연결된다.

ShuffleNet Unit

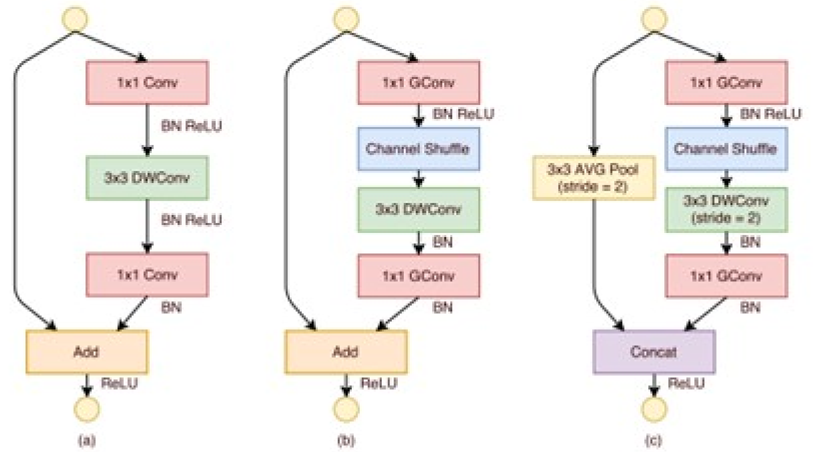

Figure 2. ShuffleNet Units. a) bottleneck unit [9] with depthwise convolution (DWConv) [3, 12]; b) ShuffleNet unit with pointwise group convolution (GConv) and channel shuffle; c) ShuffleNet unit with stride = 2.

Channel shuffle, pointwise group conv로 연산량을 줄여 효율적으로 계산하며 더 많은 feature를 사용할 수 있게 된다.

(a) Residual block, 3x3 layer의 경우 3x3 conv 적용

(b) ShuffleNet unit 구성을 위해 첫번째 1x1 conv에 pointwise group conv 적용, channel shuffle (2번째 pointwise group conv부터는 channel shuffle X)

(c) ShuffleNet에 스트라이드 적용하는 경우

03 Approach

01. Channel Shuffle

02 ShuffleNet Unit

03 Network Architecture

Network 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	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.

04

Experiments

01. Ablation Study
02. Comparison
03. Generalization Ability
04. Actual Speedup Evaluation

04 Experiments

01. Ablation Study

02 Comparison

03. Generalization Ability

04. Actual Speedup Evaluation

Ablation Study

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**

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

[채널 수를 scaling 한 ShuffleNet 비교]

그룹 수가 많아질수록 성능이 좋아지고 더 작은 모델일수록 그룹 개수별 성능의 차이가 커진다.

-> 작은 모델일수록 채널 수가 중요하다!

Ablation Study

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*)

[Channel Shuffle 했을 때 vs. 안 했을 때]

Channel shuffle을 하는 경우가 안 하는 경우보다 항상 성능이 높음

Channel shuffle을 한 경우끼리 비교해 봤을 때는 그룹 수가 많은 경우에 성능이 더 높음

04 Experiments

Comparison

Complexity (MFLOPs)	VGG-like	ResNet	Xception-like	ResNeXt	ShuffleNet (ours)
140	50.7	37.3	33.6	33.3	**32.4** ($1\times$, $g = 8$)
38	-	48.8	45.1	46.0	**41.6** ($0.5\times$, $g = 4$)
13	-	63.7	57.1	65.2	**52.7** ($0.25\times$, $g = 8$)

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 $2\times$ ($g = 3$)	26.3	**524**
GoogleNet [33]*	31.3	1500
ShuffleNet $1\times$ ($g = 8$)	32.4	**140**
AlexNet [21]	42.8	720
SqueezeNet [14]	42.5	833
ShuffleNet $0.5\times$ ($g = 4$)	41.6	**38**

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

Genralization Ability

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.

[MS COCO object detection]

MobileNet과 비교했을 때 ShuffleNet이 더 좋은 성능을 보인다.

04 Experiments

01. Ablation Study

02 Comparison

03. Generalization Ability

04. Actual Speedup Evaluation

Actual Speedup Evaluation

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**.

[ARM 플랫폼 사용하는 모바일 장치]

이론상으로 그룹이 많을수록 성능이 더 좋아야 하지만 실제로는 g=3이 적당

Complexity가 4배 줄어들 때마다 속도가 2.6배 향상 되어야 맞지만 .. 아님!
AlexNet보다 18배 빠른 것을 기대했지만 실제로는 13배정도 빠르다.

암튼 빠름!!!

감사합니다

16기 최 리