ShuffleNet

김초은

Introduction

- MobileNet의 구조를 기본적으로 사용
- 모델 경량화를 목표로 함
- 더 많은 feature map channel을 사용하여 더 많은 정보를 인코딩할 수 있도록 함
- → 매우 작은 네트워크에서의 성능에 큰 영향을 미침

주요 아키텍처

- Depthwise separable convolution
- Grouped Convolution
- Channel Shuffle

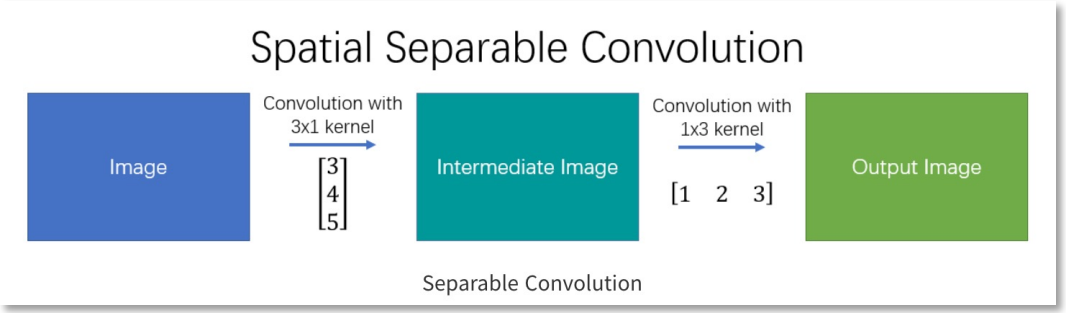
Architecture - Depthwise Separable Convolution

Separable Convolution

kernel을 column vector와 row vector의 곱으로 바꾸어 사용

2D conv 1회 → 1D conv 2회 : computational cost가 줄어듦

모든 kernel이 separable하지는 않음 (rank가 1인 2D conv kernel만 separable)

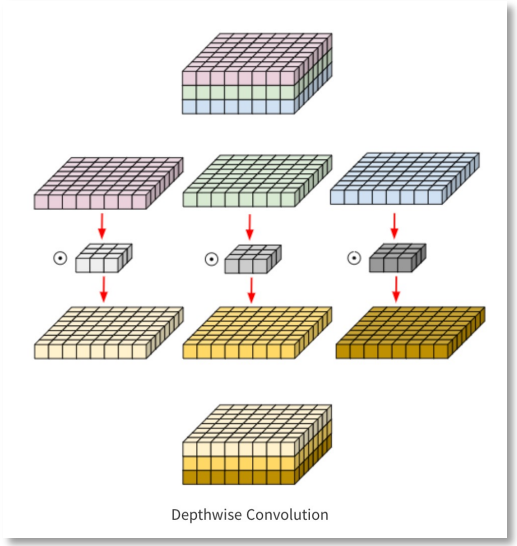


Depthwise Convolution

각 단일 채널에 대해서만 수행되는 필터들을 사용

채널 방향의 conv는 진행하지 않고, 공간 방향의 conv만 진행

→ 완전히 특정 채널만의 공간적 특성을 추출할 수 있음



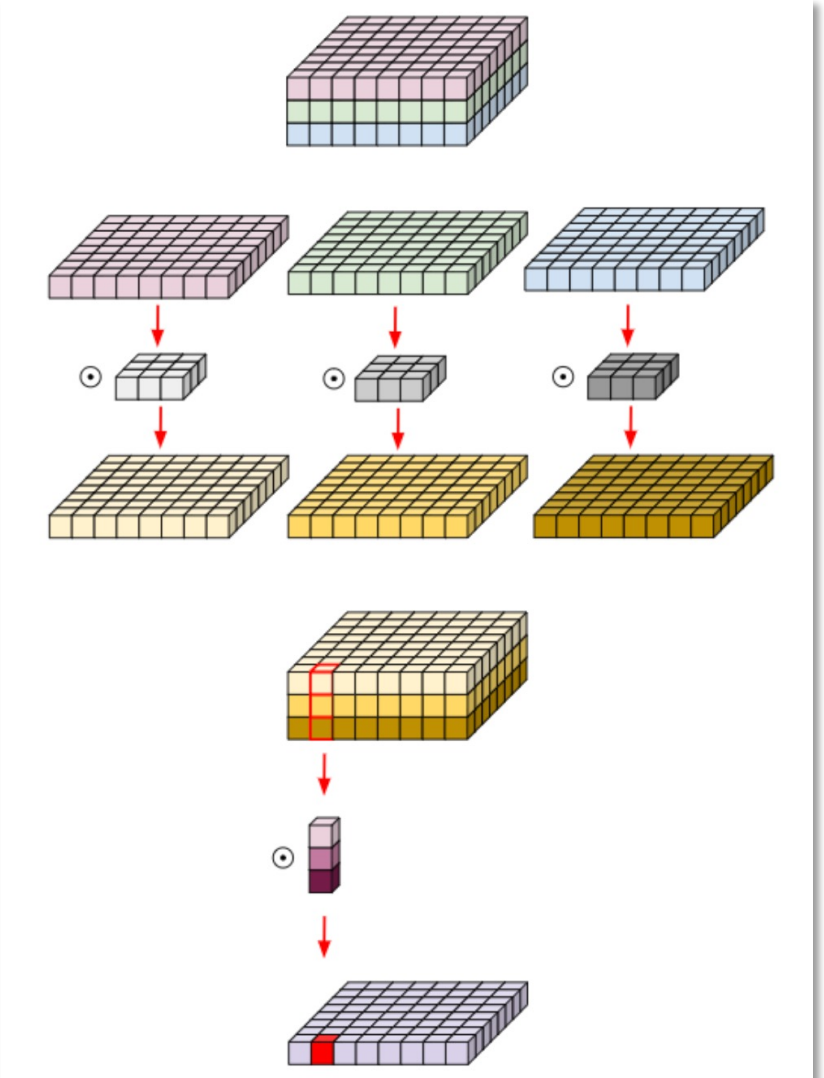
Architecture - Depthwise Separable Convolution

Depthwise Separable Convolution

Depthwise conv의 output에 추가적인 convolution을 진행하여 output 채널의 수를 1로 줄임. (h, w는 그대로)

- 공간적 conv 이후에 채널 conv를 함. 즉 원래 conv를 이 두가지로 separate한 것. → separable
- Depthwise conv를 진행 → depthwise

→ 기존의 convolution 연산과 거의 유사하게 동작하지만 파라미터의 수가 훨씬 작음

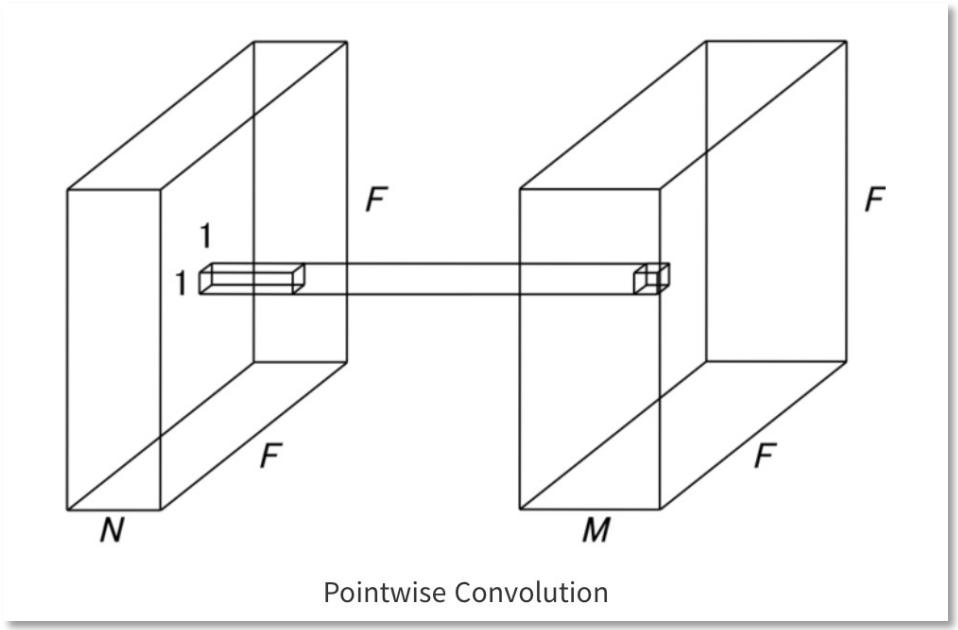

Depthwise Separable Convolution

Architecture - Grouped Convolution

Pointwise Convolution

채널 방향의 conv만 진행. Channel Reduction에 사용.

- 다채널 입력 영상을 더 작은 채널의 영상으로의 embedding하는 것
- 출력 채널 수가 줄어듦에 따라 파라미터의 양도 줄어듦. 불필요한 채널들에 작은 계수를 적용하여 희석시킬 수도 있음
- tradeoff : 속도 vs. 정보손실
- Inception, Xception, ResNeXt 등이 사용



Pointwise Convolution

→ 그러나 pointwise convolution은 작은 모델에서는 너무 복잡한 방법.
ResNeXt에서도 pointwise conv가 연산량의 93.4%를 차지

Architecture - Grouped Convolution

Grouped Convolution

입력 값의 채널들을 여러 개의 그룹으로 나누어 독립적으로 conv 연산을 수행.

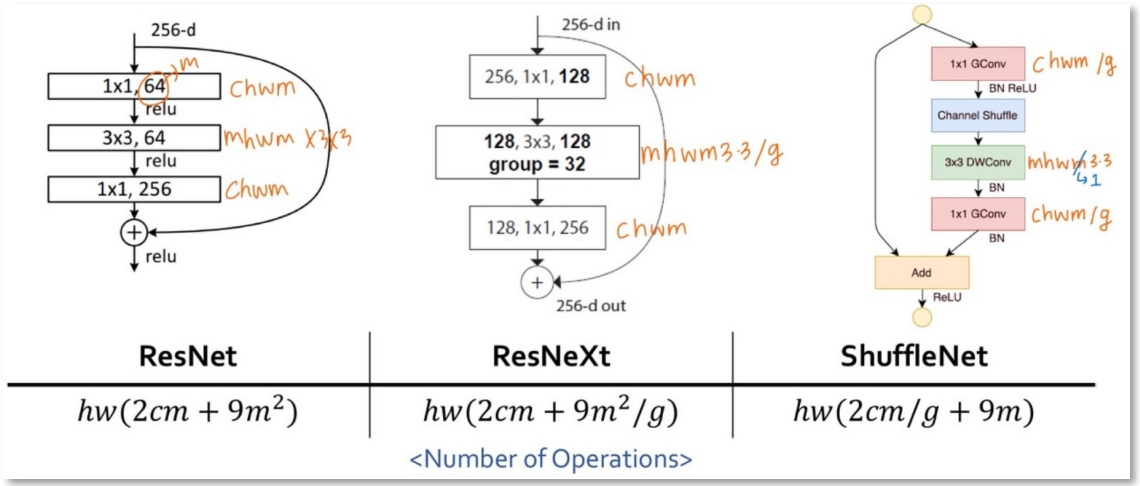
- 병렬 처리에 유리
- 낮은 파라미터 수와 연산량
- 각 그룹에 높은 Correlation을 가지는 채널이 학습될 수 있음
- 심지어 그룹의 수가 늘어나면 파라미터는 줄면서도 성능 향상이 일어나는 경우가 있음

연산량 계산

K^2CHWM → K^2CHWM/g

K : 커널의 크기, C : input 채널 수, H, W : 이미지 크기,

M : output 채널 수, g: 그룹 수

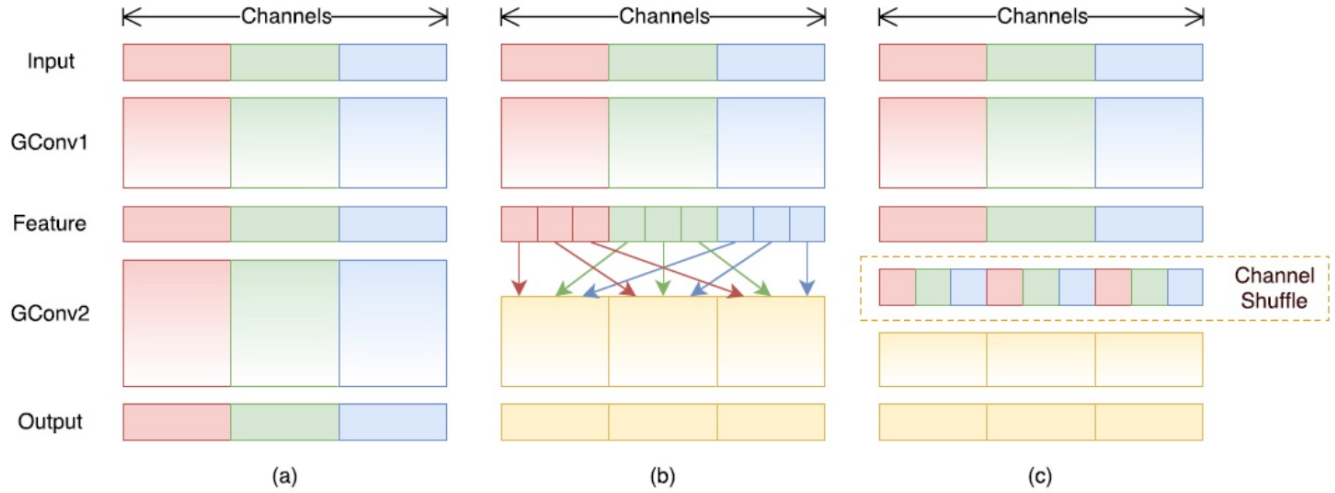


ResNet	ResNeXt	ShuffleNet
$hw(2cm + 9m^2)$	$hw(2cm + 9m^2/g)$	$hw(2cm/g + 9m)$

<Number of Operations>

Architecture - Channel Shuffle

Channel Shuffle



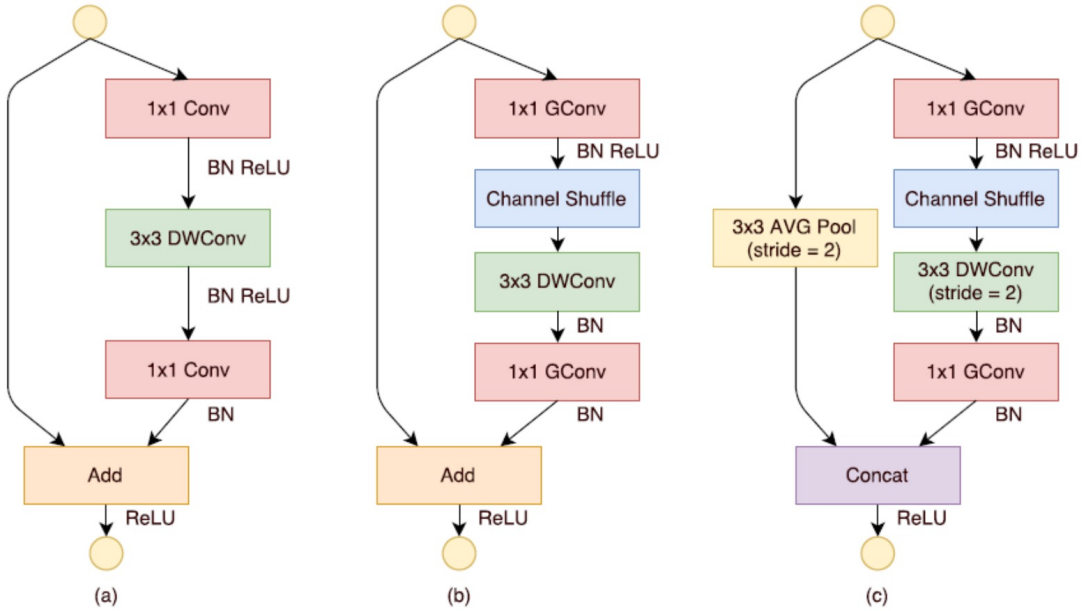
(a) : 채널을 그룹으로 나누고 그 그룹에 convolution을 진행. 이 경우에는 나누어진 그룹에만 계속 conv를 진행

→ 각 그룹에 해당하는 것만 학습하게 됨

(b), (c) : 각 채널을 g개로 나누고, 나누어진 소단위들이 각 그룹에 한 번씩 포함될 수 있도록 섞음

Architecture - ShuffleNet Units

MobileNet의 구조에 **Group Convolution**과 **Channel Shuffle**을 적용한 구조



(a) : MobileNet에 residual connection을 추가한 형태

(b) : Group Convolution과 Channel Shuffle을 추가. 첫 레이어 뒤에만 ReLU를 적용

(c) : (b)에 stride 2를 줘서 가로 세로 사이즈를 반으로 줄이고, residual에서의 output과 channel-wise concatenate

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

group의 개수가 많아지면 연산량이 줄어드는데 그만큼 channel의 수를 크게 하여 그룹의 개수에 따라 모델의 complexity 가 다르지 않도록 함.

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**

$s\times$ 는 필터의 개수가 s 배 되었다는 의미

→ complexity는 약 s^2 배가 됨

group의 수가 많아질수록 error가 낮아지는 경향을 보임

Performance

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

비슷한 complexity지만 error는 MobileNet보다 낮음