📖

[논문리뷰] MobileNetV2 [2018]

Introduction

✨ 해당 논문은 New Mobile Architecture인 MobileNetV2를 소개합니다.

MobileNetV2는 Inverted Residual Structure을 base로 만들어졌습니다. Intermediate expansion layer는 경량화된 Depthwise Convolutions를 사용하여 비선형적으로 features를 filter합니다.

강력한 표현력(representation power)를 유지하기 위해서는 **narrow layer에서는 비선형성을 제거하는 것이 중요하다는 것을 발견**하였습니다.

> 💡 **Our main contribution is novel layer module: the inverted residual with linear bottleneck**

Inverted residual with linear bottleneck

- **고차원으로 확장되었다, 경량화된 depwthwise convolution의 filter을 통하여 낮은 차원으로 압축한 표현방법을 사용**합니다.

어떻게 **Neural Network이 작동**하는지, **단순한 Network design이 가능한 지에 관한 직관을 제공하려고 합니다.**

Preliminaries, dicussion and intuition

Linear Bottlenecks

- 실제 **Image Input에 대해서, 각각의 Pixel 및 Channel에 대해 $h_i * w_i * d_i$ 크기의 activation이 존재**합니다. 이러한, **layer activation은 “manifold of interest”를 형성**합니다.
- **신경망에서의 manifold of interest는 저차원의 subspace로 embedded**로 될 수 있습니다.

MobileNetV1에서는 Low dimensional에서 Subspace로 보냈다가 다시 복구하는 BottleNeck layer를 적용하여, activation space의 차원을 효과적으로 줄일 수 있었습니다.

또한, **ReLU와 같은 것이 있기 때문에 일부 소실**될 수 있습니다.

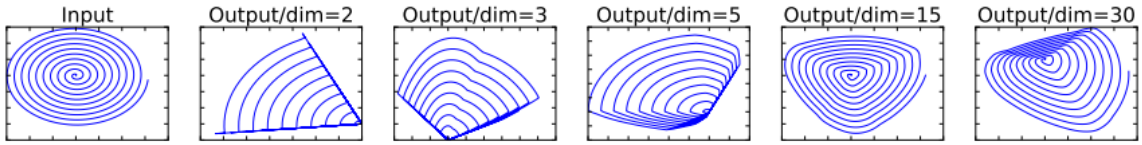

Figure 1: Examples of ReLU transformations of low-dimensional manifolds embedded in higher-dimensional spaces. In these examples the initial spiral is embedded into an n-dimensional space using random matrix T followed by ReLU, and then projected back to the 2D space using T^{-1}. In examples above $n = 2, 3$ result in information loss where certain points of the manifold collapse into each other, while for $n = 15$ to 30 the transformation is highly non-convex.

✨ **정리하자면,**

- **Manifold of interest가 ReLU 변환 이후에도 non-zero volume으로 남아 있다면, 그것은 선형 변환에 부합**할 것이다.
- **ReLU가 입력 Manifold에 대해 정보를 완전히 보전하는 경우**는, 입력 manifold가 입력 space의 저차원 subspace에 있을 때에만 그렇습니다.
- 그러므로, ReLU와 같은 비선형 Activation이 없는 layer를 하나 더 추가하자는 것이다.

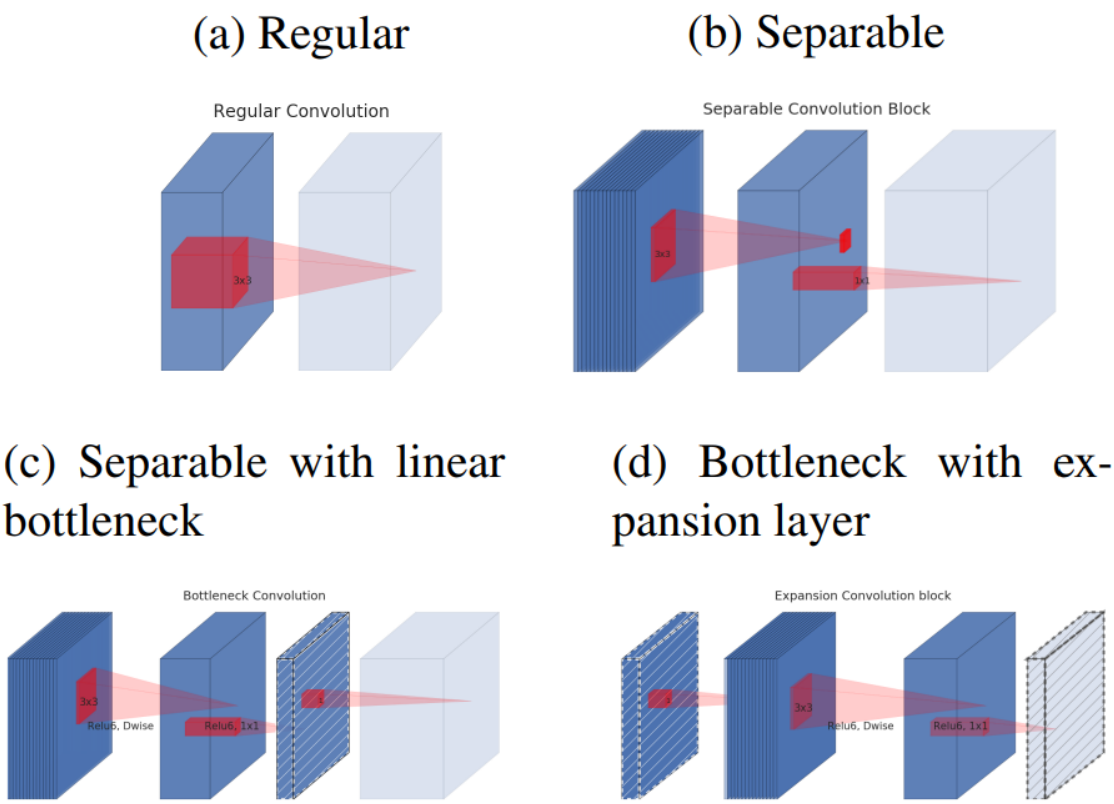

Figure 2: Evolution of separable convolution blocks. The diagonally hatched texture indicates layers that do not contain non-linearities. The last (lightly colored) layer indicates the beginning of the next block. Note: 2d and 2c are equivalent blocks when stacked. Best viewed in color.

Inverted Residual

BottleNeck Block은 Residual Block과 상당히 유사합니다.

하지만, 일반적인 Residual Block과 다릅니다.

보통은 **wide-narrow-wide의 layer가 있고, wide layer끼리 연결을 추가하는 방식**이지만, 해당 논문에서는

- narrow - wide - narrow layer가 기본입니다.
- narrow layer끼리 연결되어 있습니다.

이러한 이유는 **narrow에 존재하는 저차원의 layer에 있는 정보만 압축된 채로 저장**되어 있다고 가정하기 때문입니다. **즉 필요한 정보는 narrow에 이미 존재하기에, skip connection으로 사용해도 정보를 더 깊이 layer까지 전달**할 수 있을 것이라고 생각하기 때문입니다.

(a) Residual block (b) Inverted residual block

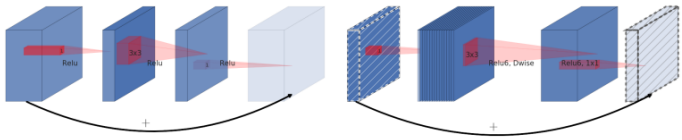

Figure 3: The difference between residual block [8, 30] and inverted residual. Diagonally hatched layers do not use non-linearities. We use thickness of each block to indicate its relative number of channels. Note how classical residuals connects the layers with high number of channels, whereas the inverted residuals connect the bottlenecks. Best viewed in color.

Information flow interpretation

우리의 구조의 흥미로운 특성은 **input/output domains 영역에서 자연스러운 분리**가 이루어진다는 것이다. 그리고 **input에서 ouput으로 전환될때, layer가 비선형적**이게 됩니다.

Model Architecture

MobileNetV2의 구조는 **32개의 filter를 갖는 fully conv layer를 가지고 있으며, 19개의 Residual Block을 가지고 있습니다.**

낮은 정밀도 계산을 할 때, 견고함을 위해 비선형함수로는 ReLU6를 이용합니다.
연구에서는 **확장 비율(Expansion factor)을 6**으로 고정하였습니다.

Input	Operator	Output
$h \times w \times k$	1x1 conv2d , ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

Table 1: *Bottleneck residual block* transforming from k to k' channels, with stride s, and expansion factor t.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Size	MobileNetV1	MobileNetV2	ShuffleNet (2x,g=3)
112x112	64/1600	16/400	32/800
56x56	128/800	32/200	48/300
28x28	256/400	64/100	400/600K
14x14	512/200	160/62	800/310
7x7	1024/199	320/32	1600/156
1x1	1024/2	1280/2	1600/3
max	1600K	**400K**	600K

Experiment

기존의 tailored for model and 제약된 환경에서의 모델의 성능이 뛰어나다는 것을 알 수 있습니다.

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

Network	mAP	Params	MAdd	CPU
SSD300[34]	23.2	36.1M	35.2B	-
SSD512[34]	26.8	36.1M	99.5B	-
YOLOv2[35]	21.6	50.7M	17.5B	-
MNet V1 + SSDLite	22.2	5.1M	1.3B	270ms
MNet V2 + SSDLite	22.1	**4.3M**	**0.8B**	200ms

Conclustion and futurework

매우 친숙하고, 효율적인 방식으로 단순한 Model Architecture를 describe하였습니다. 메모리에서 효율적인 방법과 계산량에서 이점을 가지는 MobileNetV2를 제안합니다.

MobileNetV2: Inverted Residuals and Linear Bottlenecks

In this paper we describe a new mobile architecture, MobileNetV2, that improves the state of the art performance of mobile models on multiple tasks and benchmarks as well as

https://arxiv.org/abs/1801.04381

Python, Machine & Deep Learning

이 글에서는 Google Inc.에서 발표한 MobileNet V2 논문을 간략하게 정리한다. 논문 링크: MobileNets: Inverted Residuals and Linear Bottlenecks Github:

https://greeksharifa.github.io/computer%20vision/2022/02/10/MobileNetV2/