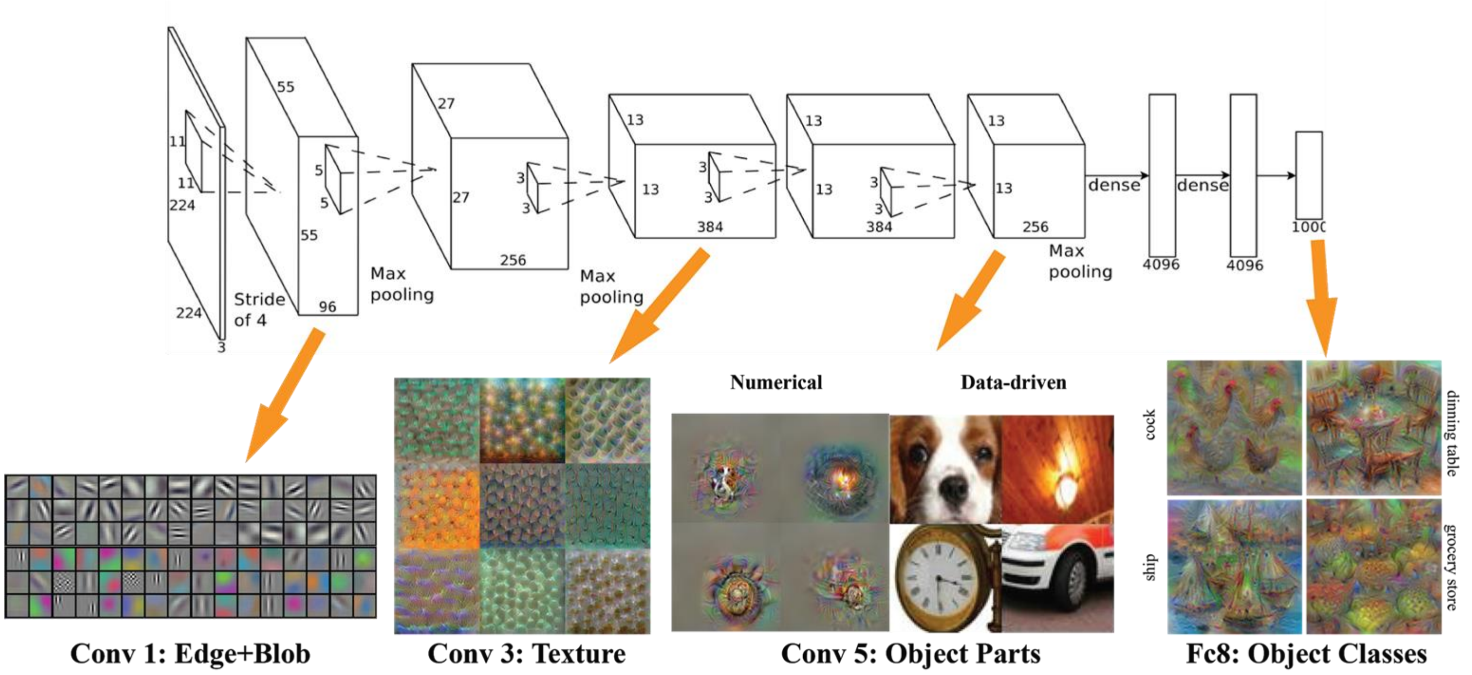
AlexNet
VGG16
MobileNet

Week 2

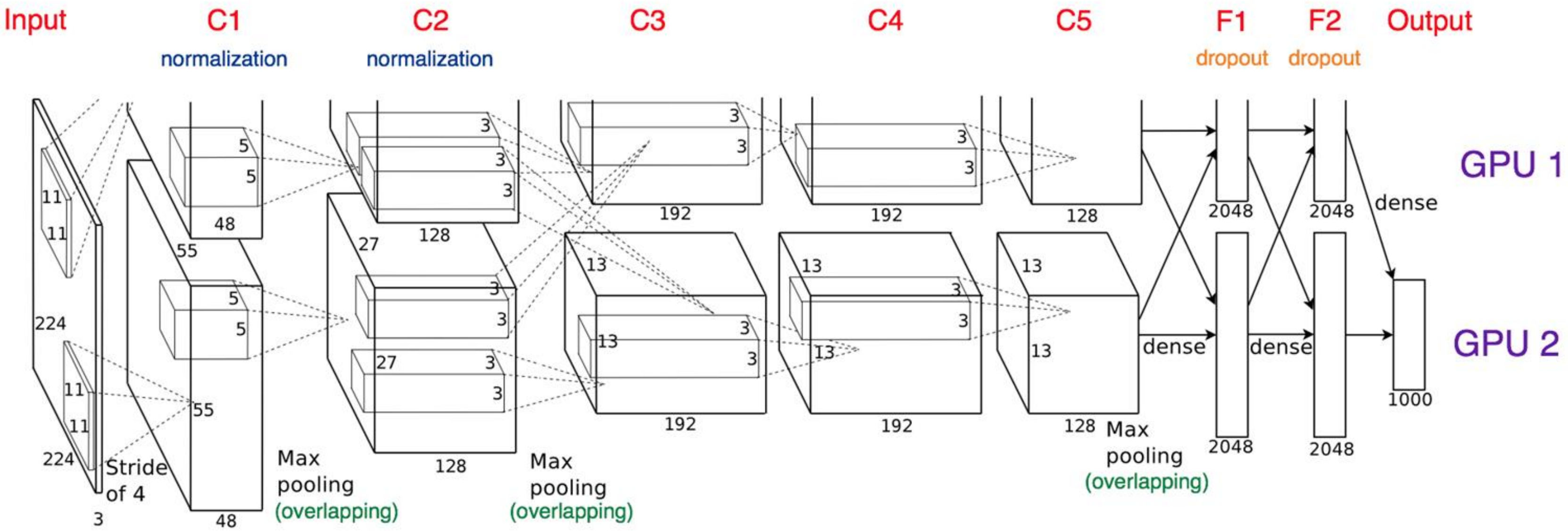
Presented by Group 3

Table of Contents


11
11
224
224
3
55
55
5
5
Stride of 4
96
Max pooling
27
27
3
3
256
Max pooling
13
13
3
3
384
13
13
3
3
384
13
13
256
Max pooling
dense
dense
4096
4096
1000
Numerical
Data-driven
cock
dinning table
ship
grocery store
Conv 1: Edge+Blob
Conv 3: Texture
Conv 5: Object Parts
Fc8: Object Classes

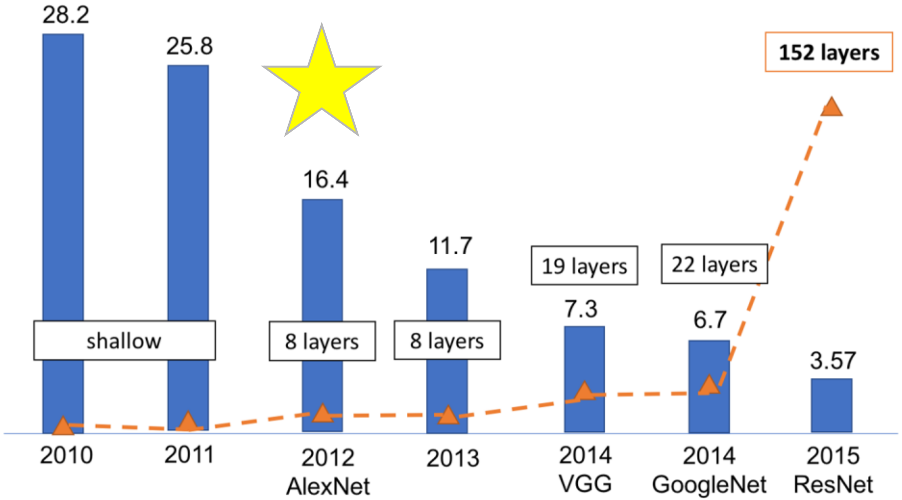
AlexNet

AlexNet (2012)



AlexNet (2012)

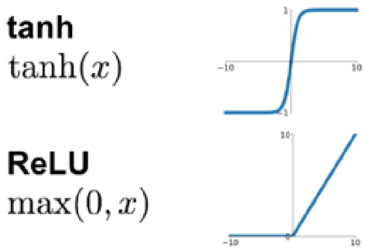
1. Training on Multiple GPUs
2. ReLU Activation Function
3. LRN - Local Response Normalization
4. Overlapping Pooling
5. Dropout & Data Augmentation

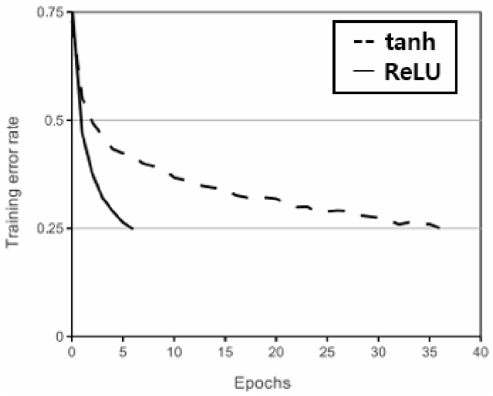


< Results of ILSVRC >

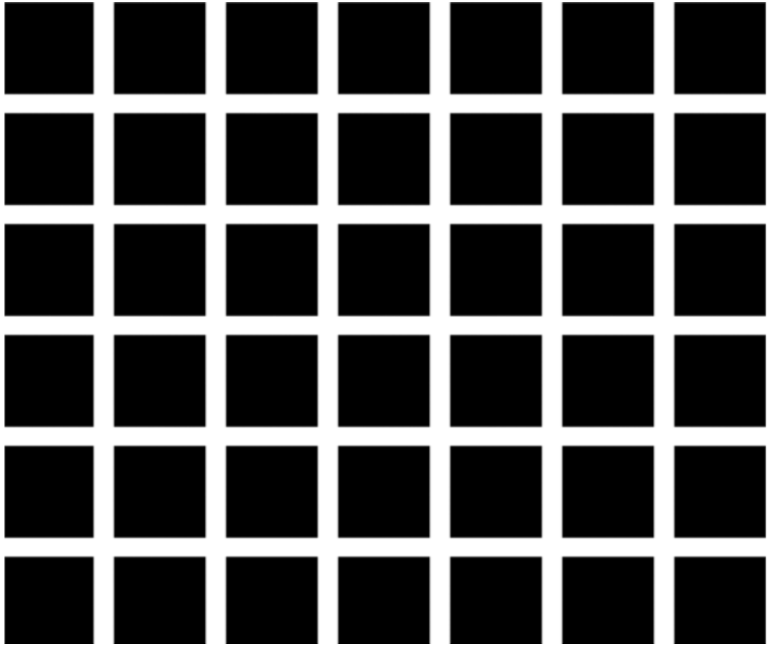
ReLU Activation Function

- A 4 layer CNN with ReLUs (solid line) converges six times faster than an equivalent network with tanh neurons (dashed line) on CIFAR-10 dataset.
- Derivative of Relu is 1 or 0, so it can only vanish gradients if they <0, which can accelerate learning and solve the vanishing gradient problem.
- Dead Neurons : If the units are not activated initially, then they are always in the off-state as zero gradients flow through them. This can be solved by enforcing a small negative gradient flow through the network (Leaky ReLU).





LRN - Local Response Normalization



$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=max(0,i-n/2)}^{j=min(N-1,i+n/2)} {a^j_{x,y}}^2\right)^\beta$$

where

$b^i_{x,y}$ – regularized output for kernel i at position x, y

$a^i_{x,y}$ – source ouput of kernel i applied at position x, y

N – total number of kernels

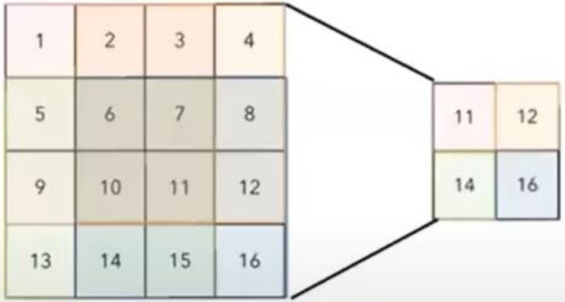
n – size of the normalization neigbourhood

$\alpha, \beta, k, (n)$ – hyperparameters

Overlapping Pooling

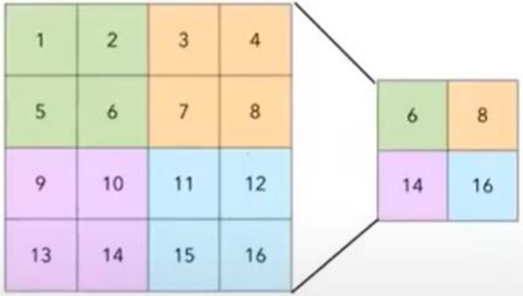
- **Overlapping pooling**

Pooling window 의 크기 > stride의 크기

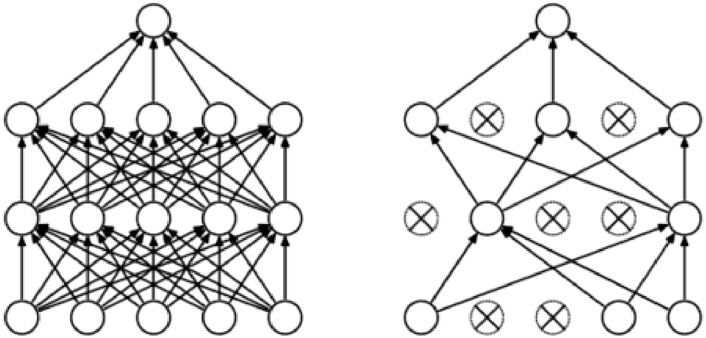


- **Traditional pooling**

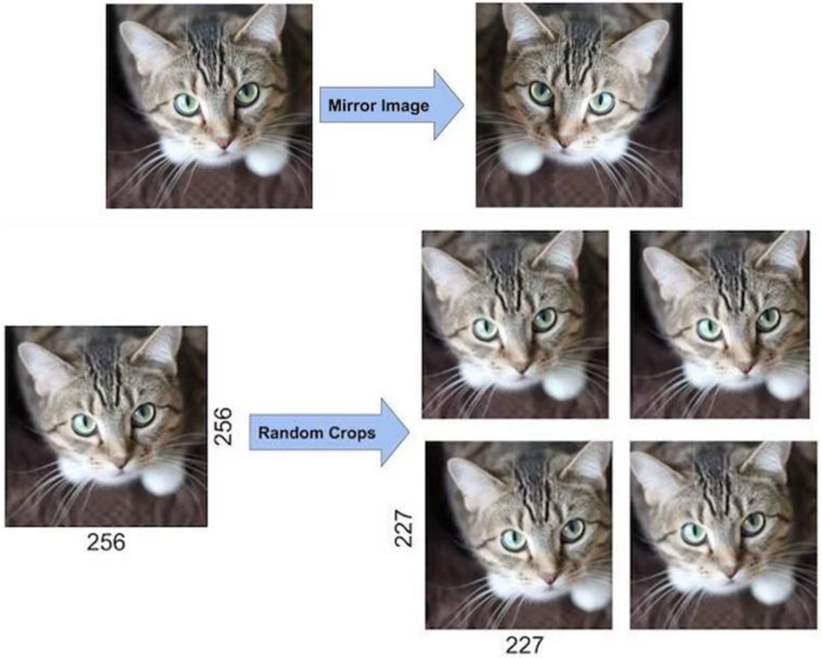
Pooling window 의 크기 = stride의 크기



Dropout & Data Augmentation



(a) Standard Neural Net (b) After applying dropout.



VGG 16

VGG - Visual Geometry Group

- Oxford University에서 개발되었으며, 2014 ImageNet Challenge에서 준우승을 한 모델

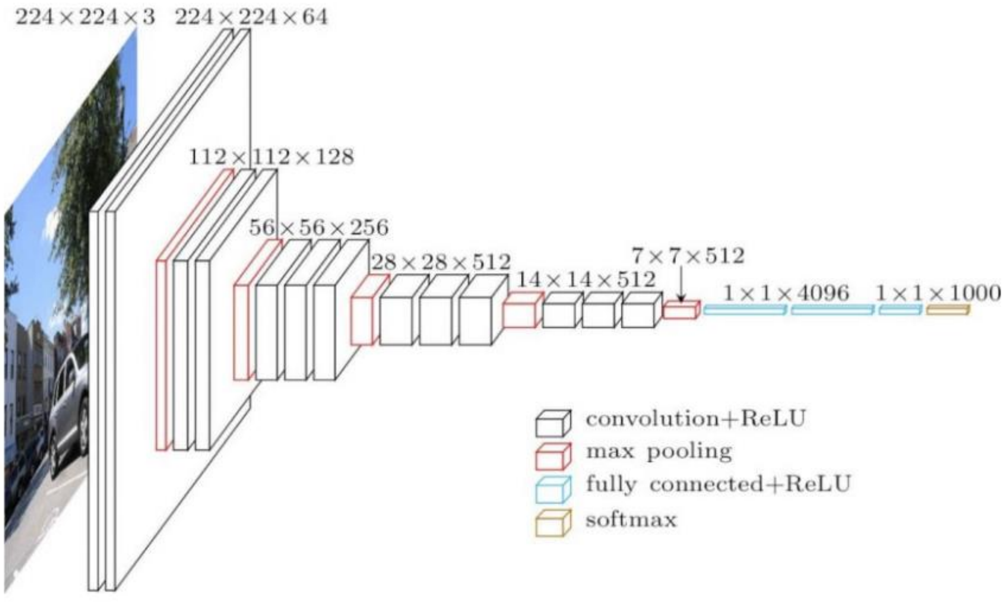
- VGGNet 연구의 핵심은 '네트워크의 깊이'가 성능에 어떤 영향을 미치는지를 확인하고자 한 것. 따라서 컨볼루션 필터의 사이즈를 가장 작은 3x3으로 고정함.

- VGG 연구팀의 실험 결과를 통해 네트워크의 깊이가 깊어질수록 이미지 분류 정확도가 높아지는 것을 확인할 수 있었음.

VGG 16 VGG 19

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

VGG-16



VGG16 Architecture

VGG-16 Architecture의 구성:

- 13 Convolution Layers + 3 Fully-connected Layers
- 3x3 convolution filters
- stride: 1 & padding: 1
- 2x2 max pooling (stride : 2)
- ReLU

KU-BIG

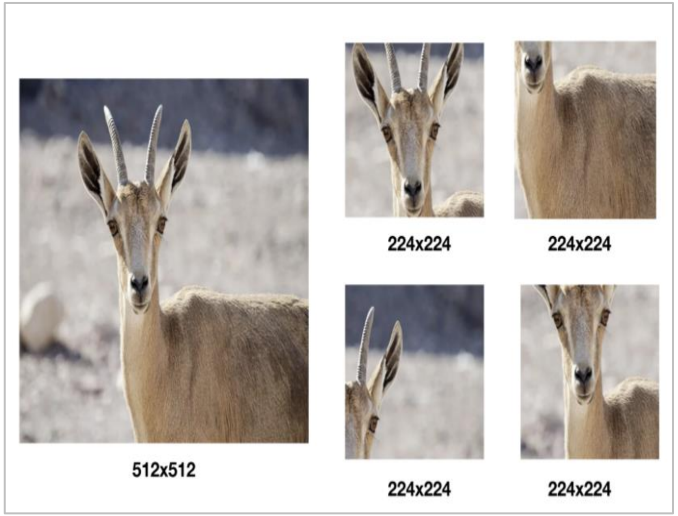
VGG 특징

- 모든 Conv Layer에 3x3 필터를 사용하여 파라미터 수가 감소하며, 층을 여러 개 쌓으므로 비선형성이 증가하게 됨.

- 1x1 Conv layer를 사용하여 의사결정함수에 Non-linearity를 부여

- Conv Layer의 수와는 관계없이 다섯 장의 고정된 Pooling Layer만을 사용함.

- 학습 데이터를 다양한 크기로 변환하고 그 중 일부분을 샘플링해 사용하여 한정적인 데이터의 수를 늘림. (data augmentation)



VGG 결론

- GoogLeNet에 비해 굉장히 간단한 구조를 취하면서도 거의 상이한 퍼포먼스를 보여줬다는 점에서, 신경망의 깊이의 유효성을 증명함.
- 네 개의 NVIDIA Titan Black GPU를 사용했음에도 네트워크 하나를 학습시키는데 2~3주가 걸렸다는 비효율의 문제를 안고 있음.

MobileNet

MobileNet

- 사물 인터넷(IoT)이나 5G 같은 저전력 통신망과 딥러닝이 발달하면서, 다양한 곳에서 딥러닝을 활용하려는 시도

- MobileNet은 고성능이 아닌 환경(컴퓨터 성능이 제한되거나 배터리 퍼포먼스가 중요한 상황)에서 딥러닝 모델을 구현하기 위해 설계됨

- Depthwise Separable Convolutions

→ *" 딥러닝 모델의 경량화 "*

Google Alphago Tpu Computer

Figure 1. MobileNet models can be applied to various recognition tasks for efficient on device intelligence.

MobileNet

- 19개의 layers

- Stride=1 or 2

- 3 x 3 kernels

- 동일한 구조가 여러 번 씩 반복되는 구성

- 3x3 -> 1x1 순서로 conv block 구성

- ReLU 활성화 함수 & Batch Normalization

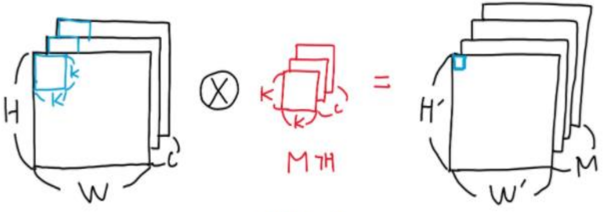
Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

MobileNet

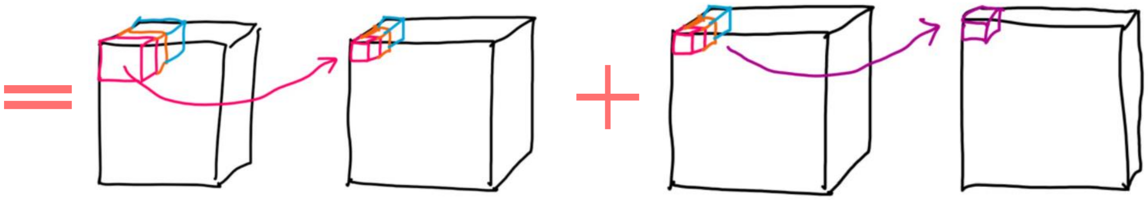
✓ 기존 CNN의 구조

✓ Depthwise Separable Convolution

→ Standard convolution = Depthwise conv + Pointwise conv



기존 CNN의 구조

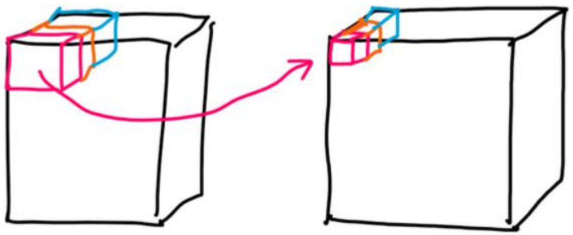


Depthwise conv. Pointwise conv.

출처: http://melonicedlatte.com/machinelearning/2019/11/01/212800.html

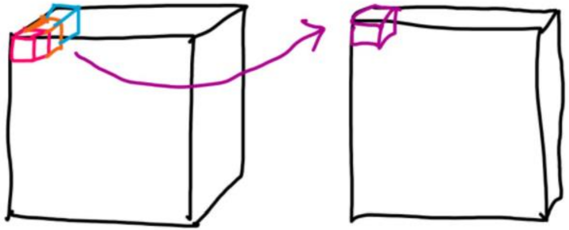
MobileNet

✓ Depthwise Convolution



Kernel을 width * height * depth(=1)로 설정하여 입력 이미지의 각 channel마다 독립적으로 convolution 진행

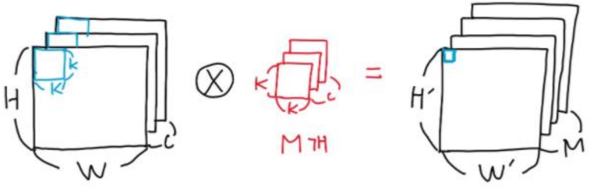
✓ Pointwise Convolution



Kernel을 width(=1) * height(=1) * depth로 설정하여 1 by 1 convolution 진행

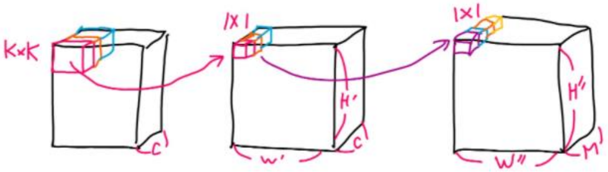
출처: http://melonicedlatte.com/machinelearning/2019/11/01/212800.html

MobileNet



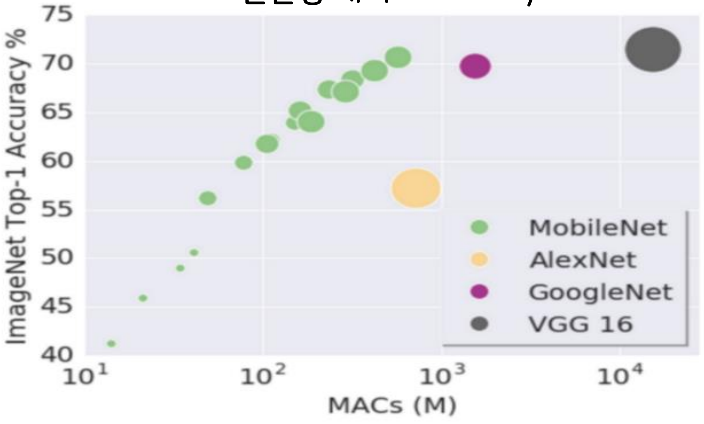
기존 CNN의 구조

- ✓ 기존 CNN 연산량: K^2CMH'W'



- ✓ Separable Convolution 연산량: K^2CW'H' + CMW"H"

연산량 대비 Accuracy



imagenet accuracy compare

$$W = W' = W'', H = H' = H''$$

$$\left(\frac{(K^2+M)WHC}{(K^2M)WHC}\right) = \left(\frac{K^2+M}{K^2M}\right) = \left(\frac{1}{M} + \frac{1}{K^2}\right)$$

K=3일 경우 약 8~9배 더 효율적

KU-BIG

출처

- https://medium.com/@msmapark2/vgg16-%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0-very-deep-convolutional-networks-for-large-scale-image-recognition-6f748235242a
- http://blog.naver.com/PostView.nhn?blogId=siniphia&logNo=221376966415&parentCategoryNo=&categoryNo=23&viewDate=&isShowPopularPosts=true&from=search
- https://bskyvision.com/504
- http://melonicedlatte.com/machinelearning/2019/11/01/212800.html

Q & A