CNN 이란?
(Convolutional Neural Network)

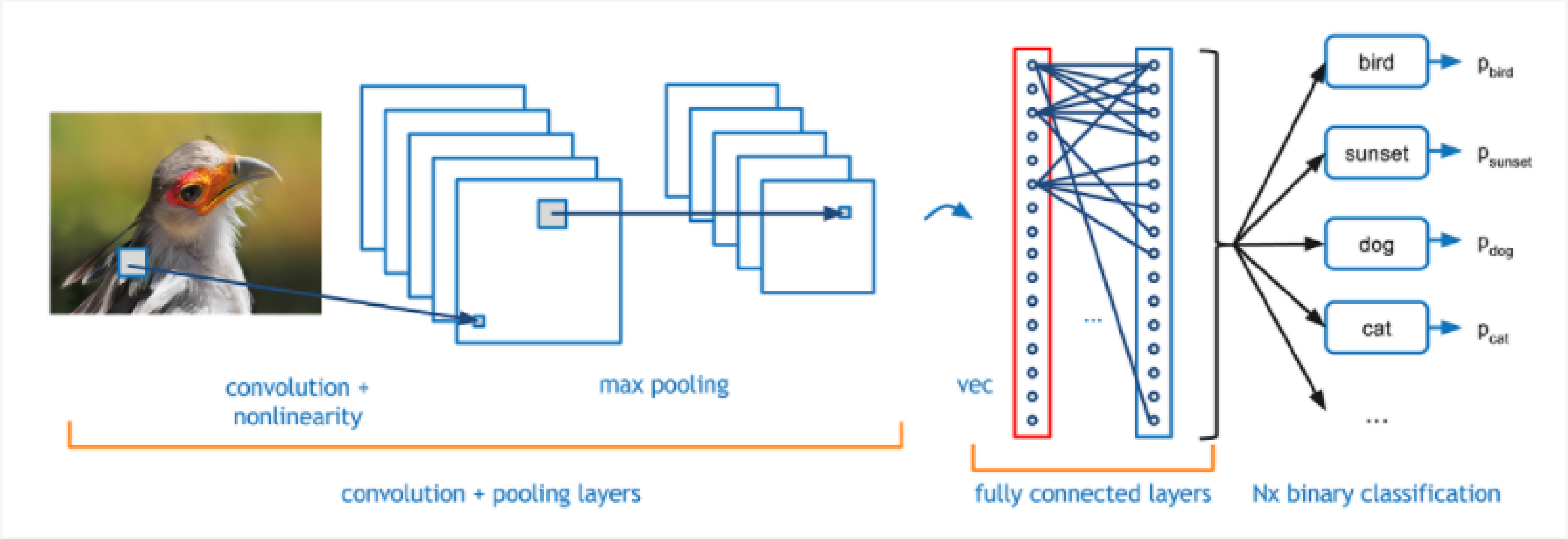
KUBIG 12기 이나윤

INDEX

2021년 여름방학 딥러닝 분반 3**주차**

01. CNN이란?

CNN(Convolutional Neural Network)

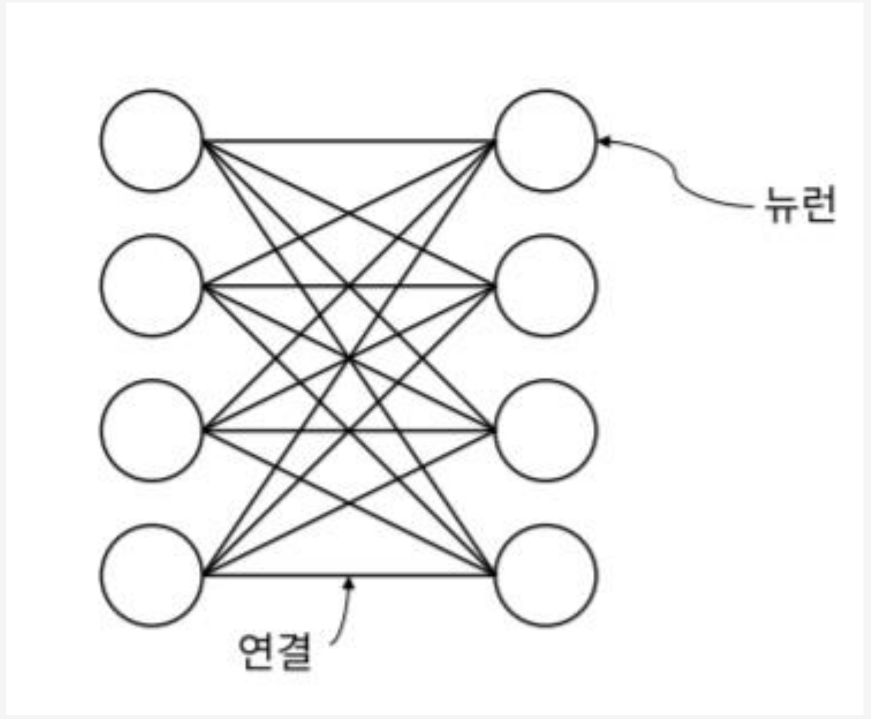


01. CNN – Convolutional Layer/ Fully Connected Layer

CNN(Convolutional Neural Network)

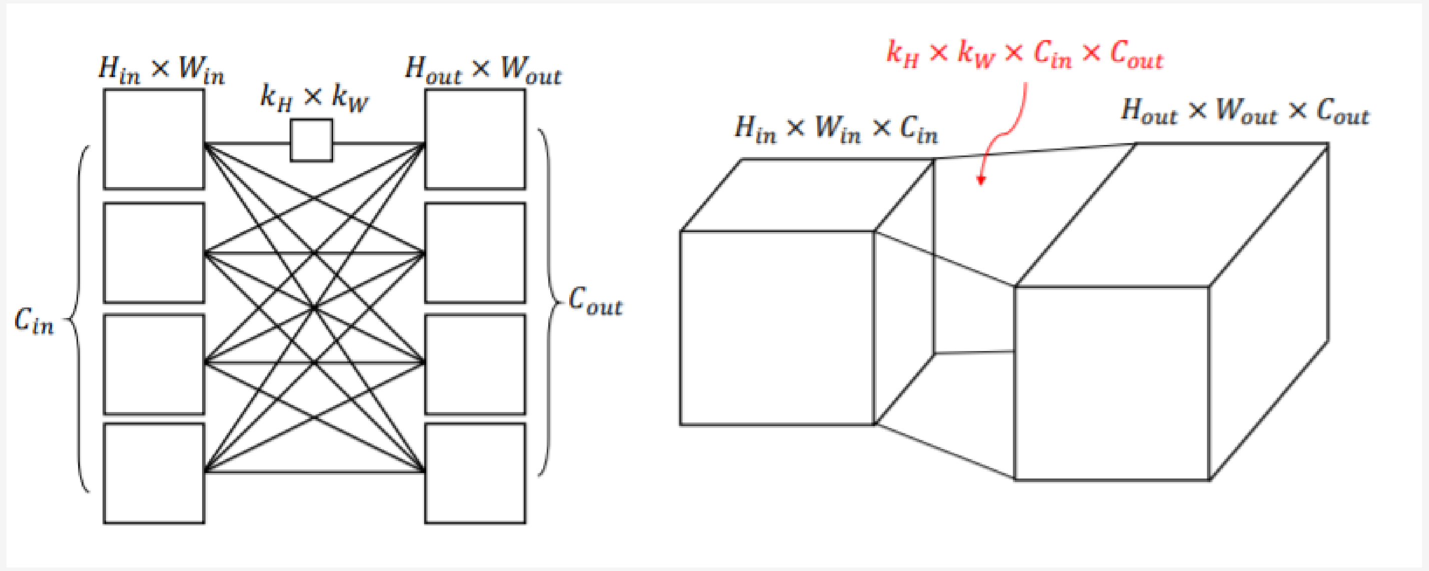
Fully Connected Layer

: 이전 계층의 모든 뉴런이 현재 계층의 모든 뉴런과 연결된 경우



Convolutional Layer

: 합성곱으로 이루어진 뉴런을 Fully Connected 형태로 연결한 경우

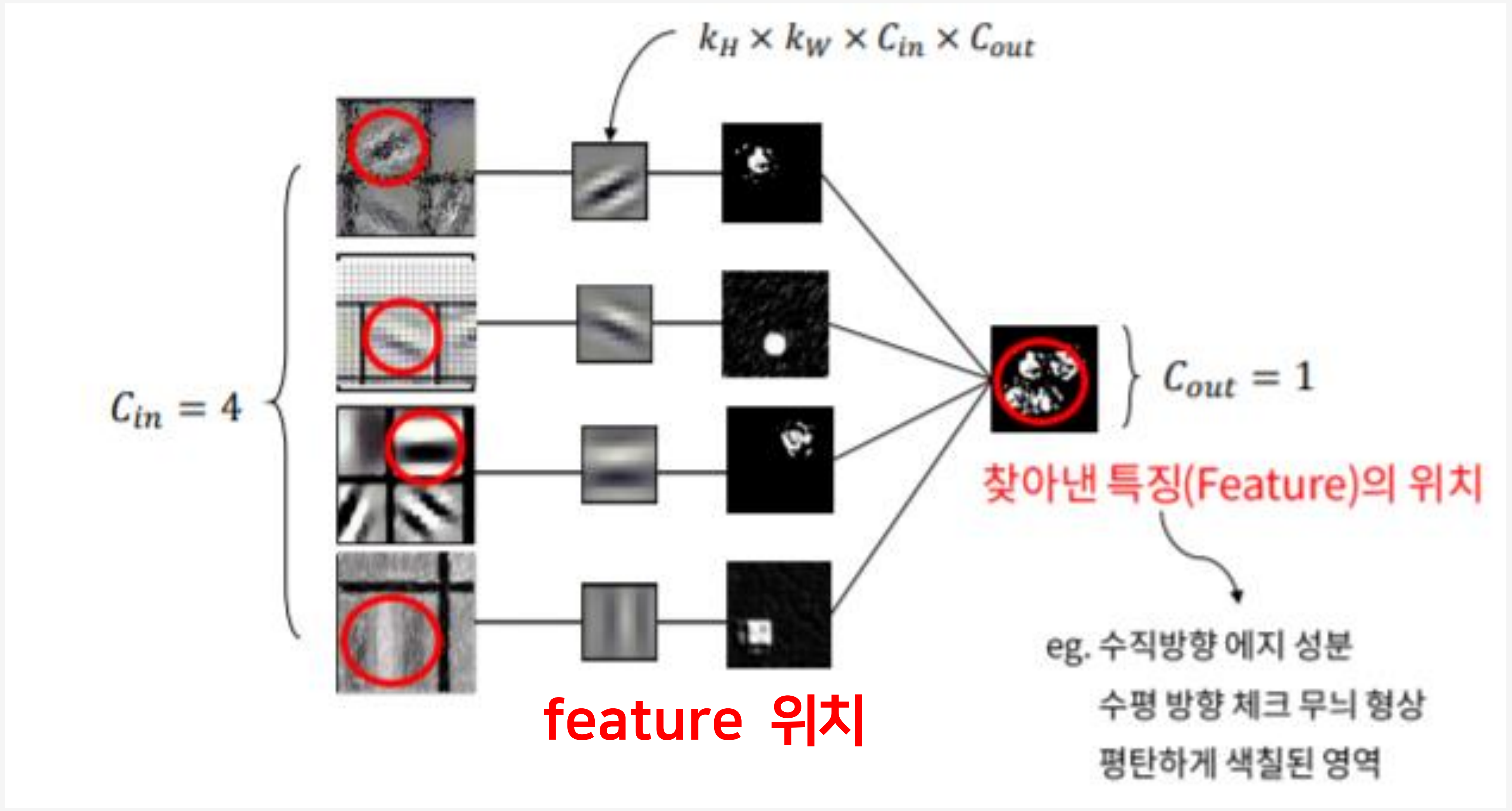

$H_{in} \times W_{in}$
$k_H \times k_W$
$H_{out} \times W_{out}$
C_{in}
C_{out}
$k_H \times k_W \times C_{in} \times C_{out}$
$H_{in} \times W_{in} \times C_{in}$
$H_{out} \times W_{out} \times C_{out}$

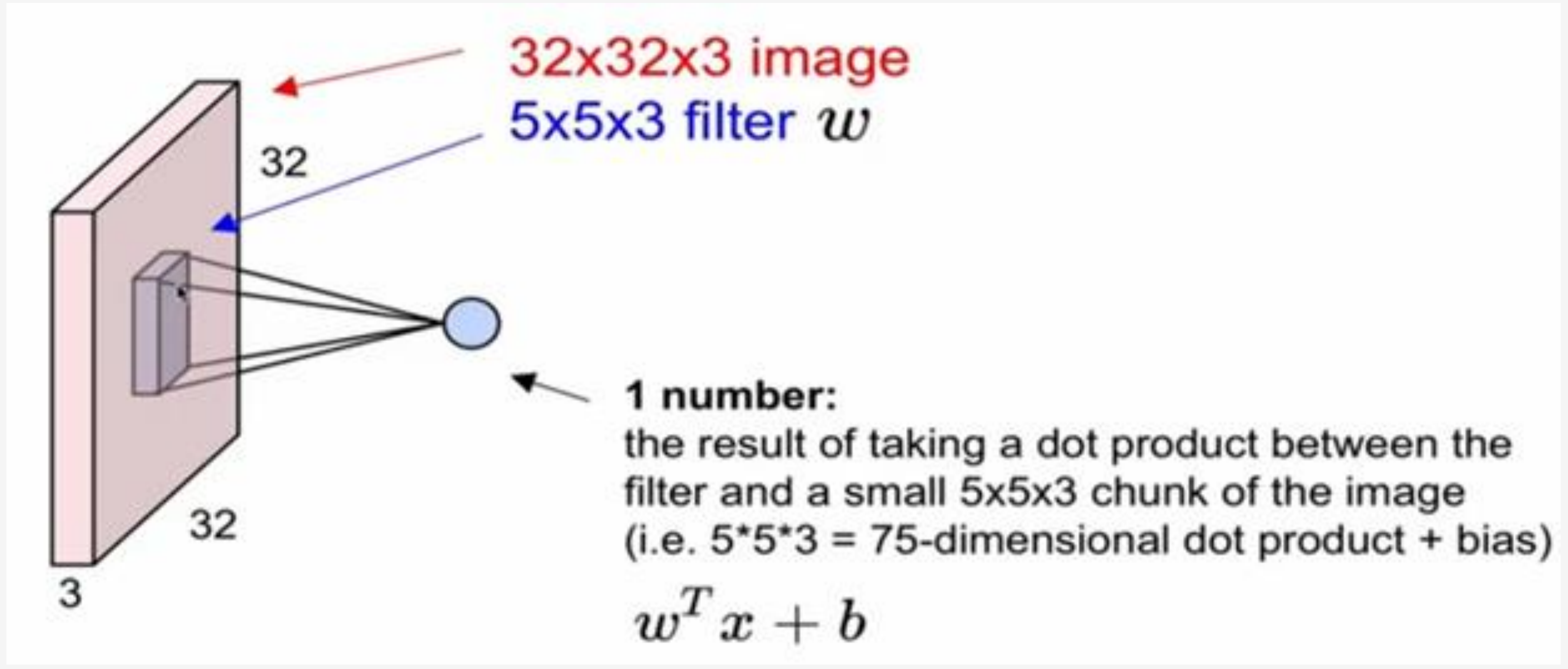
01. CNN – Convolutional Layer

Convolutional Layer

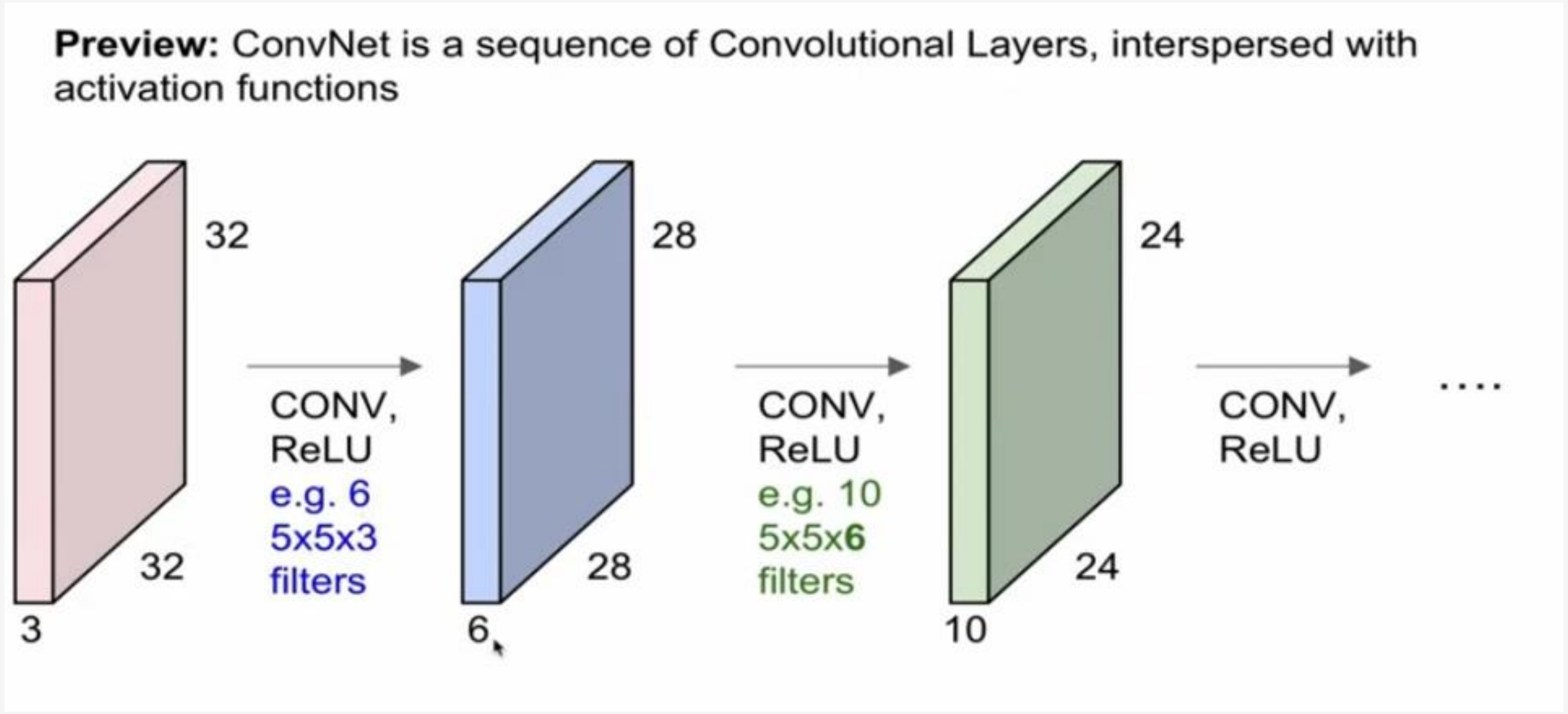
: Consolve the filter with the image

: Slide over the image spatially, computing dot products




32x32x3 image
5x5x3 filter w
32
32
3
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)
$w^T x + b$

01. CNN – Convolutional Layer


Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions
32
32
3
CONV, ReLU
e.g. 6 5x5x3 filters
28
28
6
CONV, ReLU
e.g. 10 5x5x6 filters
24
24
10
CONV, ReLU
....

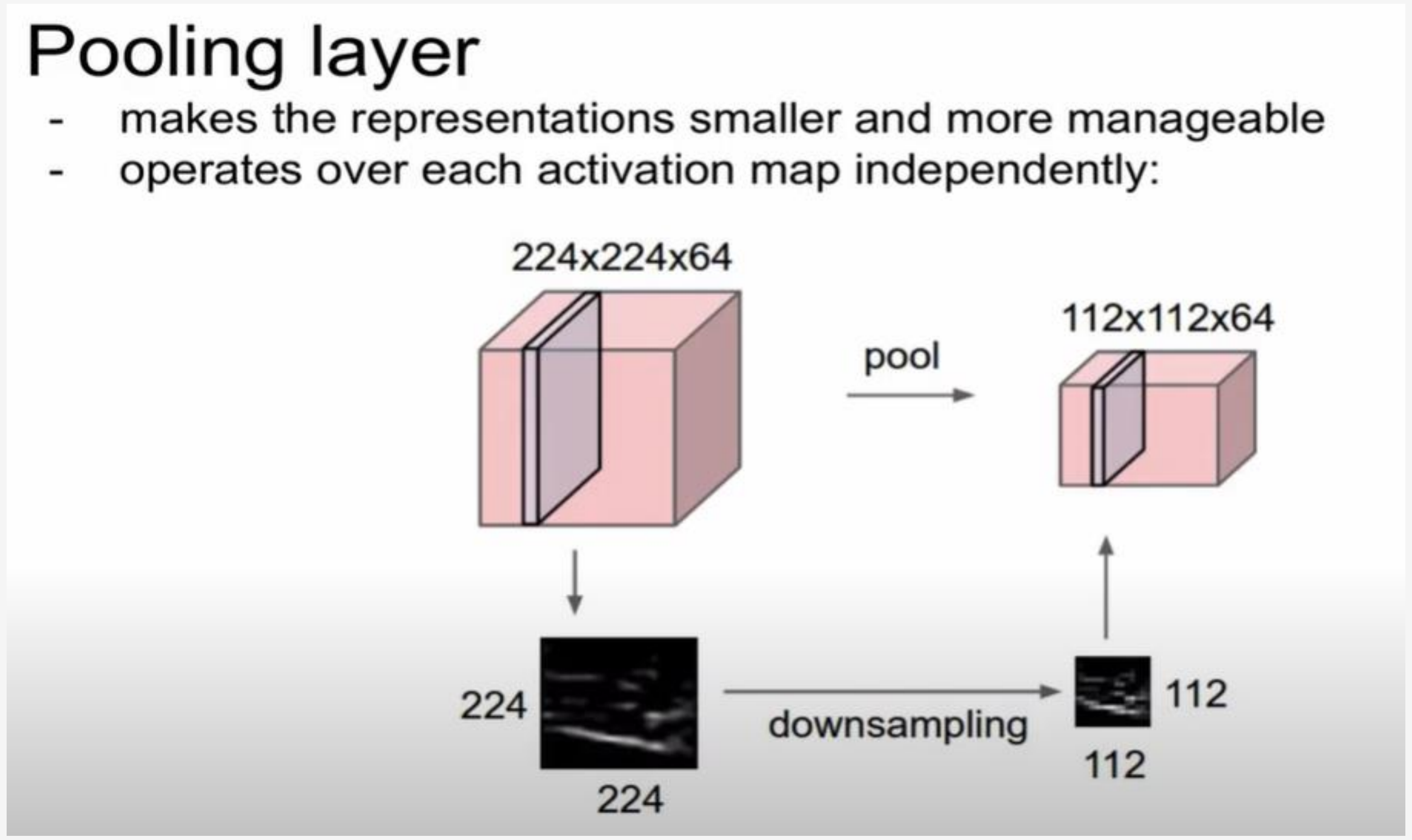
Output depth = filter(kernel) 개수

01. CNN – Pooling layer

Pooling Layer : convolution layer의 출력 데이터를 받아, 크기를 줄이거나
특정 데이터를 강조하는 용도로 사용 (정보 종합)

-> 학습대상 Parameter 없음

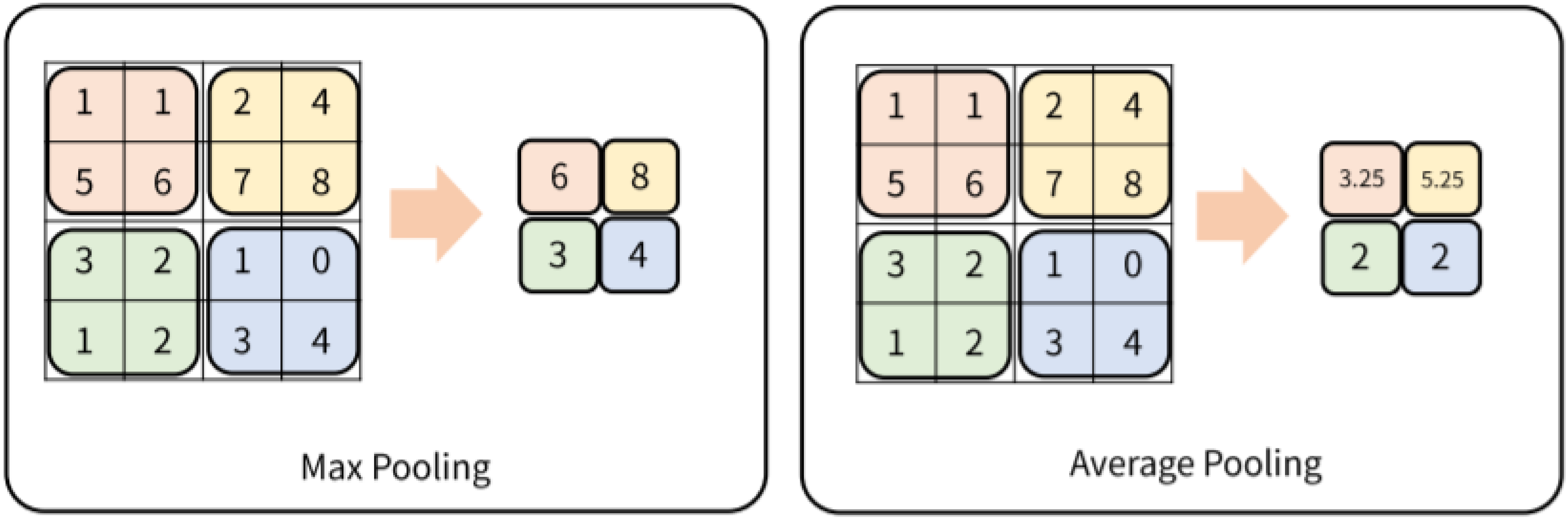
-> pooling layer를 지나도, 채널 수 변경 없음


Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:

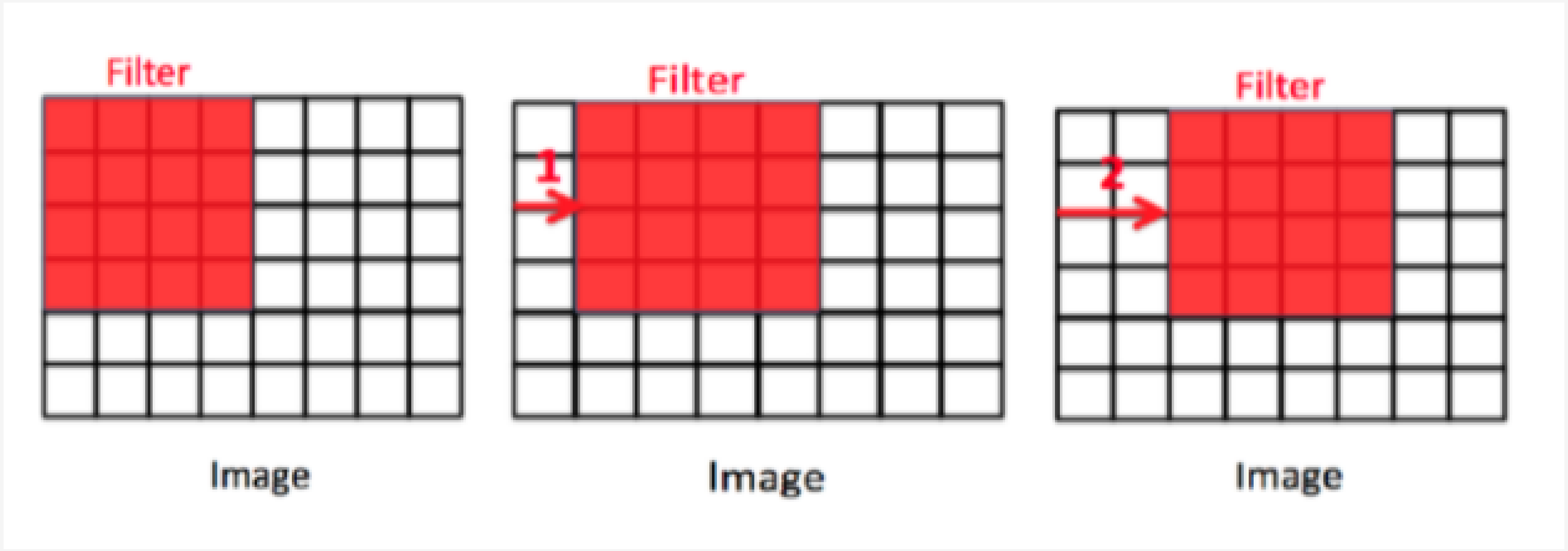
01. CNN – Pooling layer

Pooling Layer



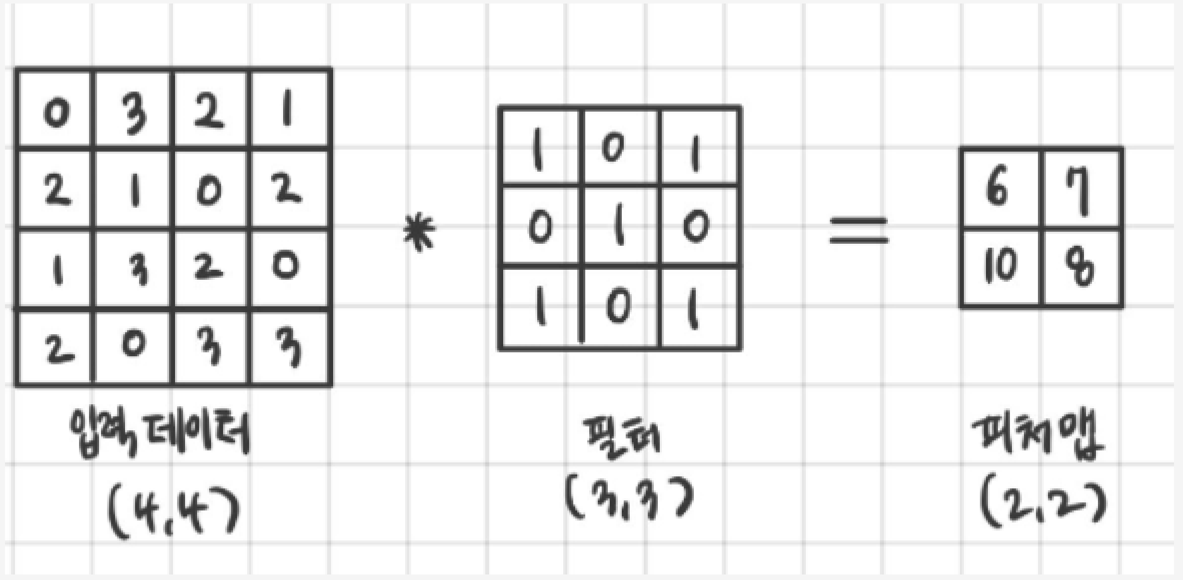
01. CNN - Stride

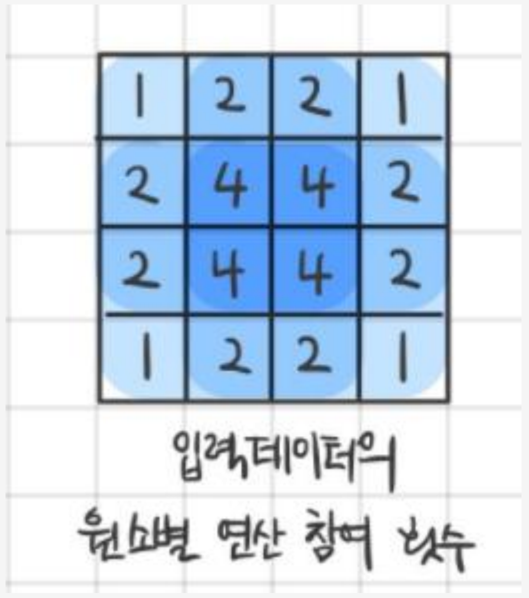
Stride: 입력 이미지에서 필터를 몇 칸씩 건너띄며 적용할지

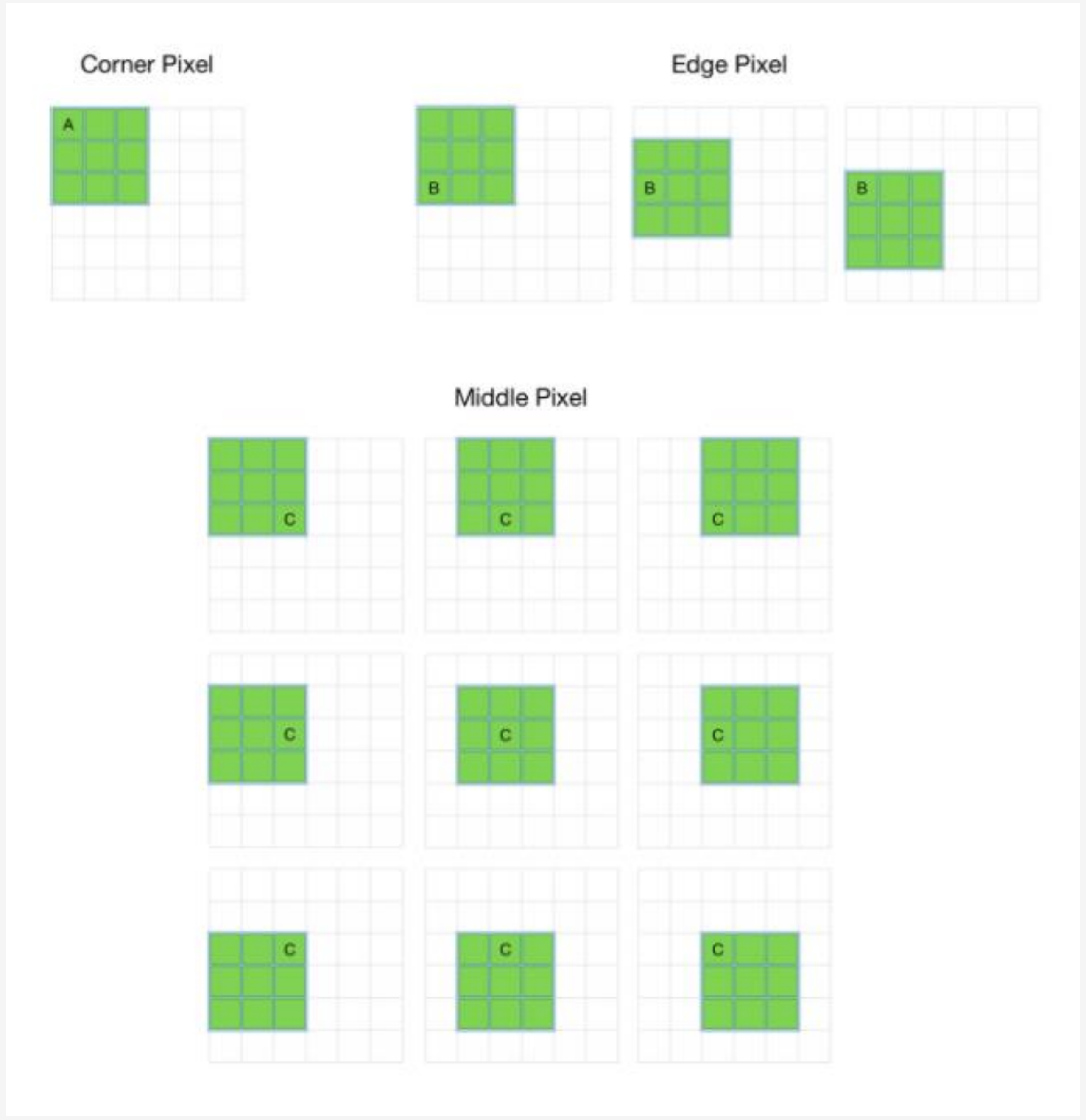


01. CNN - Padding

padding을 추가하지 않고, filter 통과시킬 경우:







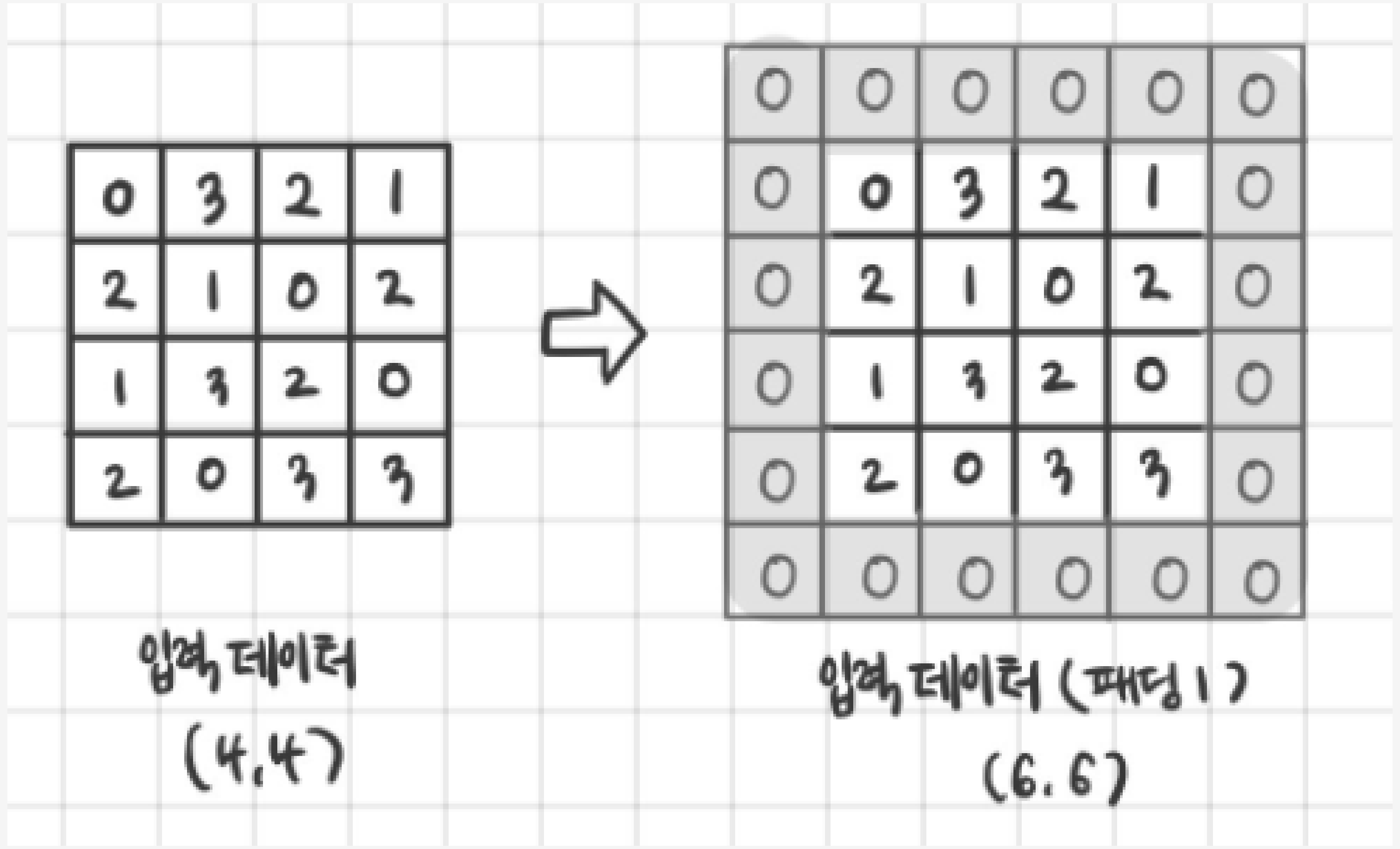
-> 이미지 데이터의 축소를 막기 위해

-> Edge data를 충분히 활용하기 위해

01. CNN - Padding

(zero) Padding:

출력 크기를 보정하기 위해 입력 데이터의 사방을 특정 값(일반적으로 0)으로 채우는 것



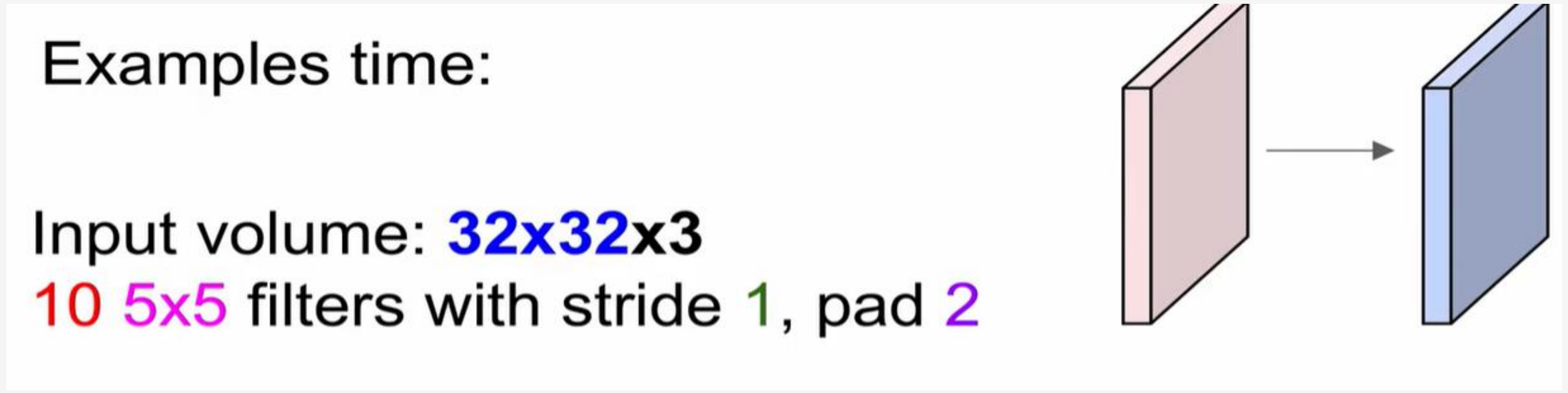
0	3	2	1
2	1	0	2
1	3	2	0
2	0	3	3

0	0	0	0	0	0
0	0	3	2	1	0
0	2	1	0	2	0
0	1	3	2	0	0
0	2	0	3	3	0
0	0	0	0	0	0

-> output size = (n+ 2*p-f)/s + 1(bias)

(n,n): origin data, p: padding, f: filter, s: stride

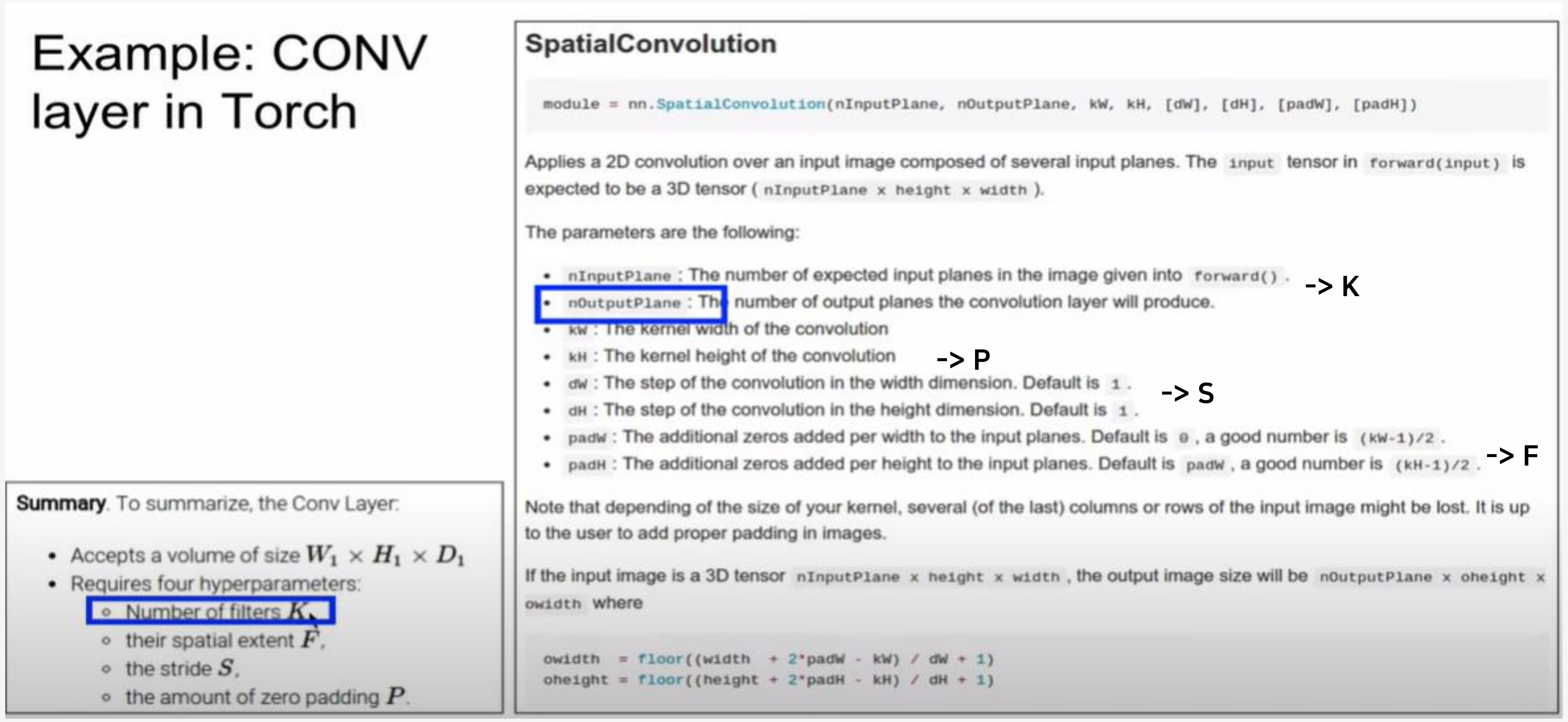
01. CNN이란?



Examples time:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10



Example: CONV layer in Torch

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.

SpatialConvolution

```
module = nn.SpatialConvolution(nInputPlane, nOutputPlane, kW, kH, [dW], [dH], [padW], [padH])
```

Applies a 2D convolution over an input image composed of several input planes. The `input` tensor in `forward(input)` is expected to be a 3D tensor (`nInputPlane x height x width`).

The parameters are the following:

- `nInputPlane` : The number of expected input planes in the image given into `forward()` . -> K
- `nOutputPlane` : The number of output planes the convolution layer will produce.
- `kW` : The kernel width of the convolution
- `kH` : The kernel height of the convolution -> P
- `dW` : The step of the convolution in the width dimension. Default is `1` . -> S
- `dH` : The step of the convolution in the height dimension. Default is `1` .
- `padW` : The additional zeros added per width to the input planes. Default is `0` , a good number is `(kW-1)/2` .
- `padH` : The additional zeros added per height to the input planes. Default is `padW` , a good number is `(kH-1)/2` . -> F

Note that depending of the size of your kernel, several (of the last) columns or rows of the input image might be lost. It is up to the user to add proper padding in images.

If the input image is a 3D tensor `nInputPlane x height x width` , the output image size will be `nOutputPlane x oheight x owidth` where

```
owidth  = floor((width  + 2*padW - kW) / dW + 1)
oheight = floor((height + 2*padH - kH) / dH + 1)
```

02. CNN의 흐름 및 모델

CNN(Convolutional Neural Network)

: 1989년 ‘Backpropagation applied to handwritten zip code recognition, Le Cun’

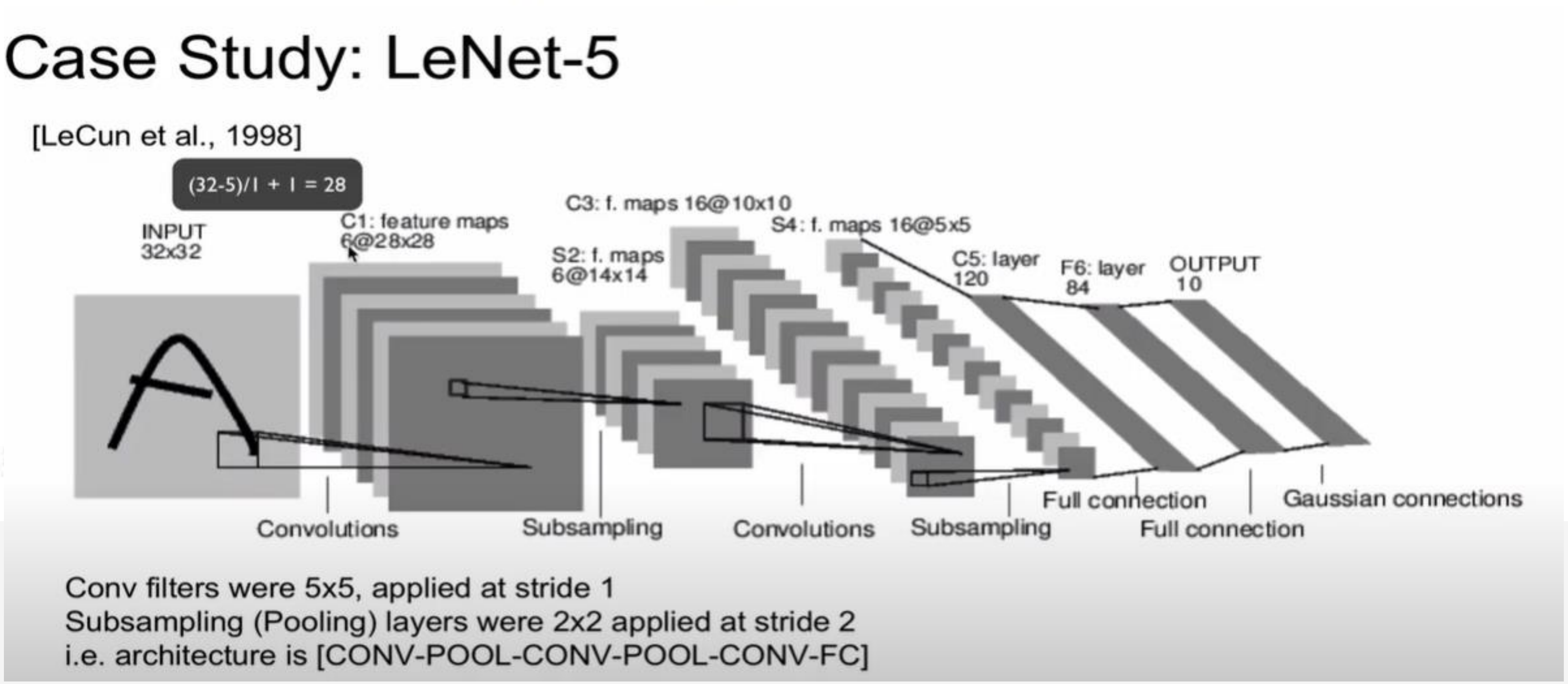
에서 처음 소개됨.

: 이미지를 flat하게 MLP로 학습하면, 지역적 정보가 날아가버림.

따라서, 바로 연산을 하여 지역적 정보를 잘 살릴 수 있는 모델인 CNN이 고안됨.

02. CNN의 흐름 및 모델

1998: LeNet – Gradient-based Learning Applied to Document Recognition



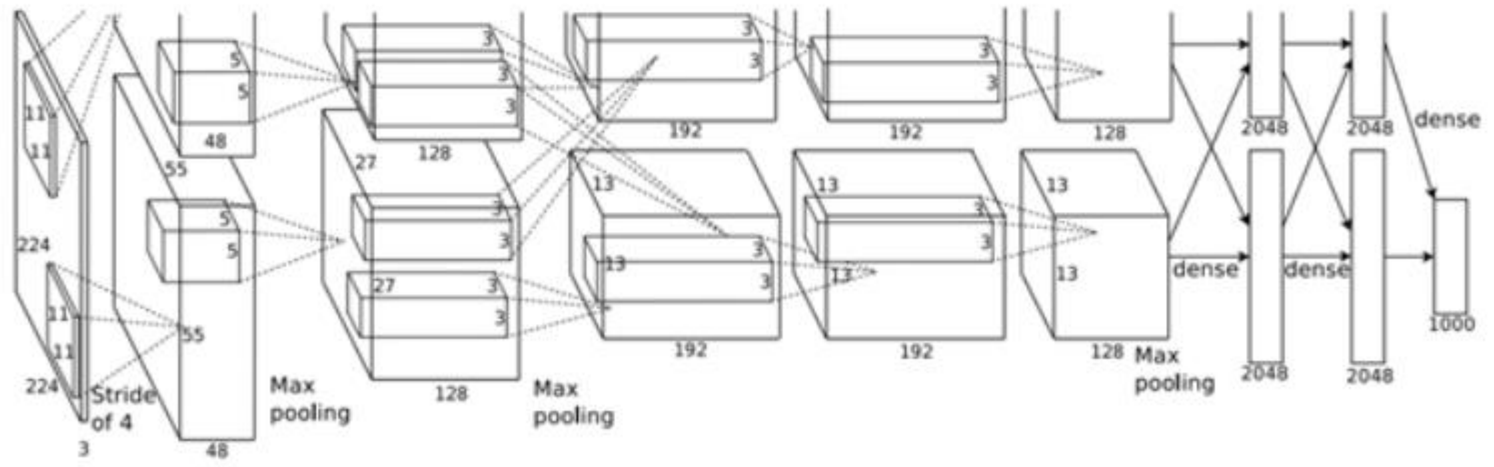
Case Study: LeNet-5

[LeCun et al., 1998]

Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC]

02. CNN의 흐름 및 모델

2012: AlexNet – ImageNet Classification with Deep Convolutional Neural Network

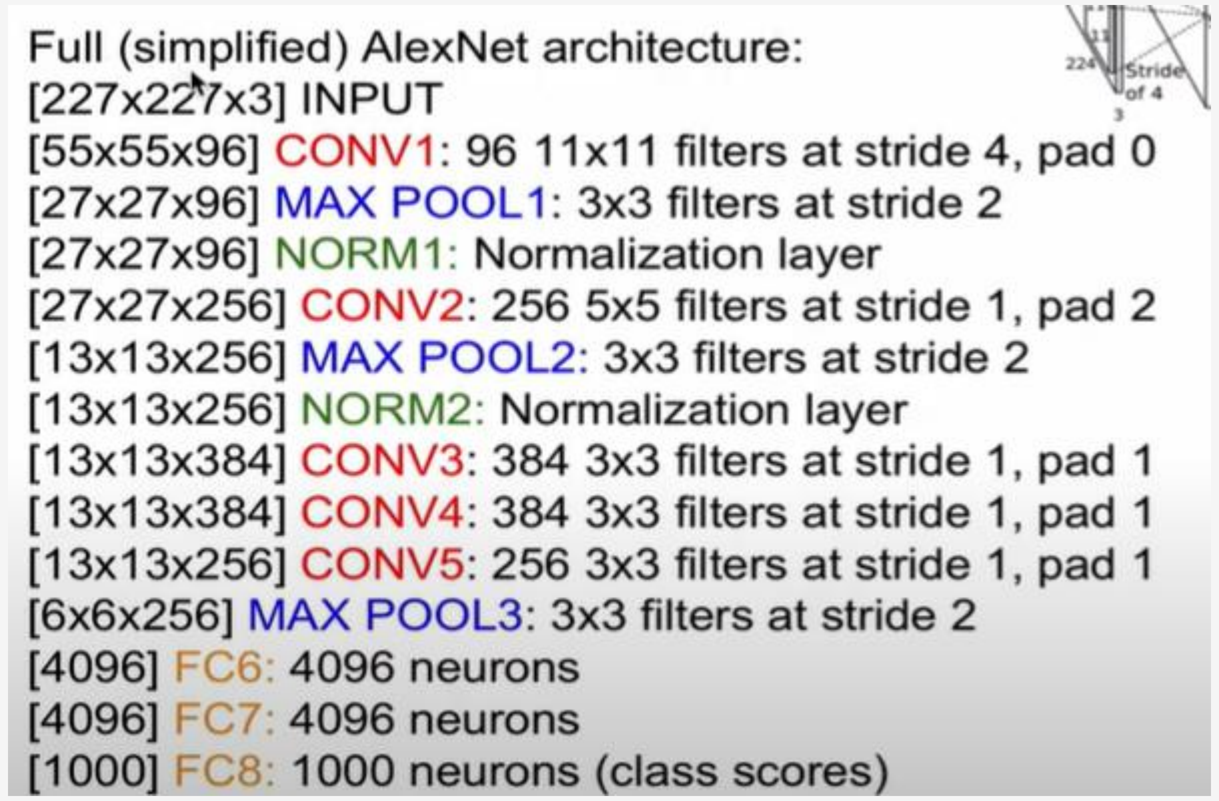


Input: 227x227x3 images

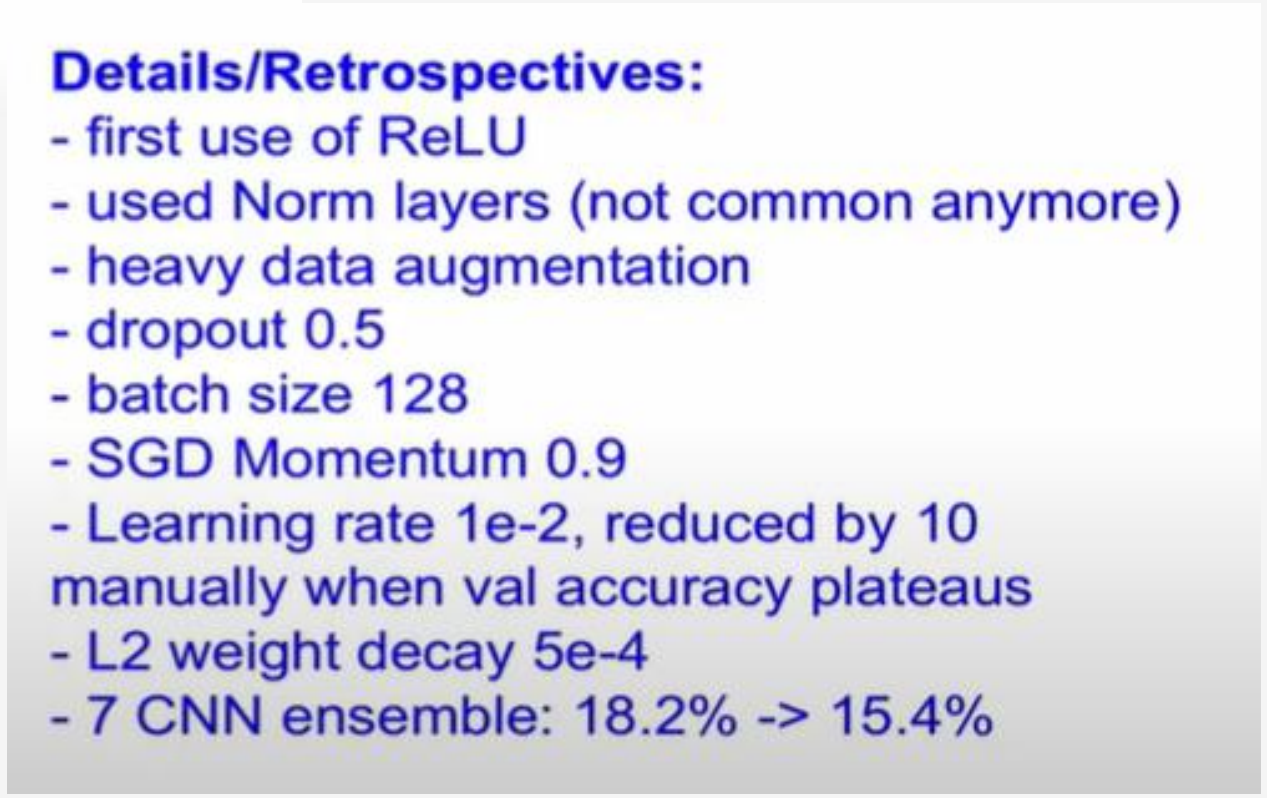
First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Q: what is the output volume size? Hint: (227-11)/4+1 = 55

Second layer (POOL1): 3x3 filters applied at stride 2

Q: what is the output volume size? Hint: (55-3)/2+1 = 27



Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)



Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10
manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

02. CNN의 흐름 및 모델

2014: VggNet – Very Deep Convolutional Networks for Large-Scale Image Recognition

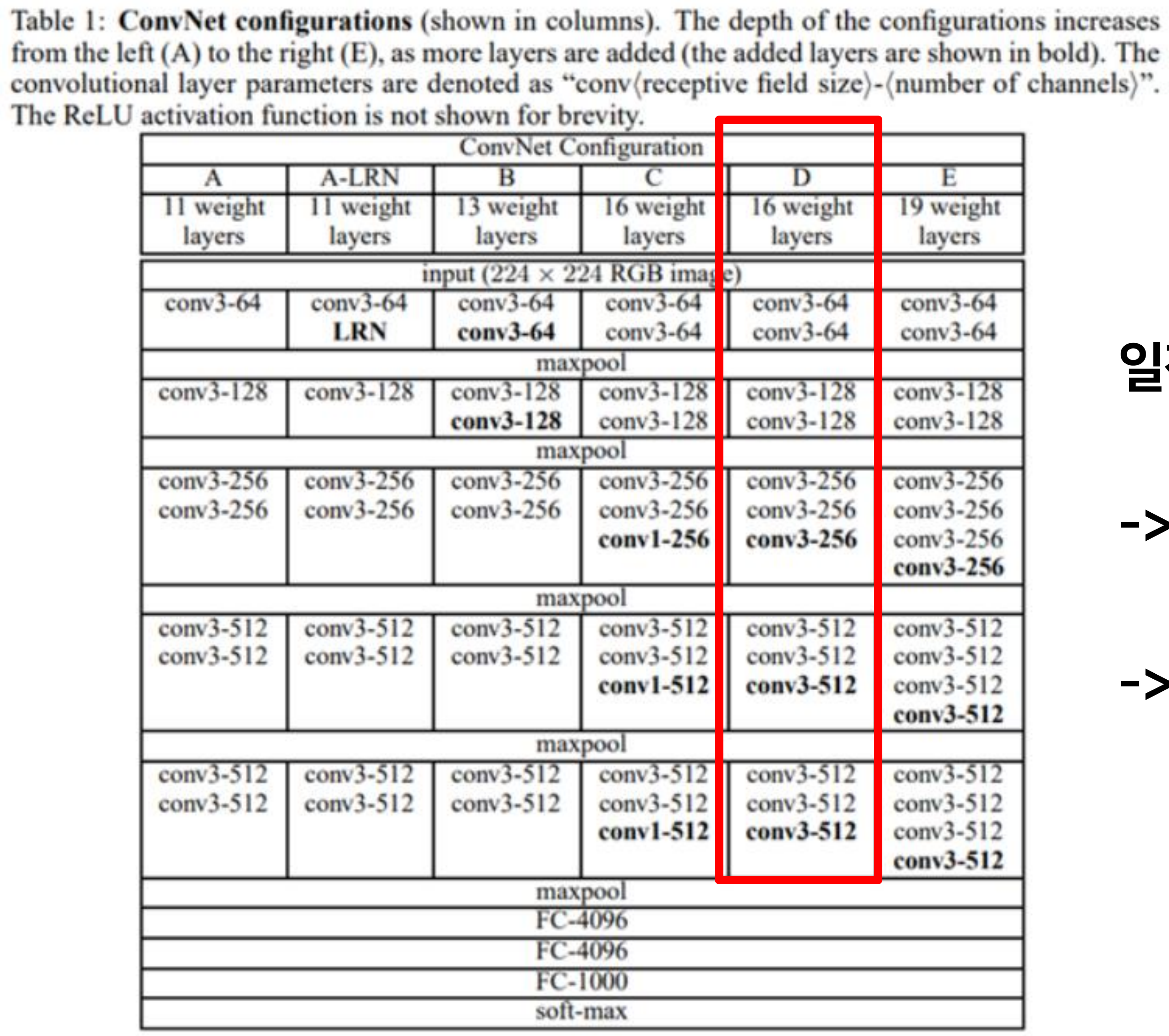


Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

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	conv3-64	conv3-64	conv3-64	conv3-64
	LRN	**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	conv3-256	conv3-256	conv3-256	conv3-256
			conv1-256	**conv3-256**	conv3-256
					conv3-256
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	**conv3-512**	conv3-512
					conv3-512
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	**conv3-512**	conv3-512
					conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

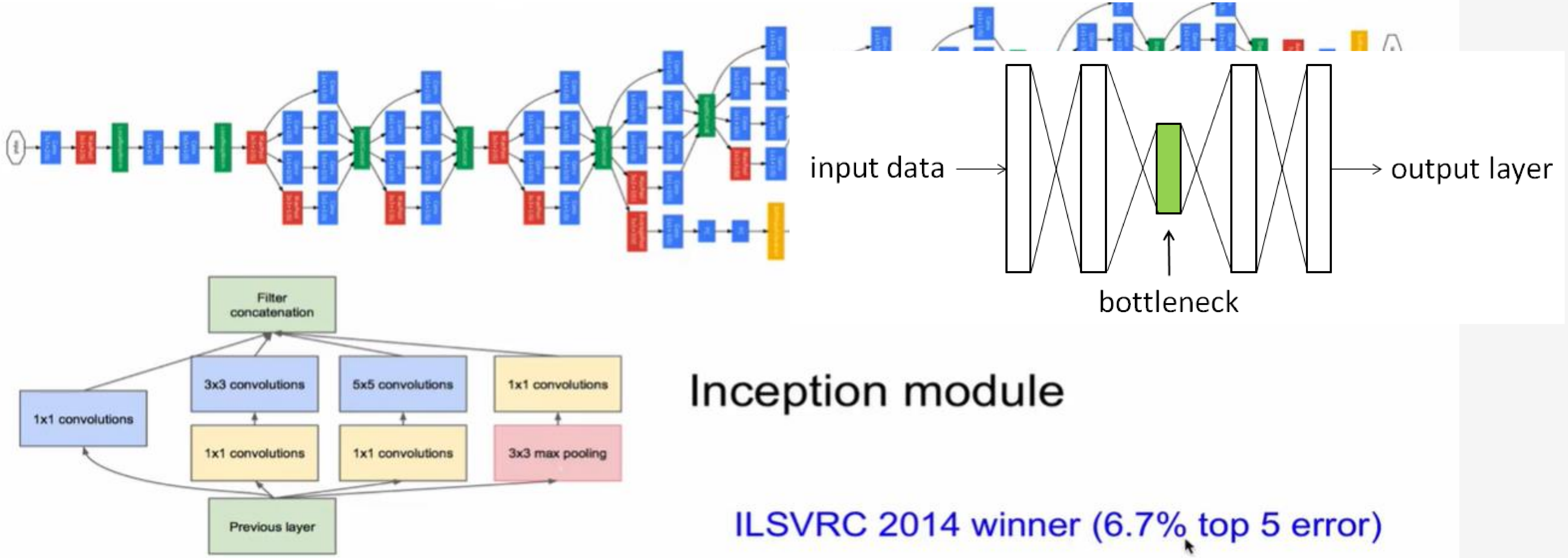
일정한 사이즈의 Conv filter (3*3) 중첩시켜 사용

-> 계산 비용 줄어듬, 성능 향상 O

-> 그러나, Gradient Vanishing 문제 발생

02. CNN의 흐름 및 모델

2014: GooLeNet – Going Deeper with Convolutions

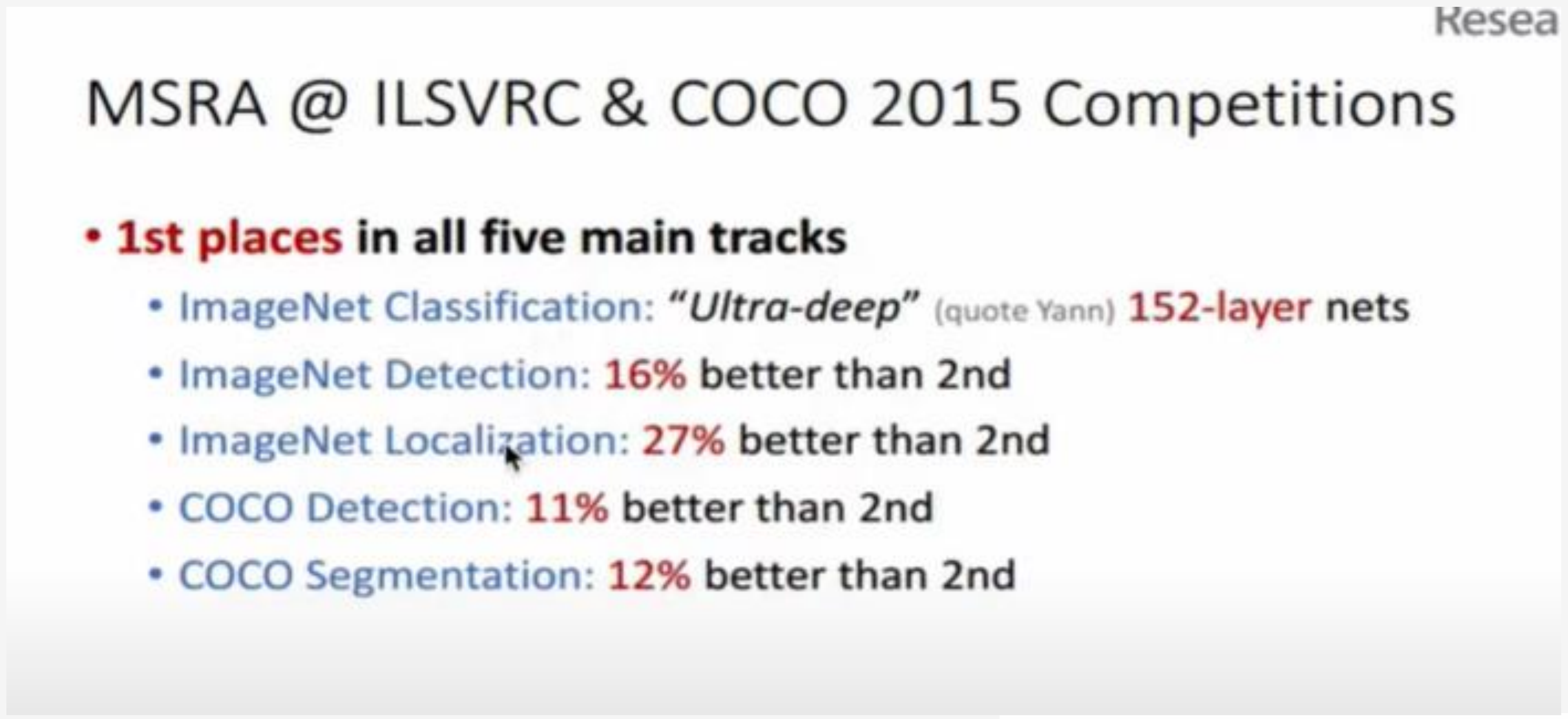


Inception module

ILSVRC 2014 winner (6.7% top 5 error)

https://www.youtube.com/watch?v=W9MlakX3vko

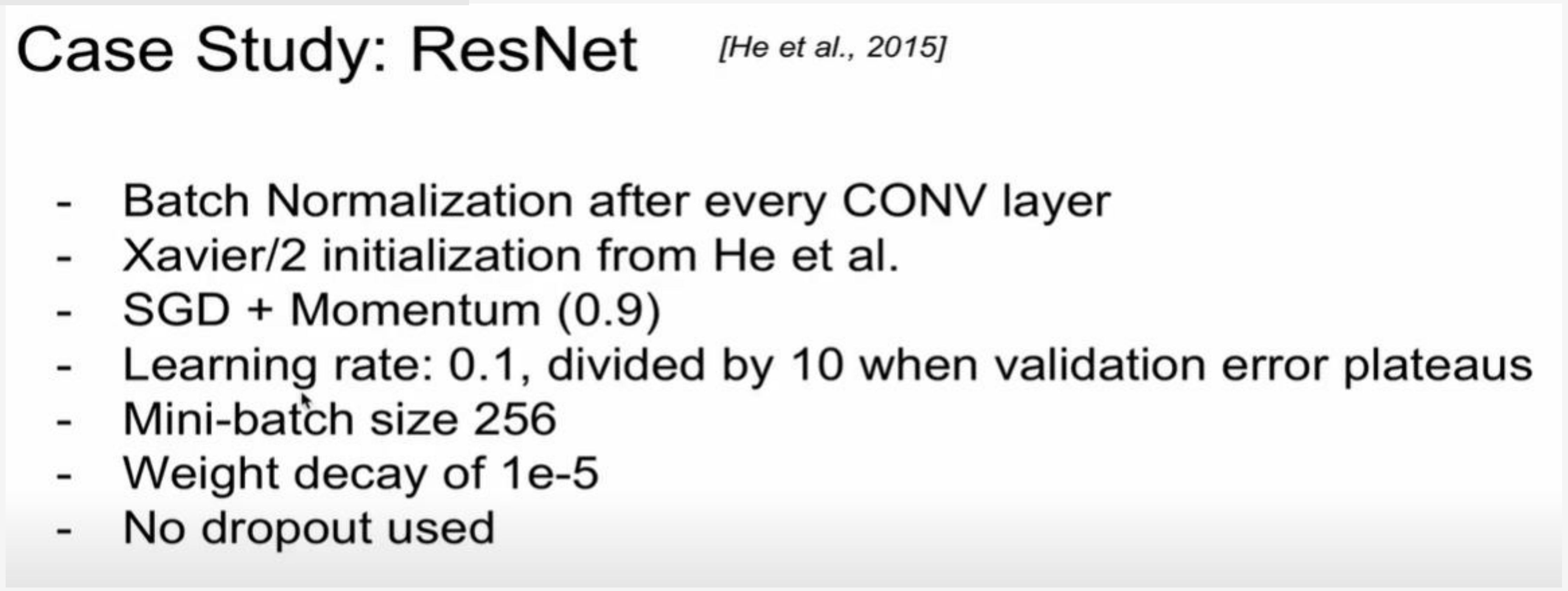
02. CNN의 흐름 및 모델



Resea

MSRA @ ILSVRC & COCO 2015 Competitions

- 1st places in all five main tracks
  - ImageNet Classification: *"Ultra-deep"* (quote Yann) 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd

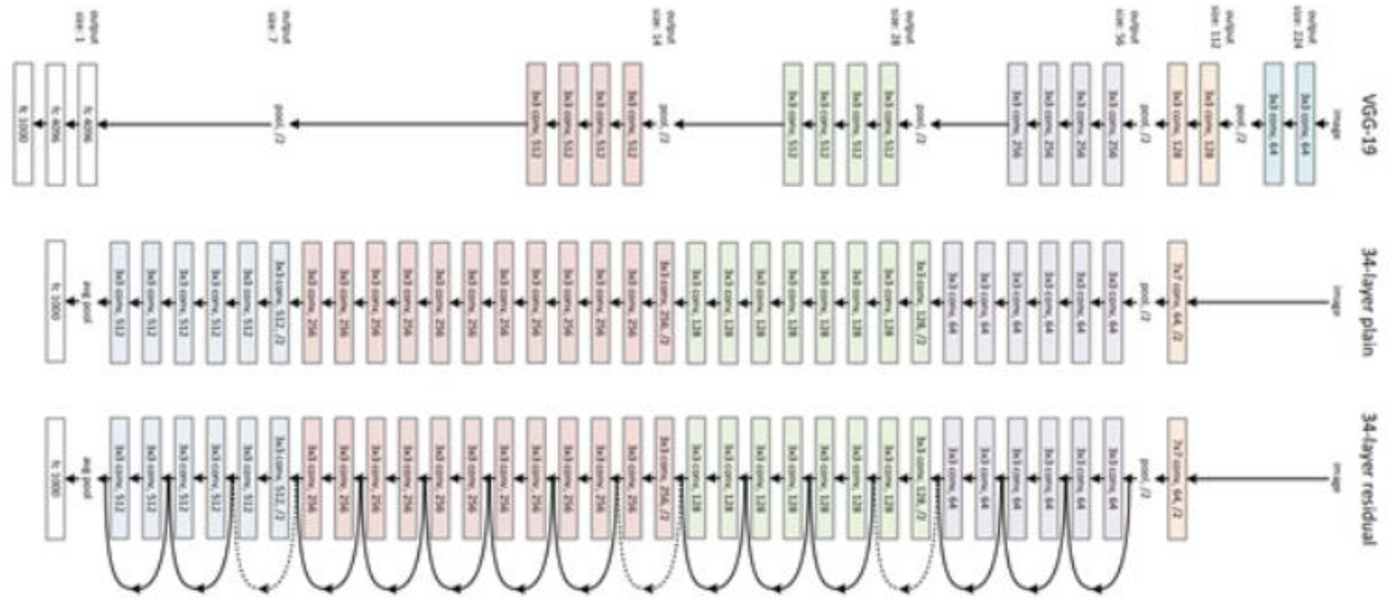


Case Study: ResNet *[He et al., 2015]*

- Batch Normalization after every CONV layer
- Xavier/2 initialization from He et al.
- SGD + Momentum (0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5
- No dropout used

02. CNN의 흐름 및 모델

2015: ResNet – Deep Residual Learning for Image Recognition



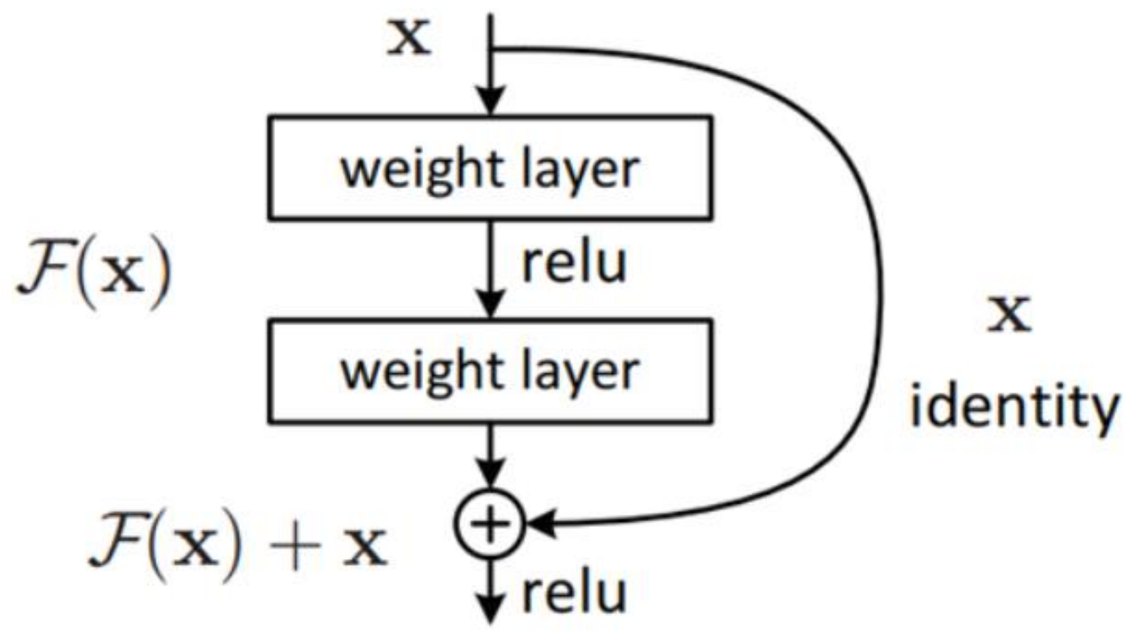
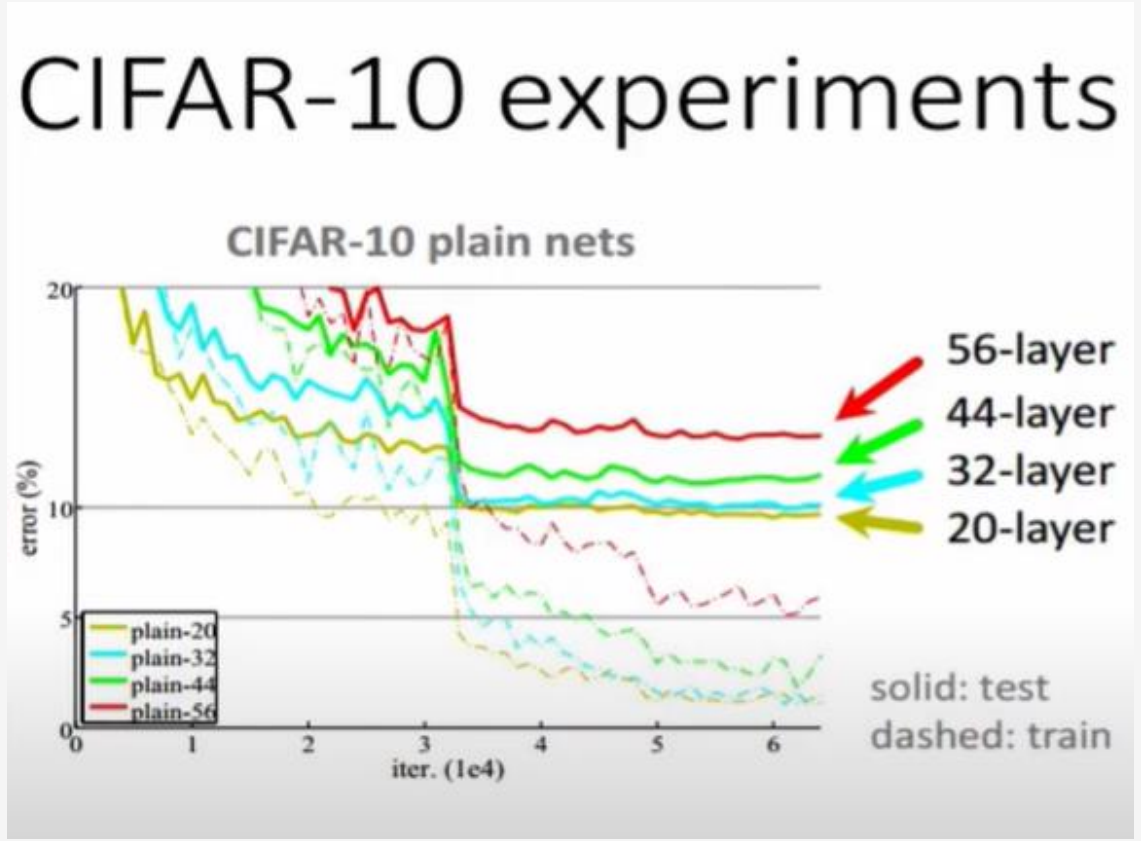
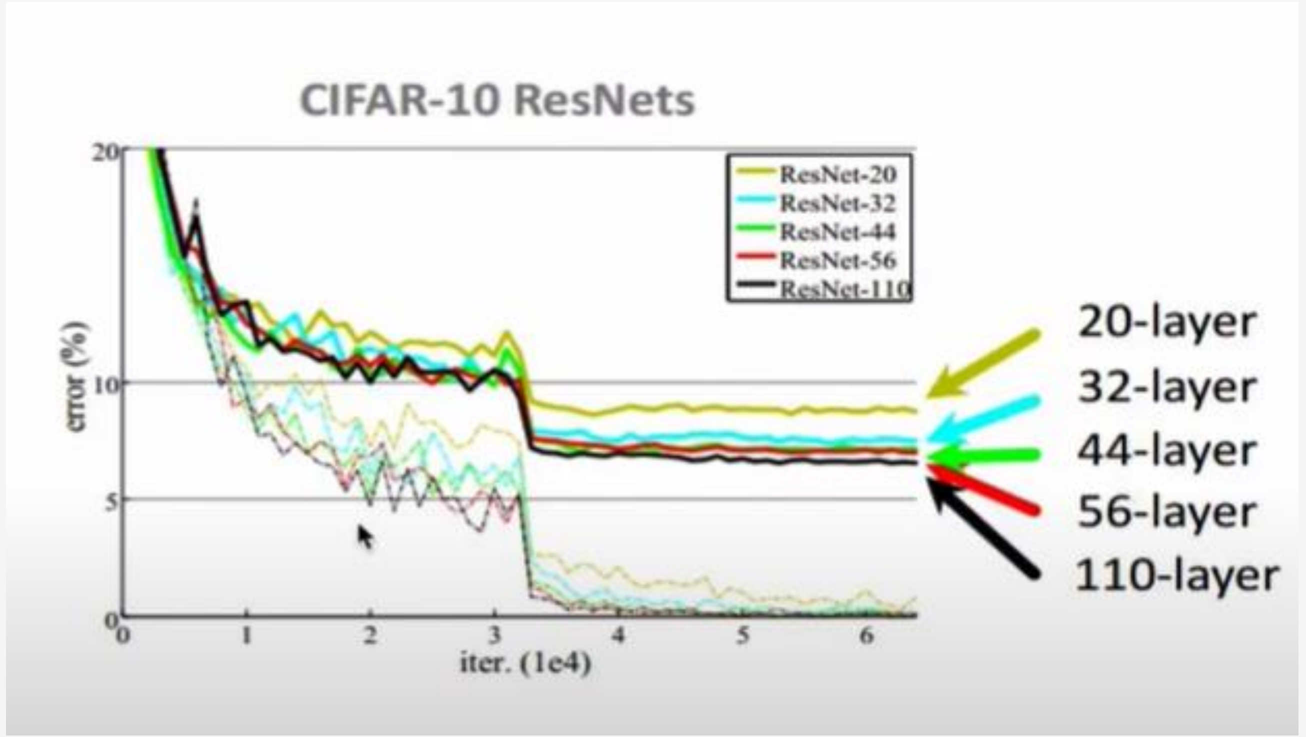

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

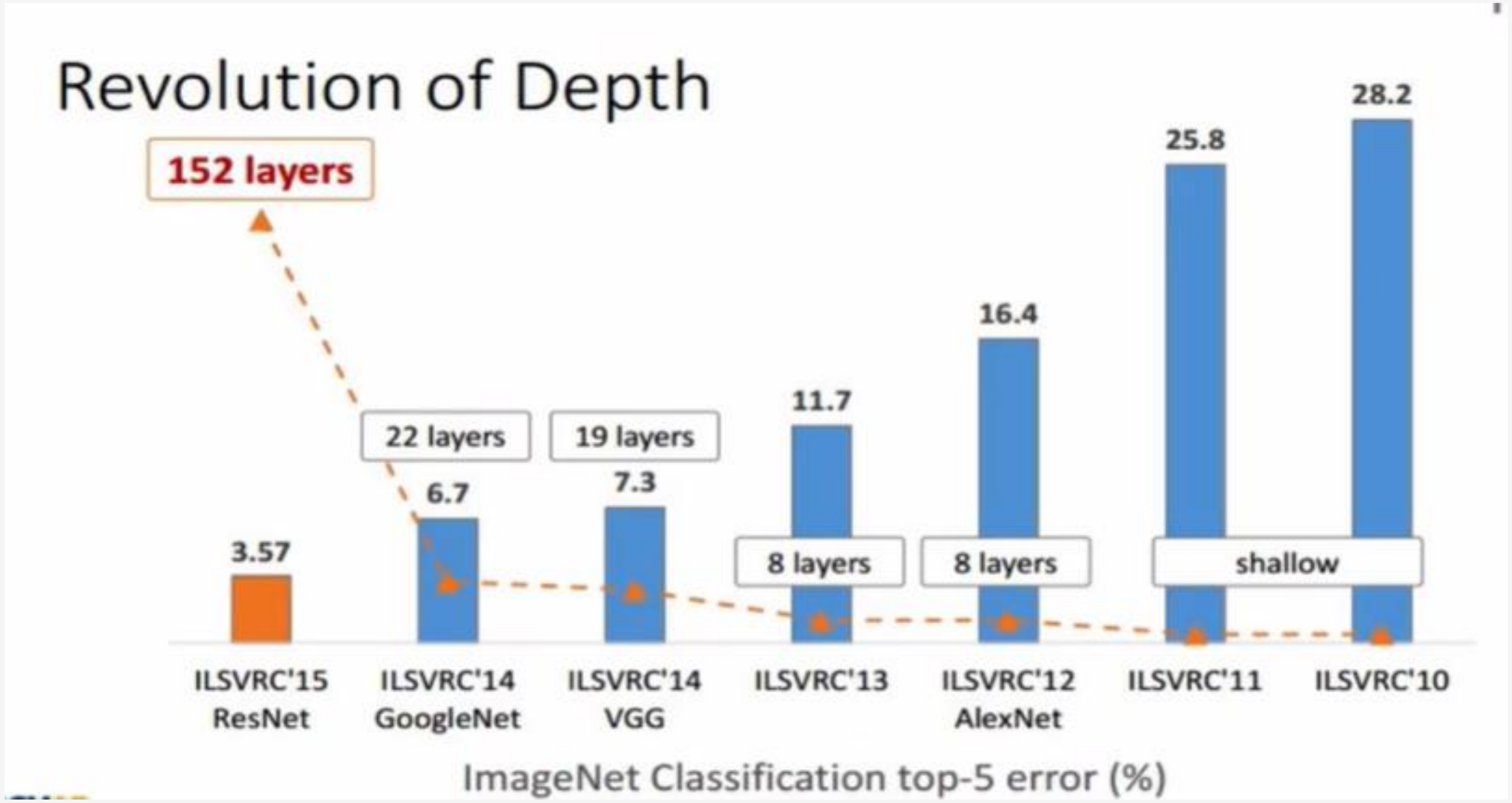
Figure 2. Residual learning: a building block.

Degradation problem



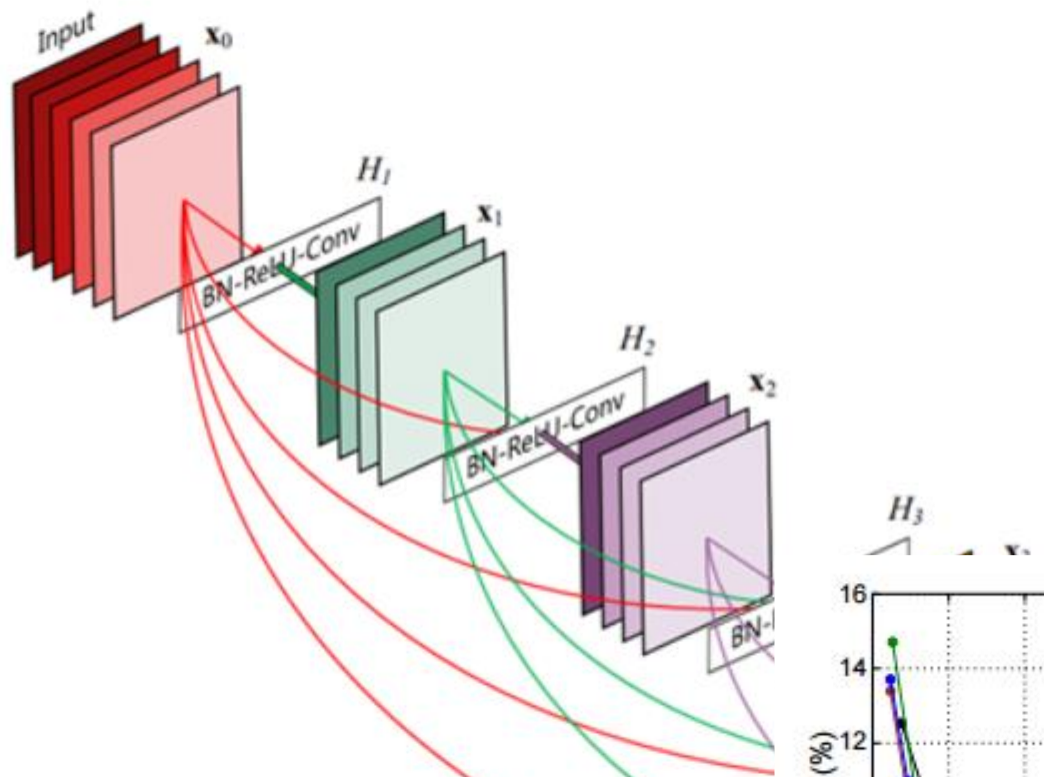


02. CNN의 흐름 및 모델



02. CNN의 흐름 및 모델

2017: DenseNet – Densely Connected Convolutional Networks



$$\mathbf{x}_\ell = H_\ell([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}]),$$

Figure 1: A 5-layer dense block w
Each layer takes all preceding feature

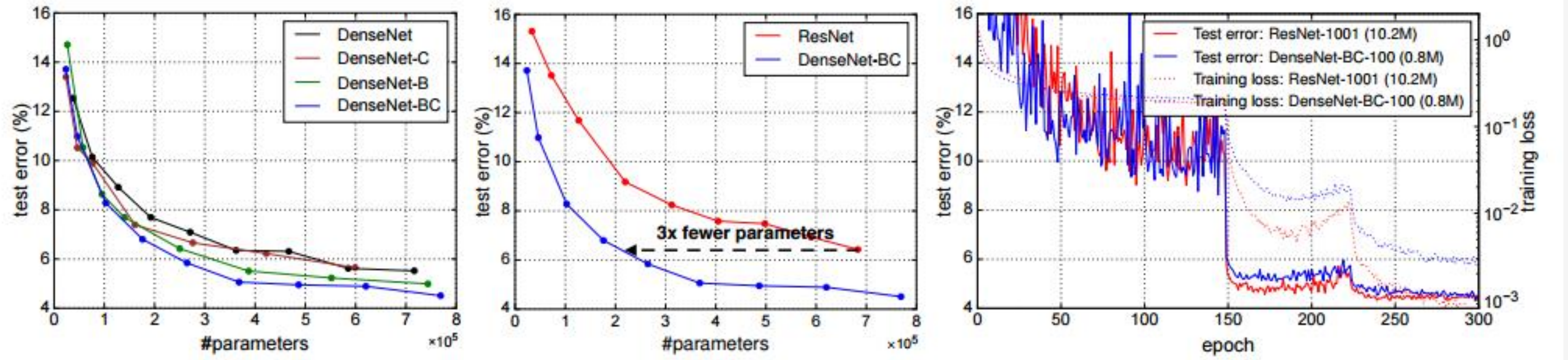


Figure 4: *Left:* Comparison of the parameter efficiency on C10+ between DenseNet variations. *Middle:* Comparison of the parameter efficiency between DenseNet-BC and (pre-activation) ResNets. DenseNet-BC requires about 1/3 of the parameters as ResNet to achieve comparable accuracy. *Right:* Training and testing curves of the 1001-layer pre-activation ResNet [12] with more than 10M parameters and a 100-layer DenseNet with only 0.8M parameters.

- 모든 Layer의 feature map 연결
  -> Gradient vanishing 문제 완화
- 파라미터 수, 연산량이 적음.

02. CNN의 흐름 및 모델

2017: MobileNet – MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Application

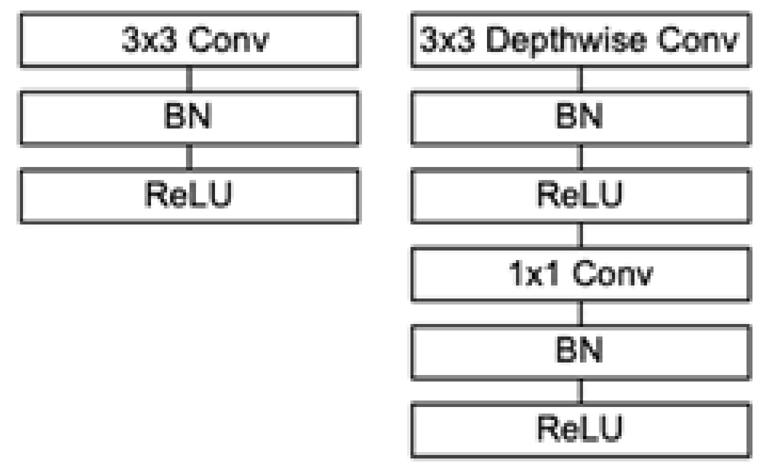


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

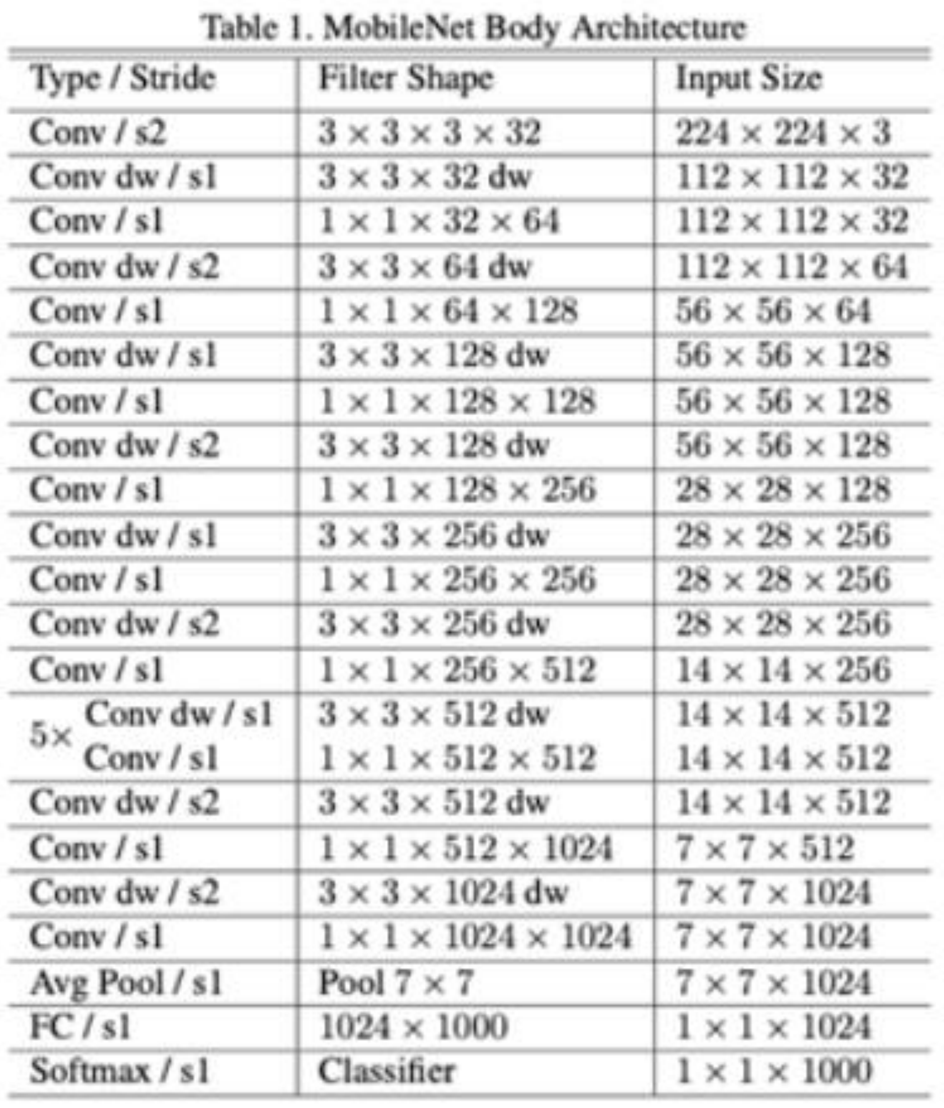


Table 1. MobileNet Body Architecture

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

2017: ShuffleNet – ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices

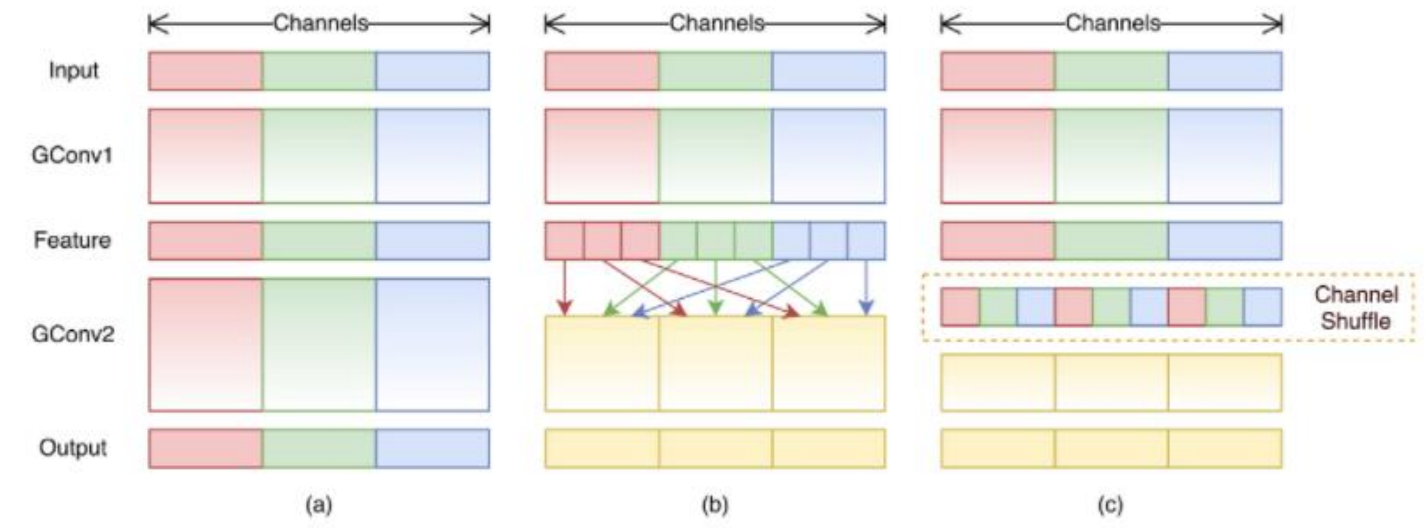


Figure 1. Channel shuffle with two stacked group convolutions. GConv stands for group convolution. a) two stacked convolution layers with the same number of groups. Each output channel only relates to the input channels within the group. No cross talk; b) input and output channels are fully related when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

02. CNN의 흐름 및 모델

2019: EfficientNet – EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

- 모델의 크기를 어떻게 키울 것인가?

-> 모델 Scale up의 원리는 무엇일까?

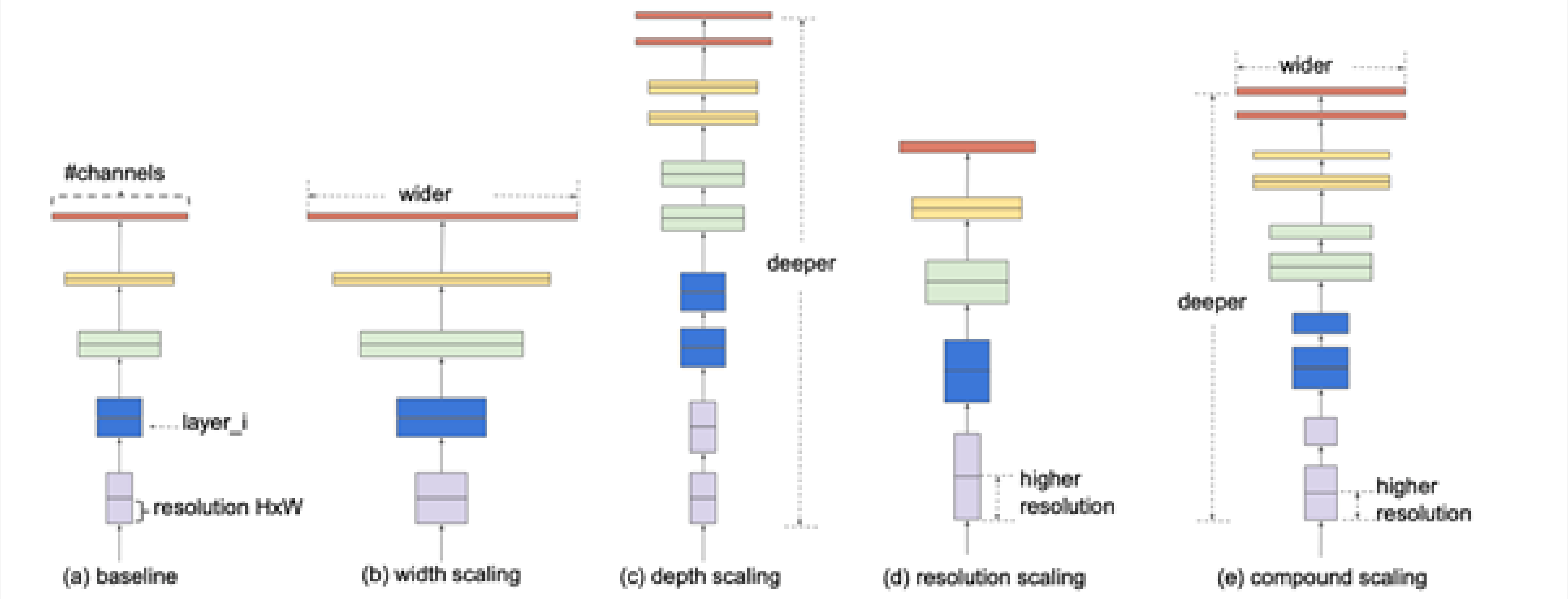


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

02. CNN의 흐름 및 모델

○ **Problem Formulation:**

$$\mathcal{N} = \bigodot_{i=1\ldots s} \mathcal{F}_i^{L_i}\left(X_{\langle H_i, W_i, C_i \rangle}\right)$$

- $\mathcal{N}$: CNN 모델
- F_i : Height, Width, Channel로 구성된 <u>**Layer**</u> (Operator)
- L_i : Stage i 에서의 반복 수(Repeated Times in stage i)

$$\begin{aligned} \text{depth: } & d = \alpha^{\phi} \\ \text{width: } & w = \beta^{\phi} \\ \text{resolution: } & r = \gamma^{\phi} \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned}$$

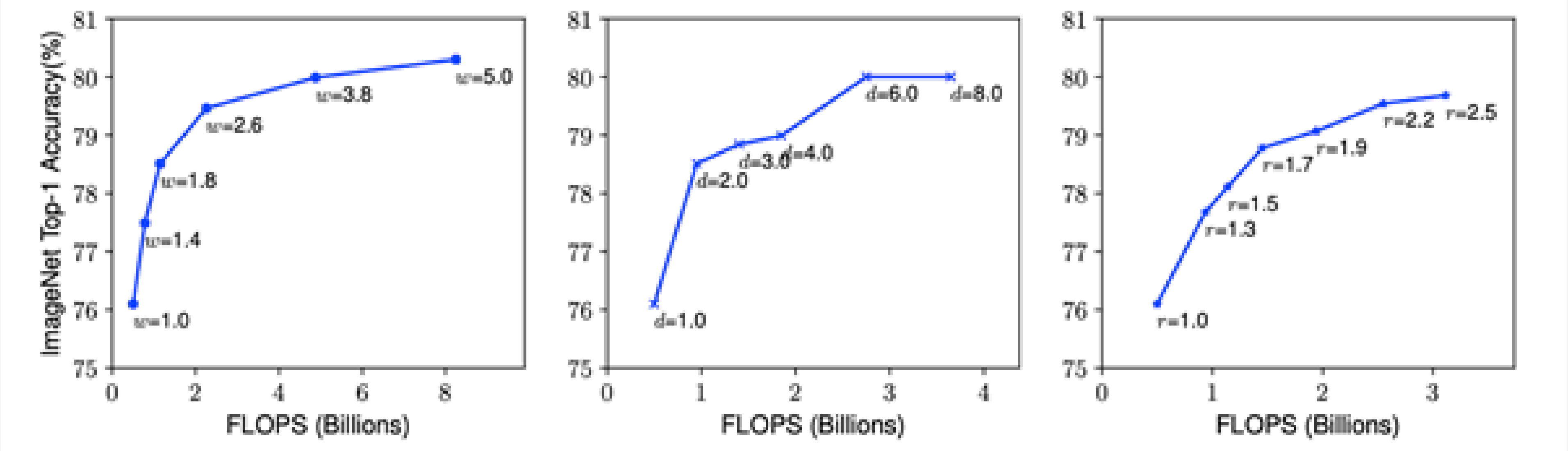


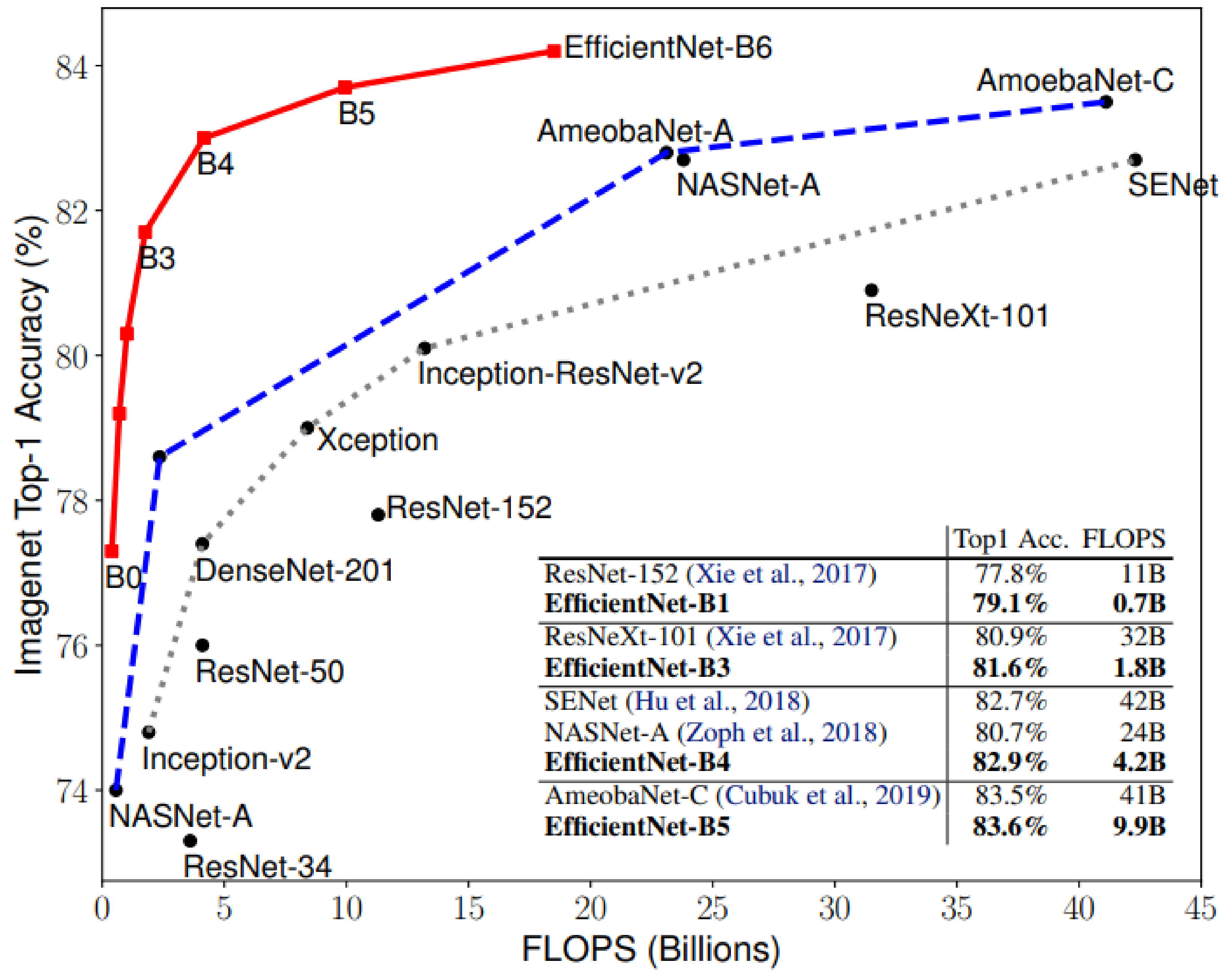
Figure 3. **Scaling Up a Baseline Model with Different Network Width (w), Depth (d), and Resolution (r) Coefficients.** Bigger networks with larger width, depth, or resolution tend to achieve higher accuracy, but the accuracy gain quickly saturate after reaching 80%, demonstrating the limitation of single dimension scaling. Baseline network is described in Table 1.

- **Width** :
  * 특징 : 1. 작은 모델의 사이즈를 키울 때 많이 사용됨
    2. 세밀(Fine-Grained)한 Feature를 Capture 하기 위해 많이 사용됨
  * 단점 : 1. Shallow 모델(= Layer가 낮은)에서, 상위 레벨의 복합적인 Feature를 파악하기 어려움
    2. Width가 넓어짐에 따라 빠르게 포화(Saturate)됨

- **Depth** :
  * 장점 : 1. Capture Richer and more Complex Feature
  * 단점 : 1. Skip-Connection이나 Batch Norm으로 완화하고 있지만, Vanishing Gradient로 학습하기 어려움
    2. 특정 층 이상으로 쌓이면, 성능이 저하됨

- **Resolution** :
  * 장점 : 1. 세밀(Fine-Grained)한 Feature를 Capture 하기 위해 많이 사용됨
    2. Detection처럼 복잡한 Task에서 더 높은 Resolution을 사용
    3. Resolution의 증가는 정확도에 영향을 줌
  * 단점 : 1. Resolution의 증가할수록, 성능 증가폭은 감소됨
    (r = 1.0 - 224x224 / r = 2.5 - 560x560)

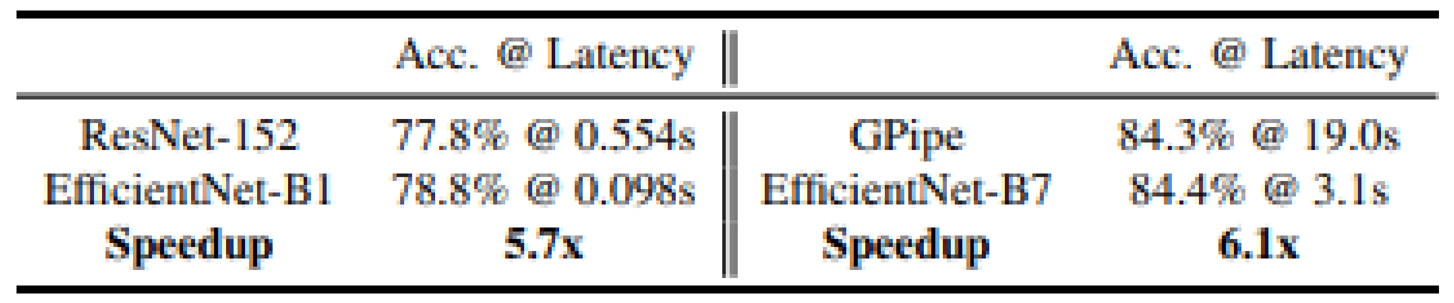
02. CNN의 흐름 및 모델



	Top1 Acc.	FLOPS
ResNet-152 (Xie et al., 2017)	77.8%	11B
EfficientNet-B1	**79.1%**	**0.7B**
ResNeXt-101 (Xie et al., 2017)	80.9%	32B
EfficientNet-B3	**81.6%**	**1.8B**
SENet (Hu et al., 2018)	82.7%	42B
NASNet-A (Zoph et al., 2018)	80.7%	24B
EfficientNet-B4	**82.9%**	**4.2B**
AmeobaNet-C (Cubuk et al., 2019)	83.5%	41B
EfficientNet-B5	**83.6%**	**9.9B**

Figure 5. **FLOPS vs. ImageNet Accuracy –** Similar to Figure 1 except it compares FLOPS rather than model size.

Table 4. **Inference Latency Comparison –** Latency is measured with batch size 1 on a single core of Intel Xeon CPU E5-2690.



	Acc. @ Latency		Acc. @ Latency
ResNet-152	77.8% @ 0.554s	GPipe	84.3% @ 19.0s
EfficientNet-B1	78.8% @ 0.098s	EfficientNet-B7	84.4% @ 3.1s
Speedup	**5.7x**	**Speedup**	**6.1x**

02. CNN의 흐름 및 모델

https://paperswithcode.com/sota/image-classification-on-imagenet

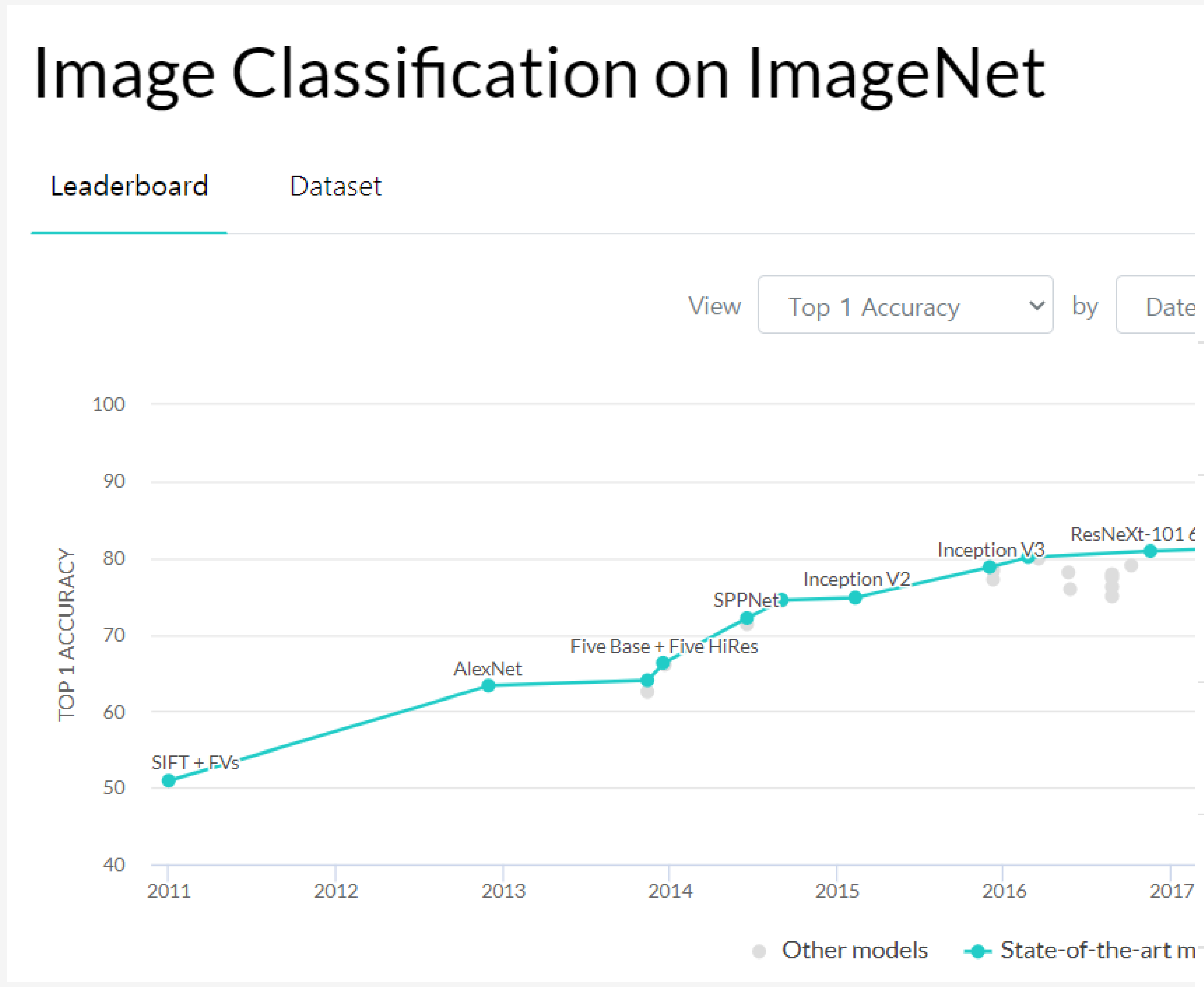
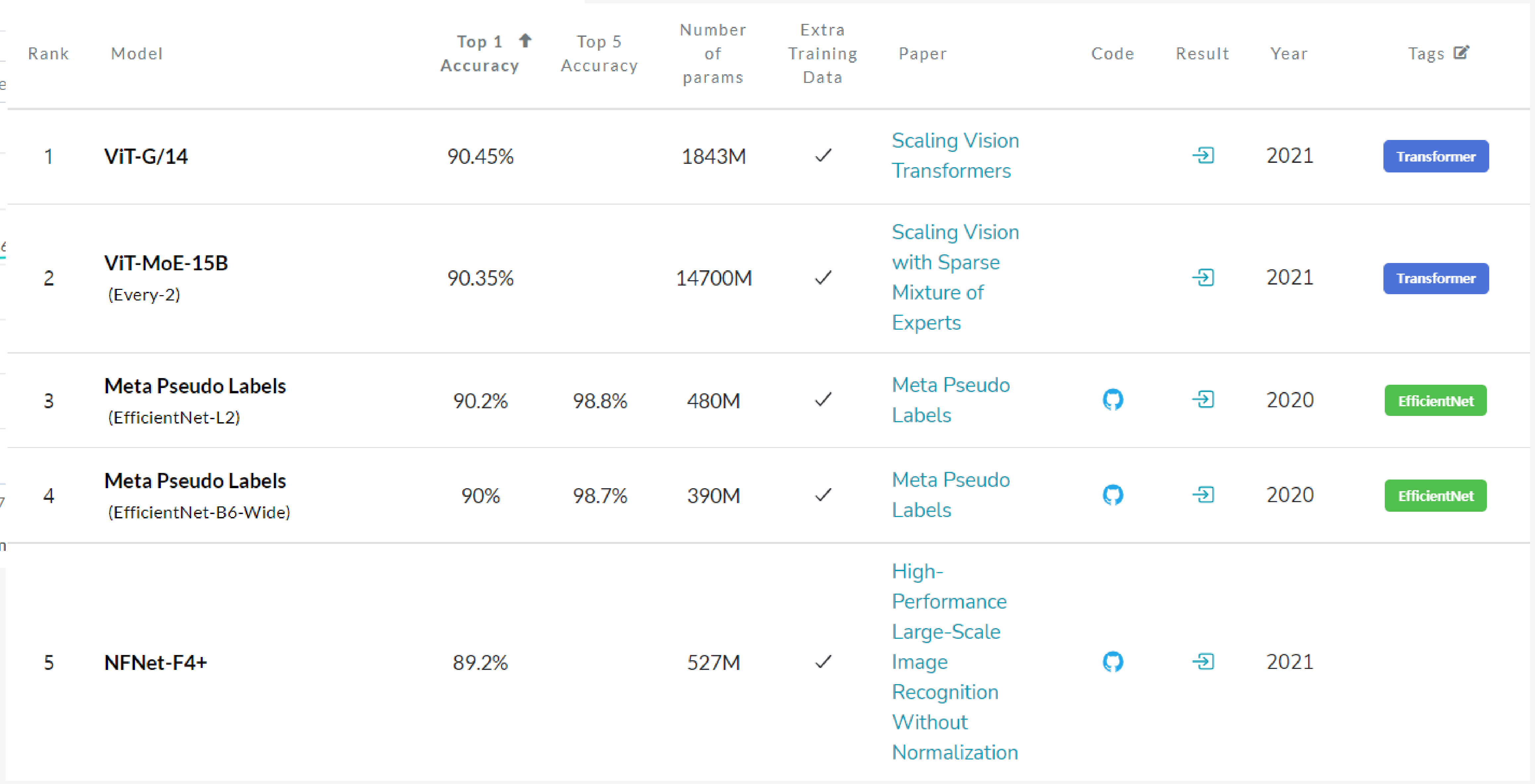


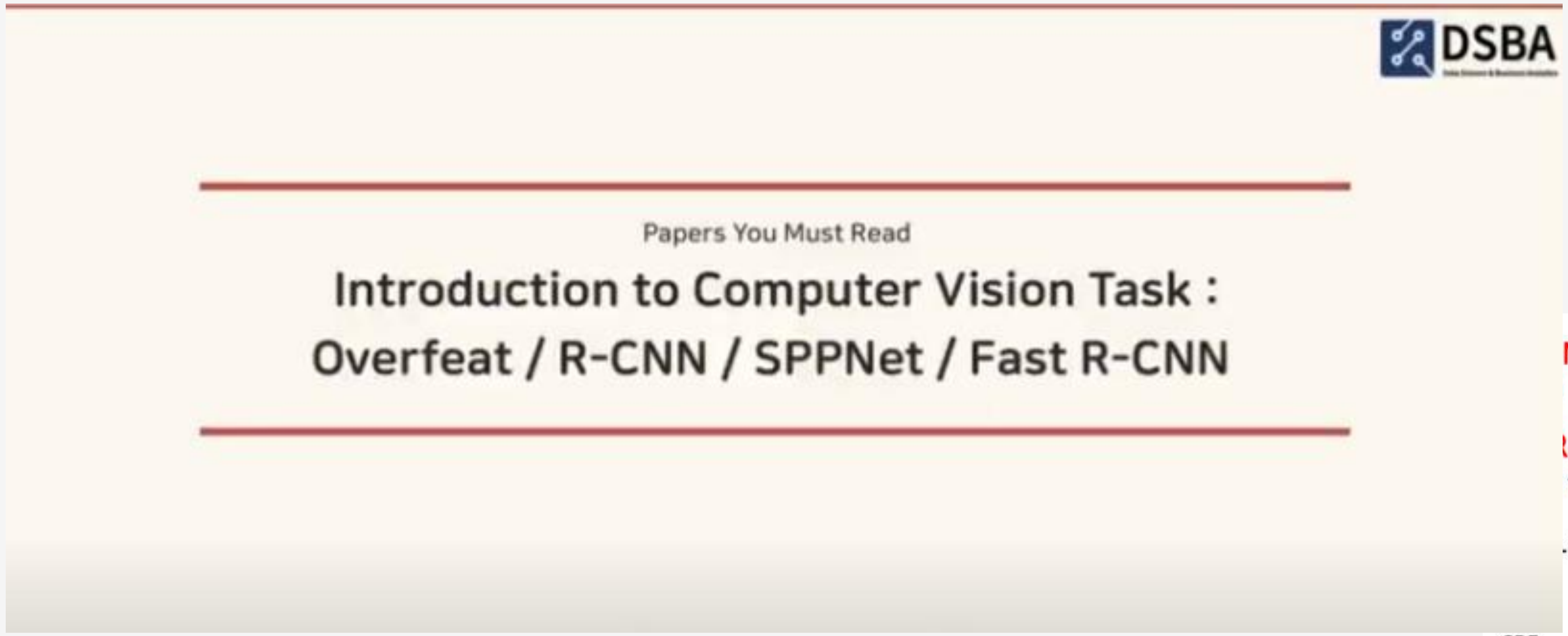
Image Classification on ImageNet

Leaderboard Dataset



Rank	Model	Top 1 Accuracy	Top 5 Accuracy	Number of params	Extra Training Data	Paper	Code	Result	Year	Tags
1	**ViT-G/14**	90.45%		1843M	✓	Scaling Vision Transformers		⇥	2021	Transformer
2	**ViT-MoE-15B** (Every-2)	90.35%		14700M	✓	Scaling Vision with Sparse Mixture of Experts		⇥	2021	Transformer
3	**Meta Pseudo Labels** (EfficientNet-L2)	90.2%	98.8%	480M	✓	Meta Pseudo Labels	●	⇥	2020	EfficientNet
4	**Meta Pseudo Labels** (EfficientNet-B6-Wide)	90%	98.7%	390M	✓	Meta Pseudo Labels	●	⇥	2020	EfficientNet
5	**NFNet-F4+**	89.2%		527M	✓	High-Performance Large-Scale Image Recognition Without Normalization	●	⇥	2021	

02. CNN의 흐름 및 모델

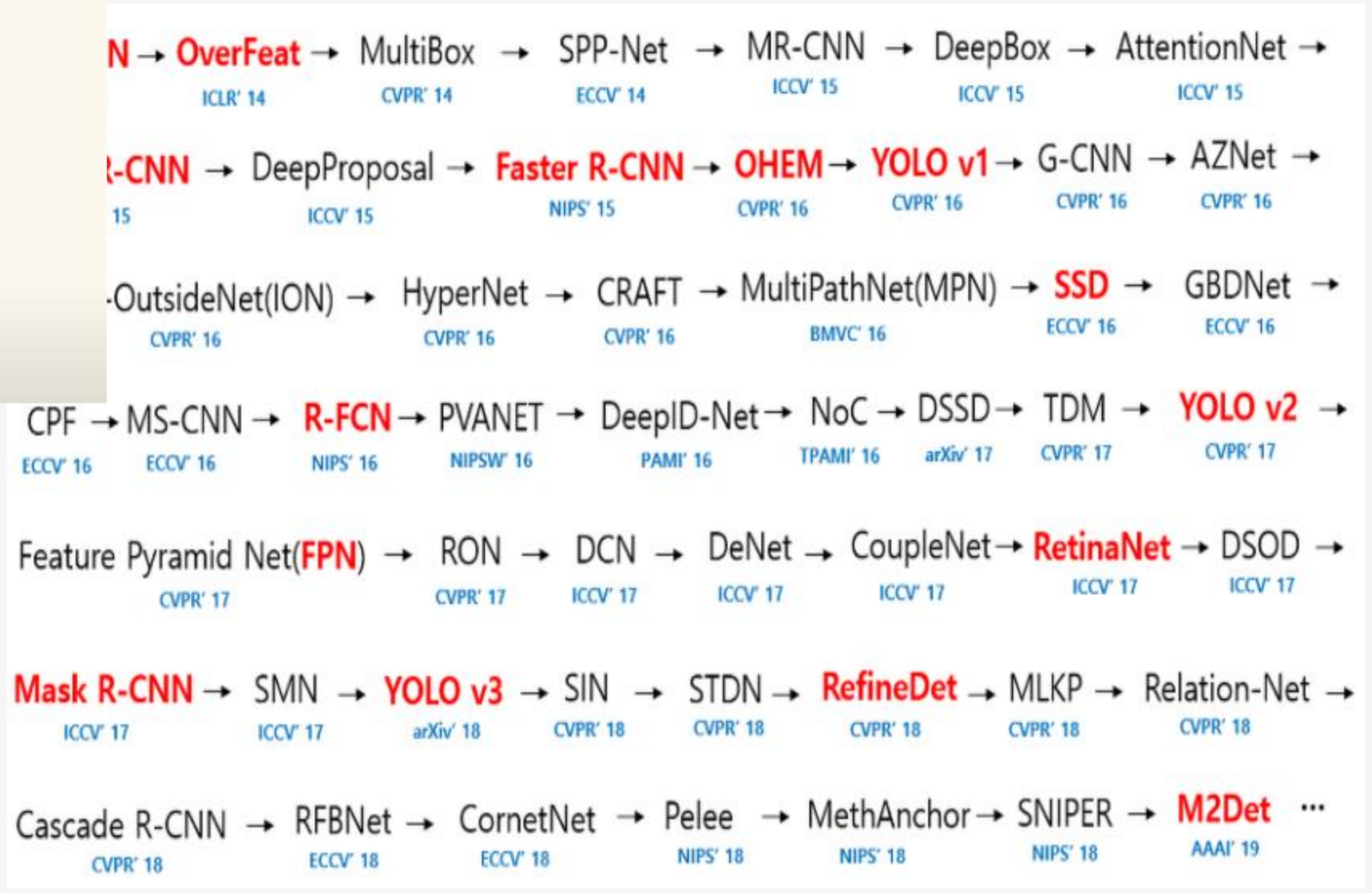


DSBA

Papers You Must Read

Introduction to Computer Vision Task :
Overfeat / R-CNN / SPPNet / Fast R-CNN

[Paper Review] Introduction to Object Detection Task : Overfeat, RCNN, SPPNet, FastRCNN - YouTube


N → OverFeat → MultiBox → SPP-Net → MR-CNN → DeepBox → AttentionNet →
ICLR' 14 CVPR' 14 ECCV' 14 ICCV' 15 ICCV' 15 ICCV' 15
-CNN → DeepProposal → Faster R-CNN → OHEM → YOLO v1 → G-CNN → AZNet →
15 ICCV' 15 NIPS' 15 CVPR' 16 CVPR' 16 CVPR' 16 CVPR' 16
-OutsideNet(ION) → HyperNet → CRAFT → MultiPathNet(MPN) → SSD → GBDNet →
CVPR' 16 CVPR' 16 CVPR' 16 BMVC' 16 ECCV' 16 ECCV' 16
CPF → MS-CNN → R-FCN → PVANET → DeepID-Net → NoC → DSSD → TDM → YOLO v2 →
ECCV' 16 ECCV' 16 NIPS' 16 NIPSW' 16 PAMI' 16 TPAMI' 16 arXiv' 17 CVPR' 17 CVPR' 17
Feature Pyramid Net(FPN) → RON → DCN → DeNet → CoupleNet → RetinaNet → DSOD →
CVPR' 17 CVPR' 17 ICCV' 17 ICCV' 17 ICCV' 17 ICCV' 17 ICCV' 17
Mask R-CNN → SMN → YOLO v3 → SIN → STDN → RefineDet → MLKP → Relation-Net →
ICCV' 17 ICCV' 17 arXiv' 18 CVPR' 18 CVPR' 18 CVPR' 18 CVPR' 18 CVPR' 18
Cascade R-CNN → RFBNet → CornetNet → Pelee → MethAnchor → SNIPER → M2Det ···
CVPR' 18 ECCV' 18 ECCV' 18 NIPS' 18 NIPS' 18 NIPS' 18 AAAI' 19

THANK YOU

감사합니다.