DeepIntoDeep

2. Convolutional Neural Networks

발표자: 김채현

2. Convolutional Neural Networks

🙌 This lecture is revised based on Dongwhan Chi's 23S D2D Lecture 2.

Chaehyun Kim

Artificial Intelligence in Korea University(AIKU)

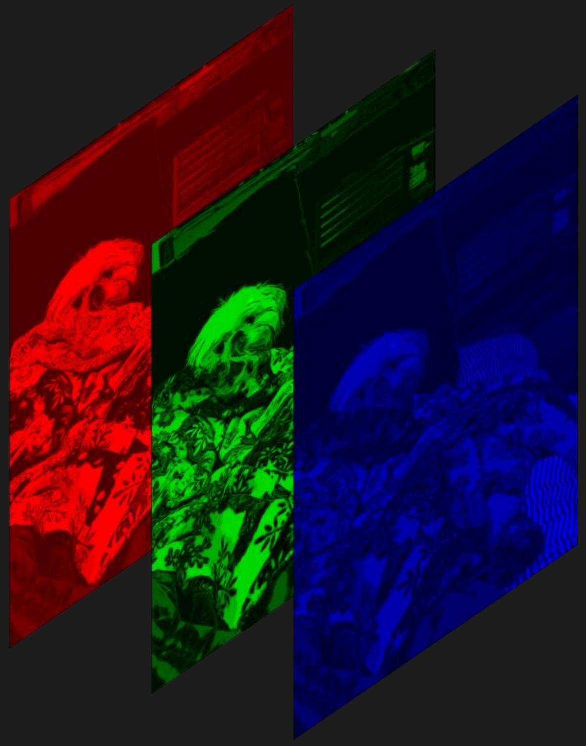
Department of Computer Science and Engineering, Korea University

Contents

Why do we need convolution?

An image is a collection of numbers!

- 이미지는 고차원 데이터, 사실은 3차원 데이터다



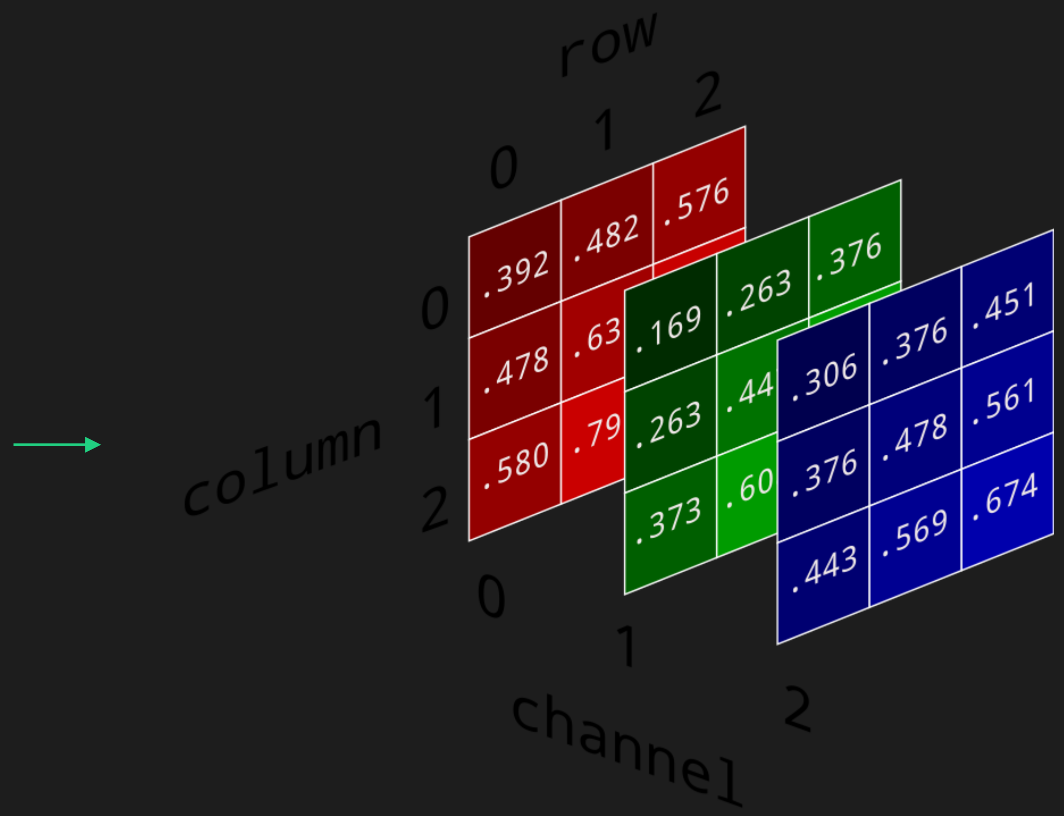
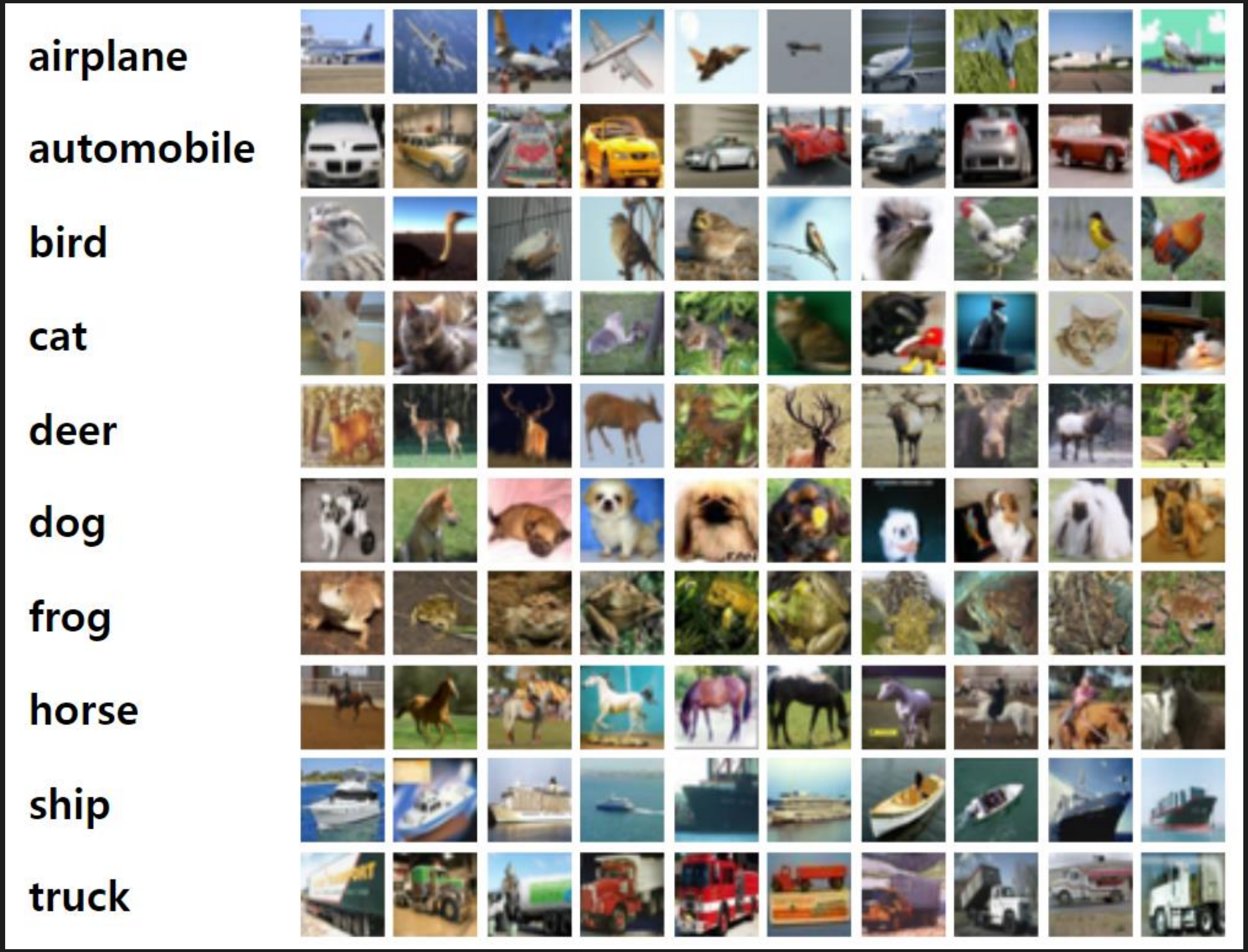


Image Credit: How to Convert an RGB Image to Grayscale

Image Classification

[Getting started with Image Recognition and Convolutional Neural Networks in 5 minutes]()

The CIFAR-10 Dataset [desc]



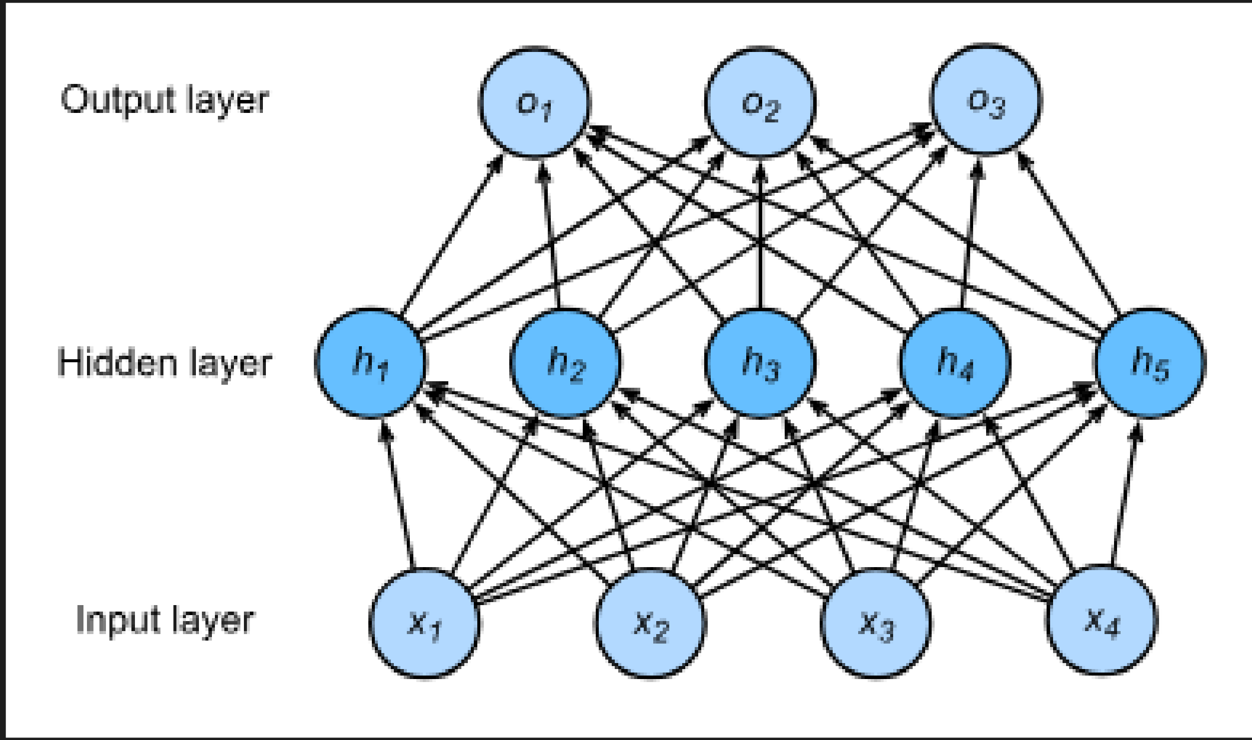
- Canadian Institute For Advanced Research : ML, CV를 위해 일반적으로 사용되는 image의 모음

- 32x32x3 이미지
- 10개의 클래스, 각 6000장

… 어떻게 해결할까?

MLP는 어떨까?

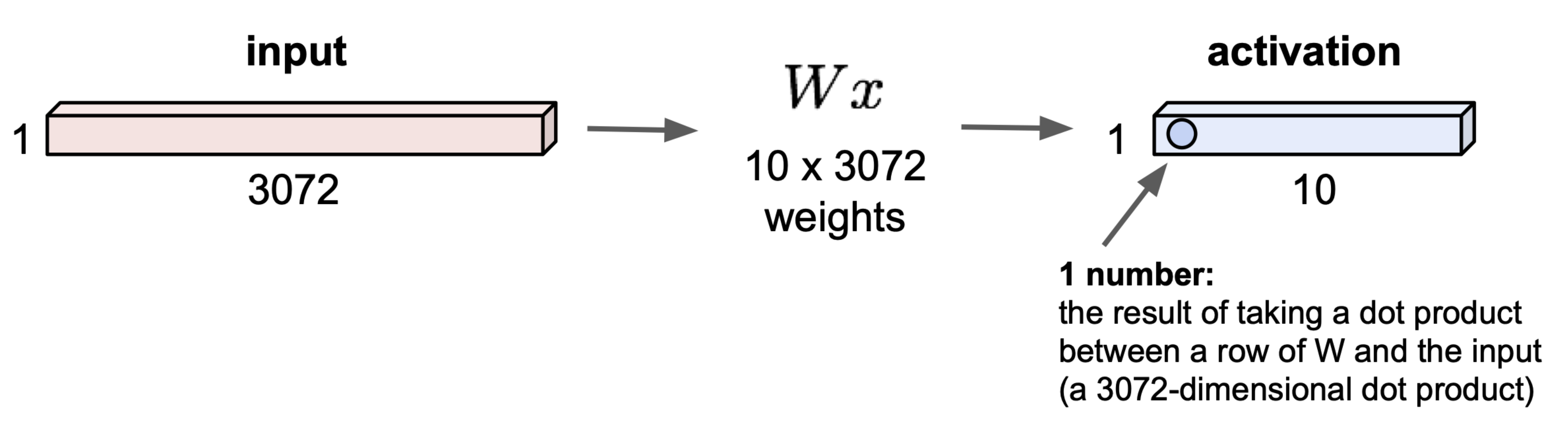
- Non-linearity
- More complex prediction



출처: [Dive into Deep Learning]

Image Classification with MLP

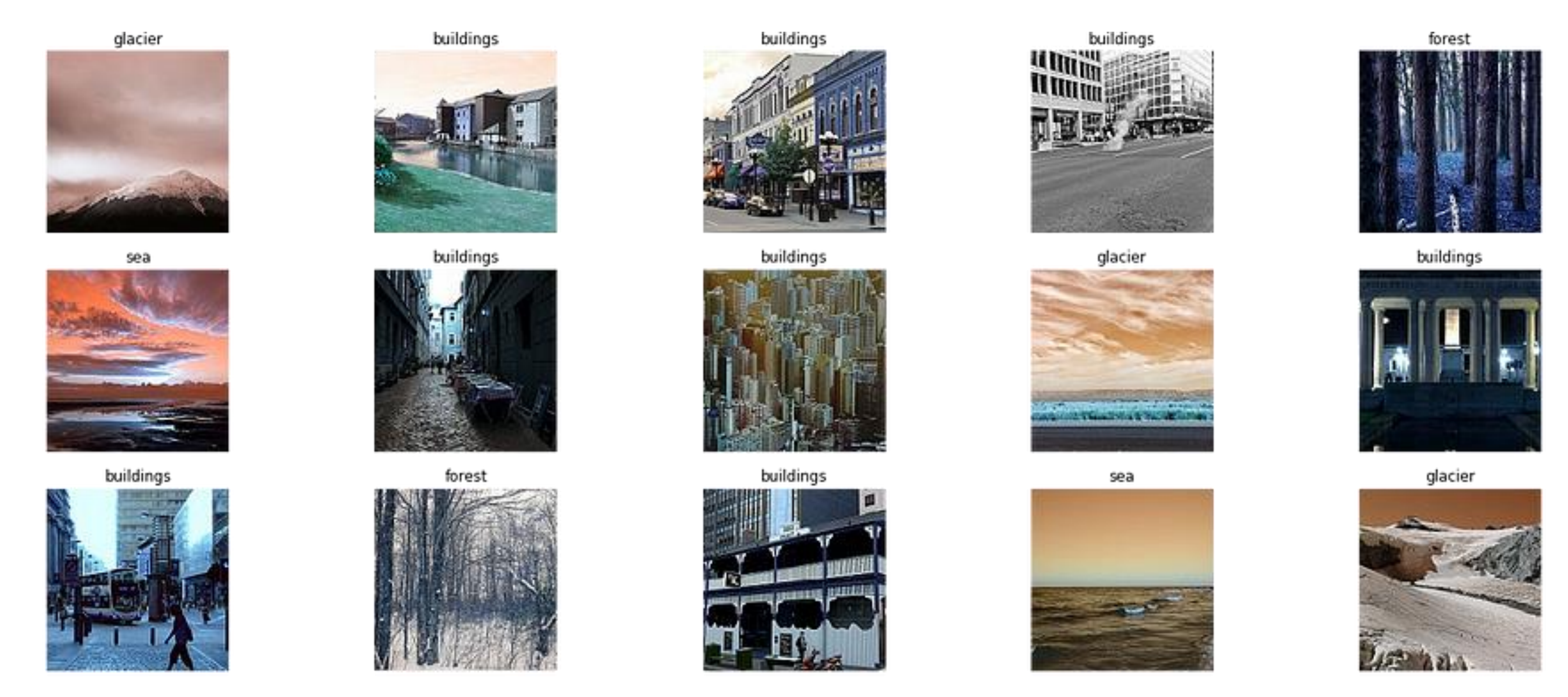
- MLP의 Parameters 수
  - CIFAR-10 image size: 32x32x3 = 3072
  - # of output class : 10
  - ➔ 3072 x 10 weights … 할만한데..?



출처 : CS231n (2017) Lecture 5

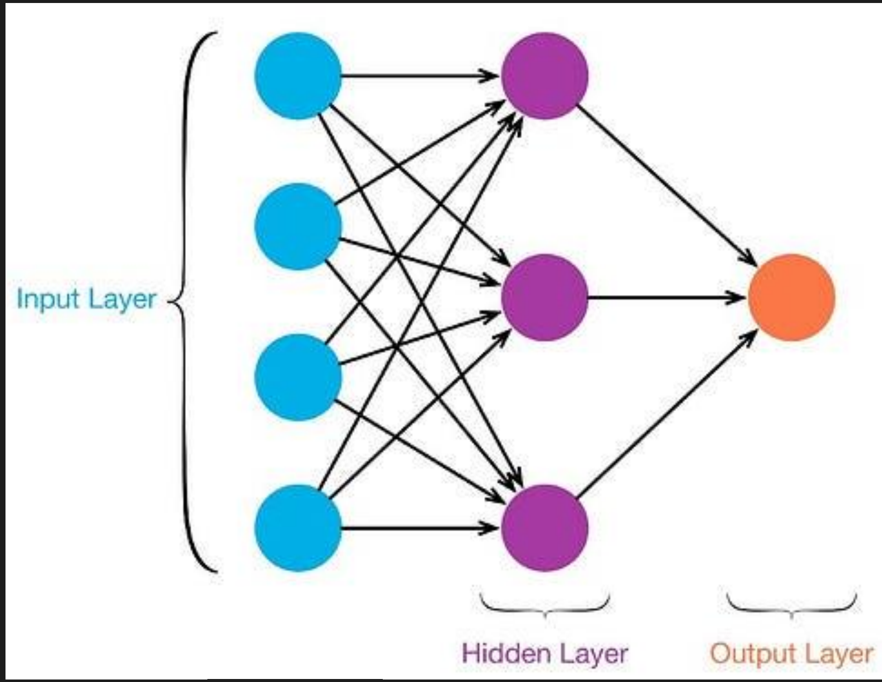
Why not 'MLP'?

- Intel Image Classification Dataset [desc]
  - Image resolution: 150 x 150 x 3



Why not ‘MLP’?

- Input image resolution: 150 x 150 x 3
- Too much parameters! → overfitting



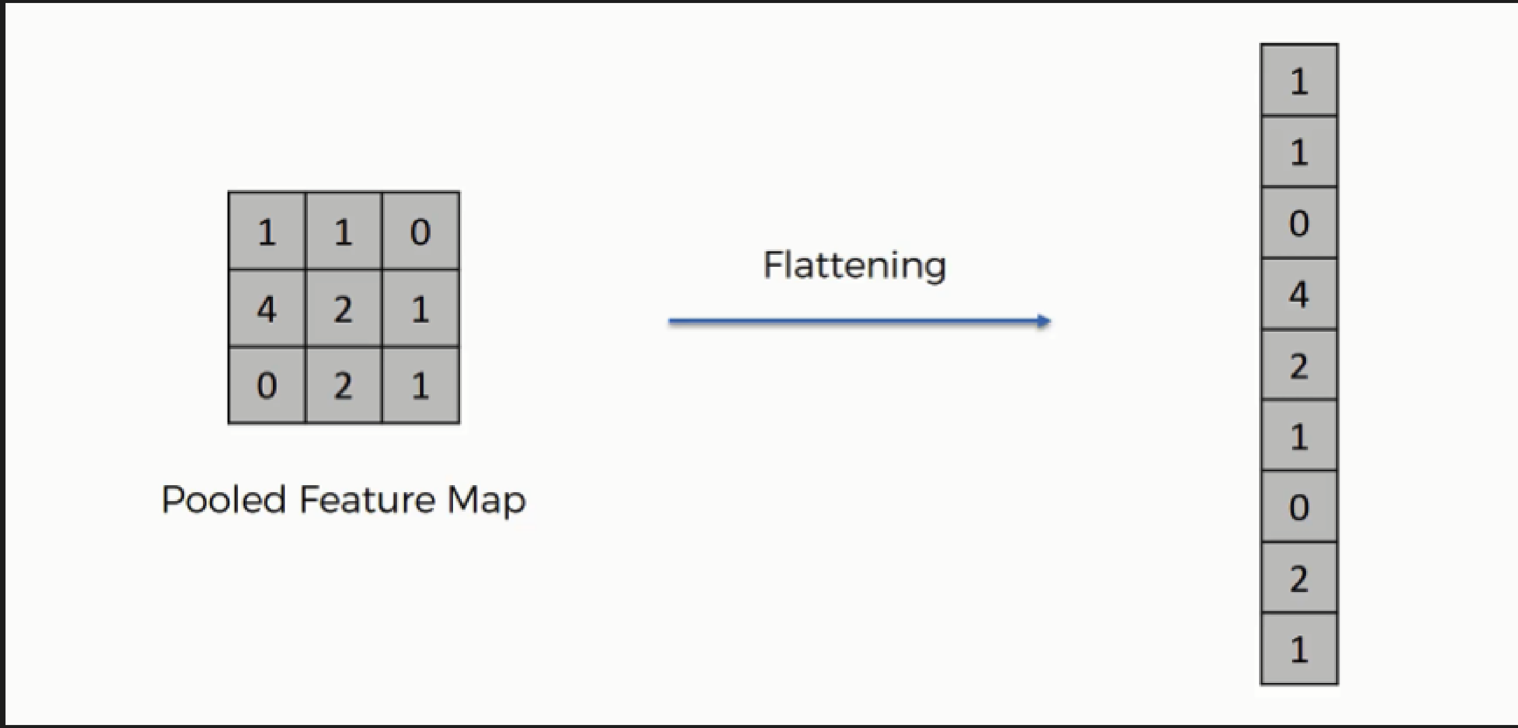
? 128 6

```
from torch import nn

# image shape = [150, 150, 3]

model = nn.Sequential(
  nn.flatten(),
  nn.linear( ?  128, bias=True)
  nn.ReLU(),
  nn.linear(128, 6, bias=True)
  nn.Softmax()
)
```

출처: [Why You Should Use Convolutions in Your Next Neural Net — Using TensorFlow]

Why not 'MLP'?

- 이미지를 'flatten' 하는 과정에서 구조적인 정보 파괴



출처 : https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening

그래서…

공간적인 정보를 유지하고

불필요하게 많은 Weight을 줄이는

Convolution이 탄생했다

Architecture Overview

What is ‘Convolution’?

- Convolution : 합성곱 (🤔?)
  - 하나의 함수와 또 다른 함수를 반전 이동한 값을 곱한 다음, 구간에 대해 적분하여 새로운 함수를 구하는 수학 연산자 - Wikipedia (😩…?)

- Convolutional Neural Network (CNN)
  - 동의어 : Shift Invariant or Space Invariant Artificial Neural Networks
  - Regularization을 위한 접근
  - 같은 Parameter (filter)를 공유하며 사용 + 계층적 (구조적) 패턴
    - Overfitting을 막아 보자!

Convolutional Neural Networks

- 각 Patch에서 동일한 일을 한다.

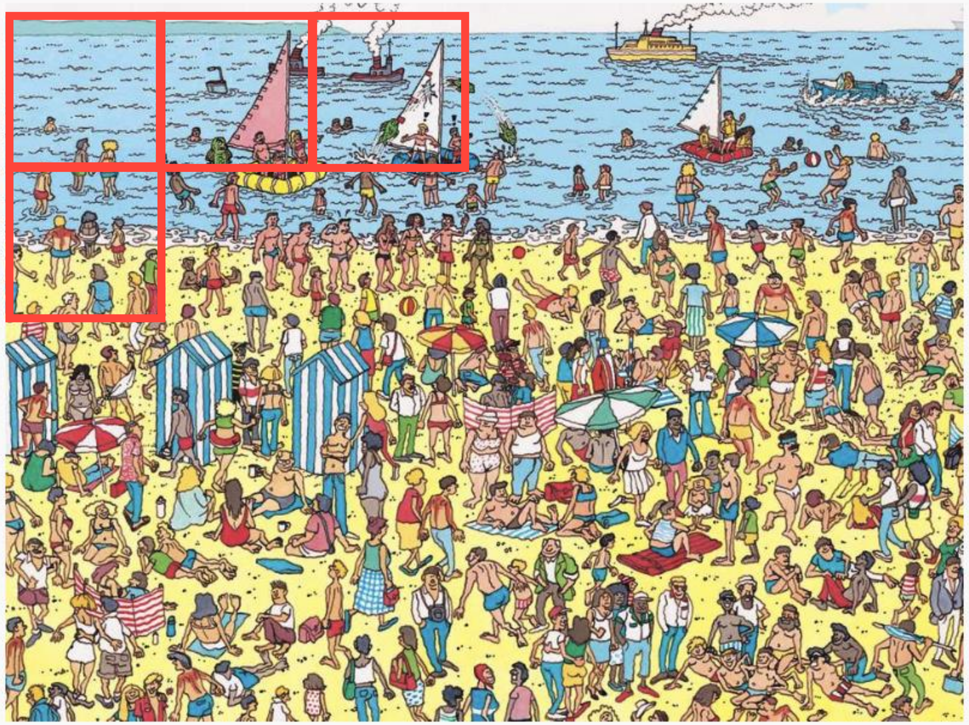


Fig. 7.1.1 An image of the “Where’s Waldo” game.

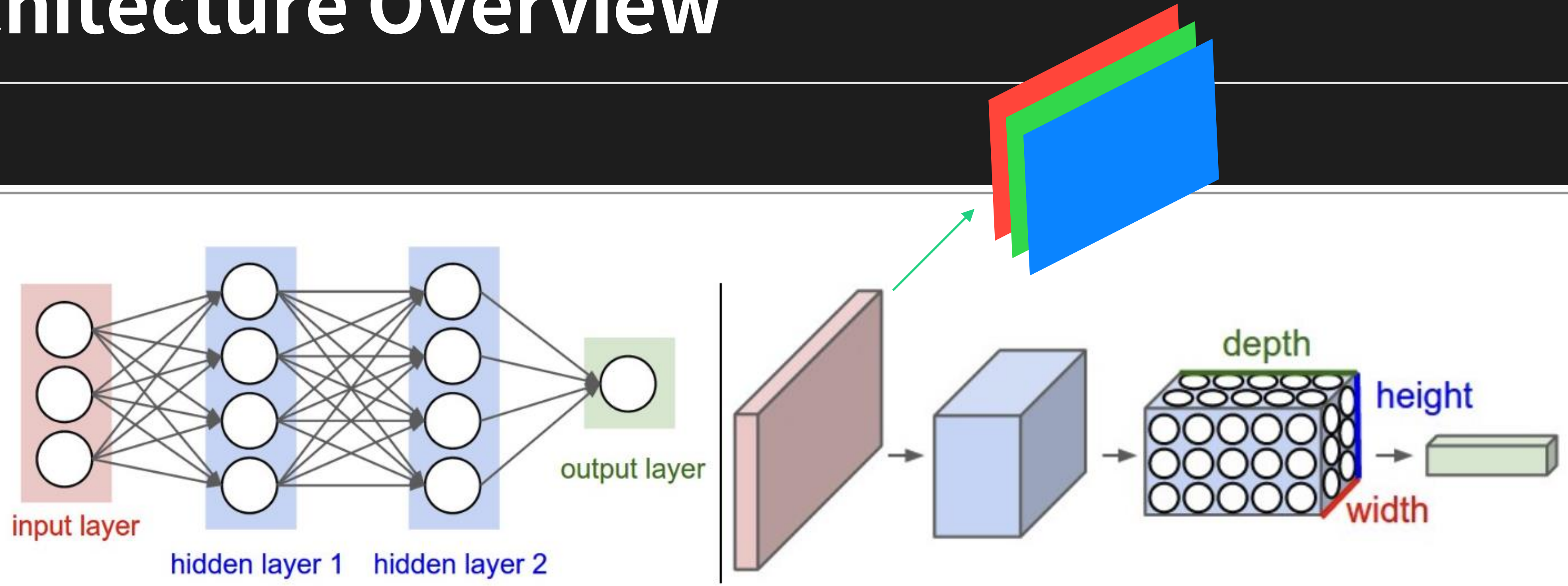
출처 : [Dive Into Deep Learning]

Architecture Overview

- 지금까지 배운 MLP
  - hidden layers!
- MLP의 Parameters 수
  - CIFAR-10 : 32x32x3 * 10 = 3072 * 10 weights, FCN도 가능
  - 더 큰 image라면?

- Convolutional Layer는 3D neurons을 가진다.
  - Figure에서 확인해 볼 것!

Architecture Overview



Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

출처 : https://cs231n.github.io/convolutional-networks/

Convolutional Layers

Convolutional Layer

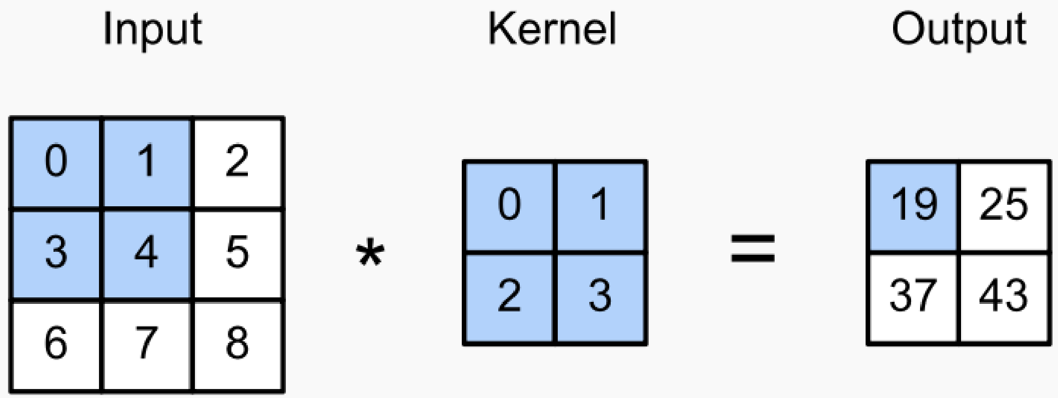
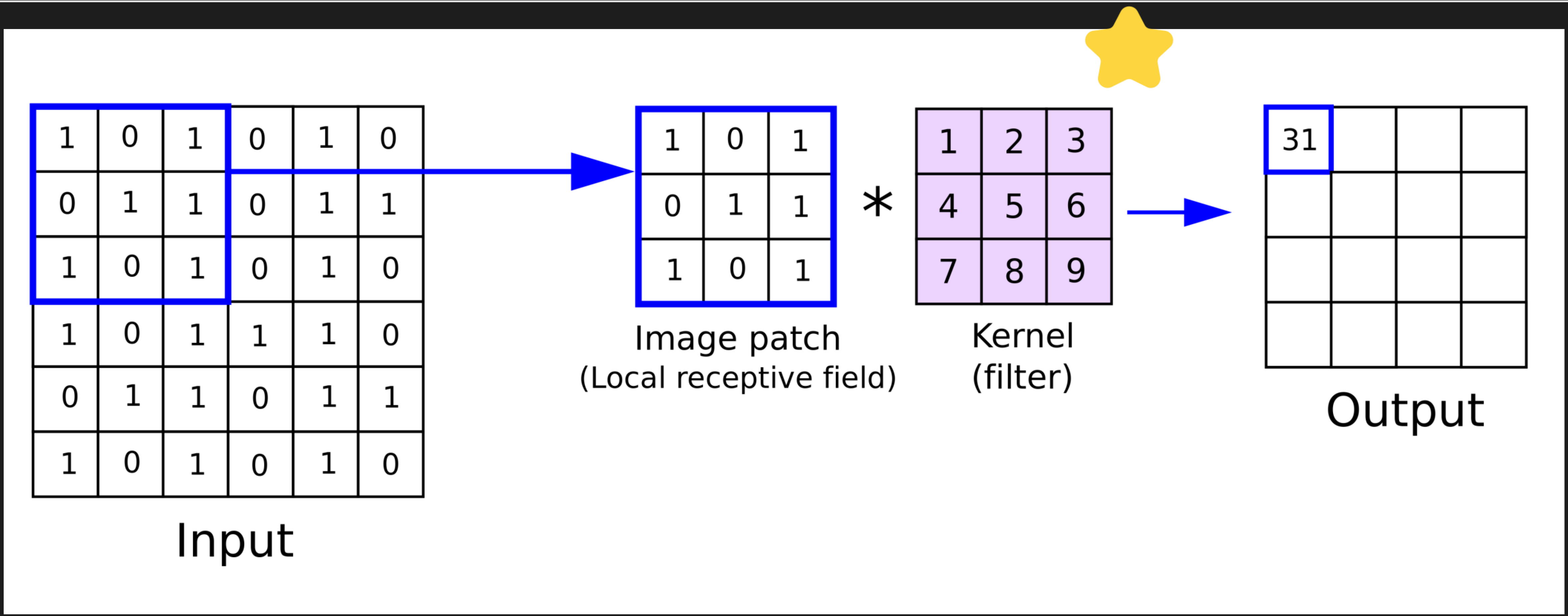


Fig. 7.2.1 Two-dimensional cross-correlation operation. The shaded portions are the first output element as well as the input and kernel tensor elements used for the output computation: $0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$.

출처 : [Dive Into Deep Learning]

Convolutional Layer



출처 : https://anhreynolds.com/blogs/cnn.html

Reference: Sobel Edge Detector

- filter의 역할은 무엇일까?
- 기존 이미지 처리 방법론에서도 사용
- 이미지에서 ‘feature’를 잡아내는 역할!
    - Edge
    - Shape
    - …
- 예: Sobel filter

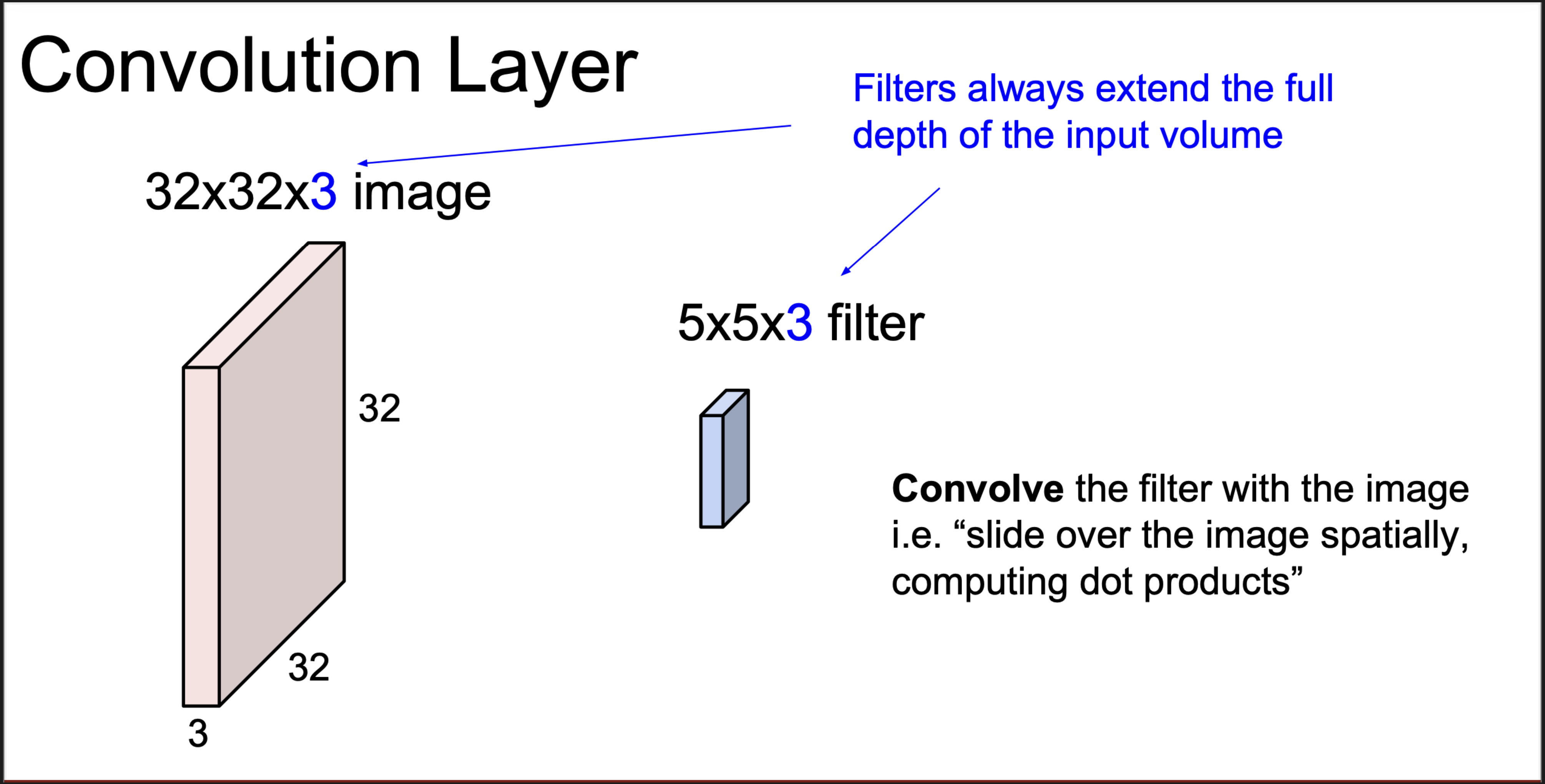
-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

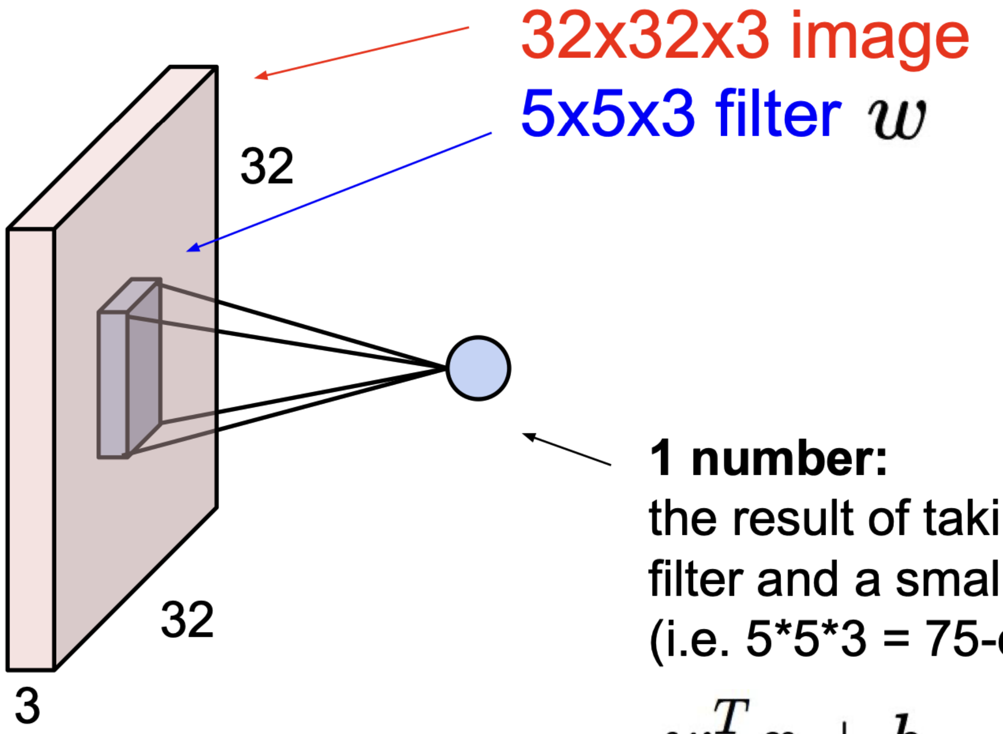
Convolutional Layer



출처 : CS231n (2017) Lecture 5

Convolutional Layer

Convolution Layer

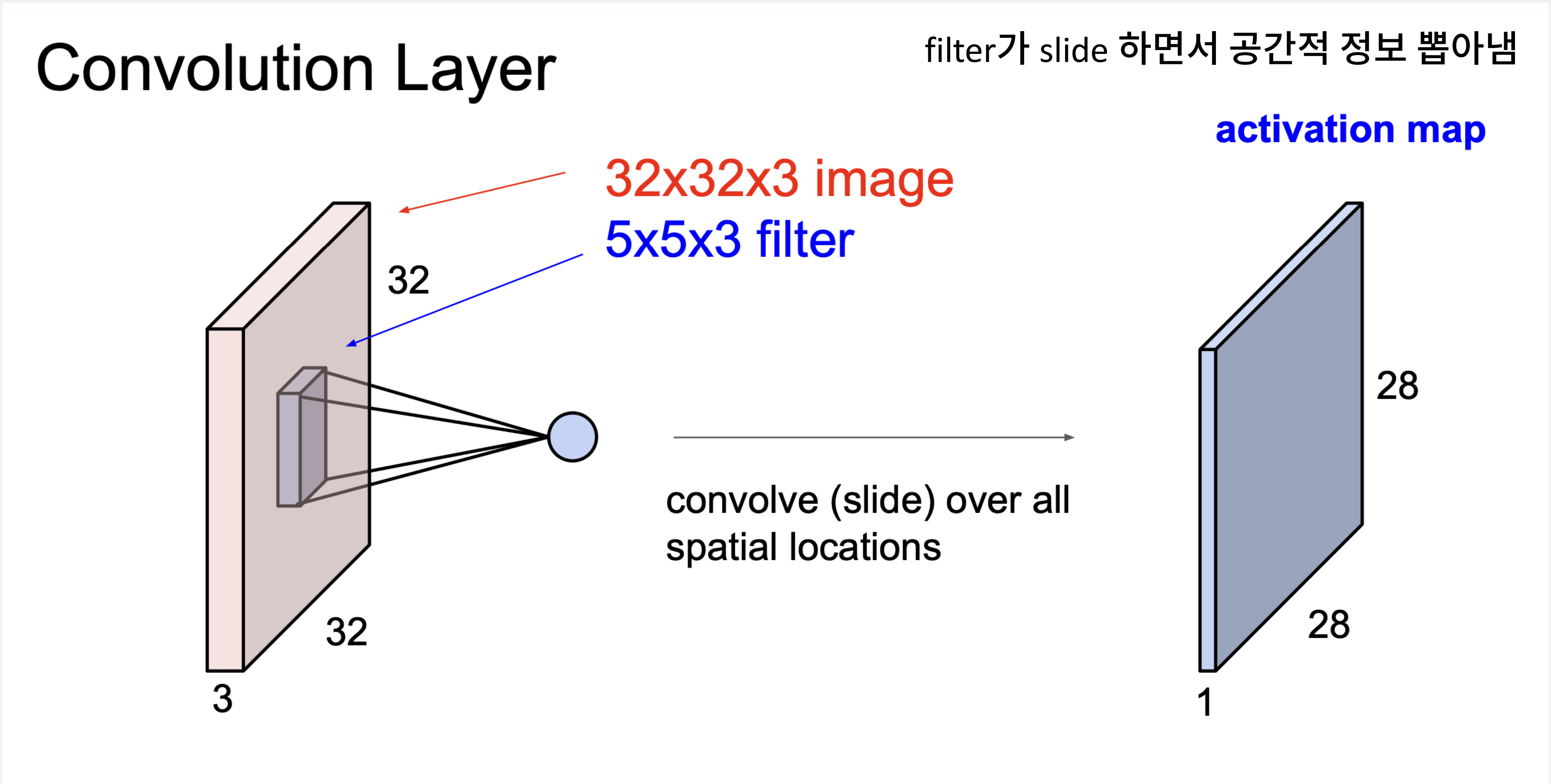


1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

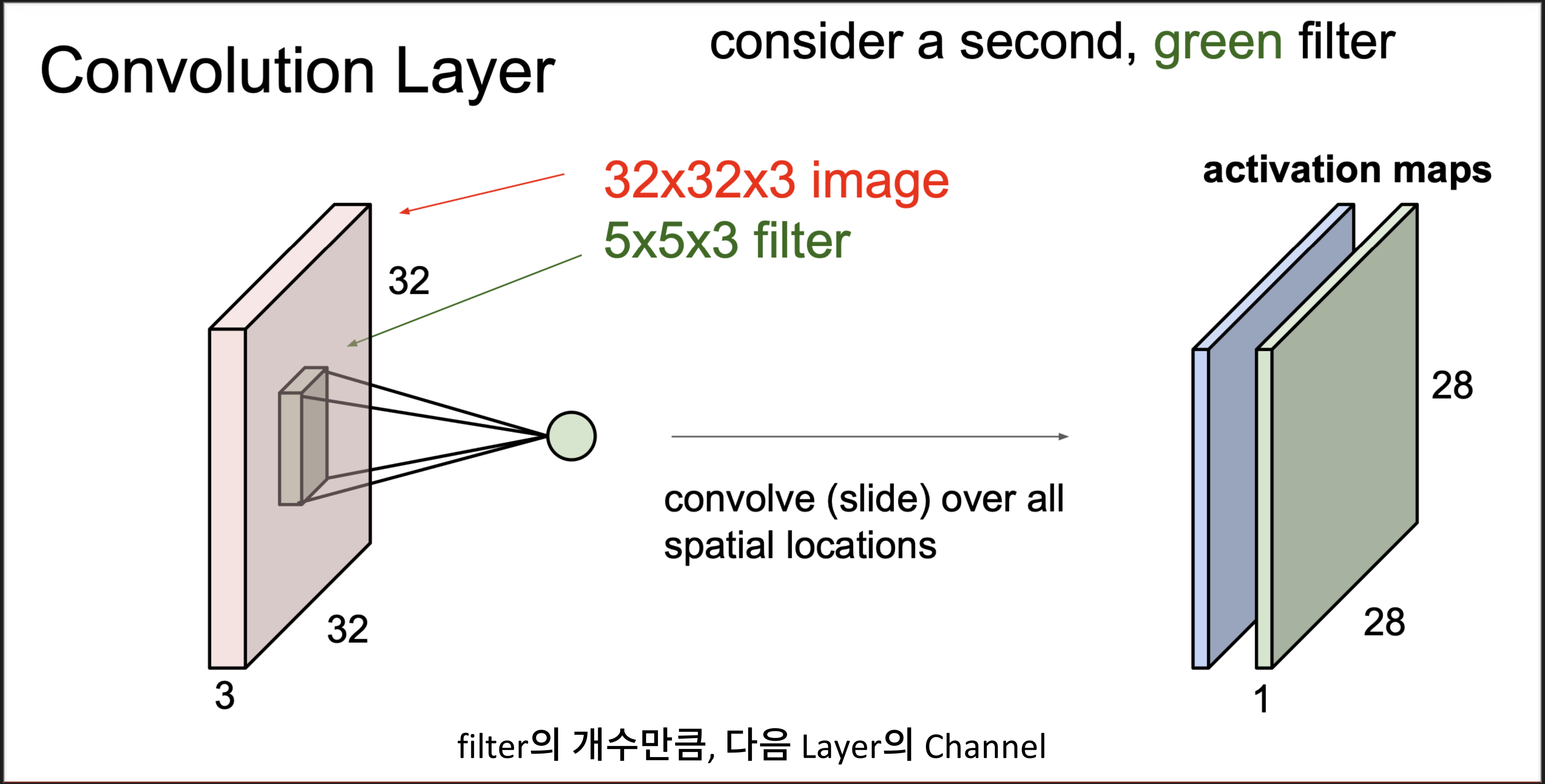
출처 : CS231n (2017) Lecture 5

Convolutional Layer



출처 : CS231n (2017) Lecture 5

Convolutional Layer



출처 : CS231n (2017) Lecture 5

Convolutional Layer

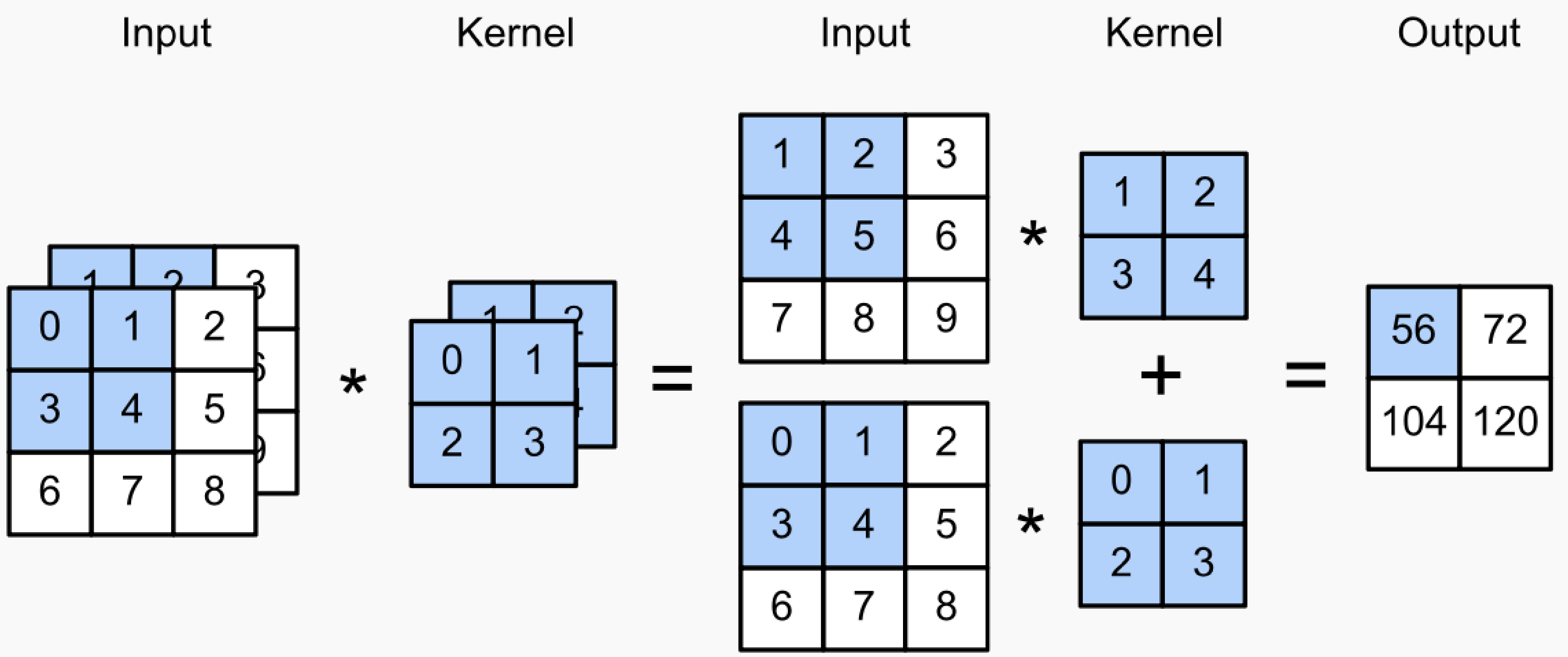
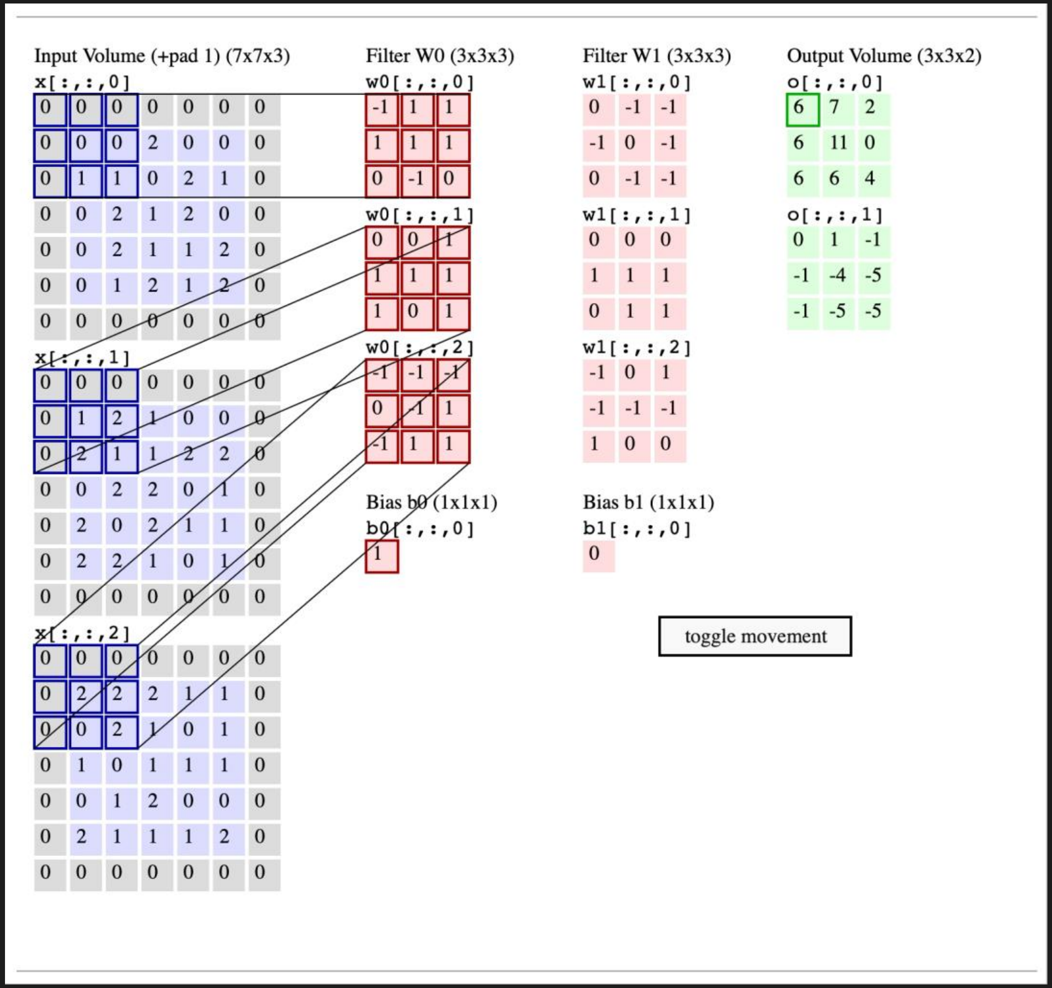


Fig. 7.4.1 Cross-correlation computation with 2 input channels.

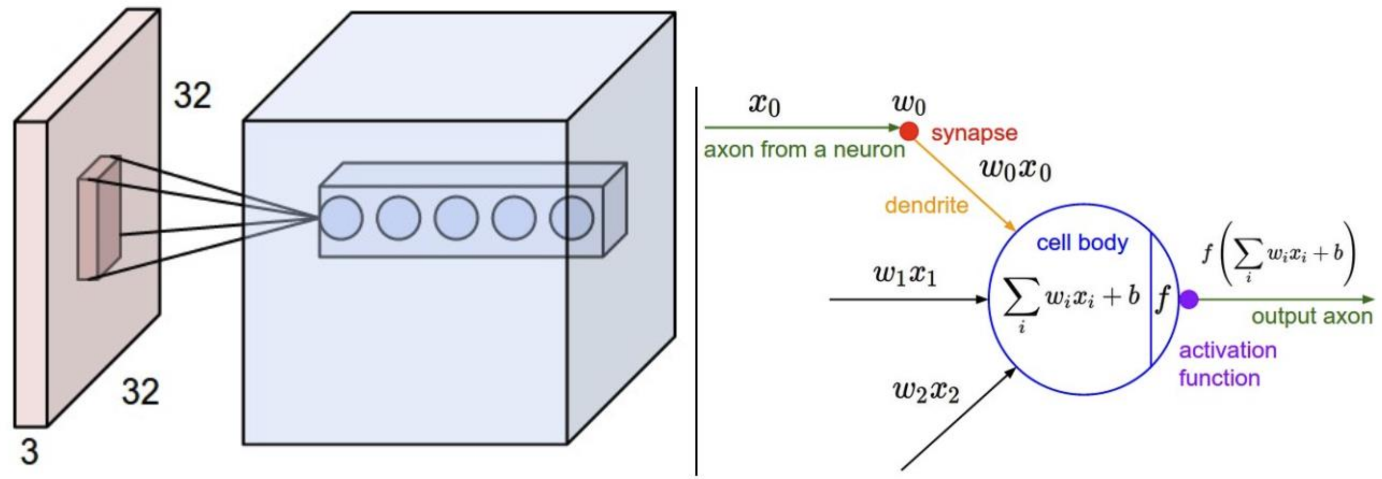
출처 : Dive Into Deep Learning

Convolutional Layer



출처 : https://cs231n.github.io/convolutional-networks/#conv

Convolutional Layer



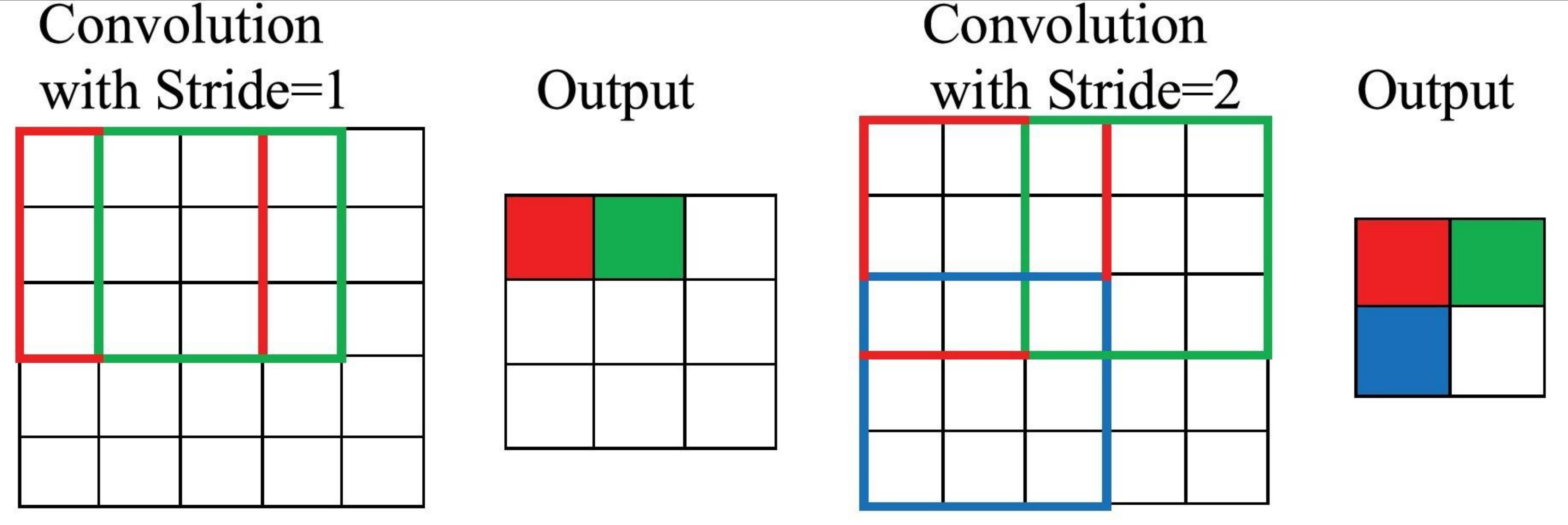
Left: An example input volume in red (e.g. a 32x32x3 CIFAR-10 image), and an example volume of neurons in the first Convolutional layer. Each neuron in the convolutional layer is connected only to a local region in the input volume spatially, but to the full depth (i.e. all color channels). Note, there are multiple neurons (5 in this example) along the depth, all looking at the same region in the input: the lines that connect this column of 5 neurons do not represent the weights (i.e. these 5 neurons do not share the same weights, but they are associated with 5 different filters), they just indicate that these neurons are connected to or looking at the same receptive field or region of the input volume, i.e. they share the same receptive field but not the same weights. **Right:** The neurons from the Neural Network chapter remain unchanged: They still compute a dot product of their weights with the input followed by a non-linearity, but their connectivity is now restricted to be local spatially.

출처 : https://cs231n.github.io/convolutional-networks/#conv

Convolutional Layer - Spatial arrangement

- Convolutional Layer는 input volume과 output volume에서 맞춰줘야 하는 값이 있다.
  - **depth** of the output volume : hyperparameter, 얼마나 많은 filter 쓰는지.
  - **stride** : filter를 얼마 만큼 slide 할 것인가? (1 또는 2 많이 씀)
  - **zero-padding** : 테두리에 0으로 채워 넣어 줌. output volume 조절에 유용.
- input volume size (W), receptive field size (F), stride (S), zero padding (P)
  - Output volume : $O = \frac{W-F+2P}{S} + 1$
  - 직사각형이면? $(OH, OW) = (\frac{W-FH+2P}{S} + 1, \frac{W-FW+2P}{S} + 1)$

Convolutional Layer – stride



출처 : https://www.analyticsvidhya.com/blog/2022/03/basics-of-cnn-in-deep-learning/

Convolutional Layer - stride

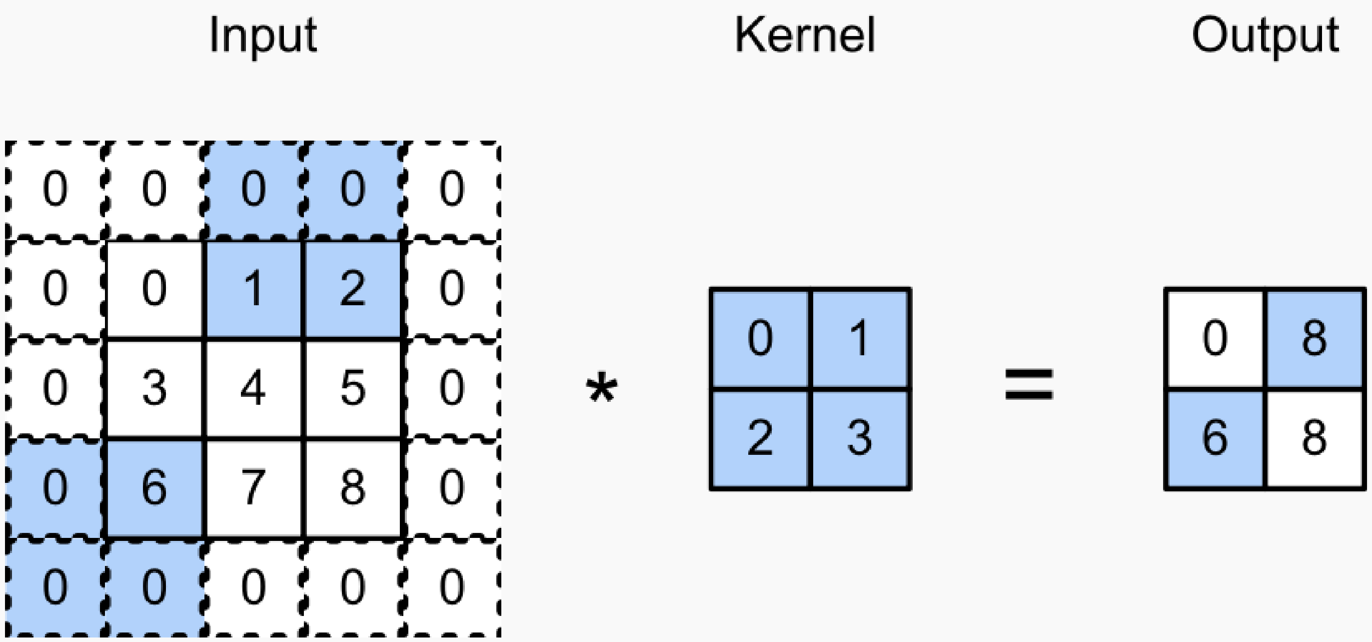


Fig. 7.3.3 Cross-correlation with strides of 3 and 2 for height and width, respectively.

출처 : Dive Into Deep Learning

Convolutional Layer - zero padding

0	0	0	0	0	0
0	35	19	25	6	0
0	13	22	16	53	0
0	4	3	7	10	0
0	9	8	1	3	0
0	0	0	0	0	0

출처 : https://blog.xrds.acm.org/2016/06/convolutional-neural-networks-cnns-illustrated-explanation/

Convolutional Layer - zero padding

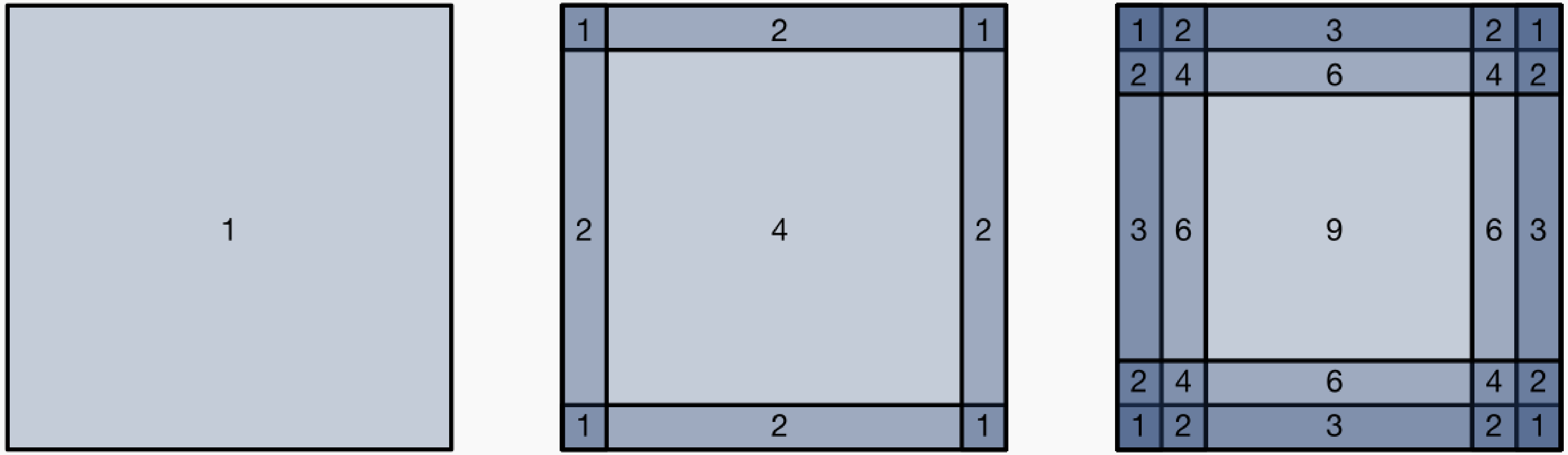


Fig. 7.3.1 Pixel utilization for convolutions of size 1×1, 2×2, and 3×3 respectively.

출처 : Dive Into Deep Learning

Convolutional Layer - zero padding

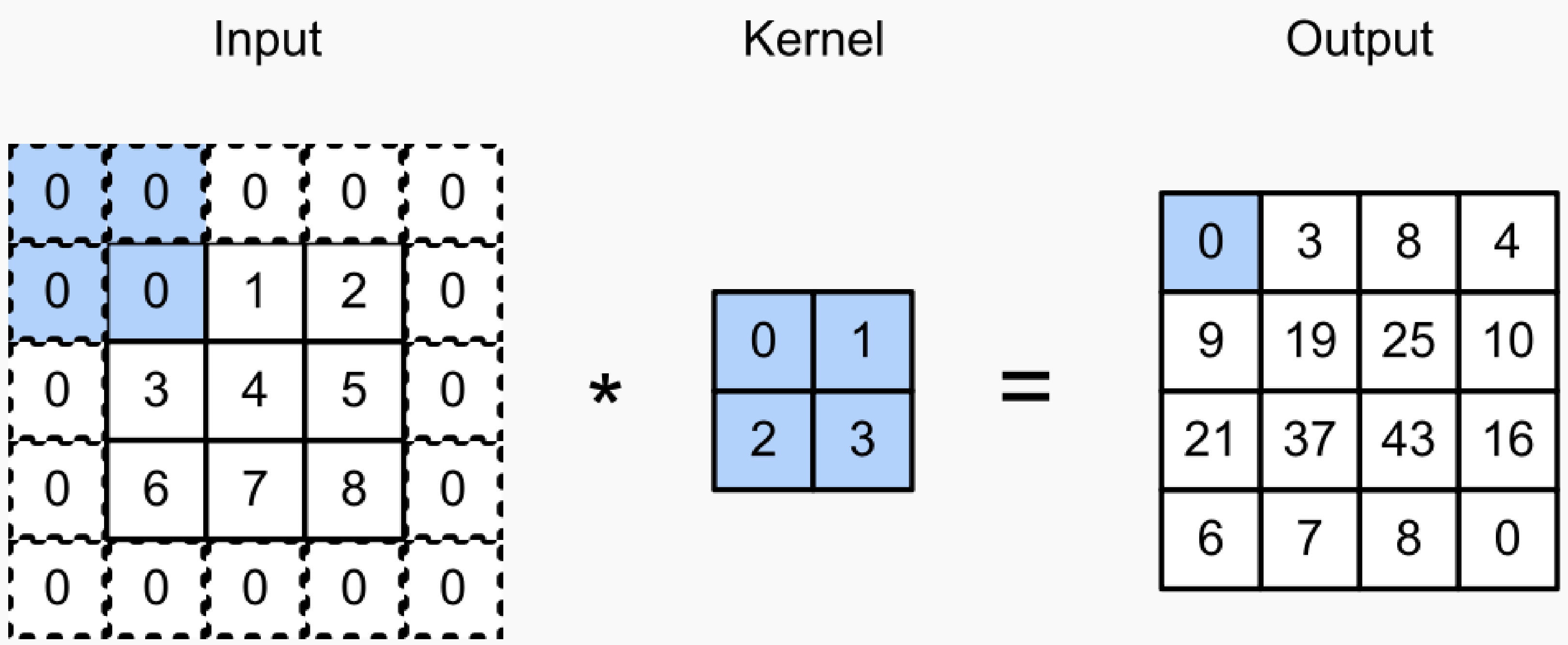


Fig. 7.3.2 Two-dimensional cross-correlation with padding.

출처 : Dive Into Deep Learning

Convolutional Layer – Spatial arrangement

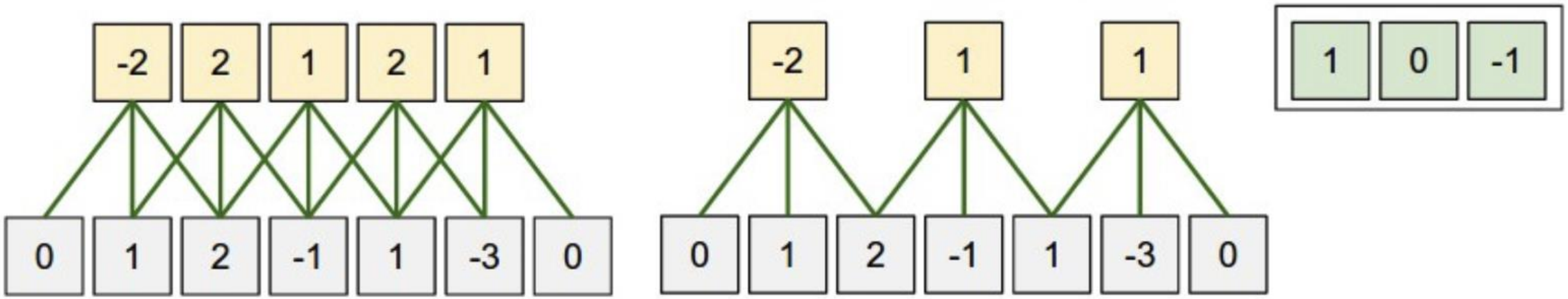


Illustration of spatial arrangement. In this example there is only one spatial dimension (x-axis), one neuron with a receptive field size of F = 3, the input size is W = 5, and there is zero padding of P = 1. **Left:** The neuron strided across the input in stride of S = 1, giving output of size (5 - 3 + 2)/1+1 = 5. **Right:** The neuron uses stride of S = 2, giving output of size (5 - 3 + 2)/2+1 = 3. Notice that stride S = 3 could not be used since it wouldn't fit neatly across the volume. In terms of the equation, this can be determined since (5 - 3 + 2) = 4 is not divisible by 3.
The neuron weights are in this example [1,0,-1] (shown on very right), and its bias is zero. These weights are shared across all yellow neurons (see parameter sharing below).

출처 : https://cs231n.github.io/convolutional-networks/#conv

Convolutional Layer - Let's Check!

- Input volume : 32 x 32 x 3
- 10개의 5 x 5 x 3 filters
- Stride 1
- zero padding 2

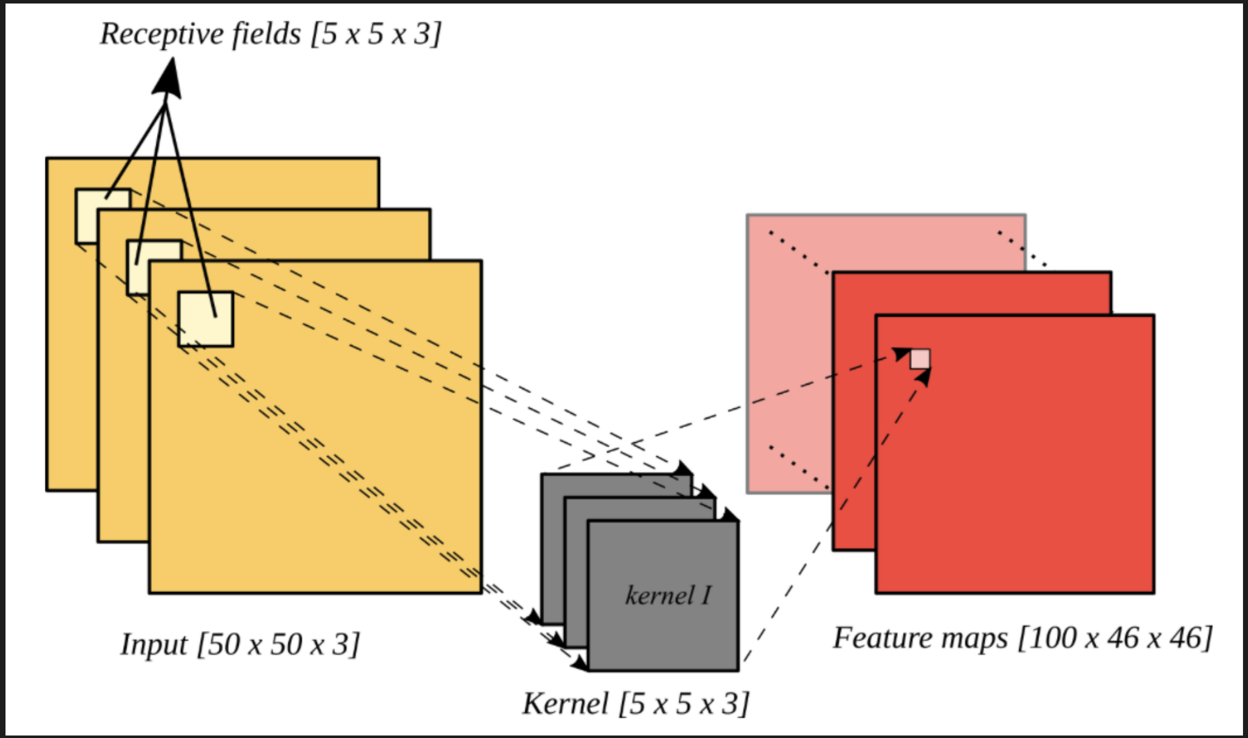
- 이 때, Output Volume size는?

Convolutional Layer - Let's Check!

- Input volume : 32 (W) x 32 x 3
- 10개의 5 (F) x 5 x 3 filters
- Stride 1 (S)
- zero padding 2 (P)

- 이 때, Output Volume size는?
  - Output volume = (W - F + 2P)/S + 1 = (32 - 5 + 2 * 2) / 1 + 1 = 32
  - 32 x 32 x 10

Convolutional Layer –Feature Map and Receptive Field



출처 : https://ai.stackexchange.com/questions/8701/what-is-the-difference-between-a-receptive-field-and-a-feature-map

- feature map : convolution operation의 output
- receptive field : neuron의 activation 계산에 사용되는 부분

Convolutional Layer - Feature Map and Receptive Field

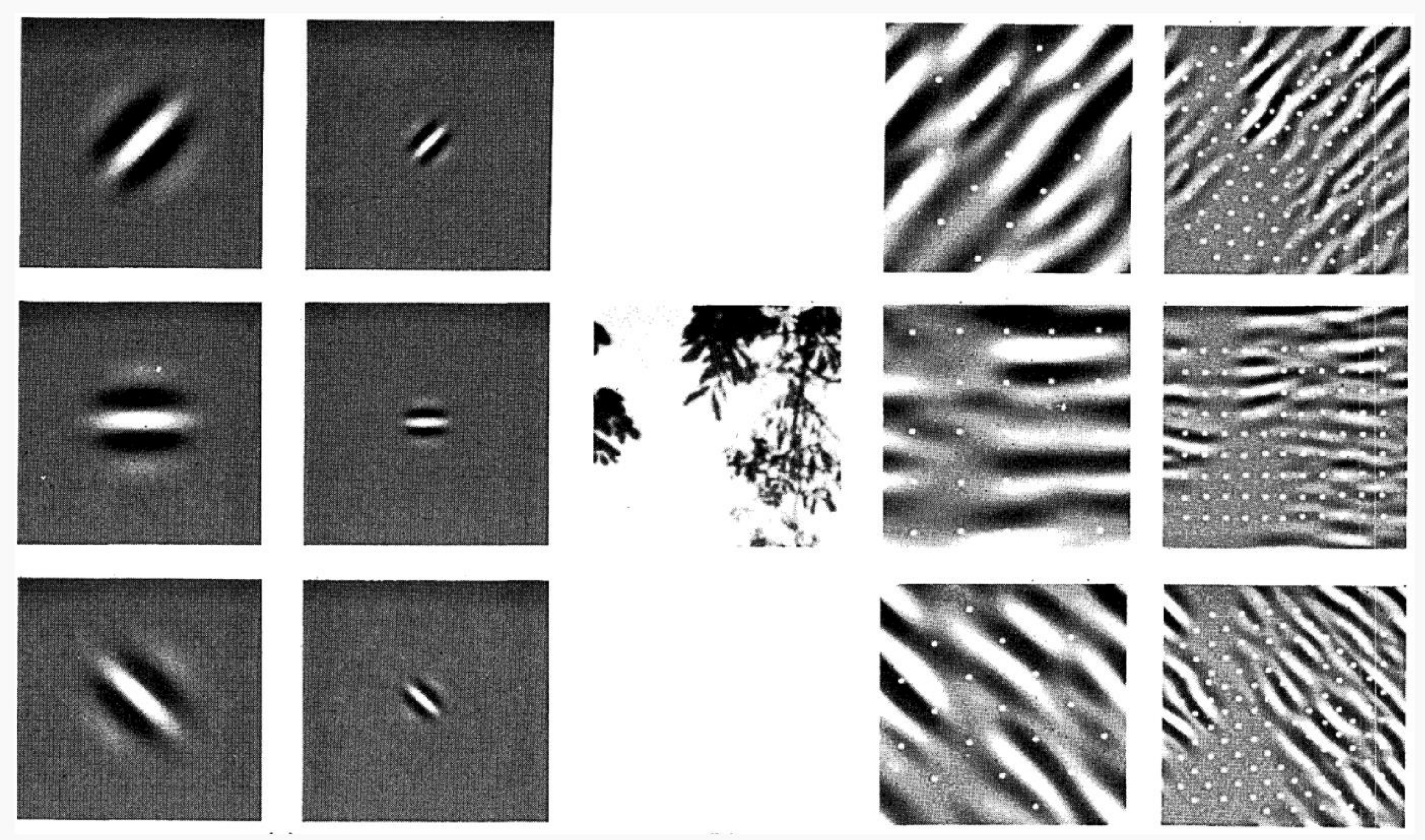
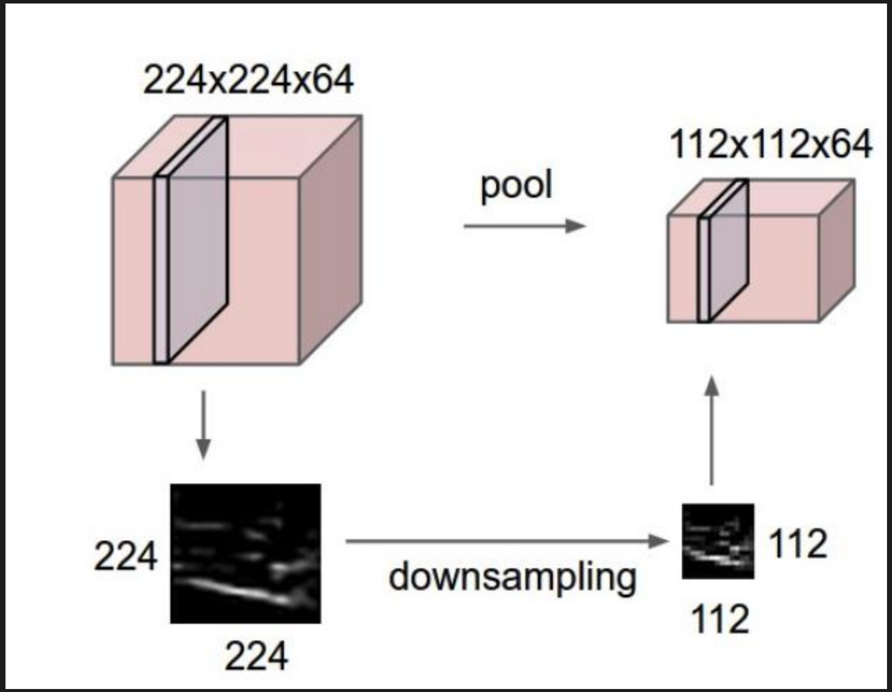


Fig. 7.2.2 Figure and caption taken from Field (1987): An example of coding with six different channels. (Left) Examples of the six types of sensor associated with each channel. (Right) Convolution of the image in (Middle) with the six sensors shown in (Left). The response of the individual sensors is determined by sampling these filtered images at a distance proportional to the size of the sensor (shown with dots). This diagram shows the response of only the even symmetric sensors.

출처 : Dive Into Deep Learning

Pooling Layer

Pooling?



출처 : https://cs231n.github.io/convolutional-networks/#conv

- 계속해서 Convolutional Layer의 정보를 더해 간다.
  - Layer가 깊어질 수록, Receptive field가 넓어진다.
- Convolutional Layer의 민감성을 완화하고, 공간적으로 downsampling 하기 위해!

Max-pooling

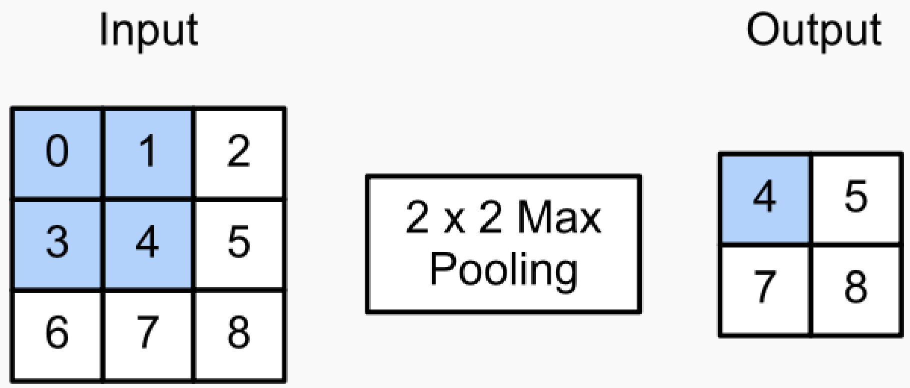
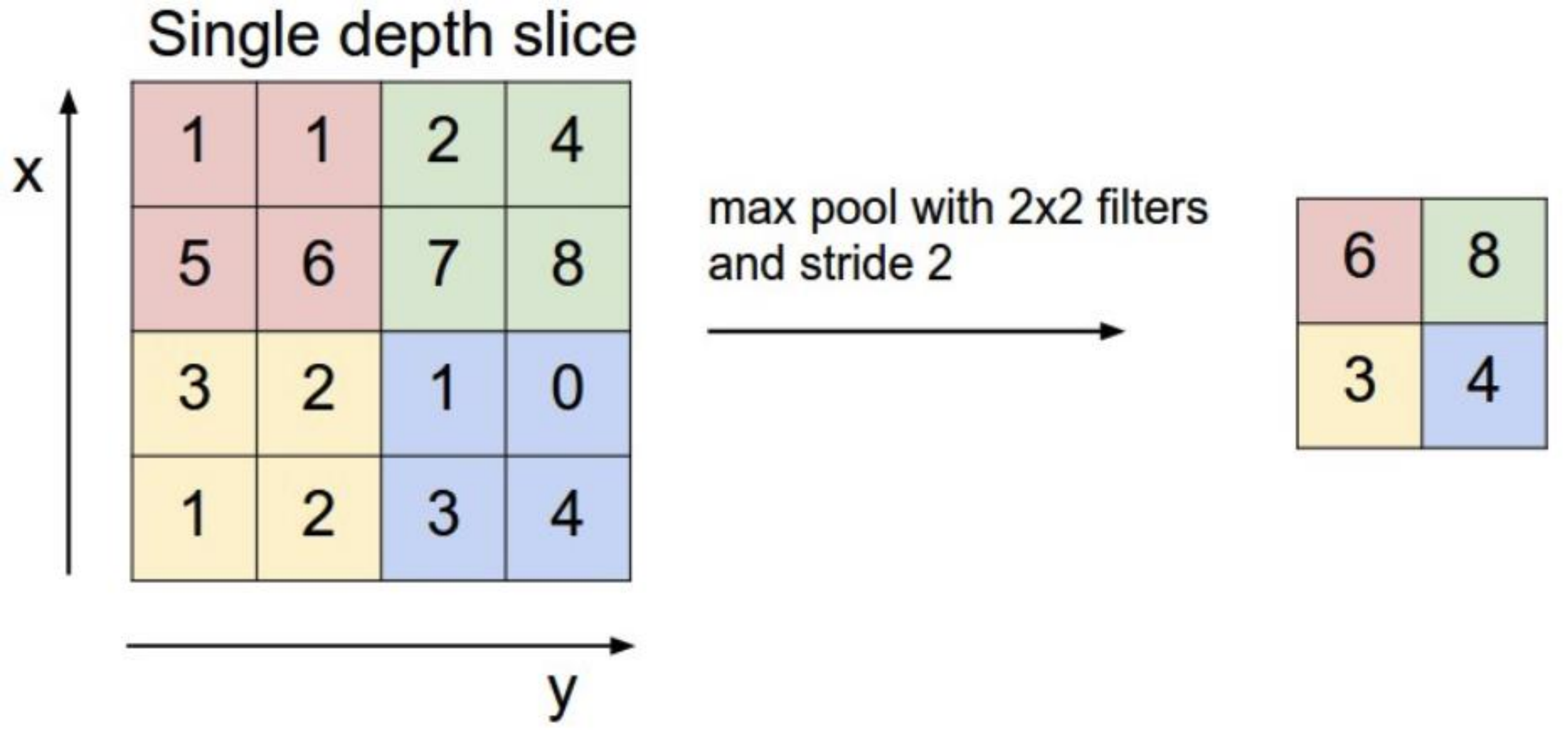


Fig. 7.5.1 Max-pooling with a pooling window shape of 2×2. The shaded portions are the first output element as well as the input tensor elements used for the output computation: $\max(0, 1, 3, 4) = 4$.

출처 : Dive Into Deep Learning

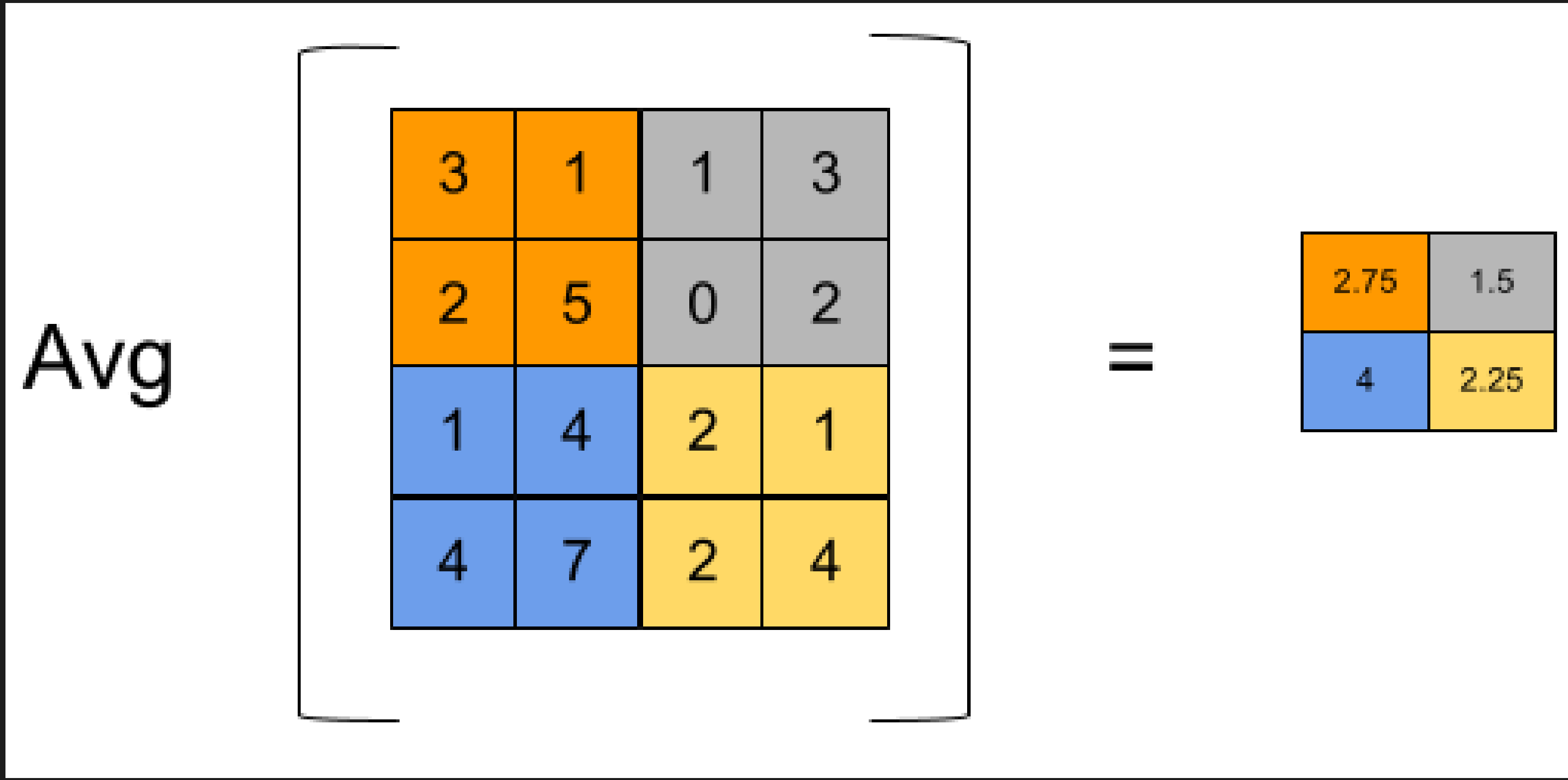
Max-pooling



출처 : https://cs231n.github.io/convolutional-networks/#conv

Average Pooling

- Image 분야에선 Max Pooling이 일반적

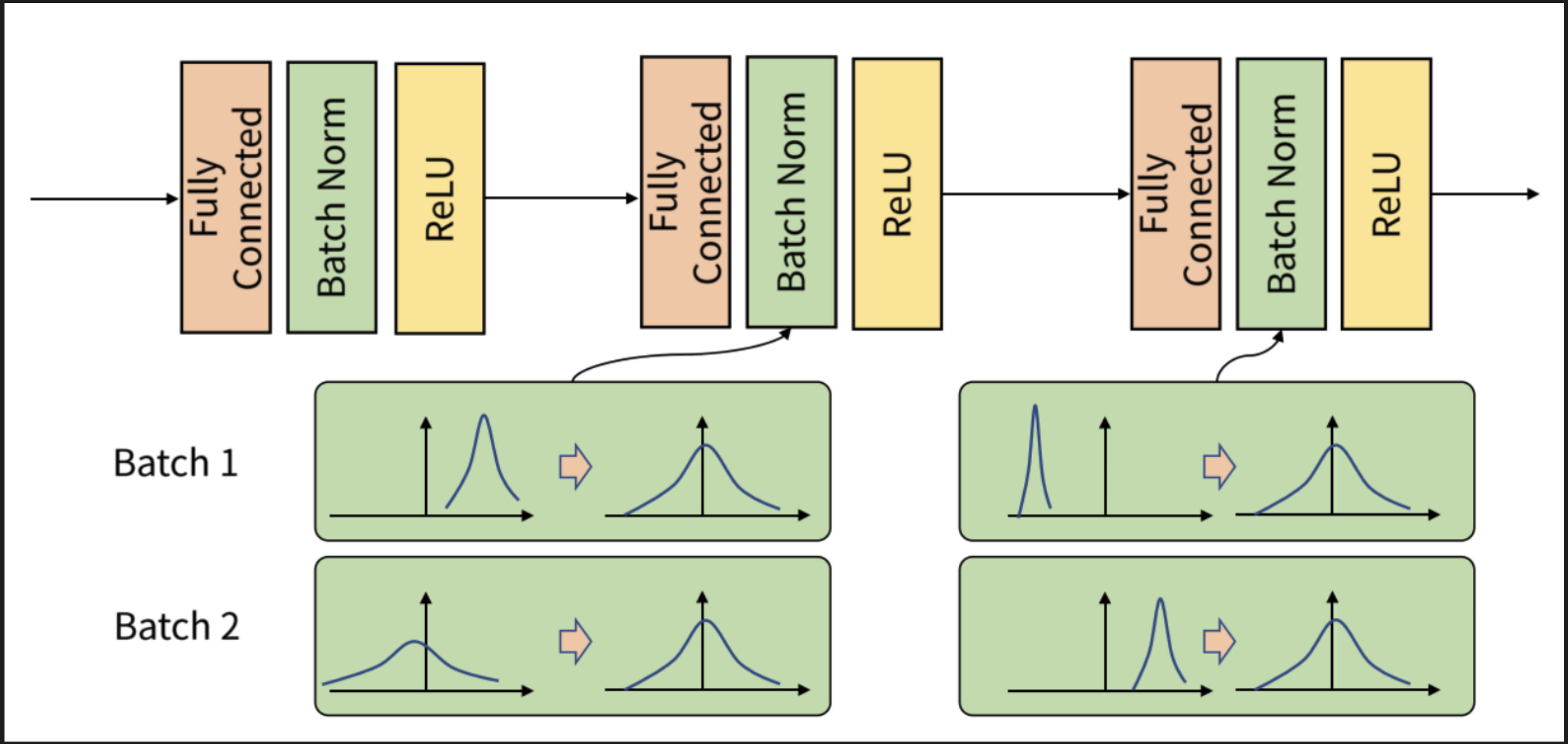


Other Techniques

- Batch Normalization
  - Stochastic Gradient Descent 하는 단위 : Batch
  - Batch data들의 distribution이 다르다. 이를 맞춰주자!

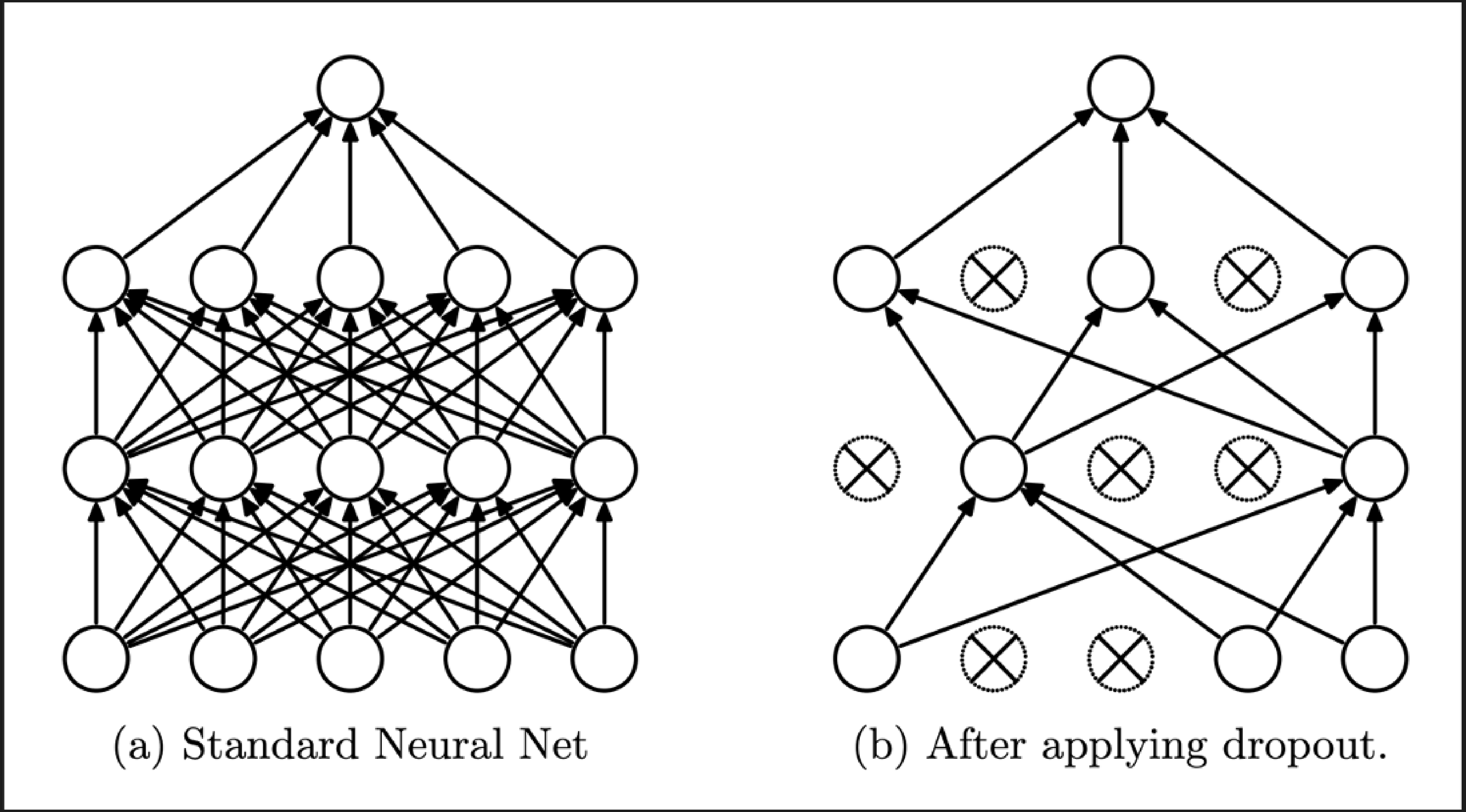
- Dropout
  - Parameter를 Random하게 학습 시키지 않아 Overfitting 방지
  - Ensemble과 같은 효과

Batch Normalization



출처 : https://gaussian37.github.io/dl-concept-batchnorm/

Dropout



출처 : Dropout: a simple way to prevent neural networks from overfitting, Srivastava et al., Journal of Machine Learning Research, 2014

Case Study : Modern CNNs

LeNet Overview

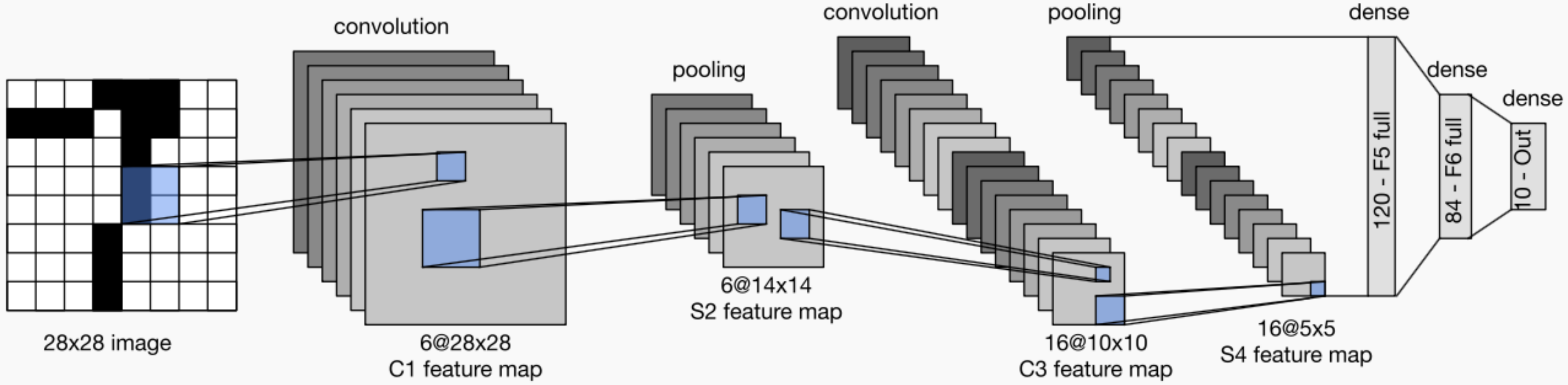


Fig. 7.6.1 Data flow in LeNet. The input is a handwritten digit, the output a probability over 10 possible outcomes.

출처

출처 : Dive Into Deep Learning

- Yann LeCun!

LeNet-5

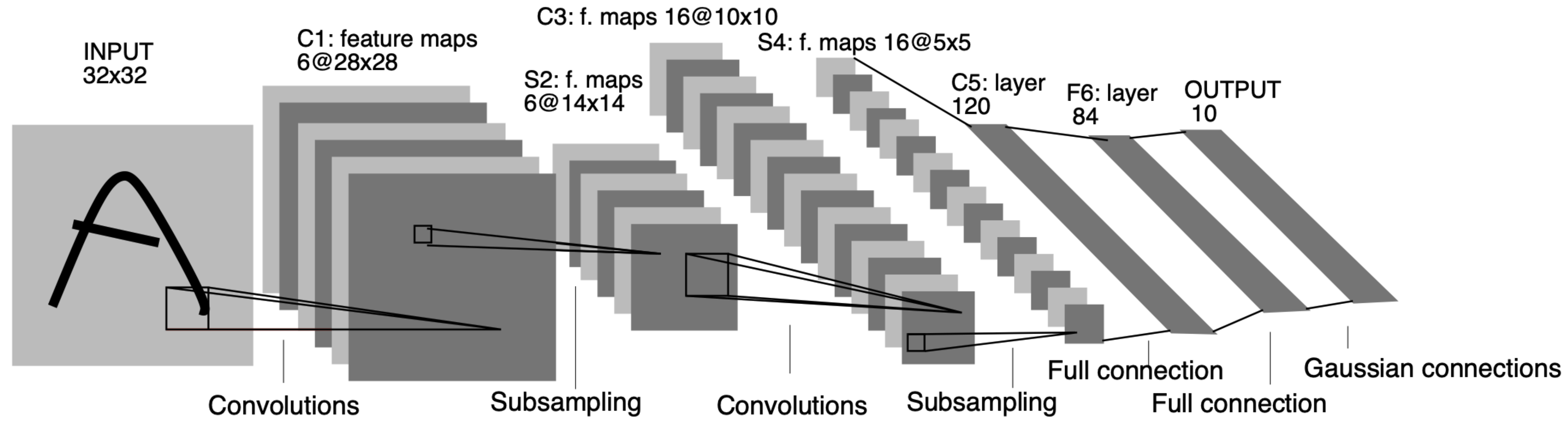


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

출처 : Gradient Based Learning Applied to Document Recognition (LeCun et al., 1995)

- Breakthrough for Neural Networks

Limitation of LeNet

- 더 크고, realistic dataset에 적용 시키기 어려웠다.
  - 현실 세계의 image는 더 고차원
  - representation learning

- 왜 LeNet(1995년)에는 불가능했지만, AlexNet(2012년)에는 가능했는가?
  - Missing Ingredient

Missing Ingredient: Data

- ImageNet dataset (2009년 released)
  - 100만 장 (CIFAR-10은 6만장)
  - 224 x 224 pixels

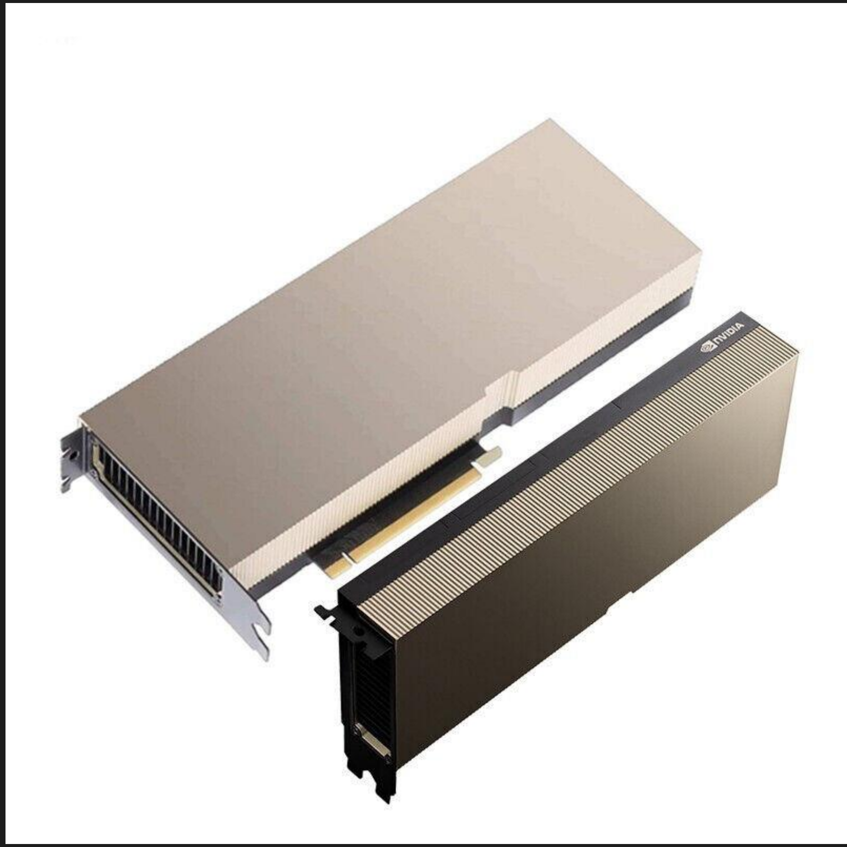
출처 : https://www.kaggle.com/datasets/sautkin/imagenet1kvalid

Missing Ingredient: Hardware

NVIDIA GeForce GTX 580

출처 : https://www.techpowerup.com/gpu-specs/geforce-gtx-580.c270

NVIDIA Tesla A100 Ampere



출처 : https://www.ebay.com/p/26042475154

Deep Convolutional Neural Networks (AlexNet)

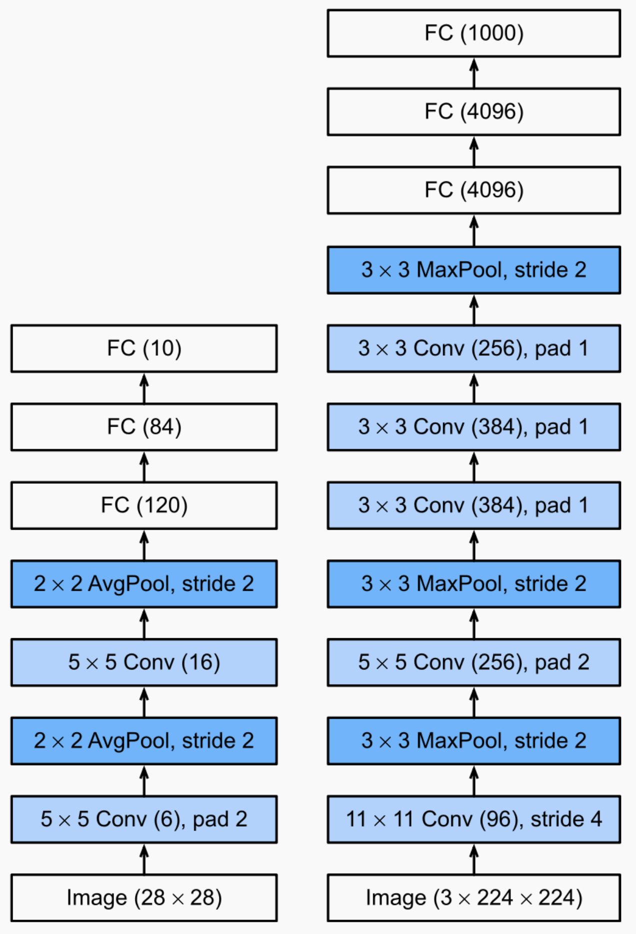


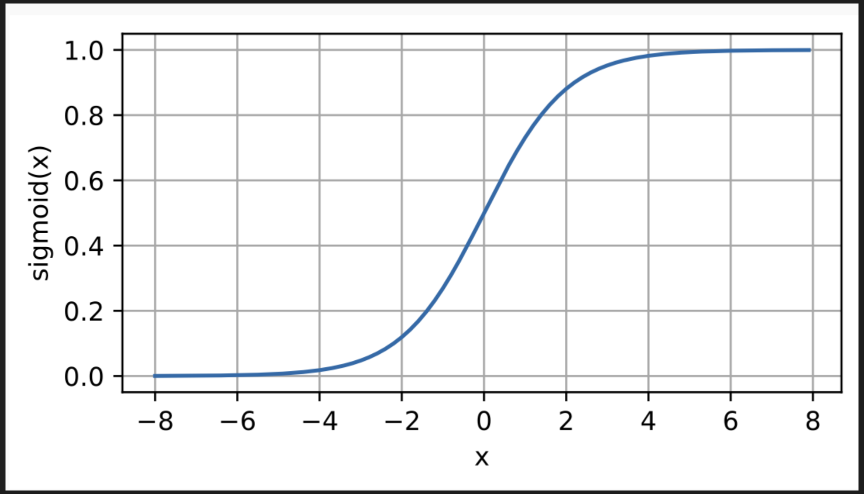
Fig. 8.1.2 From LeNet (left) to AlexNet (right).

출처 : Dive Into Deep Learning

- ImageNet Challenge 2012년에 혜성 같이 등장한 'AlexNet'
- Geoffrey Hinton
- 두 개의 GPU에서 학습 되도록 설계
- LeNet과의 차이점
  - Deeper!
  - No sigmoid, Yes ReLU!

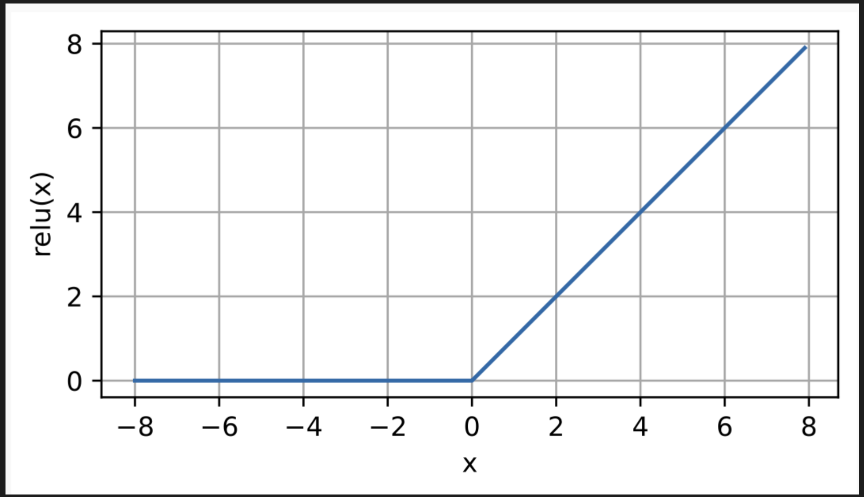
Recap: Sigmoid vs ReLU

Sigmoid



출처 : Dive Into Deep Learning

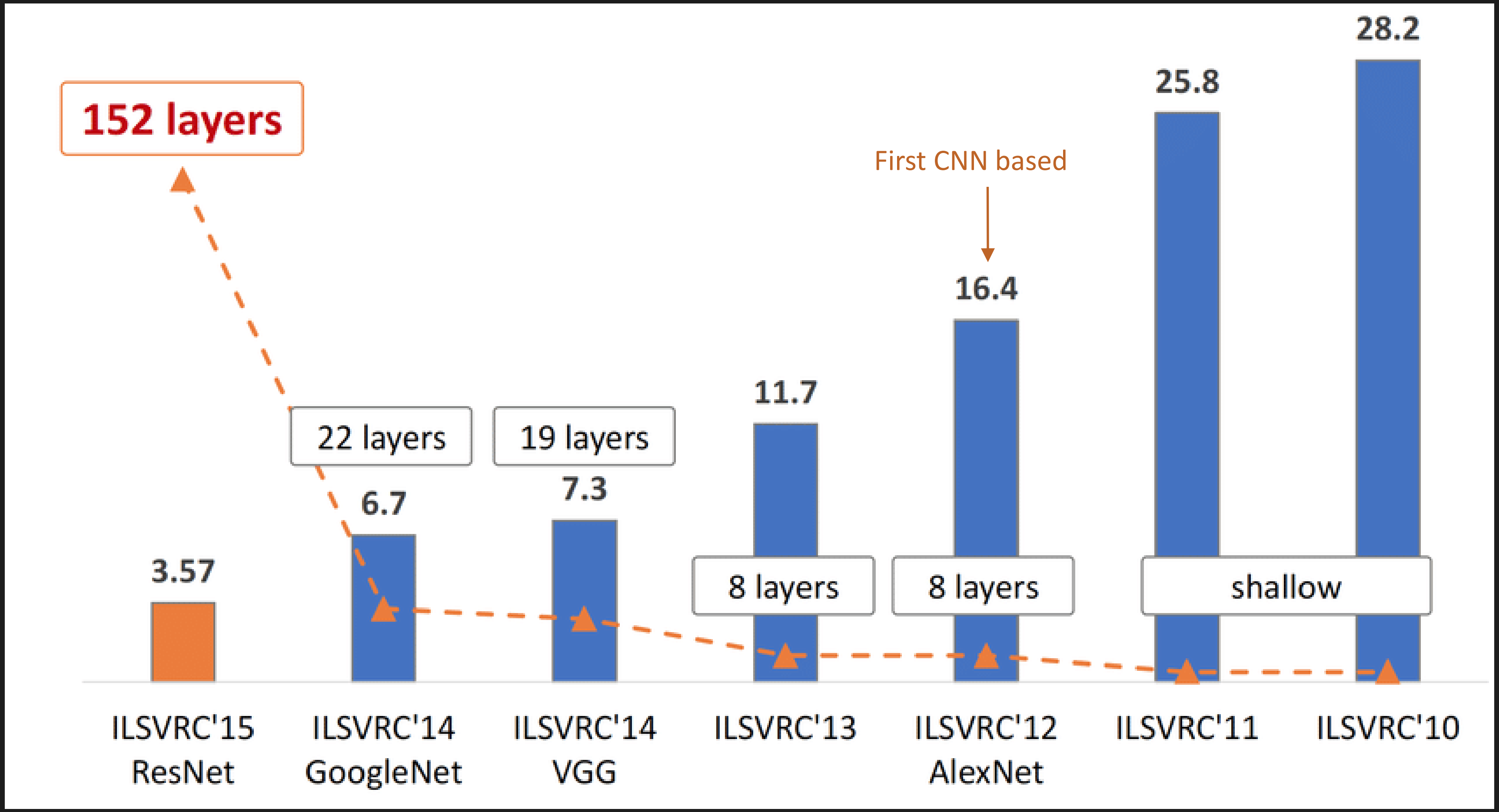
ReLU



출처 : Dive Into Deep Learning

- 더 간단한 ReLU (exponentiation ❌) → 미분이 쉽다!
- Sigmoid(x)가 0 또는 1에 가까우면 기울기가 0이 된다. initialized가 잘 되어야 함.
  - ReLU는 기울기 전달 가능. 학습이 쉽다.

ImageNet Challenge – Much Deeper!



출처

출처 : https://www.researchgate.net/figure/The-evolution-of-the-winning-entries-on-the-ImageNet-Large-Scale-Visual-Recognition_fig1_321896881

VGG Network

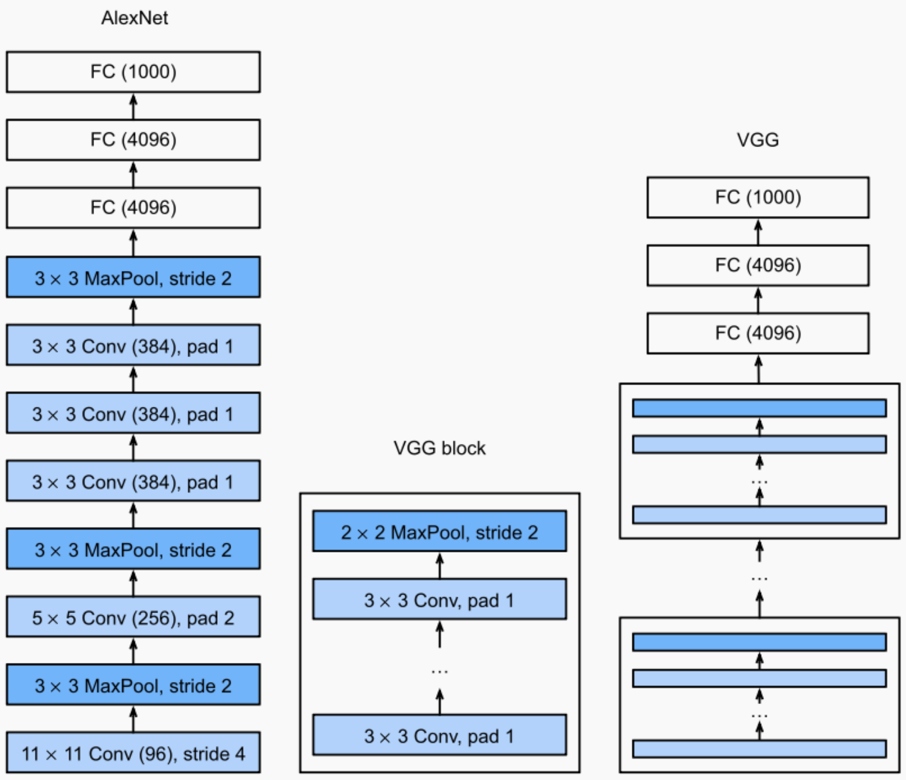
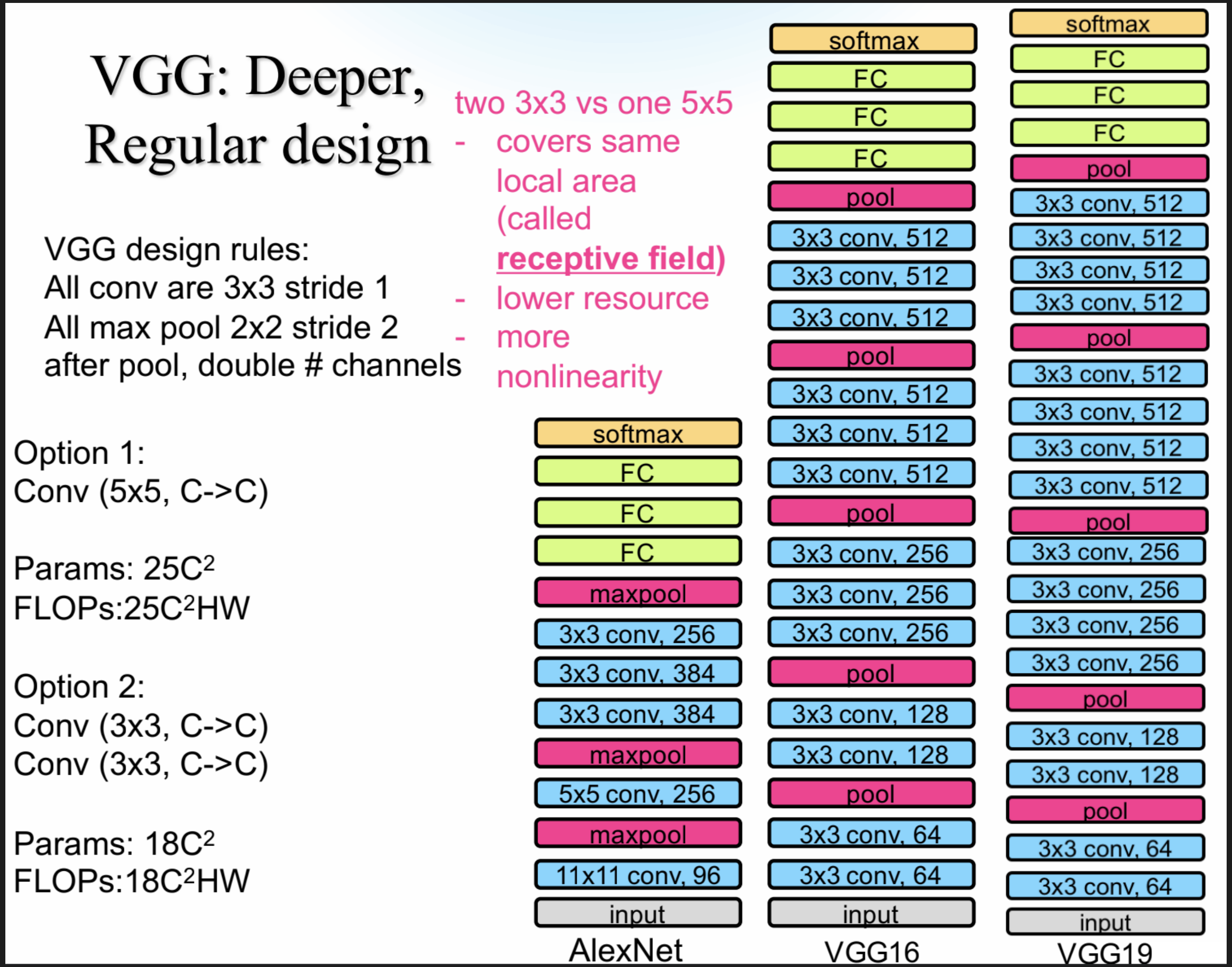


Fig. 8.2.1 From AlexNet to VGG. The key difference is that VGG consists of blocks of layers, whereas AlexNet's layers are all designed individually.

출처 : Dive Into Deep Learning

- 더 깊어진 Network
- Heuristic concepts

VGG Network



출처 미상

- 3 x 3 두 개 == 5 x 5 한 개
  - 같은 receptive field
- 하지만 3 x 3이 더 적은 resource로 더 많은 유동성.
  - 3 x 3 Conv로 통일!

GoogLeNet

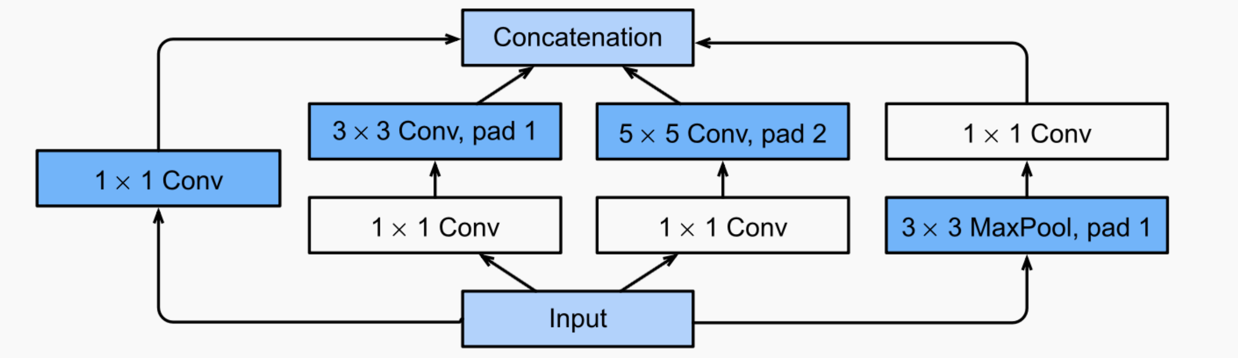


Fig. 8.4.1 Structure of the Inception block.

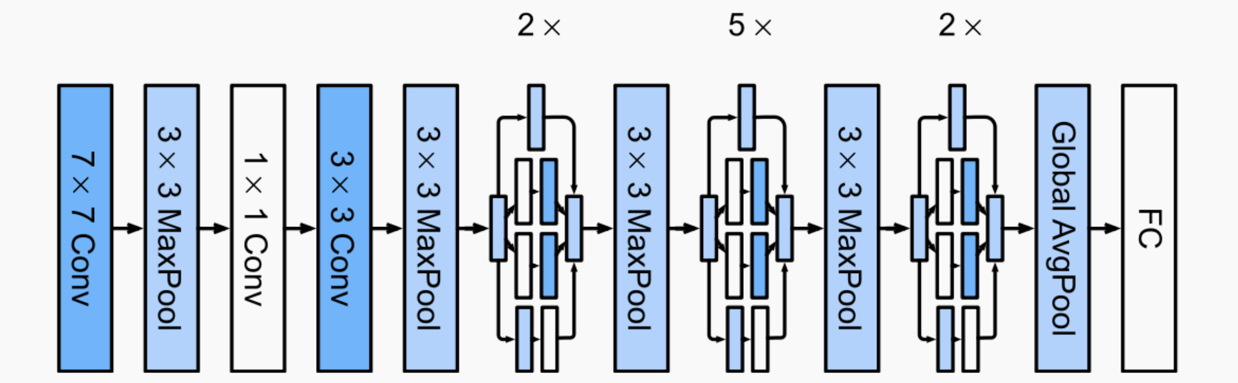
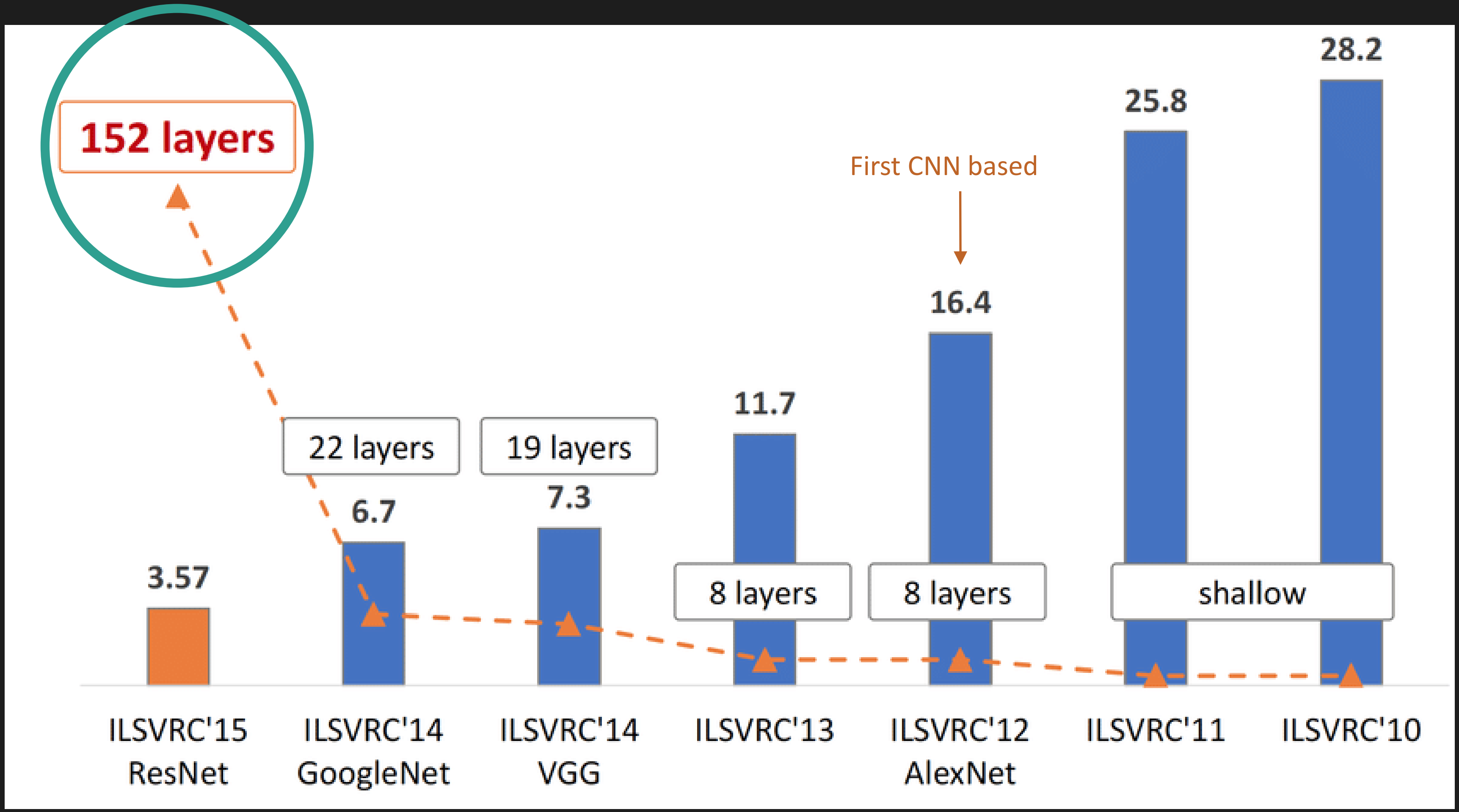


Fig. 8.4.2 The GoogLeNet architecture.

출처 : Dive Into Deep Learning

- 'Network in Network' (NiN) 구조
  - FC가 많은 Parameter 차지
  - 중간에 FC Layer를 못 넣음
  - 1 x 1 Conv로 nonlinearity
  - global average pooling

What the hack??



출처 : https://www.researchgate.net/figure/The-evolution-of-the-winning-entries-on-the-ImageNet-Large-Scale-Visual-Recognition_fig1_321896881

Much, Much, Much Deeper - Residual Connection

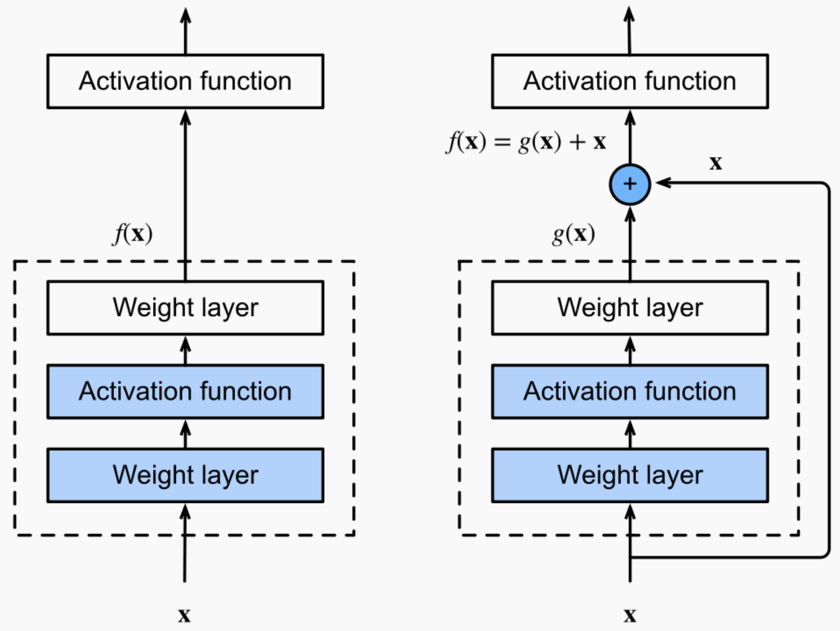


Fig. 8.6.2 In a regular block (left), the portion within the dotted-line box must directly learn the mapping $f(\mathbf{x})$. In a residual block (right), the portion within the dotted-line box needs to learn the residual mapping $g(\mathbf{x}) = f(\mathbf{x}) - \mathbf{x}$, making the identity mapping $f(\mathbf{x}) = \mathbf{x}$ easier to learn.

출처 : Dive Into Deep Learning

- Vanishing Gradient Problem
  - Function Classes
- Fit layer to ‘residual’ :
  - $g(x) = f(x) - x$

ResNet

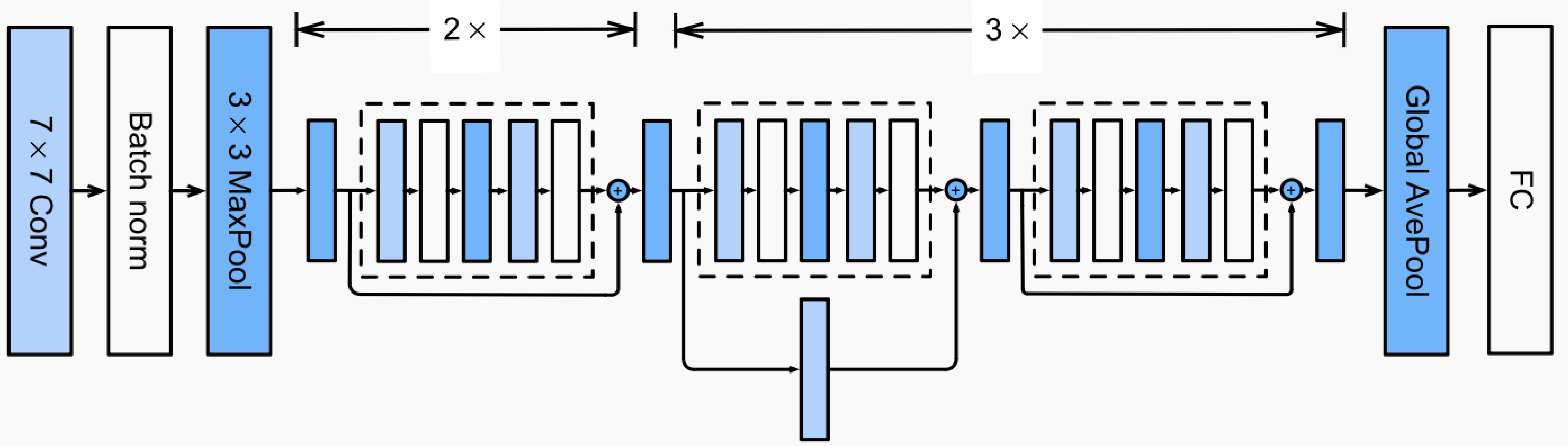
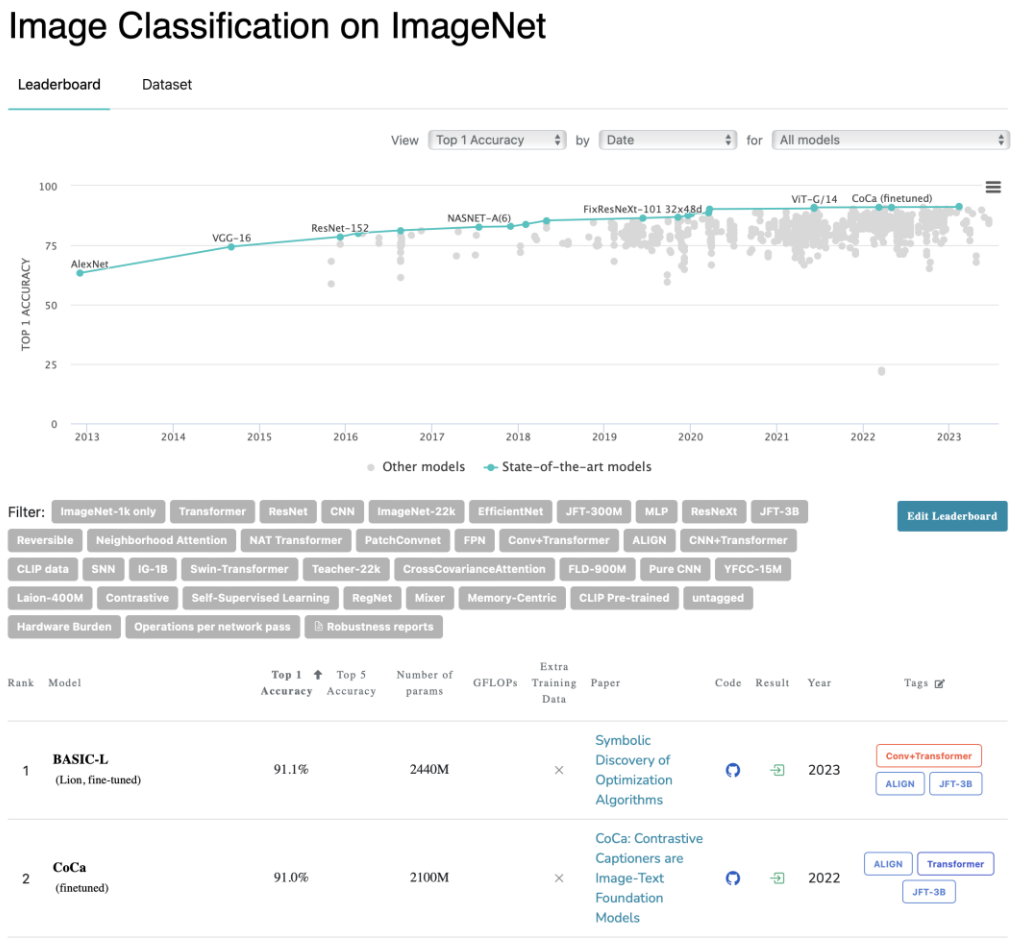


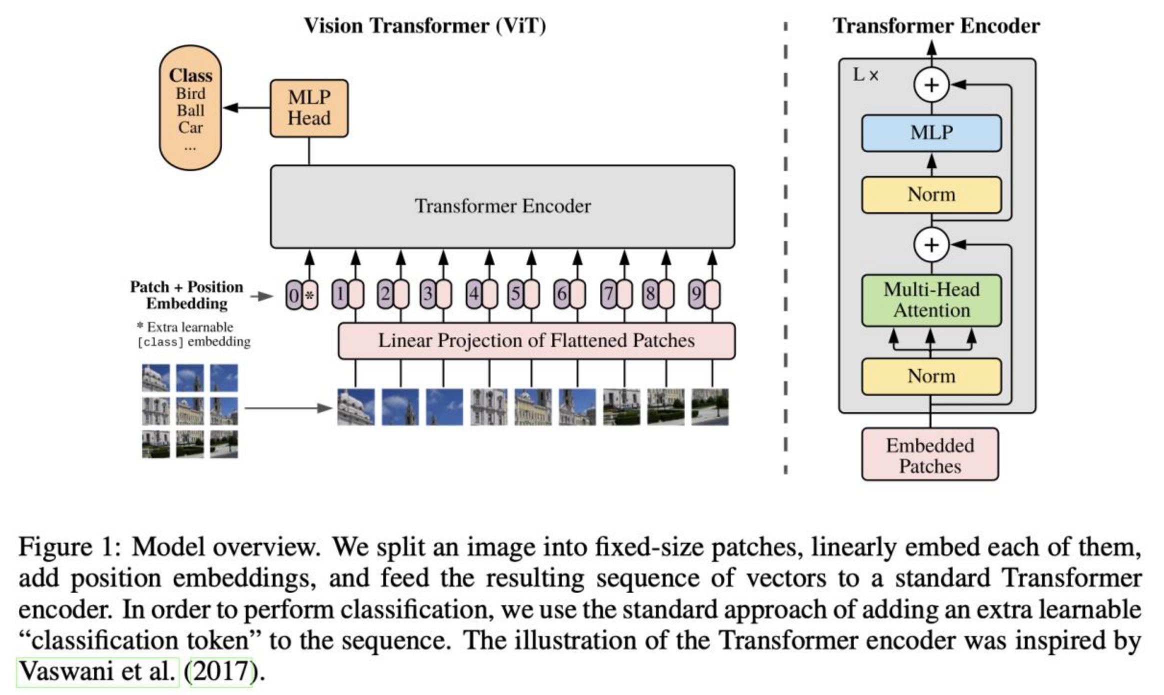
Fig. 8.6.4 The ResNet-18 architecture.

출처 : Dive Into Deep Learning

Now then…



출처 : [Papers with code]



출처 : [An image is worth 16x16 words transformers for image recognition at scale (ICLR 2021)]

- Transformer 전성 시대!

Other Image Tasks

- Classic
  - Classification, Object Detection, Semantic Segmentation, Instance Segmentation

- New
  - ViT, Image-Text multimodality, Video, 3D Vision, Generation Model

Discussion

팀 질문

- CNN에 Average pooling보다 Max pooling이 더 많이 사용되는 이유는 무엇일까?

개인 질문

- Inductive bias란? Convolution에는 어떤 inductive bias가 숨어있을까?
  딥러닝 모델에 inductive bias는 어떤 영향을 줄까?

- Due: 다음 수업 시작 전까지
  - AIKU Notion → 일정 → DeepIntoDeep 해당 회차
  - 본인 칸에 팀별 토론 내용 & 자신의 생각 정리!

Introduction to PyTorch

PyTorch?

출처 : https://tutorials.pytorch.kr/

- 오픈소스 딥러닝 라이브러리
- Tensor를 GPU 상에 올려 연산 가능.
- 추천하는 사이트 : 파이토치 한국어 튜토리얼
- Documentation
- Rival : TensorFlow
  - PyTorch가 모델을 수정하기 더 쉬워 연구자들 사이에서는 더 일반적.

개발 환경

출처 : Google

- 서버에서 작업한다면 PyTorch + CUDA
  - 보통 Anaconda를 사용해 가상 환경 사용
- 하지만… PyTorch 버젼 관리가 어려움
  - Colab!

Tensor

- Tensor
  - array, matrix.
    - Numpy의 ndarray와 유사. GPU 상에서 돌아간다는 차이만! (.to(device))
    - Autograd에 최적화 됨.
- 참고 : Tensor 차원 관리가 어렵다면? einops

```
import torch
import numpy as np

# Directly
data = [[1, 2], [3, 4]]
x_data = torch.tensor(data)

# From Numpy
np_arrary = np.array(data)
x_np = torch.from_numpy(np_array)
```

Data, DataLoader

- Data를 불러오고, 학습에 사용하기 편하도록 묶어주는 class
  - Custom dataset : __init__, __len__, __getitem__
  - 과제에서는 주어짐
- 전반적인 학습의 과정
  - Dataloader 만들기
  - 모델 정의하기
  - train 함수 만들기
- 튜토리얼 꼭 읽어 보시길 추천 드립니다!

FAQ

- PyTorch Document 어디를 보면 되나?
  - torch.nn, torch.nn.functional, torch.optim을 주로 봅니다.
  - Dataloader, train 함수는 비슷하게 반복되어 사용됩니다.
- nn과 nn.functional은 뭐가 다른가?
  - nn은 Class로 정의되어 있고, functional은 함수로 정의되어 있습니다.
- model.eval()이 무슨 뜻인가요? torch.no_grad()는?
  - eval : model을 test 모드로 전환해 줍니다.
  - no_grad : test 할 때에는 model의 Gradient를 추적하는 것을 꺼 버립니다. 참고

과제 설명

- Due: 2024.01.25. 제출 방법은 추후 공지

1. 과제 다운로드, 본인 Google Drive에 과제 업로드
2. Colab으로 열기.
   - 런타임 → 런타임 유형 변경 → 하드웨어 가속기 'GPU'
3. 맨 위 코드 블럭에 'FOLDERNAME' 변경하기
   - **자신이 업로드 한 경로 제대로 적기!!** → 안 되면 문의
4. 위에서부터 하나씩 실행해 가면서 코드 이해해 보기
   - 'TODO'로 감싸져 있는 부분만 코드 추가하기!
   - 중간 중간에 있는 설명 작성 부분에 자기 생각 적기

주목해야 할 Class / Function

- Class
  - Linear
  - Conv2D
  - SGD
  - Dropout2D
  - MaxPool2D
  - BatchNorm2D
- Function
  - Conv2D
  - ReLU

Discussion

팀 질문

- CNN에 Average pooling보다 Max pooling이 더 많이 사용되는 이유는 무엇일까?

개인 질문

- Inductive bias란? Convolution에는 어떤 inductive bias가 숨어있을까?
  딥러닝 모델에 inductive bias는 어떤 영향을 줄까?

- Due: 다음 수업 시작 전까지
  - AIKU Notion → 일정 → DeepIntoDeep 해당 회차
  - 본인 칸에 팀별 토론 내용 & 자신의 생각 정리!

감사합니다.