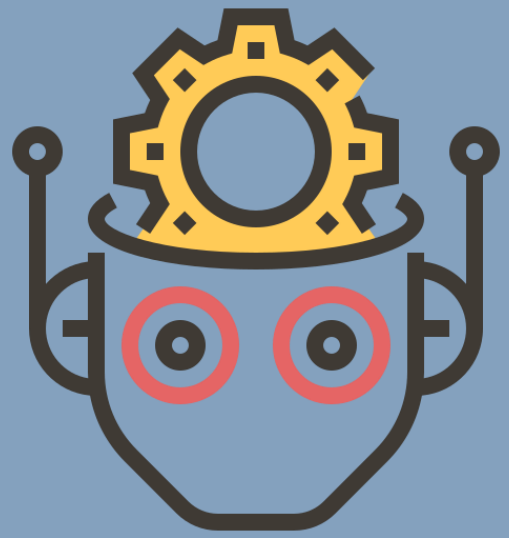
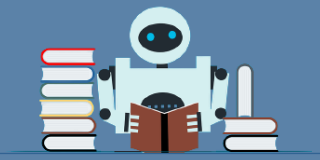


Deep Learning

스마트인재개발원
Smart Human Resources Development
정 봉 균 연구원

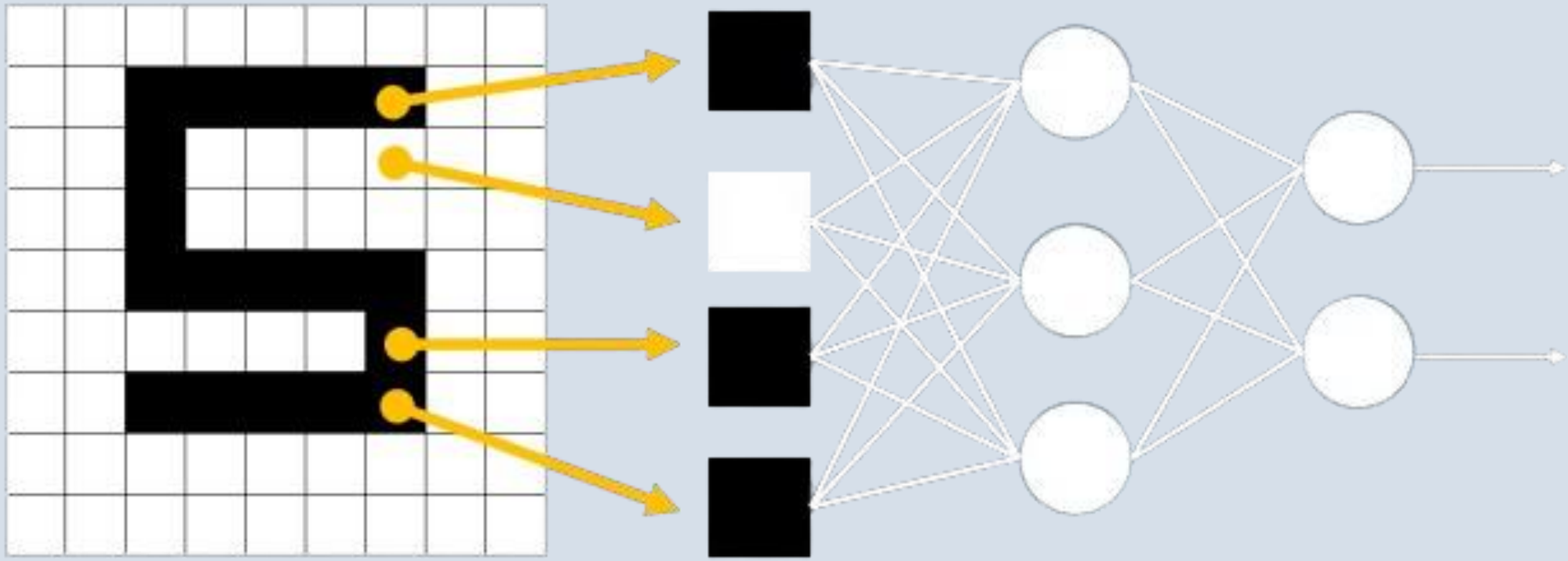


1	합성곱 신경망에 대해 이해할 수 있다.
2	Keras를 활용해 합성곱 신경망을 구성 할 수 있다.
3	신경망 성능 개선을 위한 방법을 이해 할 수 있다.

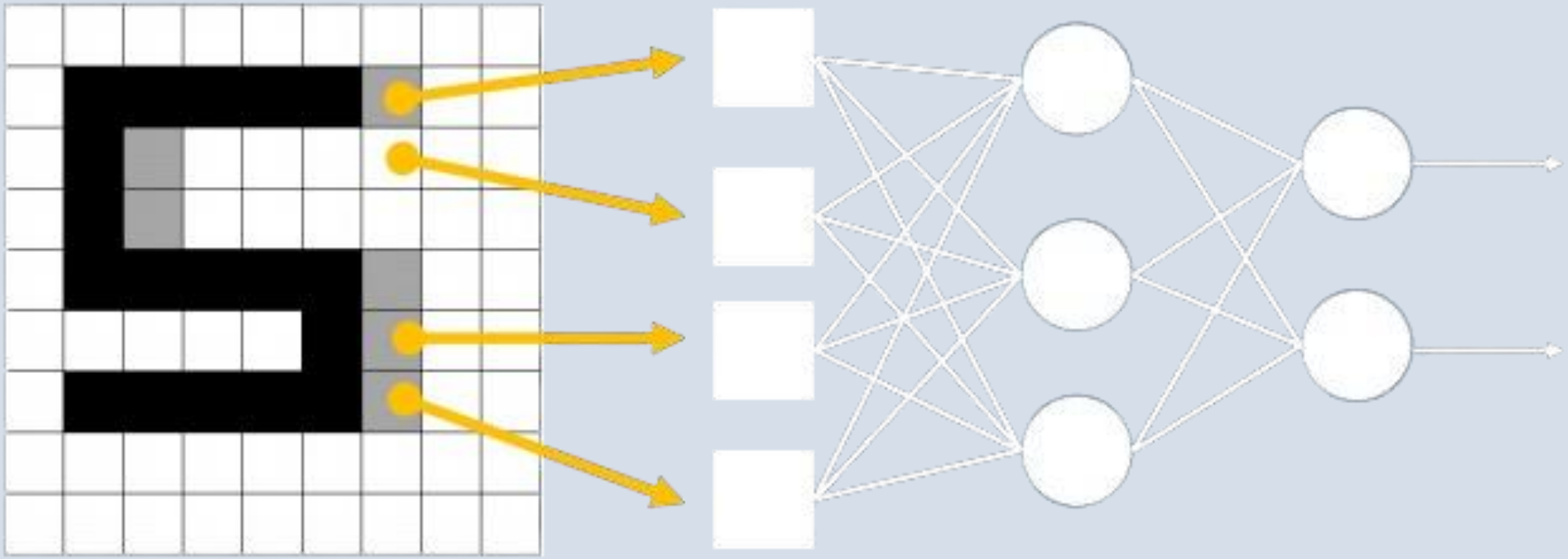
MLP 이미지 분석

- MLP는 층이 깊어지고 뉴런의 수가 많아지면 가중치 수가 급격히 늘어남
- MLP 신경망을 이미지 처리에 사용한다면 이미지의 어떤 특정 패턴이 존재하는 위치에 민감하게 동작하며 패턴의 위치에 종속적인 결과를 얻게 됨
- MLP의 경우 아래 3 개의 숫자 "5"는 패턴이 다르다고 판단함
- MLP로 이러한 숫자 인식을 하려면 숫자의 크기를 비슷하게 맞추어야 함

CNN(Convolutional Neural Network)

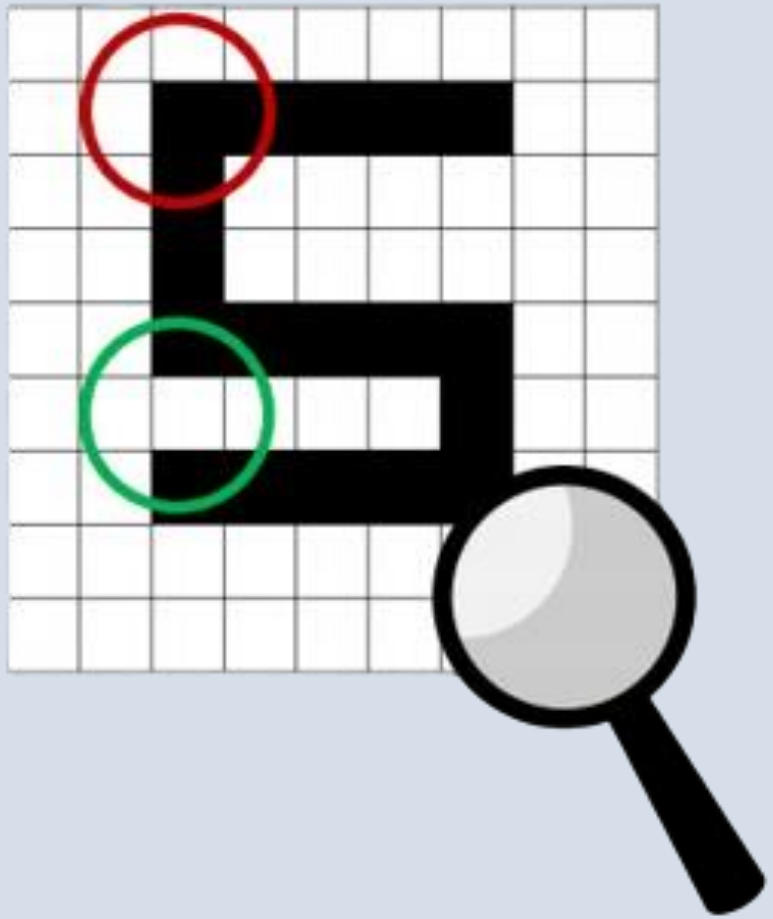


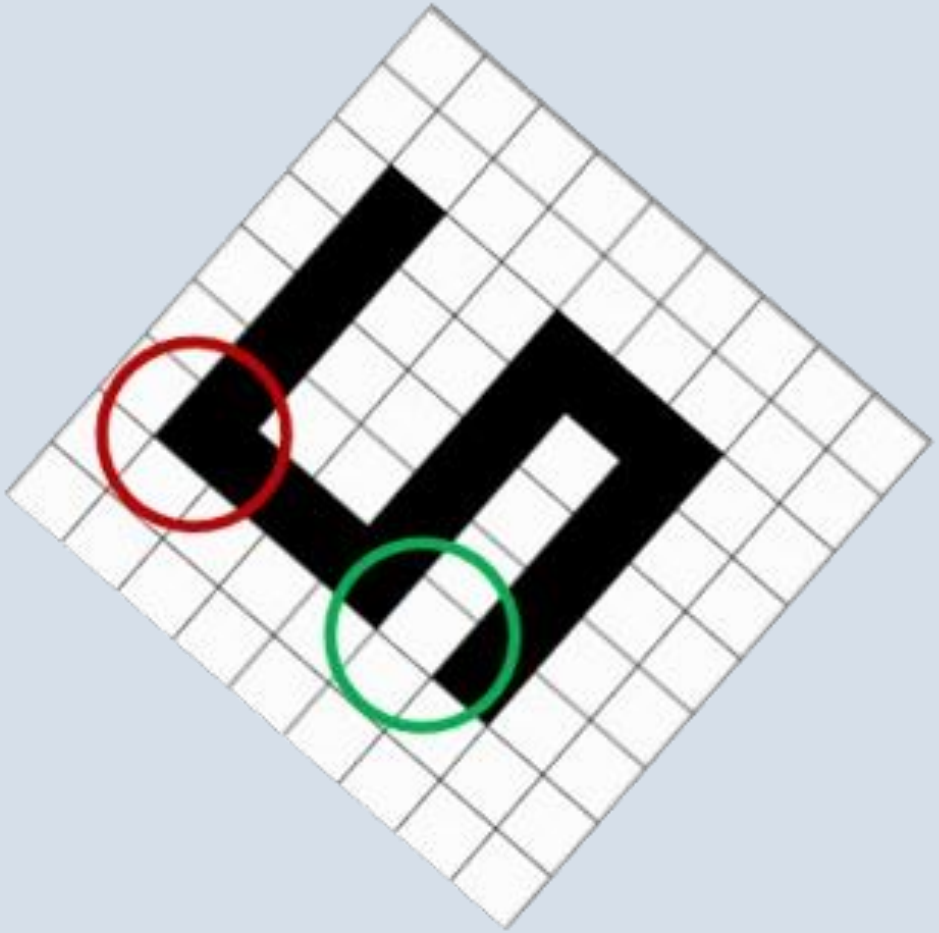
CNN(Convolutional Neural Network)



CNN(Convolutional Neural Network)

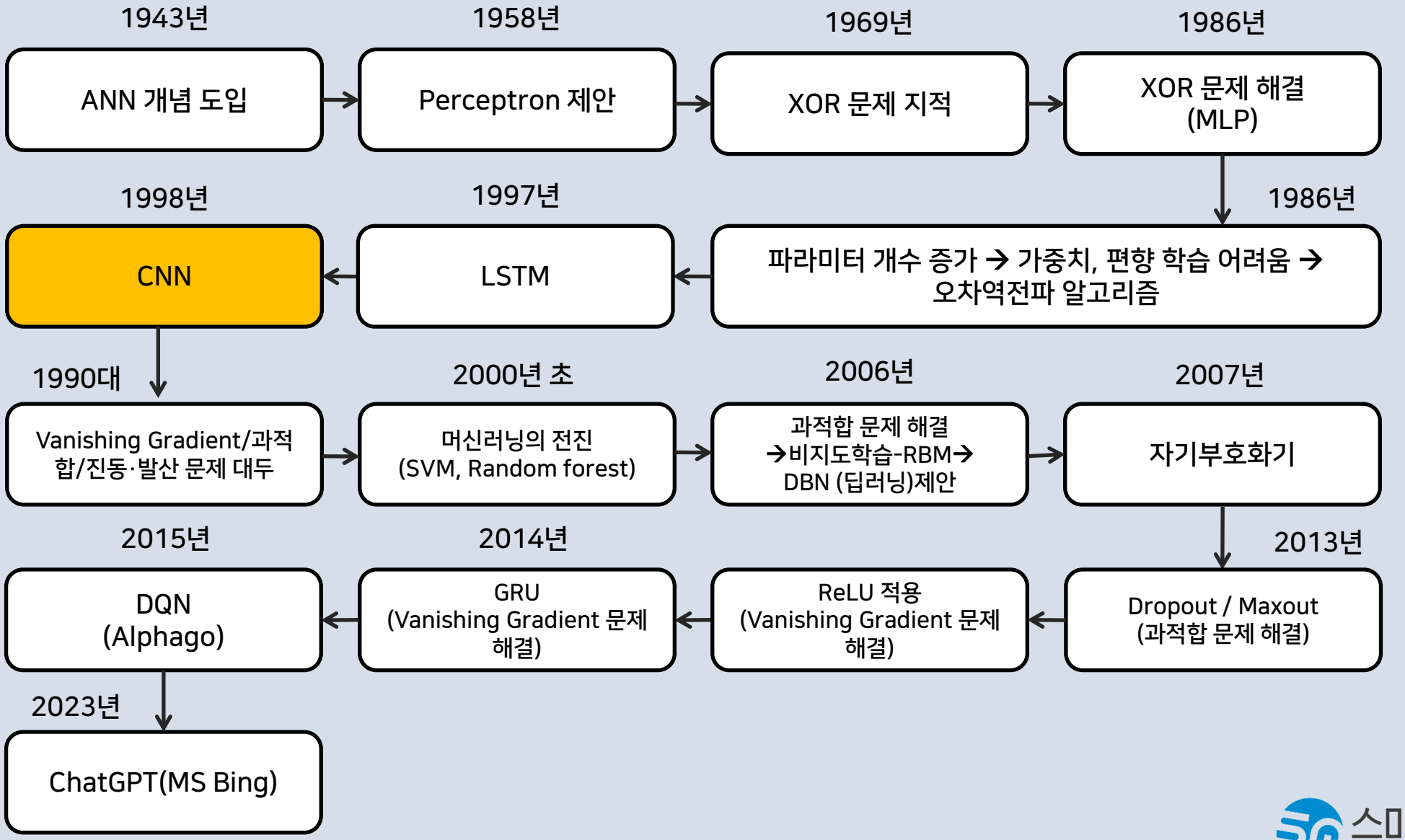
특징을 추출해보자!





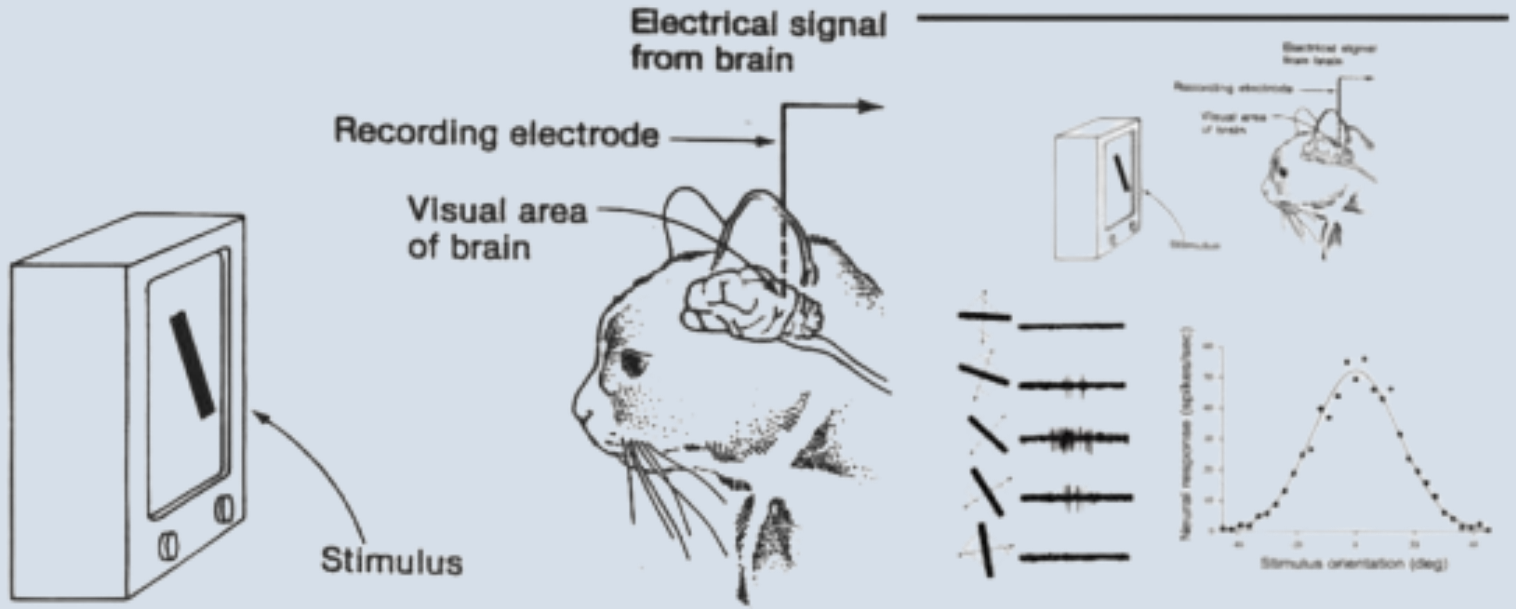
CNN(Convolutional Neural Network)

딥러닝 역사


1943년
ANN 개념 도입
1958년
Perceptron 제안
1969년
XOR 문제 지적
1986년
XOR 문제 해결 (MLP)
1986년
파라미터 개수 증가 → 가중치, 편향 학습 어려움 → 오차역전파 알고리즘
1997년
LSTM
1998년
CNN
1990대
Vanishing Gradient/과적합/진동·발산 문제 대두
2000년 초
머신러닝의 전진 (SVM, Random forest)
2006년
과적합 문제 해결 →비지도학습-RBM→ DBN (딥러닝)제안
2007년
자기부호화기
2013년
Dropout / Maxout (과적합 문제 해결)
2014년
ReLU 적용 (Vanishing Gradient 문제 해결)
GRU (Vanishing Gradient 문제 해결)
2015년
DQN (Alphago)
2023년
ChatGPT(MS Bing)

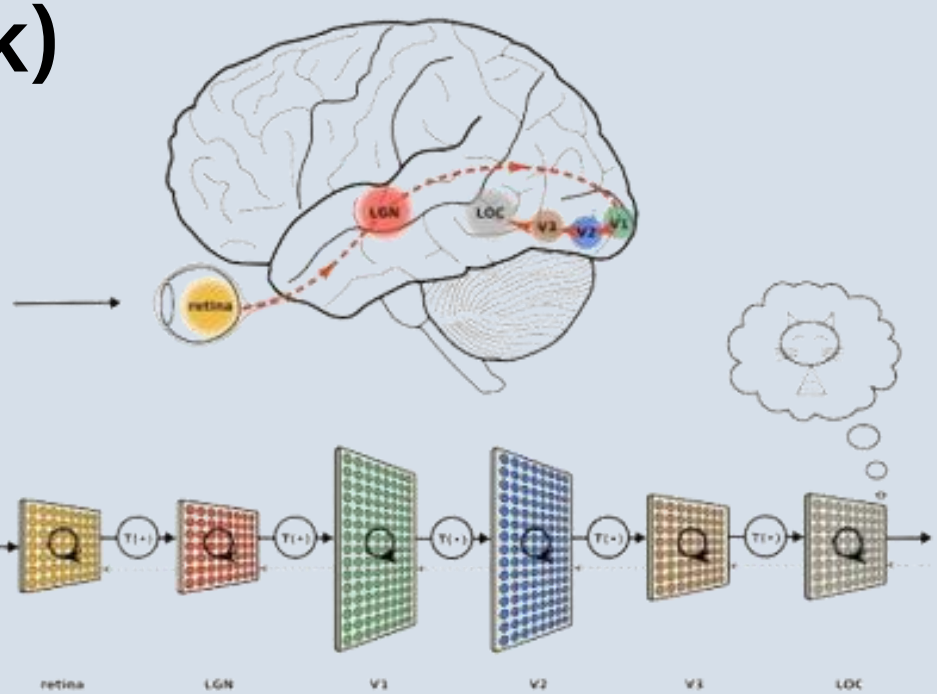
CNN(Convolutional Neural Network)

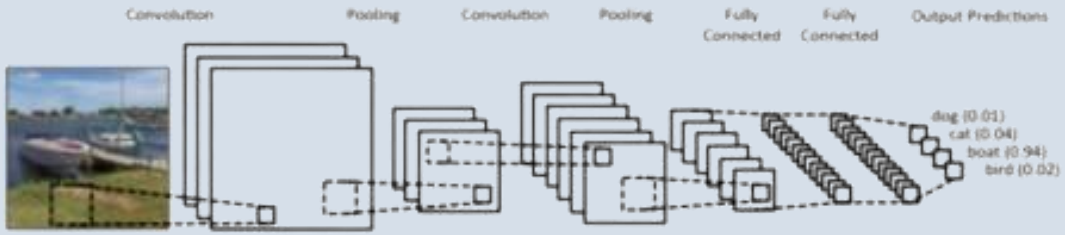
- 1998년 Yann Lecun 교수에 의해 1950년 대 수행했던 고양이의 뇌파 실험에 영감을 얻어 이미지 인식을 획기적으로 개선 할 수 있는 CNN 제안



CNN(Convolutional Neural Network)

- 고양이의 눈으로 보는 사물의 형태에 따라 뇌의 특정 영역(뉴런)만 활성화 된다는 결과를 기반으로 제안

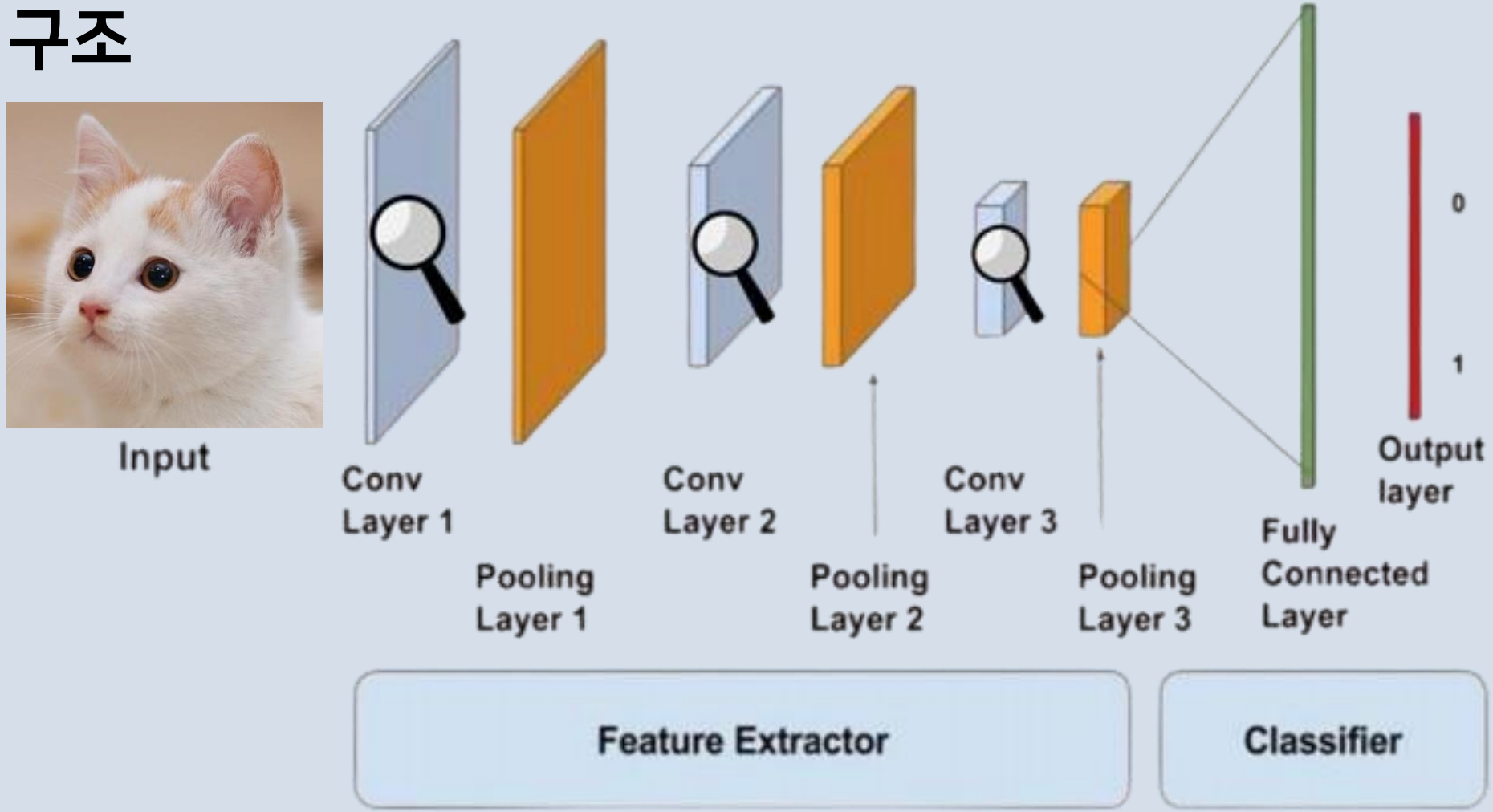




CNN(Convolutional Neural Network)

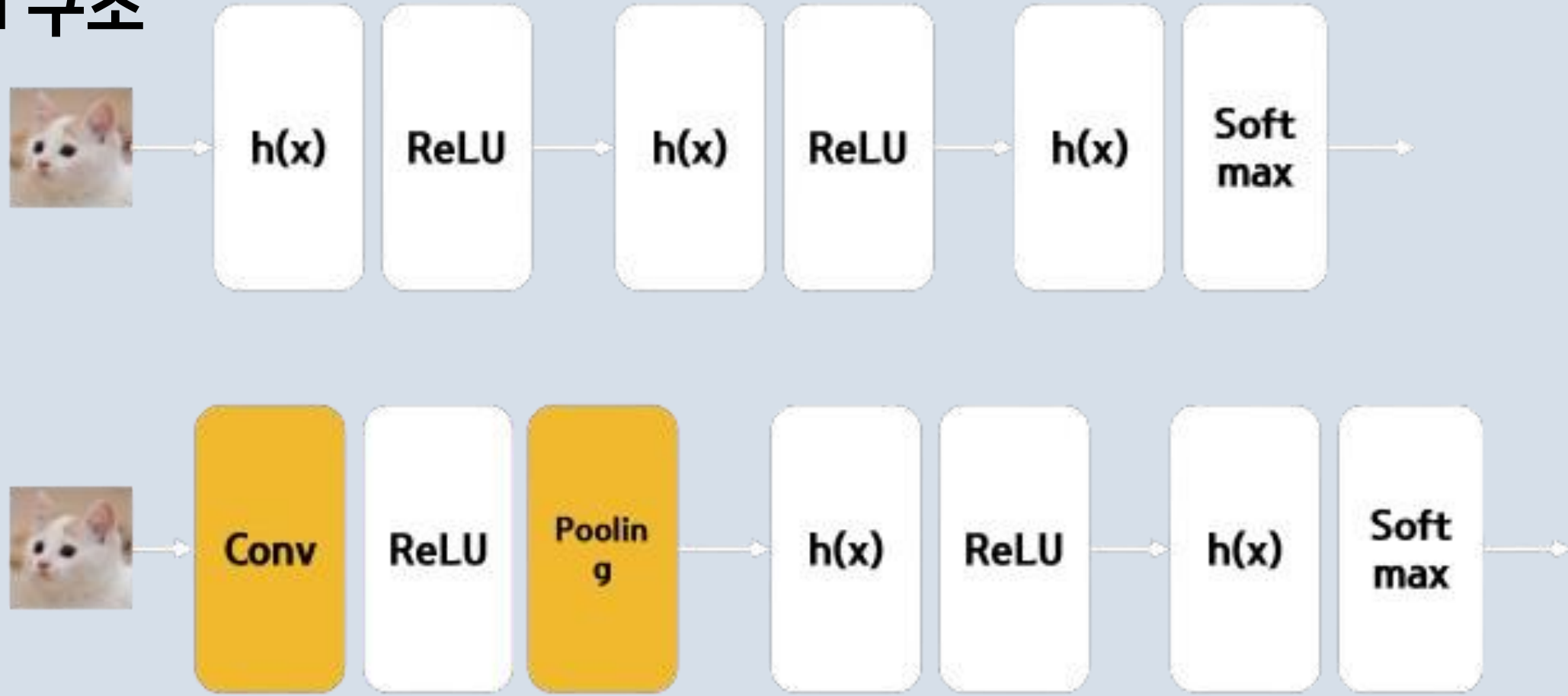
CNN(Convolutional Neural Network)

CNN 구조



CNN(Convolutional Neural Network)

CNN 구조

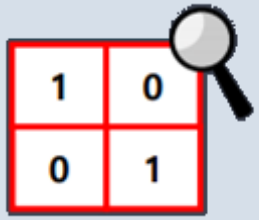


Convolutional(합성곱)

- CNN(합성곱)은 입력된 이미지에서 특징을 추출하기 위해 Filter(Kernel)를 도입하는 방법
- 이미지 전체를 한 번에 처리하지 않고, 작은 영역별로 중요한 특징을 추출하여 처리한다면, 훨씬 효과적일 것이라는 아이디어 제시

1	0	1	0
1	0	1	0
1	1	1	1
0	0	1	0

이미지



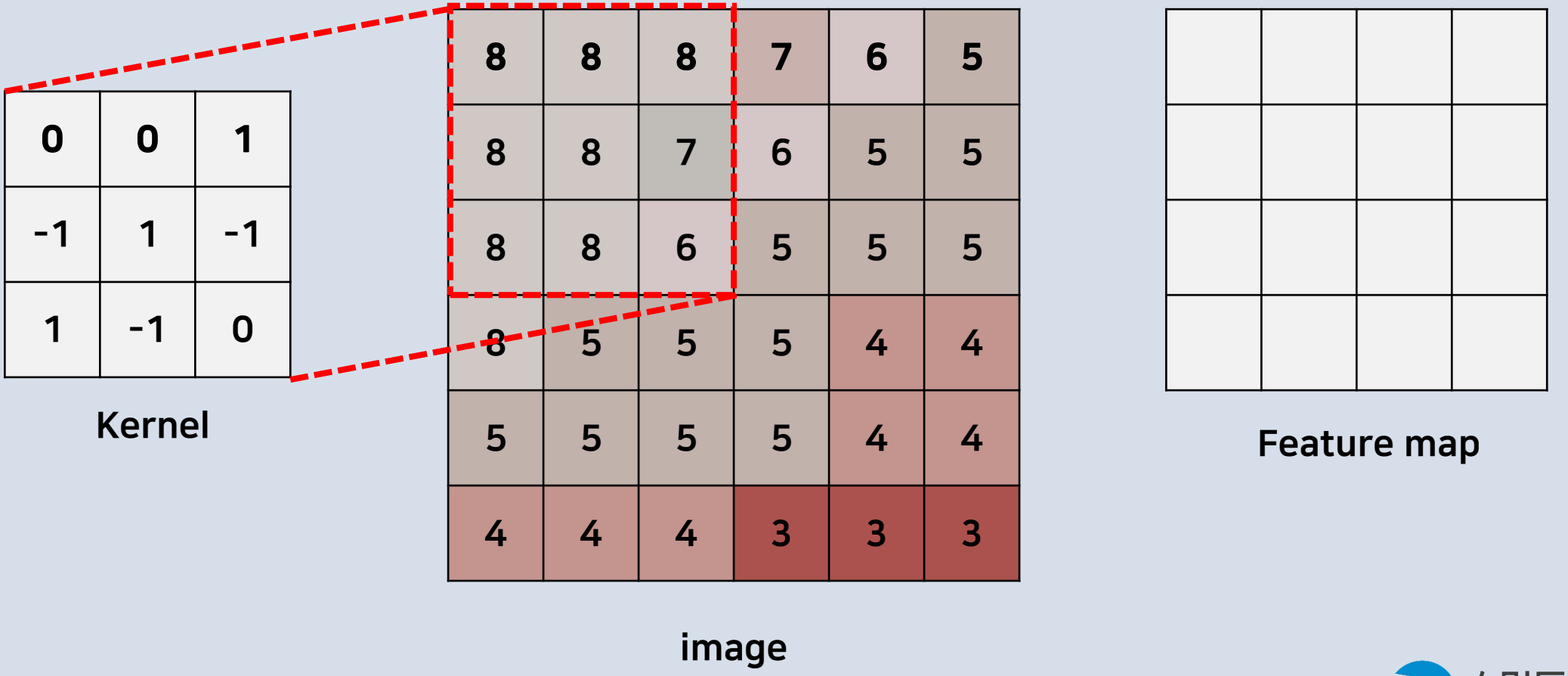
Kernel(Filter)

합성곱 연산

CNN(Convolutional Neural Network)

합성곱 연산

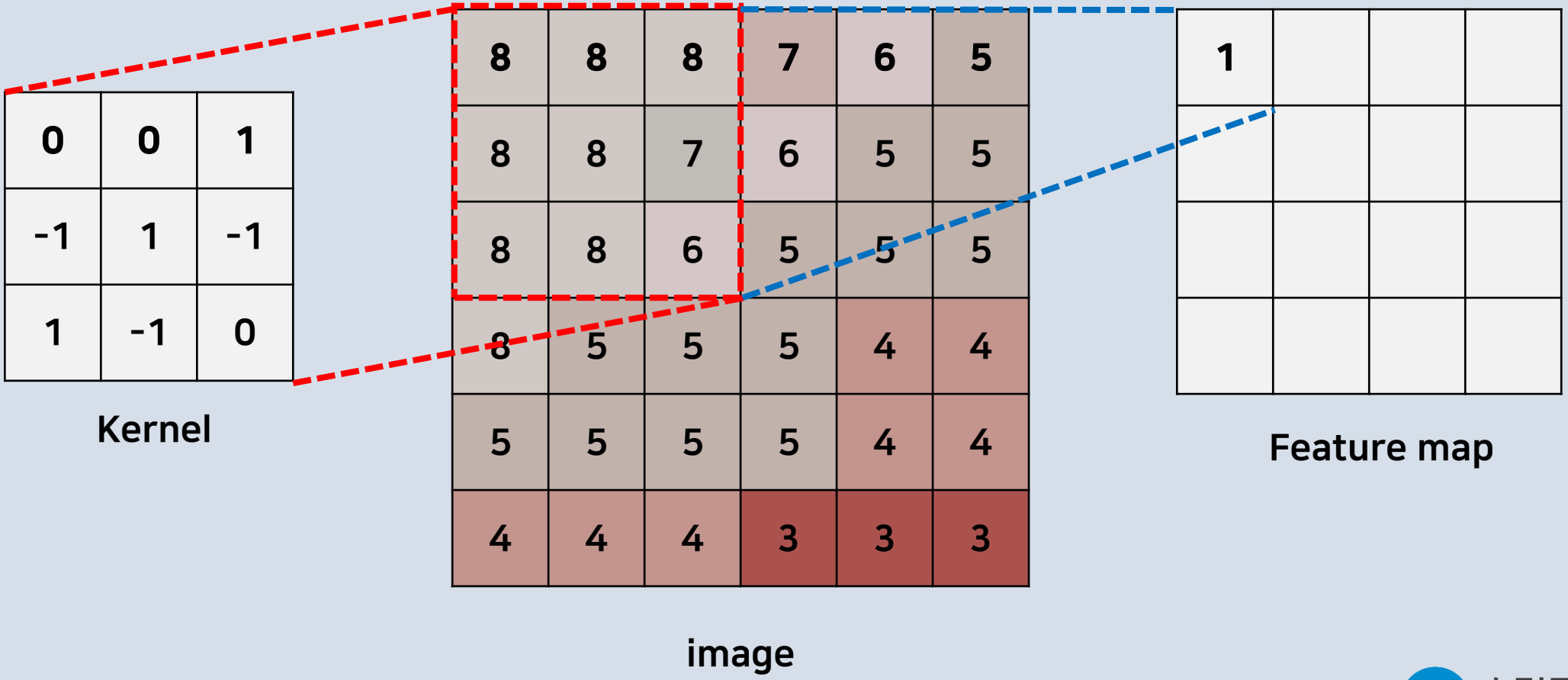
- 커널과 이미지의 각 픽셀별로 곱하여 합산하는 과정



CNN(Convolutional Neural Network)

합성곱 연산

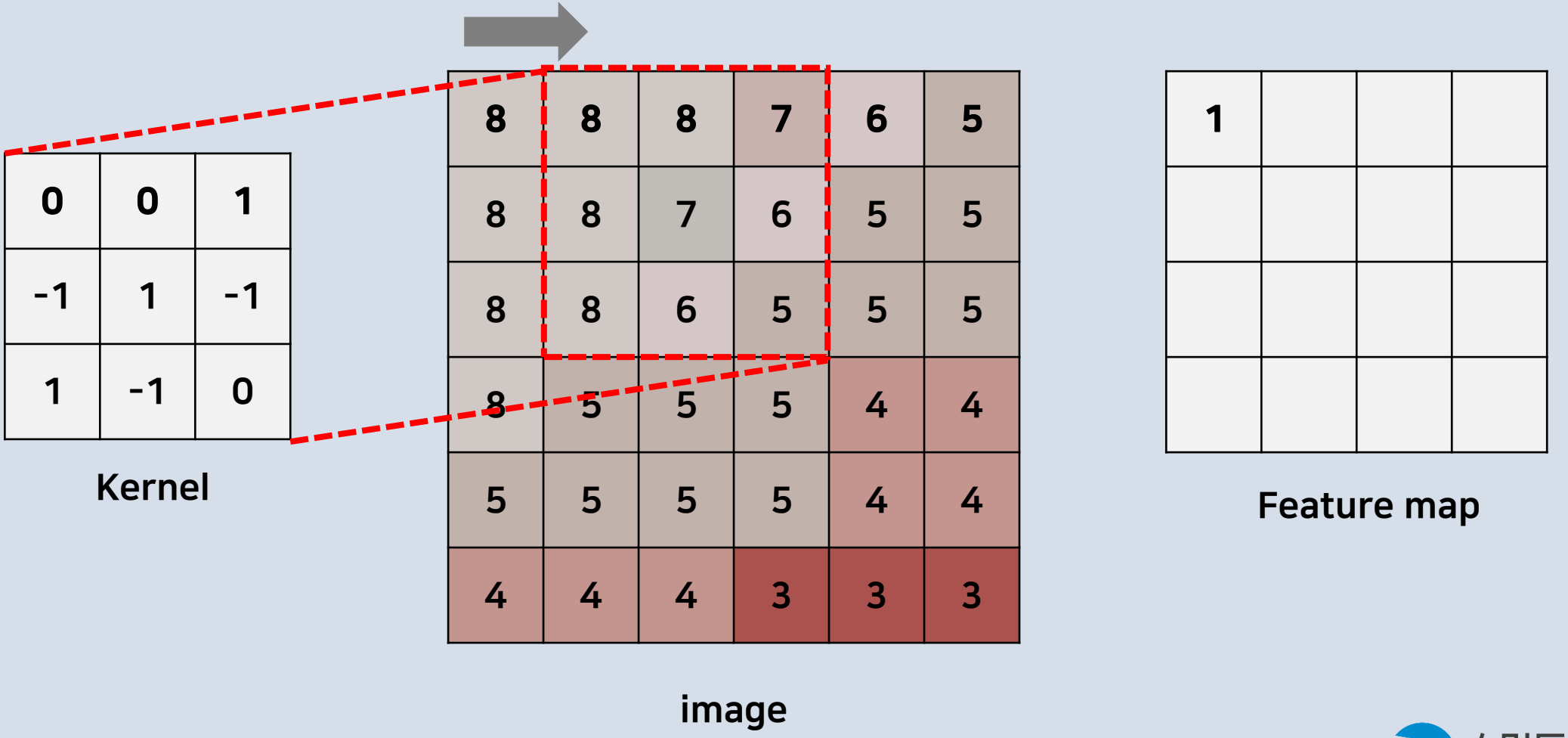
- 커널과 이미지의 각 픽셀별로 곱하여 합산하는 과정



CNN(Convolutional Neural Network)

합성곱 연산

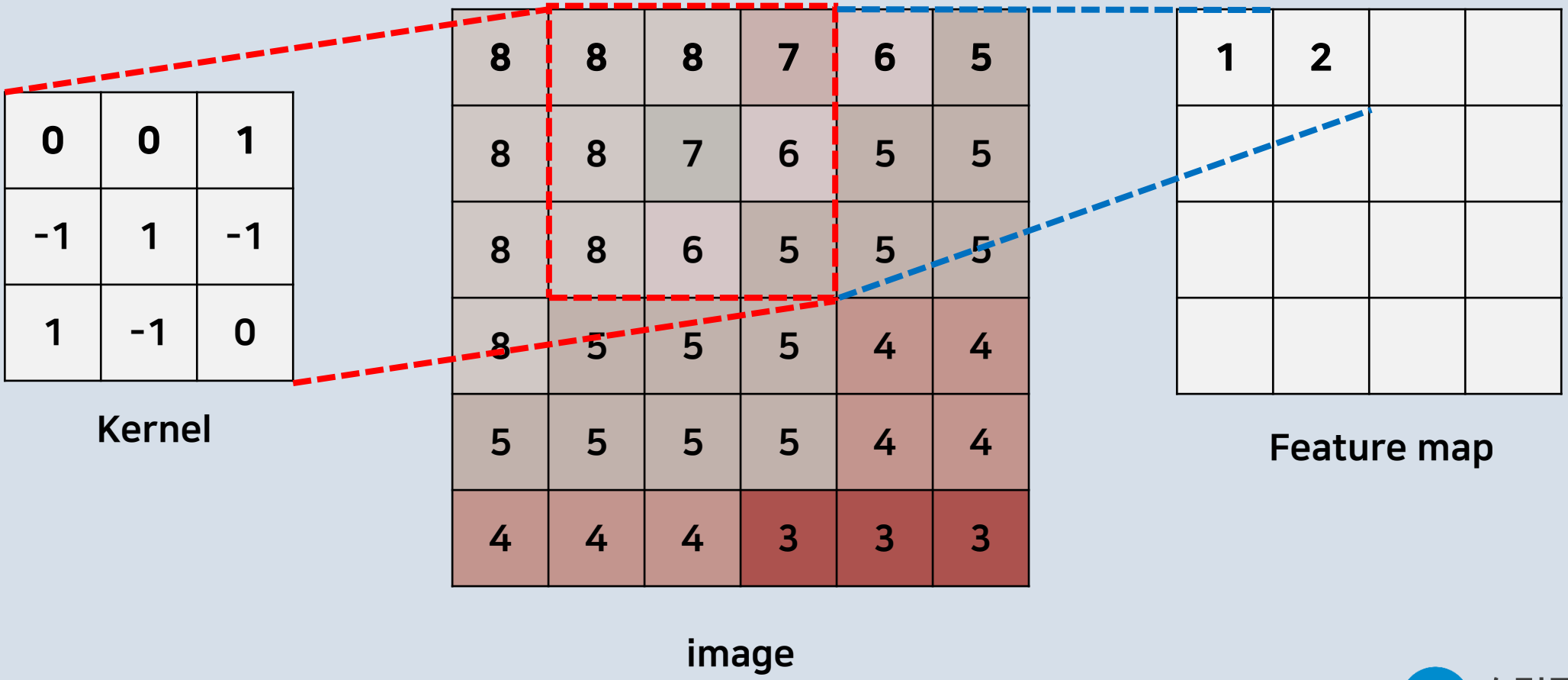
- 커널과 이미지의 각 픽셀별로 곱하여 합산하는 과정



CNN(Convolutional Neural Network)

합성곱 연산

- 커널과 이미지의 각 픽셀별로 곱하여 합산하는 과정



CNN(Convolutional Neural Network)

합성곱 연산

- 커널과 이미지의 각 픽셀별로 곱하여 합산하는 과정

0	0	1
-1	1	-1
1	-1	0

Kernel

8	8	8	7	6	5
8	8	7	6	5	5
8	8	6	5	5	5
8	5	5	5	4	4
5	5	5	5	4	4
4	4	4	3	3	3

image

1	2	1	

Feature map

CNN(Convolutional Neural Network)

합성곱 연산

- 커널과 이미지의 각 픽셀별로 곱하여 합산하는 과정

0	0	1
-1	1	-1
1	-1	0

Kernel

8	8	8	7	6	5
8	8	7	6	5	5
8	8	6	5	5	5
8	5	5	5	4	4
5	5	5	5	4	4
4	4	4	3	3	3

image

1	2	1	-1

Feature map

CNN(Convolutional Neural Network)

합성곱 연산

- 커널과 이미지의 각 픽셀별로 곱하여 합산하는 과정

0	0	1
-1	1	-1
1	-1	0

Kernel

8	8	8	7	6	5
8	8	7	6	5	5
8	8	6	5	5	5
8	5	5	5	4	4
5	5	5	5	4	4
4	4	4	3	3	3

image

1	2	1	-1
4			

Feature map

CNN(Convolutional Neural Network)

합성곱 연산

- 커널과 이미지의 각 픽셀별로 곱하여 합산하는 과정

0	0	1
-1	1	-1
1	-1	0

Kernel

8	8	8	7	6	5
8	8	7	6	5	5
8	8	6	5	5	5
8	5	5	5	4	4
5	5	5	5	4	4
4	4	4	3	3	3

image

1	2	1	-1
4	-1		

Feature map

CNN(Convolutional Neural Network)

합성곱 연산

- 커널과 이미지의 각 픽셀별로 곱하여 합산하는 과정

0	0	1
-1	1	-1
1	-1	0

Kernel

8	8	8	7	6	5
8	8	7	6	5	5
8	8	6	5	5	5
8	5	5	5	4	4
5	5	5	5	4	4
4	4	4	3	3	3

image

1	2	1	-1
4	-1	-1	

Feature map

CNN(Convolutional Neural Network)

합성곱 연산

- 커널과 이미지의 각 픽셀별로 곱하여 합산하는 과정

0	0	1
-1	1	-1
1	-1	0

Kernel

8	8	8	7	6	5
8	8	7	6	5	5
8	8	6	5	5	5
8	5	5	5	4	4
5	5	5	5	4	4
4	4	4	3	3	3

image

1	2	1	-1
4	-1	-1	1

Feature map

CNN(Convolutional Neural Network)

합성곱 연산

- 커널과 이미지의 각 픽셀별로 곱하여 합산하는 과정

0	0	1
-1	1	-1
1	-1	0

Kernel

8	8	8	7	6	5
8	8	7	6	5	5
8	8	6	5	5	5
8	5	5	5	4	4
5	5	5	5	4	4
4	4	4	3	3	3

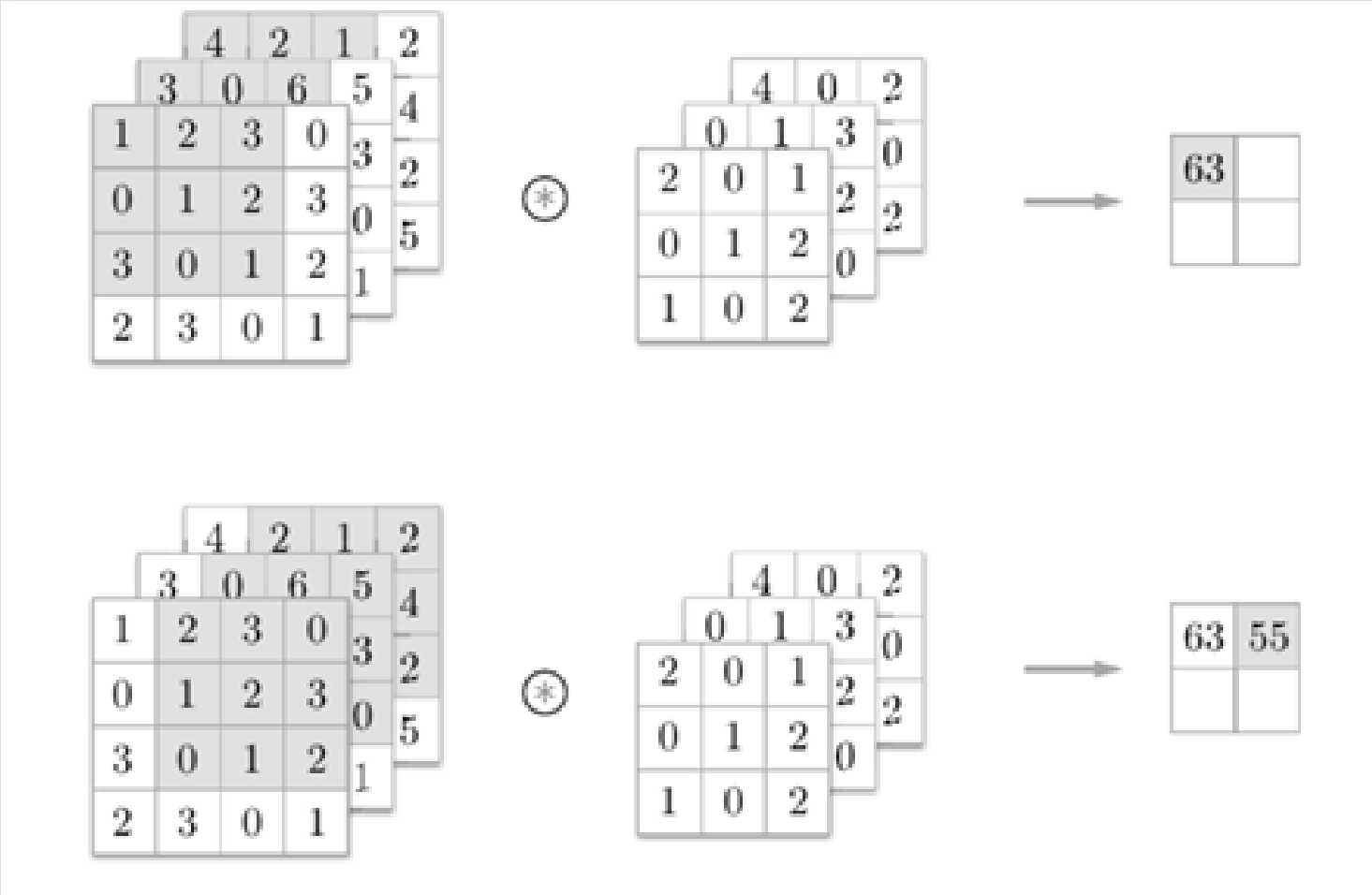
image

1	2	1	-1
4	-1	-1	1
-2	0	1	1
0	0	1	-1

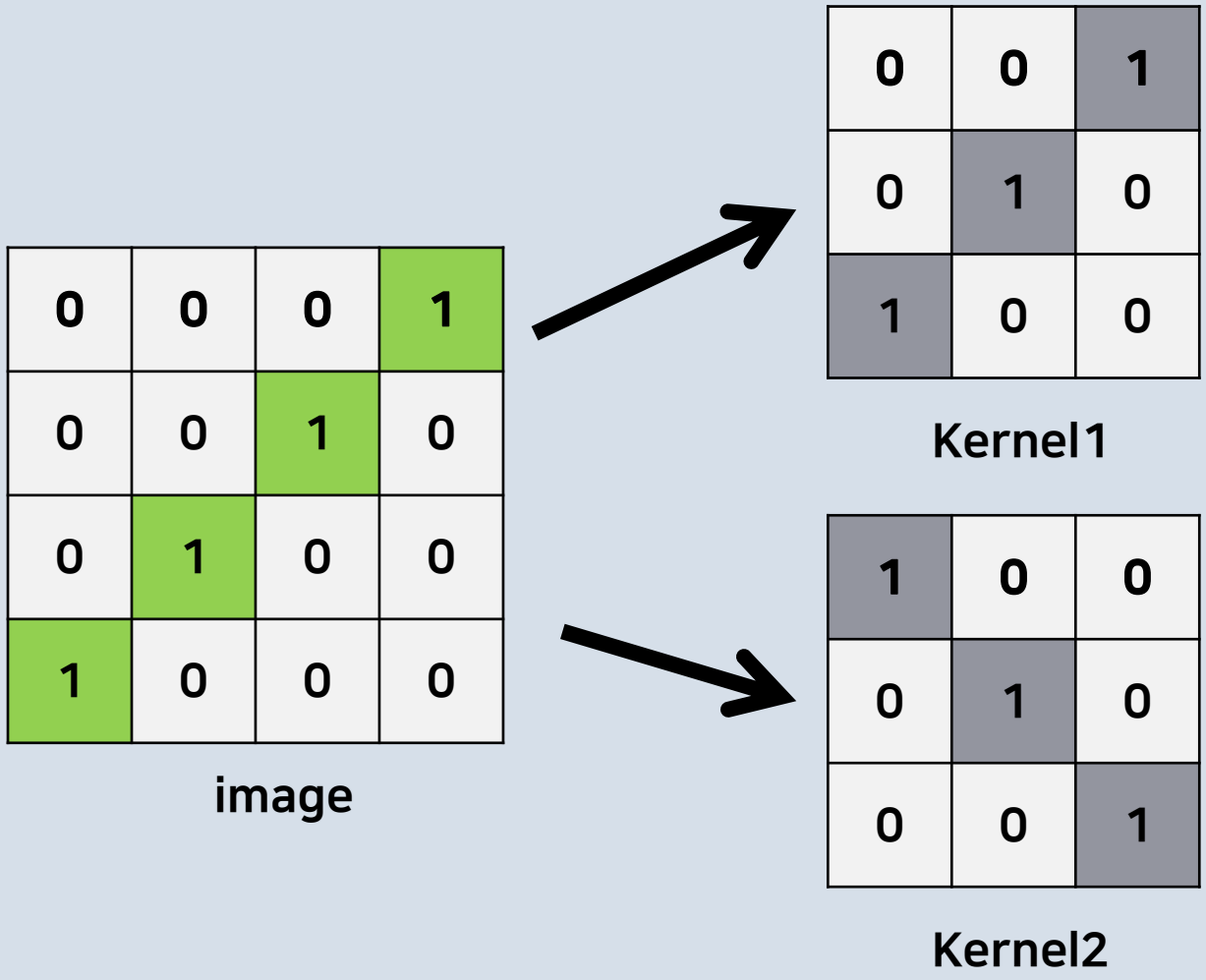
Feature map

CNN(Convolutional Neural Network)

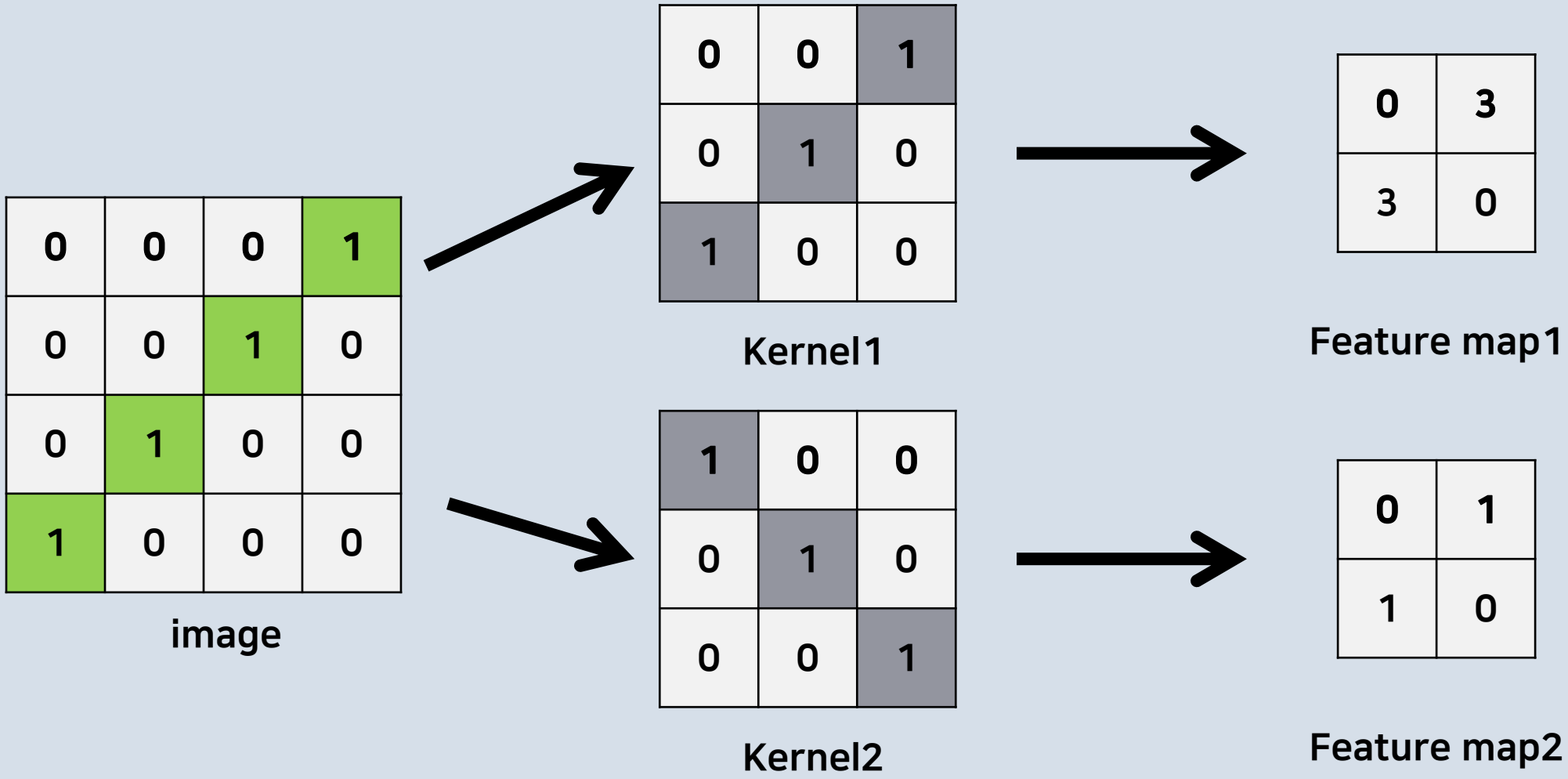
- **3차원인 경우**



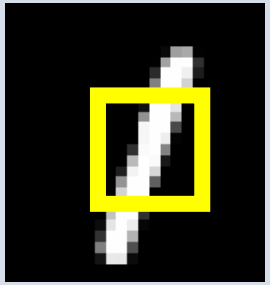
CNN(Convolutional Neural Network)

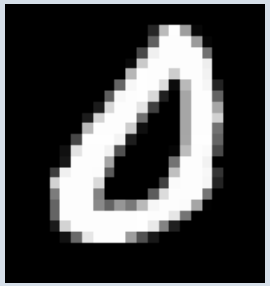


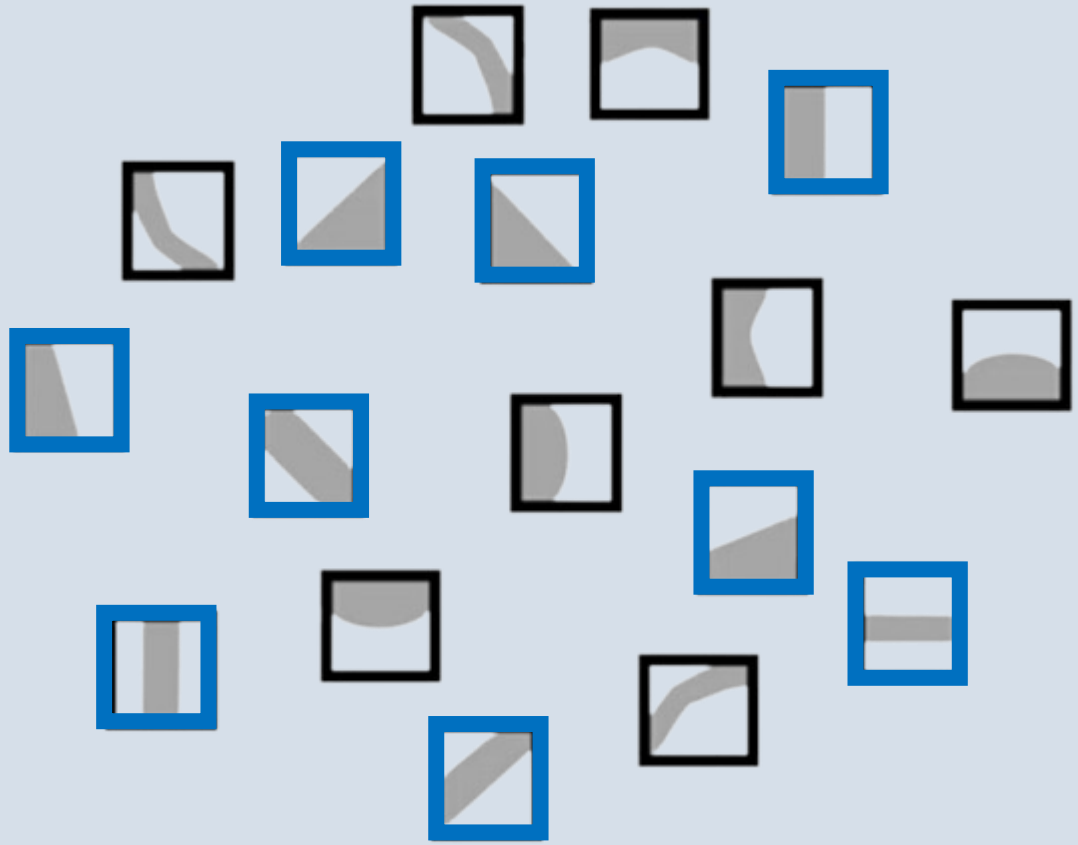
CNN(Convolutional Neural Network)



CNN(Convolutional Neural Network)

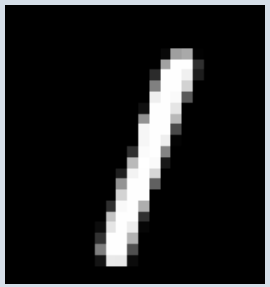


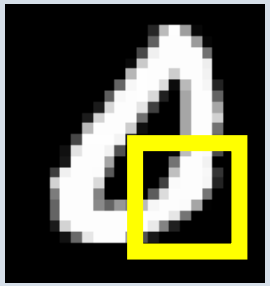


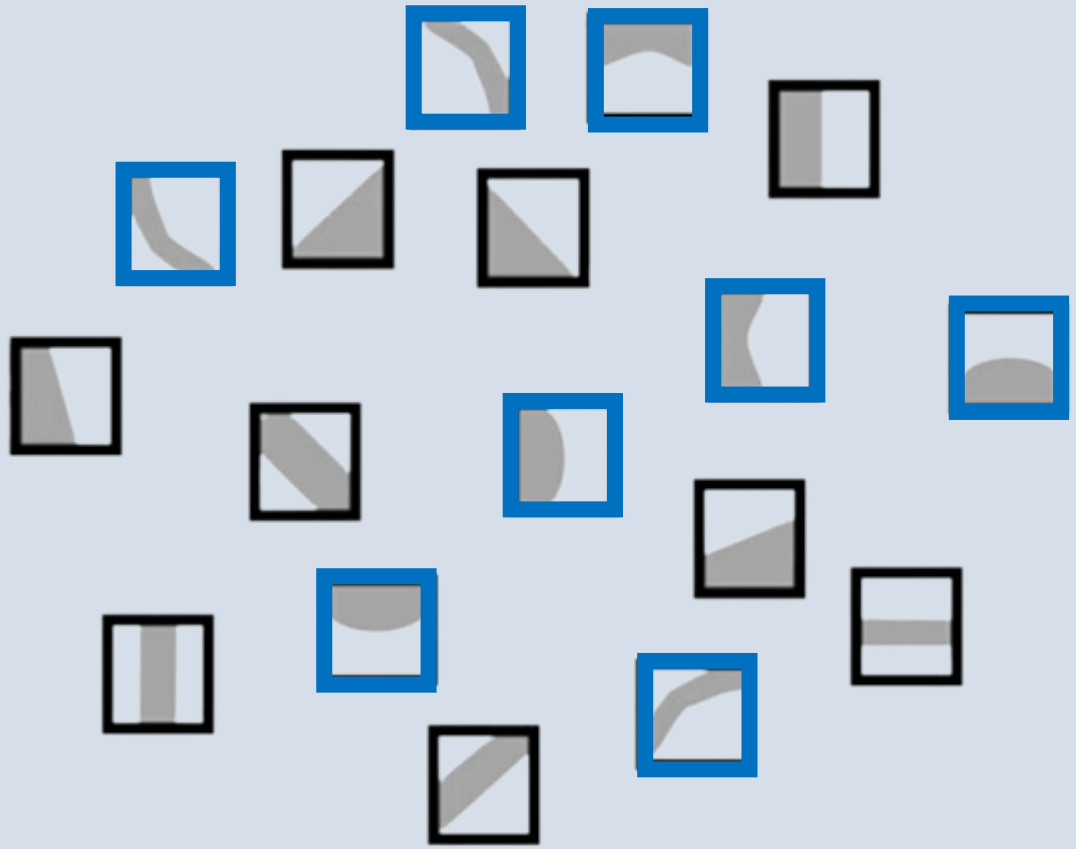


Kernel

CNN(Convolutional Neural Network)

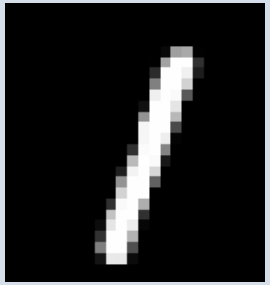


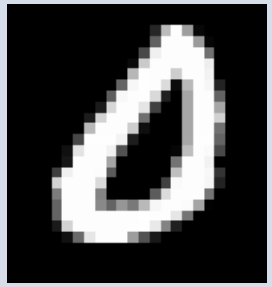


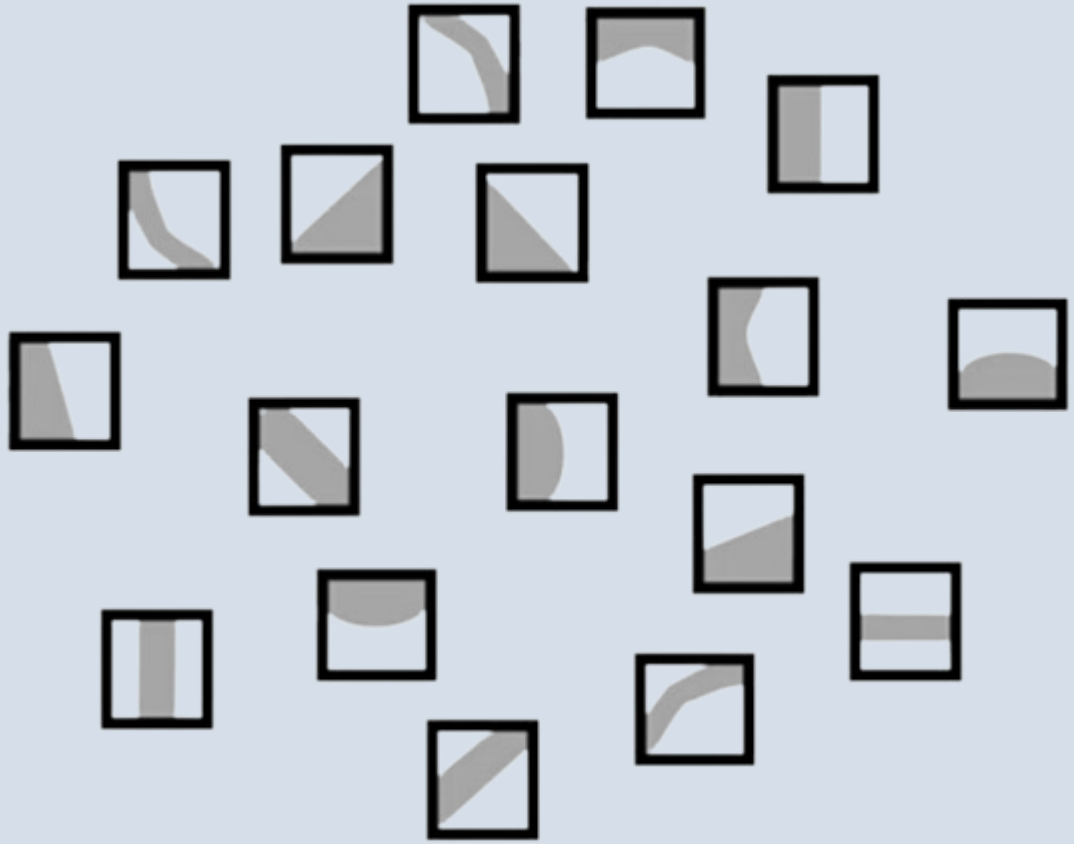


Kernel

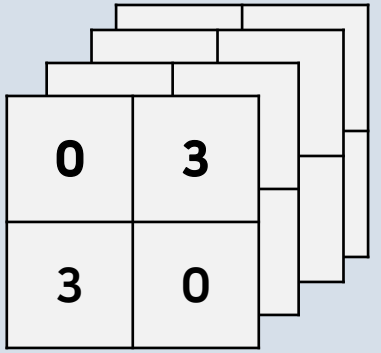
CNN(Convolutional Neural Network)





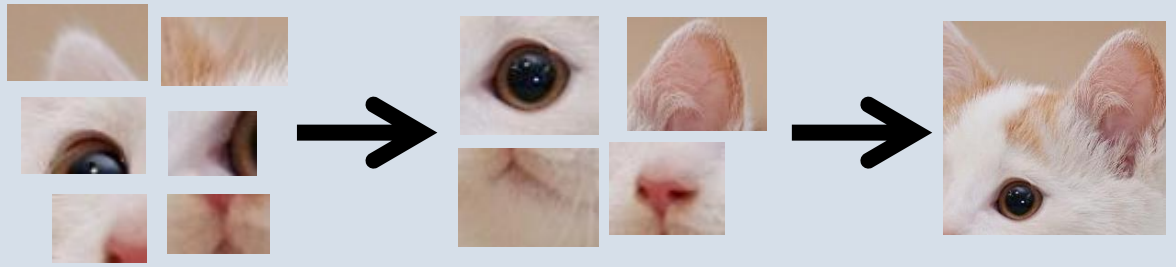


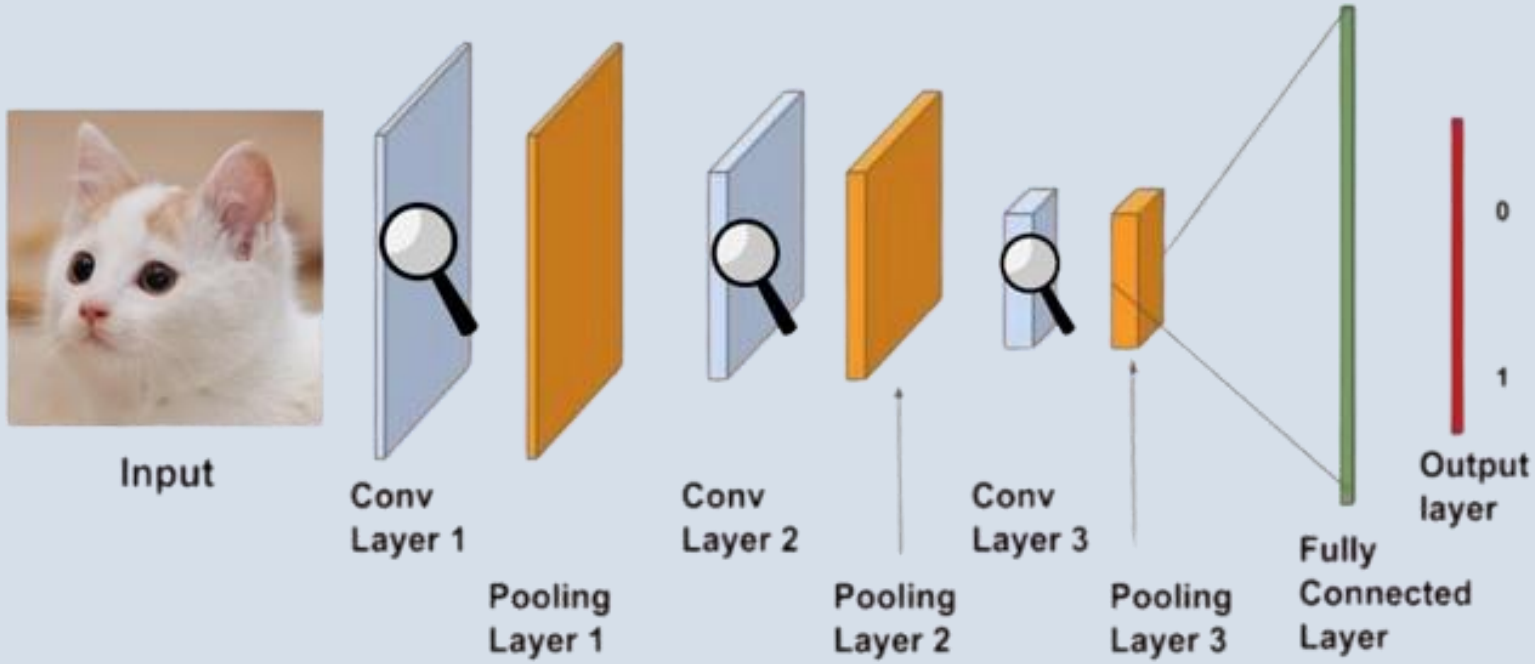
Kernel



Feature map

CNN(Convolutional Neural Network)

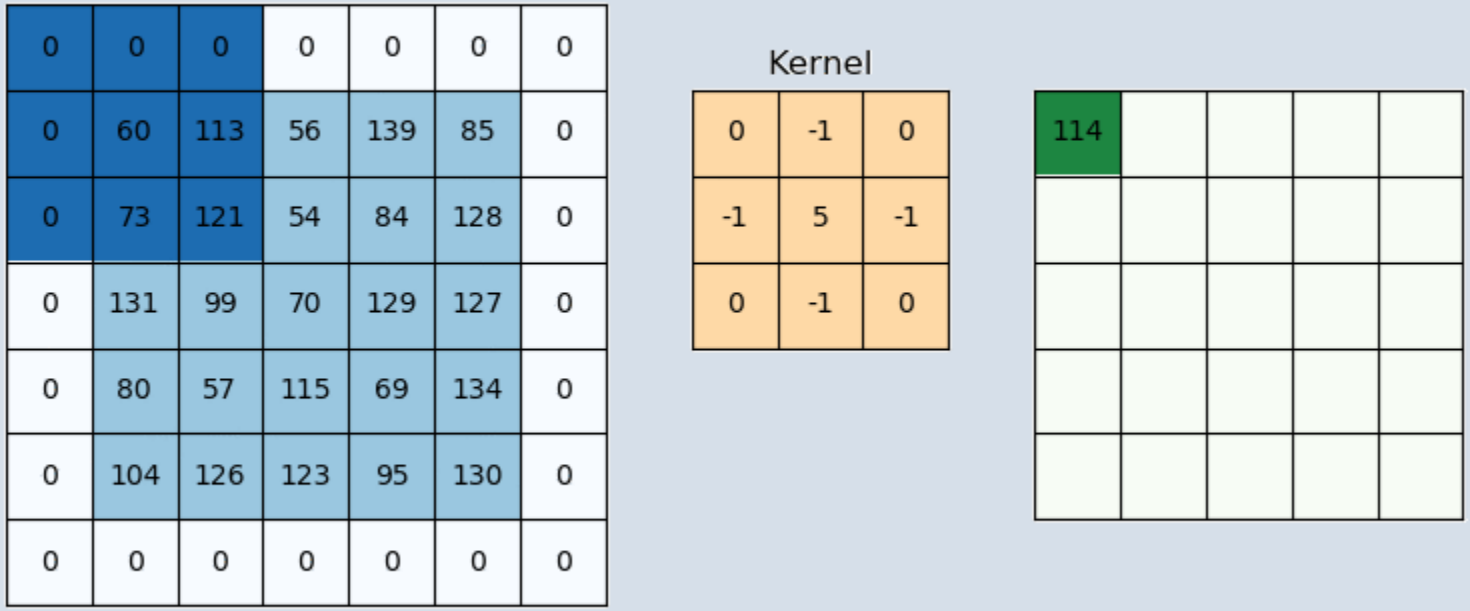




CNN(Convolutional Neural Network)

Padding(패딩)

- 합성곱 연산에 의해 가장자리 부분의 데이터가 부족해서, 입력과 출력의 크기가 달라짐
- 이를 보정하기 위해 **가장자리를 채워 넣는 것**을 패딩(Padding)이라고 함

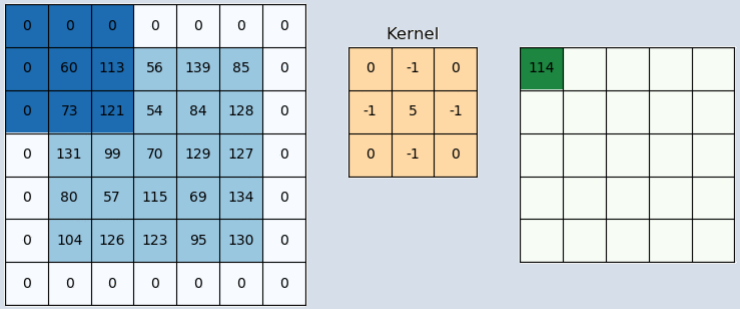


출처 : https://www.analyticsvidhya.com/blog/2022/03/basics-of-cnn-in-deep-learning/

Padding(패딩)

- 합성곱 연산에 의해 가장자리 부분의 데이터가 부족해서, 입력과 출력의 크기가 달라짐
- 이를 보정하기 위해 가장자리를 채워 넣는 것을 패딩(Padding)이라고 함
- 크기 축소 방지, 정보 손실 방지 등의 효과
- Conv2D 계층에서는 padding 인자를 사용
  - ✓ valid : 패딩을 허용하지 않음
  - ✓ same : 입력과 출력의 크기가 같게끔

    Zero Padding 적용(0으로 채우기)



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

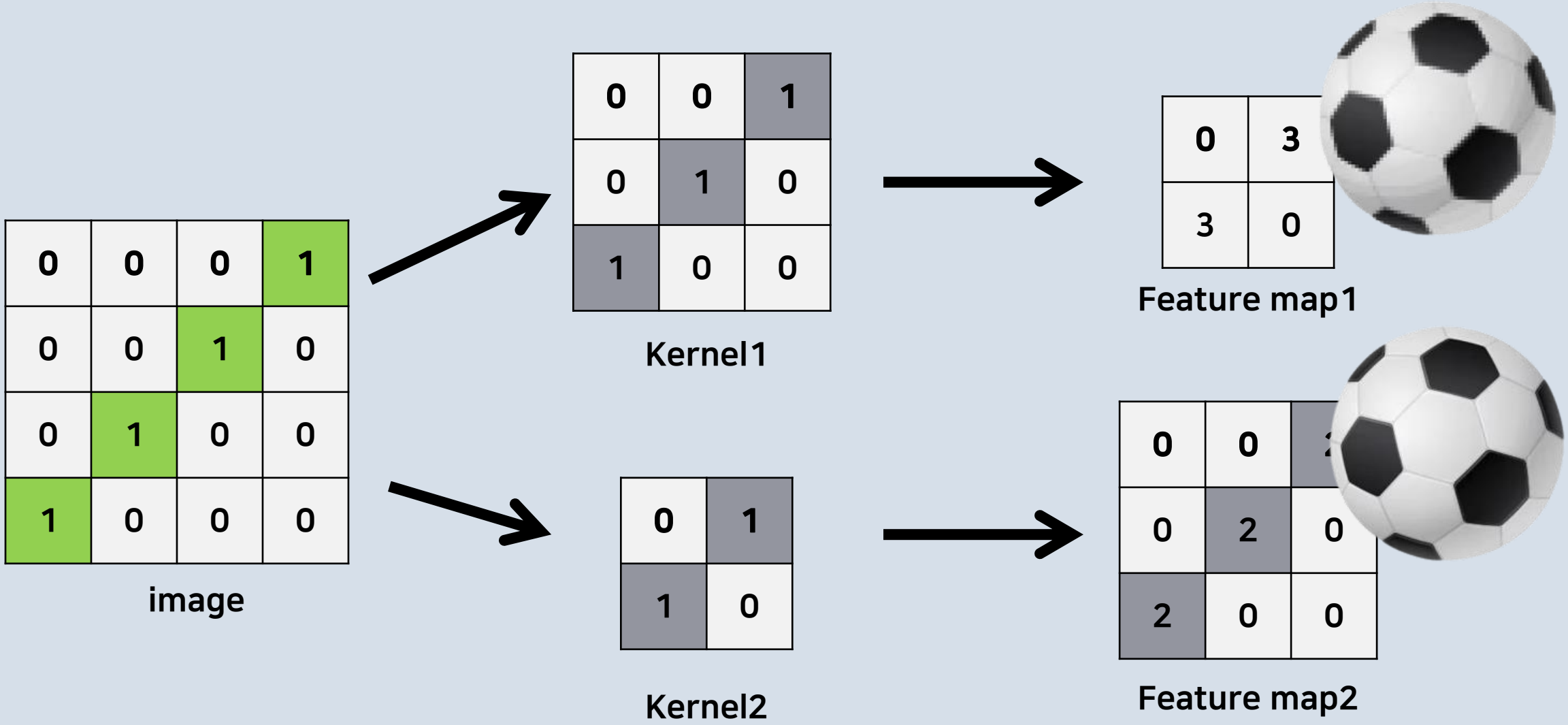
축소 샘플링

- 합성곱을 수행한 결과 신호를 다음 계층으로 전달할 때, 모든 정보를 전달하지 않고, 일부만 샘플링하여 넘겨주는 작업을 축소 샘플링(subsampling)이라고 함

- 크게 2가지 방법 -> Stride, Pooling

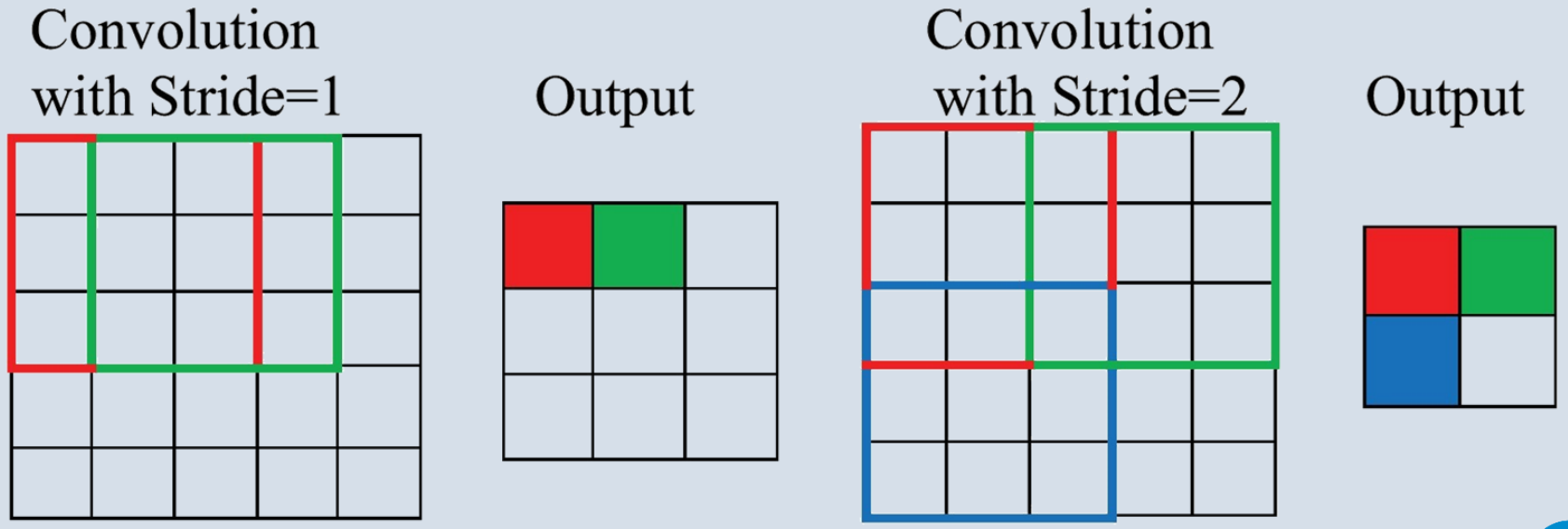
- 목적 : 좀 더 가치있는 정보만을 다음 단계로 넘겨주는 것을 목표
  ✓ 연산량 감소, 모델의 복잡도를 낮추어 과적합 방지

CNN(Convolutional Neural Network)



Stride(보폭)

- 합성곱 연산 시 Kernel이 이동하는 간격을 의미
- 2*2(Stride 2)의 경우 오른쪽으로 2픽셀씩, 아래쪽으로 2픽셀씩 이동하며 출력맵 생성

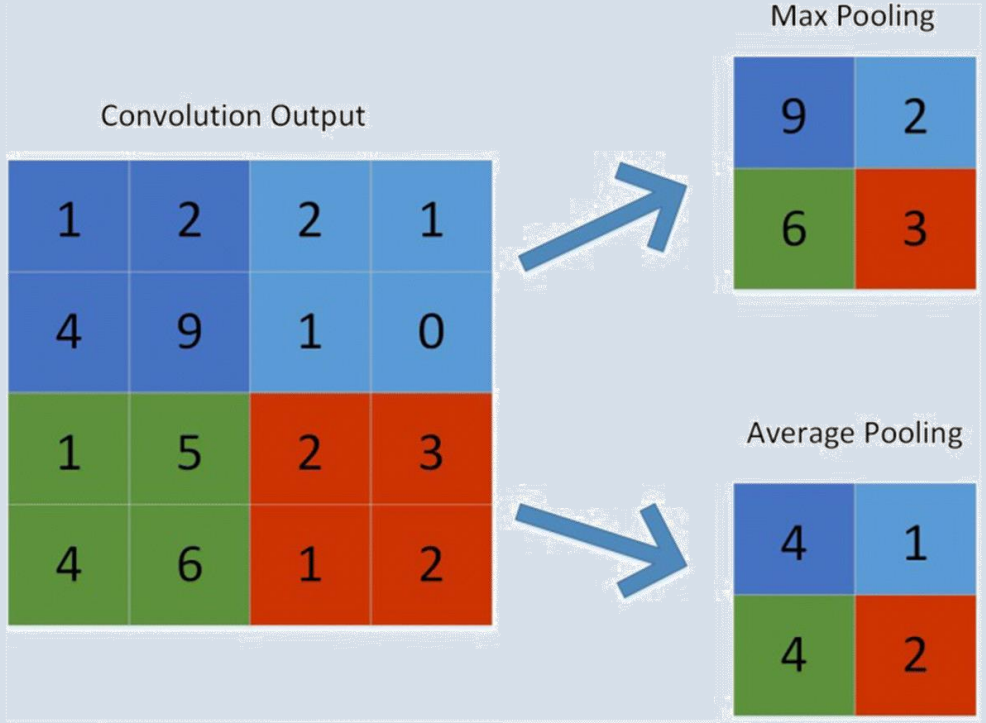


출처 : https://www.analyticsvidhya.com/blog/2022/03/basics-of-cnn-in-deep-learning/

CNN(Convolutional Neural Network)

Pooling(풀링)

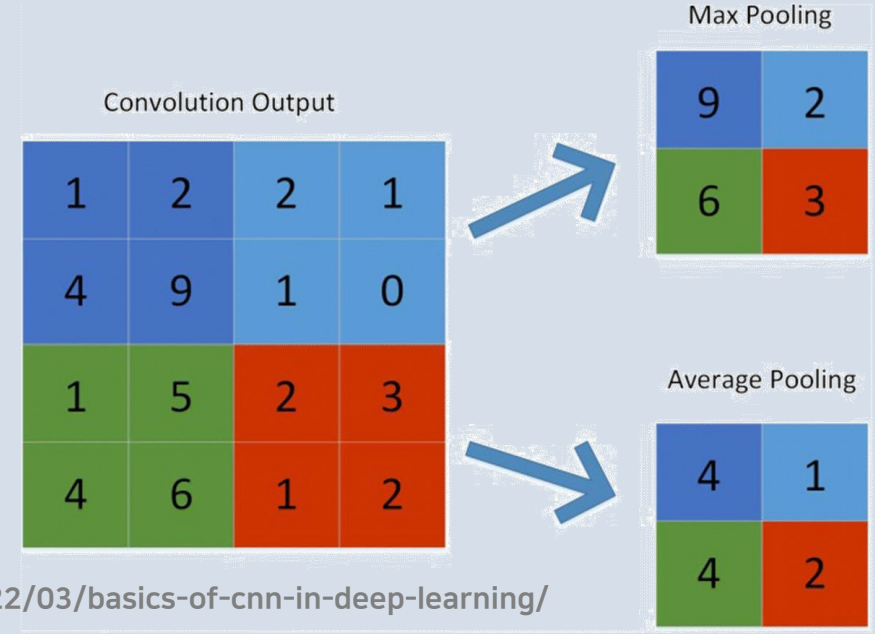
- 합성곱 연산 결과를 다음 계층으로 모두 넘기지 않고, 일정 영역 내에서 대표 값을 추출하여 다운 샘플링하는 연산



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

Pooling(풀링)

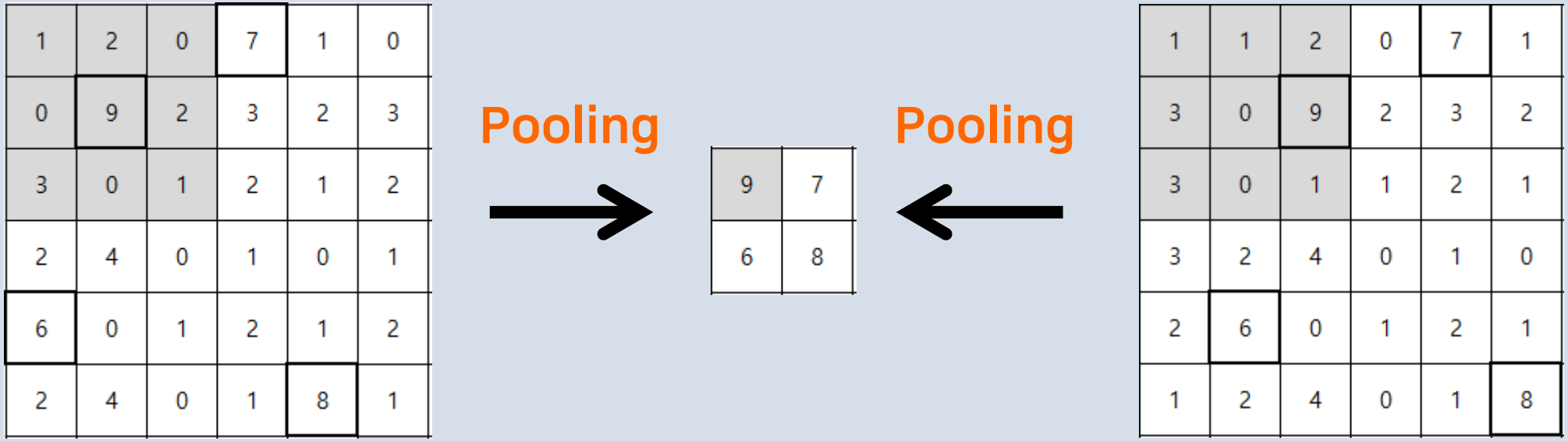
- Pooling Window : Pooling 층에서 사용되는 작은 영역
- Max Pooling : Pooling Window 내 최대값 선택하여 출력맵 생성
- Average Pooling : Pooling Window 내 평균을 계산하여 출력맵 생성



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

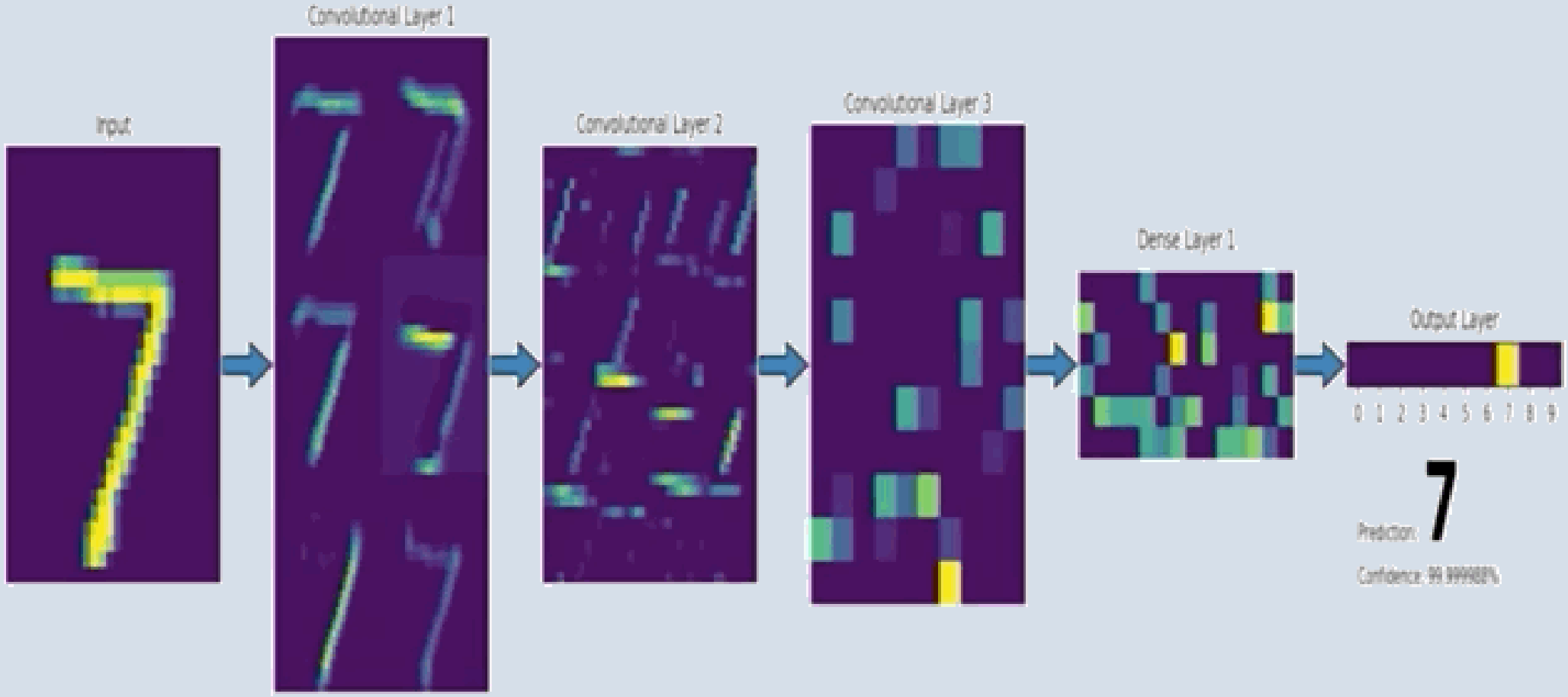
CNN(Convolutional Neural Network)

Pooling(풀링)



- 입력의 변화에 영향을 적게 받음

CNN(Convolutional Neural Network)

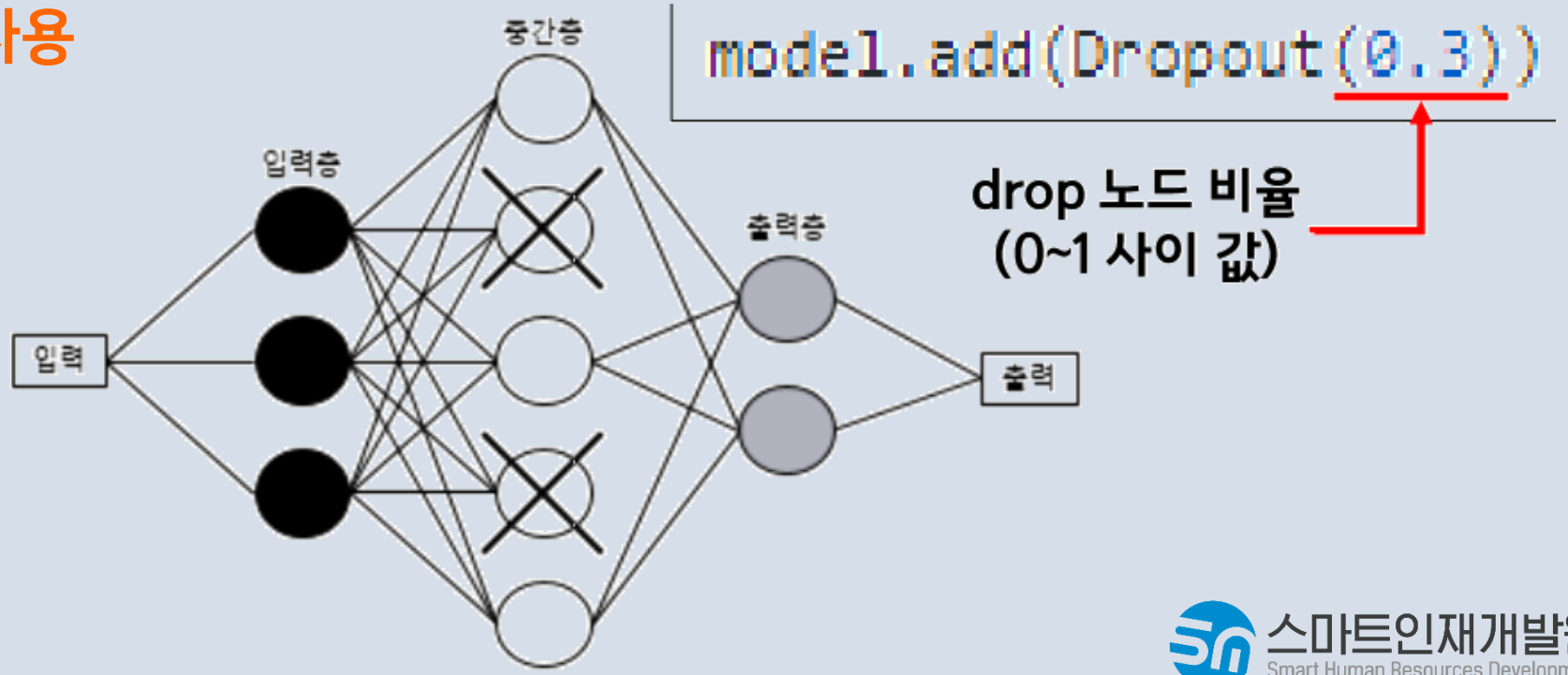


과적합 피하기

- 학습 조기 중단(Early Sopping)
- 규제(Requlariz ation)
- 드롭 아웃(Dropout)
- 데이터 증강(Data Augmentation)

드롭 아웃(Dropout)

- 학습 중 일부 뉴런을 무작위로 비활성화하여 모델의 일반화 성능을 향상시키는 기법
- Epochs 동안의 다양한 드롭 아웃을 적용한 상태에서 학습을 수행
- 예측 시에 모든 뉴런 사용



데이터 증강(Data Augmentation)

- 이미지 회전, 이동, 확대, 축소 등을 통해 개별 원본 이미지를 변형하여 훈련 데이터를 늘리는 방법
- 과대적합 원인 중 하나는 훈련 데이터의 부족일 수 있음
- 훈련 데이터가 충분히 많다면, 과대적합을 줄여볼 수 있음

데이터 증강(Data Augmentation)

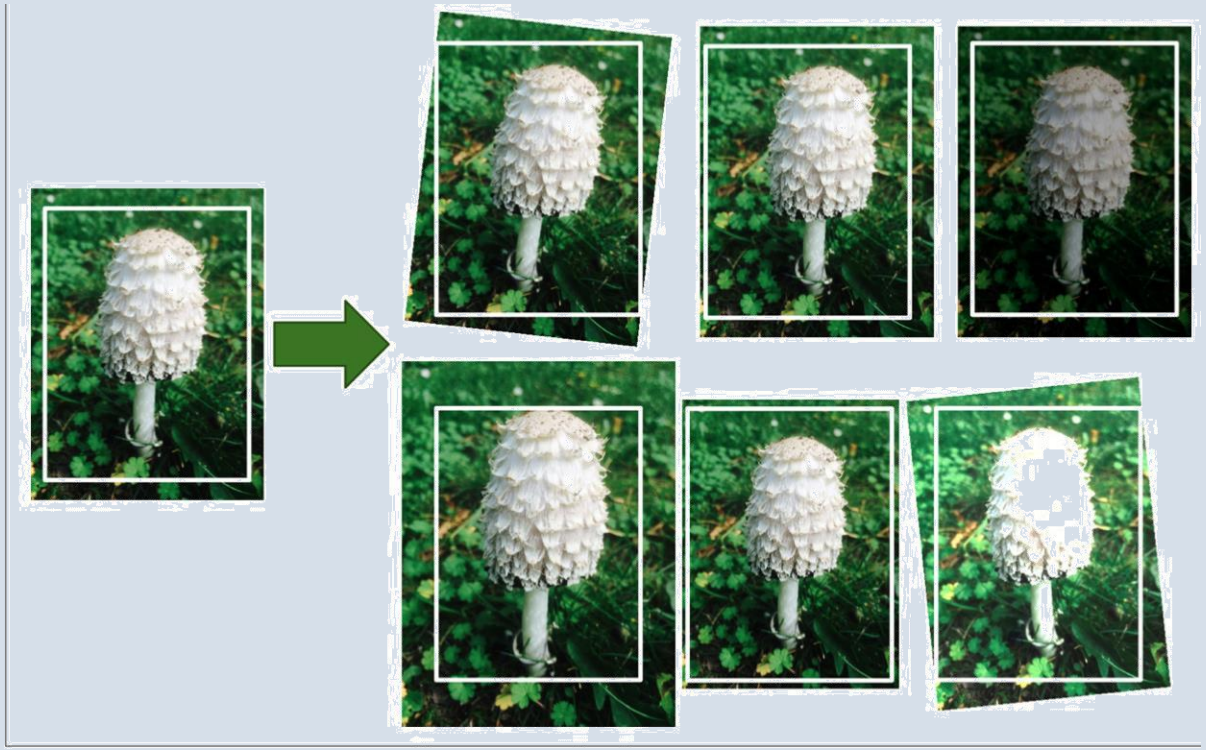
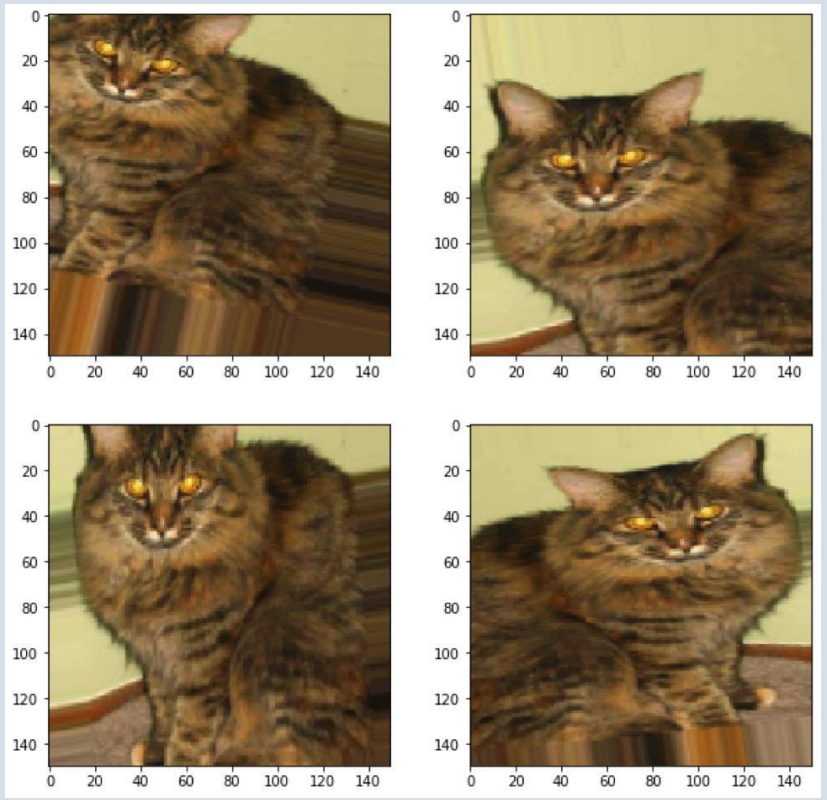


Figure 11-10. Generating new training instances from existing ones



기타 성능 개선

- 배치 정규화 (Batch Normalization)
  - 활성화 함수 앞 또는 뒤에서 평균 0, 분산 1로 정규화하는 방법
  - 각 층에서 값들이 적당한 분포를 갖도록 조정하는 것

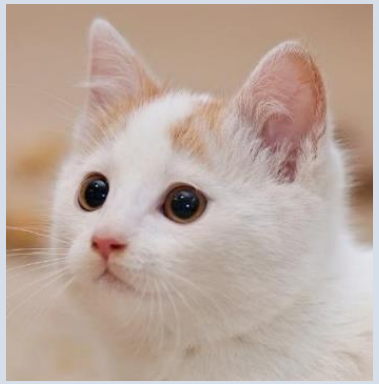
전이학습(Transfer Learning)

- 다른 데이터 셋을 사용하여 이미 학습한 모델을 유사한 다른 데이터를 인식하는데 사용하는 기법
- 특히 새로 훈련 시킬 데이터가 충분히 확보되지 못 한 경우에 학습 효율을 높임
- 사전학습 모델을 이용하는 방법
  - 특성 추출(Feature Extraction) 방식
  - 미세조정(Fine-tuning) 방식

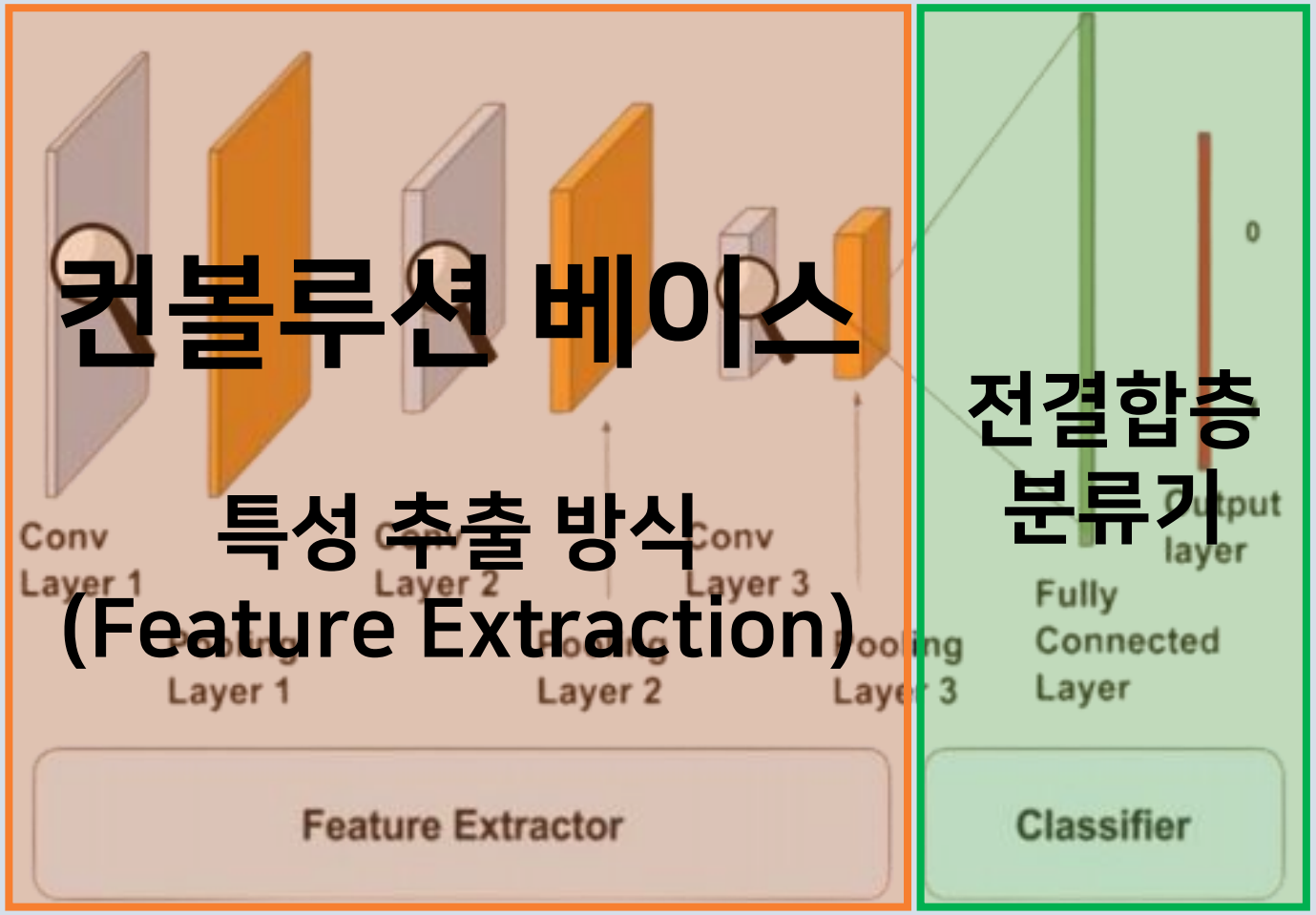
전이학습(Transfer Learning)

전이학습(Transfer Learning)

- 특성 추출(Feature Extraction)



Input



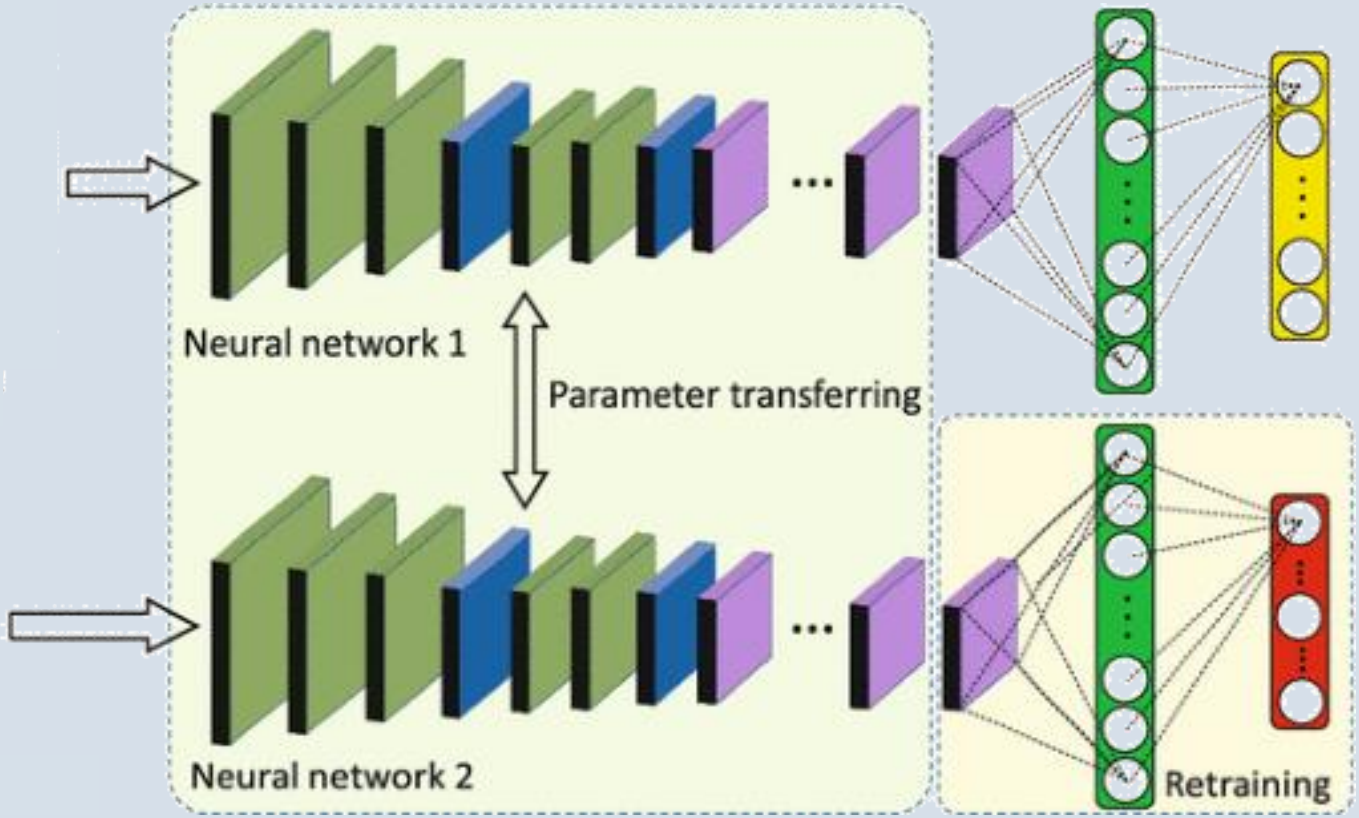
전이학습(Transfer Learning)

- 특성 추출(Feature Extraction)

- 컨볼루션 베이스 부분만 재사용 하는 이유
  - 상당히 일반적인 학습 정보를 포함하고 있기 때문
- 앞 단의 계층 : Edge, Color, Texture 등 * 대부분의 이미지에서 공통적으로 존재하는 아주 기본적인 특징들
- 뒷 부분의 깊은 계층 : 고양이 귀, 강아지 귀 등* 추상적인 정보 (다양한 이미지에 존재하지 않음)
- 컨볼루션의 계층별 정보 수준을 고려하여, 계층 선택에 대한 다양한 전략 가능

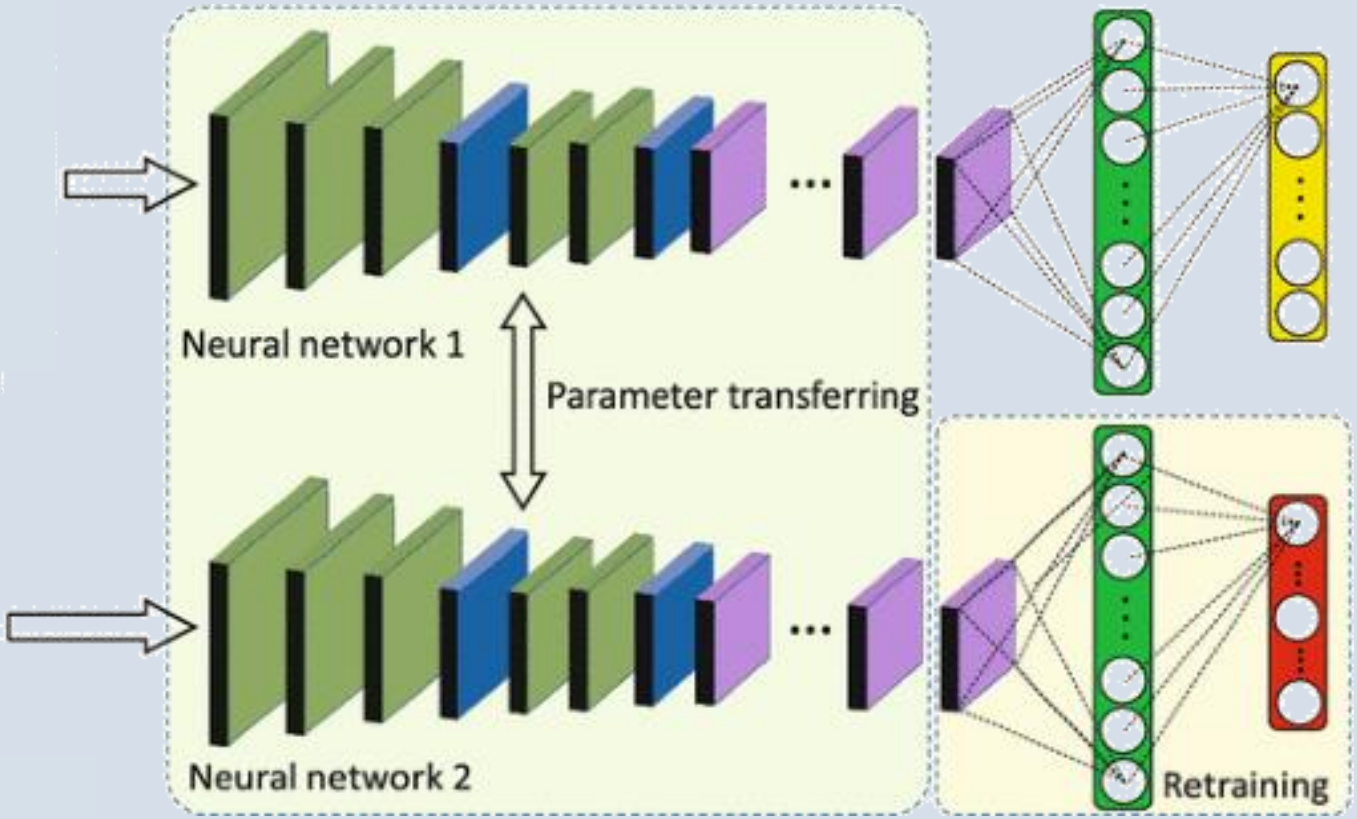
전이학습(Transfer Learning)

- 특성 추출(Feature Extraction)

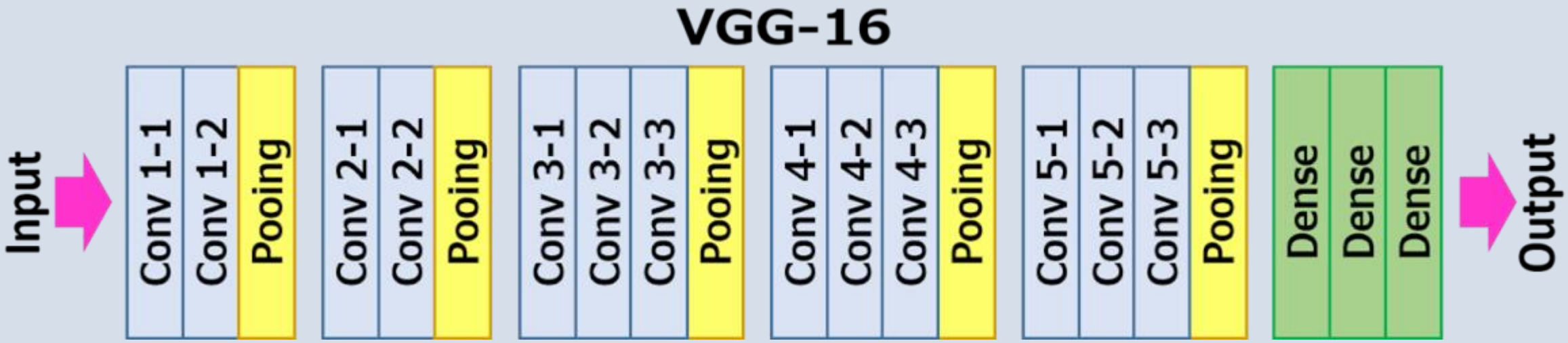


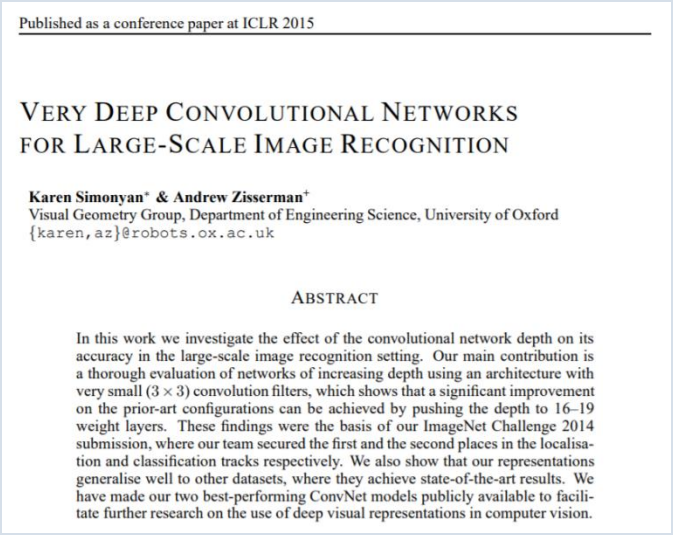
전이학습(Transfer Learning)

- 특성 추출(Feature Extraction)



전이학습(Transfer Learning) – 사전 학습 모델 소개





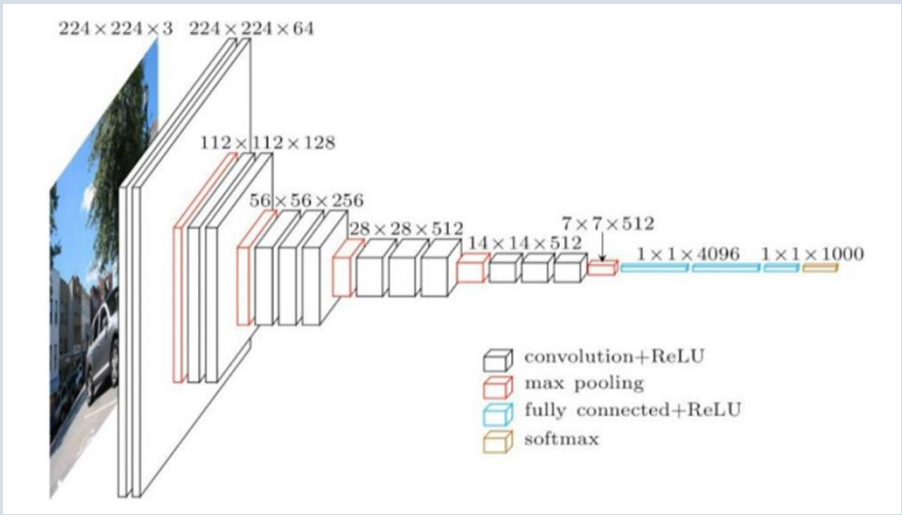
Published as a conference paper at ICLR 2015

VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

Karen Simonyan* **& Andrew Zisserman+**
Visual Geometry Group, Department of Engineering Science, University of Oxford
{karen,az}@robots.ox.ac.uk

ABSTRACT

In this work we investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. Our main contribution is a thorough evaluation of networks of increasing depth using an architecture with very small (3 × 3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16–19 weight layers. These findings were the basis of our ImageNet Challenge 2014 submission, where our team secured the first and the second places in the localisation and classification tracks respectively. We also show that our representations generalise well to other datasets, where they achieve state-of-the-art results. We have made our two best-performing ConvNet models publicly available to facilitate further research on the use of deep visual representations in computer vision.



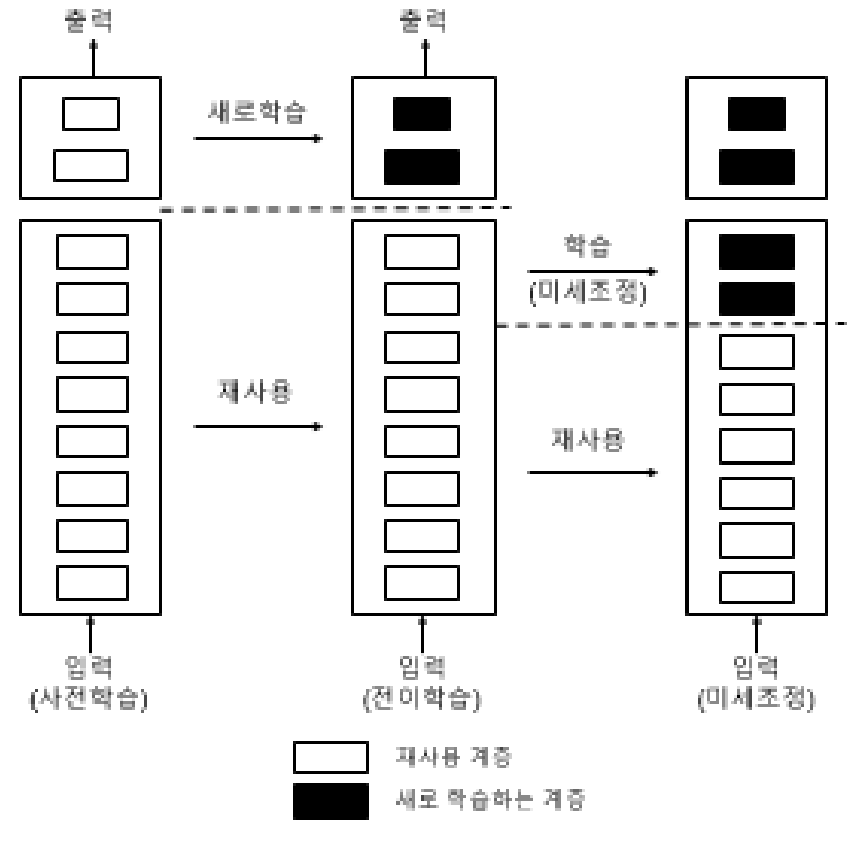
출처 : https:// arxiv.org/abs/1409.1556

전이학습(Transfer Learning) – 사전 학습 모델 소개

Model Name	Key Features	Image Input Size	Number of Parameters	Paper Reference
VGG16	Deep and simple architecture, 16 layers	224x224	~138 million	Very Deep Convolutional Networks for Large-Scale Image Recognition (2014)
VGG19	Extension of VGG16, 19 layers	224x224	~144 million	Very Deep Convolutional Networks for Large-Scale Image Recognition (2014)
ResNet50	Uses residual blocks, efficient in deep networks	224x224	~25.6 million	Deep Residual Learning for Image Recognition (2015)
ResNet101	Deeper residual network	224x224	~44.7 million	Deep Residual Learning for Image Recognition (2015)
ResNet152	Deepest residual network	224x224	~60.2 million	Deep Residual Learning for Image Recognition (2015)
InceptionV3	Applies convolutions of various kernel sizes, high efficiency	299x299	~23.9 million	Rethinking the Inception Architecture for Computer Vision (2015)
InceptionResNetV2	Combines advantages of Inception and ResNet	299x299	~55.9 million	Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning (2016)
Xception	Deep convolutional network, high performance	299x299	~22.9 million	Xception: Deep Learning with Depthwise Separable Convolutions (2016)
MobileNet	Lightweight model, efficient on mobile devices	224x224	~4.2 million	MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications (2017)
MobileNetV2	Improved version of MobileNet, less computation	224x224	~3.5 million	MobileNetV2: Inverted Residuals and Linear Bottlenecks (2018)
DenseNet121	Uses dense blocks, high computational efficiency	224x224	~8 million	Densely Connected Convolutional Networks (2016)
DenseNet169	Deeper DenseNet structure	224x224	~14 million	Densely Connected Convolutional Networks (2016)
DenseNet201	Deepest DenseNet structure	224x224	~20 million	Densely Connected Convolutional Networks (2016)
EfficientNetB0	Efficient model designed using neural architecture search	224x224	~5.3 million	EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks (2019)
EfficientNetB7	Largest model of EfficientNet	600x600	~66 million	EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks (2019)

전이학습(Transfer Learning)

- 미세 조정 방식(Fine-tuning)



	Fine-Tuning	Transfer-Learning
Freeze	4가지 방식이 존재 1) 추가된 Layer 빼고 Freeze 2) 많이 Freeze 3) 약간만 Freeze 4) 모두 Freeze 안 시키기	마지막 Layer만 학습시키는 것 나머지 부분은 모두 Freeze
Learning Rate	매우 작아야 함	관계 없음

*** 보통 10~100배 작게 설정하는 것이 일반적**

ex) Adam default α = 0.001 → 0.0001(or 0.00001)로 설정

전이학습(Transfer Learning)

- 미세 조정 방식(Fine-tuning)

- 모델 베이스 중 상위 몇 개의 계층은 완전 연결층 분류기와 함께 새로 학습시키는 방식
- 최종 분류기의 파라미터가 랜덤하게 초기화되어 있으므로, 이를 먼저 학습(Transfer Learning)

  * 즉, 이 동안은 미세조정을 하지 않도록
  상위 계층의 파라미터를 고정(freeze)시켜 둠

- 먼저 분류부를 학습시킨 다음, 그 이후에 미세조정을 수행

  * 처음부터 모델 베이스 중 상위 계층의 파라미터를 같이 훈련시키면
  분류기에서 발생하는 초기 Epoch 동안의 큰 에러 값으로 인해, 사전 학습된 정보가 손실

전이학습(Transfer Learning)

- 미세 조정 방식(Fine-tuning)

1) 사전 학습 된 특성추출부 상단에 새로운 분류부 추가

2) 사전 학습 된 특성추출부 고정(freeze)

3) 새로운 분류부 학습

4) 사전 학습 된 특성 추출부 중 학습시킬 상위 부분의 고정(freeze) 풀기

5) 고정을 푼 계층과 새로운 분류부를 함께 작은 학습률로 Training

전이학습(Transfer Learning) 전략