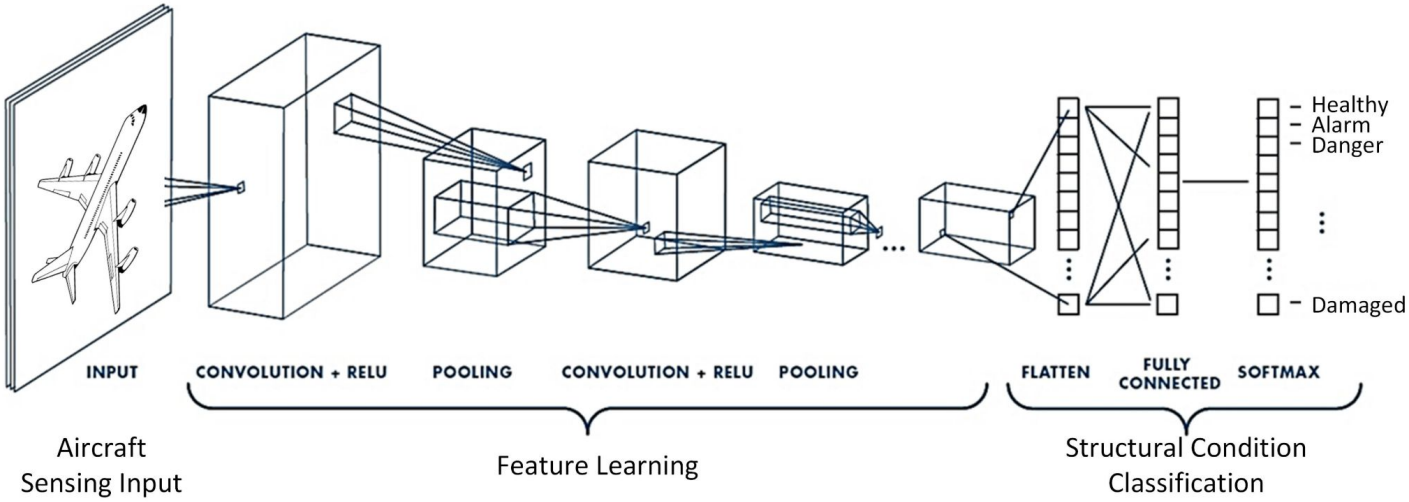
Convolutional Neural Network (CNN)

이미지 구조

합성곱 신경망 개요

합성곱 신경망이란?

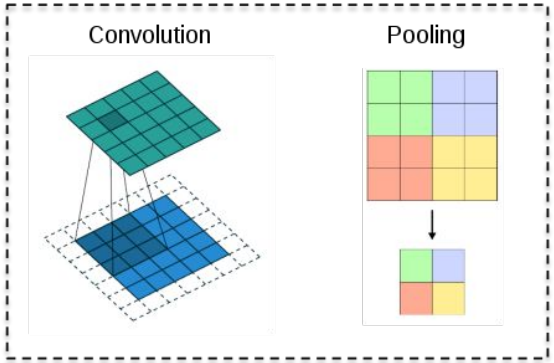
- 합성곱 신경망(Convolutional Neural Network ;CNN)은 딥러닝의 한 분야로, 이미지 인식과 처리에 주로 사용되는 인공신경망(Artificial Neural Network; ANN) 의 한 종류
- 이미지 분류, 객체 인식 및 탐지, 얼굴 인식, 의료 이미지 분석, 자연어 처리, 음성 인식 등에 응용

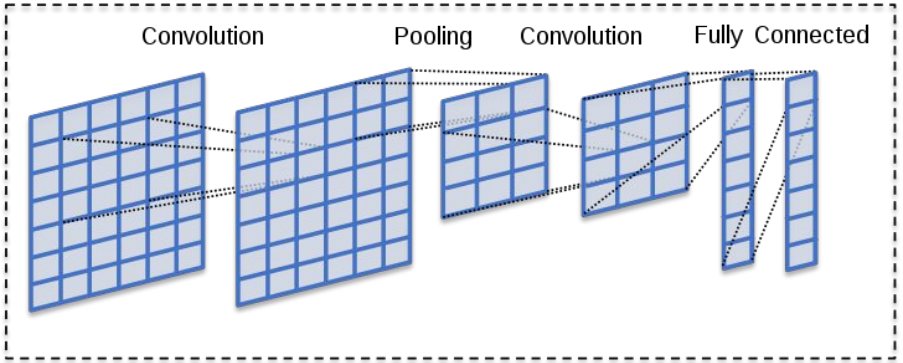


합성곱 신경망 개요

합성곱 신경망 특징

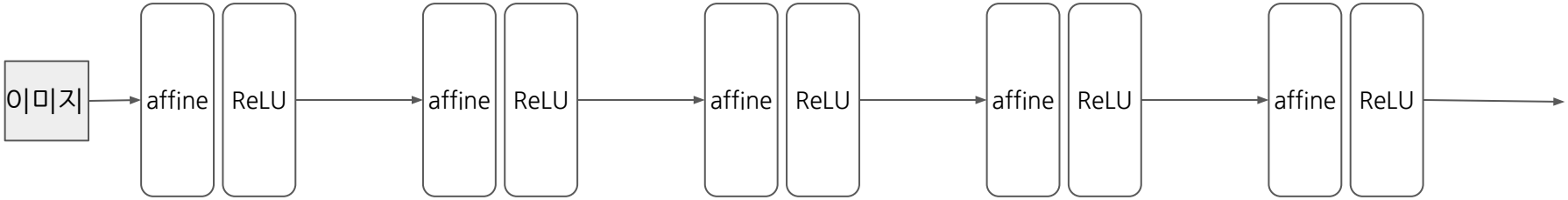
- 일반 신경망은, 입력할 이미지 전체를 하나의 데이터로 처리하므로, 이미지의 특성을 찾기 힘들고, 이미지의 위치에 조금의 변동이 있거나 왜곡될 경우 좋은 성능을 낼 수 없음.
- 합성곱 신경망은 여러 개의 합성곱 레이어(Convolutional layer)와 풀링 레이어(Pooling layer)를 통해 입력 이미지의 특징을 추출하고, 이를 바탕으로 이미지를 분류하거나 다른 작업을 수행함
- 이미지의 부분적 특성을 추출할 수 있어 이미지가 왜곡 되더라도 괜찮은 성능을 낼 수 있음



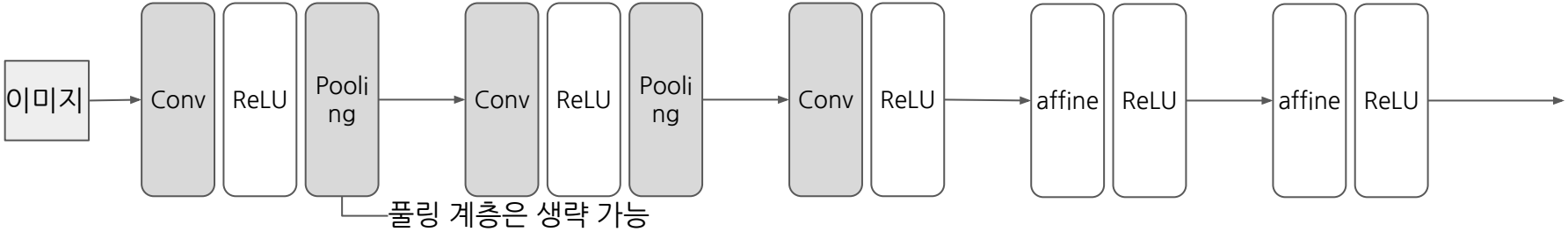


합성곱 신경망 개요

기존의 인공 신경망(ANN) 구조 :
- 인접한 계층의 모든 뉴런이 결합되어진 완전 연결(fully-connected). Affine 계층으로 구성



합성곱 신경망(CNN) 구조 :
- 인공 신경망 구조에 합성곱 계층(Convolutional Layer)과 풀링 계층(Pooling Layer)이 추가됨

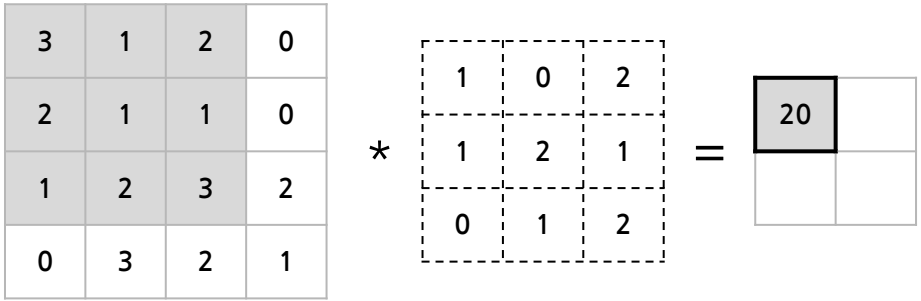


합성곱 연산

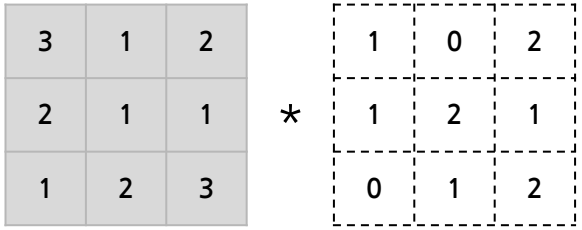
합성곱 연산이란

이미지 처리, 딥러닝, CNN에서의 합성곱 연산이란, 입력데이터와 필터(=마스크) 연산을 의미

- 2차원 입력 데이터(=이미지 데이터)를, 필터가 **일정한 간격으로 이동**하며 연산
- 필터(filter, =kernel, =window, =mask)
- 입력 데이터와 필터에서 **각각에 대응하는 원소를 곱**한 후 그 총합을 구해 결과를 출력
- 이 과정을 **모든 장소에서 수행**하여 합성곱 연산의 값을 출력



입력 데이터 필터(마스크)

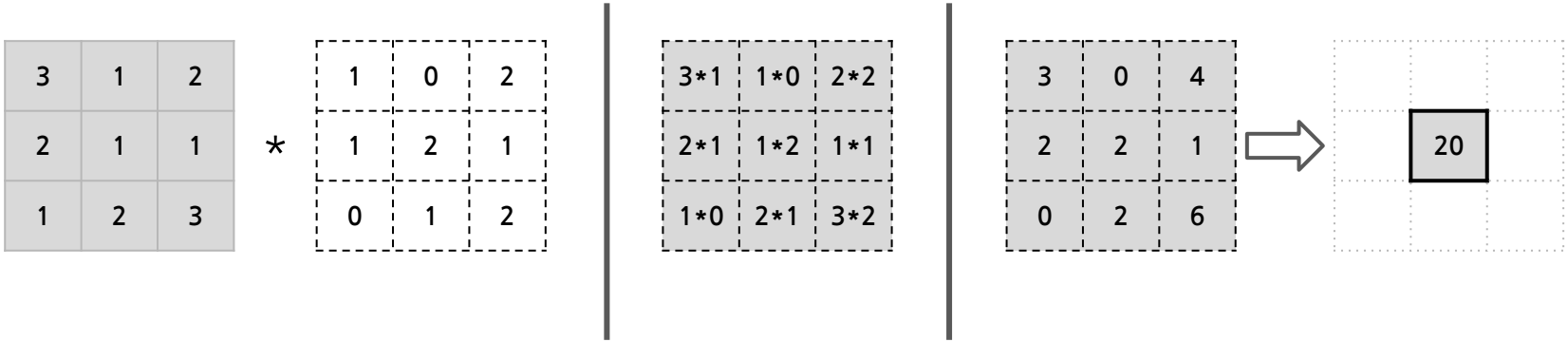


(3*1) + (1*0) + (2*2) +
(2*1) + (1*2) + (1*1) +
(1*0) + (2*1) + (3*2) = 20

합성곱 연산이란

이미지 처리, 딥러닝, CNN에서의 합성곱 연산이란, 입력데이터와 필터(=마스크) 연산을 의미

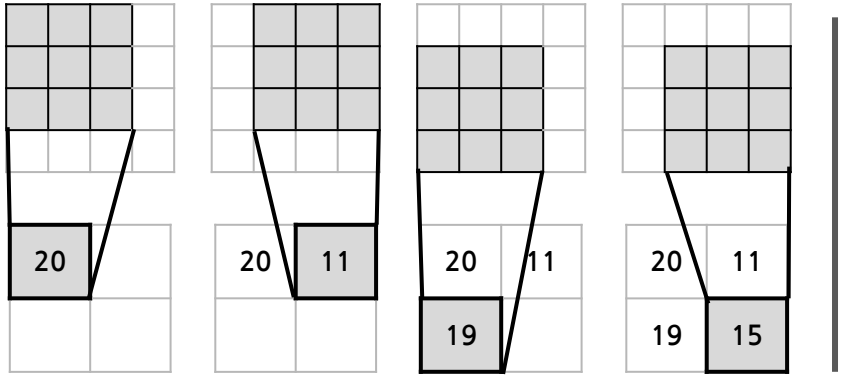
- 2차원 입력 데이터(=이미지 데이터)를, 필터가 **일정한 간격으로 이동**하며 연산
- 필터(filter, =kernel, =window, =mask)
- 입력 데이터와 필터에서 **각각에 대응하는 원소를 곱**한 후 그 총합을 구해 결과를 출력
- 이 과정을 **모든 장소에서 수행**하여 합성곱 연산의 값을 출력



합성곱 연산이란

이미지 처리, 딥러닝, CNN에서의 합성곱 연산이란, 입력데이터와 필터(=마스크) 연산을 의미

- 2차원 입력 데이터(=이미지 데이터)를, 필터가 **일정한 간격으로 이동**하며 연산
- 필터(filter, =kernel, =window, =mask)
- 입력 데이터와 필터에서 **각각에 대응하는 원소를 곱**한 후 그 총합을 구해 결과를 출력
- 이 과정을 **모든 장소에서 수행**하여 합성곱 연산의 값을 출력



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

입력 데이터

*

1	0	2
1	2	1
0	1	2

필터(마스크)

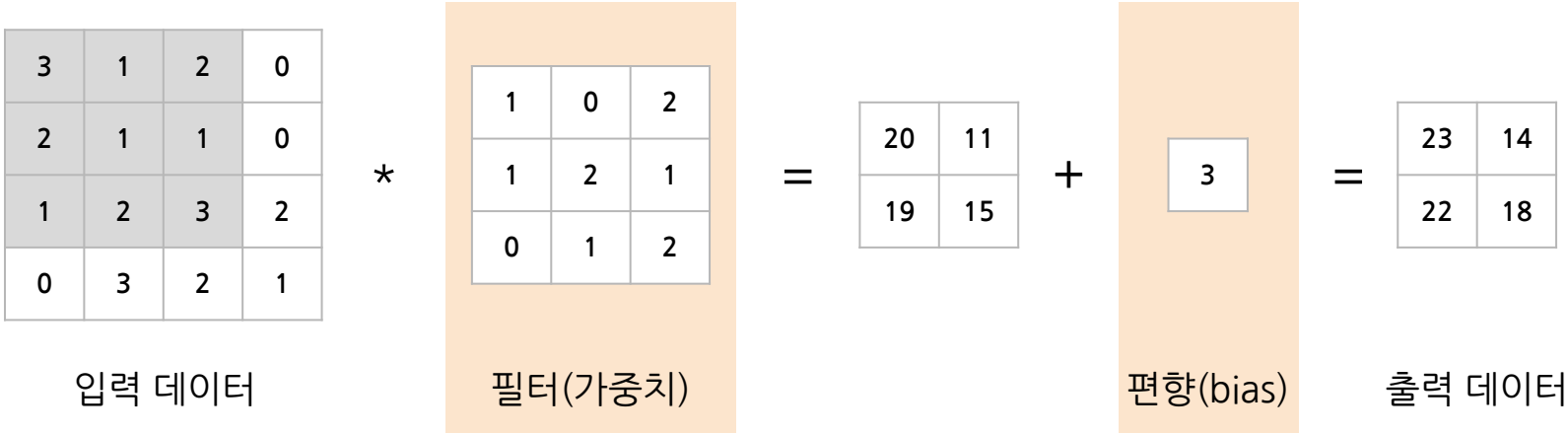
=

20	11
19	15

편향(bias)

필터 적용 후 편향(bias)을 더함

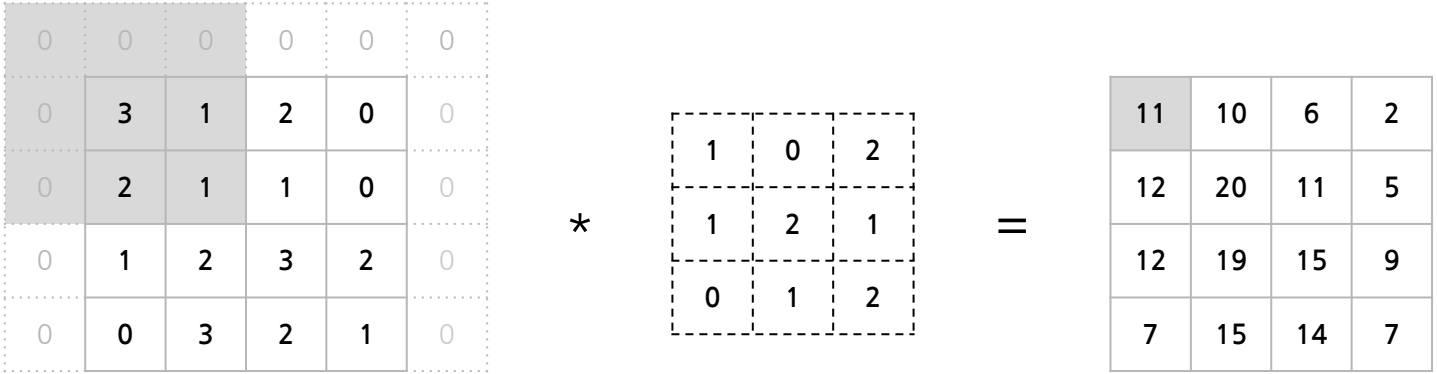
- 완전 연결 신경망(Fully Connected Neural Network)에는 가중치와 편향 매개변수가 존재
- 합성곱 신경망(CNN)에서는 이와 유사하게, 필터(가중치), 편향이 학습을 시킬 매개변수
- 편향은 **하나의 값**으로 존재하며, 필터를 적용한 후 **모든 원소에 더해짐**
- 즉, 합성곱 신경망을 통해 학습이 거듭되며 필터의 원소 값과 편향이 매번 갱신됨



패딩(Padding)

입력 데이터와 출력 데이터의 크기를 맞추기 위해 사용

- 합성곱 연산을 수행하기 전에 사용함
- 입력 데이터 주변을 특정 값으로 채움(일반적으로 0을 사용=Zero padding)



입력 데이터
(4,4) ➜ zero padding:1 ➜(6,6)

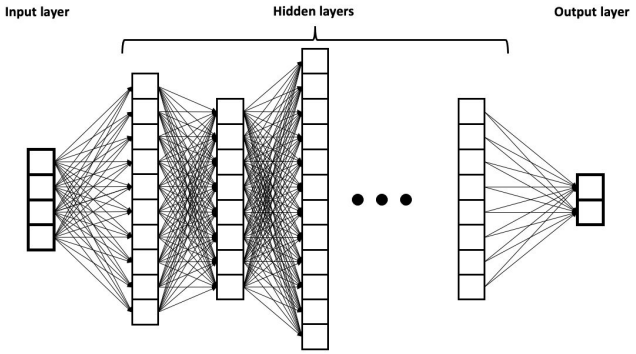
필터
(3,3)

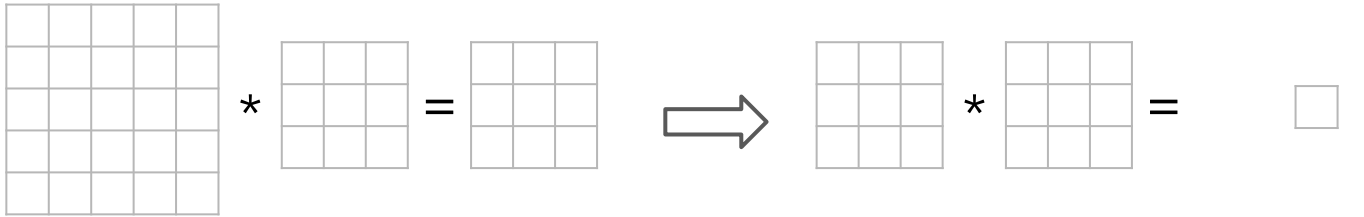
출력 데이터
(4,4)

패딩(Padding)

패딩(padding) 처리를 하는 이유

- 예를 들어 (5,5) 입력 데이터에 (3,3) 필터를 적용하면 출력은 (3,3)가 되어, 입력 크기보다 작아짐
- 합성곱 연산을 여러 번 반복하는 심층 신경망에서는 더이상 합성곱 연산을 할 수 없게 출력 크기가 1이 되어버리는 문제가 발생할 수 있음
- 이를 해결하기 위해 패딩을 사용하여 출력 크기를 조정

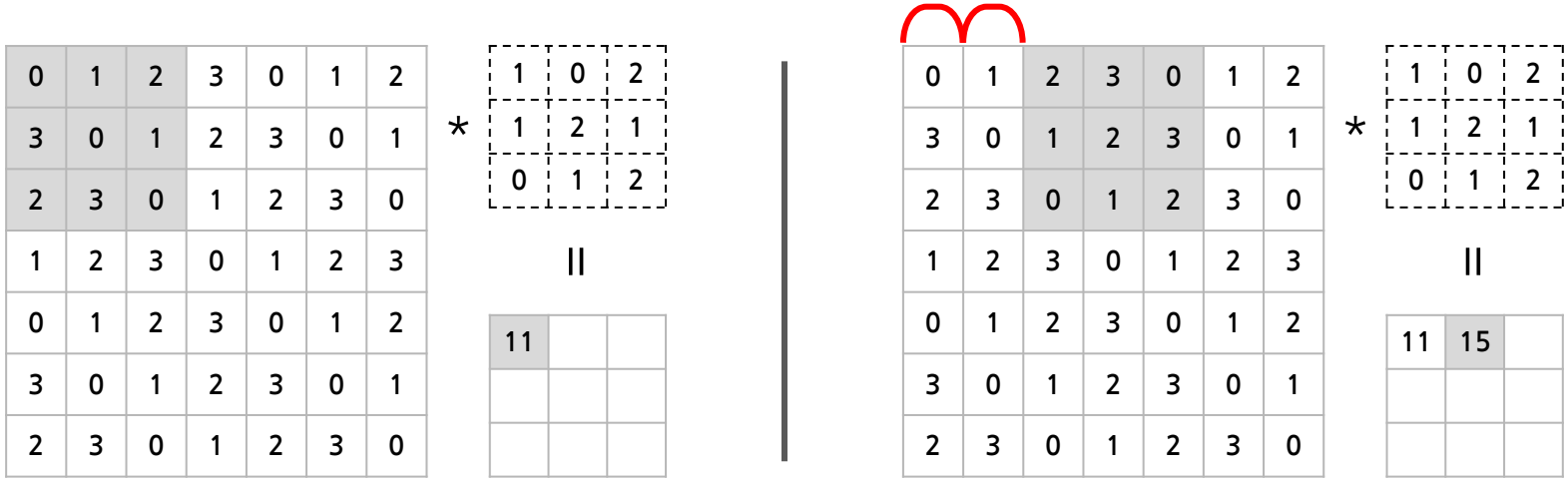




스트라이드(Stride)

입력 데이터에 필터를 순차적으로 적용할때, 필터의 위치를 정하는 간격

- 예를 들어, 스트라이드(Stride)가 2라면 필터가 첫 번째 연산 후 2칸의 간격으로 이동하며 연산함
- 스트라이드(Stride)가 커지면 출력 데이터의 크기가 작아짐



스트라이드 출력 크기

- 입력 데이터 크기(높이, 너비) : (H, W)
- 필터 크기(높이, 너비) : (FH, FW)
- 출력 데이터 크기(높이, 너비) : (OH, OW)
- 패딩 크기 : P
- 스트라이드 크기 : S

$$OW = \frac{W + 2P - FW}{S} + 1$$

$$OH = \frac{H + 2P - FH}{S} + 1$$

$$\text{출력너비} = \frac{(\text{너비} + 2\text{패딩} + \text{필터너비})}{\text{스트라이드}} + 1$$

$$\text{출력높이} = \frac{(\text{높이} + 2\text{패딩} + \text{필터높이})}{\text{스트라이드}} + 1$$

스트라이드 출력 크기

- 입력 데이터 크기(높이, 너비) : (4, 4)
- 필터 크기(높이, 너비) : (3, 3)
- 출력 데이터 크기(높이, 너비) : (OH, OW)
- 패딩 크기 : 1
- 스트라이드 크기 : 1

$$OW = \frac{4 + (2 \cdot 1) - 3}{1} + 1 = 4$$

$$OH = \frac{4 + (2 \cdot 1) - 3}{1} + 1 = 4$$

입력 데이터 (4,4)

패딩 : 1,
스트라이드 : 1

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

*

1	0	2
1	2	1
0	1	2

필터
(3,3)

=

11	10	6	2
12	20	11	5
12	19	15	9
7	15	14	7

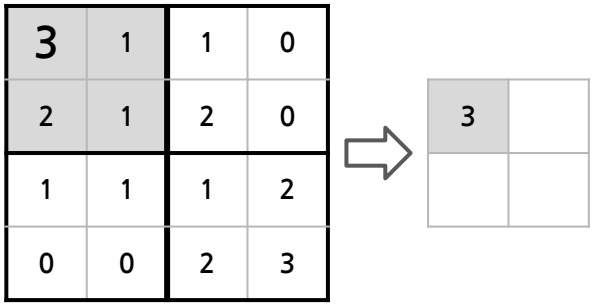
출력 데이터
(4,4)

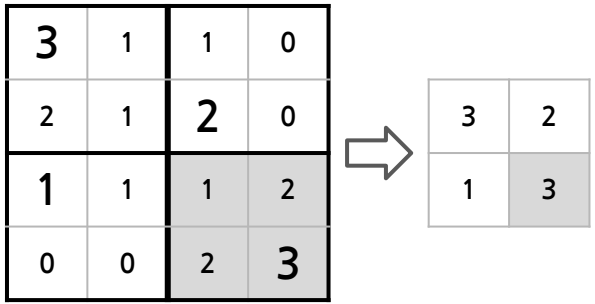
풀링(Pooling)

풀링 연산이란 입력 데이터의 공간을 축소시키는 연산. (=Down sampling, =Sub sampling)

- 예를 들어, (2,2) 영역을 (1,1)로 축소
- 주로 풀링의 윈도우 크기와 스트라이드의 값을 같은 크기로 설정함
- 최대 풀링(Max pooling)은 윈도우 영역에서 최댓값을 가져오고, 평균 풀링(Average pooling)은 평균 값을 계산해 가져옴
- 컴퓨터 비전 분야에서는 주로 Max pooling을 사용

Max pooling 예시(풀링 윈도우: 2*2, 스트라이드: 2)

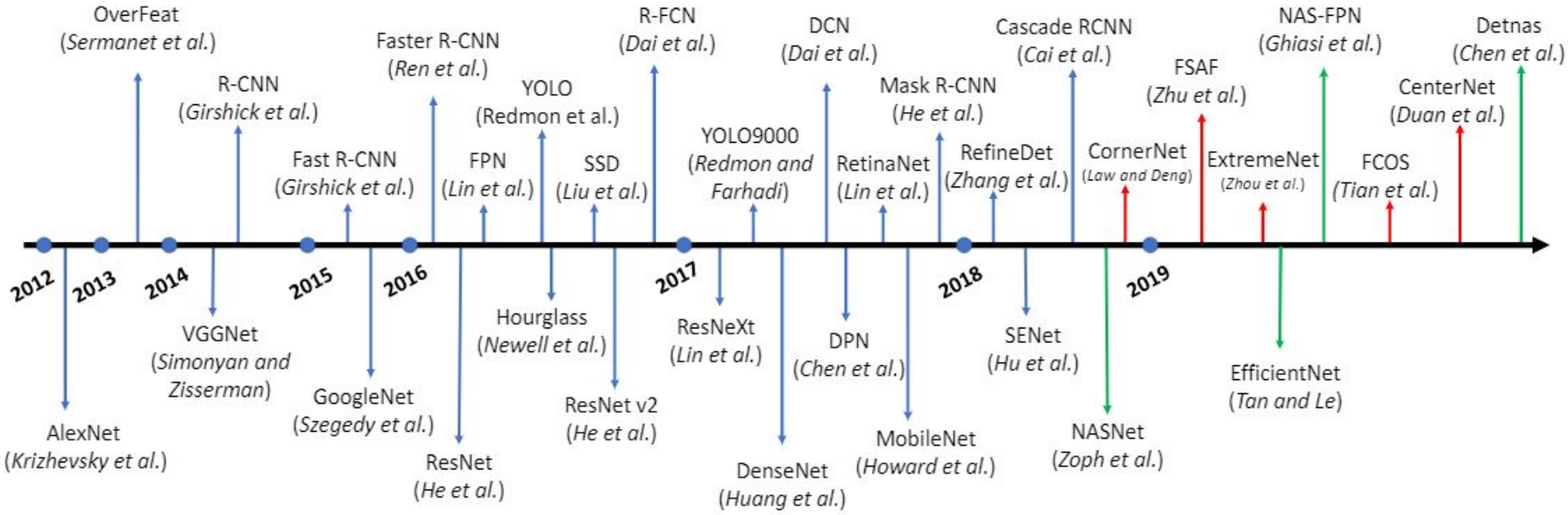




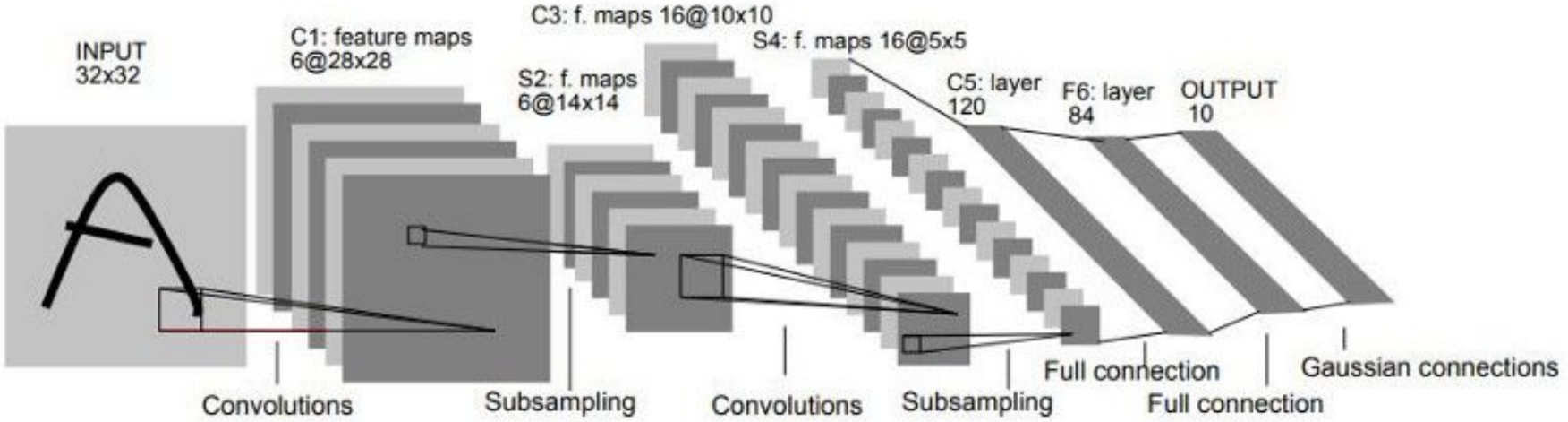
풀링(Pooling)

- 각각 기호를 아래와 같이 정의
  - O: Size(width) of output image
  - I: Size(width) of input image
  - S: Stride of the convolution operation
  - P_s: Pooling size
- O(Size(width) of output image)는 다음과 같이 정의 됨

$$O = \frac{I - P_s}{s} + 1$$


2012
2013
2014
2015
2016
2017
2018
2019
AlexNet
(Krizhevsky et al.)
OverFeat
(Sermanet et al.)
VGGNet
(Simonyan and Zisserman)
R-CNN
(Girshick et al.)
GoogleNet
(Szegedy et al.)
Fast R-CNN
(Girshick et al.)
ResNet
(He et al.)
Faster R-CNN
(Ren et al.)
Hourglass
(Newell et al.)
FPN
(Lin et al.)
ResNet v2
(He et al.)
SSD
(Liu et al.)
YOLO
(Redmon et al.)
DenseNet
(Huang et al.)
R-FCN
(Dai et al.)
ResNeXt
(Lin et al.)
YOLO9000
(Redmon and Farhadi)
MobileNet
(Howard et al.)
DPN
(Chen et al.)
DCN
(Dai et al.)
RetinaNet
(Lin et al.)
Mask R-CNN
(He et al.)
SENet
(Hu et al.)
RefineDet
(Zhang et al.)
NASNet
(Zoph et al.)
Cascade RCNN
(Cai et al.)
CornerNet
(Law and Deng)
EfficientNet
(Tan and Le)
ExtremeNet
(Zhou et al.)
FSAF
(Zhu et al.)
FCOS
(Tian et al.)
NAS-FPN
(Ghiasi et al.)
CenterNet
(Duan et al.)
Detnas
(Chen et al.)

Lenet 5


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections