



Convolutional Neural Network (CNN)

딥러닝 / 머신러닝

임 경 태

Week	Chapter	Contents
1	1, 2장	강의 소개, 파이썬 복습
2	1, 3장	파이썬 복습, Numpy, Pandas
3	1, 4장	딥러닝을 위한 미분
4	5장	회귀
5	5장	분류
6	6장	XOR문제
7	7장	딥러닝
8	1~7장	중간고사
9	8장	MNIST 필기체 구현 및 팀 프로젝트 소개
10	9장	오차역전파
11	11장	Jetbot 자율주행 (Collision Avoidance, Transfer Learning)
12	12장	특강 (인공지능 활용 연구) + 팀프로젝트 자율 실습
13	10장	Jetbot 자율주행 (Road Following)
14	11장	합성곱 신경망(CNN), 순환 신경망(RNN) + 팀프로젝트 자율 실습
15	8~12장	기말고사 (or 프로젝트 발표)

CONTENTS

—

- ① CNN 개요
- ② CNN 구조
- ③ CNN 개념
- ④ CNN 이해



목적 : CNN의 구조와 CNN개념 이해



목표 : CNN에 관련된 용어와 사용하는 이유 이해



내용 : CNN 구조, CNN관련 용어 등

CONTENTS

—

① CNN 개요

② CNN 구조

③ CNN 개념

④ CNN 이해

CNN 개요

● FFNN의 배신

- 너무 많은 파라미터를 가짐, 파라미터 개수가 많기 때문에 가설 함수 $H(X)$ 가 복잡해지고 Overfitting에 취약함
 - MNIST의 경우 784개의 input 이었고 layer를 쌓을 수록 지수 형태의 파라미터 개수 필요.



뭔가 핵심 정보만 압축할 수 있는 방법 없냐?

CNN 개요

What is CNN(Convolution Neural Network)?

- 한꺼번에 **전체를** 보는게 **아닌 부분**을 보며 **특징점**을 찾는 것이 핵심 아이디어
- 이미지 인식 – 패턴 학습에 특화된 신경망
- Convolution 연산을 통한 연산
- Fully-Connected Layer에 비해 매우 빠르고 적은 파라미터를 가짐.

CONTENTS

① CNN 개요

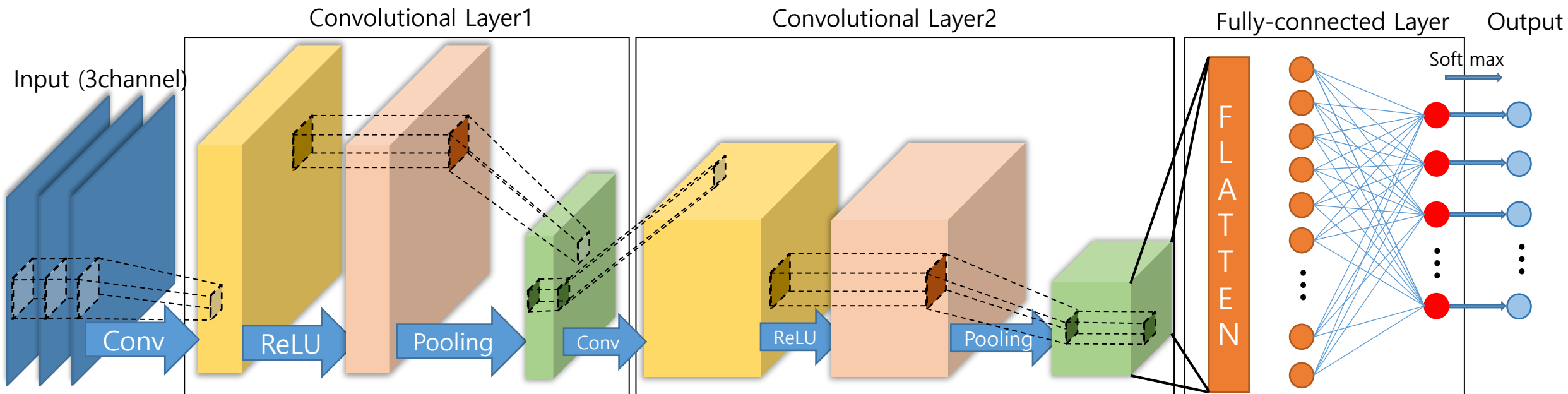
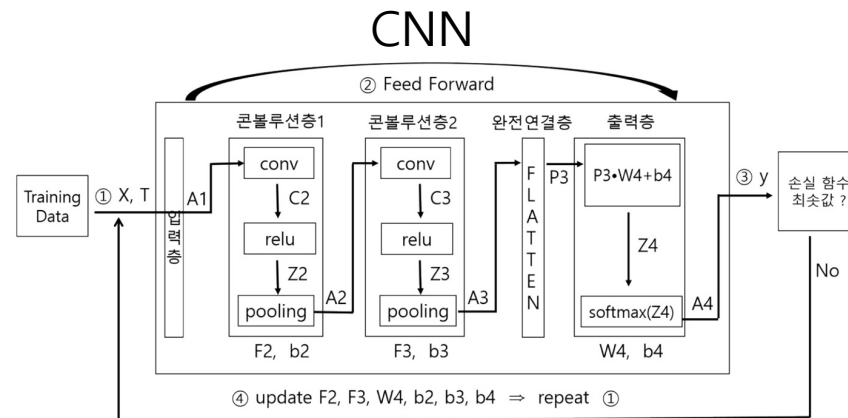
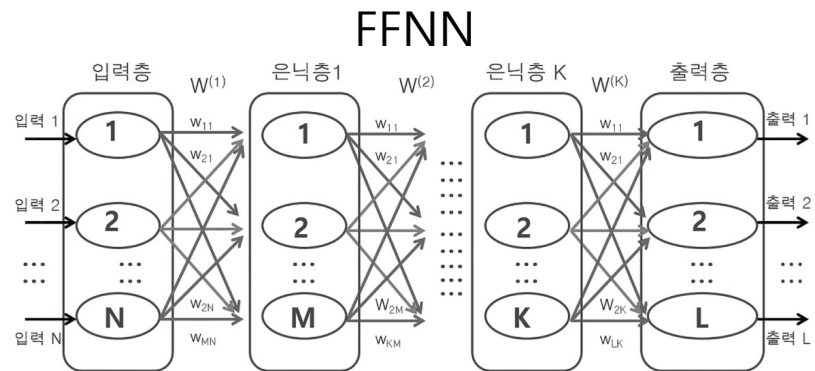
② CNN 구조

③ CNN 개념

④ CNN 이해

CNN 구조

✎ 기본적인 CNN 구조



CONTENTS

① CNN 개요

② CNN 구조

③ CNN 개념

④ CNN 이해

CNN 개념

용어

1. Filter (Kernel)
2. Convolution Filter (Kernel)
3. Stride
4. Pooling
5. Feature Map, Activation Map

CNN 개념

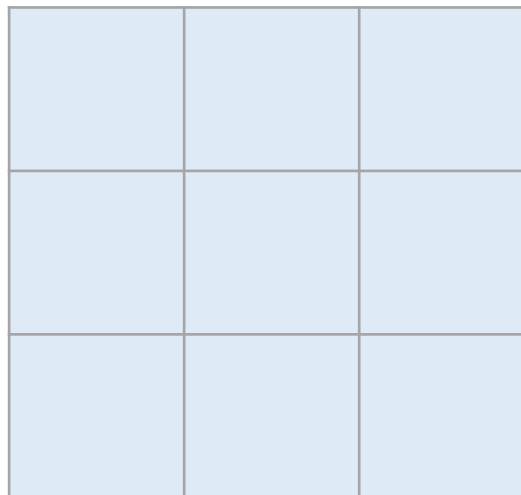
🖋 Filter (Kernel)?

- 이미지의 **특징점**을 찾아내기 위한 **Parameter (기계학습으로 찾아야하는 값)**
- 필터를 사용하여 **parameter 수 감소**
- 필터를 사용하여 입력 데이터의 **부분만** 볼 수 있음
→ 공간적 정보 이용 가능
(공간적 정보 : 이미지에서 “인접한 픽셀 값이 비슷함”, “거리가 먼 픽셀은 연관이 없음” 등)

Filter – 2x2



Filter – 3x3



CNN 개념

📝 Intuition of Filter (엣지를 찾는 필터: 소벨 필터)

- 엣지: 이미지에서 픽셀의 밝기가 급격히 변하는 부분 → 객체와 배경의 경계를 나타냄
- 엣지검출: 픽셀값이 급격하게 변화는 부분을 찾으면 됨 → 변화? → 변화율? → 미분? → 중앙차분!

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

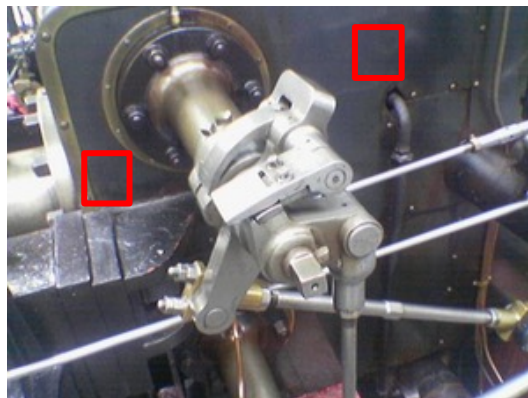
-1	0	+1
-2	0	+2
-1	0	+1

x filter

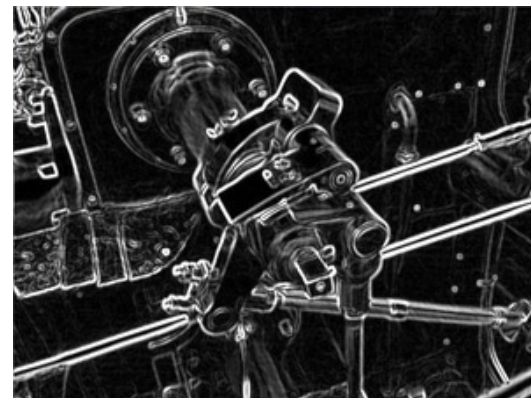
+1	+2	+1
0	0	0
-1	-2	-1

y filter

x필터 값이 0



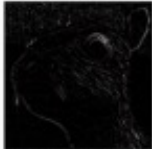




x필터 값이 0



CNN 개념

Intuition of Filter

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	



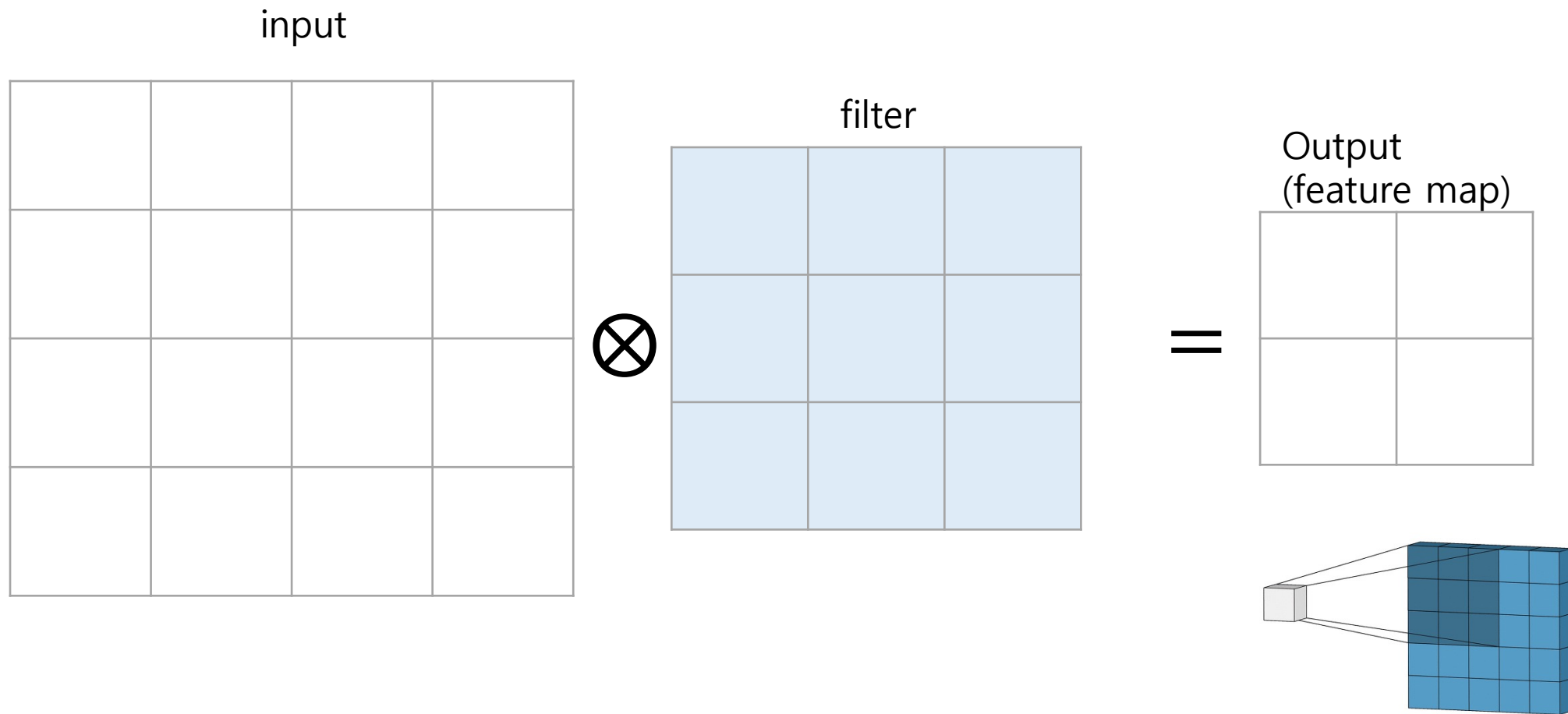
Input

CNN 개념

✎ What is Convolution ?

** : Convolution symbol in mathematics*

$$t[n] = x[n] * h[n] = \sum_{k=0}^N x[k]h[n-k] \quad (\text{Discrete})$$



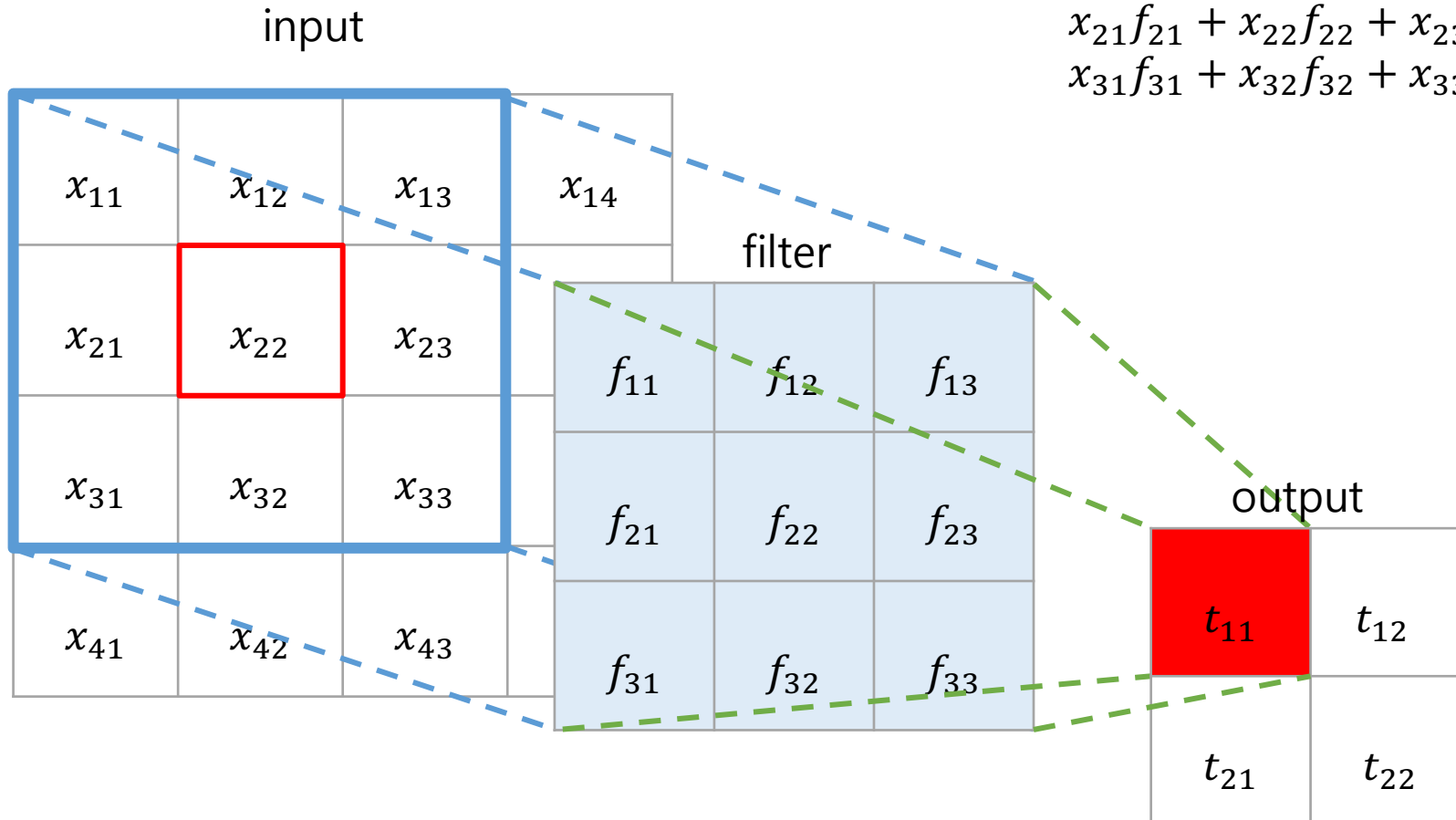
CNN 개념

● Convolution Operation

$*$: Convolution symbol in mathematics

$$t[n] = x[n] * h[n] = \sum_{k=0}^N x[k]h[n - k] \quad (\text{Discrete})$$

$$t_{11} = x_{11}f_{11} + x_{12}f_{12} + x_{13}f_{13} + \\ x_{21}f_{21} + x_{22}f_{22} + x_{23}f_{23} + \\ x_{31}f_{31} + x_{32}f_{32} + x_{33}f_{33}$$



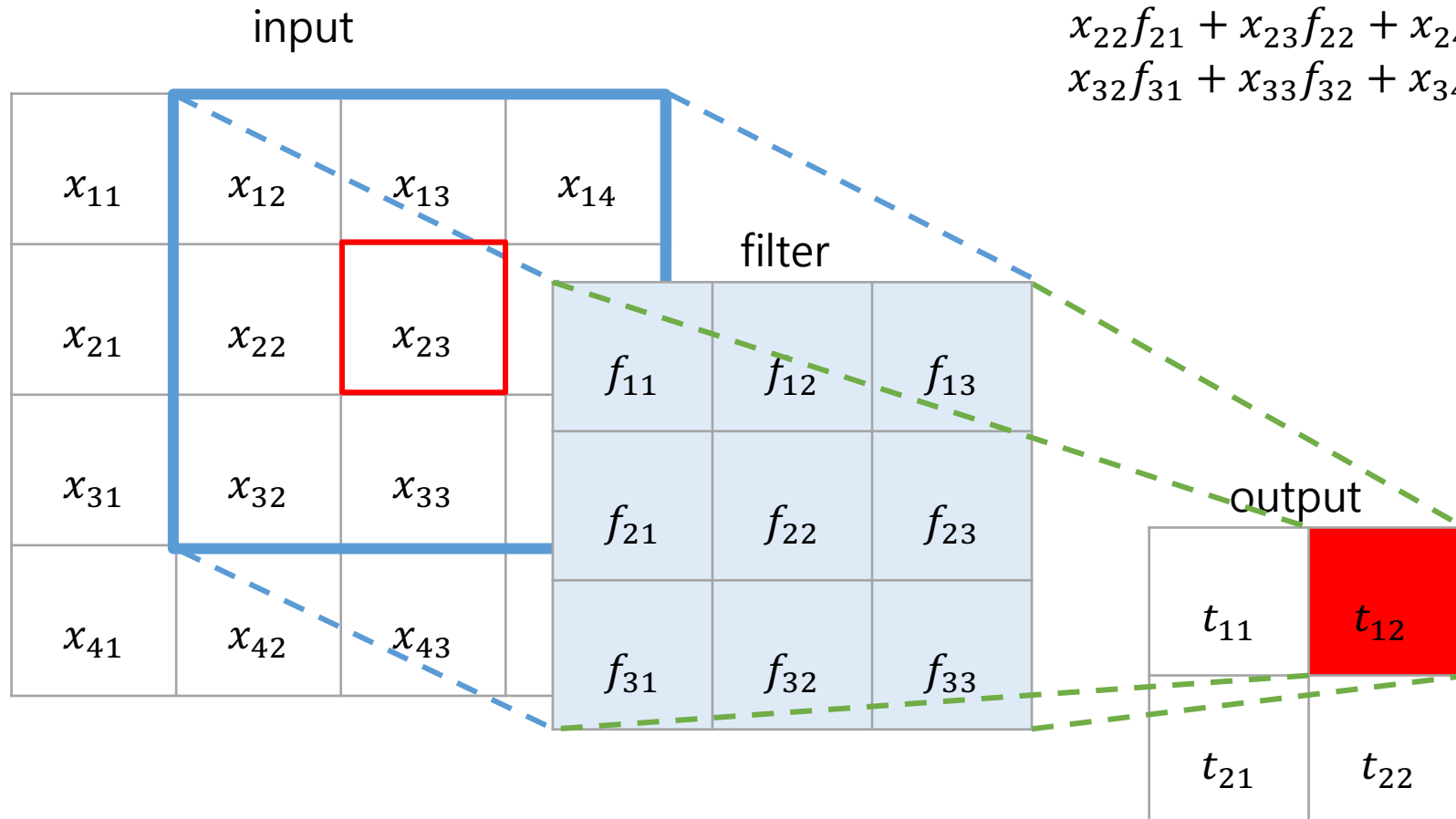
CNN 개념

● Convolution Operation

$*$: Convolution symbol in mathematics

$$t[n] = x[n] * h[n] = \sum_{k=0}^N x[k]h[n-k] \quad (\text{Discrete})$$

$$t_{12} = x_{12}f_{11} + x_{13}f_{12} + x_{14}f_{13} + \\ x_{22}f_{21} + x_{23}f_{22} + x_{24}f_{23} + \\ x_{32}f_{31} + x_{33}f_{32} + x_{34}f_{33}$$



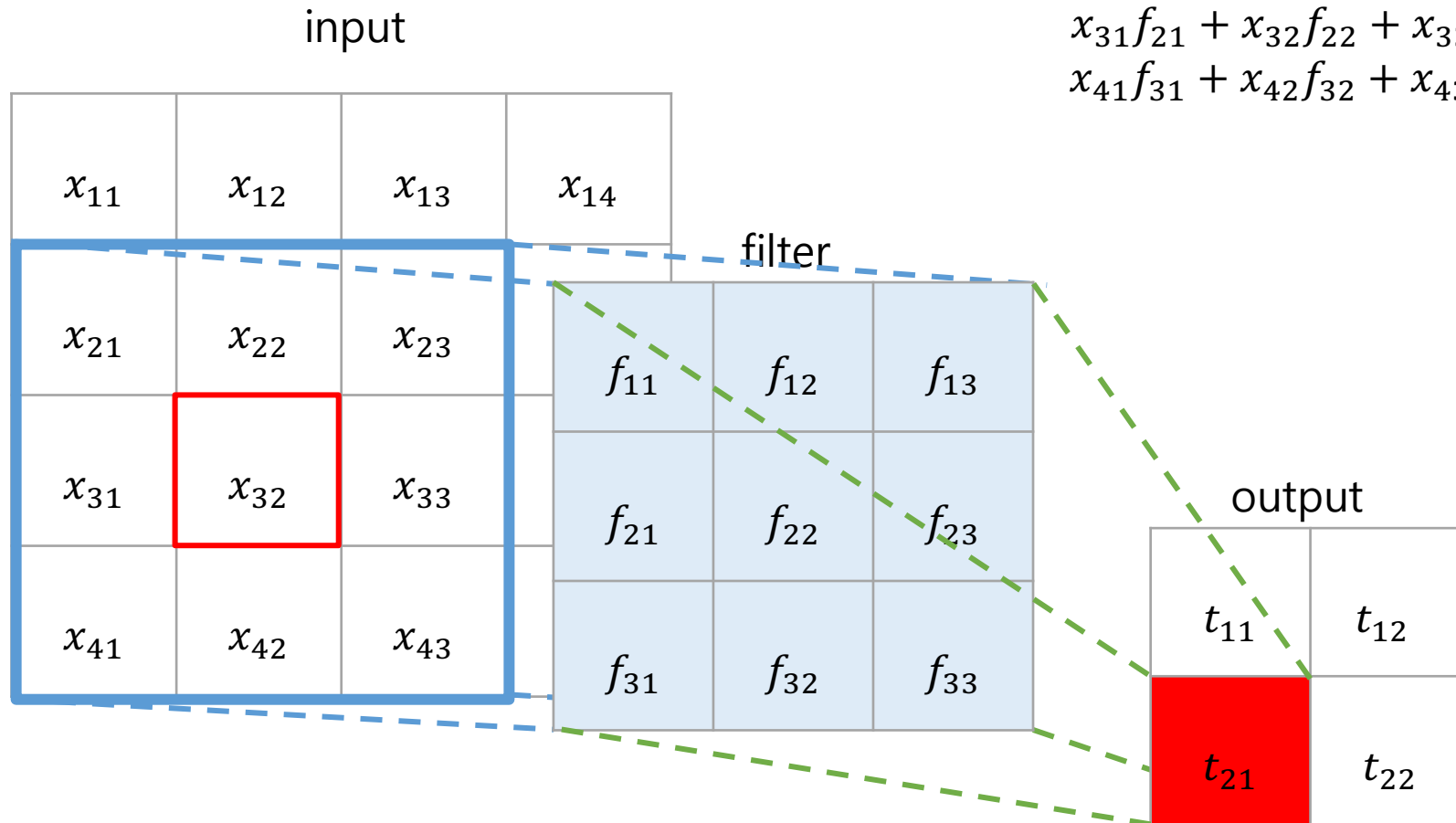
CNN 개념

● Convolution Operation

** : Convolution symbol in mathematics*

$$t[n] = x[n] * h[n] = \sum_{k=0}^N x[k]h[n-k] \quad (\text{Discrete})$$

$$t_{21} = x_{21}f_{11} + x_{22}f_{12} + x_{23}f_{13} + \\ x_{31}f_{21} + x_{32}f_{22} + x_{33}f_{23} + \\ x_{41}f_{31} + x_{42}f_{32} + x_{43}f_{33}$$



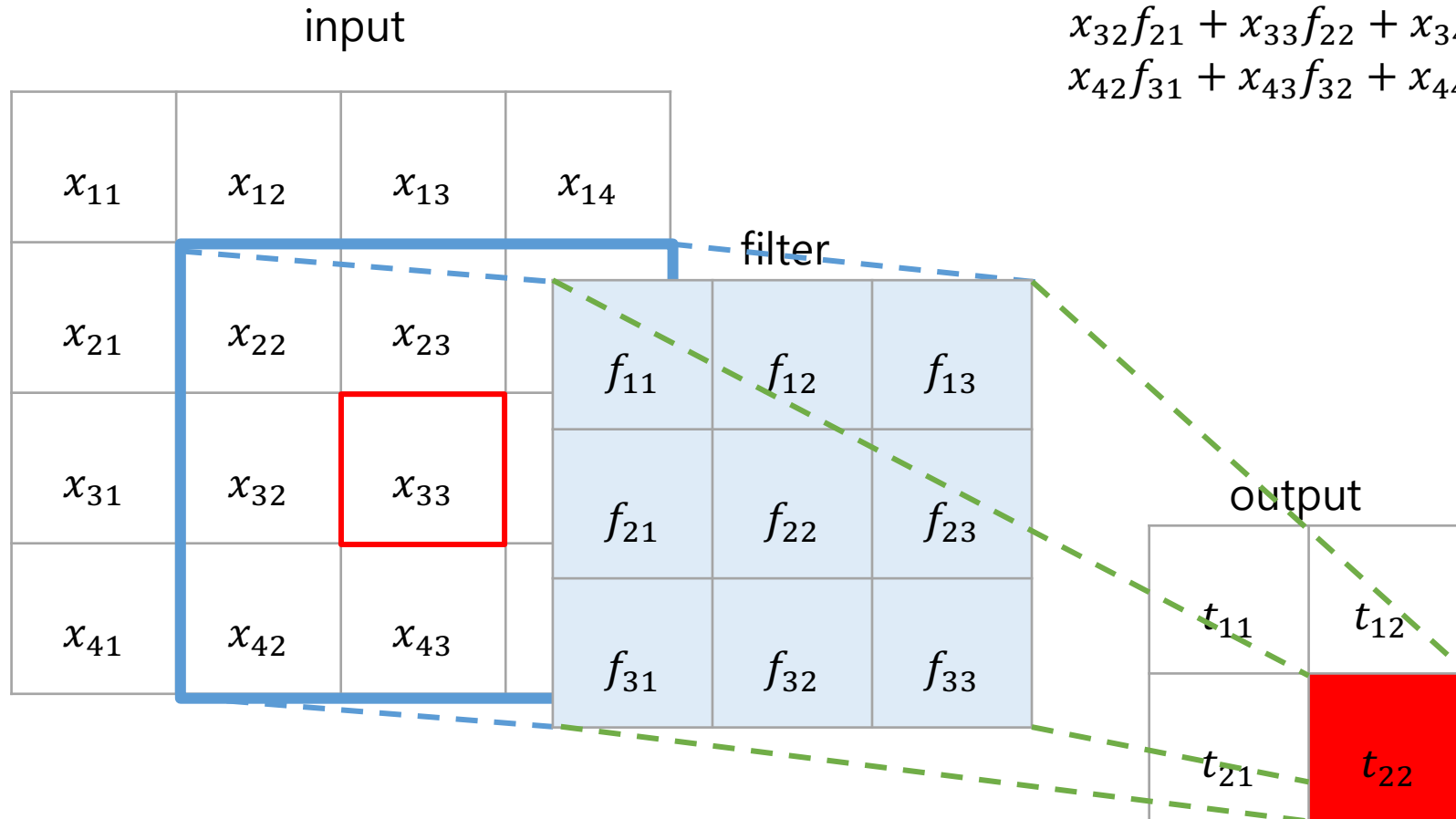
CNN 개념

🖋 Convolution Operation

$*$: Convolution symbol in mathematics

$$t[n] = x[n] * h[n] = \sum_{k=0}^N x[k]h[n-k] \quad (\text{Discrete})$$

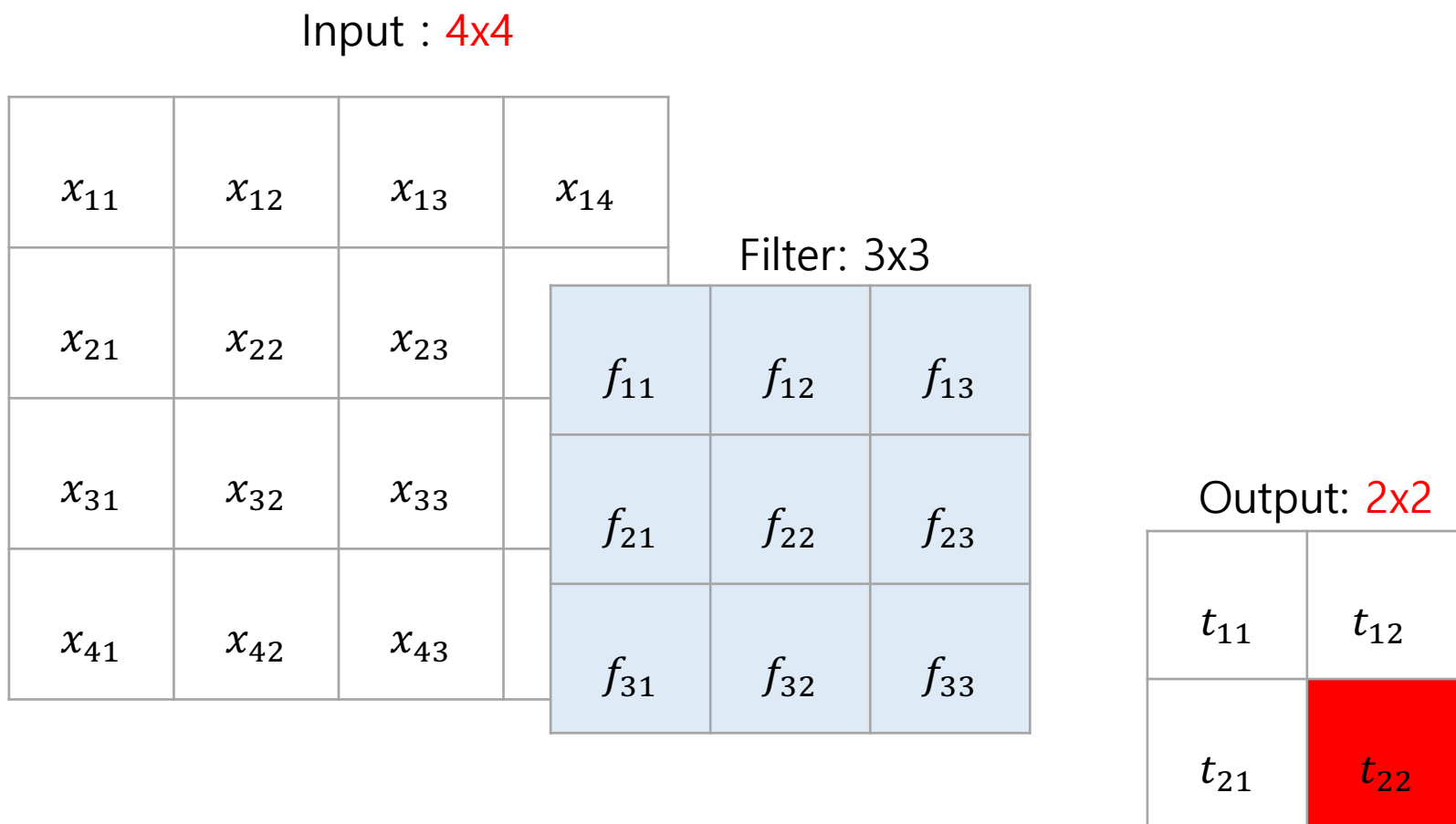
$$t_{22} = x_{22}f_{11} + x_{23}f_{12} + x_{24}f_{13} + \\ x_{32}f_{21} + x_{33}f_{22} + x_{34}f_{23} + \\ x_{42}f_{31} + x_{43}f_{32} + x_{44}f_{33}$$



CNN 개념

📝 Convolution 결과에 대한 고찰

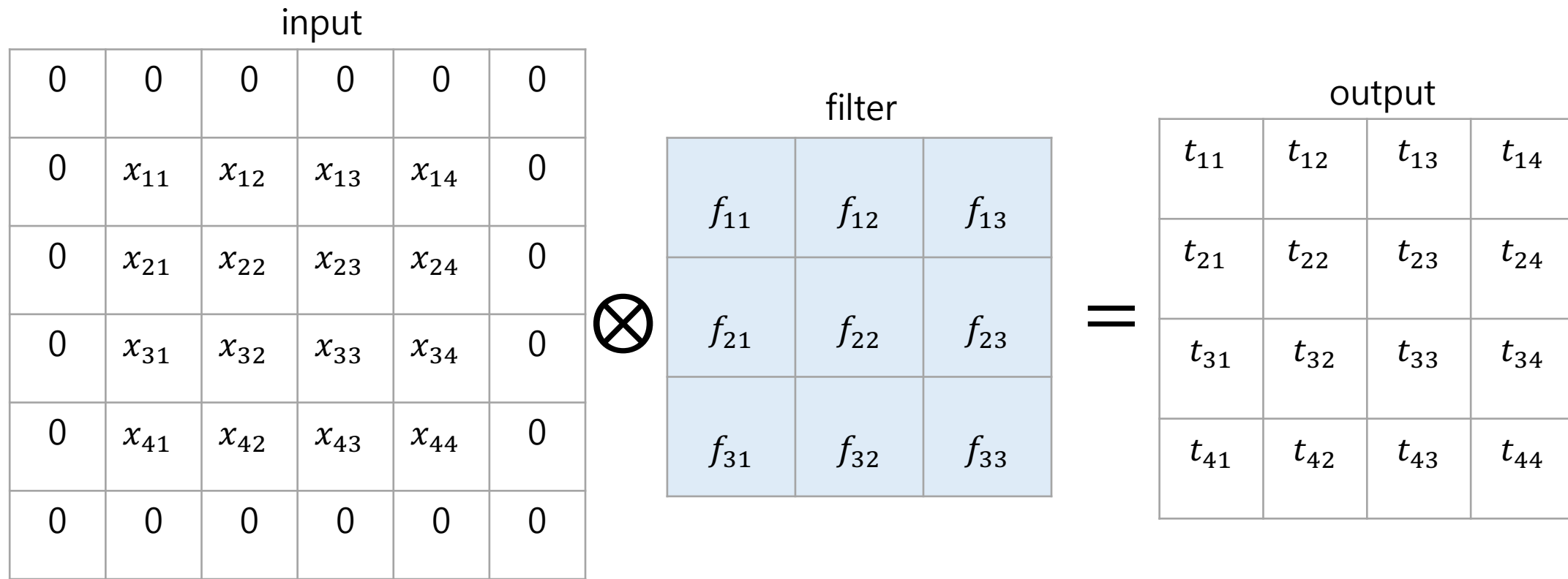
- Input과 filter의 convolution 연산을 진행하면 output size가 작아 짐 == 정보가 압축됨 (손실발생)



CNN 개념

🖋 Padding (zero padding)

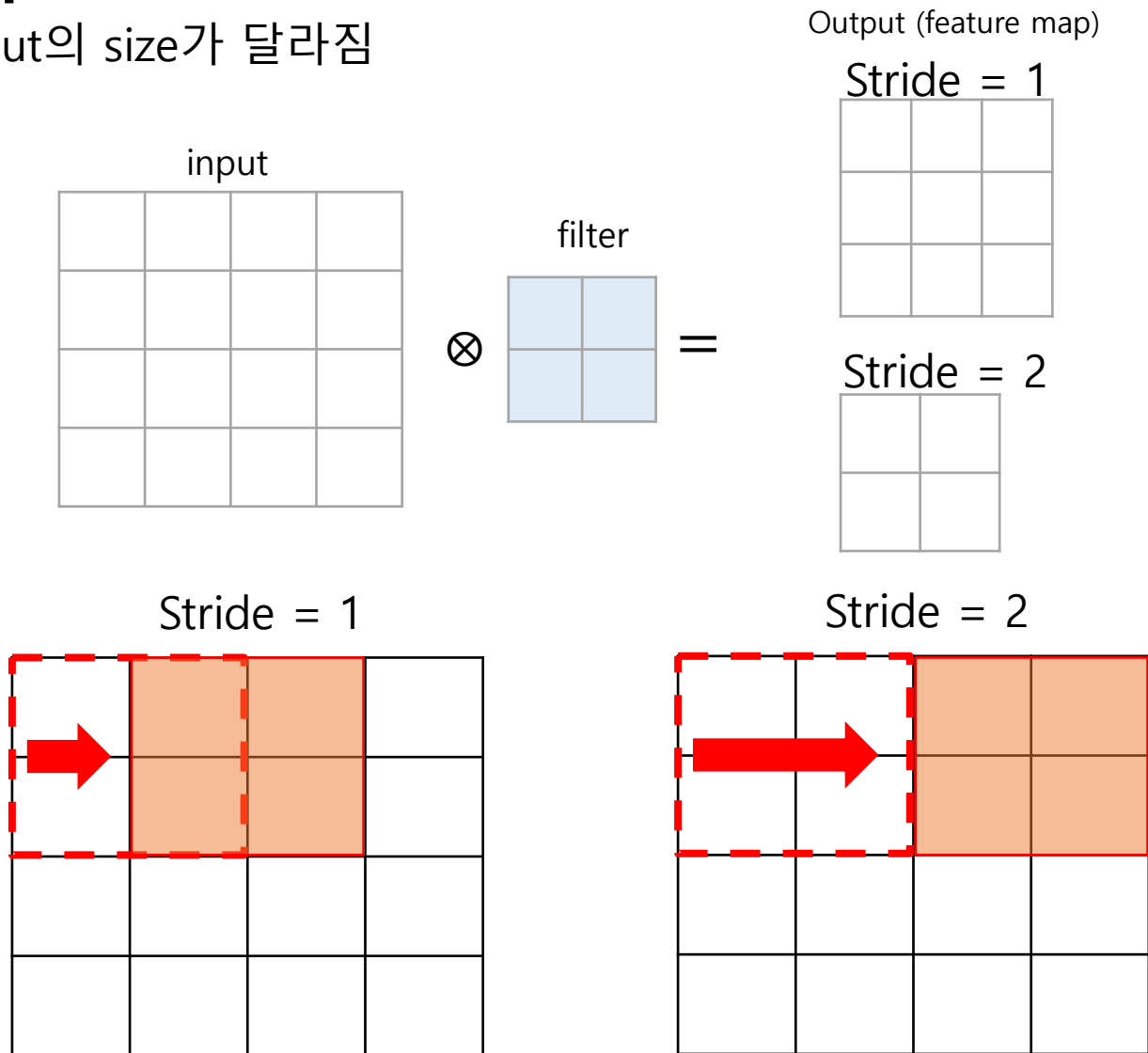
- Input과 filter의 convolution 연산을 진행하면 output size가 작아 짐
→ Padding 사용, Output의 size가 너무 작아지는 것을 방지
- Input의 edge부분의 정보도 충분히 이용할 수 있도록 변환



CNN 개념

📌 Stride : 보폭

- Stride에 따라 output의 size가 달라짐



CNN 개념

Pooling

- Max Pooling : 특정 구역(window)에서 최댓값을 뽑아 냄
- Average Pooling : 특정 구역(window)의 평균값을 뽑아 냄
- Pooling을 통한 Down Sampling
- Window size, Stride

1	2	0	1	3	2
3	1	2	1	0	1
0	4	1	0	2	5
1	3	2	4	2	1
1	2	1	4	1	2
0	2	5	4	3	1

Window size = (2, 2), Stride = 2

Max pooling



3	2	3
4	4	5
2	5	3

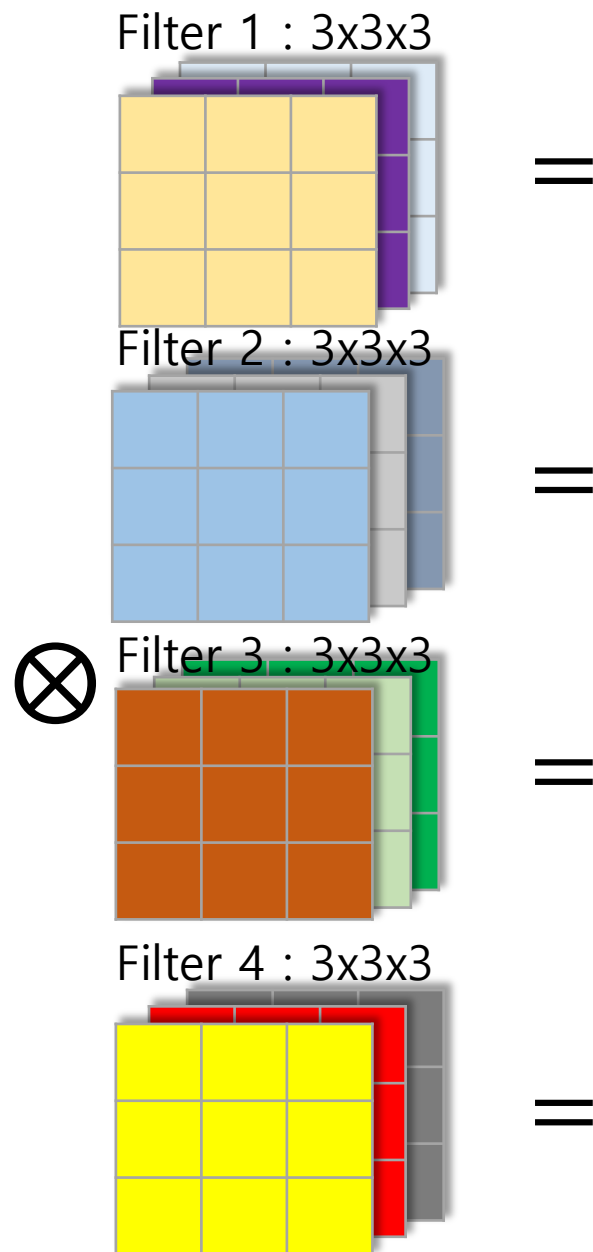
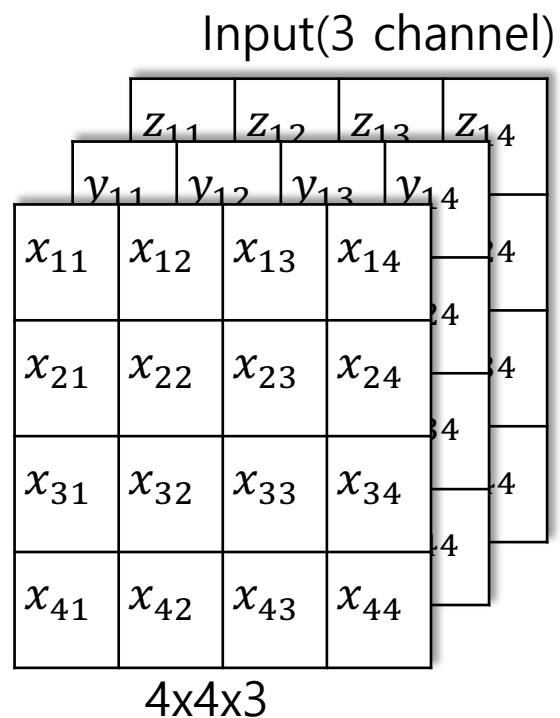
Average pooling



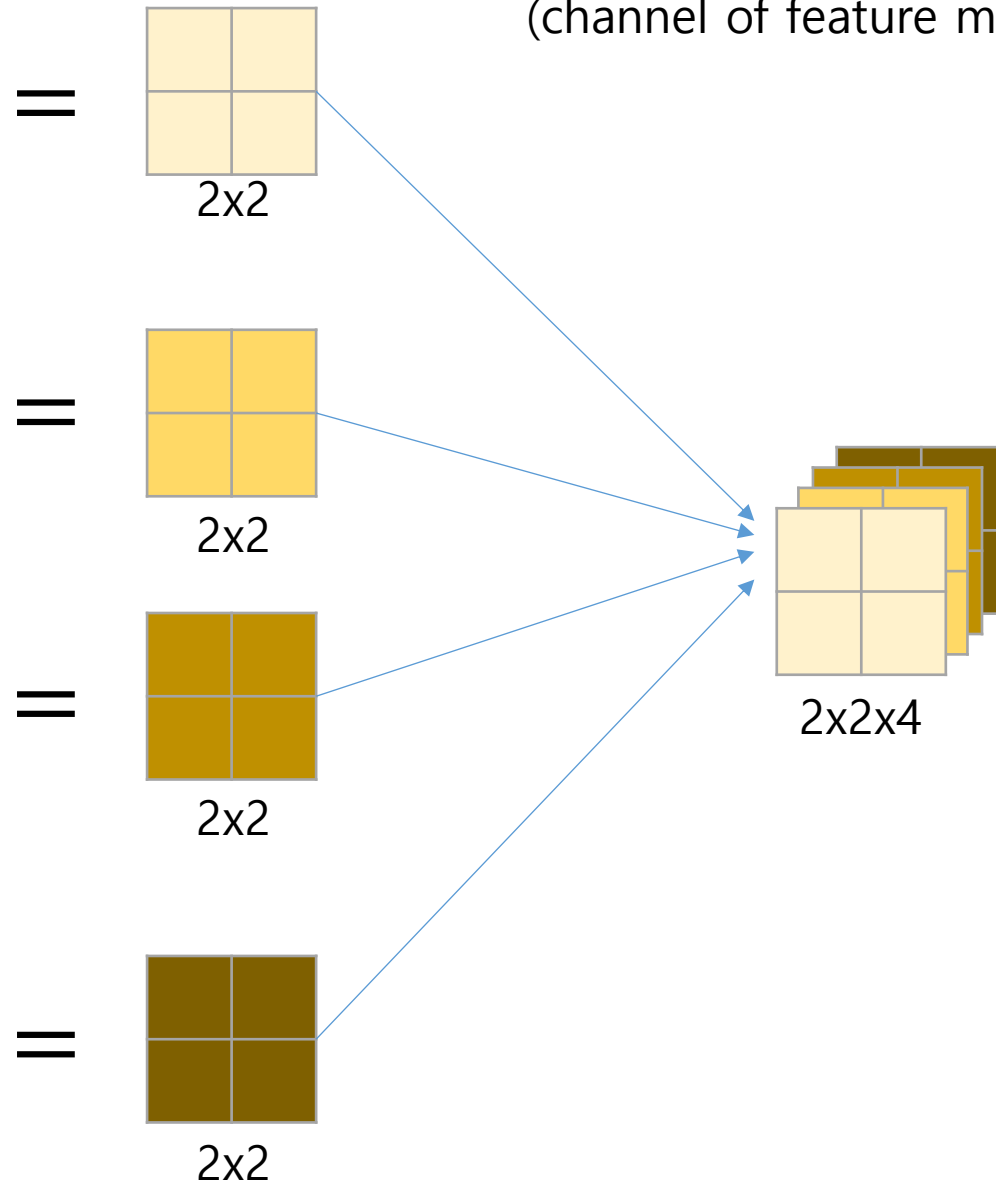
1.75	1	1.5
2	1.75	2.5
1.25	3.5	1.75

CNN 개념

Channel



- Number of filter = channel of output
(channel of feature map)



Output size 계산

입력 데이터 높이 : H

입력 데이터 너비 : W

필터 높이 : FH

필터 너비 : FW

Stride : S

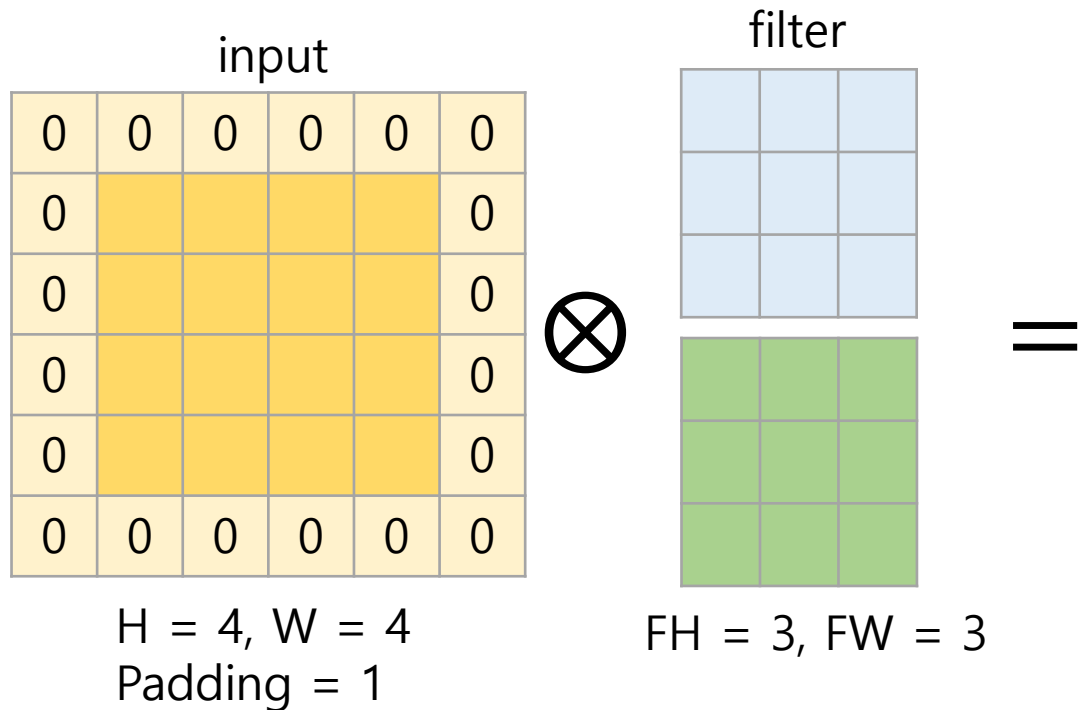
Padding size : P

Number of filter = $\#F$

$$\text{출력 데이터 높이} = OH = \frac{H+2P-FH}{S} + 1$$

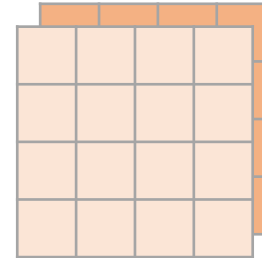
$$\text{출력 데이터 너비} = OW = \frac{W+2P-FW}{S} + 1$$

$$\text{출력 데이터 채널} = \#F$$



output

$S = 1 \rightarrow$

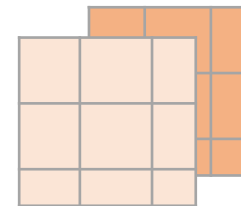


$$OH = \frac{4+2 \times 1 - 3}{1} + 1 = 4$$

$$OW = \frac{4+2 \times 1 - 3}{1} + 1 = 4$$

$$\text{Output channel} = \#F = 2$$

$S = 2 \rightarrow$

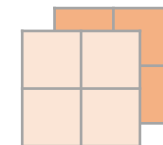


$$OH = \frac{4+2 \times 1 - 3}{2} + 1 = 2.5 \text{ (불가능)}$$

$$OW = \frac{4+2 \times 1 - 3}{2} + 1 = 2.5 \text{ (불가능)}$$

(Filter size가 자연수가 나오도록 조절해야 함)

$S = 1 \rightarrow$



$$OH = \frac{4+2 \times 1 - 3}{3} + 1 = 2$$

$$OW = \frac{4+2 \times 1 - 3}{3} + 1 = 2$$

$$\text{Output channel} = \#F = 2$$

CONTENTS

—

① CNN 개요

② CNN 구조

③ CNN 개념

④ CNN 이해

CNN 이해

📝 Convolution 어떤 특징점을 찾을까?

- MNIST 예제를 이용해 3가지 필터를 가정해 Convolution 연산을 해보고 결과값을 비교해보자

Input : 6x6

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

가로 필터

1	1	1
0	0	0
0	0	0

대각선 필터

0	0	1
0	1	0
1	0	0

세로 필터

0	0	1
0	0	1
0	0	1

결과값

1	2	2	1
1	1	1	1
0	0	1	1
0	1	1	1

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

1	1	2	0
0	1	2	0
1	1	1	0
2	2	1	0

CNN 이해

🖋 Convolution 어떤 특징점을 찾을까?

- 가로 필터를 활용한 연산 결과

Input : 6x6

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



Filter: 3x3

1	1	1

=

Output: 4x4

1	2	2	1
1	1	1	1
0	0	1	1
0	1	1	1

+

bias

-1

=

Feature Map: 4x4

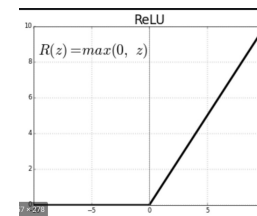
0	1	1	0
0	0	0	0
-1	-1	0	0
-1	0	0	0

Max pooling output

1	1
0	0

Relu output

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



CNN 이해

📝 Convolution 어떤 특징점을 찾을까?

- 대각선 필터를 활용한 연산 결과

Input : 6x6

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



Filter: 3x3

0	0	1
0	1	0
1	0	0

=

Output: 4x4

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

+

bias

-1

=

Feature Map: 4x4

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



Relu output

1	0	0	0
0	0	0	1
0	0	2	0
0	2	0	0



Max pooling output

1	1
2	2

CNN 이해

📝 Convolution 어떤 특징점을 찾을까?

- 세로 필터를 활용한 연산 결과

Input : 6x6

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



Filter: 3x3

0	0	1
0	0	1
0	0	1

=

Output: 4x4

1	1	2	0
0	1	2	0
1	1	1	0
2	2	1	0

+

bias

-1

=

Feature Map: 4x4

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



Relu output

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



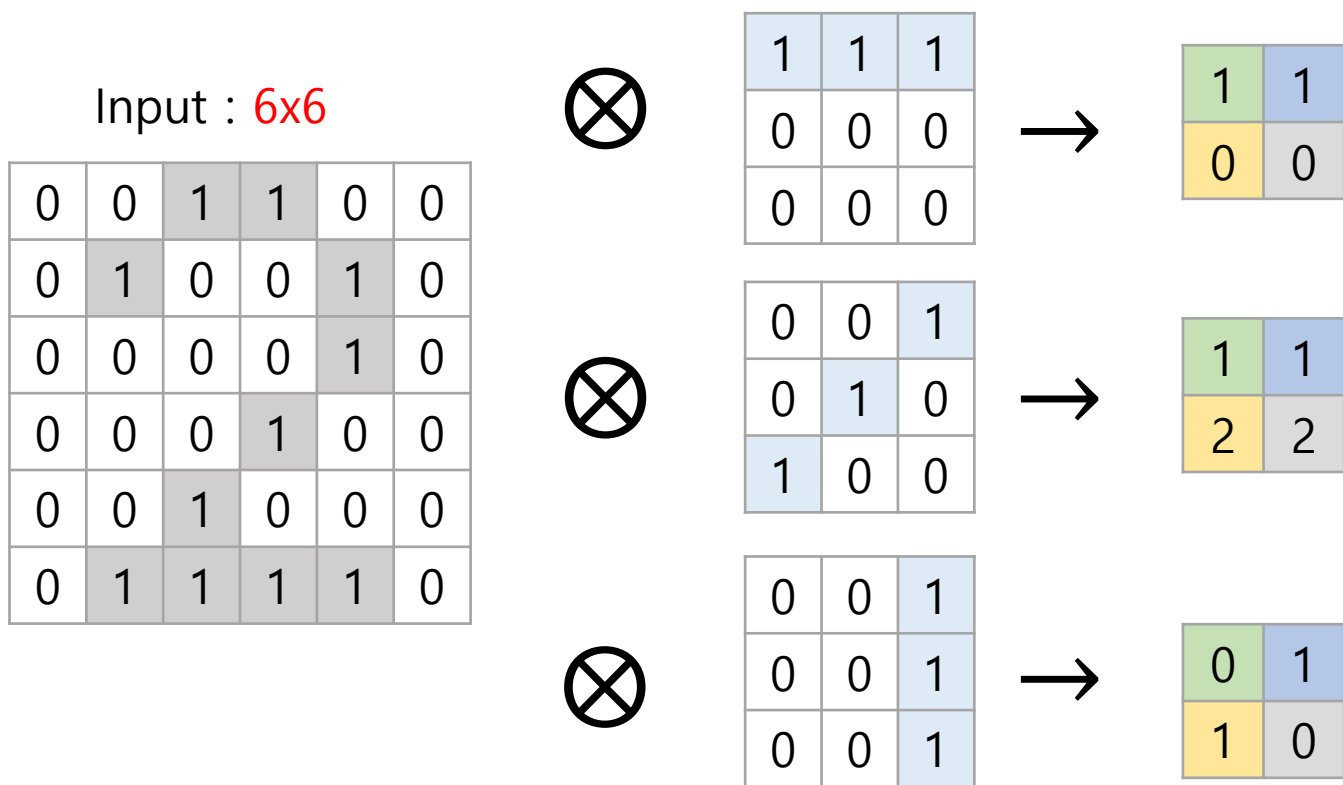
Max pooling output

0	1
1	0

CNN 이해

🖋️ Convolution 어떤 특징점을 찾을까?

- 대각선 필터 값이 가장 높다! 대각선 필터의 결과값이 높은 입력은 숫자 "2"일 가능성이 높겠네?
- 반대로 말하면 "2"의 **한** 가지 **특징점**을 찾았다



CNN 이해

📝 Convolution 어떻게 특징점을 찾을까?

- 앞선 예제는 세로 필터를 사람이 직접 선택했다. 엉? ➔ 명시적 프로그래밍
- 하지만, 필터는 weight parameter라고 했으니 예측 결과와 정답을 비교하여 필터값 업데이트 가능!
- 여러개의 필터를 랜덤으로 초기화 후 각 필터들이 task(MNIST 숫자)를 잘 구분하도록 업데이트 됨
- 결과적으로 입력값에 대한 특징점을 잘 찾는 필터값을 찾게 된다는 이야기!

Input : 6x6

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



0	0	1
0	1	0
1	0	0



1	1
2	2

CONTENTS

① CNN 개요

② CNN 구조

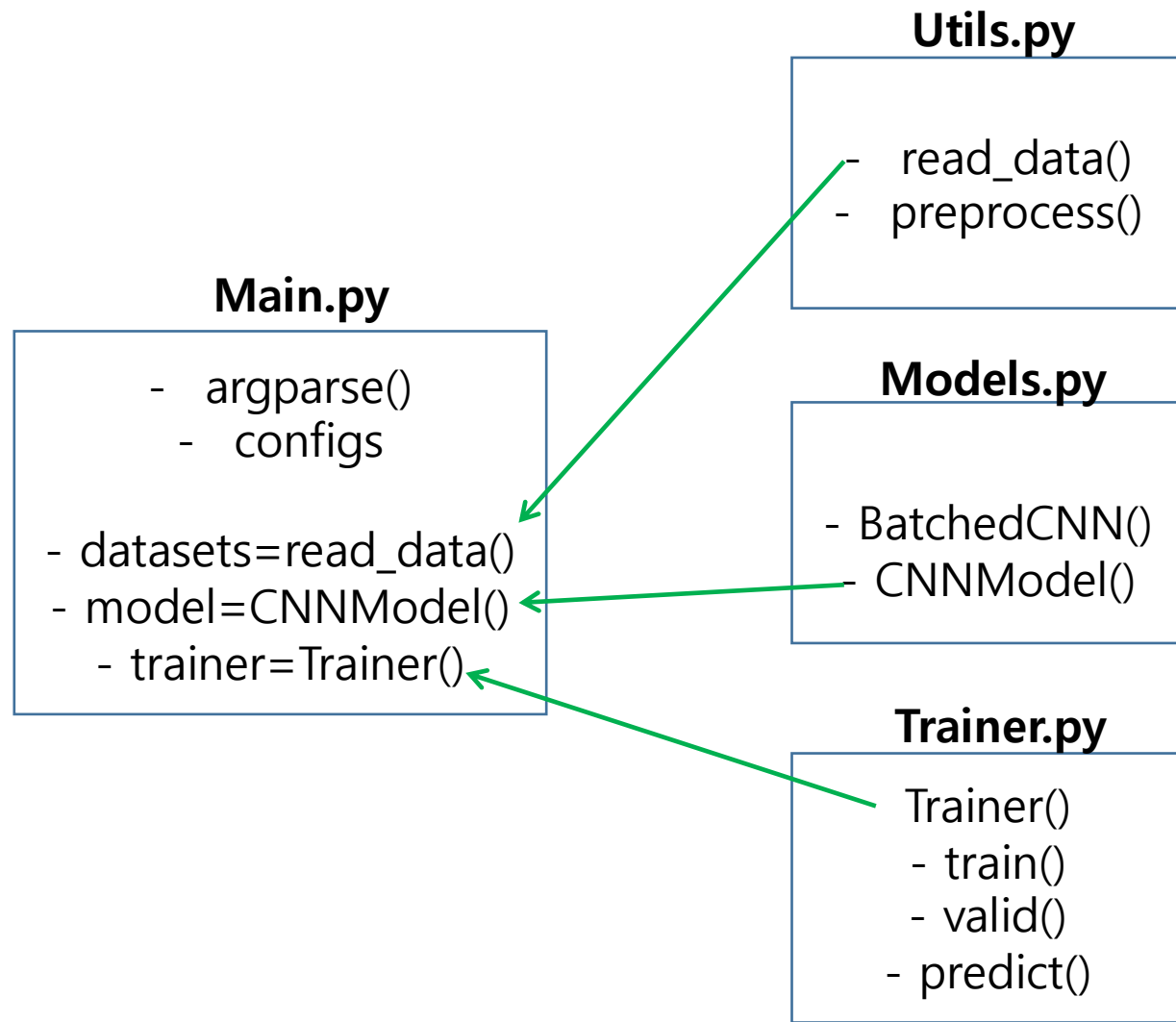
③ CNN 개념

④ CNN 이해

⑤ CNN 구현

CNN을 이용한 MNIST 구현

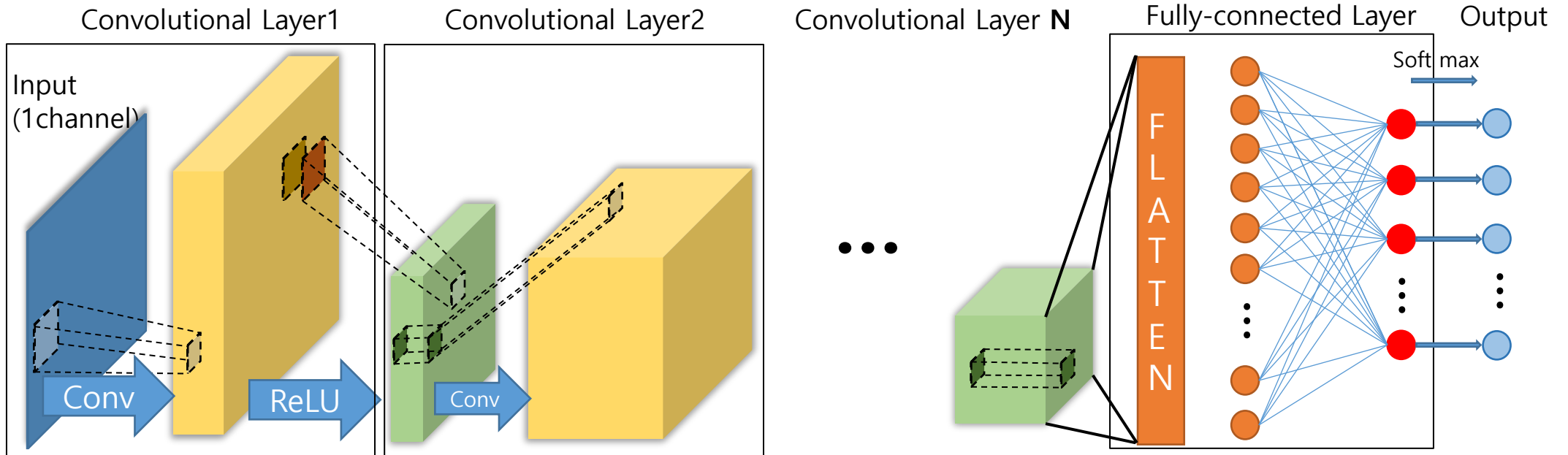
코드의 전체 구조



CNN 구현

Model 구조

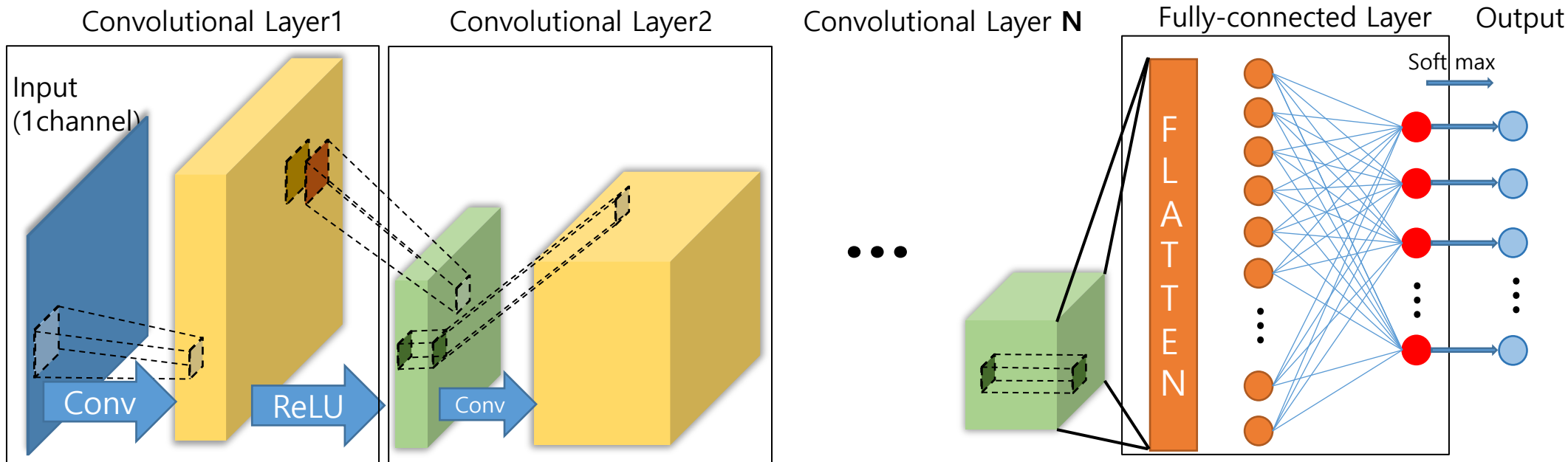
```
class BatchedCNNLayer(nn.Module):  
  
    def __init__(self, input_channel, output_channel):  
        self.input_channel = input_channel  
        self.output_channel = output_channel  
        super().__init__()  
  
        self.layer = nn.Sequential(  
            nn.Conv2d(self.input_channel, self.output_channel, kernel_size=(3,3), padding=1),  
            nn.ReLU(),  
            nn.Conv2d(self.output_channel, self.output_channel, kernel_size=(3,3), stride=2, padding=1),  
            nn.ReLU()  
        )  
  
    def forward(self, input_data):  
        return self.layer(input_data)
```



CNN 구조

📝 기본적인 CNN 구조

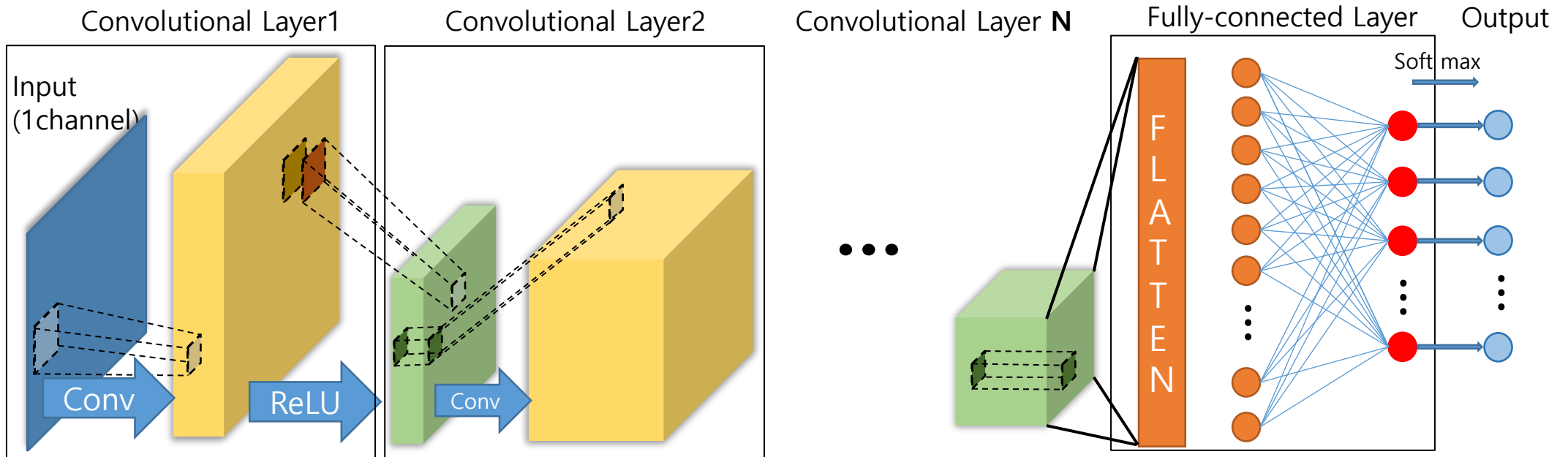
```
class CNNModel(nn.Module):  
  
    def __init__(self, num_classes):  
        self.num_classes = num_classes  
        super().__init__()  
  
        self.cnn_encoder = nn.Sequential(  
            BatchedCNNLayer(1, 32),  
            BatchedCNNLayer(32, 64),  
            BatchedCNNLayer(64, 128),  
            BatchedCNNLayer(128, 256),  
            BatchedCNNLayer(256, 512),  
        )
```



CNN 구조

✎ 기본적인 CNN 구조

```
class CNNModel(nn.Module):  
  
    def __init__(self, num_classes):  
        self.num_classes = num_classes  
        super().__init__()  
  
        self.cnn_encoder = nn.Sequential(  
            BatchedCNNLayer(1, 32),  
            BatchedCNNLayer(32, 64),  
            BatchedCNNLayer(64, 128),  
            BatchedCNNLayer(128, 256),  
            BatchedCNNLayer(256, 512),  
        )  
  
        self.classifier = nn.Sequential(  
            nn.Linear(512, 200),  
            nn.ReLU(),  
            nn.Linear(200, self.num_classes)  
        )
```





감사합니다.