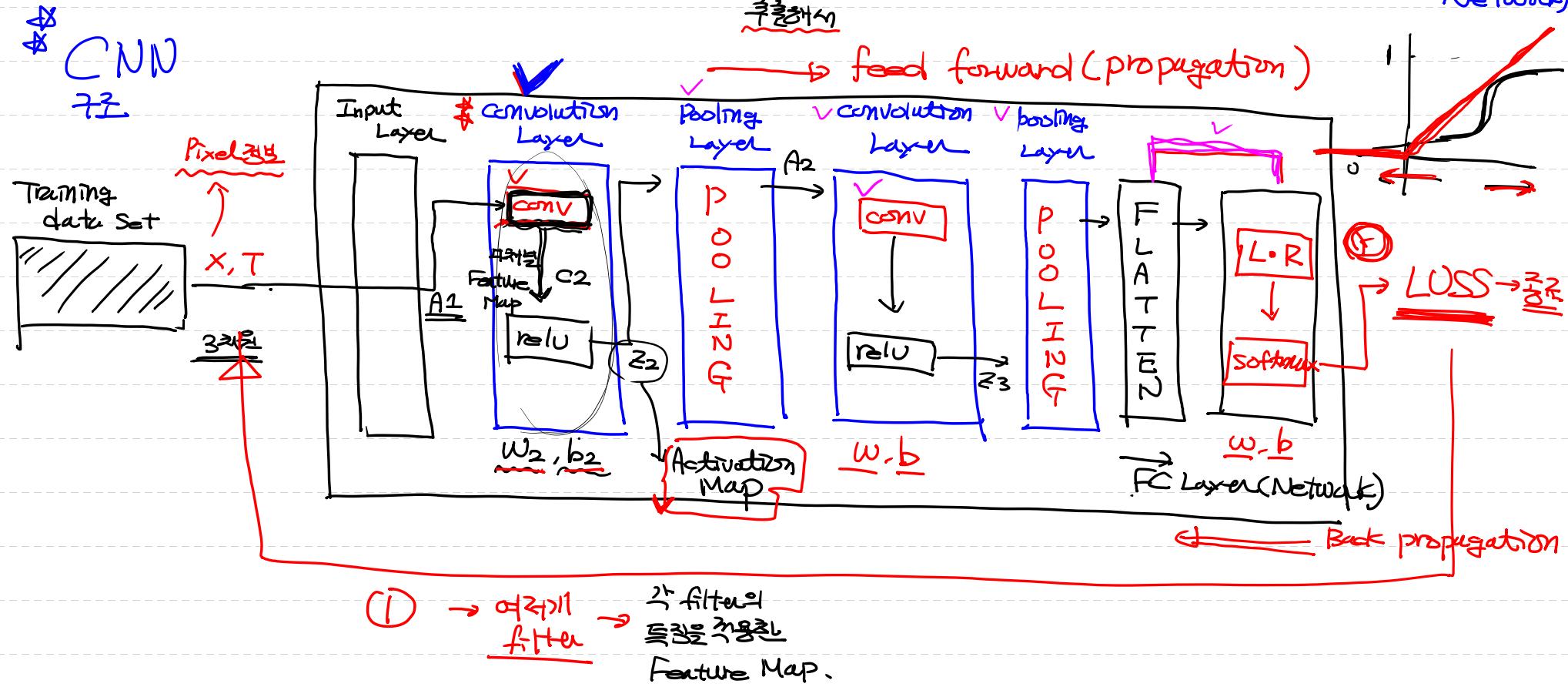


\* 03/22 기존 DNN을 이용해 Image를 학습하는데 예전까지 한계점이 존재  
 → 사용이 그림을 파열화도 입지의 특징을 <sup>↑</sup>학습하는데 예측하는 한계  $\Rightarrow$  CNN (Convolutional Neural Network)



• Convolution (컨볼루션) ✓

✓ → 총선곱 연산을 두 함수  $f, g$ 가 합쳐질 때

하나의 출력을 만들 (reverse), 쪼이 (shift) 시킨 후 다른 출력과 합해요. → 이 결과를 합집합

✓ Image data

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

✓ Filter (kernel)

1	0	1
0	1	0
1	0	1

$$\begin{aligned}
 & (1 \times 1) + 1 \times 0 + 1 \times 1 \\
 & + \\
 & 0 \times 0 + 1 \times 1 + 1 \times 0 \\
 & + \\
 & 0 \times 1 + 0 \times 0 + 1 \times 1 \\
 & \Downarrow \\
 & 4
 \end{aligned}$$

→ 같은 위치에 있는 요소끼리 곱한 뒤 그 결과값들의 합집합

- channel

→ 이미지 pixel의 차이차이나는 풍연히 측정함.

↳ color인 경우 각 pixel은 RGB 3개의 실수로 표현

→ color 이미지는 3차원으로 표현 (R,G,B 3개의 channel로 구성)

→ gray-scale(흑백) 이미지는 2차원 데이터로 표현이 가능

→  $(\text{height}, \text{width}, 1) \rightarrow$  3차원 (channel: 1) ] \*\*  
[  $(\text{height}, \text{width}, 3) \rightarrow$  3차원 (channel: 3)  
[  $(\text{height}, \text{width}) \rightarrow$  2차원 ]

- 일반적으로 이미지를 처리할 때 전처리 과정을 통해 gray-scale 이미지로 변환 후 처리  
즉, channel을 "1"로 만들어서 처리



- Filter & stride

CNN이란 ~~Filter~~ <sup>\*</sup> ~~Filter~~ → 이미지의 특징을 찾아내기 위한 공통 parameter

이 Filter를 다른말로 kernel

⇒ 일반적으로  $3 \times 3$ ,  $4 \times 4$ 의 같은 전통적인 형태로 전파

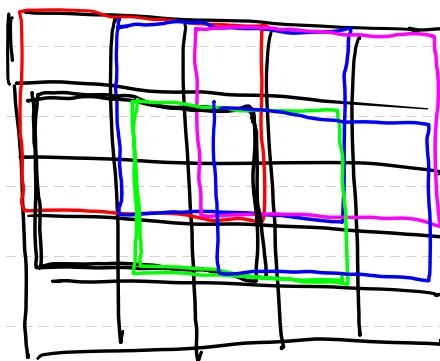
이 Filter의 구성요소들이 결국 CNN에서 핵심의 대상이에요!!!

\* 이걸 Filter를 몇개만 사용하나요?? → 이미지의 특징들을 과정에 활용하기 위해 여러개의 Filter를 사용

이걸 필터는 크기가 큰 Filter보다 크기가 작은 Filter여러개를 사용하는게 더 좋았어요.

↳ 결국  
 $3 \times 3$  size Filter를 이용.

Image data (Input data)

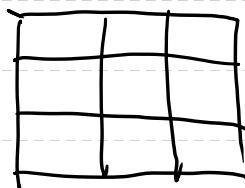
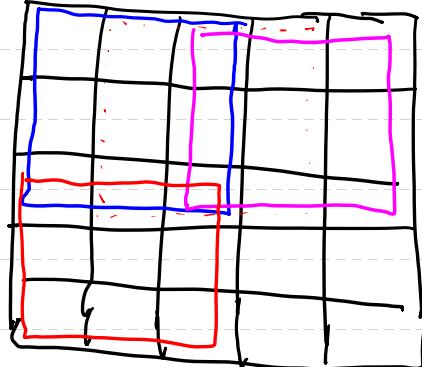


(filter)  
kernel

Filter를 이미지의 이미지에 직접적 간접모습  
이동하면서 convolution 수행

Scalor  
"Stride"

• Image data (5x5) Filter (3x3) Stride : 2 ↗ stride 값이 커지면 convolution은 결과의 개수가 줄어들어요!



if stride → 3 ? 사용할 3 없어요!!

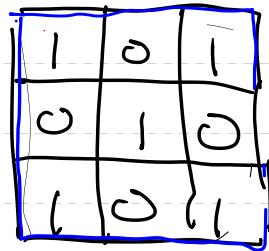
✓ Stride = 1

결과물 (3x3) ↗ 9개

Image data (5x5)

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

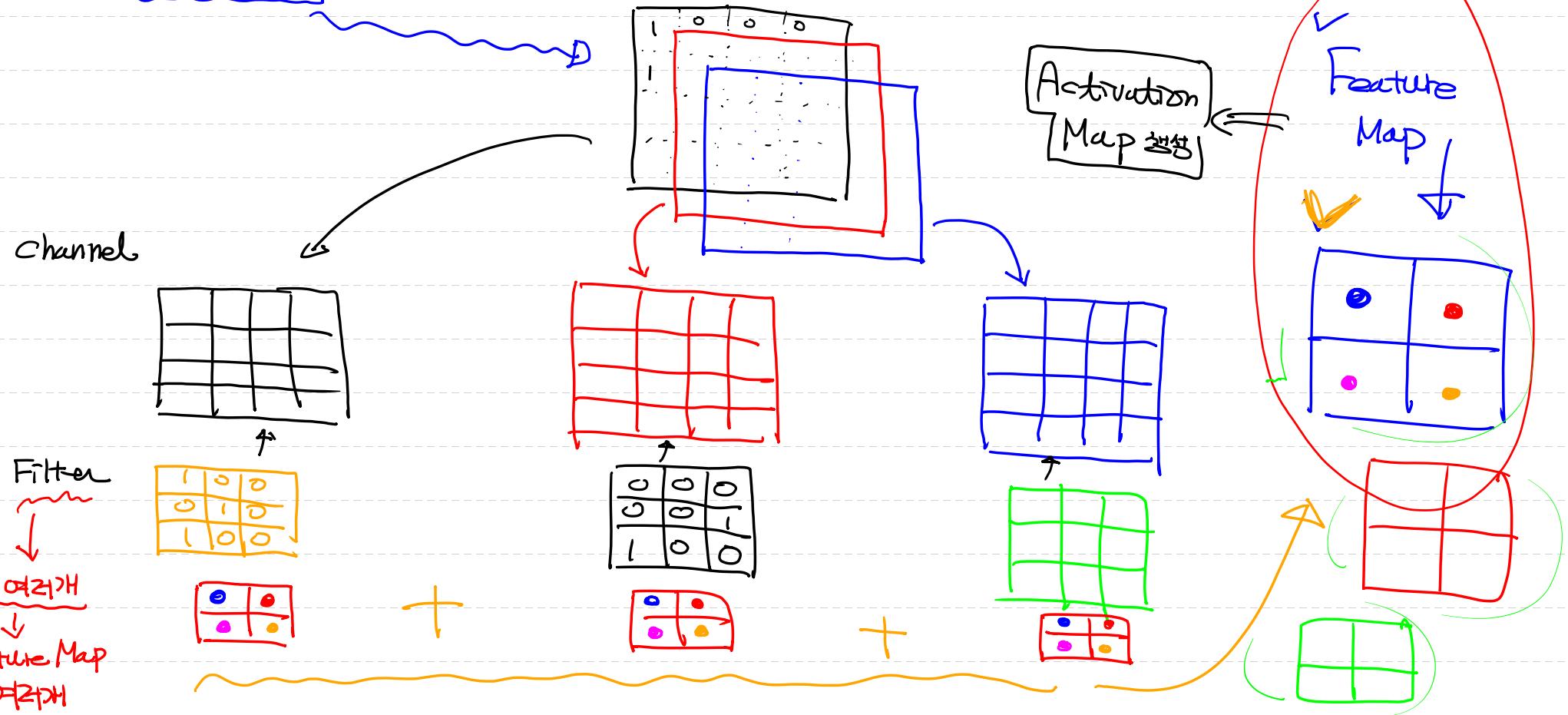
✓ Filter (3x3)



↙ Feature Map

4	3	4
2	...	...
...	...	...

- 이미지의 크기 ( $4 \times 4$ ) ]을 입력으로 Feature Map을 생성  $\Rightarrow$  (filter (3 channel))  
이미지의 channel 3



· Padding  $\Rightarrow$  입력이미지  $\rightarrow$  [Stride  
Filter]  $\rightarrow$  Feature Map  
 ↴ 입력데이터보다  
 크기가 작아져요.  $\rightarrow$  짜작적으로 레이어 작아지는걸  
 보려는 경향  $\Rightarrow$  Padding

Padding  $\rightarrow$  입력데이터 외곽에 "0"으로 채워서 처리

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

Zero padding  
 ↴ 득점에 영향지 주지 않기 때문에.

code로 처리할 때 (TF 1, x)

↗  
 Padding  
 2가지 option  
 VALID : Padding X  $\rightarrow$  입력보다 출력결과  
 크기가 작아요!!!  
 SAME : Padding O  $\rightarrow$  입력과 출력의 크기가  
 같도록 생성!!!

## • "pooling"

Stride와 kernel size에 따라 Feature Map의 size가 결정!

일반적으로 크기로 흘러들어오( no padding ) + 만약 Filter를 여러개 사용하면 Feature Map의

→ 실제 학습해야 하는 계산량을 많아지게 되요!!

→ ~~특정~~ feature를 강조 + 계산량을 줄이기 위해 Pooling을 사용.

Feature Map

MAX  
Average  
MIN

개수가 증가 (channel이 증가)

kernel size ( $2 \times 2$ ) ↑

