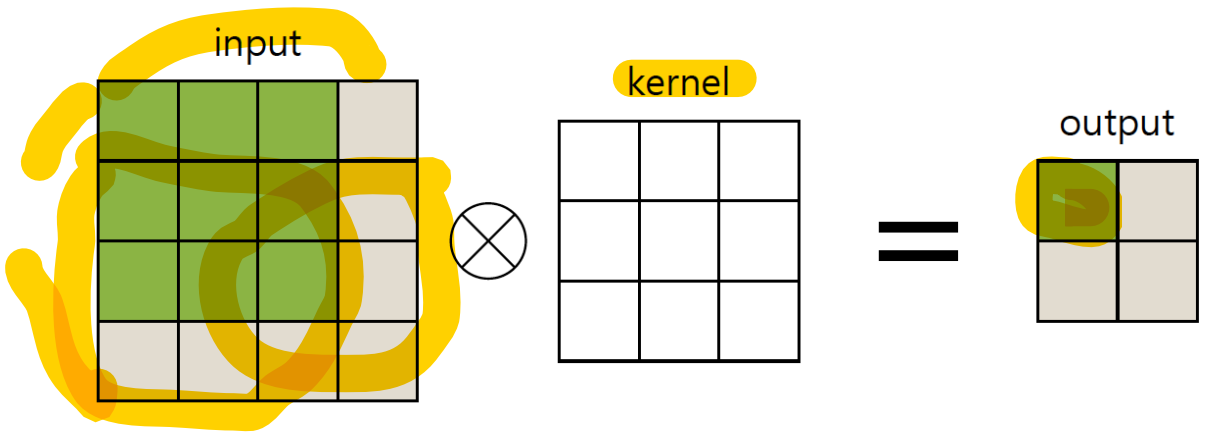
CNN

Convolution Newral Network

▼ Convolution Filter

> **input** : 4 x 4, **kernel** : 3 x 3, **output** : 2 x 2

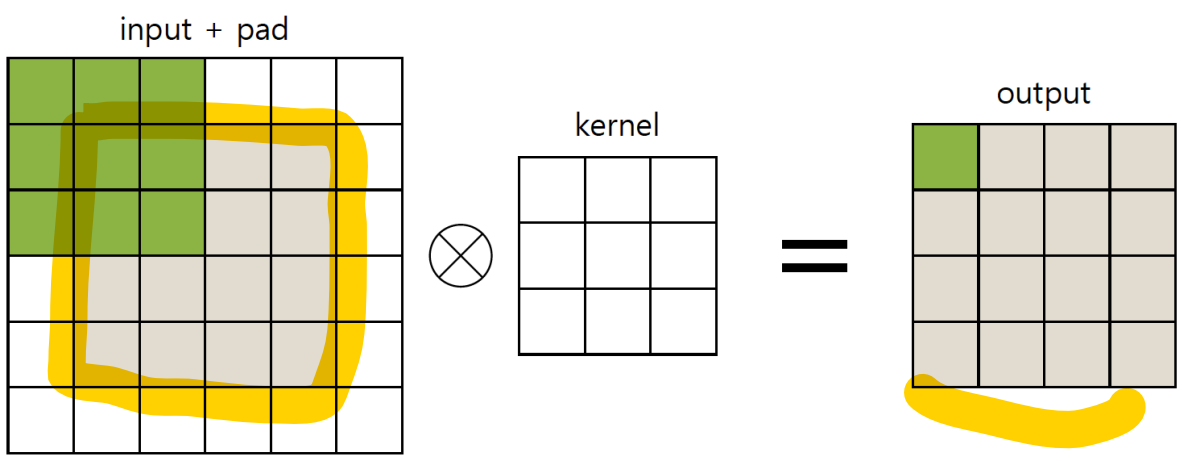
- kenel은 일반적으로 3 x 3 사용



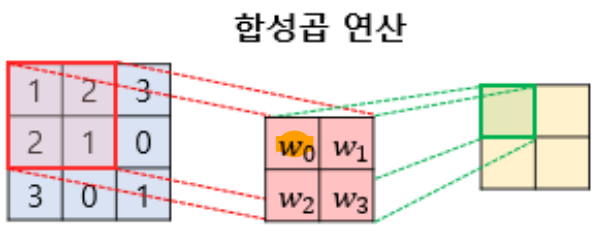
▼ Convolution Filter with PAD

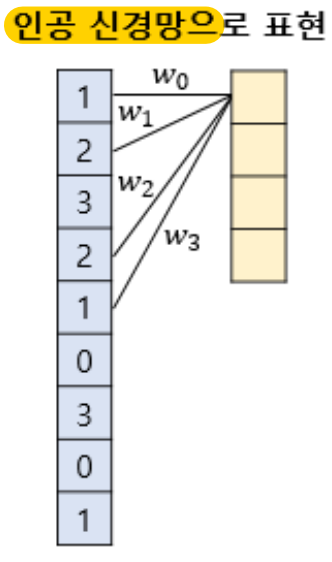
> 원본의 input인 4 x 4를 output으로 유지

- **input** : 6 x 6(with pad), **kernel** : 3 x 3, **output** : 4 x 4



> 다중퍼셉트론의 가중치(Weight)는 CNN에서는 Kernel
> → 압축된 정보를 담고 있다. 즉, 지역정 정보를 가지고 있다





▼ **How it Works**

> PhotoApp에서 사용하는 filter

0	-1	0
-1	5	-1
0	-1	0

sharpen filter

0.0625	0.125	0.0625
0.125	0.25	0.125
0.0625	0.125	0.0625

blur filter

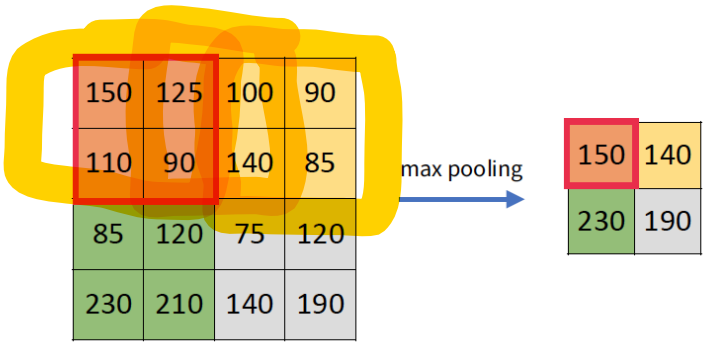
HeatMap

Max Pooling & Stride

▼ **Max Pooling & Stride**

> (b, C, W, H) 중 W, H를 줄이는 것은 MaxPooling과 Stride가 있기 때문인다.
> → **목적 : Parameters를 줄이기 위함**

- Pooling : Max, Average 등을 사용
- Stride : 이동범위를 넓게 함



Input & Output Tensors(In Image)

▼ **Tensor 모양 (N,C,H,W)**

> **★☆★ Channels이 Filters으로 변환된다**
> **N = Batch Size**

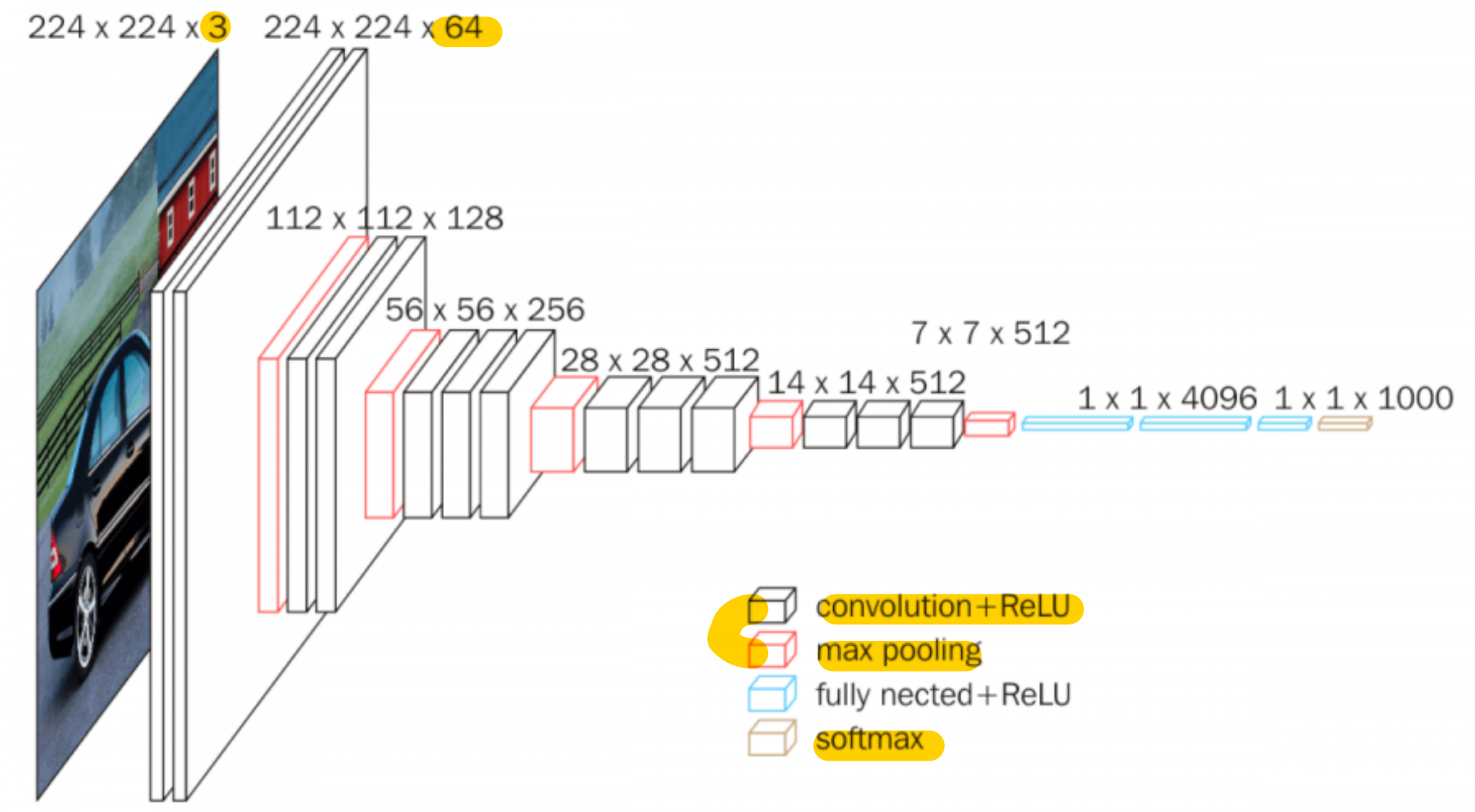
- Channels이 3개(RGB), Filters가 5개로 하는 것 처럼
  → Filter 1개가 각각의 Channel마다 Weighted Sum을 적용하여 Channels의 값들은 압축된다.
- 지역정보를 압축해서 하나의 Scalar값으로 표현

▼ ***CNN Architecture***

> CNN Block

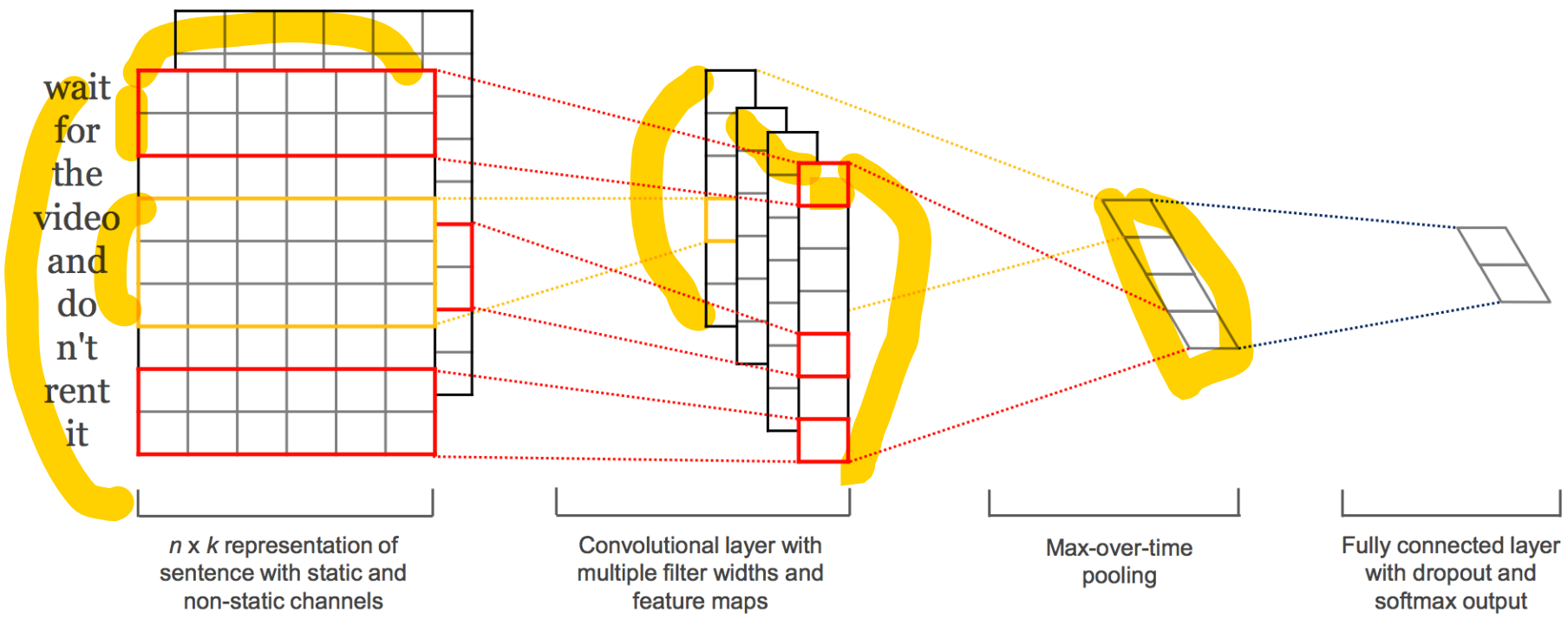
- Convolution Layer

- Relu
- **VGG16 Architecture** (Source: https://neurohive.io/en/popular-networks/vgg16/)
  (w, h, c)



Input & Output Tensors(In NLP)

▼ Text Classification



- n은 문장의 길이, k는 임베딩 차원
- Conv1D에서는 높이만 지정 넓이는 임베딩 차원

▼ 입출력 계산 방법

$$|x| = (\ b,\ C_{in},\ x_{height},\ x_{width}\ )$$
$$|y| = (\ b,\ C_{out},\ x_{height} - k_{height} + 1,\ x_{width} - k_{width} + 1\ )$$
$$|y| = (\ b,\ C_{out},\ x_{height} + 2 * P_{height} - k_{height} + 1,\ x_{width} + 2 * P_{width} - k_{width} + 1\ )$$

> ★☆★ K_{size}가 3*3이고 P_{size}가 1인 경우에는 $|x|_{height}$와 $|y|_{height}$가 동일

- 샘플계산) input $|x|_{height}$10+2*P_{size}1-k_{height}3+1 = 10

Convolution Layer의 특징

▼ 속도, 성능

> 패턴 인식에 탁월함으로 Computer Vision뿐만 아니라 NLP에서도 사용함
> 적은 Parameter 사용으로 연산이 적어 속도가 빠름

- **위치에 구애 받지 않고 패턴을 인식한다.**

활용분야

▼ Computer Vision, Speech Recognition, Text Classification, Time Series