



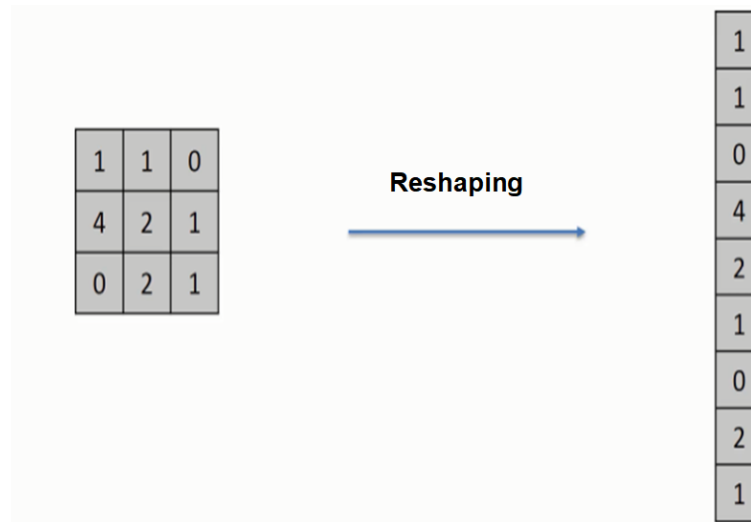
# Convolutional Neural Network

---

Sang Yup Lee

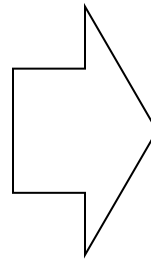
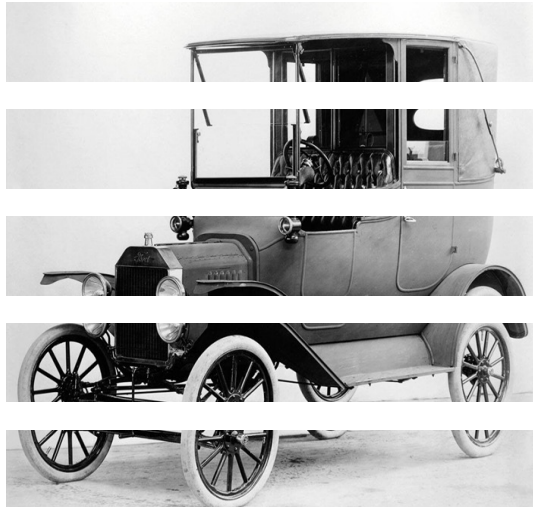
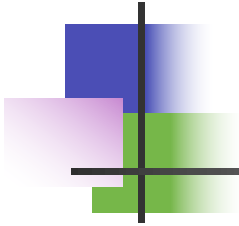
# CNN (합성곱 신경망)

- FNN에서의 이미지 데이터 처리
  - 흑백이미지의 경우



- 단점
  - 이웃 픽셀 정보(공간정보)를 잘 활용하지 못한다.
  - 파라미터의 수가 많아진다.







# CNN

---

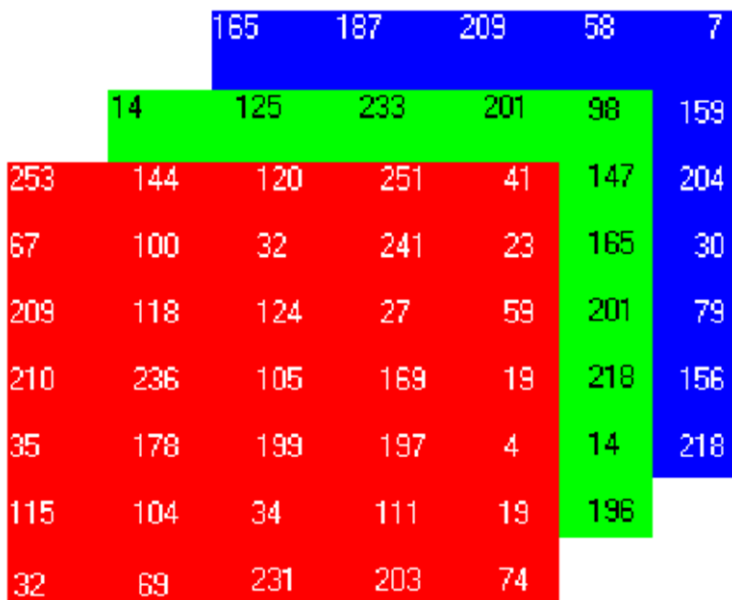
- 장점

- 데이터가 가지고 있는 spatial 정보를 추출하는데 유리
- 이미지 데이터 분석에 주로 적용
- FNN 적용시, image 정보를 vector (즉, 1D array)로 표현하게 되는데, 이렇게 함으로써 spatial 정보를 잃게 된다.

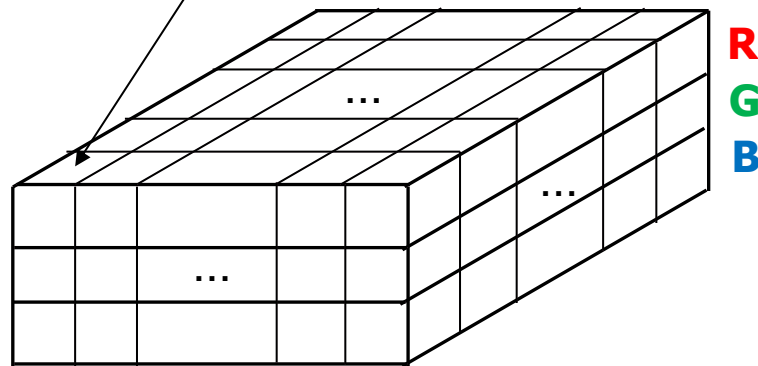
# CNN

- 칼라 이미지

- 칼라 이미지의 경우, 하나의 픽셀이 보통 3개의 색 정보를 지님 (Red, Green, Blue, a.k.a., RGB; 이를 채널이라고 함  $\Rightarrow$  채널수=depth=3 )
  - 3차원 array로 저장됨  $\Rightarrow$   $m \times n$  칼라 이미지의 경우,  $m \times n \times 3$ 의 3차원 array로 데이터 저장



각 셀이 하나의 색상 정보를 지님  
각 셀이 하나의 feature (독립변수)의 값이 됨





# CNN

## ■ CNN

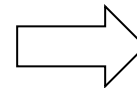
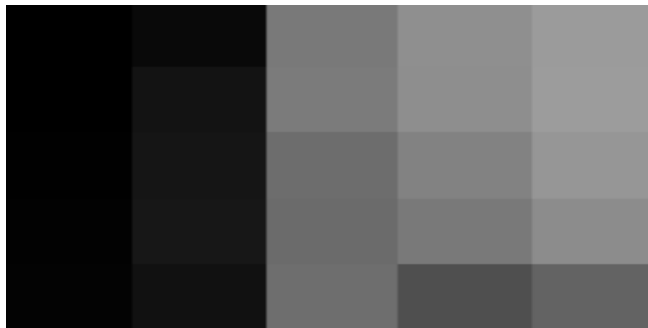
- 그렇다면, 어떻게 공간적인 정보를 추출하는가? 또는 파라미터수를 줄일 수 있는가?
- 이미지의 정보를 추출하기 위해 filter를 적용
  - 하나의 필터를 옆으로 (아래로) 이동시키면서 정보를 추출
- filter의 크기는  $F \times F \times D$ 
  - $D$ =입력이미지의 depth (= 채널 수)
  - $F$ =가로 또는 세로의 길이
  - 흑백 이미지:  $D=1$ 
    - 즉,  $F \times F \times 1$  필터 사용
  - 칼라 이미지:  $D=3$ 
    - 즉,  $F \times F \times 3$  필터 사용
- filter의 각 셀은 고유의 가중치를 갖음

depth = 1인 필터

w11	w12	w13
w21	w22	w23
w31	w32	w33

# CNN의 작동원리

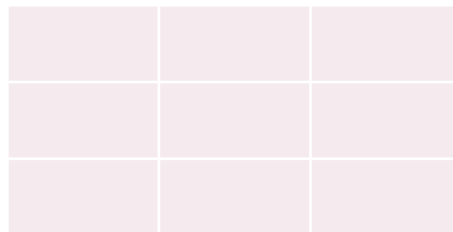
- 흑백 이미지에 적용하는 경우
  - 크기: 5x5 pixels



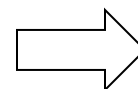
1	10	122	143	155
1	20	124	142	156
2	22	110	130	150
3	24	108	122	140
4	18	111	80	100

- Filter의 크기: 3x3

kernel이라고도 불림



필터의 경우, 각 셀이 가중치를 지님



w11	w12	w13
w21	w22	w23
w31	w32	w33



# CNN의 작동원리

- 이미지의 왼쪽 위에서부터 filter를 적용
- 옆과 아래로 순차적으로 이동 (한번에 몇칸을 이용하는지는 사용자가 결정)

w11	w12	w13
w21	w22	w23
w31	w32	w33



1	10	122	143	155
1	20	124	142	156
2	22	110	130	150
3	24	108	122	140
4	18	111	80	100



1	10	122	143	155
1	20	124	142	156
2	22	110	130	150
3	24	108	122	140
4	18	111	80	100



1	10	122	143	155
1	20	124	142	156
2	22	110	130	150
3	24	108	122	140
4	18	111	80	100



1	10	122	143	155
1	20	124	142	156
2	22	110	130	150
3	24	108	122	140
4	18	111	80	100

...



1	10	122	143	155
1	20	124	142	156
2	22	110	130	150
3	24	108	122	140
4	18	111	80	100

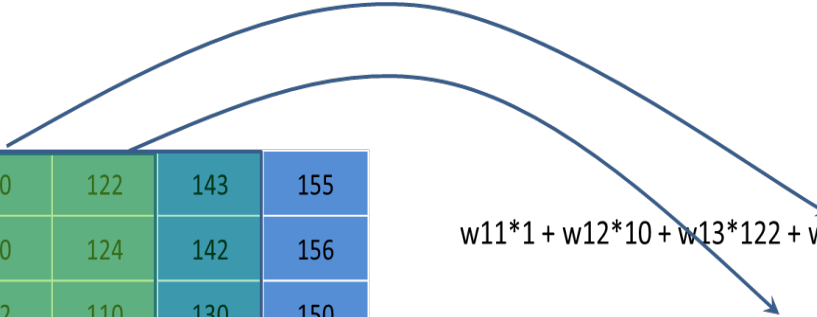
CNN

# CNN 작동 원리

즉, 같은 자리에 있는  
셀의값(원소)끼리 곱해서 더한다.

## ■ Filter 적용

- Filter 적용시, filter가 적용되는 이미지 부분의 색 정보와 filter의 각 가중치 간에 내적 연산 (하나의 스칼라값 도출), 이를 합성곱이라고 함



1	10	122	143	155
1	20	124	142	156
2	22	110	130	150
3	24	108	122	140
4	18	111	80	100

$$w_{11} \cdot 1 + w_{12} \cdot 10 + w_{13} \cdot 122 + w_{21} \cdot 1 + \dots + w_{33} \cdot 110$$

$$w_{11} \cdot 10 + w_{12} \cdot 122 + w_{13} \cdot 143 + w_{21} \cdot 20 + \dots + w_{33} \cdot 130$$



# CNN 작동 원리

---

- 내적 연산의 결과
  - 앞과 같은 내적 연산(i.e., 합성곱)을 통해서 계산된 값에 bias를 더하여 아래와 같이 표현
  - $z_{11} = w_{11} * 1 + w_{12} * 10 + w_{13} * 122 + w_{21} * 1 + \dots + w_{33} * 110 + b$
  - 필터 하나당 bias 한개 존재
  - $z_{11}$ 에 특정 활성화 함수를 적용 (CNN에서는 보통 relu를 사용)
    - $f(z_{11})$

# CNN 작동 원리

## ■ 또 다른 예

1	10	122	143	155
1	20	124	142	156
2	22	110	130	150
3	24	108	122	140
4	18	111	80	100

⊗

w11	w12	w13
w21	w22	w23
w31	w32	w33



z11	z12	z13
z21	z22	z23
z31	z32	z33



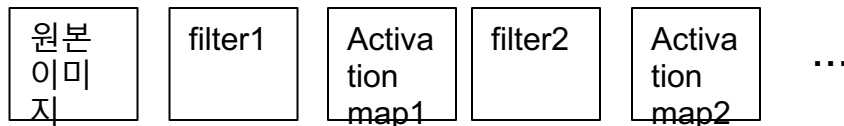
f(z11)	f(z12)	f(z13)
f(z21)	f(z22)	f(z23)
f(z31)	f(z32)	f(z33)

- 활성화함수 f() 적용
- 이미지의 경우는 보통 relu 사용

Activation map => 활성화 함수를 적용하여 출력되는 값들을 모아놓은 것, feature map 또는 response map 이라고도 함

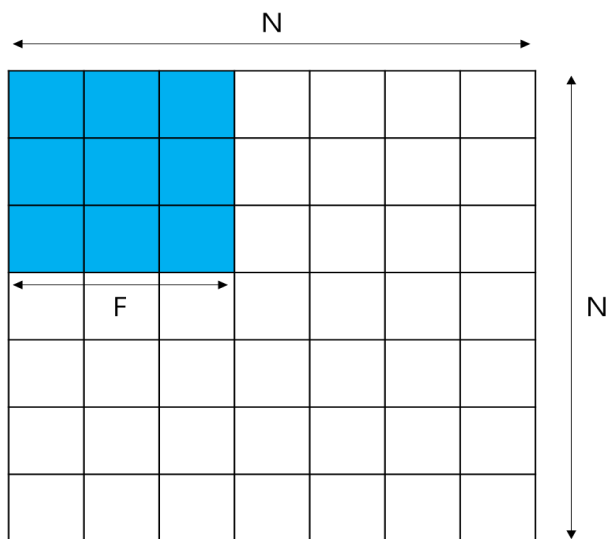
# CNN 작동 원리

- Activation map을 또 하나의 이미지라고 생각할 수 있음
  - 즉, Activation map에 또 다른 필터를 적용할 수 있다는 것을 의미



# CNN 작동 원리

## ■ Activation map의 크기



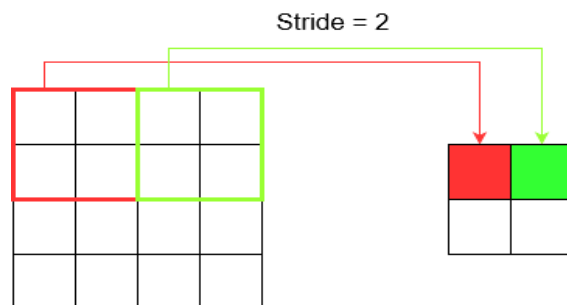
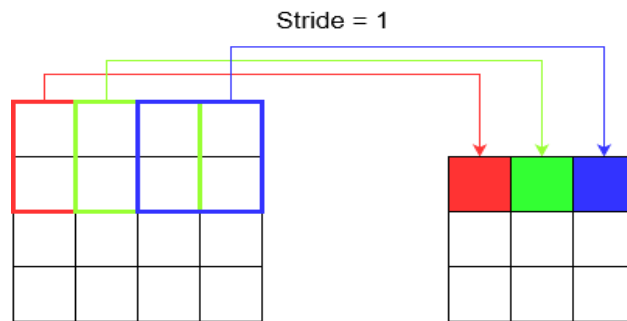
예를 들어,  $N \times N$  이미지에  $F \times F$  filter를 (왼쪽 그림 참조) 적용시키게 되면,  $(N-F)/\text{stride} + 1$  이 됩니다. . F

앞의 예에서는 한번에 1칸 이동했기 때문에  $\text{stride} = 1$  이었습니다. 즉, 위의 예에서는  $N = 5$ ,  $F = 3$ ,  $\text{stride} = 1$ 이었으므로,  $(5-3)/1 + 1 = 3$ 이 나온 것입니다. 즉, 결과로 나온 activation map의 크기가  $3 \times 3$ 이 됩니다.

# CNN 작동 원리

- 추가 용어: Stride

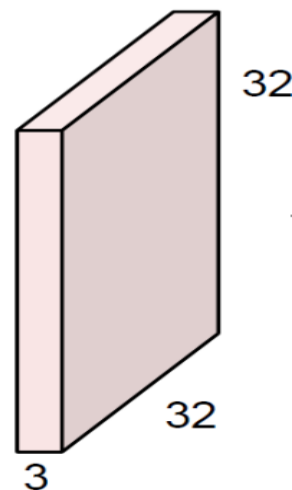
- stride는 filter를 한번에 옆으로 (혹은 아래로) 얼마나 움직이는지를 의미 (이는 hyperparameter)



Mainly for downsampling  
파라미터의 수 감소  
학습 속도 개선  
하지만, 커지면 정보 손실 =>  
성능 저하

# CNN 작동 원리

- 컬러이미지 (RGB)
  - 32x32 pixel 이미지의 경우
    - channel의수 (즉, depth) = 3
  - Filter의 크기
    - 이러한 경우에는 동일한 깊이의 filter를 사용, 위의 경우, FxFx3
    - 사용자는 filter의 가로 (또는 세로)의 크기만을 결정



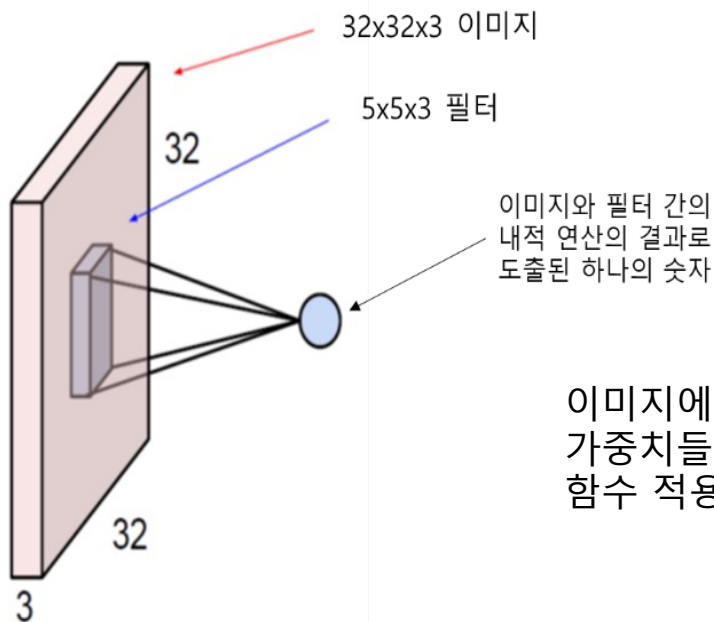


# CNN 작동 원리

## ■ Example

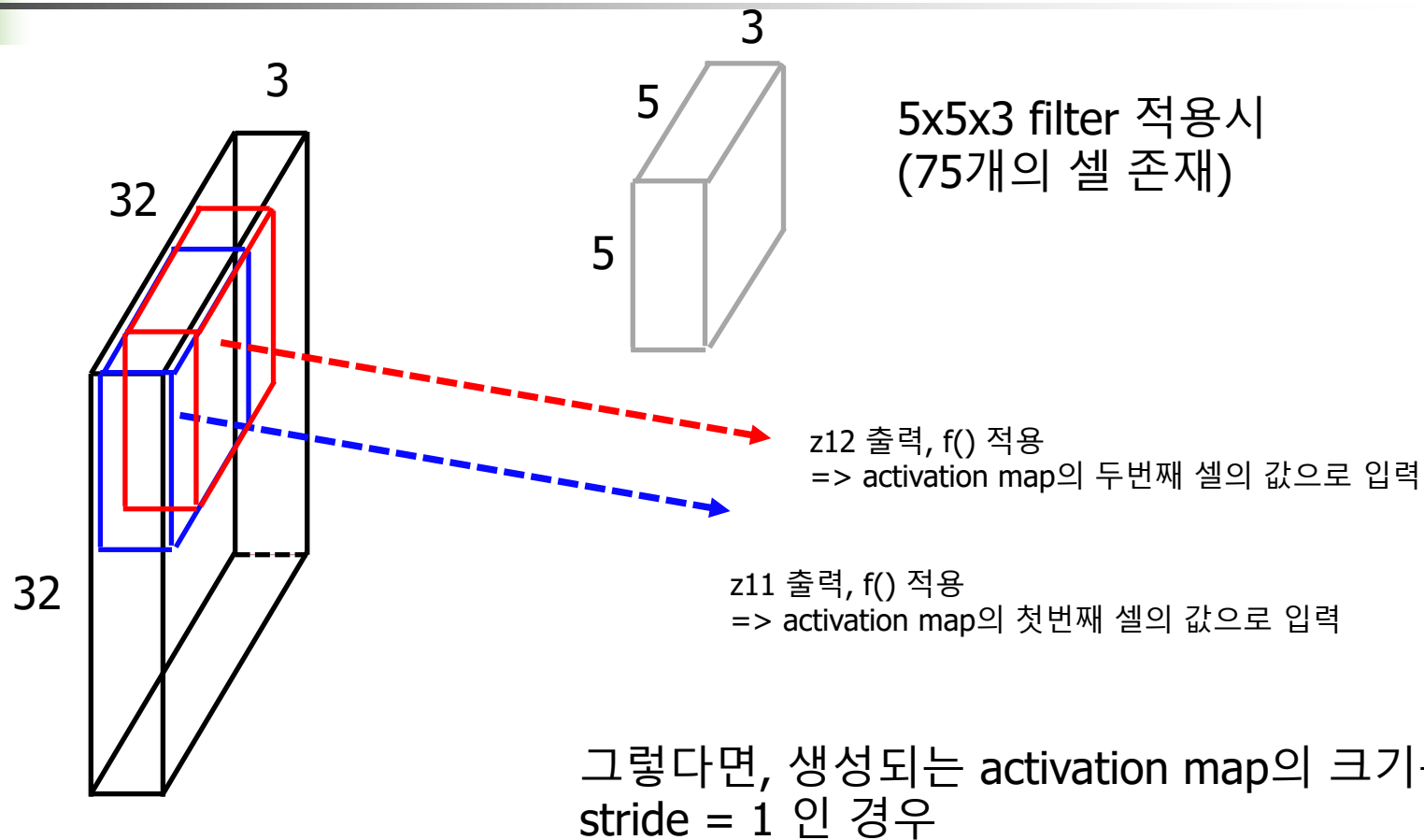
### ■ 5x5x3 filter를 적용하는 경우 (75개의 셀 존재)

- 한번 필터가 적용될 때 이미지의 5x5x3 셀에 적용
- 한번 필터가 적용될 때, 하나의 숫자가 출력



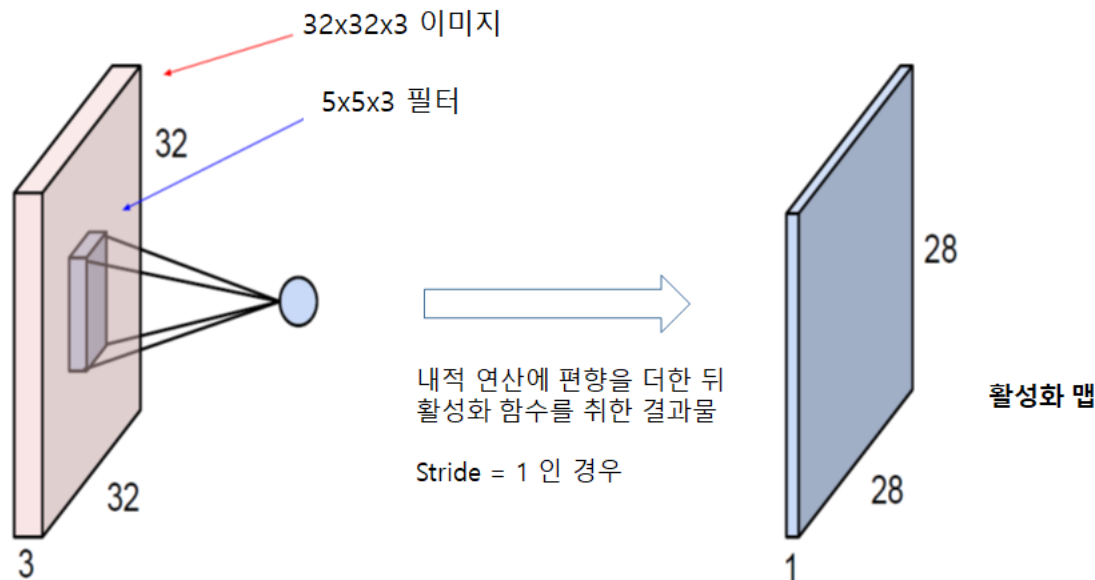
이미지에 저장되어 있는 값과 필터에 존재하는  
가중치들 간에 내적 연산 + bias  $\Rightarrow$  그 다음 활성화  
함수 적용

# CNN 작동 원리



# CNN 작동 원리

## ■ 결과물



하나의 필터를 순차적으로  
적용해서 나오는 activation  
map의 depth는 1

위의 그림은 32x32x3 이미지에 5x5x3의 필터를 적용하게 되면 결과로 28x28x1의 activation map이 얻어진다는 것을 보여주고 있습니다

# CNN 작동 원리

## ■ 필터의 수

- 하나의 이미지 (혹은 입력 데이터)에 여러개의 필터를 적용하는 것이 가능
- 각 필터는 서로 다른 파라미터를 지님

1	10	122	143	155
1	20	124	142	156
2	22	110	130	150
3	24	108	122	140
4	18	111	80	100

w11	w12	w13
w21	w22	w23
w31	w32	w33

w'11	w'12	w'13
w'21	w'22	w'23
w'31	w'32	w'33

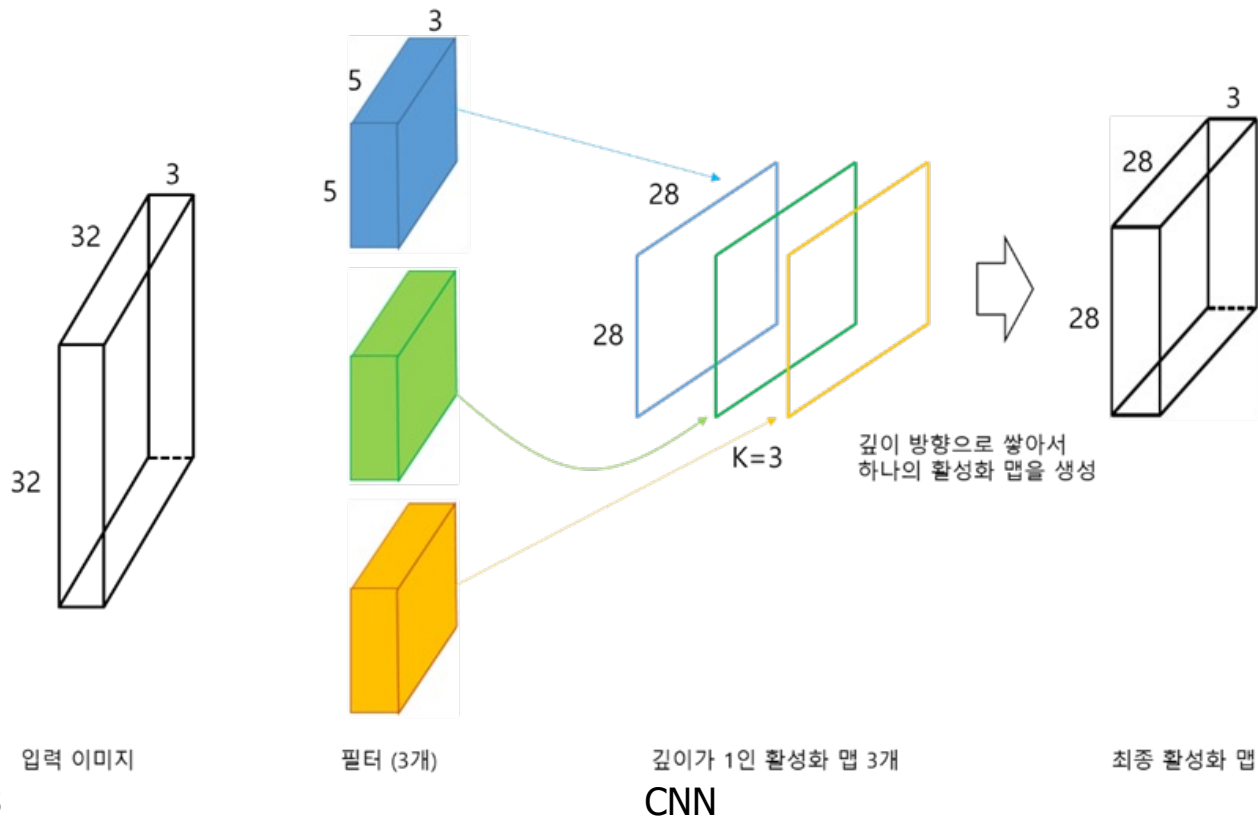
Activation maps

f(z11)	f(z12)	f(z13)
f(z21)	f(z22)	f(z23)
f(z31)	f(z32)	f(z33)

f(Z'11)	f(Z'12)	f(Z'13)
f(Z'21)	f(Z'22)	f(Z'23)
f(Z'31)	f(Z'32)	f(Z'33)

# CNN 작동 원리

- 입력된 이미지에 여러 개(예, K)의 filter를 적용하는 경우
  - 생성되는 (깊이가 1인) activation map의 수 = K





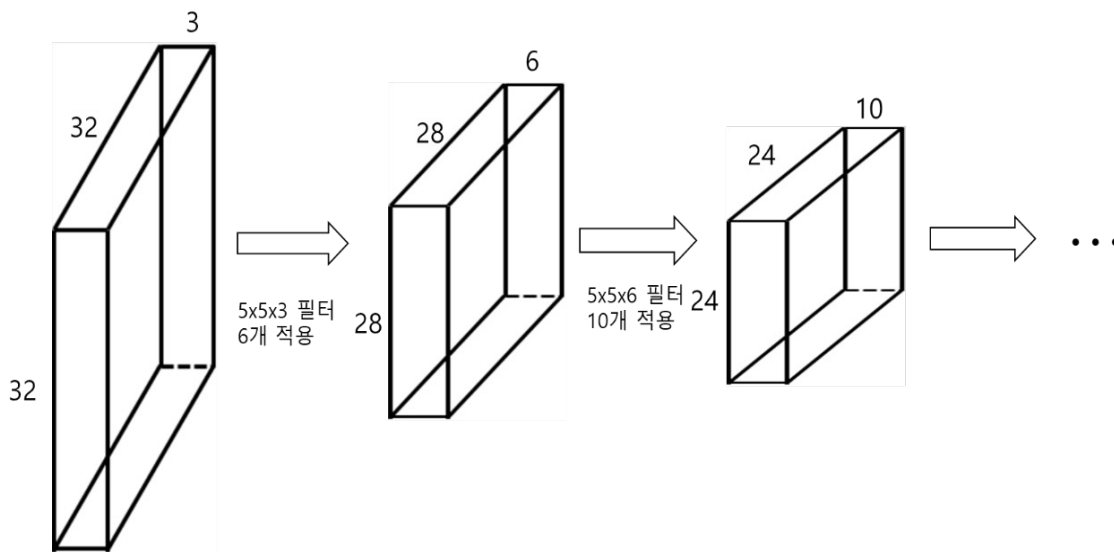
# CNN 작동 원리

---

- 파라미터의 수
  - 전체 파라미터의 수
    - $5 \times 5 \times 3$  크기에 해당하는 필터를 2개 사용한다면, 전체 파라미터의 수는  $5 \times 5 \times 3 \times 2$ 가 될 것
    - 각 필터마다 고유한  $5 \times 5 \times 3$ 개의 파라미터를 갖기 때문
    - 필터마다 하나의 편향 가중치 존재

# CNN 작동 원리

## ■ 합성곱을 여러 번 반복적으로 적용



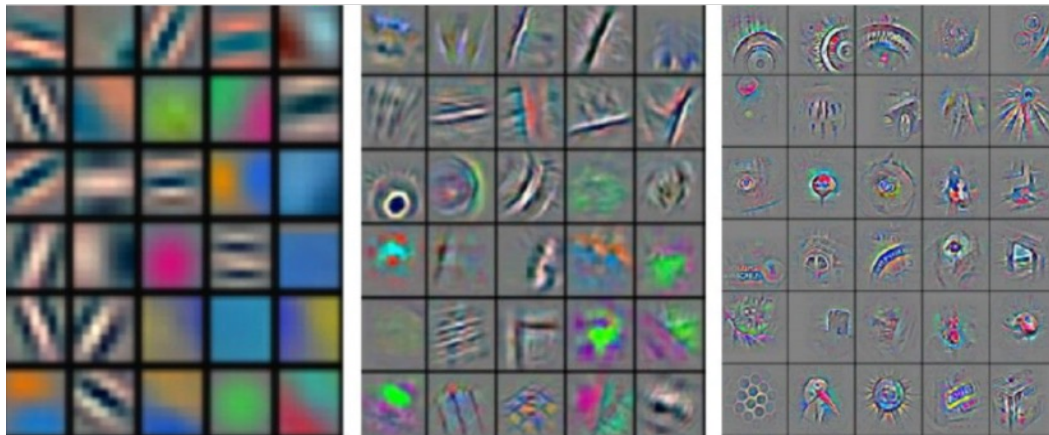
위의 그림은 32x32x3 이미지에 5x5x3 크기의 필터를 6개 적용해서 나온 activation map (28x28x6)에 10개의 5x5x6 필터를 다시 한번 적용해서 24x24x10 크기의 activation map을 얻는 경우를 보여주고 있습니다.

# CNN 작동 원리

- CNN 적용 효과

- 아래와 같이 이미지를 분류하는데 (혹은 주어진 문제를 해결하는데) 필요한 중요한 정보를 추출

합성곱 필터의 순차적 적용

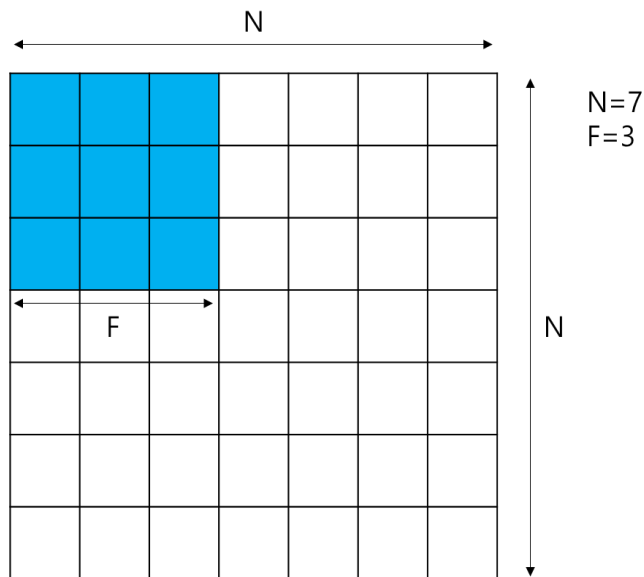




# CNN 관련 추가 용어

## ■ Padding

- 예: 7x7 이미지에 3x3 필터를 적용한다고 가정



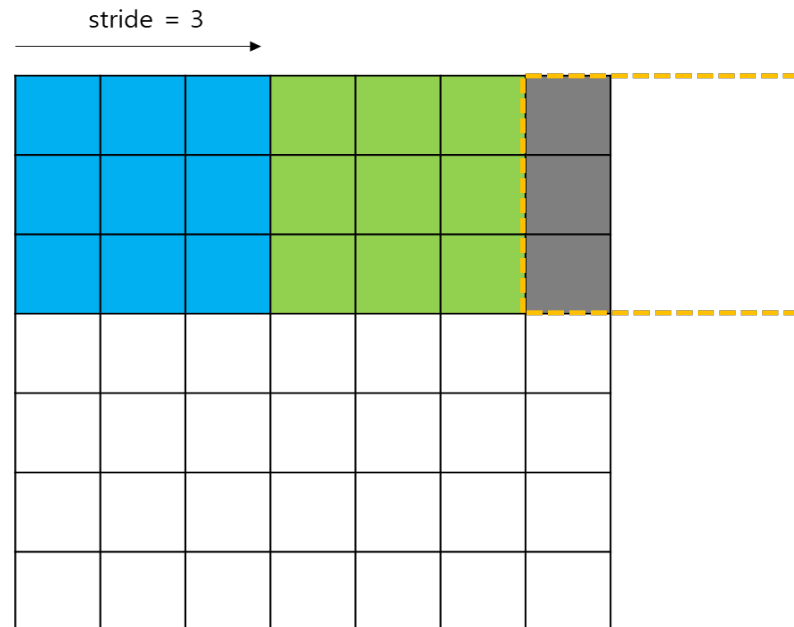
stride = 1 인 경우에, 결과로 얻어지는 activation map의 크기는  $(7-3)/1+1 = 5$  (즉, 5x5)가 됩니다. stride = 2 인 경우에는 그 값이  $(7-3)/2 + 1 = 3$  (즉, 3x3)이 됩니다. 그런데, 만약 stride = 3으로 지정하고 필터를 적용하게 되면 어떻게 될까요? 그 결과는  $(7-3)/3 + 1 = 2.3333$  이 됩니다. 이러한 경우 정보손실 발생 (filter 가 제대로 적용되지 않는 부분은 버리게 된다)

# CNN 관련 추가 용어

## ■ Padding

- 예: 7x7 이미지에 3x3 필터를 적용한다고 가정

그런데, 만약 stride = 3으로 지정하고 필터를 적용하게 되면 어떻게 될까요? 그 결과는  $(7-3)/3 + 1 = 2.3333$  이 됩니다.  
이러한 경우 정보손실 발생 (filter 가 제대로 적용되지 않는 부분은 버리게 된다)



# CNN 관련 추가 용어

## ■ Padding

- padding이란 특정한 값으로 이미지의 주변을 채워주는 것
- Zero padding: 0을 사용하여 padding

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

이렇게 하면 3x3필터를 stride = 3으로 적용해도 정보 손실이 발생하지 않는다.

$$(9-3)/3 + 1 = 3$$



# CNN 관련 추가 용어

---

- Keras에서의 padding
  - [https://www.tensorflow.org/api\\_docs/python/tf/keras/layers/Conv2D](https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D)
  - padding parameter: one of "valid" or "same" (case-insensitive). "valid" means no padding. "same" results in padding with zeros evenly to the left/right or up/down of the input such that output has the same height/width dimension as the input (when strides = (1,1))



# CNN 관련 추가 용어

---

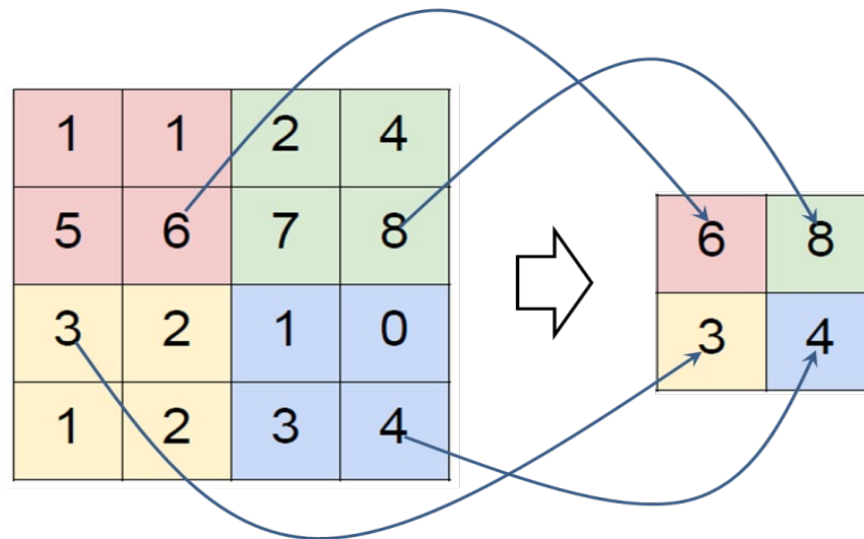
## ■ Pooling

- 필터를 적용하여 합성곱을 한 이후에는 보통 pooling이라는 과정을 거칩니다.
- 이는 activation map에 존재하는 정보 중에서 (이미지를 잘 나타내는) 일부 정보만을 추출해서(즉, pooling) 사용하겠다는 것을 의미 (downsampling의 효과)
- 주요 방법
  - max pooling과 average pooling
- Pooling을 위해서도 별도의 filter를 사용합니다. 하지만, 해당 filter에는 가중치가 없습니다
- Max pooling은 filter가 적용되는 값들 중에서 가장 큰 값을 추출 (pooling)하는 것이고, average pooling은 일련의 값들의 평균 값을 계산하여 사용하는 것입니다.

# CNN 관련 추가 용어

- Max pooling의 예
  - 2x2 filter를 적용해서 max pooling

2x2 영역에서 (max pooling의 경우) 가장 큰 값을 추출



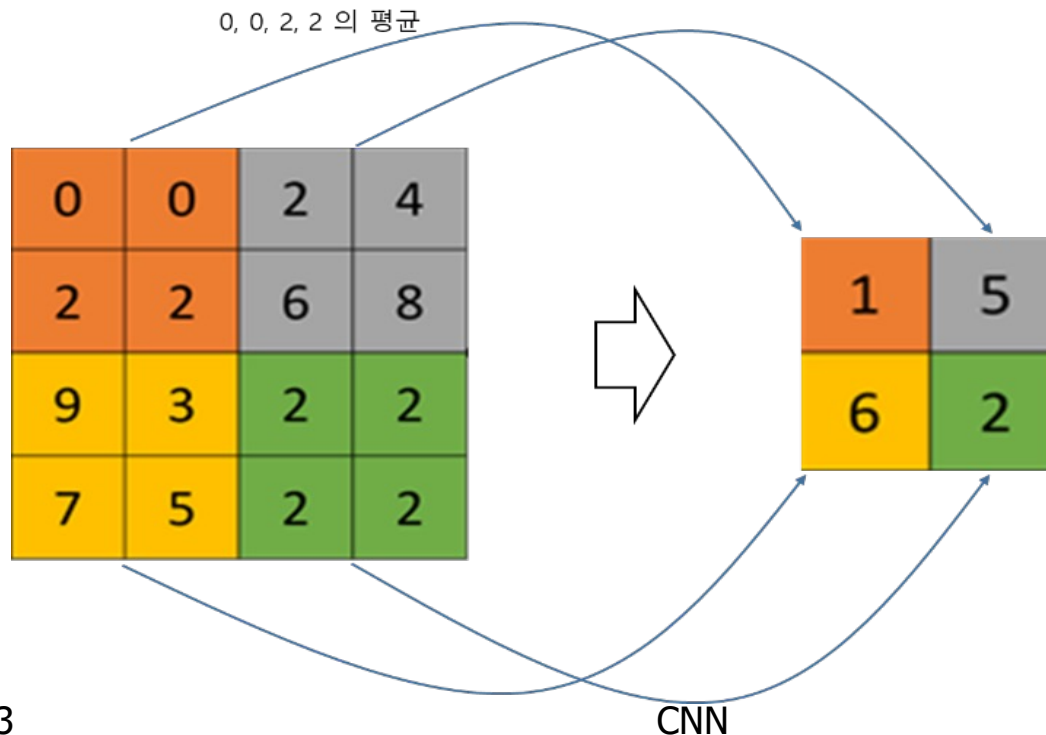
2x2를 적용하는 경우  
보통 stride = 2

activation map

CNN

# CNN 관련 추가 용어

- Average pooling의 예
  - 2x2 filter를 적용해서





# CNN 관련 추가 용어

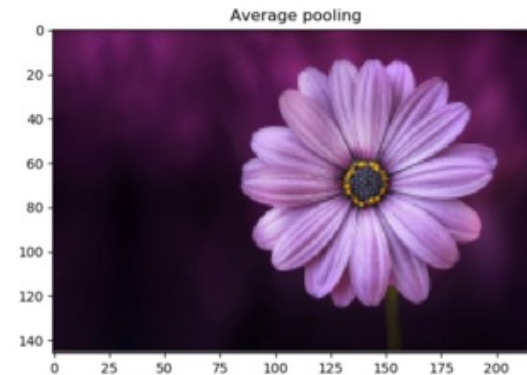
---

- Average pooling
  - 이미지를 부드럽게 하는 (smoothing out) 효과가 있음, sharp features는 잘 도출되지 않음
- Max pooling
  - 큰 숫자 => 더 밝은 색을 의미
  - Max pooling은 더 밝은 색을 추출
  - 백그라운드는 어둡고, 우리가 관심있는 물체가 밝은 경우 사용하면 유용 (MNIST 등)
- What to choose?
  - It depends on your data.



# CNN 관련 추가 용어

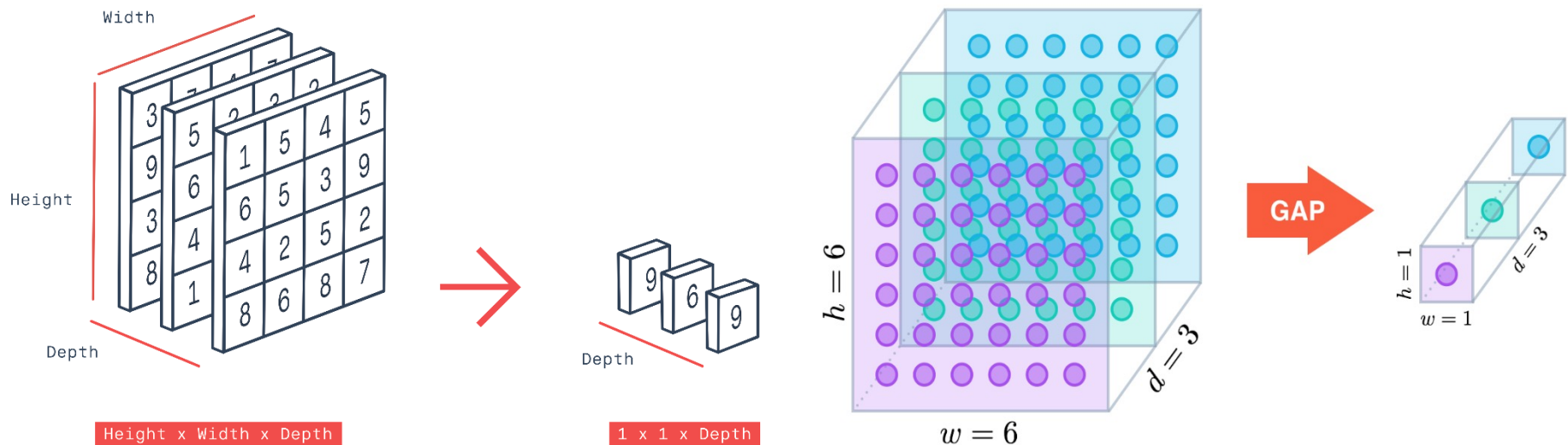
## ■ Max vs. Average pooling



# CNN 관련 추가 용어

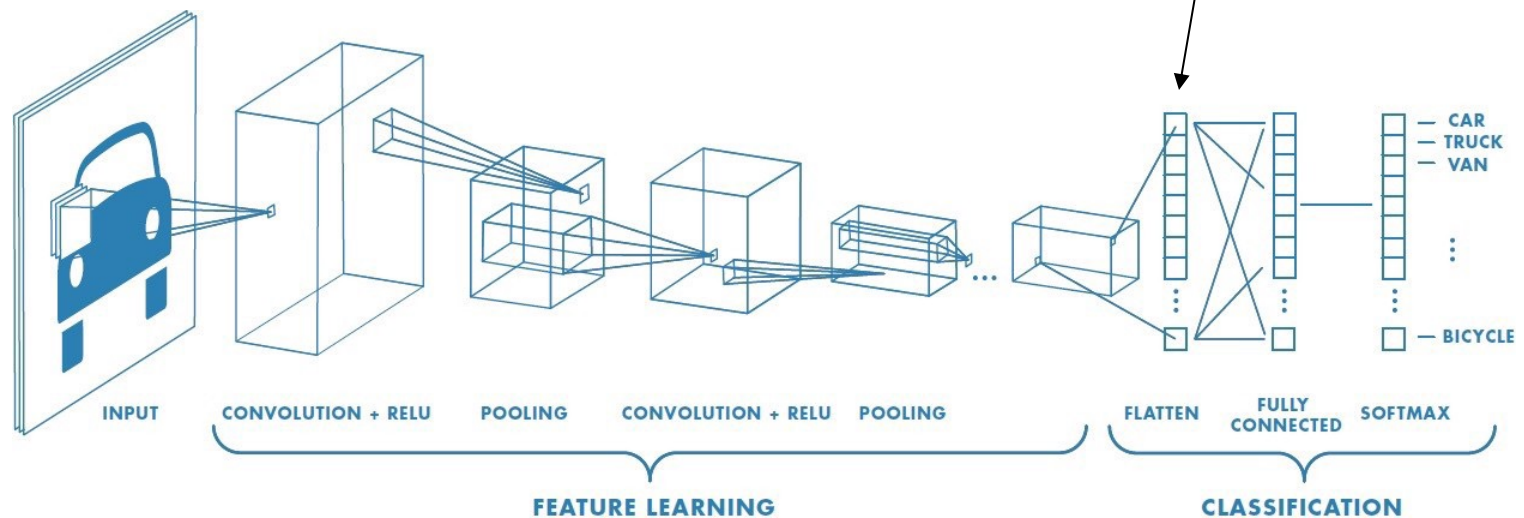
## ■ Global pooling

- 각 channel의 feature map에서 하나의 value를 추출
- Global pooling 연산의 경우, depth는 그대로 유지
- 임베딩 벡터를 얻고자 하는 경우 유용



# CNN 작동 원리

## ■ CNN의 일반적인 구조





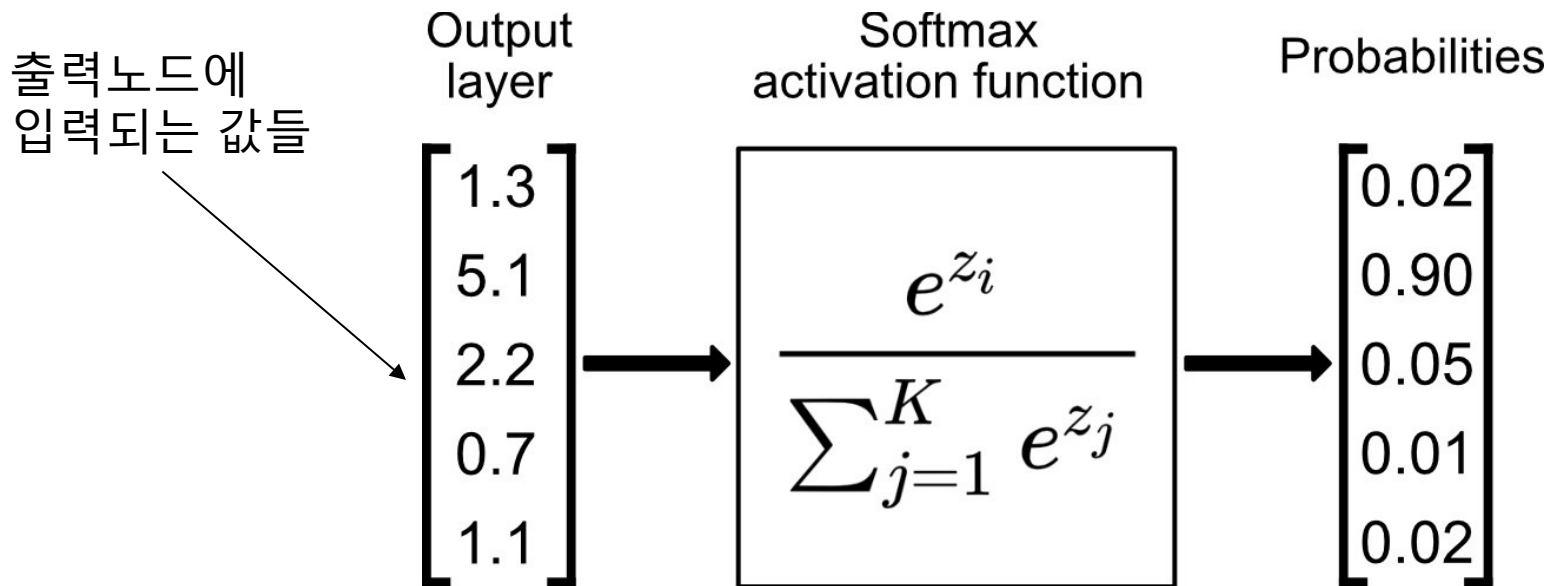
# CNN 작동 원리

---

- CNN의 출력층
  - 출력층의 형태는 FNN에서의 출력층의 형태와 동일
  - CNN을 분류문제에 적용하는 경우
    - 따라서 출력 노드의 수는 종속변수가 취할 수 있는 값의 수와 동일
    - 각 출력 노드는 종속변수가 특정한 값을 취할 확률값을 출력, 이는 softmax 함수를 사용해서 계산

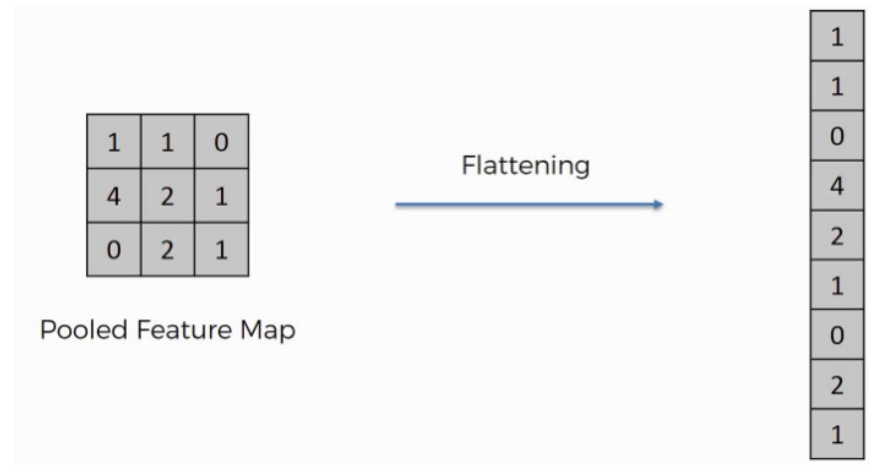
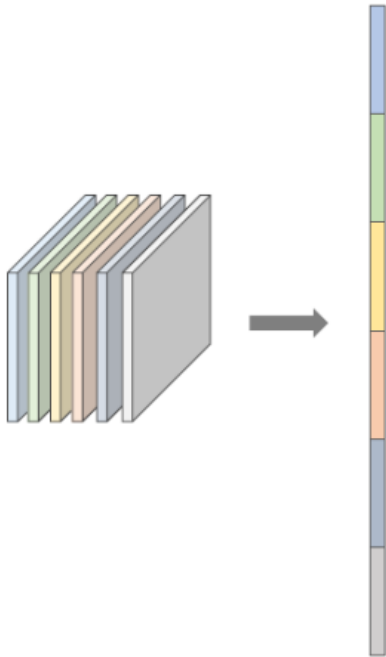
# Softmax 함수

## ■ Example



# CNN 관련 추가 용어

## ■ Flatten





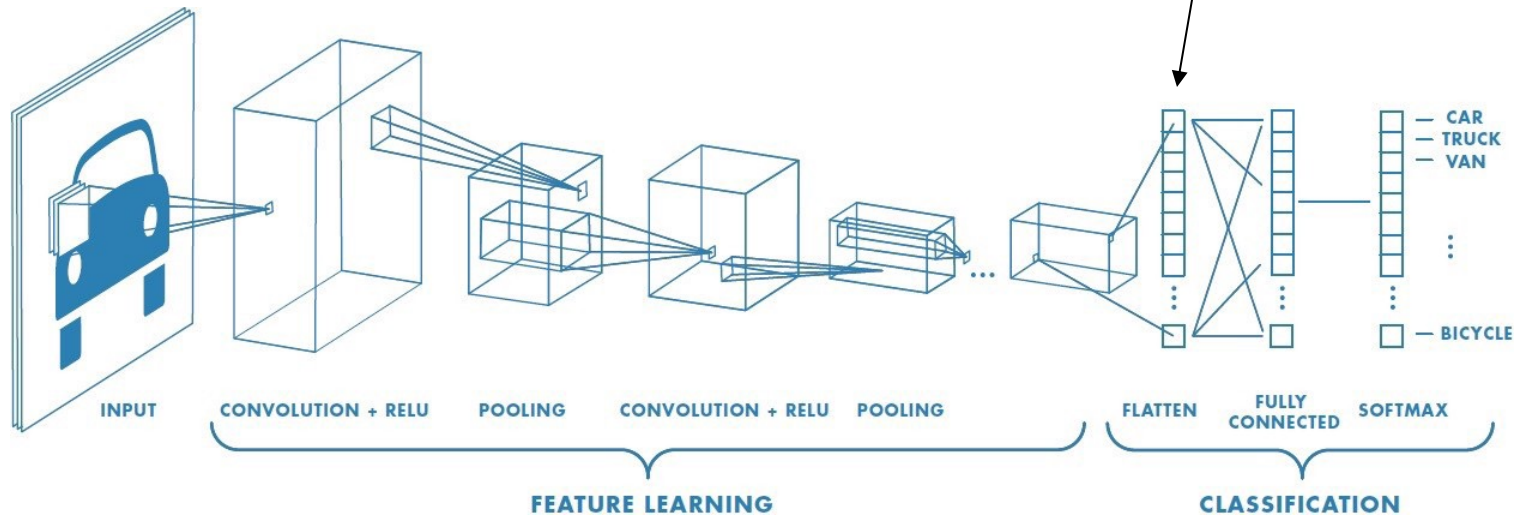
# Recap

---

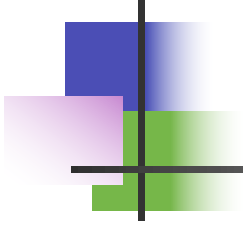
- CNN 주요 내용
  - Convolutional filter
    - filter size and depth
    - activation map size and depth
  - Stride and padding
  - Pooling
    - max pooling and average pooling
    - global pooling
  - Flattening

# Recap

## ■ CNN의 일반적인 구조



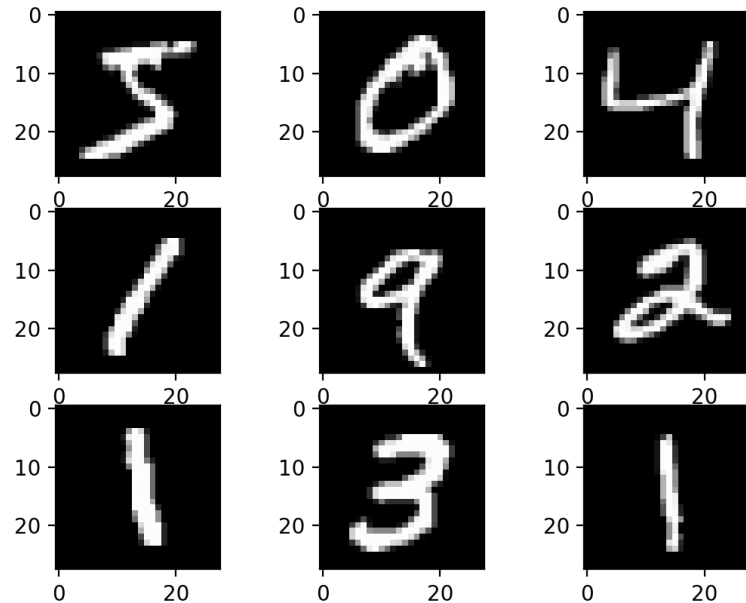




# PYTHON CODING

# MNIST

- Python을 이용한 이미지 분류 예제
  - Conv2D 1개: MNIST\_CNN\_example.ipynb
  - Conv2D 2개: MNIST CNN example2.ipynb
  - 예제 이미지



# MNIST

## ■ 신경망 구조

why 26x26?

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_1 (MaxPooling2D)	(None, 13, 13, 32)	0
flatten_1 (Flatten)	(None, 5408)	0
dense_1 (Dense)	(None, 128)	692352
dense_2 (Dense)	(None, 10)	1290
Total params: 693,962		
Trainable params: 693,962		
Non-trainable params: 0		



# MNIST

---

- 시도해 봐야 하는 것
  - stride 값 변경해보기

```
model.add(Conv2D(32, kernel_size=(3, 3),  
                 activation='relu',  
                 input_shape=input_shape, strides=(2,2)))
```

- padding 방식 변경해 보기

```
model.add(Conv2D(32, kernel_size=(3, 3),  
                 activation='relu',  
                 input_shape=input_shape, padding="same"))
```



# MNIST

- Python을 이용한 이미지 분류 예제 2
  - 조금더 복잡한 모형 사용해 보기
  - MNIST\_CNN\_example2.ipynb

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_1 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_2 (Conv2D)	(None, 11, 11, 64)	18496
average_pooling2d_1 (AveragePooling2D)	(None, 5, 5, 64)	0
flatten_1 (Flatten)	(None, 1600)	0
dense_1 (Dense)	(None, 64)	102464
dense_2 (Dense)	(None, 10)	650
Total params: 121,930		
Trainable params: 121,930		
Non-trainable params: 0		



# CIFAR-10

---

- 두 번째 예제: CIFAR-10 사진들 분류하기
  - CIFAR-10 60000개의 컬러 사진, 32x32x3
  - 10 classes
  - each class contains 6000
- 파이썬 코드: `CNN_cifar10_simple.ipynb`  
참조



airplane

automobile

bird

cat

deer

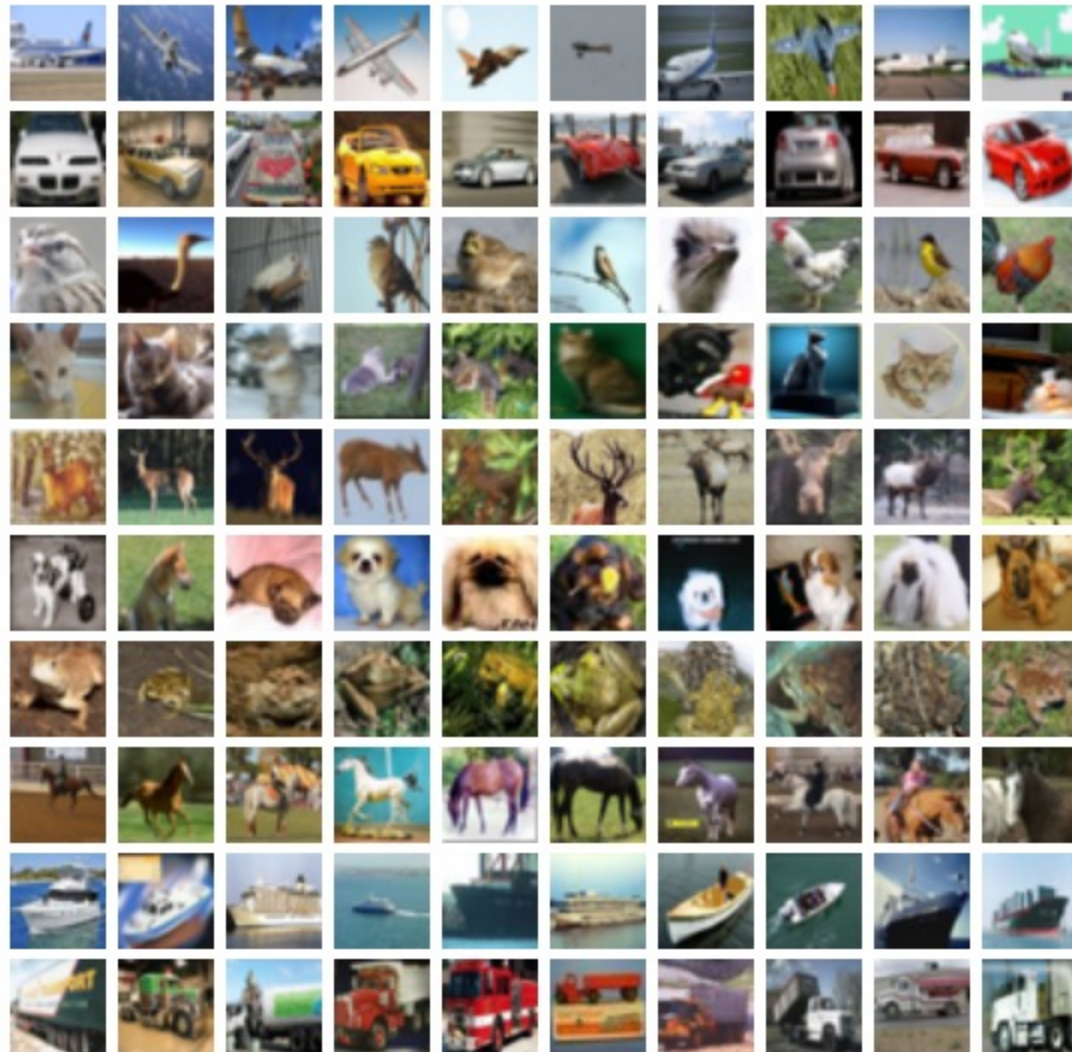
dog

frog

horse

ship

truck





# CIFAR

## ■ 네트워크 구조

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
dropout (Dropout)	(None, 15, 15, 32)	0
flatten (Flatten)	(None, 7200)	0
dense (Dense)	(None, 512)	3686912
dropout_1 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 10)	5130
Total params: 3,692,938		
Trainable params: 3,692,938		
Non-trainable params: 0		

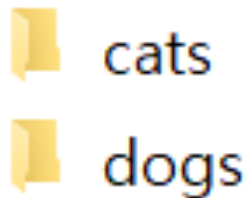




# Cats and Dogs classification

---

- 직접 수집한 데이터에 CNN 적용해 보기
  - See "CNN\_cats\_dogs.ipynb"
  - 데이터 준비
    - 각 클래스 (레이블)에 해당하는 이미지를 별도의 폴더에 저장 (폴더의 이름은 중요하지 않음)
    - 예)





# Cats and Dogs classification

---

- 직접 수집한 데이터에 CNN 적용해 보기
  - 주요 과정
    - 이미지를 특정 사이즈로 불러오기
    - 정규화 하기 (0과 1 사이의 값으로)
    - 종속변수 생성하기