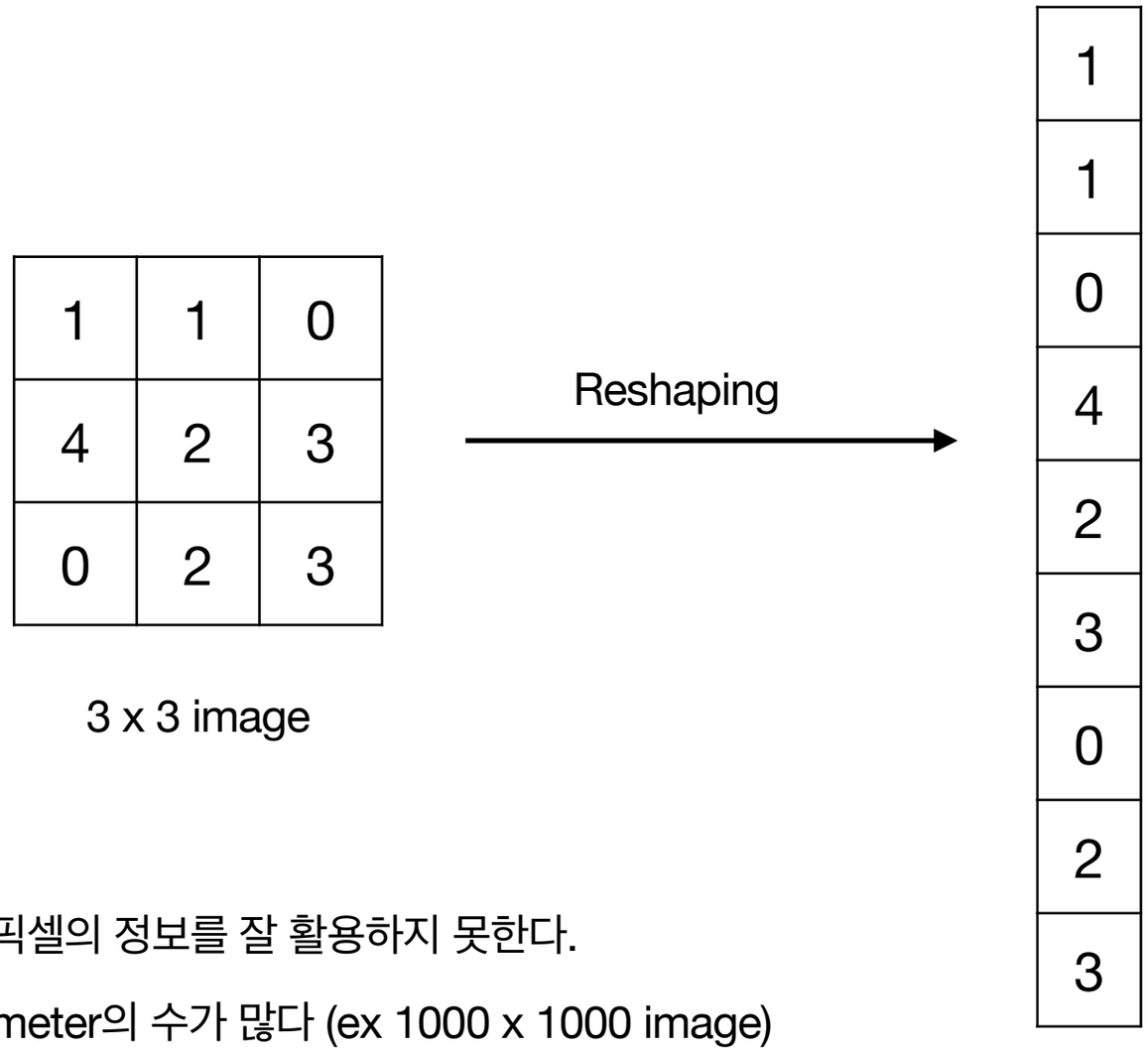
Lecture 2

Convolution Neural Network

A-Task

Yeolkyo Choi

Image Treatment using FNN



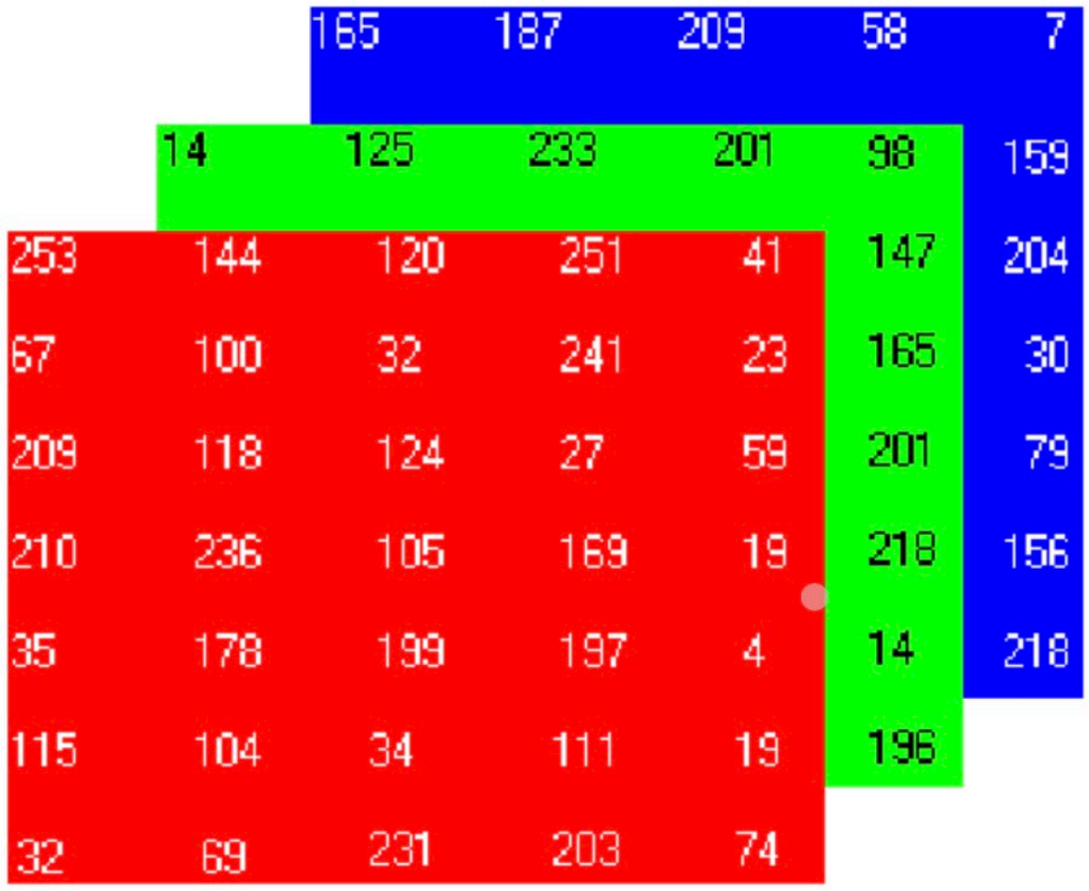
- 이웃 픽셀의 정보를 잘 활용하지 못한다.
- Parameter의 수가 많다 (ex 1000 x 1000 image)

Convolution Neural Network의 장점

- 데이터가 가지고 있는 spatial 정보를 추출하는데 유리
- 적은 parameter로 이미지 데이터 분석에 용이

Color Image

- Color image의 경우, 하나의 픽셀이 보통 3개의 색 정보를 지님 (Red, Green, Blue, a.k.a., RGB; called channel or depth = 3)
- 3차원의 array로 저장됨 (m x n 칼라 이미지의 경우, m x n x 3의 3차원 array로 데이터 저장)
- 각 cell은 하나의 색상 정보를 지니며 각각이 하나의 feature (독립변수)의 값이 됨.



Kernel (Filter) in CNN

- 이미지 정보를 추출하기 위해 filter (kernel)을 적용
- 하나의 filter를 2차원으로 이동시키면서 정보를 추출
- filter의 크기는 F x F x D

  D : 입력 이미지의 depth (channel 수)

  F : filter의 가로 또는 세로의 길이

  (흑백 이미지는 D = 1, 즉 F x F x 1 filter 사용)

W_{11}	W_{12}	W_{13}
W_{21}	W_{22}	W_{23}
W_{31}	W_{32}	W_{33}

- Filter의 각 셀은 고유의 가중치를 갖음
- Filter 적용 시, filter가 적용되는 이미지 부분의 색 정보와 filter의 각 가중치 간에 내적 연산(합성곱) 수행

(https://setosa.io/ev/image-kernels/)

1	10	122	143	155
1	20	124	142	156
2	22	110	130	150
3	24	108	122	140
4	18	11	80	100

$w_{11} \times 1 + w_{12} \times 10... + w_{33} \times 110$

1	10	122	143	155
1	20	124	142	156
2	22	110	130	150
3	24	108	122	140
4	18	11	80	100

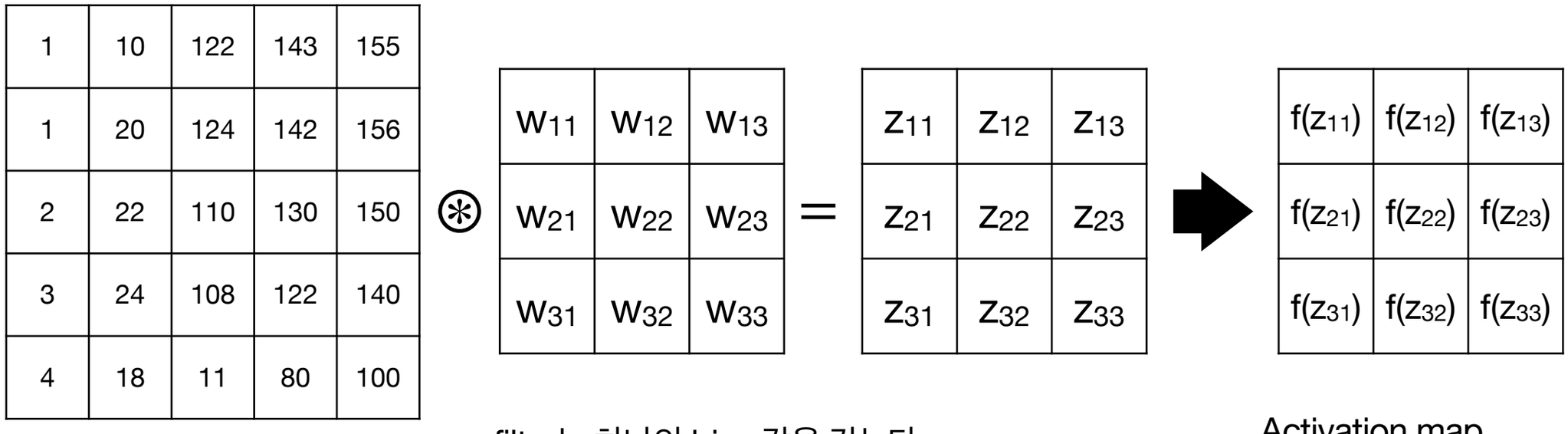
$w_{11} \times 10 + w_{12} \times 122... + w_{33} \times 130$

■ ■ ■

1	10	122	143	155
1	20	124	142	156
2	22	110	130	150
3	24	108	122	140
4	18	11	80	100

$w_{11} \times 110 + w_{12} \times 130... + w_{33} \times 100$

Activation Map



filter는 하나의 bias 값을 갖는다.

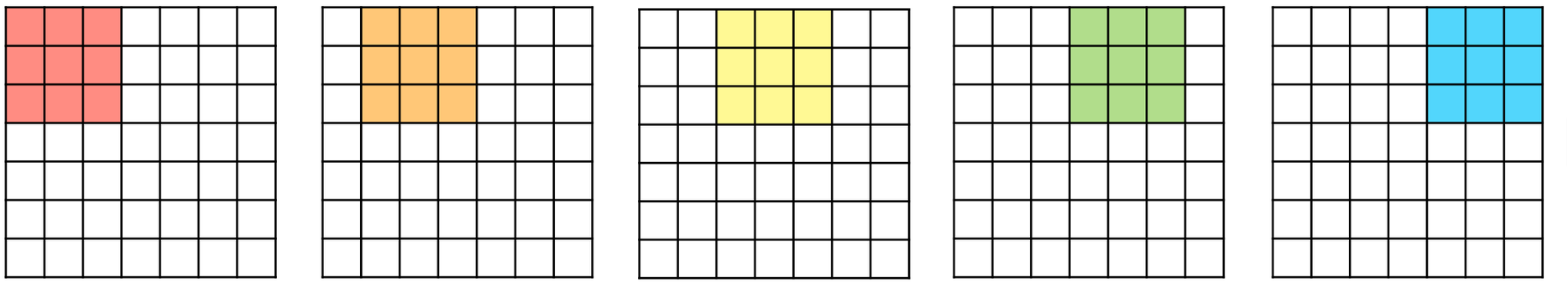
Activation map

- activation function을 적용하여 출력되는 값들을 모아놓은 것
- Feature map 또는 response map 이라고도 함

Stride in CNN

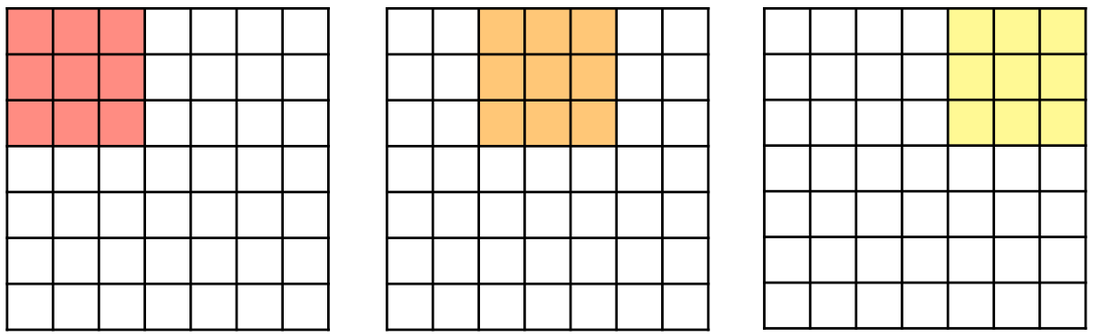
stride는 filter를 한번에 옆으로 (혹은 아래로) 얼마나 움직이는지를 결정 (이는 hyperparameter)

image = 7 x 7 / filter = 3 x 3 / stride = 1



Activation map = 5 x 5

image = 7 x 7 / filter = 3 x 3 / stride = 2

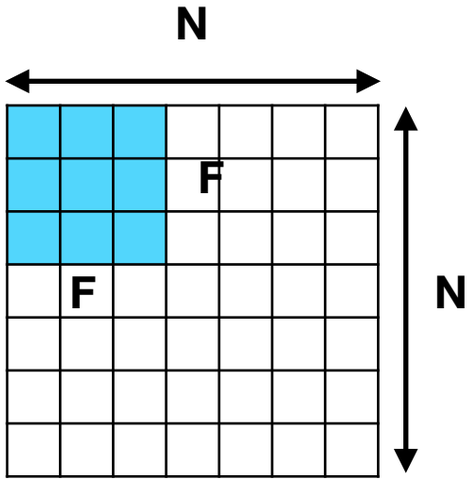


Activation map = 3 x 3

Mainly for downsampling

- parameter 수 감소
- 학습속도 개선
- 정보 손실, 성능 저하

Padding in CNN



Output size:

(N-F) / S + 1

e.g. N = 7, F = 3

stride 1 → (7-3)/1 + 1 = 5

stride 2 → (7-3)/2 + 1 = 3

stride 3 → (7-3)/3 + 1 = 2.33

In practice: Common to zero pad the border

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

e.g. input = 7 x 7

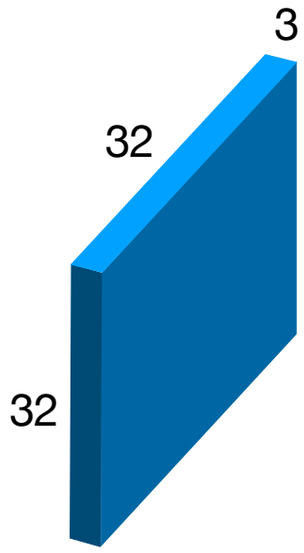
3 x 3 filter, applied with stride 1

pad with 1 pixel border → what is the output?

(N + 2P - F)/S + 1 => 7 x 7 output!

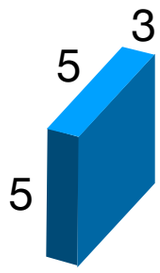
In general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2 (will preserve size spatially)

Filter with Multichannel



Color Image (RGB)

32 x 32 pixel with depth (channel) = 3



5 x 5 x 3 filter

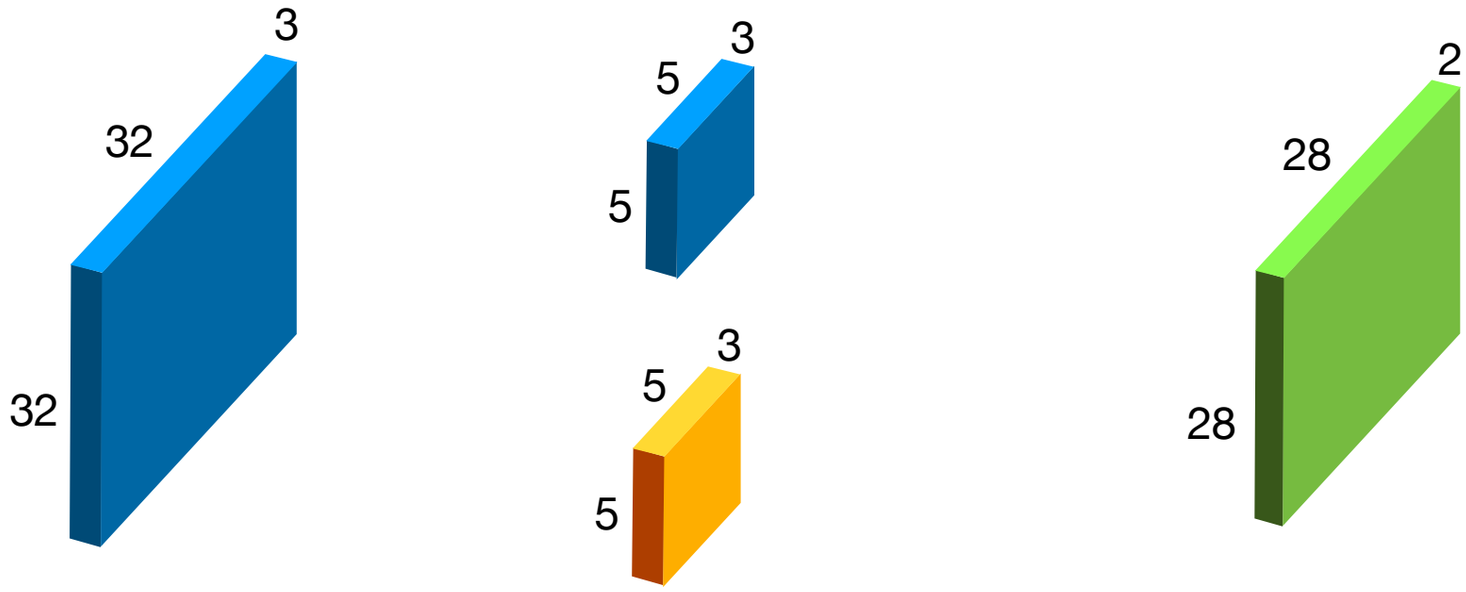
How many weights?

How many bias?

Activation map ?

Filter with Multichannel

하나의 이미지에 여러개의 필터를 적용하는 것이 가능



1	10	122	143	155
1	20	124	142	156
2	22	110	130	150
3	24	108	122	140
4	18	11	80	100

$\circledast$

w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

$=$

z_{11}	z_{12}	z_{13}
z_{21}	z_{22}	z_{23}
z_{31}	z_{32}	z_{33}

Activation maps

$f(z_{11})$	$f(z_{12})$	$f(z_{13})$
$f(z_{21})$	$f(z_{22})$	$f(z_{23})$
$f(z_{31})$	$f(z_{32})$	$f(z_{33})$

$\circledast$

$w`_{11}$	$w`_{12}$	$w`_{13}$
$w`_{21}$	$w`_{22}$	$w`_{23}$
$w`_{31}$	$w`_{32}$	$w`_{33}$

$=$

$z`_{11}$	$z`_{12}$	$z`_{13}$
$z`_{21}$	$z`_{22}$	$z`_{23}$
$z`_{31}$	$z`_{32}$	$z`_{33}$

$f(z`_{11})$	$f(z`_{12})$	$f(z`_{13})$
$f(z`_{21})$	$f(z`_{22})$	$f(z`_{23})$
$f(z`_{31})$	$f(z`_{32})$	$f(z`_{33})$

How CNN Works

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

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

x[:,:,1]

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

x[:,:,2]

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

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
0	0	1
1	-1	1

w0[:,:,1]

-1	0	1
1	-1	1
0	1	0

w0[:,:,2]

-1	1	1
1	1	0
0	-1	0

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

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

w1[:,:,1]

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

w1[:,:,2]

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

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

2	3	3
3	7	3
8	10	-3

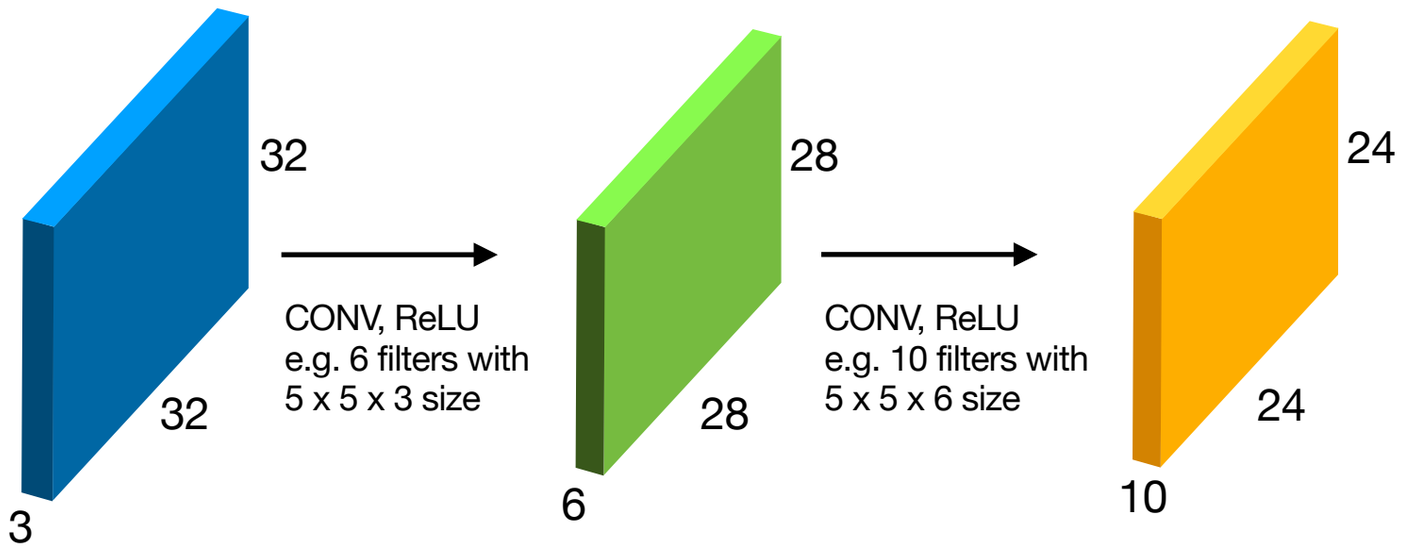
o[:,:,1]

-8	-8	-3
-3	1	0
-3	-8	-5

toggle movement

How CNN Works

- **합성곱의 반복적용**



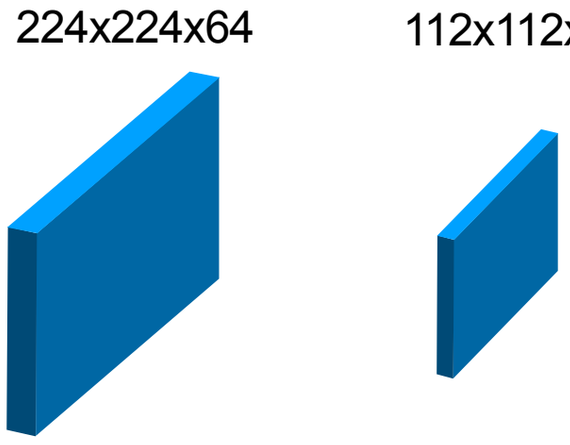
위 그림은 32x32x3 color image에 5x5x3 크기의 filter 6개를 적용해서 나온 activation map (28x28x6)에 10개의 5x5x6 filter를 다시 한번 적용하여 24x24x10 크기의 activation map을 얻는 경우를 나타낸다.

뒤로 갈수록 object의 윤곽정보를 포함함

Pooling Layer

- makes the representations smaller and more manageable
- operates over each activation map independently

MAX POOLING



Single depth slice

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

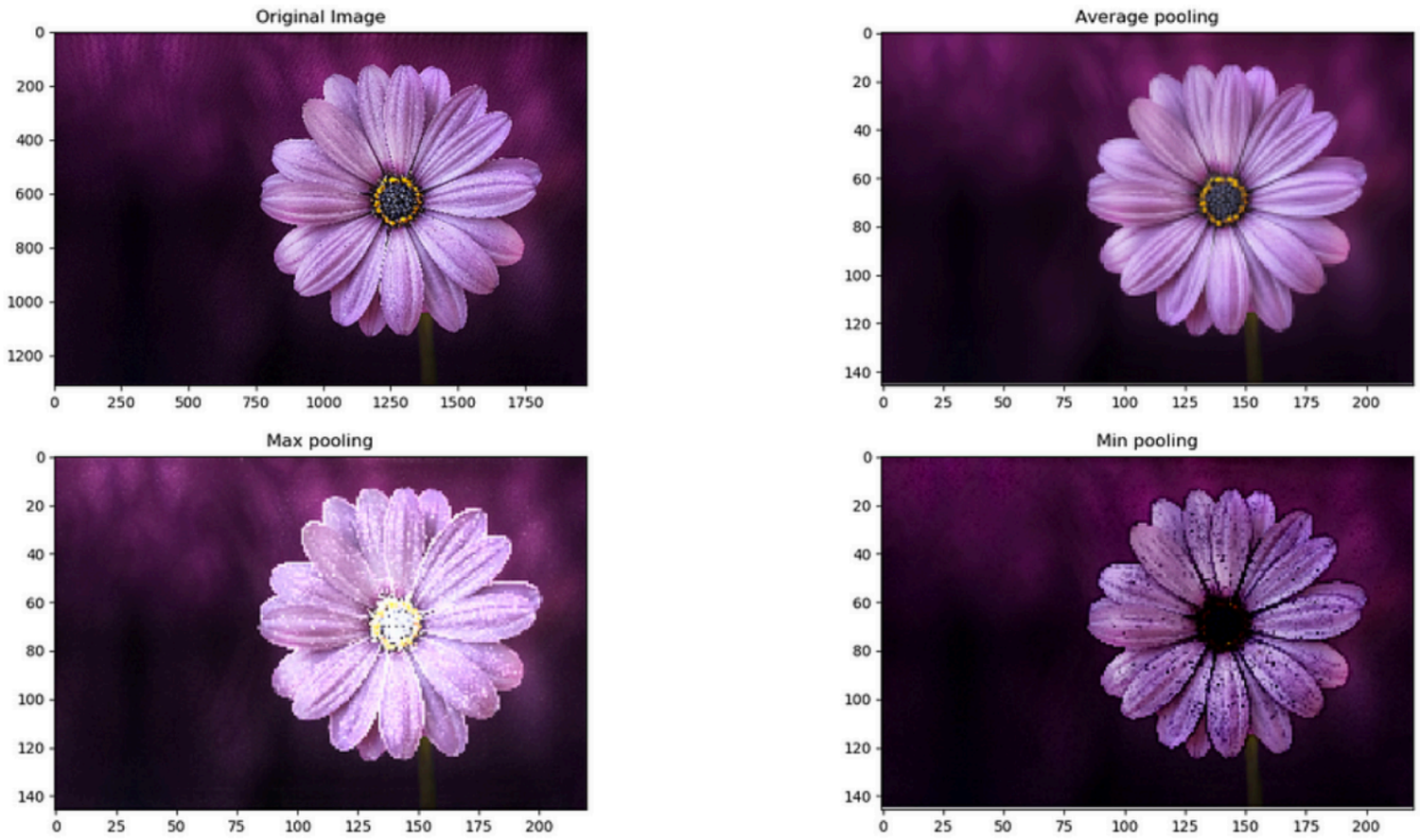
Common setting

F=2, S=2

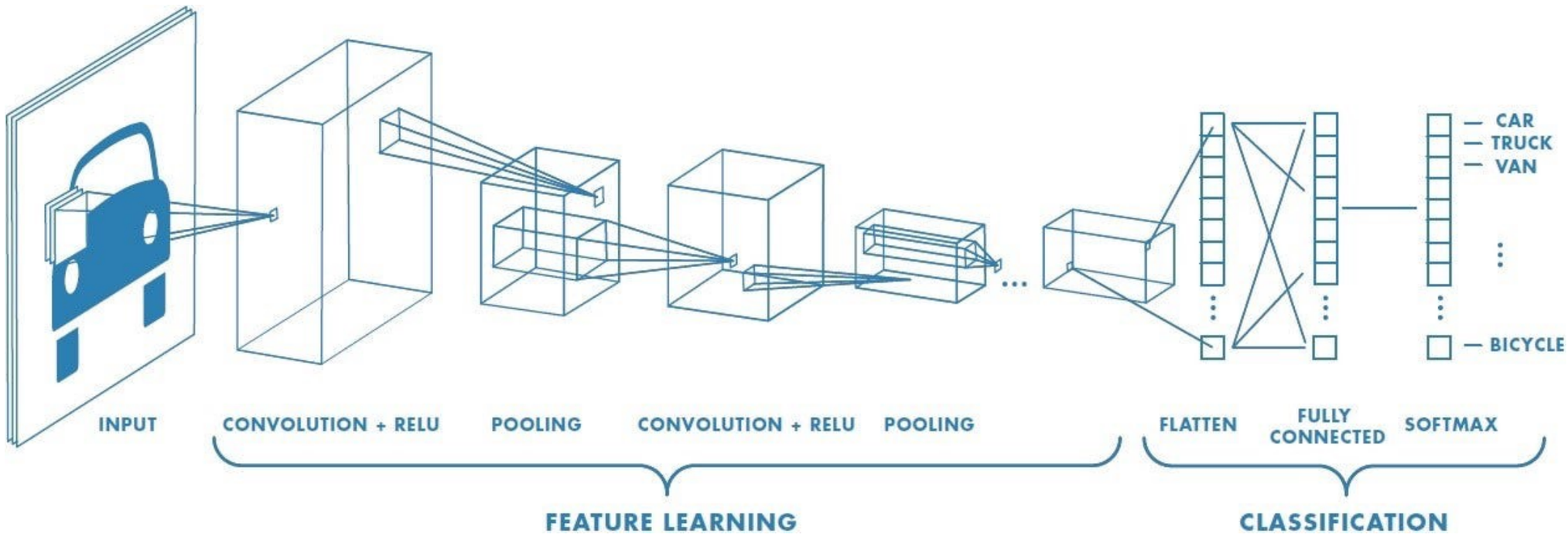
F=3, S=2

max pool with 2x2 filters and stride 2

6	8
3	4



Fully Connected Layer



Fully Connected Layer

32x32x3 image > stretch to 3072 x 1

activation

Wx

3072

10 x 3072

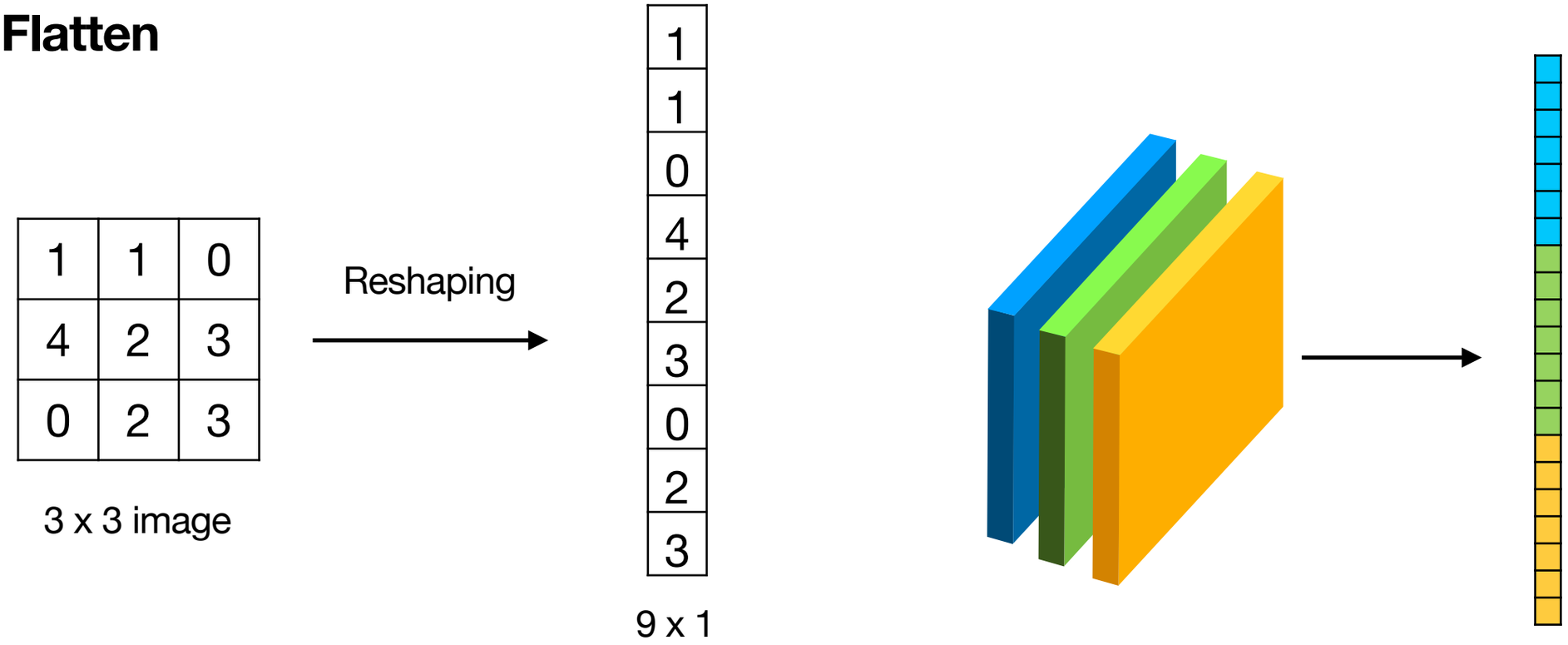
1 number:

the result of taking a dot product between W and the input

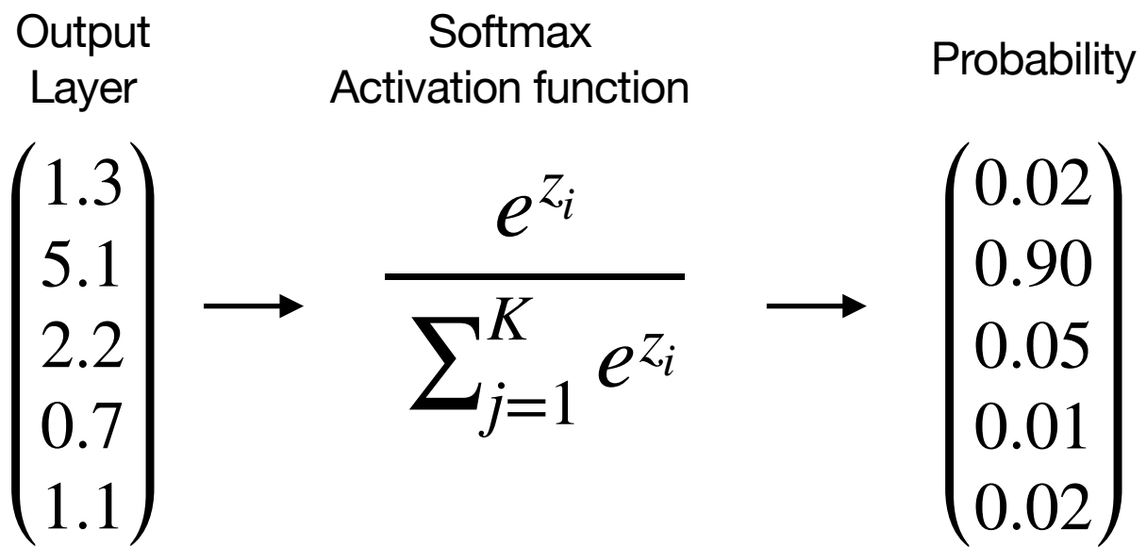
(a 3072 dimensional dot product)

Flatten and Softmax

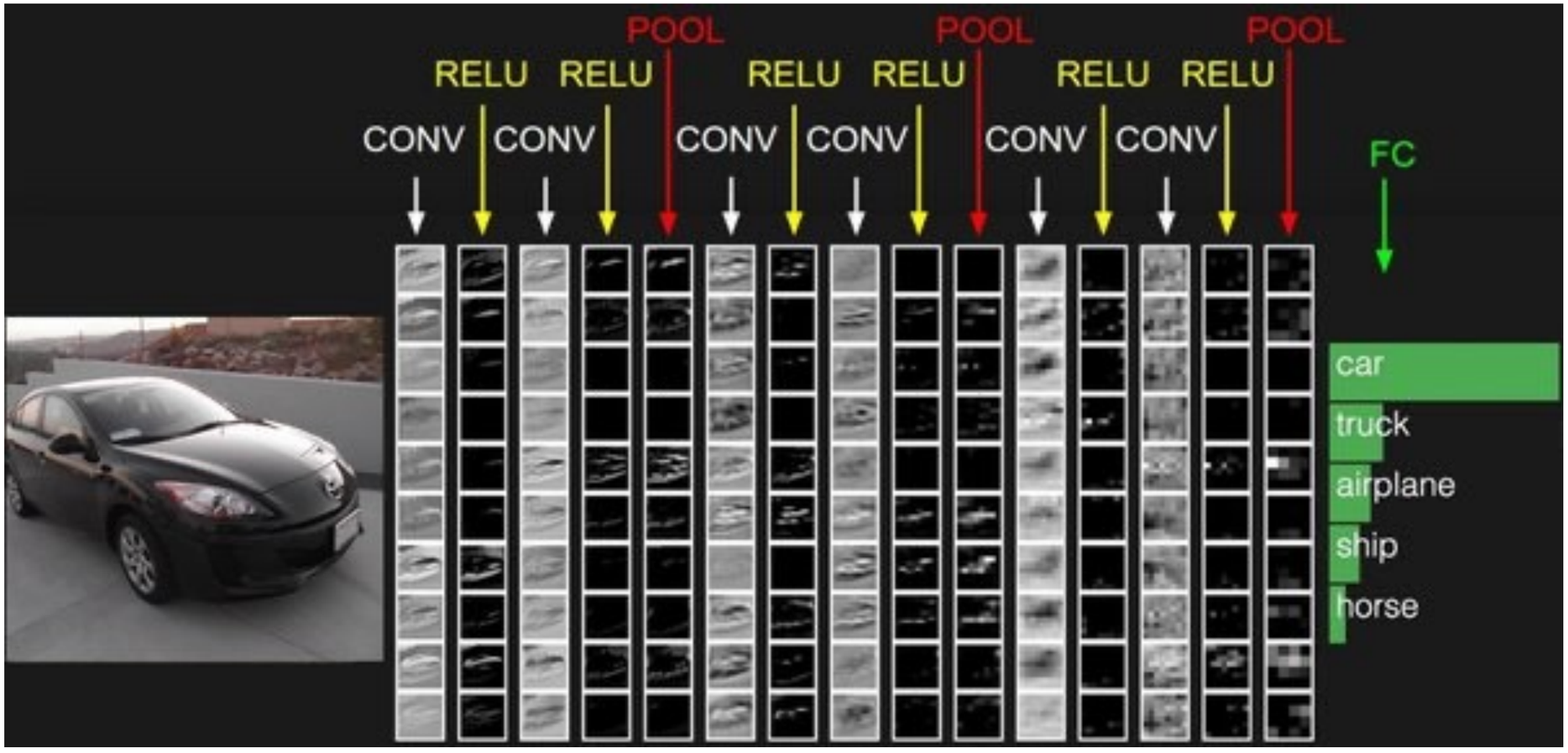
- **Flatten**



- **Softmax**



CNN Summary



- CNNs stack of CONV, POOL, FC layers
- Trend towards smaller filters and deeper architectures
- Trend towards getting rid of POOL/FC layers (just CONV)
- Typical architectures look like [(CONV-RELU) x N - POOL] x M - (FC-RELU) x K - SOFTMAX where N is usually up to ~5, M is large, 0 <= K <= 2
- Recent advances such as ResNet/GoogLeNet challenge this paradigm