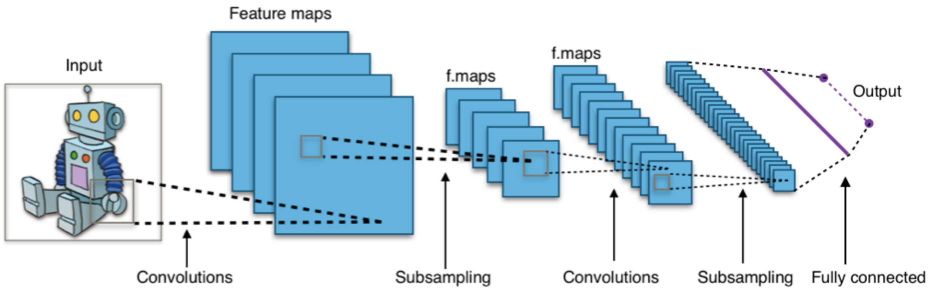
Convolution Neural Network

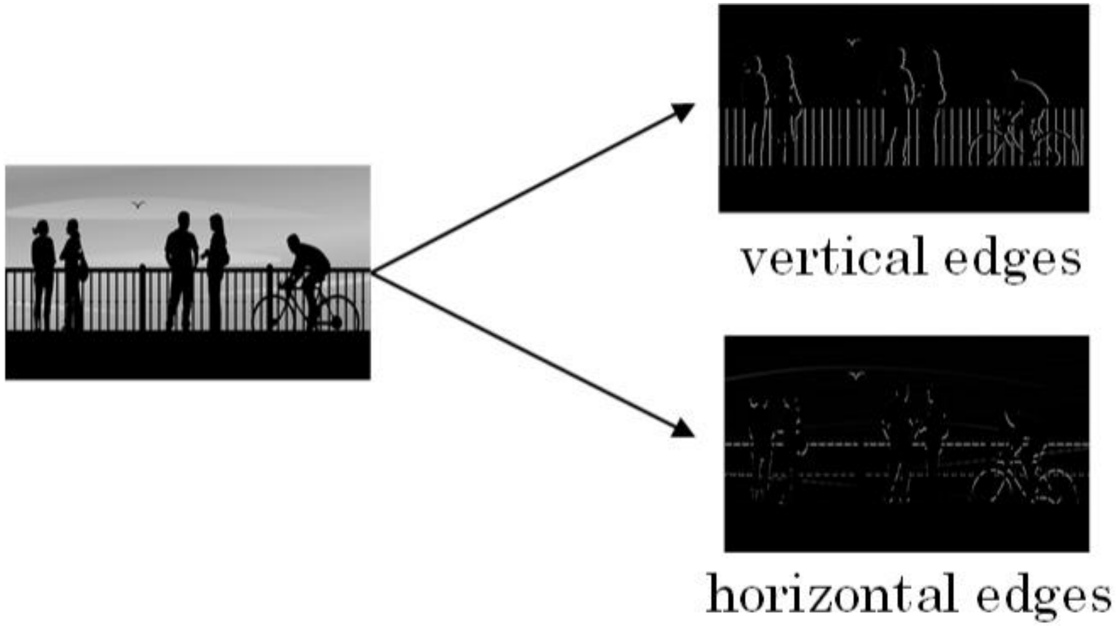
명지대학교
MYONGJI UNIVERSITY

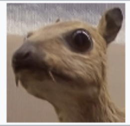
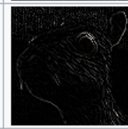
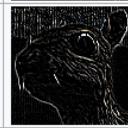
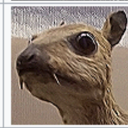
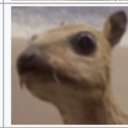
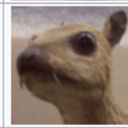
CNN

- **ANN (Artificial Neural Network) : 은닉 계층이 주로 하나**
- **DNN (Deep Neural Network) : 은닉 계층을 많이 쌓아서 만든 인공지능 기술**
- **CNN (Convolution Neural Network) : 주로 영상처리에 많이 활용되는 합성곱을 이용하는 인공 신경망**
  - 합성곱 (convolution filter)를 이용하여 신경망 동작
  - 여러 작은 이미지 필터가 이미지 위를 돌아 다니면서 특징점들을 찾아 그 합성 곱 결과를 다음 계층으로 보냄
  - 적은 수의 가중치로 이미지 처리를 효율적으로 할 수 있음



Convolution



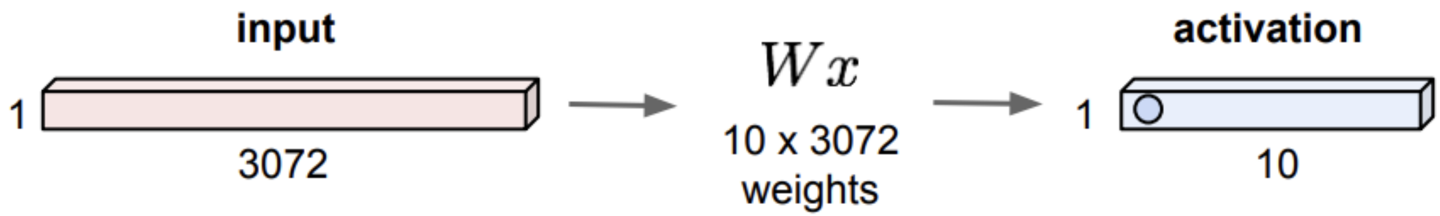
Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Gaussian blur 5 × 5 (approximation)	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

https://en.wikipedia.org/wiki/Kernel_(image_processing)

Fully connected layer

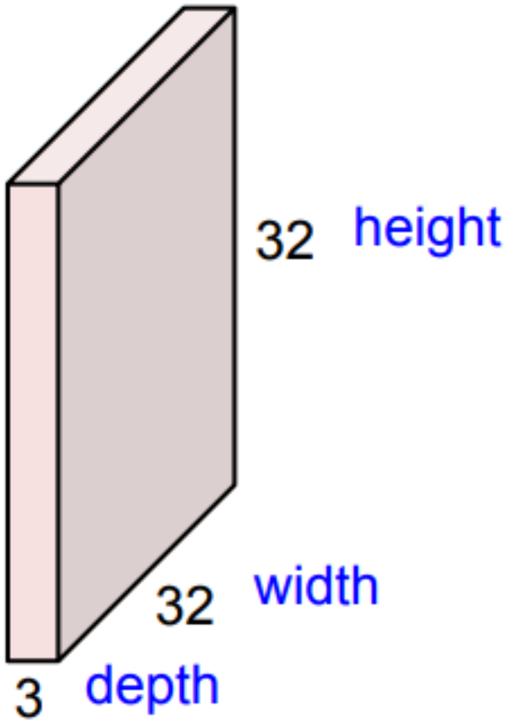
Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



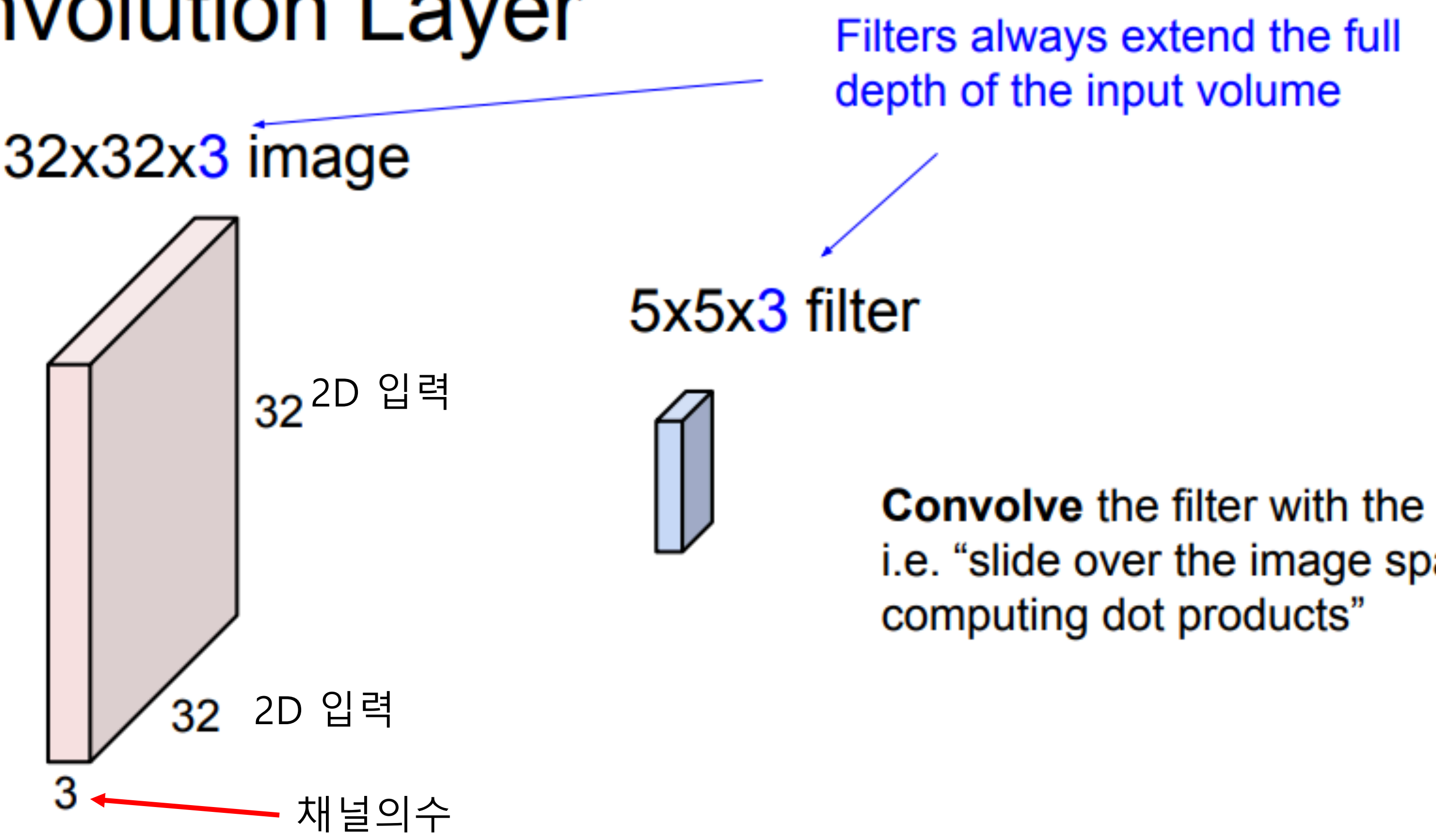
Convolution Layer

32x32x3 image -> preserve spatial structure



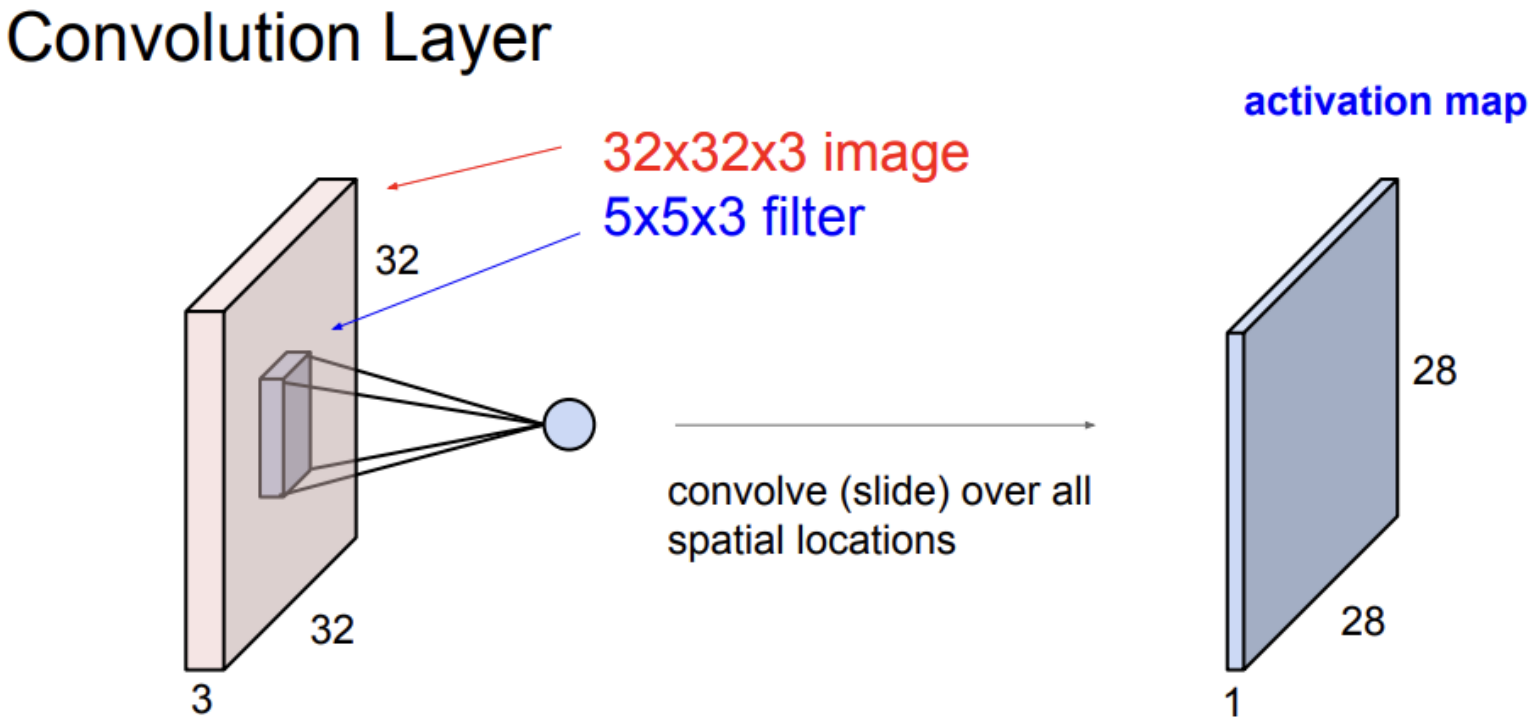
2D Convolution Layer

Convolution Layer

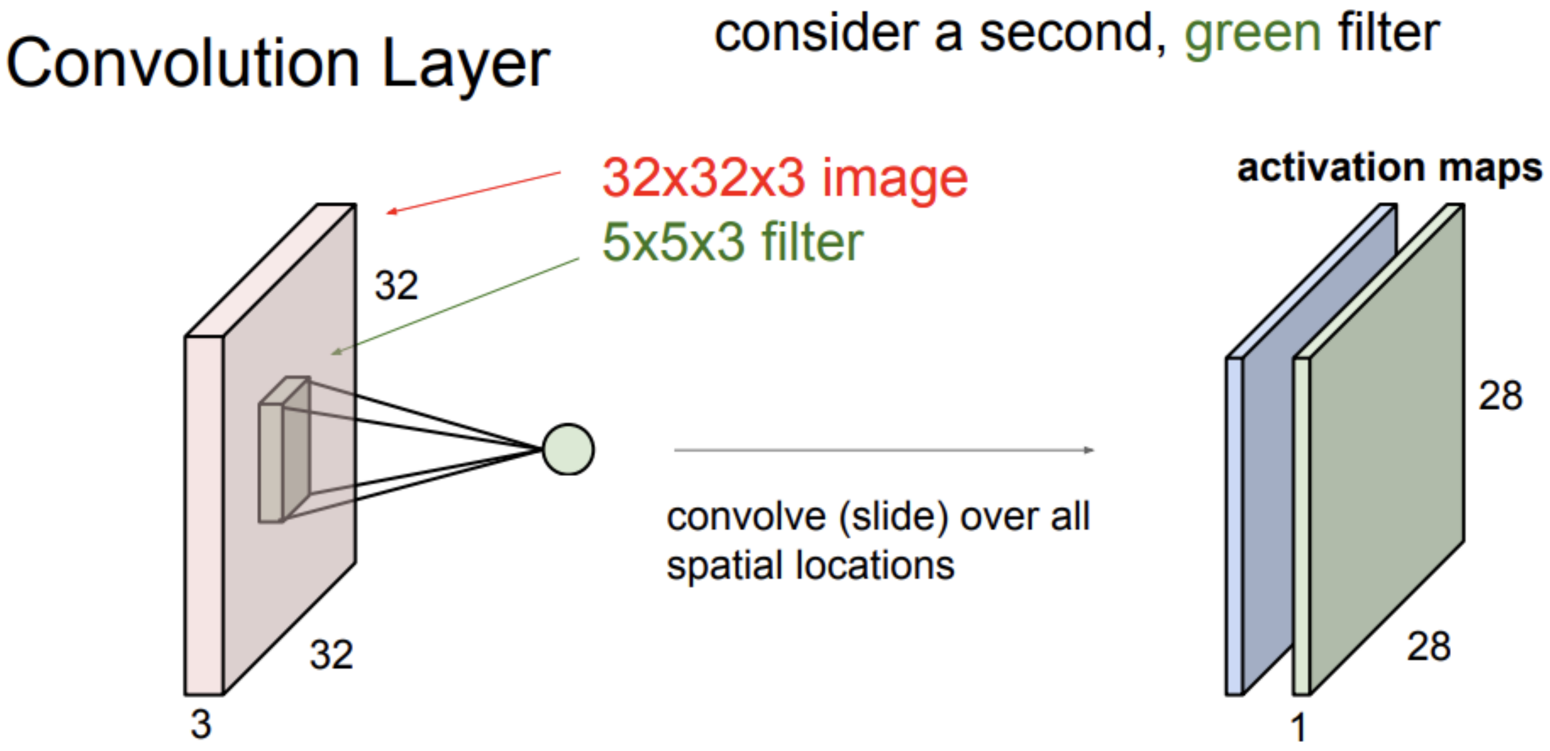


Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

2D Convolution Layer

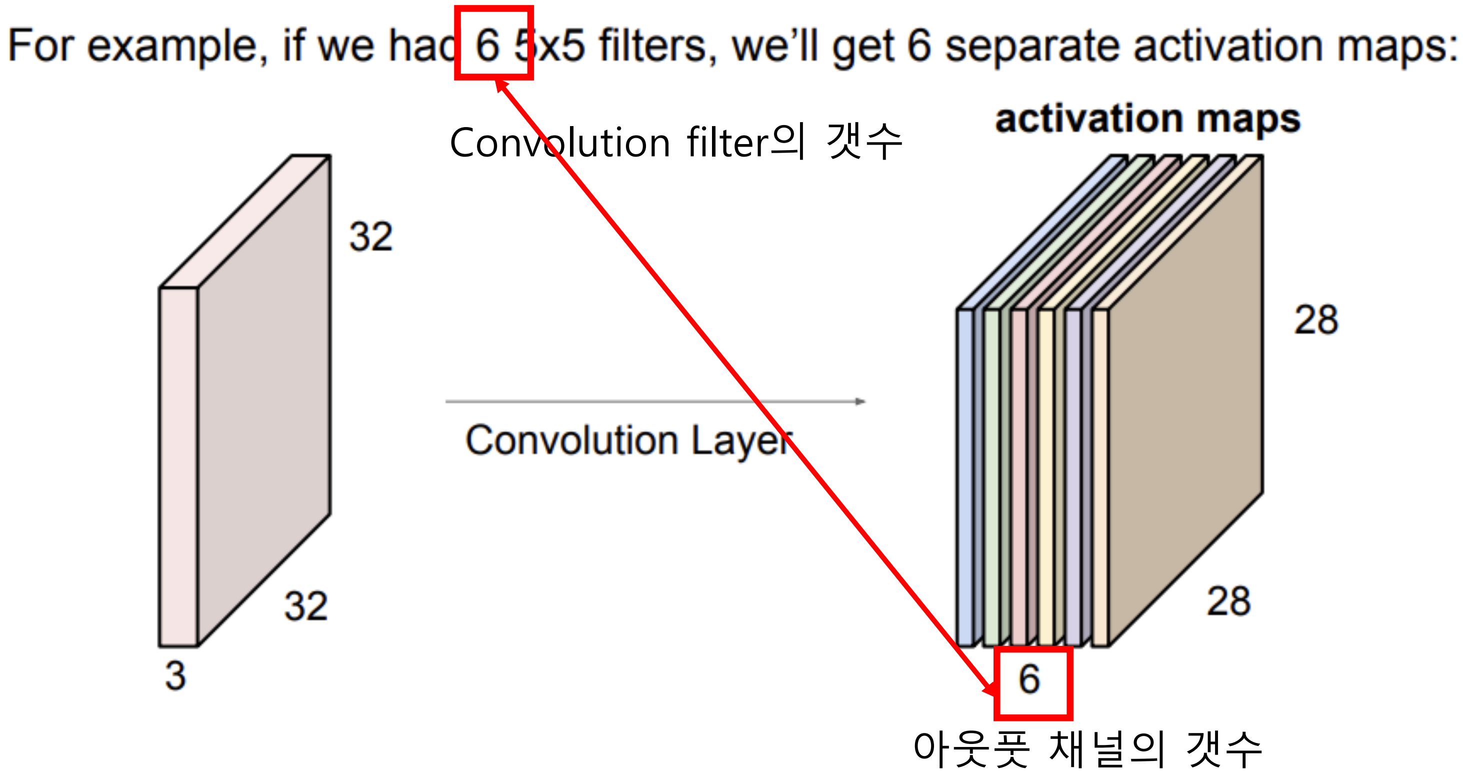


2D Convolution Layer



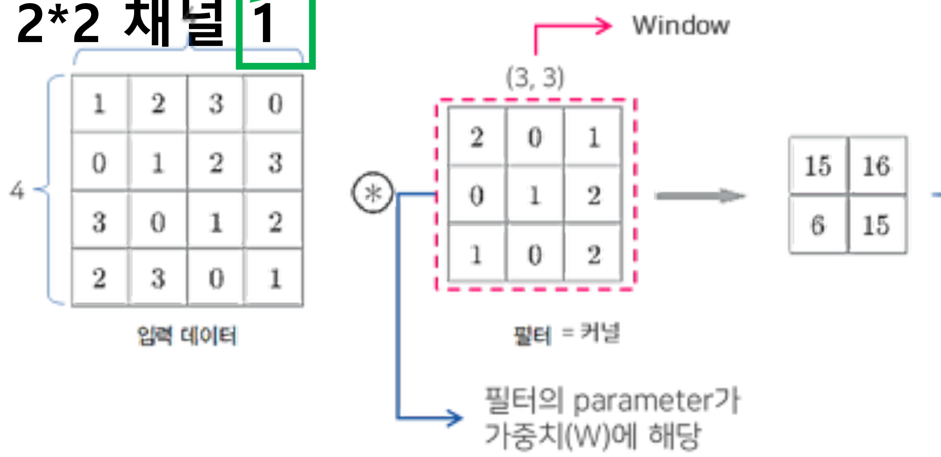
2D Convolution Layer

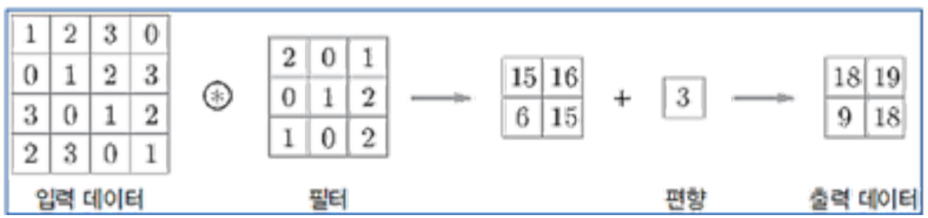
아웃풋 채널의 개수는 convolution filter의 개수와 같다.

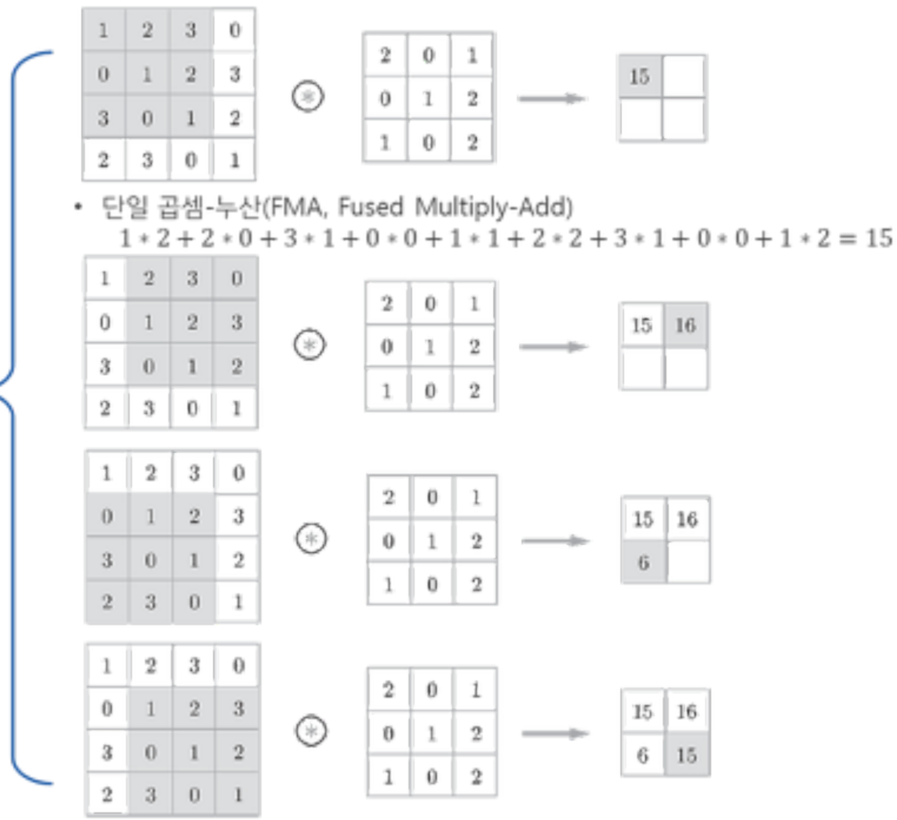


2D Convolution Layer 계산

- **입력 4*4 채널 1**
- **1개의 필터 3 * 3 채널 1**
- **결과 2*2 채널 1**

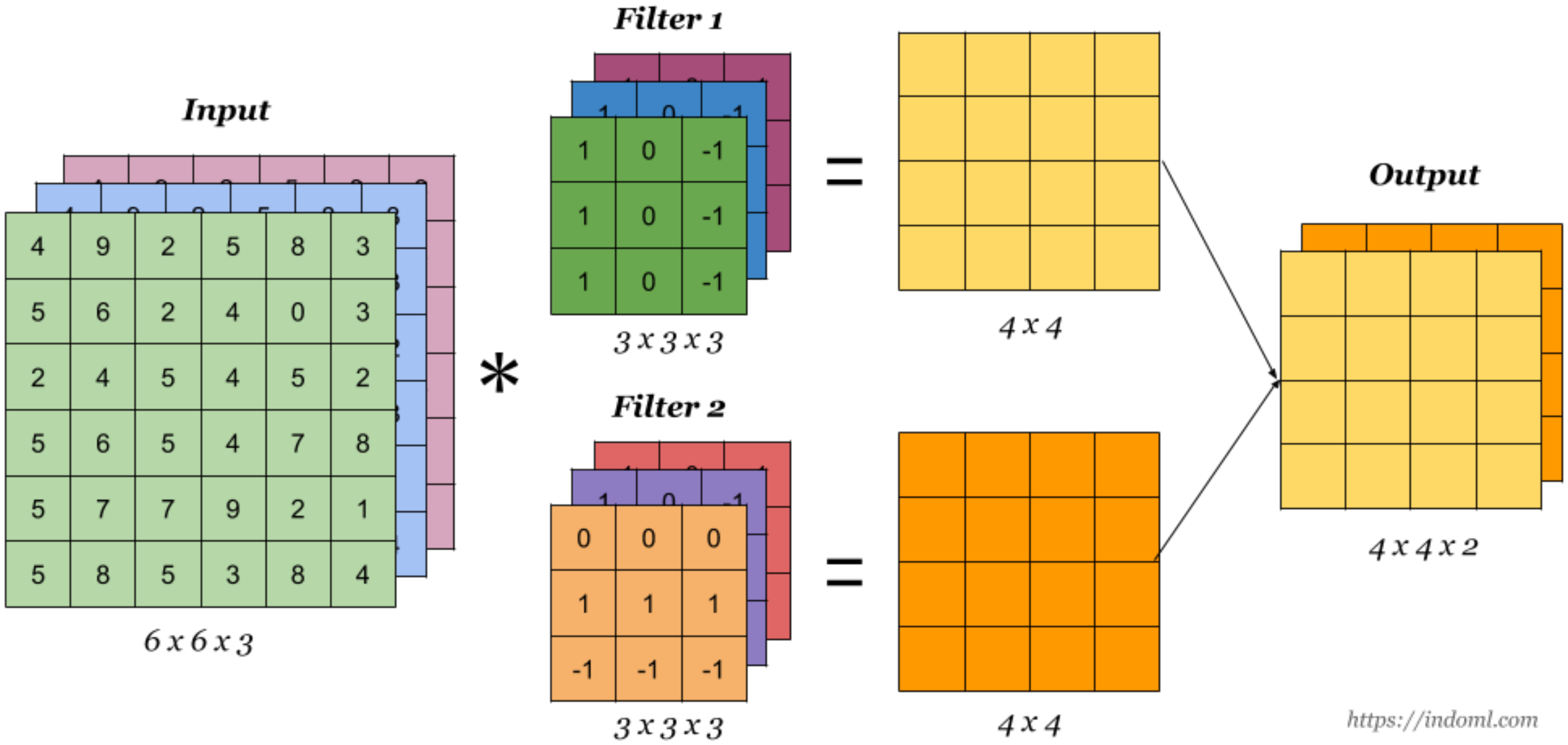






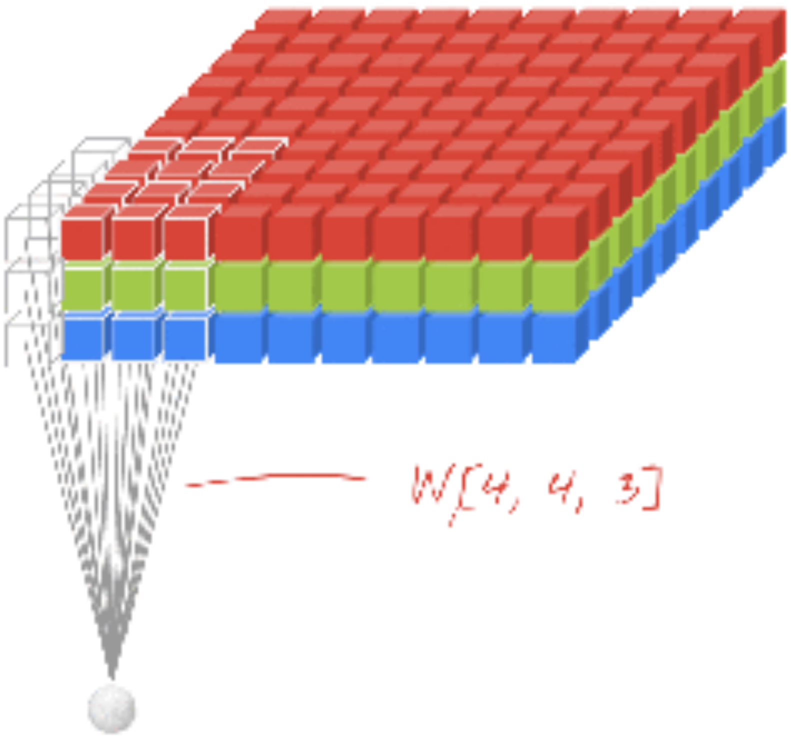
https://excelsior-cjh.tistory.com/180

2D Convolution Layer



https://indoml.files.wordpress.com/2018/03/convolution-with-multiple-filters2.png

2D Convolution Layer



https://glassboxmedicine.files.wordpress.com/2020/04/martin-gorner-rgb-conv-animation.gif?w=371

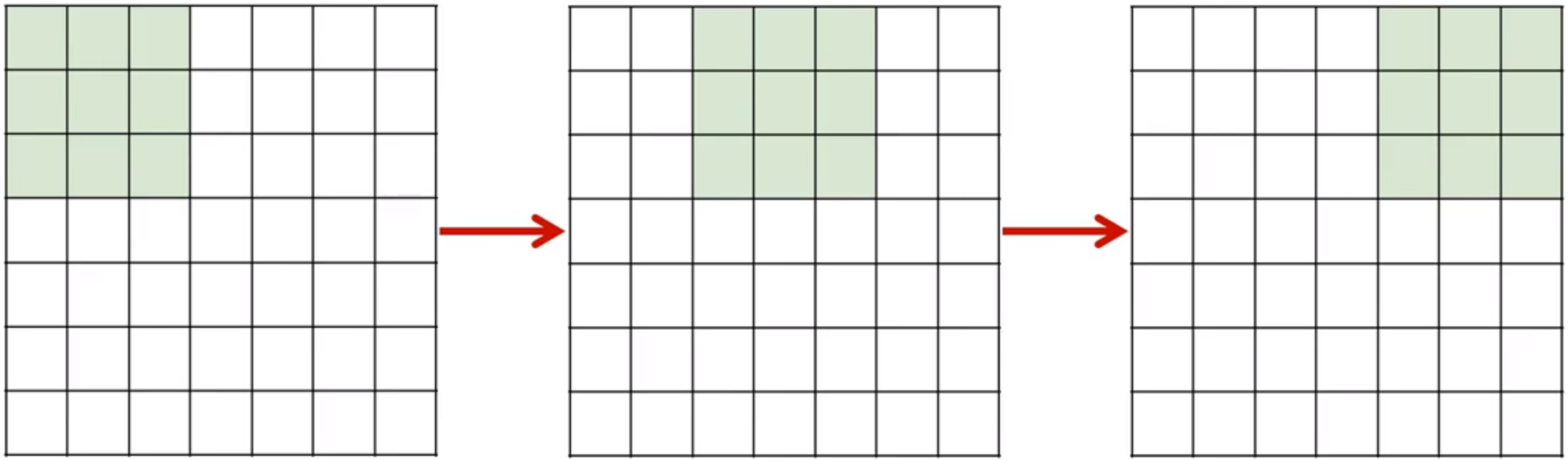
명지대학교
MYONGJI UNIVERSITY

Convolution Neural Network 2

명지대학교
MYONGJI UNIVERSITY

Convolution option - Stride

- **CNN filter 이동시 몇 칸씩 이동 한할 것인지?**
- **입력 7*7 , 필터 3*3, stride :2**



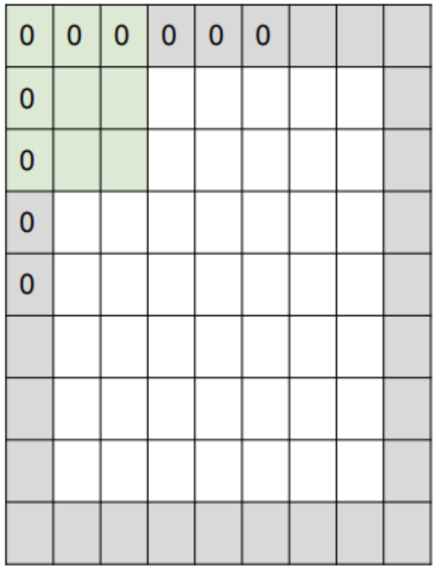
Convolution option – zero padding

Filter를 통과하면 점점 사이즈가 줄어 듦

- 앞의 예에서 32x32 (입력) 5x5(필터) -> 출력 28x28 → 점점레이어를 깊게 쌓으면? 1X1만 남게 됨
- 필터를 통과하면서 사이즈가 줄어 드는 것을 방지 하기 위해 PADDING

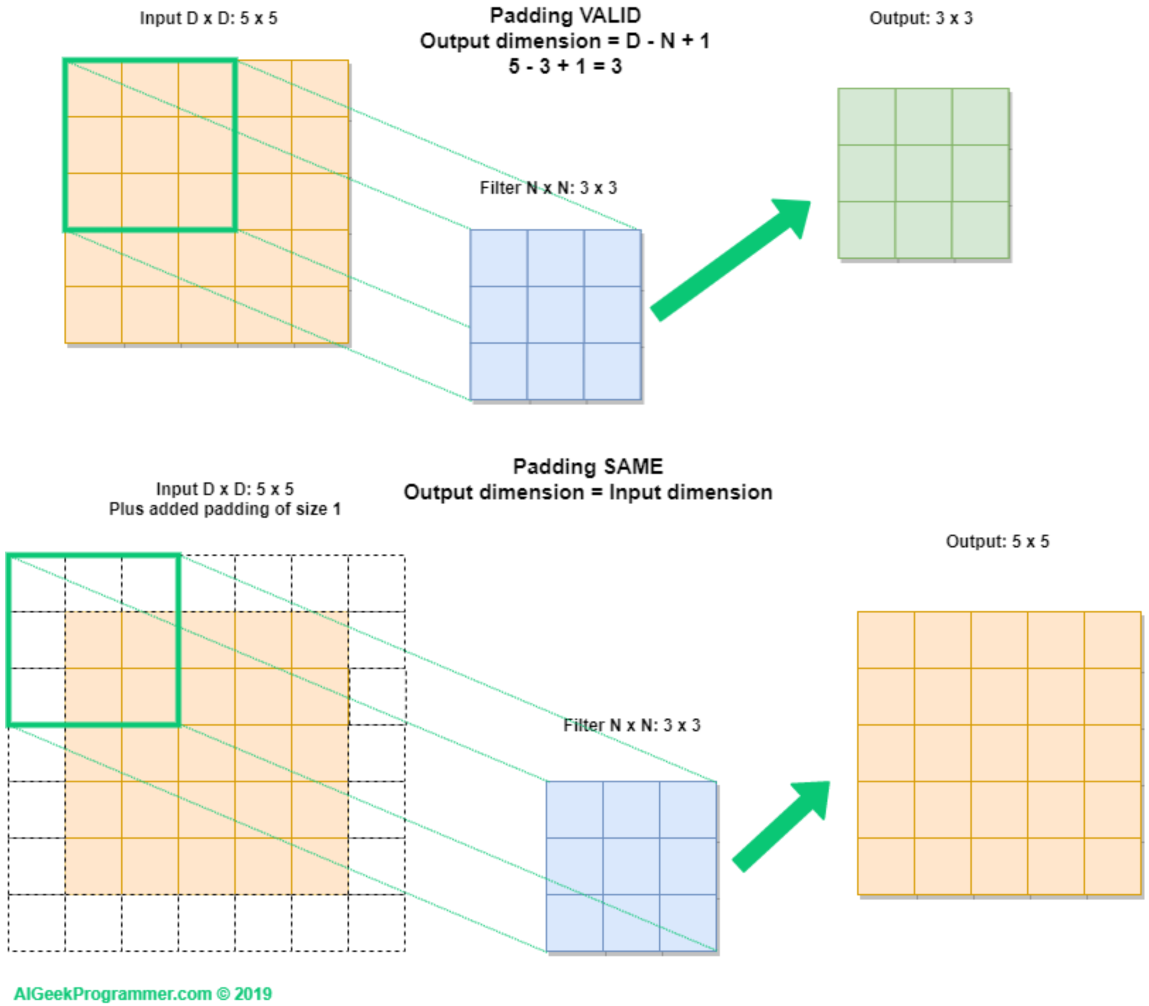
양 사방에 0을 임의적으로 넣음

- 옆의 예 7x7(입력) 3x3(필터) -> 출력 7x7



Convolution option – zero padding

- **Same : convolution후에도 사이즈가 같도록 유지**
- **Valid : convolution후에 사이즈가 줄어듬**



참고 : Tensorflow에서의 입력 채널

Data format NHWC기본 ex- **(60000, 28, 28, 1)**

- batch_shape + [height, width, channels].
- 첫번째 파라미터 : 배치 사이즈
- 두번째 파라미터 : 가로
- 세번째 파라키터 : 세로
- 네번째 파라미터 : 아웃풋 채널

•**N**: number of images in the batch
•**H**: height of the image
•**W**: width of the image
•**C**: number of channels of the image (ex: 3 for RGB, 1 for grayscale...)

- NCHW도 있음

Toy 이미지 Convolution

```
def make_toyimg():
    image = np.array([[1.],[2.],[3.],
                      [4.],[5.],[6.],
                      [7.],[8.],[9.]])
    image.astype(np.float)
    print(image.shape)
    plt.ishow(image.reshape((-1,3)), cmap='Greys')
    plt.show()
    # data format should be change to  batch_shape + [height,  width,  channels].
    image = image.reshape((1, 3, 3, 1))
    image = tf.constant(image, dtype=tf.float64)
    return image
```

Toy 이미지 Convolution

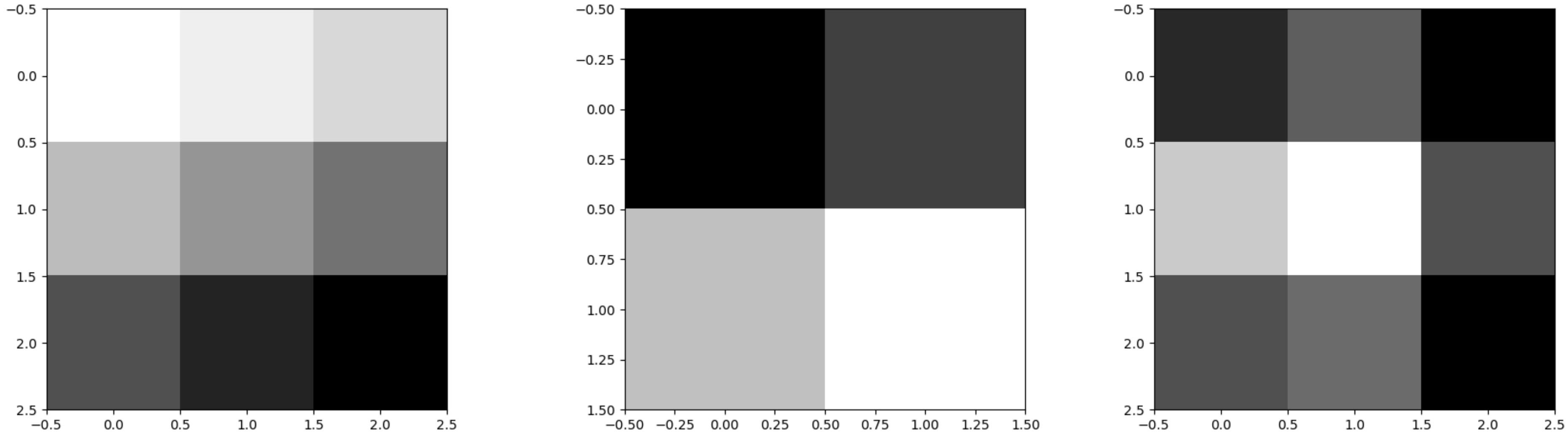
```
def make_toyfilter():
    weight = np.array([[1.],[1.],
                       [1.],[1.]])
    weight=weight.reshape((1,2,2,1))
    pint(weight.shape)
    print("weight.shape", weight.shape)
    weight_init = tf.constant_initializer(weight)

    return weight_init
```

Toy 이미지 Convolution

```
def cnn_valid(image, weight, option, sizeshape):
    conv2d = keras.layers.Conv2D(filters=1, kernel_size=2, padding=option,
                                 kernel_initializer=weight)(image)
    print("conv2d.shape", conv2d.shape)
    print(conv2d.numpy().reshape(sizeshape,sizeshape))
    plt.imshow(conv2d.numpy().reshape(sizeshape,sizeshape), cmap='gray')
    plt.show()
```

```
def main():
    img = make_toyimg()
    filter = make_toyfilter()
    cnn_valid(img, filter, 'VALID',2)
    cnn_valid(img, filter, 'SAME',3)
main()
```

Toy image를 이용한 CNN 예

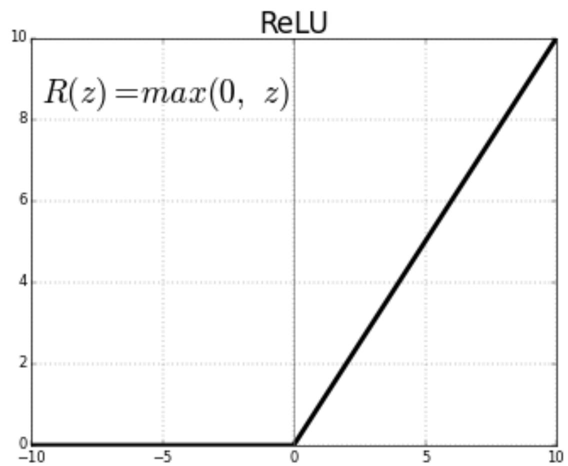


Activation Function

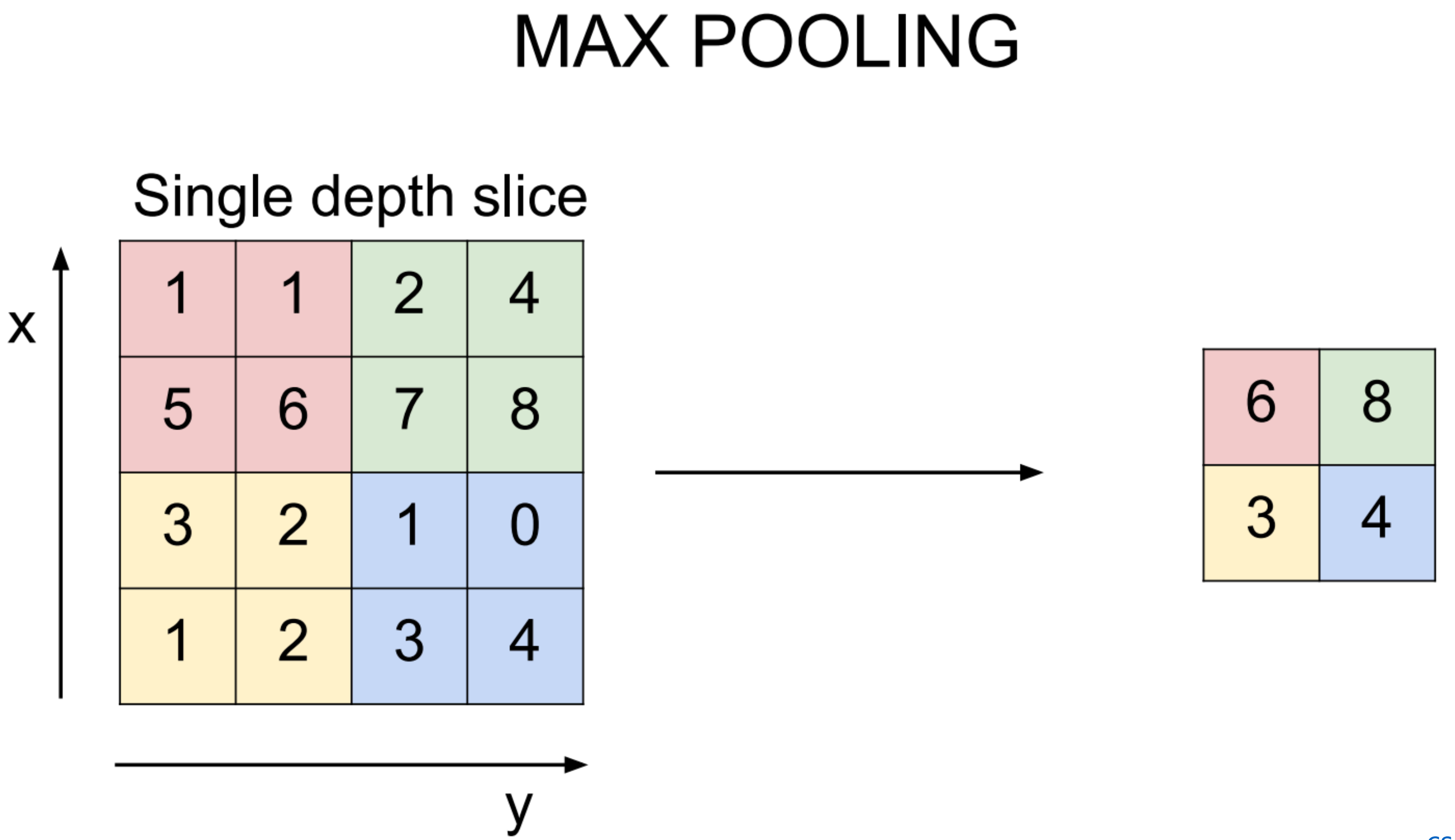
1	14	-9	4
-2	-20	10	6
-3	3	11	1
2	54	-2	80

ReLU

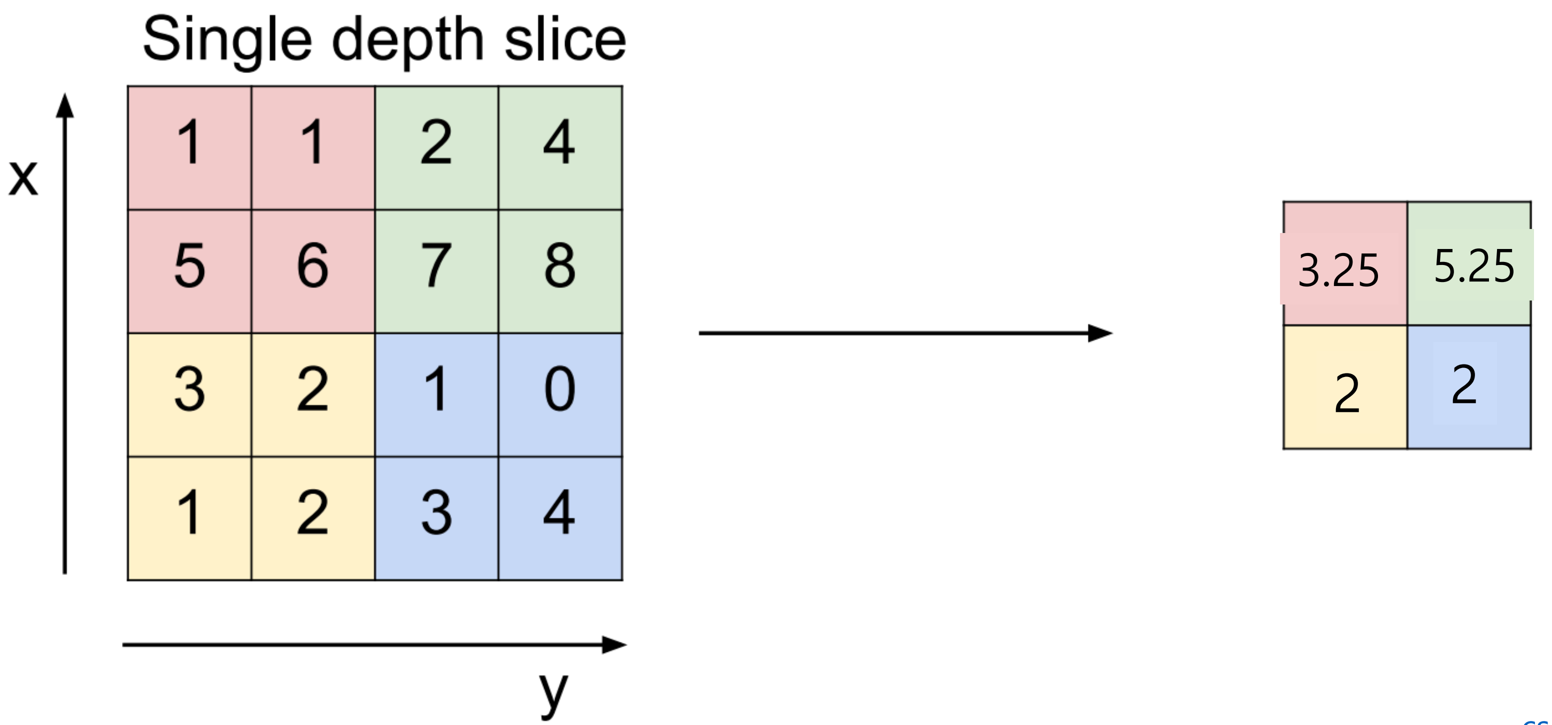
1	14	0	4
0	0	10	6
0	3	11	1
2	54	0	80



Max pooling



Average pooling



참고 : Keras Maxpooling2D

MaxPooling2D

[source]

```
keras.layers.MaxPooling2D(pool_size=(2, 2), strides=None, padding='valid', data_format=None)
```

공간적 데이터에 대한 최대값 풀링 작업.

인수

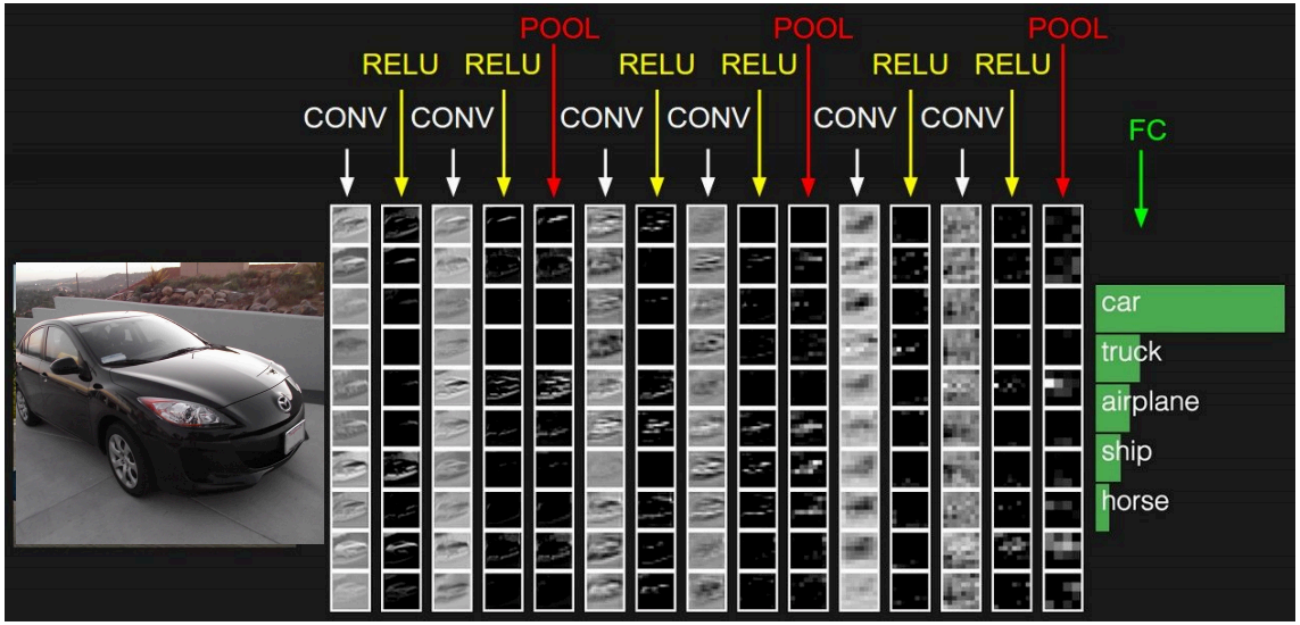
- **pool_size**: 정수 혹은 2개 정수의 튜플, 축소 인수 (가로, 세로). (2, 2)는 인풋을 두 공간 차원에 대해 반으로 축소합니다. 한 정수만 특정된 경우, 동일한 윈도우 길이가 두 차원 모두에 대해 적용됩니다.
- **strides**: 정수, 2개 정수의 튜플, 혹은 None. 보폭 값. None일 경우, 디폴트 값인 `pool_size`가 됩니다.
- **padding**: `"valid"` 혹은 `"same"` 중 하나 (대소문자 무시).
- **data_format**: 문자열, `channels_last` (디폴트 값) 혹은 `channels_first` 중 하나. 인풋 차원의 순서. `channels_last`는 `(batch, height, width, channels)` 형태의 인풋에 호응하고, `channels_first`는 `(batch, channels, height, width)` 형태의 인풋에 호응합니다. 디폴트 값은 `~/.keras/keras.json`에 위치한 케라스 구성 파일의 `image_data_format` 값으로 지정됩니다. 따로 설정하지 않으면, 이는 `"channels_last"`가 됩니다.

https://keras.io/ko/layers/pooling/

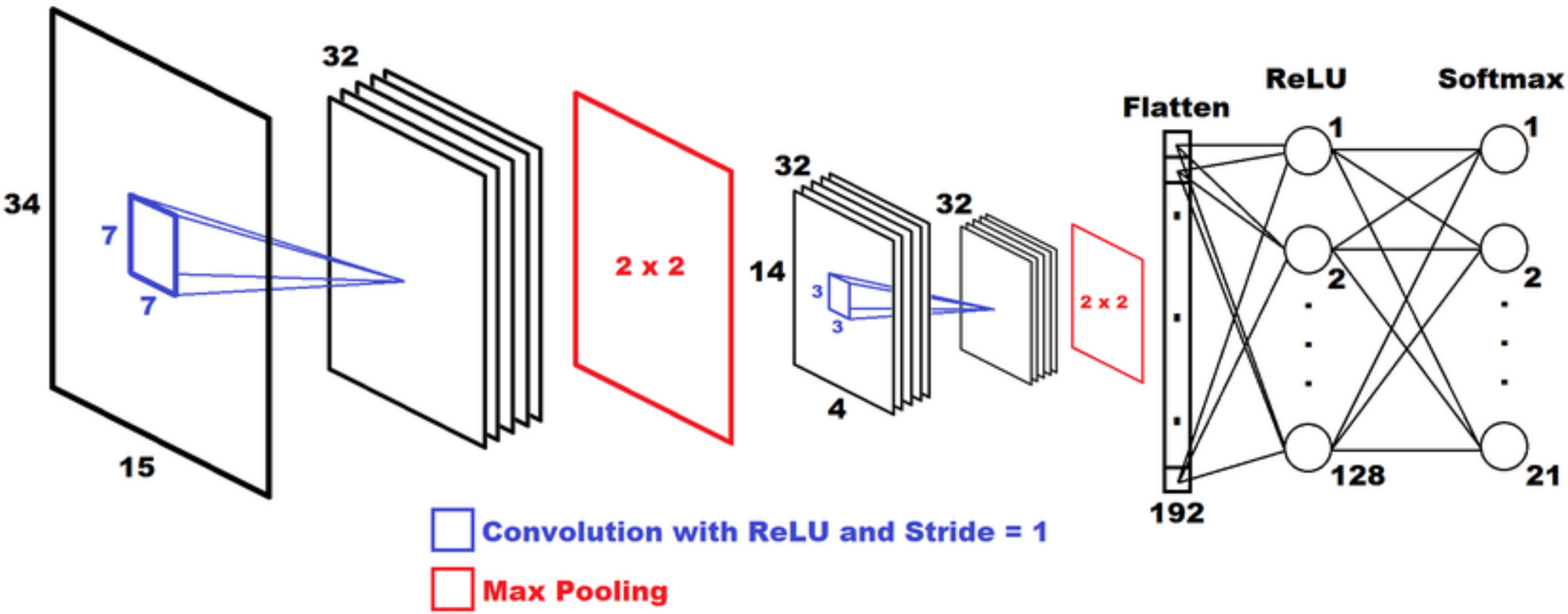
Fully connected layer

Fully Connected Layer (FC layer)

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks

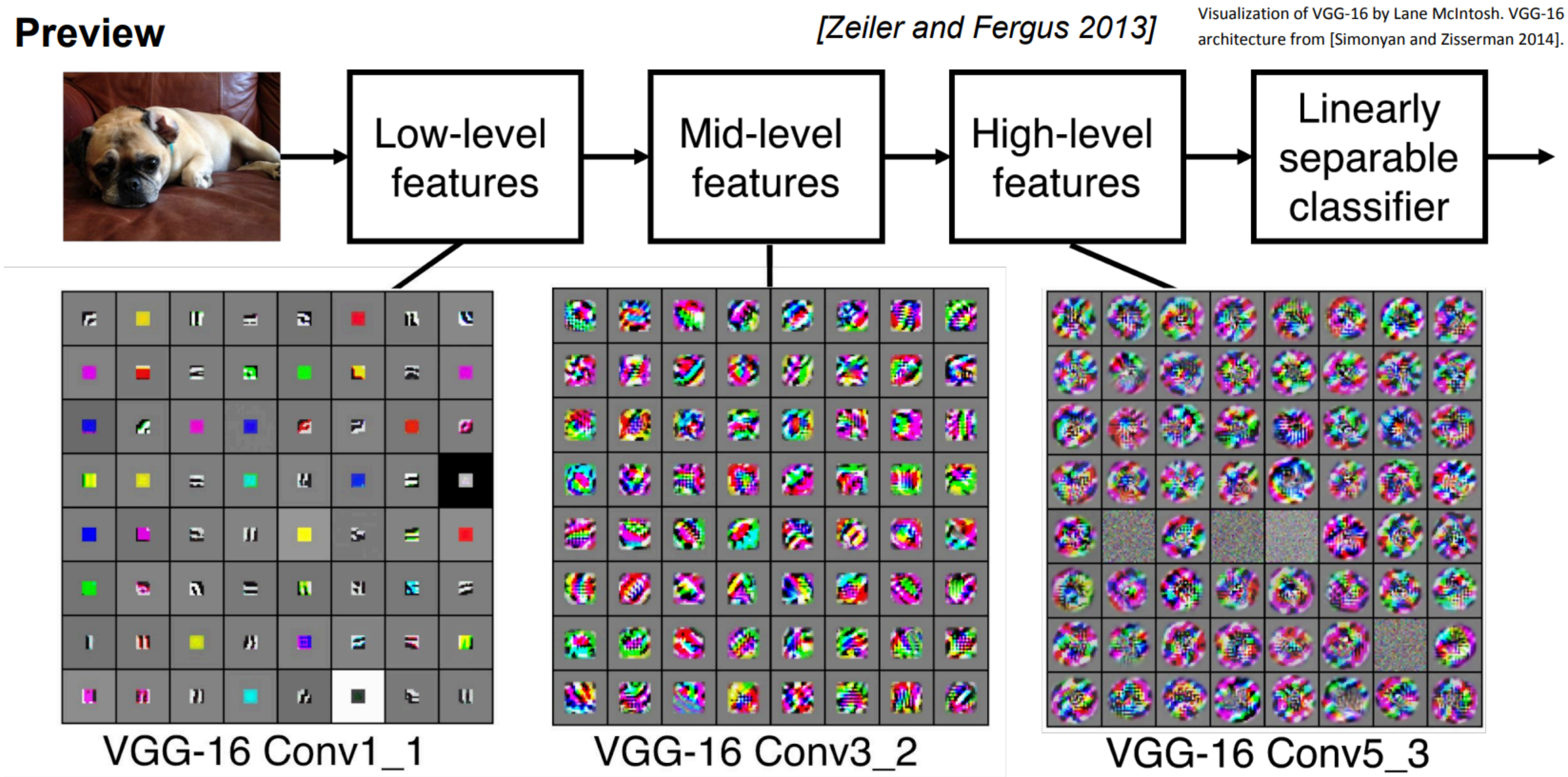


Fully connected layer



https://www.researchgate.net/profile/Zohaib_Iqbal2/publication/321241655/figure/fig1/AS:563693125632000@1511406322490/The-structure-of-the-convolutional-neural-network-is-displayed-Initially-two.png

Feature


Preview
[Zeiler and Fergus 2013]
Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].
Low-level features
Mid-level features
High-level features
Linearly separable classifier
VGG-16 Conv1_1
VGG-16 Conv3_2
VGG-16 Conv5_3

참고

참고 :

- http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture05.pdf
- https://github.com/deeplearningzerotoall/TensorFlow

Tensorflow

- https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D