# Practical Computer Vision Week 1

Computer Vision and CNN



KU-BIG

# Intro and TMI

# What Will We Do?

# What will we do? – In Session

- A New Computer Vision Application for each Week
- Presentation or Activity
- Introductory Lecture
- Keras / Tensorflow Lab

# What will we do? – Homework

- Programming Assignment
- Prepare Presentation
- Review Materials
- Optional Advanced Study Materials

# Syllabus

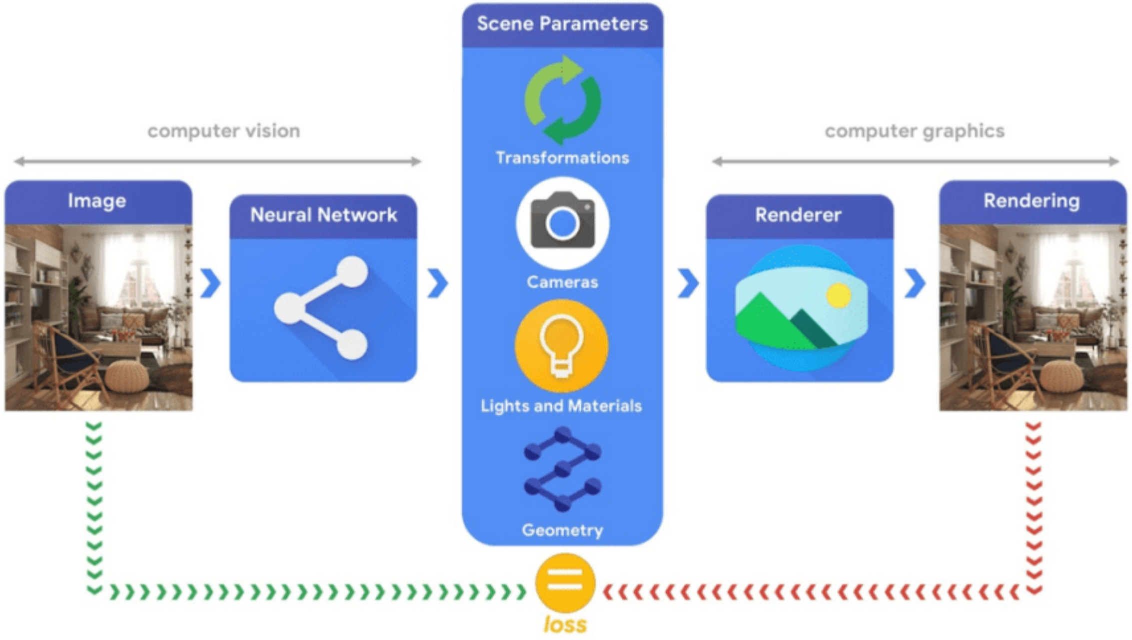
| Week | Class 1 | Class 2 | Homework |
|---|---|---|---|
| Week 1 ( 01. 14)<br>CNN and openCV | What Will We Do?<br>Review: CNN | OpenCV Practice<br>(LAB Session) | Image and Video Processing with openCV |
| Week 2 (01. 21)<br>CNN Architecture | Advancements in CNN Architecture | Image Classification with Deep Learning<br>(LAB Session) | Image Classification<br>(Dogs vs Cats) |
| Week 3 (01.28)<br>Object Detection | Object Localization and Detection | SSD & MobileNet<br>(LAB Session) | Car Detection with YOLO |

# Syllabus

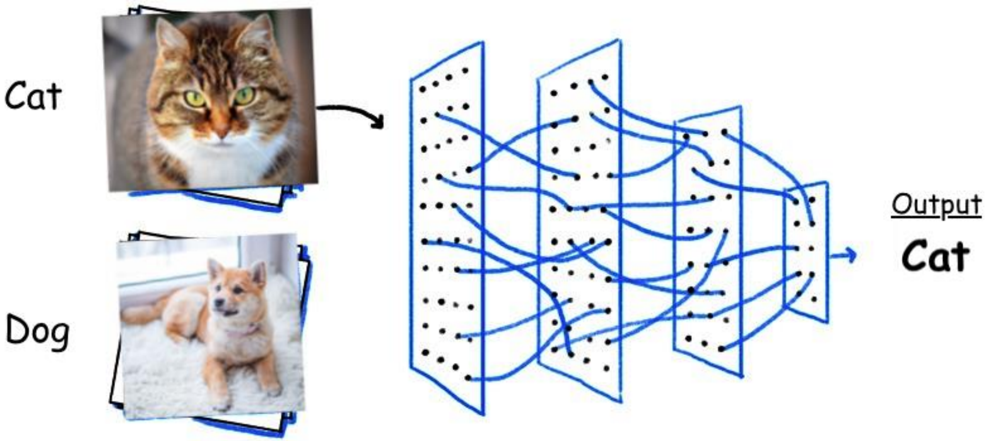
| Week | Class 1 | Class 2 | Homework |
|---|---|---|---|
| Week 4 ( 02. 04)<br>Landmark Detection &<br>Image Segmentation | Features and Object Recognition | Image Segmentation (LAB) | Face Detection and Recognition |
| Week 5 (02. 18)<br>Landmark Detection<br>and Tracking | Object Motion and Tracking | Optical Flow and Feature Matching | Vehicle Localization with SLAM |
| Week 6 (02. 25)<br>Neural Style Transfer | Neural Style Transfer | Art generation with Neural Style Transfer (LAB) | Art Generation Team Project |
| Week 7 (03. 04)<br>GANs | GANs Intuition | Image Creation with GANs (LAB) | Image Augmentation using GANs |

# Various Applications of Computer Vision

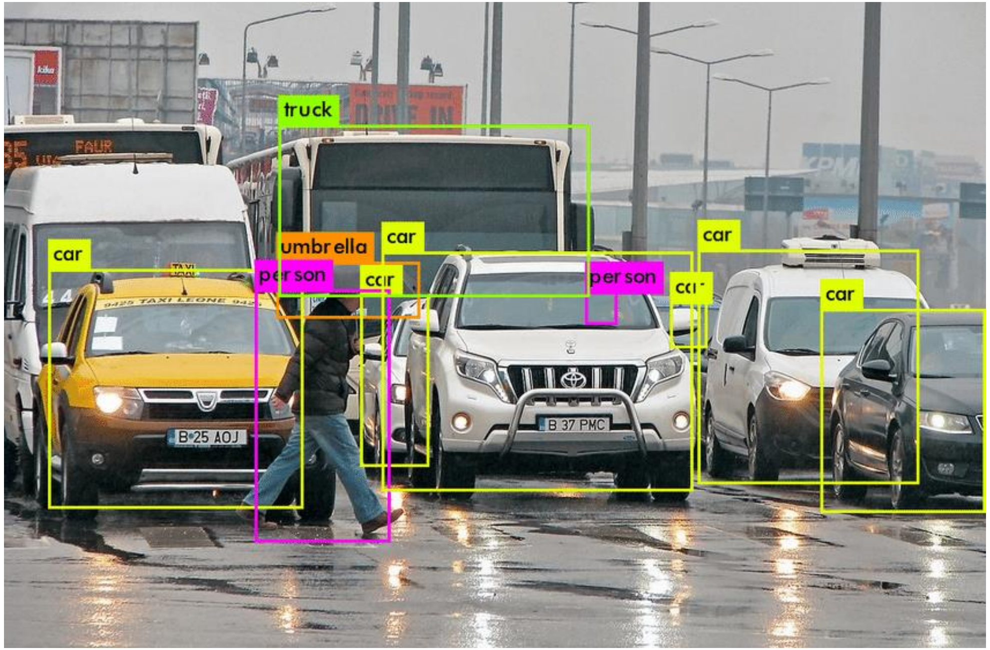
# What is Computer Vision?



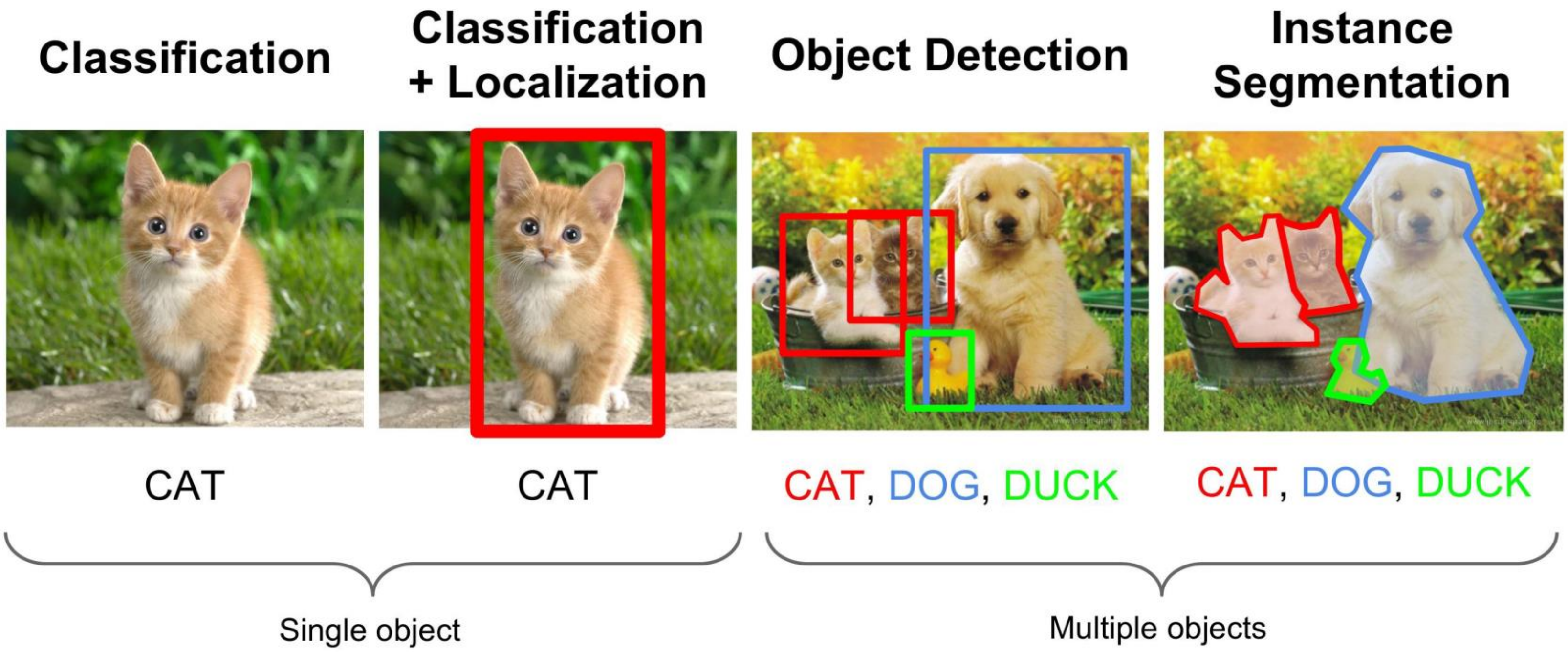
# Image Classification (Week 2)



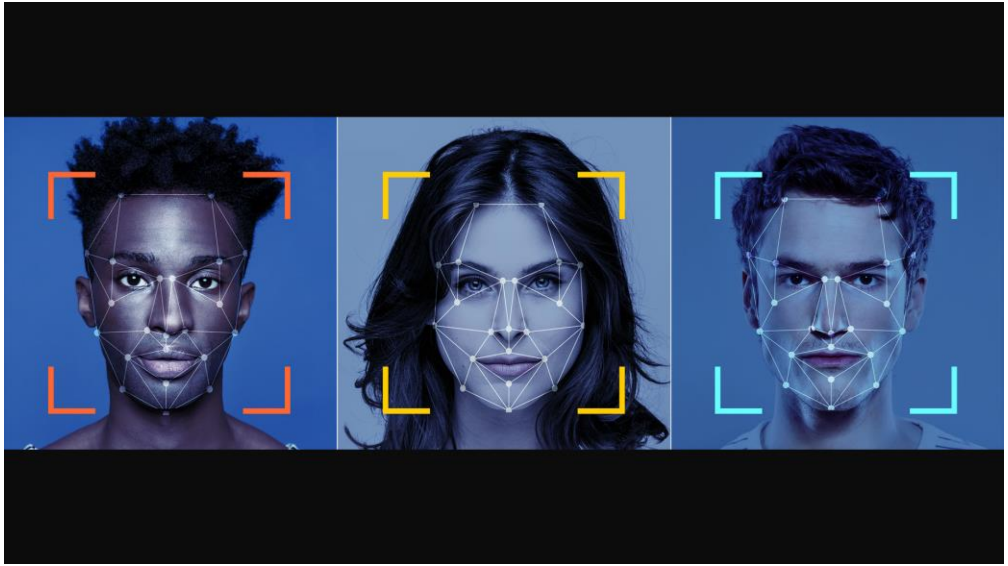
# Object Detection (Week 3)



# Image Segmentation (Week 4)



# Landmark Detection and Facial Recognition (Week 4)



# Neural Style Transfer (Week 6)

왼쪽 Content

오른쪽. Style

# Generative Adversarial Networks (Week 7)

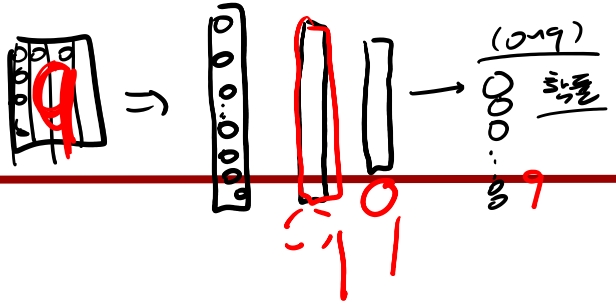
진짜

생성

# Review: CNN

# CNN In Edge Detection

True
/False




| | | | | | |
|---|---|---|---|---|---|
| 10 | 10 | 10 | 0 | 0 | 0 |
| 10 | 10 | 10 | 0 | 0 | 0 |
| 10 | 10 | 10 | 0 | 0 | 0 |
| | | | 0 | 0 | 0 |
| | | | 0 | 0 | 0 |
| | | | 0 | 0 | 0 |

*

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 1 | 0 | -1 |
| 1 | 0 | -1 |

filter

=

| | | | |
|---|---|---|---|
| 0 | 30 | 30 | 0 |
| 0 | 30 | 30 | 0 |
| 0 | 30 | 30 | 0 |
| 0 | 30 | 30 | 0 |

밝은

어두운

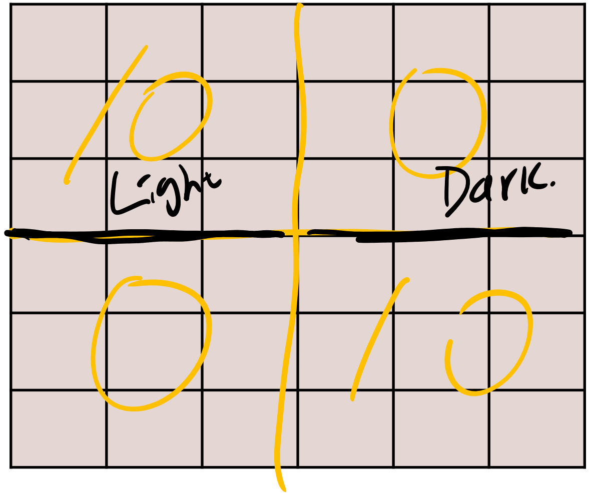
256

# CNN In Edge Detection

Backpropagation
G.D

36x16

16




*

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

적절한 필터

9개

+ bias

=

| Light | | | Dark |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 30 | 10 | -10 | -30 |
| 30 | 10 | -10 | 30 |
| 0 | 0 | 0 | 0 |
| Dark | | | Light |

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 2 | 0 | -2 |
| 1 | 0 | -1 |

"Sobel"

| | | |
|---|---|---|
| 3 | 0 | -3 |
| 10 | 0 | -10 |
| 3 | 0 | -3 |

Scharr Filter

Parameter 10개

# Padding

$(n+2p-f+1) * (n+2p-f+1)$

- Image 가 작아진다

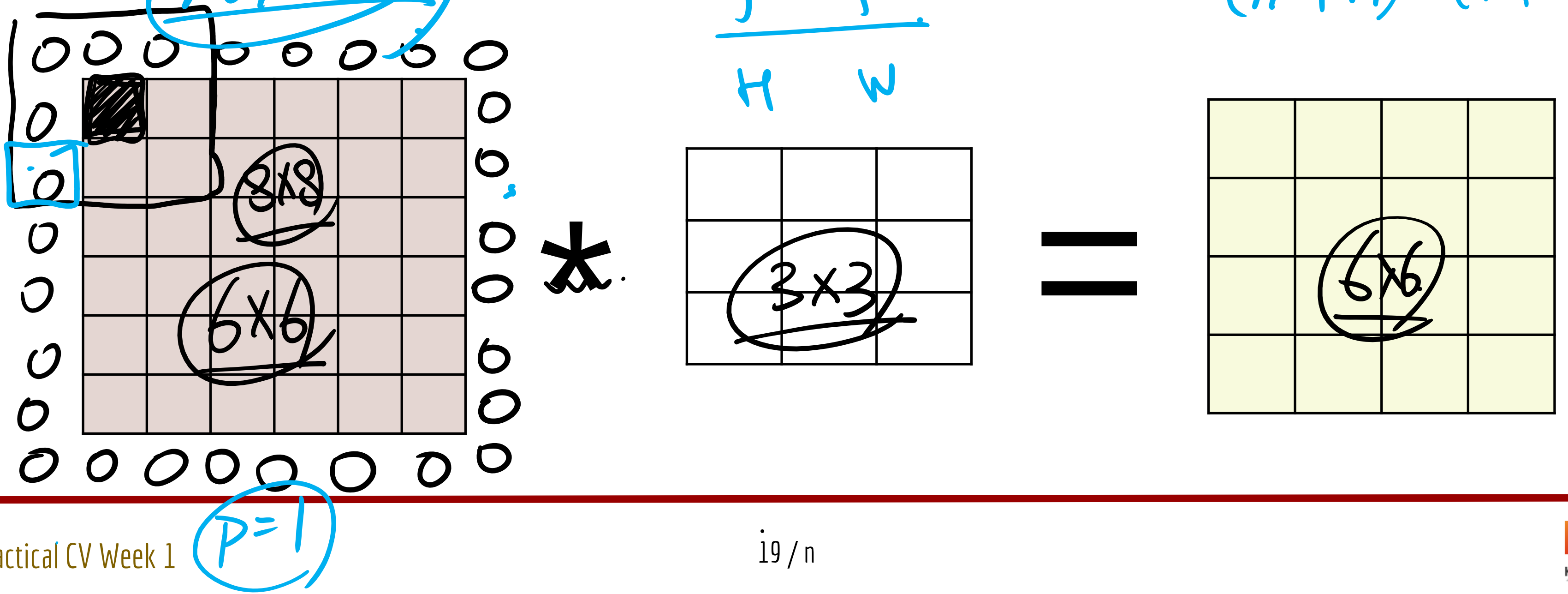
$(n-f+1) * (n-f+1)$

- 각 픽셀의 계산빈도의 분산이 크다

$n \times n$

$f \times f$

H W



# Padding

- **Valid Convolution**: No padding

- **Same Convolution**: Pad so that Output size as the same as the input size

# Strided Convolutions

| | | | | | | |
|---|---|---|---|---|---|---|
| 2 | 3 | 7 | 4 | 6 | 2 | 9 |
| 6 | 6 | 9 | 8 | 7 | 4 | 3 |
| 3 | 4 | 8 | 3 | 8 | 9 | 7 |
| 7 | 8 | 3 | 6 | 6 | 3 | 4 |
| 4 | 2 | 1 | 8 | 3 | 4 | 6 |
| 3 | 2 | 4 | 1 | 9 | 8 | 3 |
| 0 | 1 | 3 | 9 | 2 | 1 | 4 |

$n * n$

$*$

$f * f$

| | | |
|---|---|---|
| 3 | 4 | 4 |
| 1 | 0 | 2 |
| -1 | 0 | 3 |

=

| | | |
|---|---|---|
| | | |
| | | |
| | | |

| | | |
|---|---|---|
| 6 | 12 | 28 |
| 6 | 0 | 18 |
| 3 | 0 | 24 |

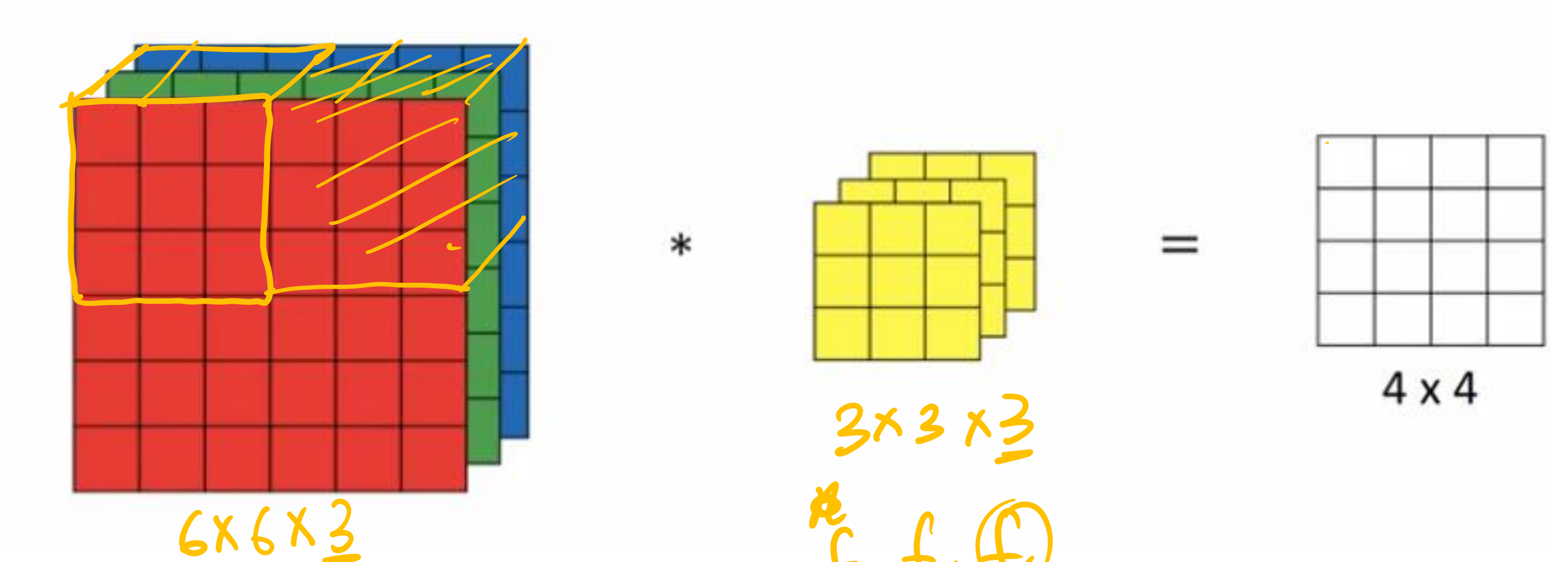
$$\left\lfloor \frac{n+2p-f}{s} \right\rfloor + 1$$

최종값.



KU-BIG

# Convolutions Over Volume



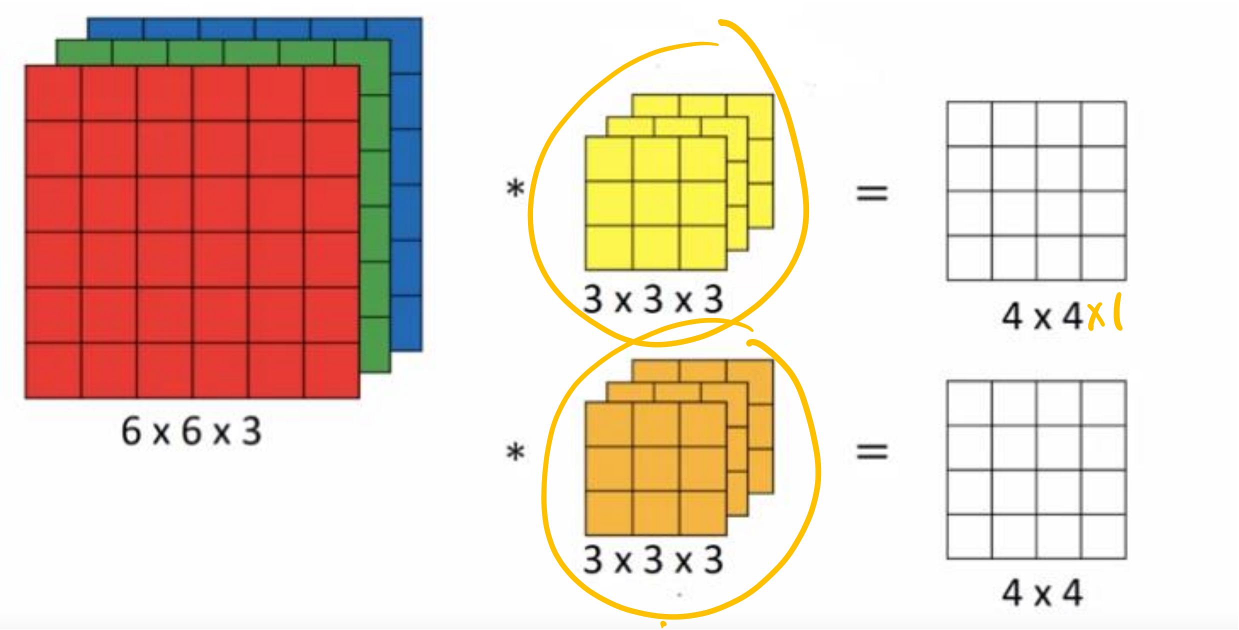

RGB

$n_H$ $n_W$ $n_C$

$f_H$ $f_W$ $f_C$

$n_C = f_C$

# Convolutions Over Volume



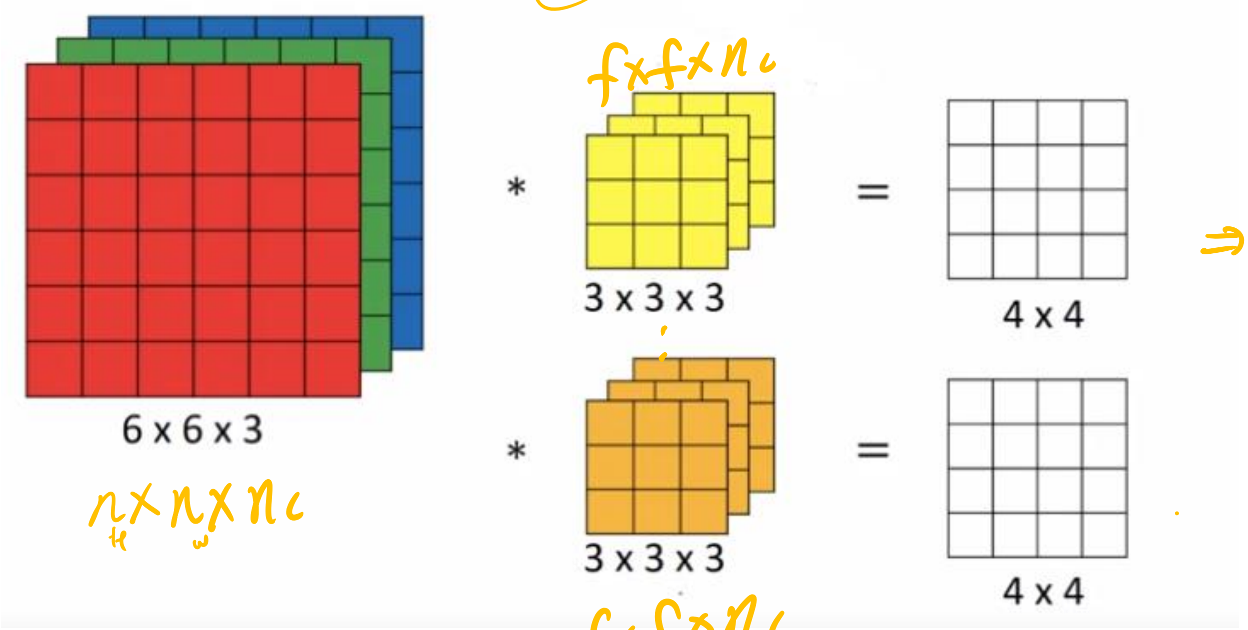

filter를 여러개

# One Layer of Convolutional Network

$$\left\lfloor \frac{n+2p-f}{s} \right\rfloor + 1$$

3 5 7

$f \times f \times n_c$

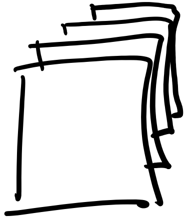



$n_H \times n_W \times n_c$

$f \times f \times n_c$

$n_f'$

$\Rightarrow$



$(n-f+1) \times (n-f+1) \times n_f'$

# Summary of Notation

$f^{[l]}$ = filter size

$p^{[l]}$ = padding

$s^{[l]}$ = stride

$n_c^{[l]}$ = number of filters

*Input* : $n_H^{[l-1]} \times n_w^{[l-1]} \times n_c^{[l-1]}$

*Output* : $n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$

$$n^{[l]} = \left\lfloor \frac{n^{[l-1]} + 2P^{[l]} - f^{[l]}}{s^{[l]}} \right\rfloor + 1$$

*Each Filter Size*: $f^{[l]} \times f^{[l]} \times n_c^{[l]}$

*Activations* : $a^{[l]} \rightarrow n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$

*Set of m Activations* : $A^{[l]} \rightarrow m \times n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$

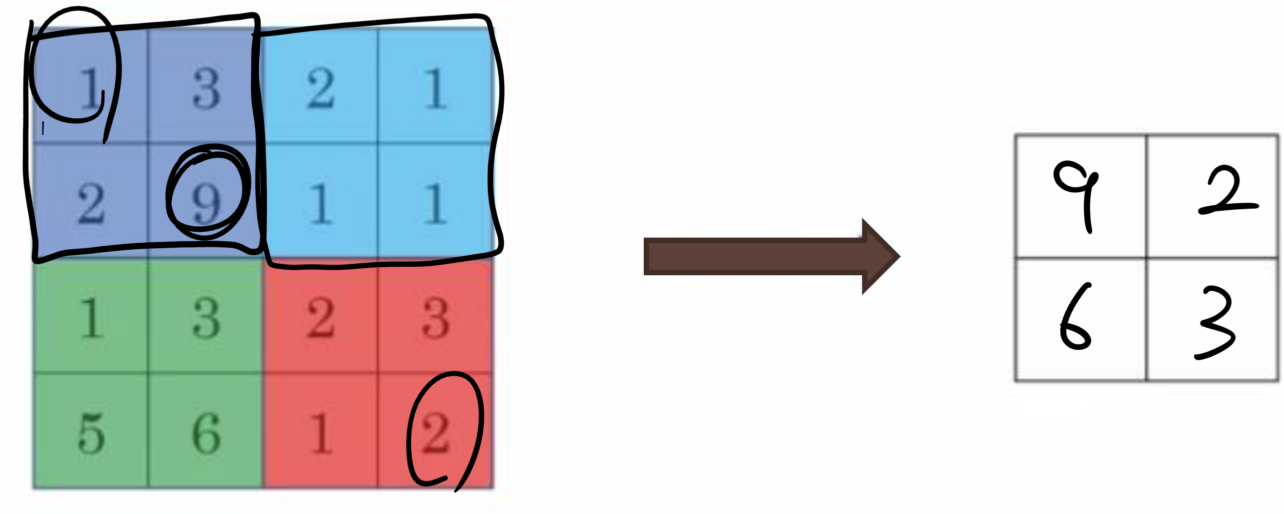
*Each Filter Size*: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$

# Pooling Layer

MaxPooling

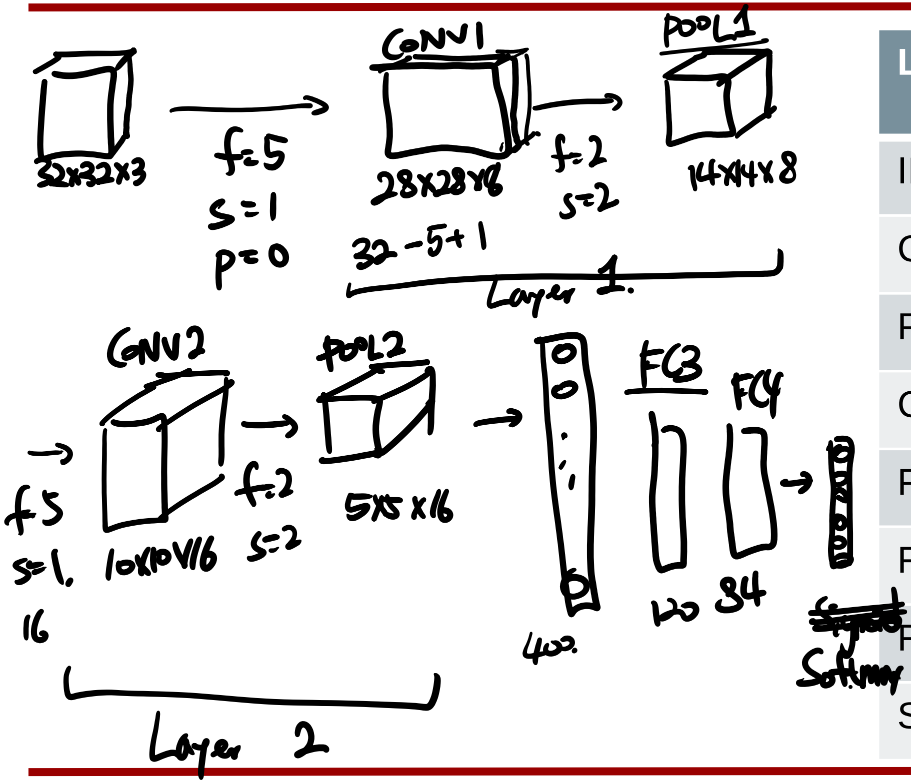
Average Pooling

- **3 Types of Layers**
- Convolution
- Pooling
- Fully Connected




$n=2$, $s=2$

# CNN Example LeNet-5




| Layer | Activation shape | Activation Size | # of Parameters |
|---|---|---|---|
| INPUT | (32, 32, 3) | 3072 | 0 |
| CONV1 | (28. 28. 8) | 6072 | (5x5x3+1)x8 =6 |
| POOL1 | (14., 14, 8) | 1568 | 0 |
| CONV2 | (10, 10, 16) | 1600 | (5x5x8+1)x16 |
| POOL2 | (5, 5, 16) | 400 | |
| FC3 | (120, 1) | 120 | |
| FC4 | (84, 1) | 84 | |
| Softmax | (10, 1) | 10 | |

# CNN Example

| Layer | Activation shape | Activation Size | # of Parameters |
|---|---|---|---|
| INPUT | (32, 32, 3) | 3072 | 0 |
| CONV1 | (28. 28. 8) | 6072 | (5*5*3+1)*8 = 608 |
| POOL1 | (14., 14, 8) | 1568 | 0 |
| CONV2 | (10, 10, 16) | 1600 | (5*5*8+1)*16 = 3216 |
| POOL2 | (5, 5, 16) | 400 | 0 |
| FC3 | (120, 1) | 120 | 400*120+ 120 = 48120 |
| FC4 | (84, 1) | 84 | 120*84+84 = 10164 |
| Softmax | (10, 1) | 10 | 84*10 + 10 = 850 |

# openCV

# Assignments

# HOMEWORK

- **GROUP Assignment**
- Study and Present CNN Architectures

- AlexNet + VGG
- GoogleNet (Inception) + Xception
- ResNet + ResNeXt

- **INDIVIDUAL Self Practice**

- OpenCV Practice
- CNN Practice (numpy, keras, tensorflow, pytorch)