Deep Neural Network
(Convolutional Neural Network)

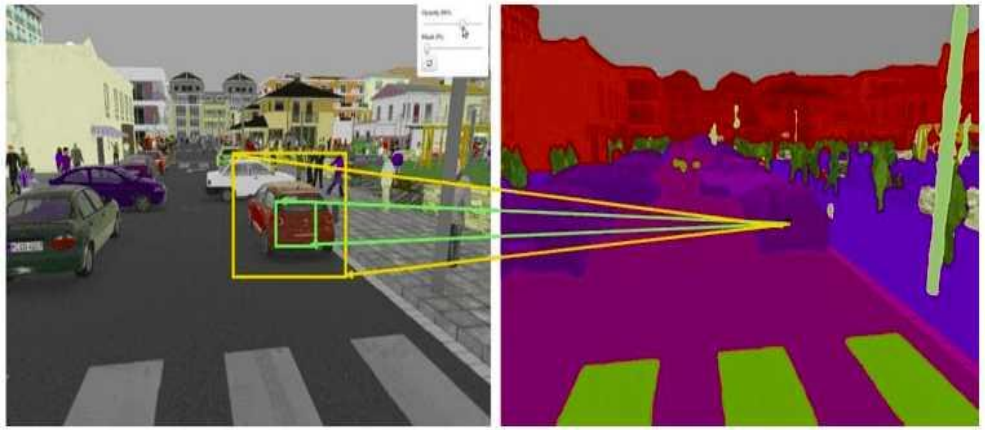
Convolutional Neural Network

- Hierarchical visual information processing of biological neural networks
  - David Hubble and Thorsten Niels Vissel published a study in 1959 on the structure and function of the visual cortex of animals

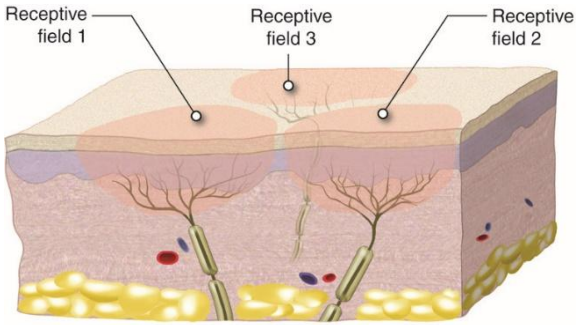
  - Neurons respond to very narrow areas of stimulation
  - The role of neurons in recognizing different shapes of characteristics
  - Hierarchical processing of visual information

Convolutional Neural Network

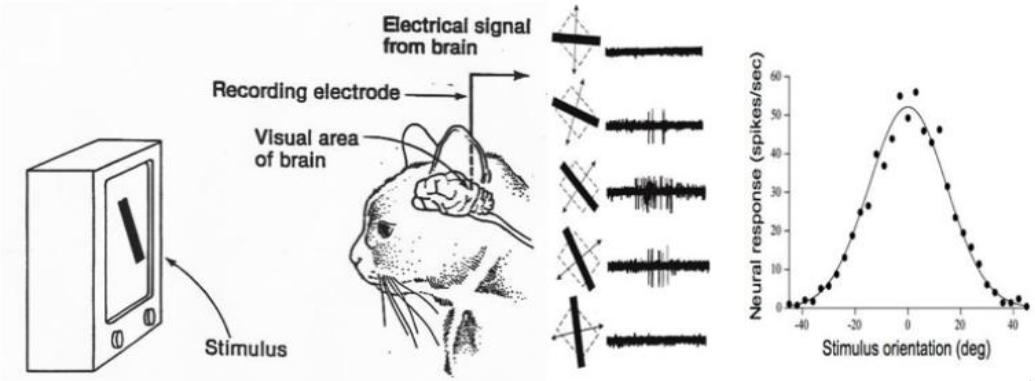
❖ Principles of visual information processing



고 수준의 뉴런이 저 수준의 뉴런의 출력에 기반한다



"고양이의 눈에서 답을 얻다."



출처: cs231

출처: https://brainconnection.brainhq.com/2004/03/06/overview-of-receptive-fields/
https://theaisummer.com/receptive-field/

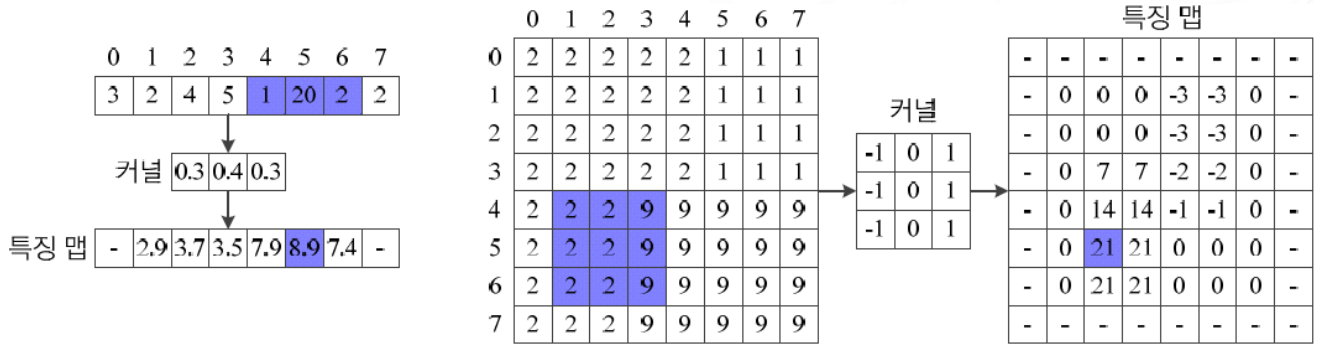
Convolution Layer

❖ Convolution

- Convolution is a linear operation that multiplies the corresponding elements and adds all the results
- u: kernel, z: input, s: output (feature map)

$$s(i) = z \circledast u = \sum_{x=-(h-1)/2}^{(h-1)/2} z(i+x)u(x) \longleftarrow \text{1D input}$$

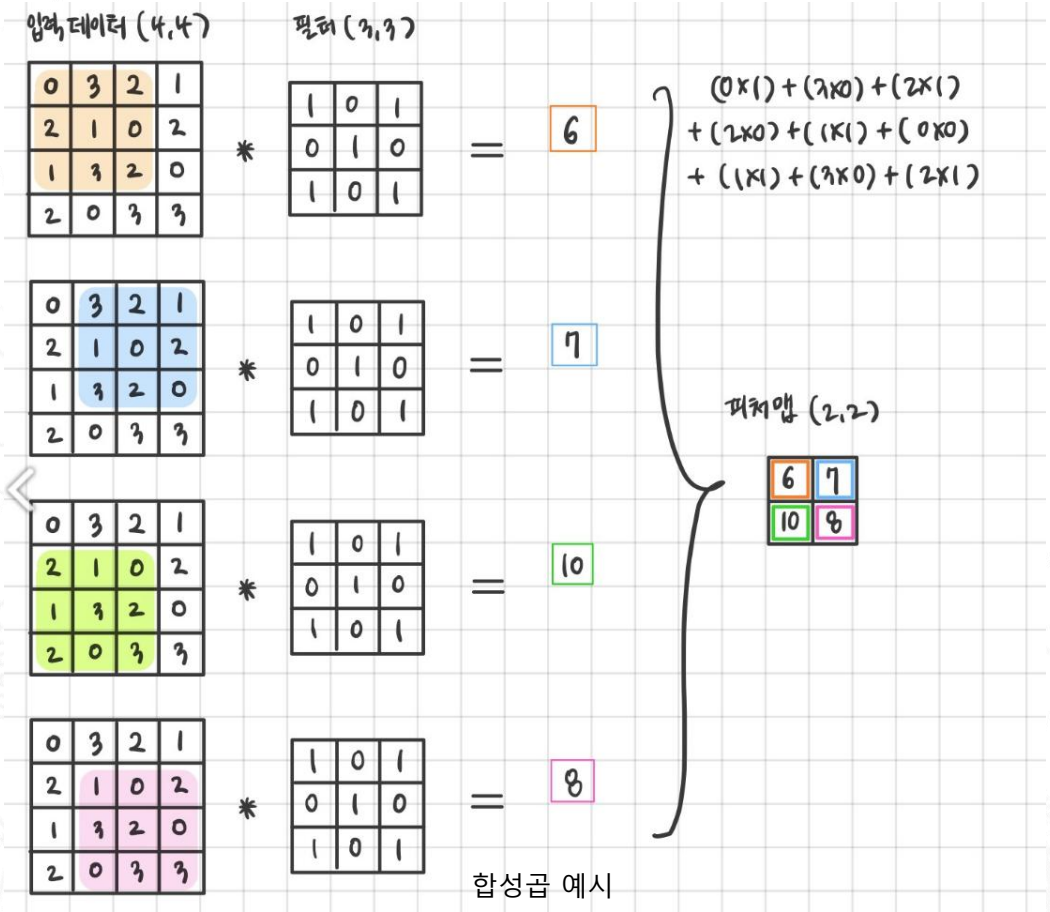
$$s(j,i) = z \circledast u = \sum_{y=-(h-1)/2}^{(h-1)/2} \sum_{x=-(h-1)/2}^{(h-1)/2} z(j+y, i+x)u(y,x) \longleftarrow \text{2D input}$$



(a) 1차원 컨볼루션 (b) 2차원 컨볼루션

Convolution Layer

❖ Characteristics 1: Convolution Kernel



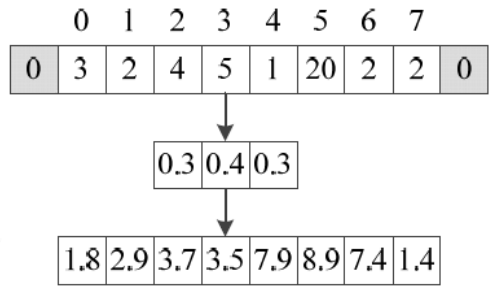
필터의 슬라이딩 순서

출처: https://www.dinolabs.ai/237

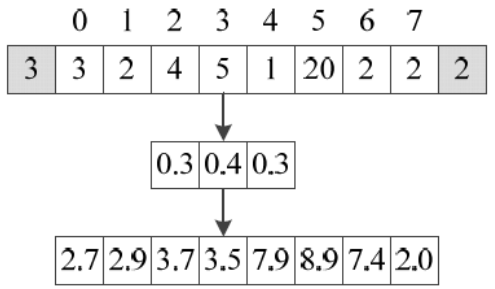
Convolution Layer

❖ Padding

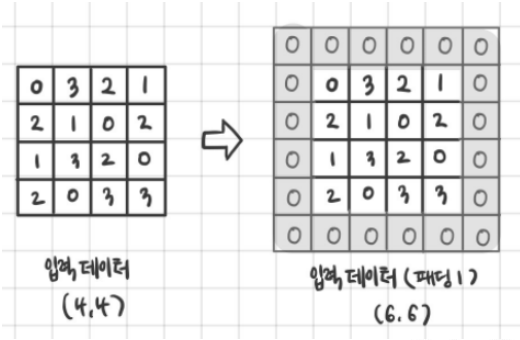
- Avoid reducing input at the edge



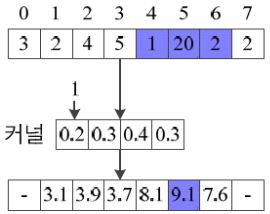
(a) 0 덧대기



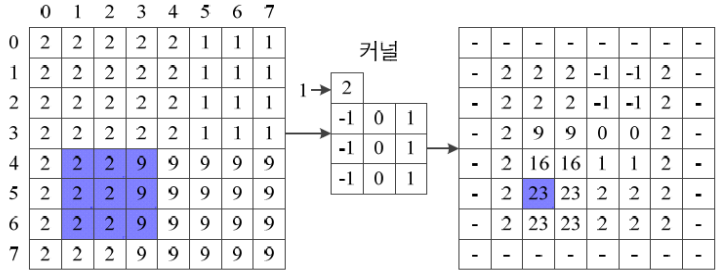
(b) 복사 덧대기



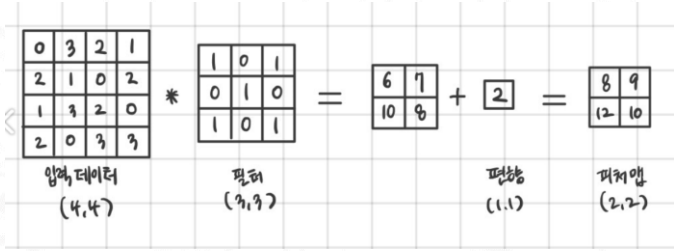
❖ Bias



(a) 1차원 컨볼루션



(b) 2차원 컨볼루션



Convolution Layer

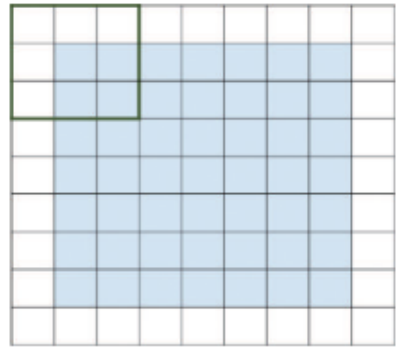
❖ Padding

- Output size of convolutional operations considering padding

$$O=\frac{(N+2\times P)-F}{S}+1$$

N: 입력 데이터 크기, P: 패딩, F: 콘벌루션 필터 크기, S: 스트라이드, O: 출력 데이터 크기

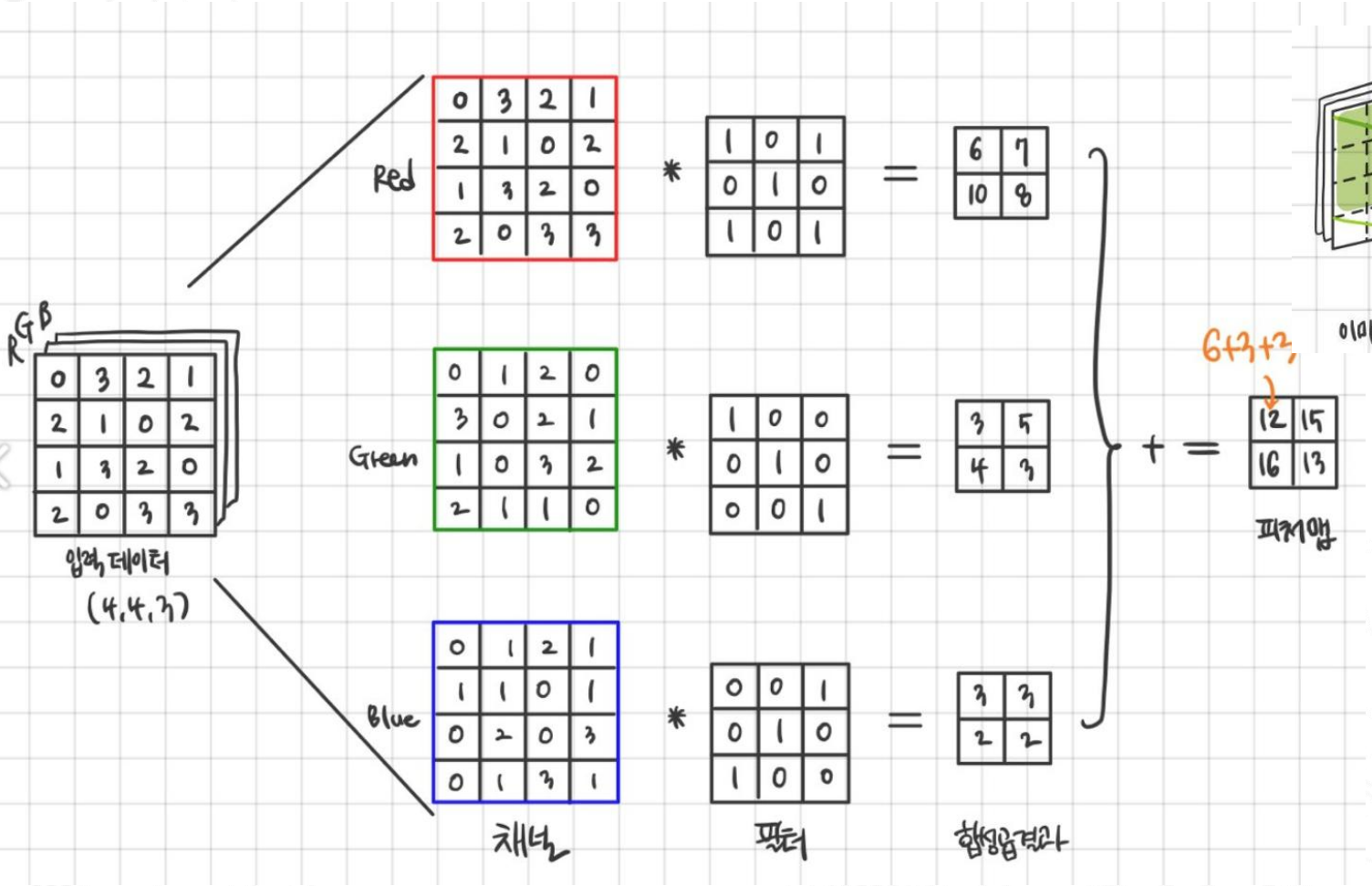
- **7×7** 이미지에 패딩을 해서 **9×9** 이미지로 만들어 보자

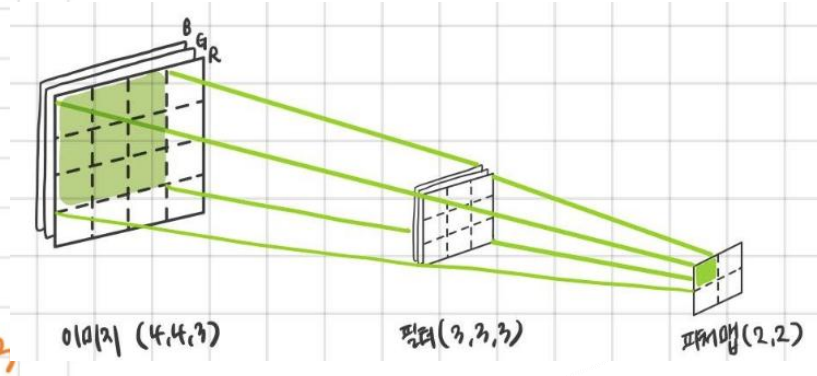


- 앞의 식에 N=7, P=1, F=3, S=1을 대입하면 컨볼루션 연산 후에도 이미지 크기가 7로 유지

$$O=\frac{(N+2\times P)-F}{S}+1=\frac{(7+2\times 1)-3}{1}+1=7$$

Convolution Layer



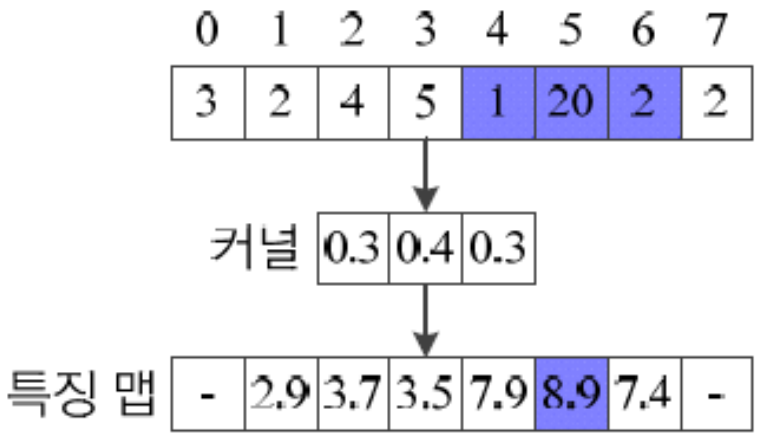


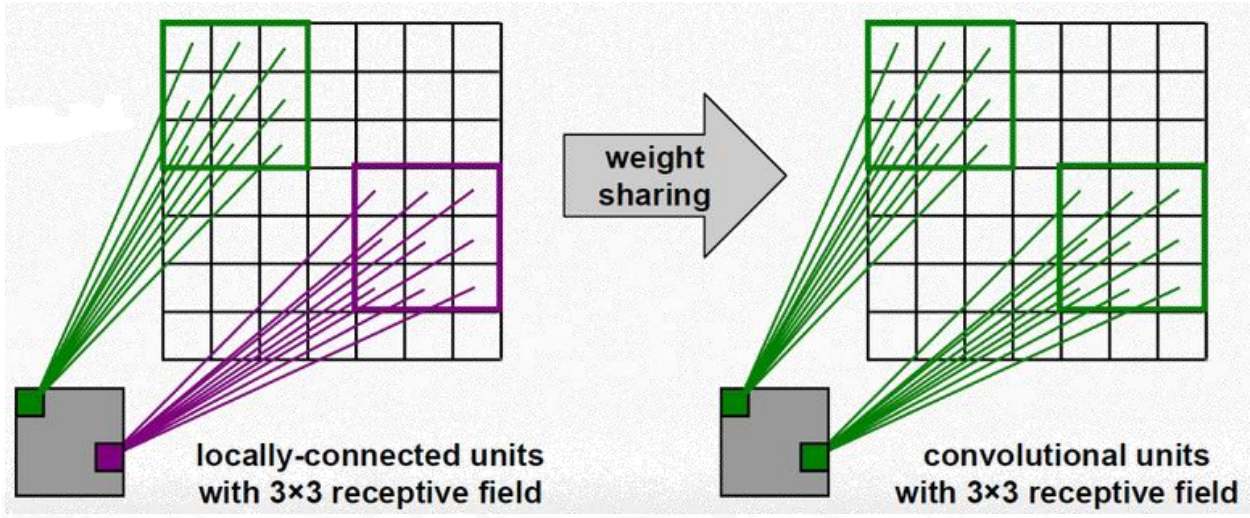
출처: https://www.dinolabs.ai/237

Convolution Layer

❖ Characteristics 2: Weight sharing

- All nodes use the same kernel
- Only three parameters (0.3, 0.4, 0.3) because they share weights
- Significantly reduced model complexity





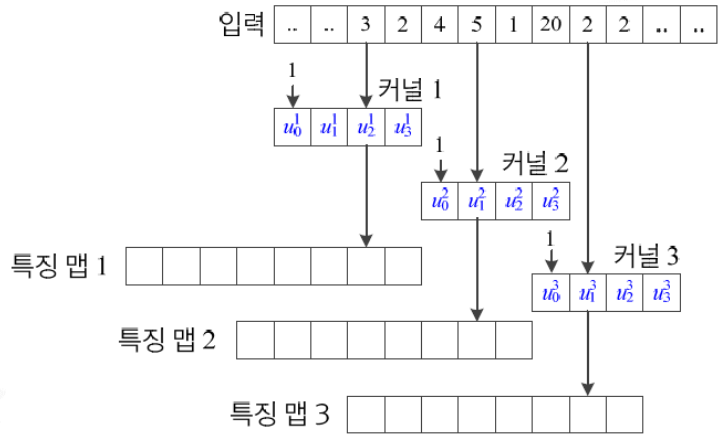
Convolution Layer

❖ Extract multiple feature maps

- Depending on the value of the kernel, the characteristics that the kernel extracts will vary
- Example) Vertical edge, horizontal edge

$$\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

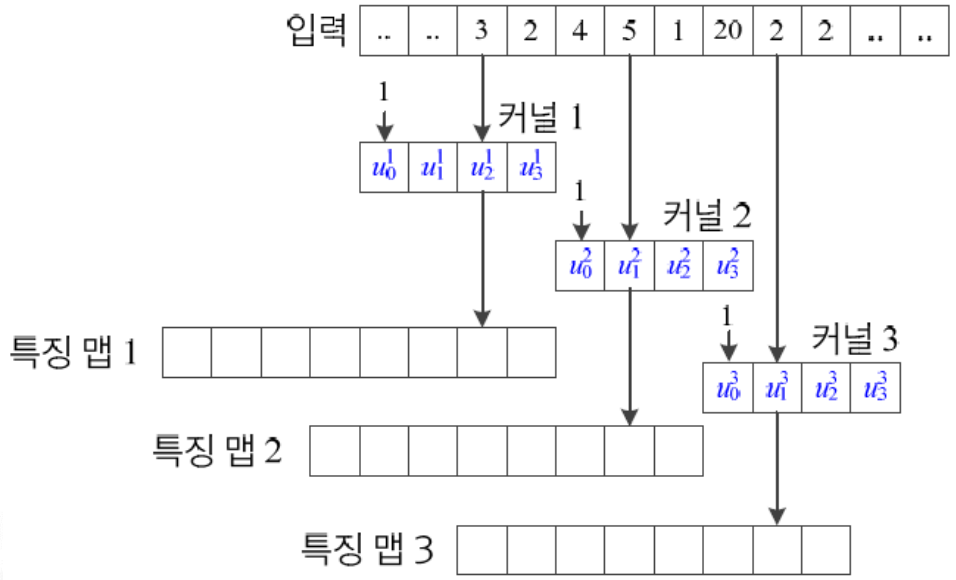
- Therefore, using only one kernel will extract too low-level features
- Figure shows a situation in which three feature maps are extracted using three kernels
- Hundreds of kernels in CNN



Convolution Layer

❖ Feature learning

- Kernel is not designed by humans, it is discovered by learning
  - u_i^k: *i*th parameter of *k*th kernel
  - 예) 2차원 영상이 7*7 커널을 64개 사용한다면, 학습 (7*7+1)*64=3200개의 매개변수를 찾아내야 함



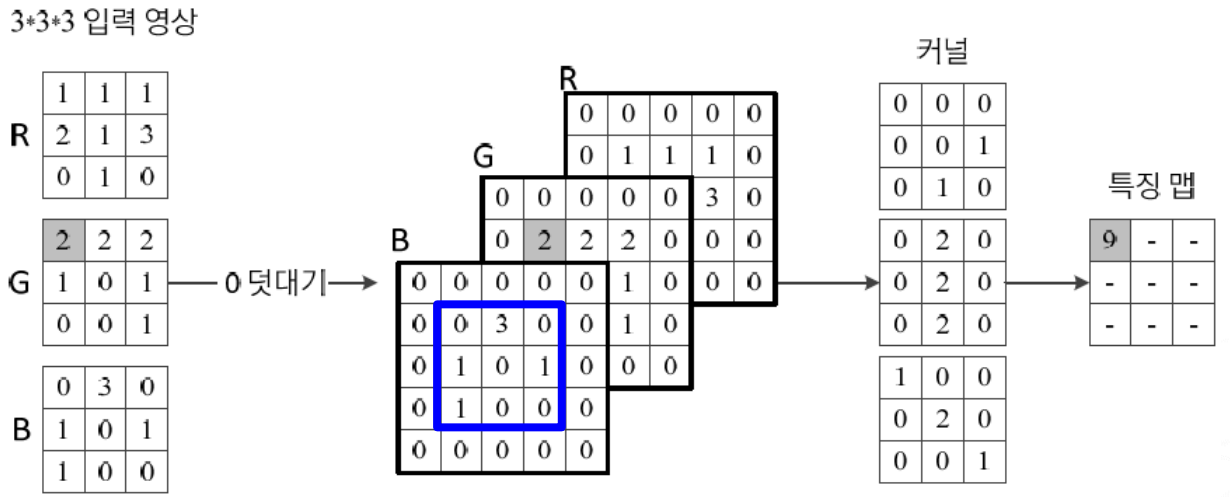
- Training the kernel using backpropagation

Convolution Layer

- Tensor: 벡터 계산을 단순화하기 위해 같은 성질의 여러 벡터를 한 행렬 안에 표기하고 그것을 단순화하여 표기한 것

❖ Apply to Tensor*

- Applicable to 3D or high-dimensional structures
  - e.g., RGB 컬러 영상은 $3*m*n$의 3차원 텐서

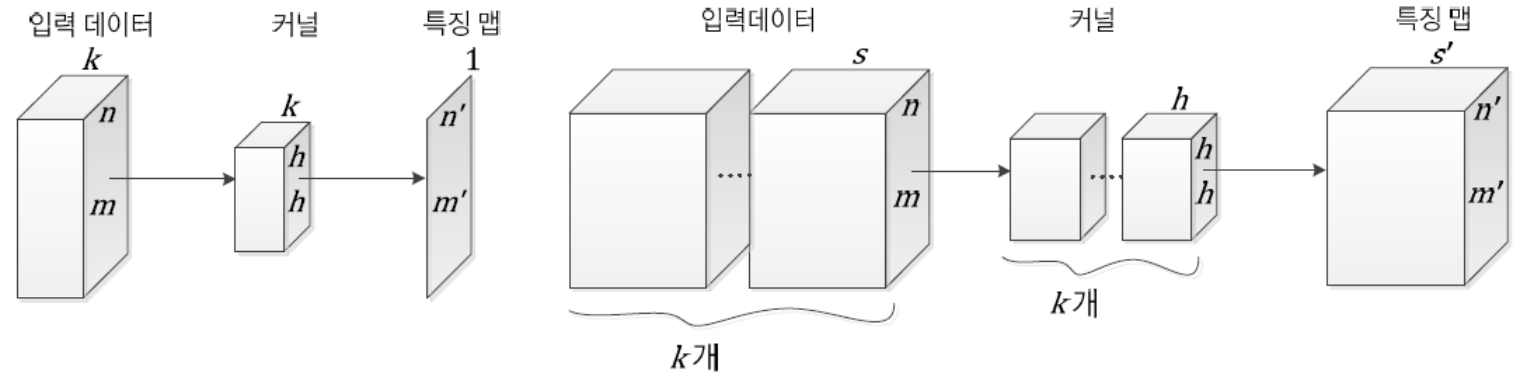


- Example of calculation of gray nodes in feature maps

$$\underbrace{\begin{pmatrix}0&0&0\\0&1&1\\0&2&1\end{pmatrix}}_{R}\underbrace{\begin{pmatrix}0&0&0\\0&2&2\\0&1&0\end{pmatrix}}_{G}\underbrace{\begin{pmatrix}0&0&0\\0&0&3\\0&1&0\end{pmatrix}}_{B}\circledast\underbrace{\begin{pmatrix}0&0&0\\0&0&1\\0&1&0\end{pmatrix}}_{c_1}\underbrace{\begin{pmatrix}0&2&0\\0&2&0\\0&2&0\end{pmatrix}}_{c_2}\underbrace{\begin{pmatrix}1&0&0\\0&2&0\\0&0&1\end{pmatrix}}_{c_3}=9$$

Convolution Layer

❖ Apply to data in 3D structure



(a) 다중채널 데이터(예: RGB 컬러 영상) (b) 3차원 데이터(예: 동영상, MRI 뇌 영상)

- Calculate the size of the input data using the output size of layer

$$O=\frac{N-F}{S}+1 \qquad N=(O-1)\times S+\mathrm{F}$$

N: 입력 데이터 크기, F: 콘벌루션 필터 크기, S: 스트라이드, O: 출력 데이터 크기

Convolution Layer

❖ Example

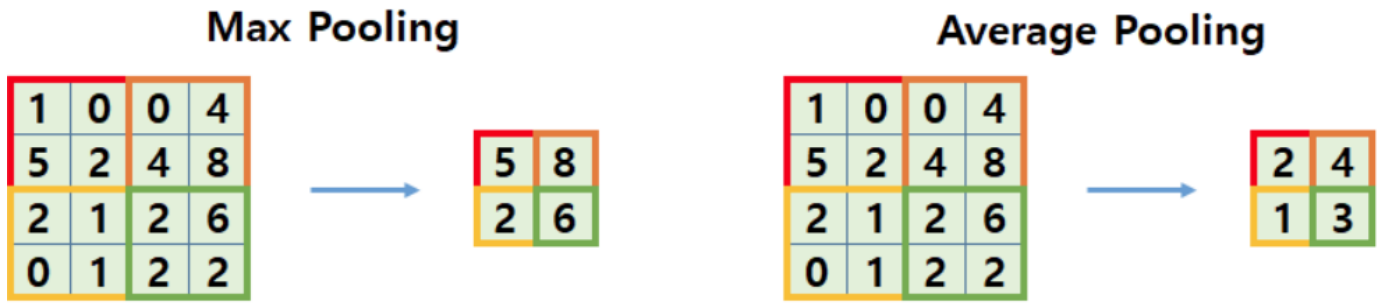
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

CNN-1	**CNN-2**	**CNN-3**
Output FC layer (44.29)	Output FC layer (91.46)	*Output FC layer (92.05)*)
10 × 10 (88.67)	10 × 10 (91.14)	10 × 10 (91.03)
16 × 10 (88.72)	16 × 10 (91.58)	16 × 10 (91.77)
32 × 10 (88.93)	*32 × 10 (91.99)*	32 × 10 (92.02)
64 × 10 (89.72)	64 × 10 (91.82)	64 × 10 (91.8)
128 × 10 (89.2)	128 × 10 (91.86)	128 × 10 (89.2)
256 × 10 (89.23)	*256 × 10 (92.02)*	256 × 10 (89.23)
512 × 10 (88.95)	512 × 10 (90.98)	512 × 10 (91.78)
1024 × 10 (89.56)	1024 × 10 (91.54)	**1024 × 10 (92.22)**
2048 × 10 (87.4)	2048 × 10 (91.27)	2048 × 10 (91.59)
4096 × 10 (86.27)	4096 × 10 (87.51)	4096 × 10 (90.68)
64 × 64 × 10 (89.35)	256 × 256 × 10 (91.97)	1024 × 1024 × 10 (91.27)
128 × 64 × 10 (89.71)	512 × 256 × 10 (91.92)	2048 × 1024 × 10 (91.43)
256 × 64 × 10 (89.79)	1024 × 256 × 10 (91.53)	4096 × 1024 × 10 (91.94)
512 × 64 × 10 (89.88)	2048 × 256 × 10 (91.95)	-
1024 × 64 × 10 (90)	**4096 × 256 × 10 (92.29)**	-
2048 × 64 × 10 (90.28)	4096 × 4096 × 256 × 10 (91.64)	-
4096 × 64 × 10 (90.59)	-	-
4096 × 4096 × 64 × 10 (90.77)	-	-
4096 × 4096 × 4096 × 64 × 10 (90.74)	-	-

Pooling Layer

❖ Characteristics 3: Pooling

- All the data in the output feature map that went through the previous layers is not required
  - Pooling: Max pooling, Avg pooling, Weight avg pooling, and etc.



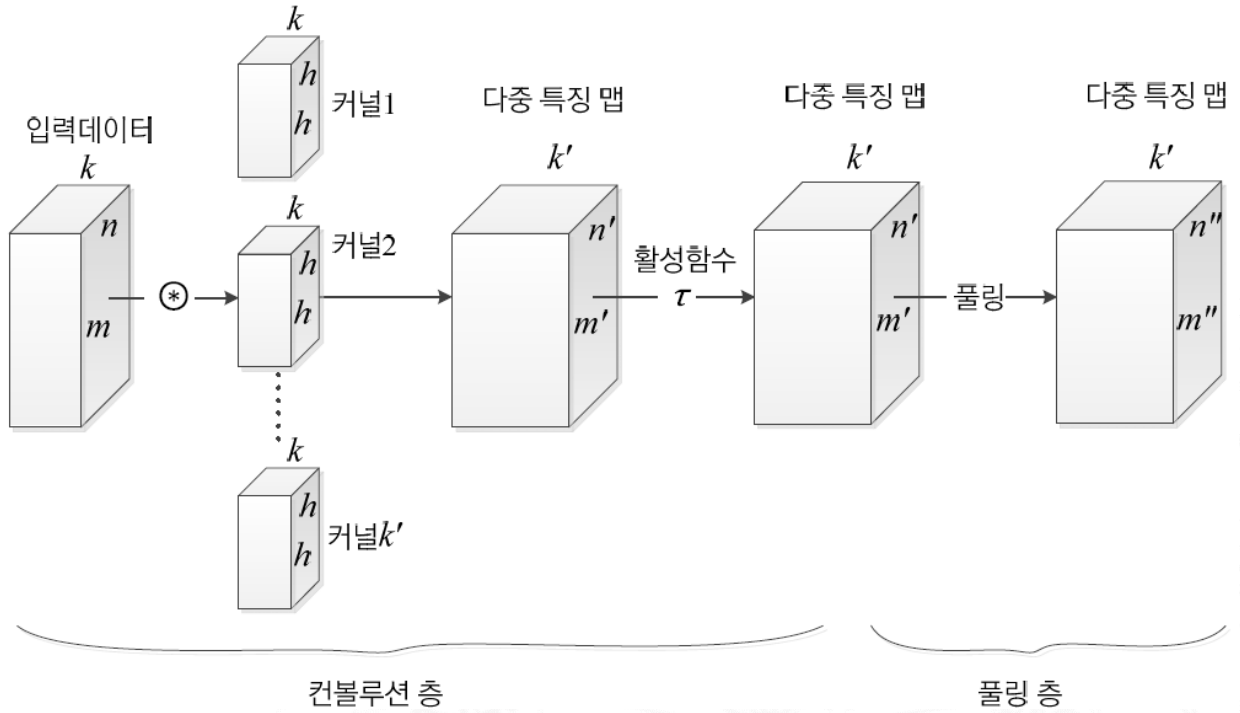
❖ Effects

- Reduce parameters, reduce network expression and suppress overfitting
- Reduced computation saves hardware resources and speeds up

Overall Structure

❖ Building block

- CNN connects building blocks to create deep architecture
- Conv. → activation function → pooling
- Extracts multiple feature maps using multiple kernels



Overall Structure

❖ Configurable parameters

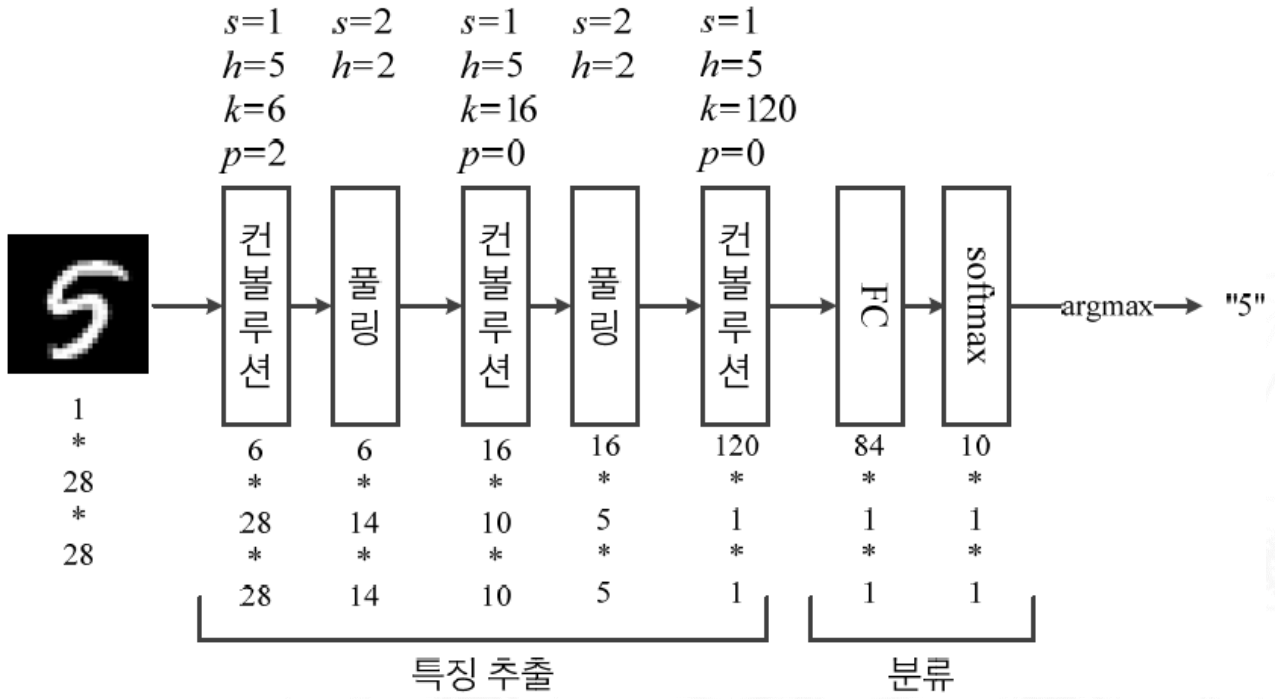
- Convolution filter
- Filter size
- Padding
- Stride
- Pooling layer

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Overall Structure

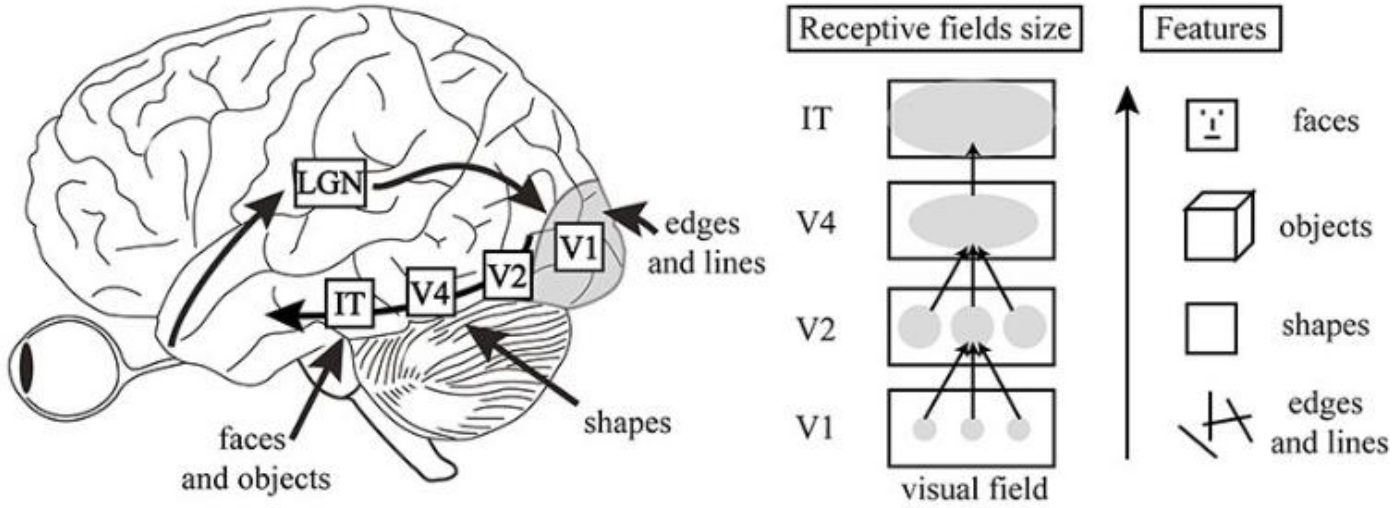
❖ LeNet-5

- Feature extraction: Converting a 28*28 contrast map into a 120-dimensional feature vector through five layers of C-P-C-P-C
- Classification: MLP with one hidden layer



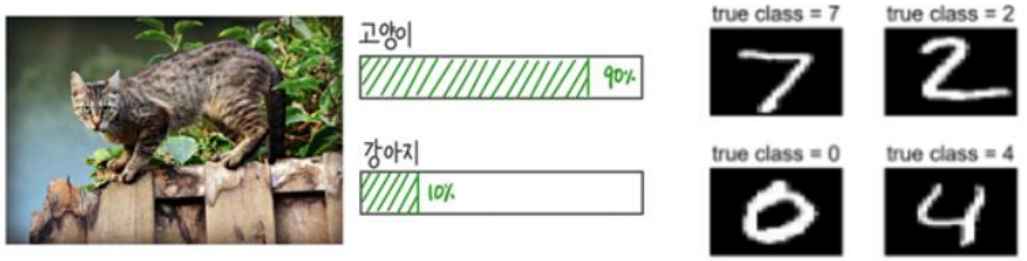
Overall Structure

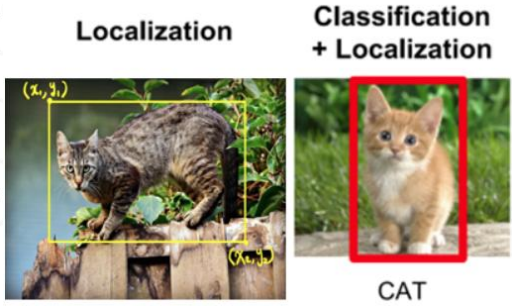
- DMLP vs. CNN
  - DMLP cannot be calculated if the feature vector is different in size
  - CNN has the advantage of handling variable sizes
    - Adjust the size of the feature map by adjusting the stride in the convolution layer or by adjusting the kernel or stride in the pooling layer

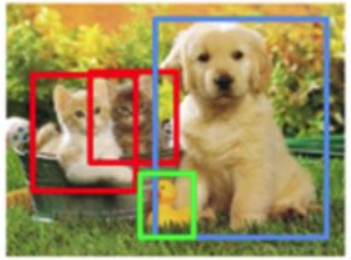


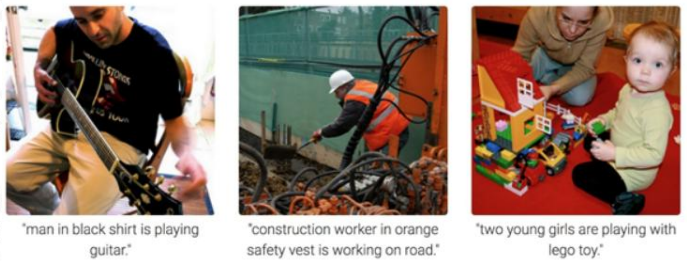
Application

- ❖ Image classification
- ❖ Object localization
- ❖ Object detection
- ❖ Image captioning





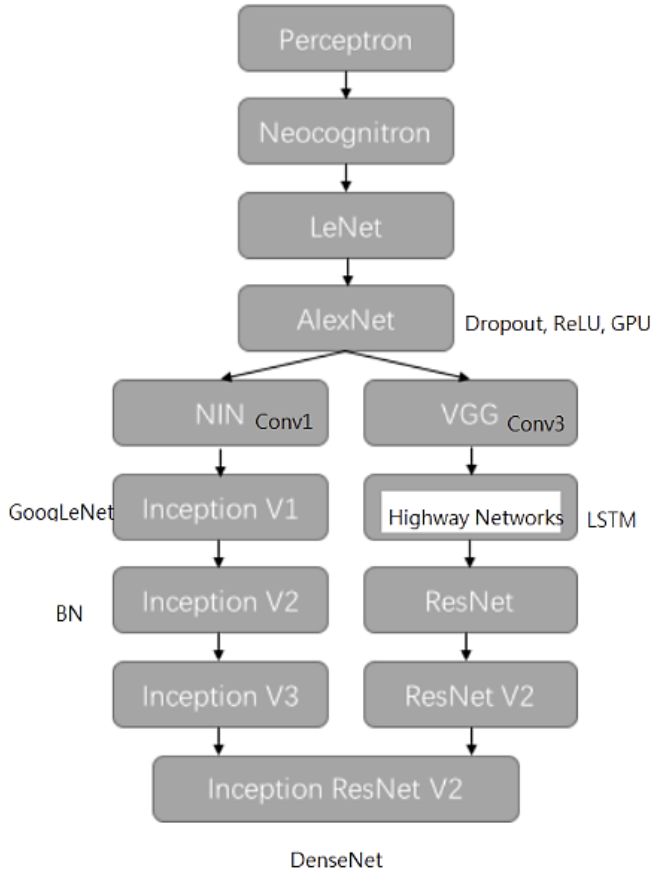




출처: https://www.dinolabs.ai/237

Representative CNN

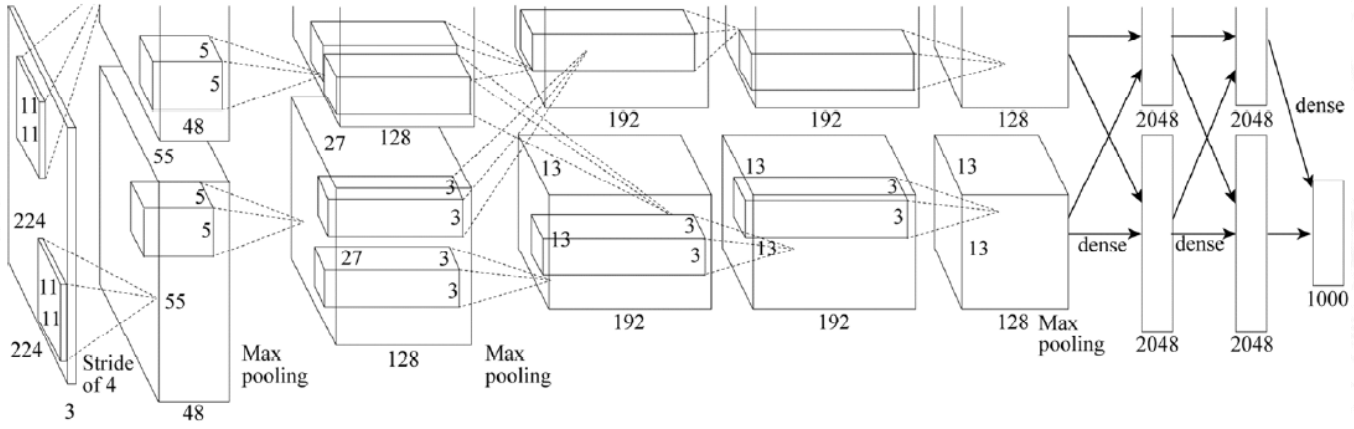
- AlexNet
- VGGNet
- GoogLeNet
- ResNet



AlexNet [Krizhevsky et al., 2012]

❖ Structure

- 5 convolutional layers and 3 fully connected (FC) layers
  - 8 layer, 290400-186624-64896-43264-4096-4096-1000 nodes
- 2 million parameters for convolution layer and 65 million parameters for FC layer
- 30x more parameters for FC layer → In the future, CNN will advanced toward reducing the parameters of FC layer



AlexNet

- What made AlexNet successful in learning?
  - External factors
    - A large database called ImageNet
    - Parallel processing using GPUs
  - Internal factors
    - Use ReLU as active function
    - Applying local response normalization techniques
    - Apply multiple regulatory techniques to prevent overfitting
      - Data augmentation (2048x growth with truncation and reversal)
      - Dropout, etc.
  - Ensemble application during test phase
  - 2 ~ 3% reduction in error rates

AlexNet

❖ The number of parameters

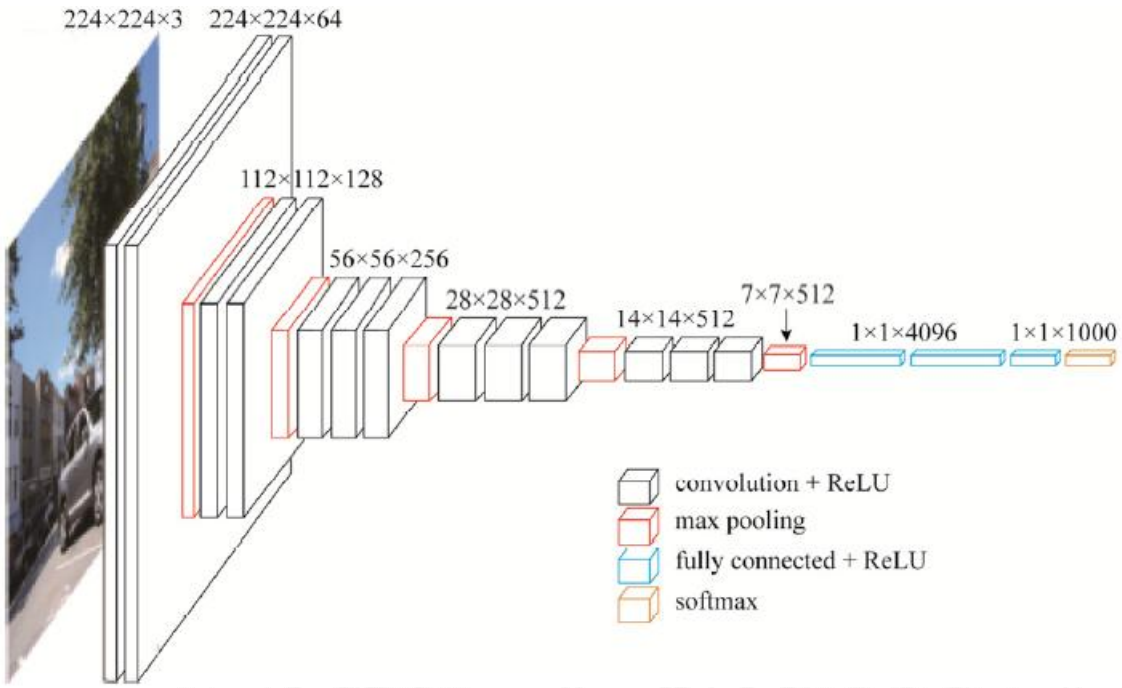
계층	텐서 크기	가중치	편향	파라미터
입력 이미지	227×227×3	0	0	0
CONV1	55×55×96	34,848	96	34,944
MaxPool-1	27×27×96	0	0	0
CONV2	27×27×256	614,400	256	614,656

계층	텐서 크기	가중치	편향	파라미터
MaxPool-2	13×13×256	0	0	0
CONV3	13×13×384	884,736	384	885,120
CONV4	13×13×384	1,327,104	384	1,327,488
CONV5	13×13×256	884,736	256	884,992
MaxPool-3	6×6×256	0	0	0
FC1	4096×1	37,748,736	4,096	37,752,832
FC2	4096×1	16,777,216	4,096	16,781,312
FC3	1000×1	4,096,000	1,000	4,097,000
출력	1000×1	0	0	0
전체				62,378,344

VGGNet [Simonyan et al., 2015]

❖ Structure

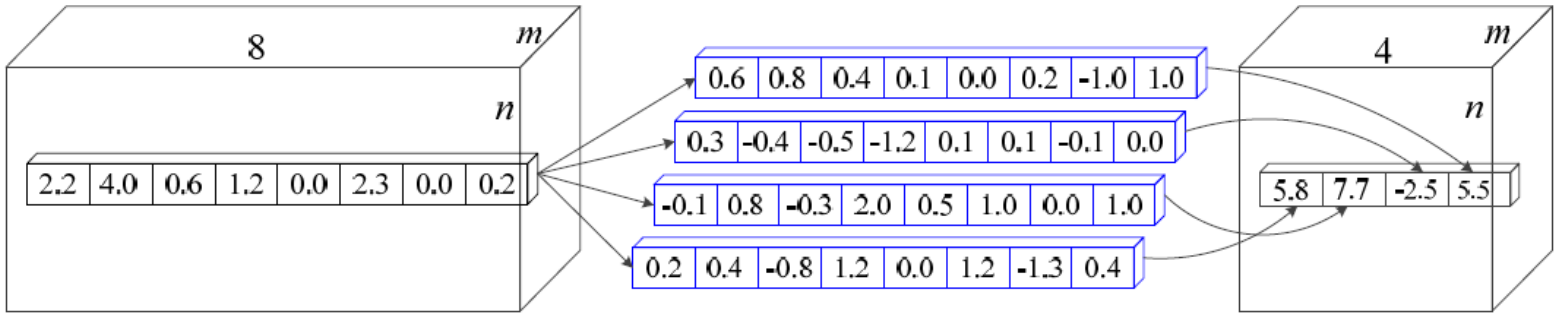
- Deepen neural networks using a small 3*3 kernel
- 8 to 16 convolution layers for 2 to 3 times deeper than 5 on AlexNet
- 16 blocks for VGG-16(13 Conv. layer + 3 FC layer)



VGGNet

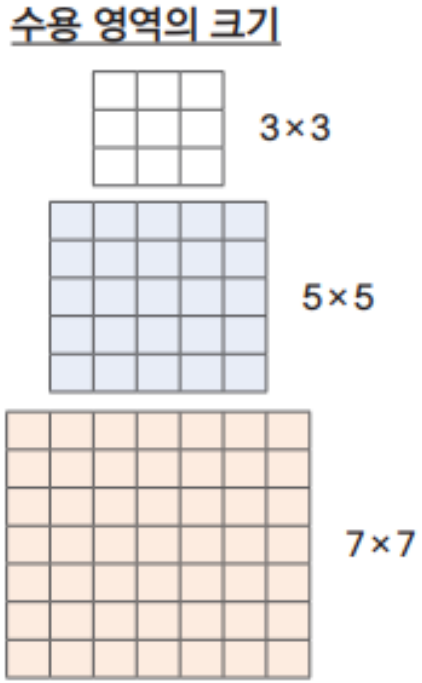
❖ 1*1 kernel

- Dimension reduction
  - Apply four 1*1 kernels to 8 feature maps of m*n → 4 feature maps of m*n
  - In other words, four*m*n tensors are output by applying four 8*1*1*1 kernels to 8*m*n tensors
- Applying nonlinear active functions such as ReLU increases the discernment of feature maps



VGGNet

❖ Comparison with large and small convolution filters



	필터 1개 사용	여러 계층에 3×3 필터 사용
	3×3 필터	3×3 필터×1개 계층
파라미터 수:	9개	9×1 = 9개
	5×5 필터	3×3 필터×2개 계층
파라미터 수:	25개	9×2 = 18개
	7×7 필터	3×3 필터×3개 계층
파라미터 수:	49개	9×3 = 27개

VGGNet

❖ Specification

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

VGGNet

❖ The number of parameters

구분 / 계층	계층 이름	출력 크기	액티베이션 맵	파라미터 수
	입력 이미지	224×224×3	224×224×3=150K	0
Conv1	Conv1-1	224×224×64	224×224×64=3.2M	(3×3×3)×64 = 1,728
	Conv1-2	224×224×64	224×224×64=3.2M	(3×3×64)×64 = 36,864
	MaxPool-1	112×112×64	112×112×64=800K	0
Conv2	Conv2-1	112×112×128	112×112×128=1.6M	(3×3×64)×128 = 73,728
	Conv2-2	112×112×128	112×112×128=1.6M	(3×3×128)×128 = 147,456
	MaxPool-2	56×56×128	56×56×128=400K	0
Conv3	Conv3-1	56×56×256	56×56×256=800K	(3×3×128)×256 = 294,912
	Conv3-2	56×56×256	56×56×256=800K	(3×3×256)×256 = 589,824
	Conv3-3	56×56×256	56×56×256=800K	(3×3×256)×256 = 589,824
	MaxPool-3	28×28×256	28×28×256=200K	0
Conv4	Conv4-1	28×28×512	28×28×512=400K	(3×3×256)×512 = 1,179,648
	Conv4-2	28×28×512	28×28×512=400K	(3×3×512)×512 = 2,359,296
	Conv4-3	28×28×512	28×28×512=400K	(3×3×512)×512 = 2,359,296
	MaxPool-4	14×14×512	14×14×512=100K	
Conv5	Conv5-1	14×14×512	4×14×512=100K	(3×3×512)×512 = 2,359,296
	Conv5-2	14×14×512	4×14×512=100K	(3×3×512)×512 = 2,359,296
	Conv5-3	14×14×512	4×14×512=100K	(3×3×512)×512 = 2,359,296
	MaxPool-5	7×7×512	7×7×512=25K	
FC	FC-1	4096	4096	7×7×512×4096 = 102,760,448
	FC-2	4096	4096	4096×4096 = 16,777,216
	FC-3	1000	1000	4096×1000 = 4,096,000
	출력	1000	0	
	전체		24M × 4 bytes ~= 96MB	138MB

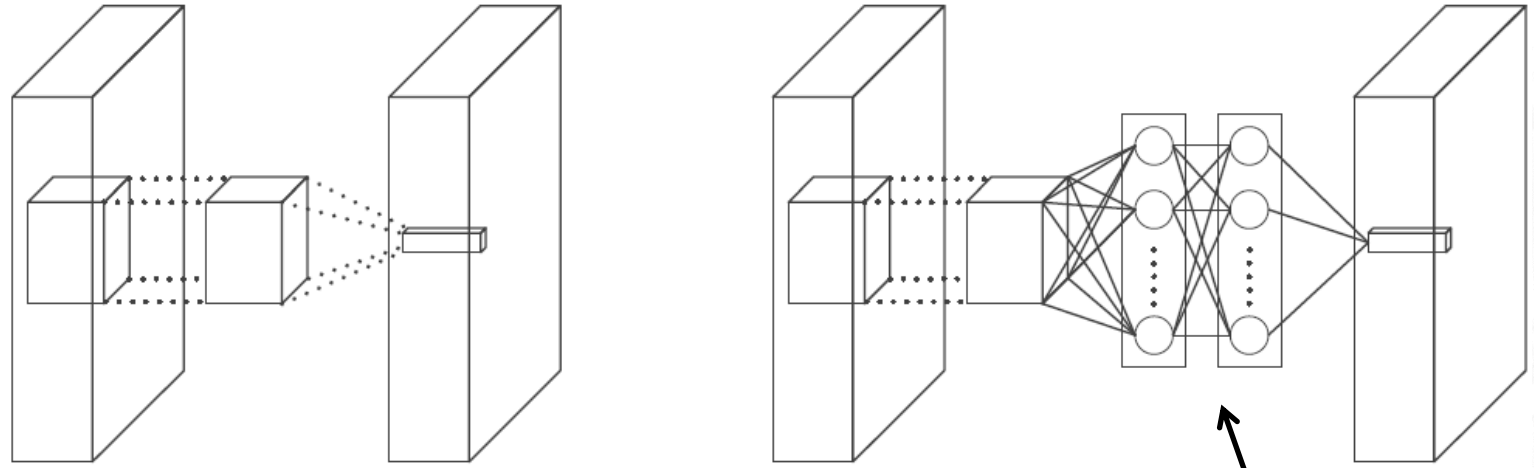
GoogLeNet [Szegedy et al., 2015]

❖ Structure

- Based on NIN(Network in Network)

❖ NIN structure

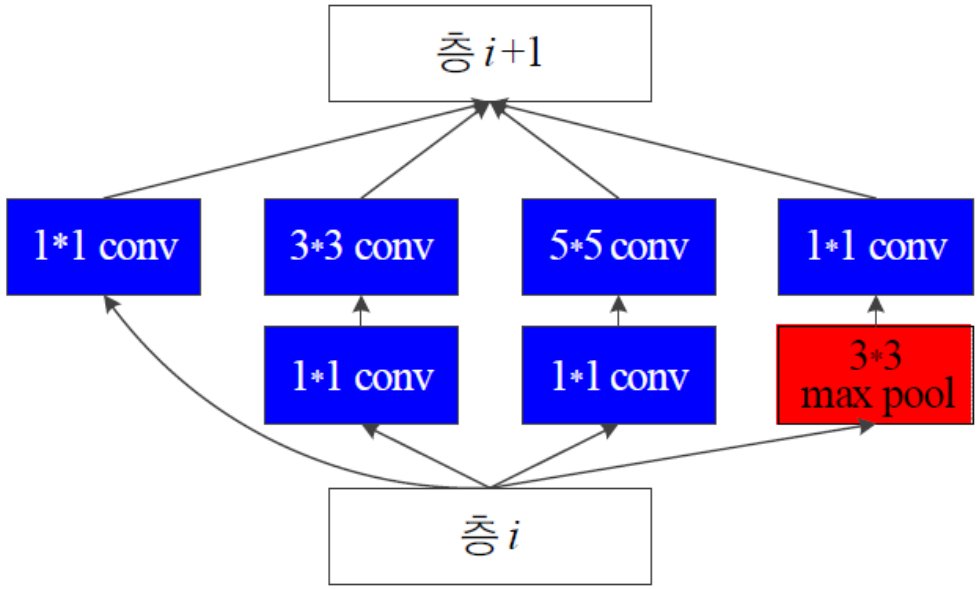
- MLPconv layer replaces convolution operation
- MLPconv performs forward computation of MLP by moving the kernel

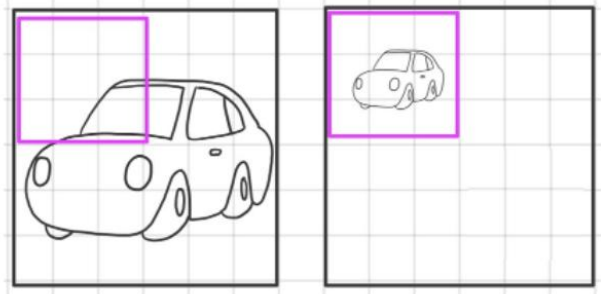


GoogLeNet

❖ GoogLeNet is a neural network that extends NIN ideas

- Inception module
  - Micronetwork uses four kinds of convolution operations instead of MLPconv
  - Use 1*1 convolution to reduce dimensions



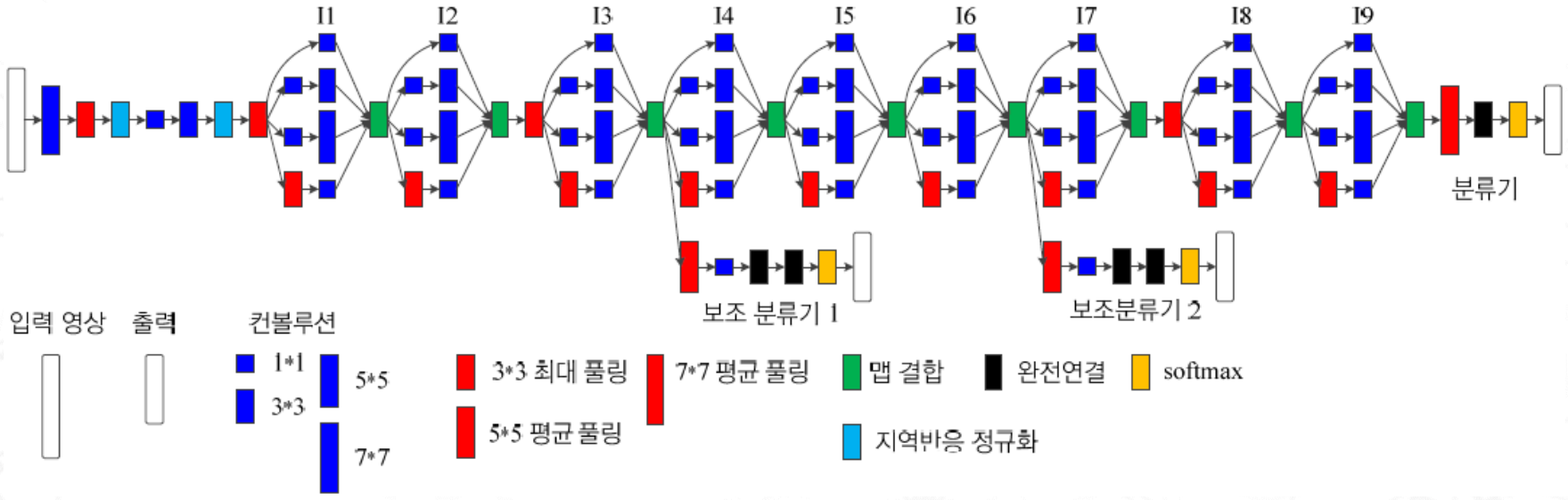


https://ardino.tistory.com/44

GoogLeNet

❖ GoogLeNet combines 9 inception modules

- 22 layers with parameters and 5 layers without parameters
- 1 FC layer



GoogLeNet

❖ Specification

type	필터 크기/ 스트라이드	출력 크기	깊이	1×1 필터수	3×3 병목계층 필터수	3×3 필터수	5×5 병목계층 필터수	5×5 필터수	맥스 풀링 병목계층 필터수	파라미터 수	연산 횟수
콘벌루션	7×7/2	112×112×64	1							2.7k	34M
맥스 풀링	3×3/2	56×56×64	0								
콘벌루션	3×3/1	56×56×192	2		64	192				112k	360M
맥스 풀링	3×3/2	28×28×192	0								
인셉션(3a)		28×28×256	2	64	96	128	16	32	32	159k	128M
인셉션(3b)		28×28×480	2	128	128	192	32	96	64	380k	304M
맥스 풀링	3×3/2	14×14×480	0								
인셉션(4a)		14×14×512	2	192	96	208	16	48	64	364k	73M
인셉션(4b)		14×14×512	2	160	112	224	24	64	64	437k	88M
인셉션(4c)		14×14×512	2	128	128	256	24	64	64	463k	100M
인셉션(4d)		14×14×528	2	112	144	288	32	64	64	580k	119M
인셉션(4e)		14×14×832	2	256	160	320	32	128	128	840k	170M
맥스 풀링	3×3/2	7×7×832	0								
인셉션(5a)		7×7×832	2	256	160	320	32	128	128	1072k	54M
인셉션(5b)		7×7×1024	2	384	192	384	48	128	128	1388k	71M
평균 풀링	7×7/1	1×1×1024	0								
드롭 아웃(40%)		1×1×1024	0								
완전 연결		1×1×1000	1							1000k	1M
소프트맥스		1×1×1000	0								

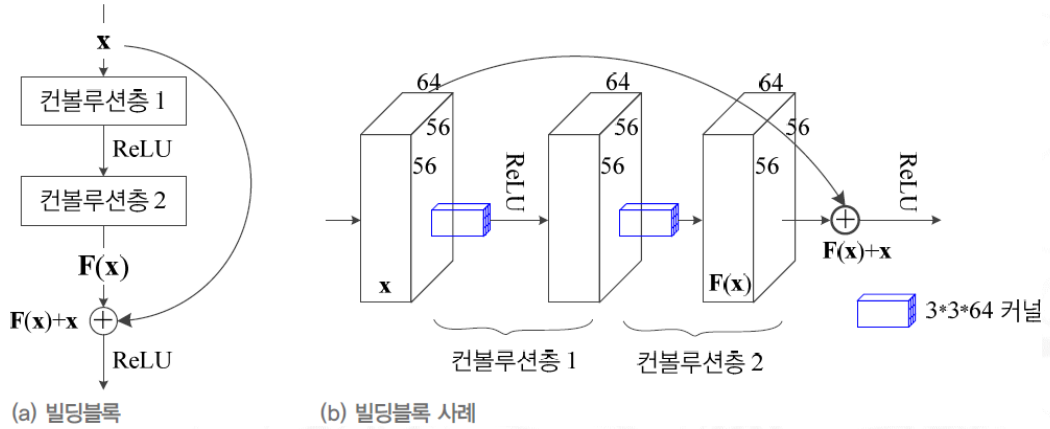
❖ ResNet

- Using the idea of residual learning to significantly increase the number of floors while avoiding performance degradation
- Basic CNN

$$\mathbf{F}(\mathbf{x}) = \boldsymbol{\tau}(\mathbf{x} \circledast \mathbf{w}_1) \circledast \mathbf{w}_2$$
$$\mathbf{y} = \boldsymbol{\tau}(\mathbf{F}(\mathbf{x}))$$

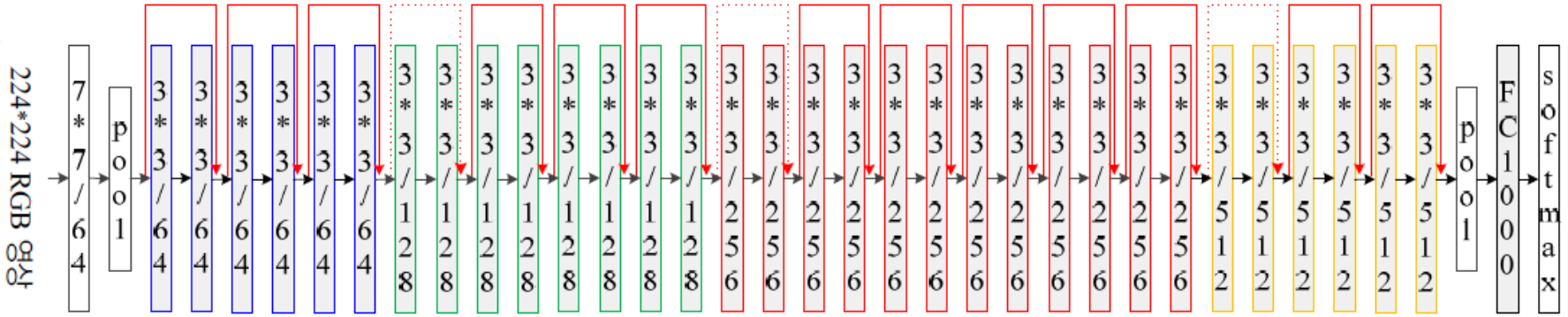
- Residual learning applies $\boldsymbol{\tau}$ to $\mathbf{x}(\mathbf{x})+\mathbf{F}$ with shortcut connected $\mathbf{x}$ $\mathbf{F}(\mathbf{x})$ remains

$$\mathbf{y} = \boldsymbol{\tau}(\mathbf{F}(\mathbf{x}) + \mathbf{x})$$



ResNet

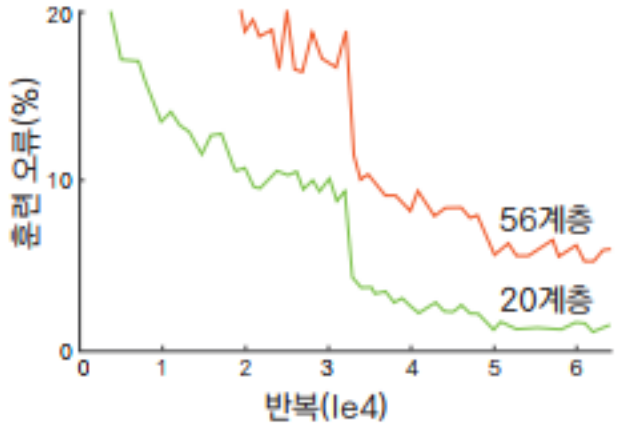
- Example architecture

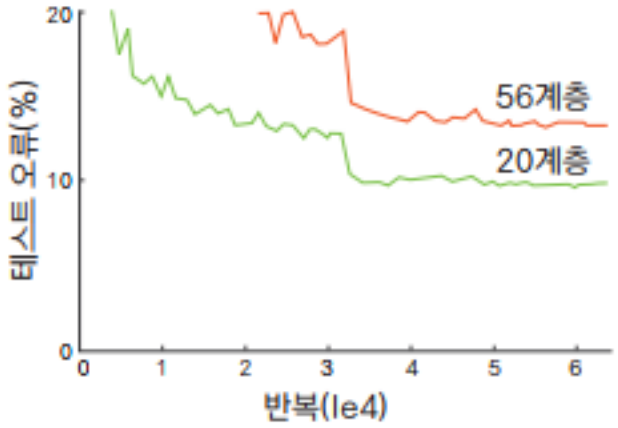


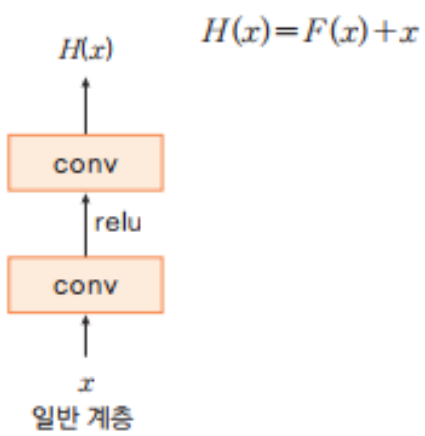
- Same as VGGNet
  - Using 3*3 kernel
- Difference as VGGNet
  - Residual learning
  - Using global average pooling (remove FC layer)
  - Apply to batch normalization

ResNet

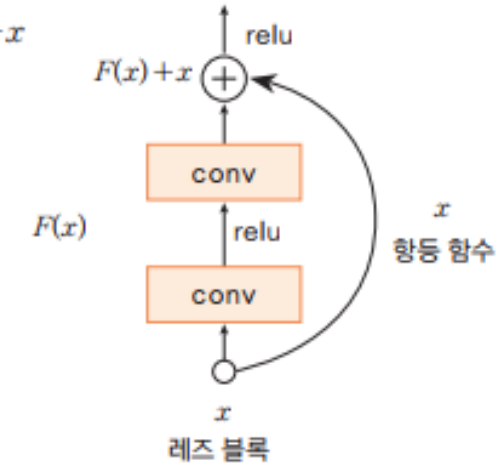
❖ Architecture Design







$H(x)=F(x)+x$



ResNet

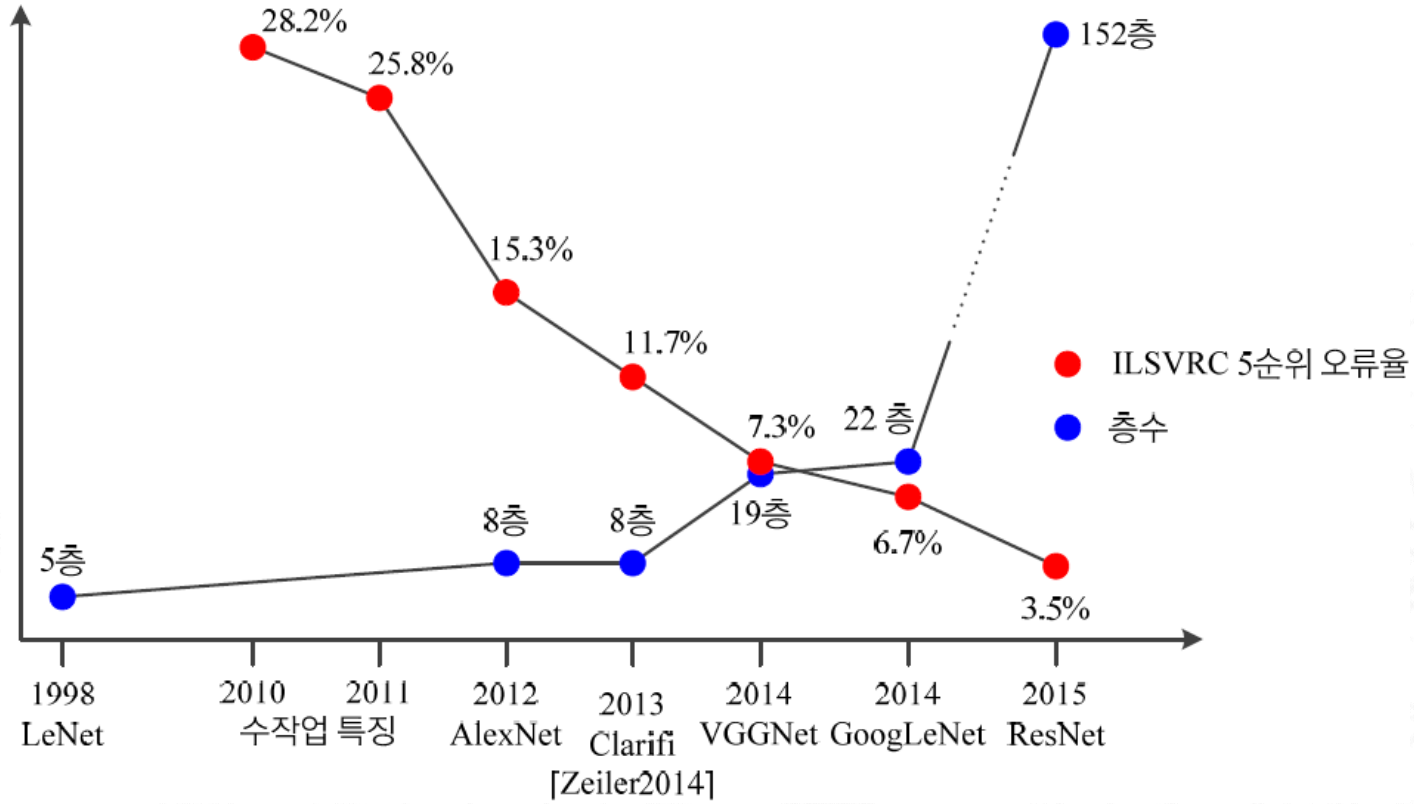
❖ Specification

계층 이름	출력 크기	18계층	34계층	50계층	101계층	152계층
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

ResNet

❖ Grade in ILSVRC competition

- 2012: AlexNet (15.3% error rate)
- 2015: ResNet (3.5% error rate)



Applications

❖ Restoring colors to black-and-white pictures and images

Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification

Satoshi Iizuka* Edgar Simo-Serra* Hiroshi Ishikawa (*equal contribution)

SIGGRAPH 2016

Colorado National Park, 1941 Textile Mill, June 1937 Berry Field, June 1909 Hamilton, 1936

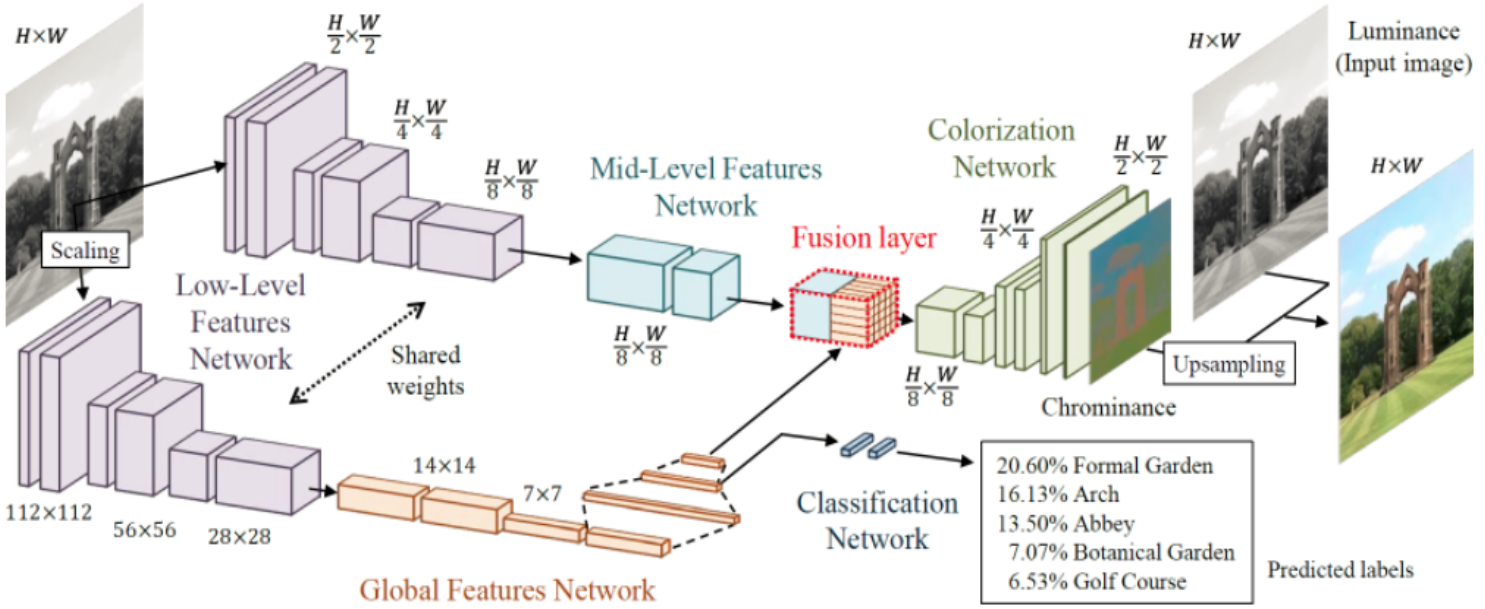
https://www.youtube.com/watch?v=ys5nMO4Q0iY&t=3s

Applications

❖ Restoring colors to black-and-white pictures and images

Colorization Architecture:

Our model consists of four main components: a low-level features network, a mid-level features network, a global features network, and a colorization network. The components are all tightly coupled and trained in an end-to-end fashion. The output of our model is the chrominance of the image which is fused with the luminance to form the output image.



Applications

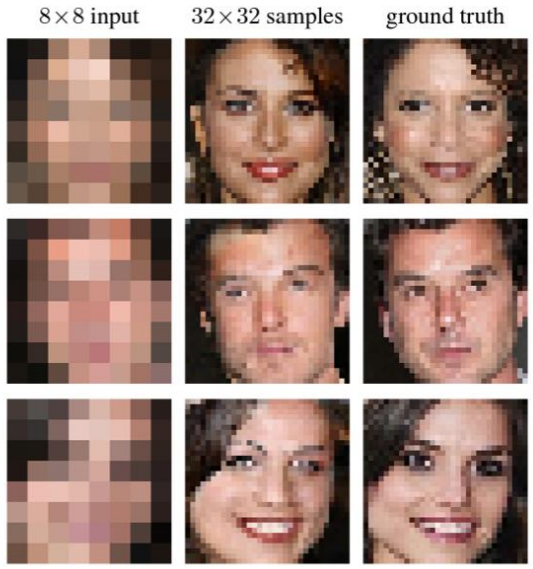
- Restore pixels

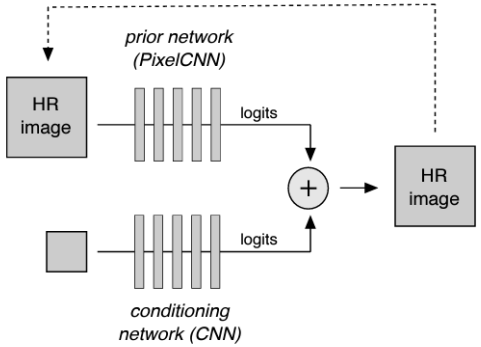
Pixel Recursive Super Resolution

Ryan Dahl * Mohammad Norouzi Jonathon Shlens
Google Brain
{rld,mnorouzi,shlens}@google.com

Abstract

Super resolution is the problem of artificially enlarging a low resolution photograph to recover a plausible high resolution version. In the regime of high magnification factors, the problem is dramatically underspecified and many plausible, high resolution images may match a given low resolution image. In particular, traditional super resolution techniques fail in this regime due to the multimodality of the problem and strong prior information that must be imposed on image synthesis to produce plausible high resolution images. In this work we propose a new probabilistic deep network architecture, a pixel recursive super resolution model, that is an extension of PixelCNNs to address this problem. We demonstrate that this model produces a diversity of plausible high resolution images at large magnification factors. Furthermore, in human evaluation studies we demonstrate how previous methods fail to fool human observers. However, high resolution images sampled from this probabilistic deep network do fool a naive human observer a significant fraction of the time.





https://www.youtube.com/watch?time_continue=56&v=3uoM5kfZIQ0&feature=emb_logo

Applications

- Estimating the movement of multiple people in real time

Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields *

Zhe Cao Tomas Simon Shih-En Wei Yaser Sheikh
The Robotics Institute, Carnegie Mellon University
{zhecao,shihenw}@cmu.edu {tsimon,yaser}@cs.cmu.edu

Abstract

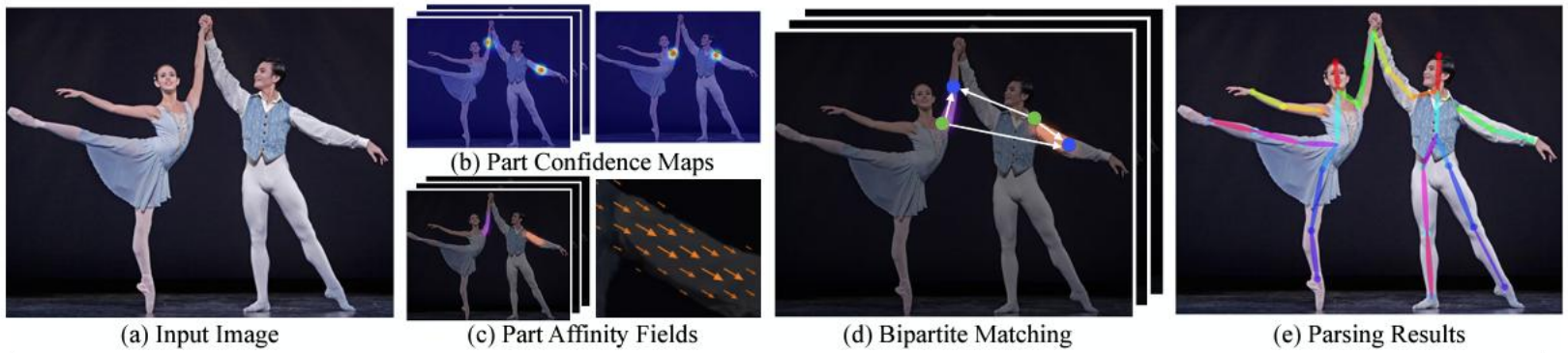
We present an approach to efficiently detect the 2D pose of multiple people in an image. The approach uses a non-parametric representation, which we refer to as Part Affinity Fields (PAFs), to learn to associate body parts with individuals in the image. The architecture encodes global context, allowing a greedy bottom-up parsing step that maintains high accuracy while achieving realtime performance, irrespective of the number of people in the image. The architecture is designed to jointly learn part locations and their association via two branches of the same sequential prediction process. Our method placed first in the inaugural COCO 2016 keypoints challenge, and significantly exceeds the previous state-of-the-art result on the MPII Multi-Person benchmark, both in performance and efficiency.

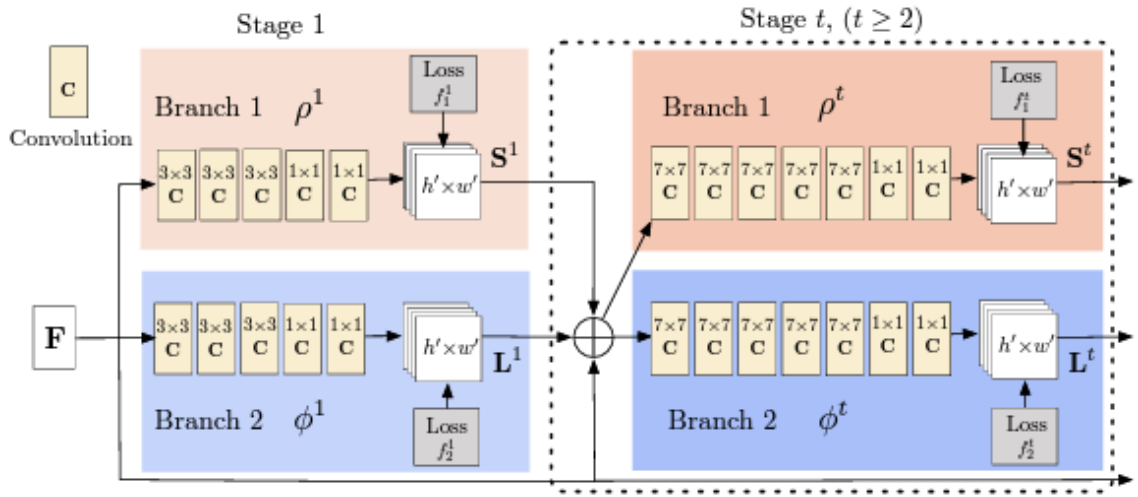
Figure 1. **Top:** Multi-person pose estimation. Body parts belonging to the same person are linked. **Bottom left:** Part Affinity Fields (PAFs) corresponding to the limb connecting right elbow and right wrist. The color encodes orientation. **Bottom right:** A zoomed in view of the predicted PAFs. At each pixel in the field, a 2D vector encodes the position and orientation of the limbs.

https://www.youtube.com/watch?v=pW6nZXeWlGM

Applications

❖ Estimating the movement of multiple people in real time





https://www.youtube.com/watch?v=pW6nZXeWlGM

Applications

❖ Describe a picture

Deep Visual-Semantic Alignments for Generating Image Descriptions

Andrej Karpathy Li Fei-Fei
Department of Computer Science, Stanford University
{karpathy,feifeili}@cs.stanford.edu

Abstract

We present a model that generates natural language descriptions of images and their regions. Our approach leverages datasets of images and their sentence descriptions to learn about the inter-modal correspondences between language and visual data. Our alignment model is based on a novel combination of Convolutional Neural Networks over image regions, bidirectional Recurrent Neural Networks over sentences, and a structured objective that aligns the two modalities through a multimodal embedding. We then describe a Multimodal Recurrent Neural Network architecture that uses the inferred alignments to learn to generate novel descriptions of image regions. We demonstrate that our alignment model produces state of the art results in retrieval experiments on Flickr8K, Flickr30K and MSCOCO datasets. We then show that the generated descriptions significantly outperform retrieval baselines on both full images and on a new dataset of region-level annotations.

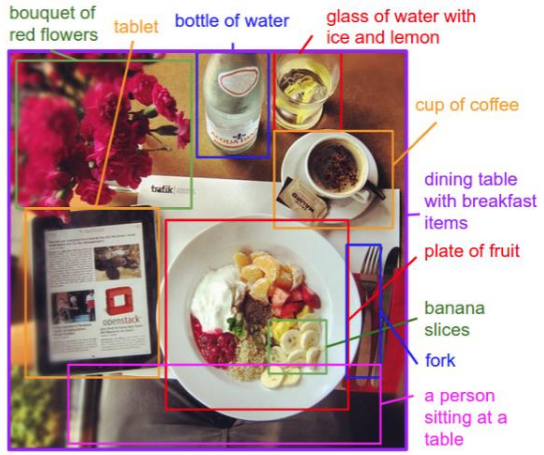
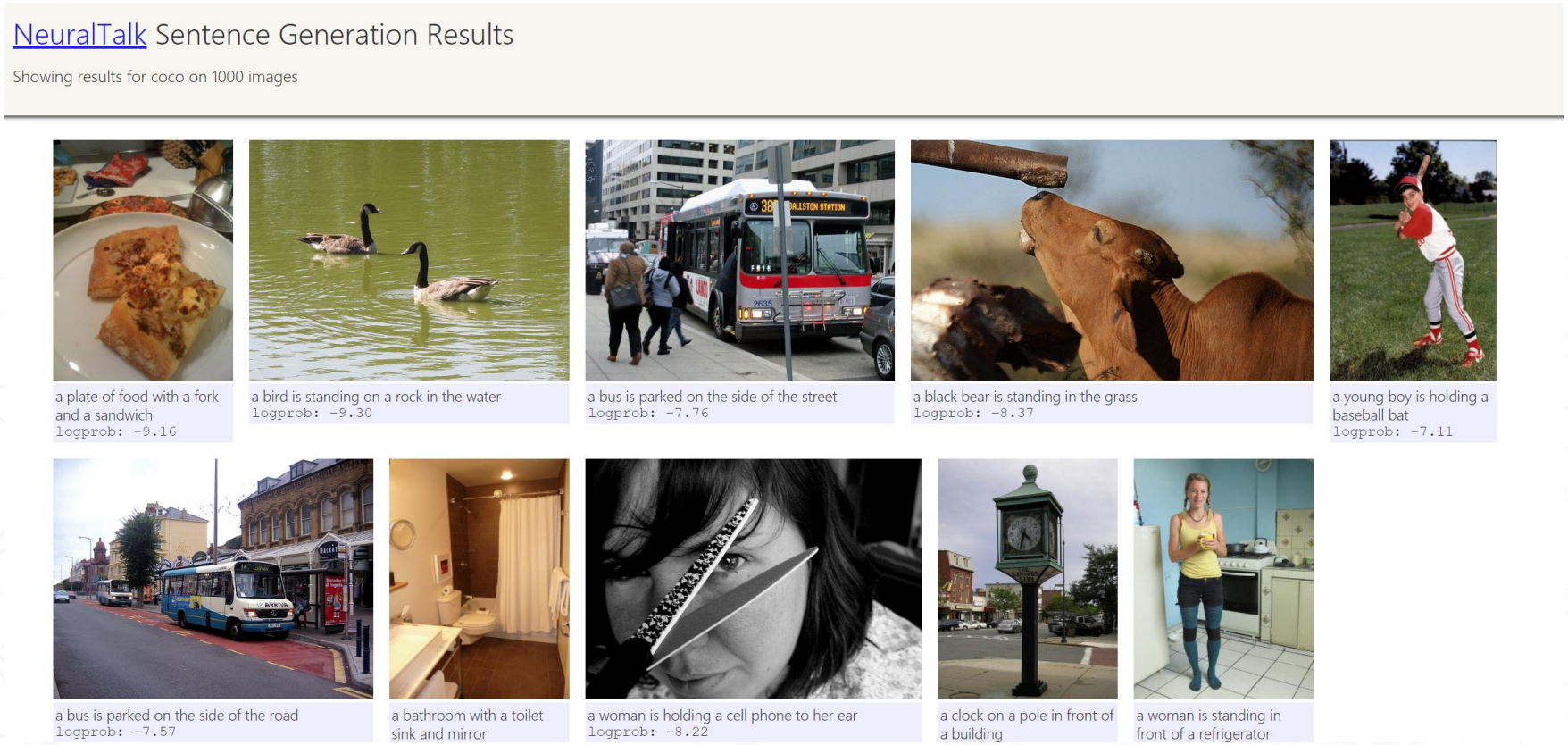


Figure 1. Motivation/Concept Figure: Our model treats language as a rich label space and generates descriptions of image regions.

https://cs.stanford.edu/people/karpathy/deepimagesent/generationdemo/

Applications

❖ Describe a picture



https://cs.stanford.edu/people/karpathy/deepimagesent/generationdemo/

Applications

- Real-time behavioral analysis

https://www.youtube.com/watch?time_continue=166&v=xhp47v5OBXQ&feature=emb_logo

Applications

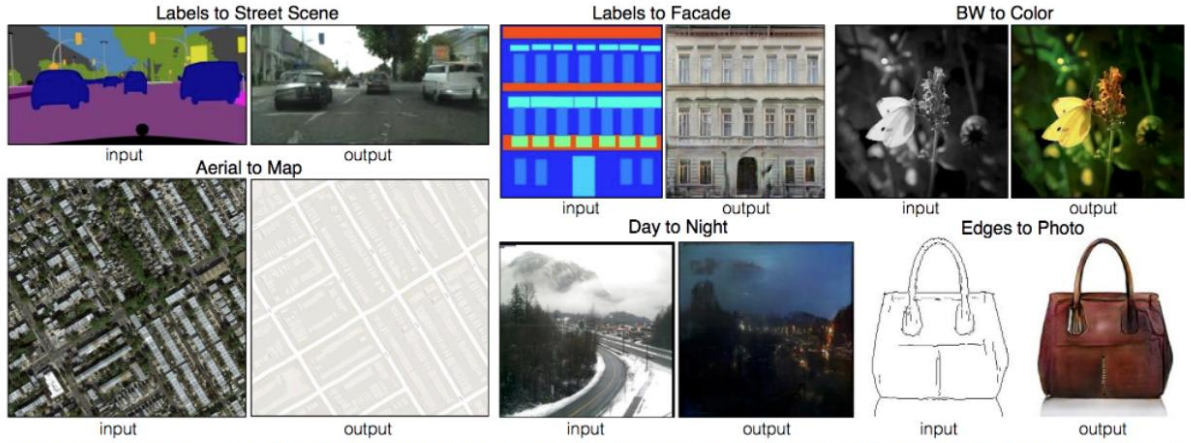
❖ Generating a new image

Image-to-Image Translation with Conditional Adversarial Nets

Phillip Isola Jun-Yan Zhu Tinghui Zhou Alexei A. Efros

University of California, Berkeley
In CVPR 2017

[Paper] [GitHub]



Example results on several image-to-image translation problems. In each case we use the same architecture and objective, simply training on different data.

Abstract

We investigate conditional adversarial networks as a general-purpose solution to image-to-image translation problems. These networks not only learn the mapping from input image to output image, but also learn a loss function to train this mapping. This makes it possible to apply the same generic approach to problems that traditionally would require very different loss formulations. We demonstrate that this approach is effective at synthesizing photos from label maps, reconstructing objects from edge maps, and colorizing images, among other tasks. As a community, we no longer hand-engineer our mapping functions, and this work suggests we can achieve reasonable results without hand-engineering our loss functions either.

https://phillipi.github.io/pix2pix/

Applications

❖ Generating a new image

사진이나 신발의 아웃라인을 그리면 딥러닝을 이용한 모형이 알아서 색을 칠해줄 것입니다.

https://phillipi.github.io/pix2pix/