CNN

-Course4-

1. 컴퓨터 비전의 발전이 어플리케이션 발전

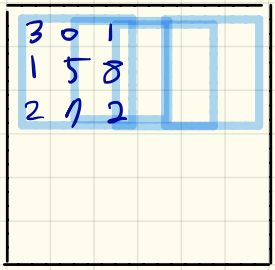
2. 창의적, 다른곳에 적용가능

64X64X3 (RGB) → 12288

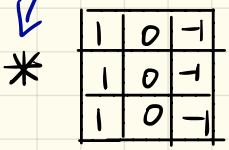
1000 X 1000 X3 (RGB) → 3million

Convolutional neural network

3x1+0+1x-1 +1x1+5x0+8x-1 ...

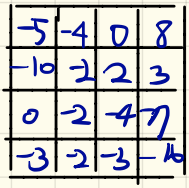


convolution



filter라 부름

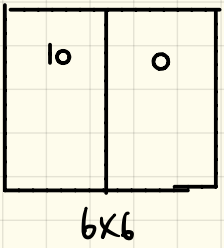
=



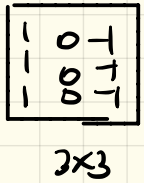
Python : conv-forword

tensorflow :

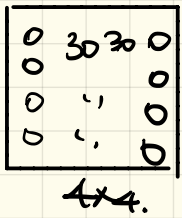
vertical edge detection



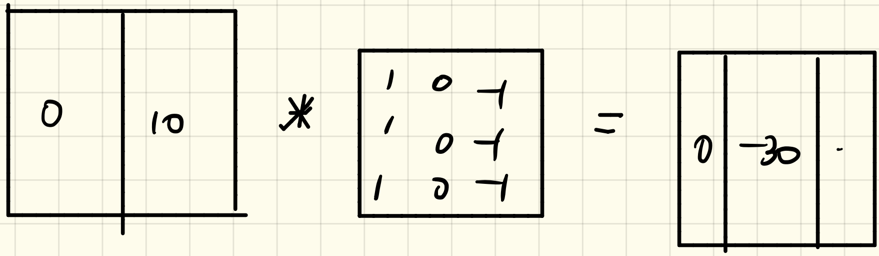
*



=



이 matrix에서 수직경계선을 찾을수 있게 해준다.



$$\begin{bmatrix}1&0&-1\\1&0&-1\\1&0&-1\end{bmatrix}\quad\begin{bmatrix}1&1&1\\0&0&0\\-1&-1&-1\end{bmatrix}\quad\begin{bmatrix}1&0&-1\\2&0&-2\\1&0&-1\end{bmatrix}\quad\begin{bmatrix}3&0&-3\\10&0&-10\\3&0&-3\end{bmatrix}$$

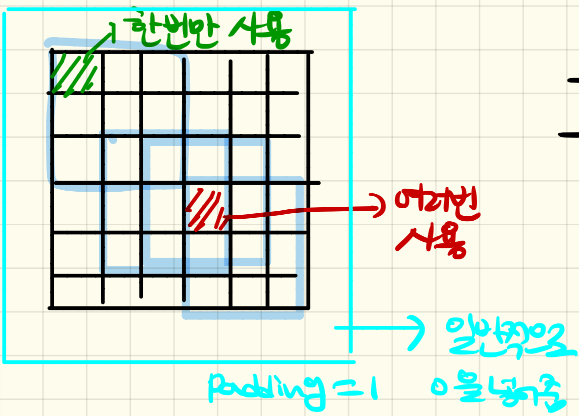
vertical　　horizontal　　Sobel filter　　Scharr filter

세로 윤곽선 검출을 위한것, 90° 회전하면 가로 윤곽선 검출

Padding.

6×6 * 3×3 = ~~4×4~~ $(n+2p-f+1)\ (n+2p-f+1)$

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



- shrinkly output
- throw away info from edge

Valid : (→ no padding) $n \times n \ * \ f \times f \rightarrow n-f+1 \times n-f+1$

Same : pad so that output size is the same as the input size

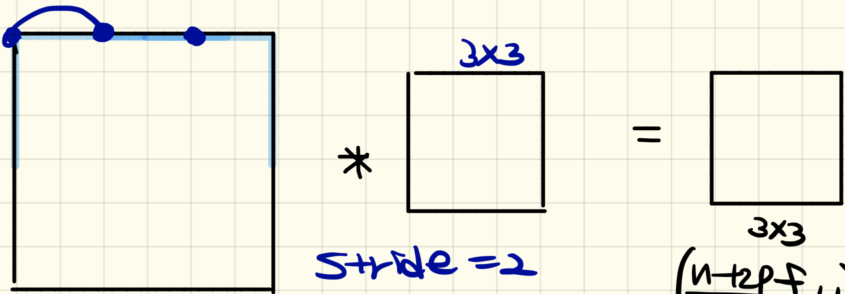
$(n+2p-f+1) \times (n+2p-f+1)$

$p = \frac{f-1}{2}$ (f → filter)

f : 홀수

① 짝수면 패딩 비대칭
② 중심 위치가 존재

Strided convolution



7x7

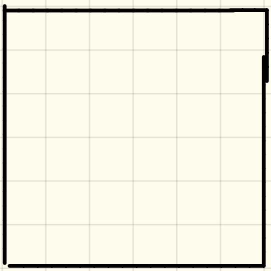
$n \times n \ * \ f \times f$

stride = 2

$\left(\frac{n+2p-f}{s}+1\right) \times \left(\frac{n+2p-f}{s}+1\right)$

$\lfloor z \rfloor$ = floor(z) 가장 작은 정수
계산 결과는 내림 정수.

Technical note on cross-correlation vs convolution.



*

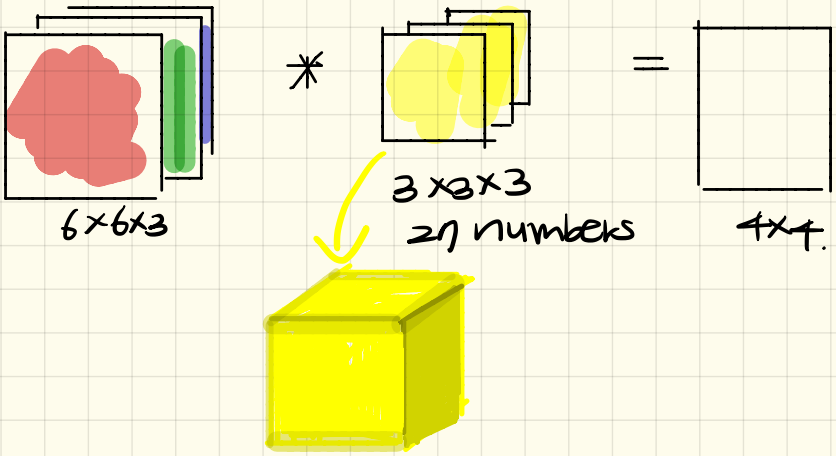
3	4	5
1	0	2
-1	9	7

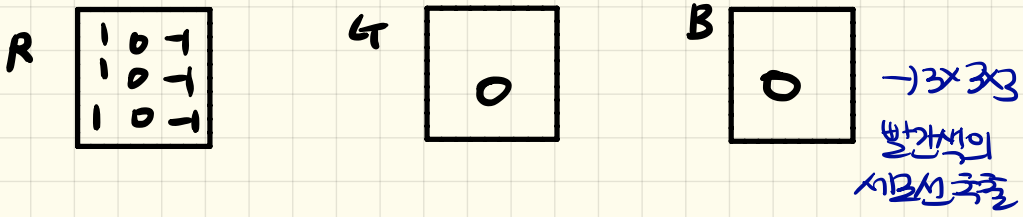
→ 가로와 세로축 뒤집기

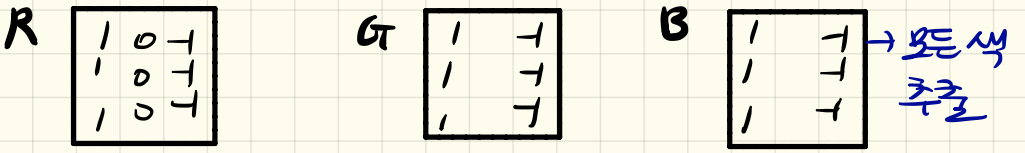
7	2	5
9	0	4
-1	1	3

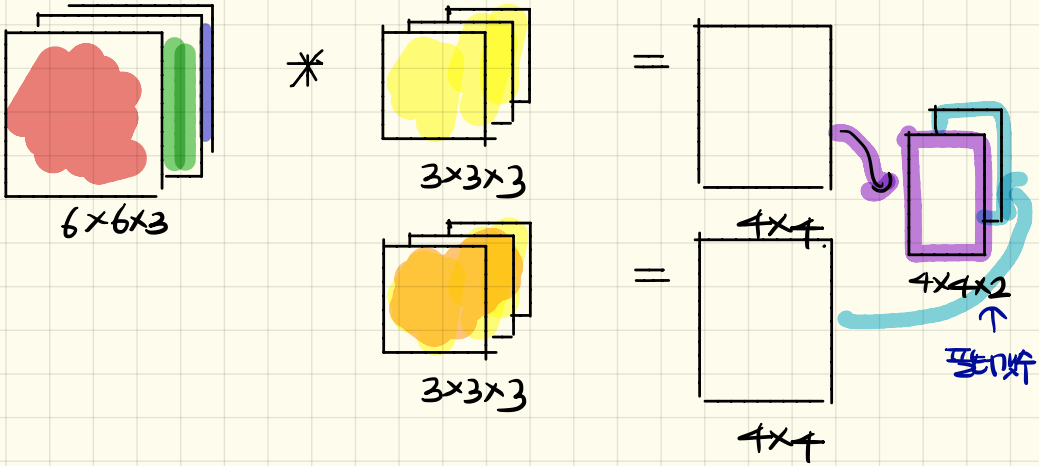
딥러닝에서 이런 과정
그걸 생략하고 함

Convolutions on RGB images.









Summary : $n \times n \times n_c$ * $f \times f \times n_c$

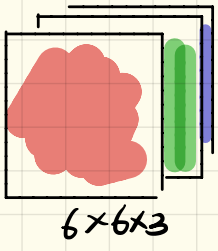
$6 \times 6 \times 3$ $\quad$ $3 \times 3 \times 3$

$\rightarrow$ $n-f+1 \times n-f+1 \times n_c'$

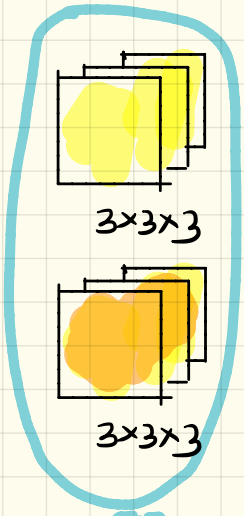
$4 \times 4 \times 2$ $\quad$ ↳ # filters

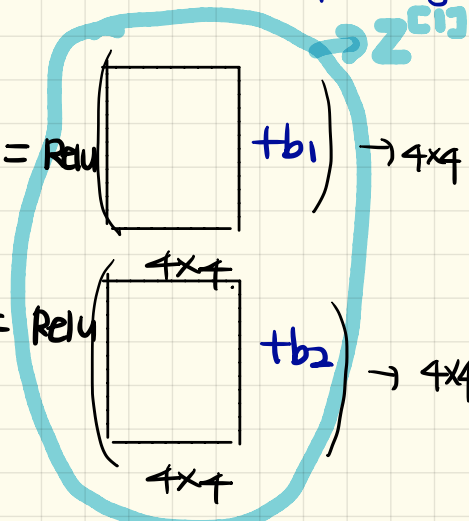
no stride

no padding



*



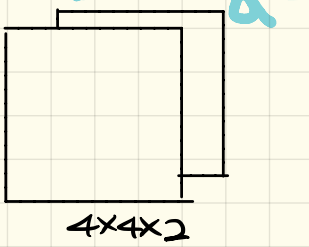


$w^{[1]}$

신경망 마지막 층으로 만들기위해 편향 추가

$z^{[1]} = w^{[1]} a^{[0]} + b^{[1]}$ ($a^{[0]}$: x)

$a^{[1]} = g[z^{[1]}]$



27 para
+ bias
→ 28 para

∴ 280 parameters

If layer l is a convolutional layer

$f^{[l]}$ = 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 : $n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$

$$n^{[l]} = \left[\frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right]$$

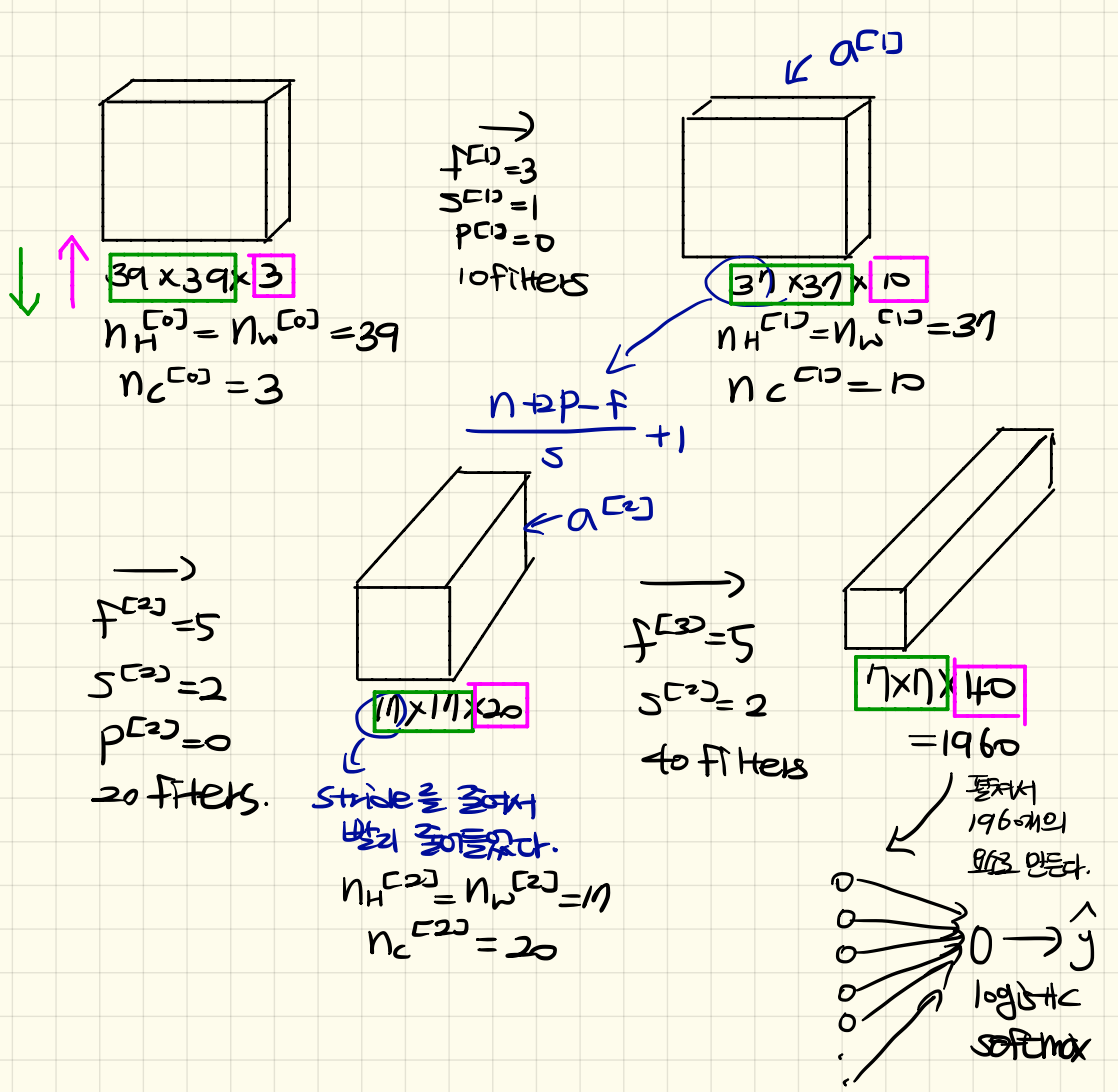
Each filter is : $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$

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

$A^{[l]} \rightarrow m \times n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$

weights : $f^{[l]} \times f^{[l]} \times f^{[l]} \times n_c^{[l]}$

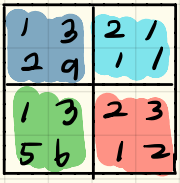
bias : $n_c^{[l]} - (1, 1, 1, n_c^{[l]})$



Types of layer in convolutional network:

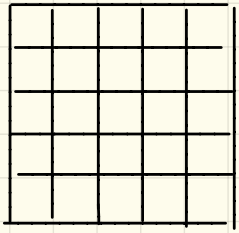
- Convolution (CONV)
- Pooling (POOL)
- Fully connected (FC)

Max Pooling

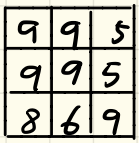


4x4

2x2
f=2
s=2



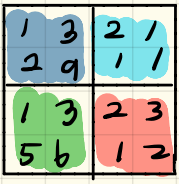
5x5



f=3
s=1

$\left(\frac{n-2p+f}{s}+1\right)$

Average Pooling



4x4

2x2
f=2
s=2

신경망 에서는 최대풀링을 평균풀링 보다 많이 사용한다.

Summary of Pooling

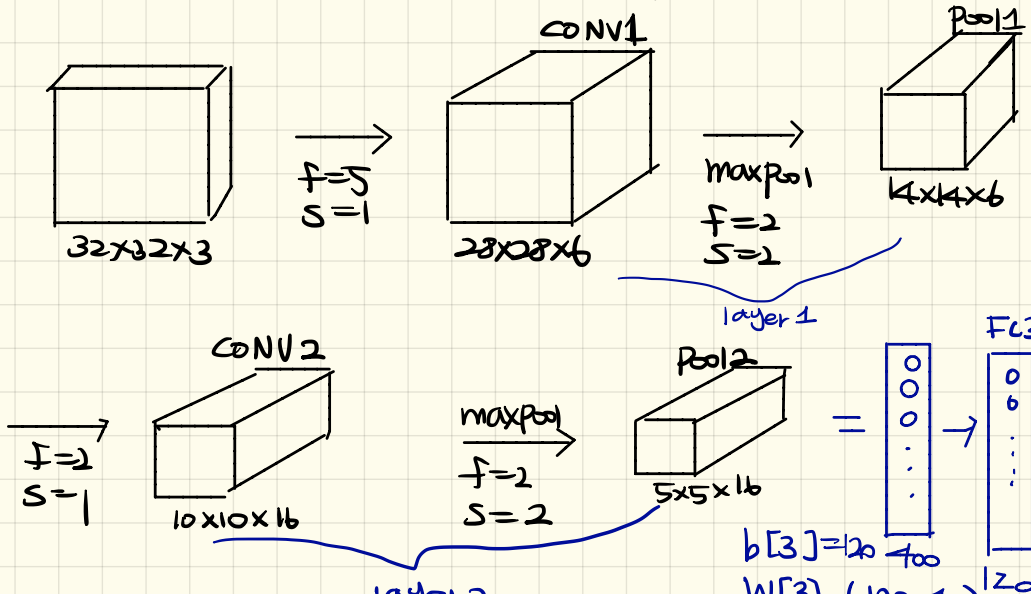
Hyper Parameters :

f: filter size — f=2, S=2 / f=3, S=2
S: Stride
Max or Average pooling

~~P: Padding~~ 패딩은 사용하지 않는다. 일반적인 값은 0.

$$n_H \times n_w \times n_c \rightarrow \left[\frac{n_H - f}{s} + 1\right] \times \left[\frac{n_w - f}{s} + 1\right] \times n_c$$

Neural Network example(LeNet-5)



Neural Network example

	Activation Shape	Activation size	# parameters
Input	(32,32,3)	3072 $a^{[0]}$	0
CONV1(f=5,s=1)	(28,28,8)	6272	208
POOL 1	(14,14,8)	1568	0 ← max Pooling
CONV2(f=5,s=1)	(10,10,16)	1600	416
POOL2	(5,5,16)	400	0 ←
FC3	(120,1)	120	48001 } 대부분의 변수는 FC층에 있음
FC4	(84,1)	84	10081 }
Softmax	(10,1)	10	841

why convolution ?

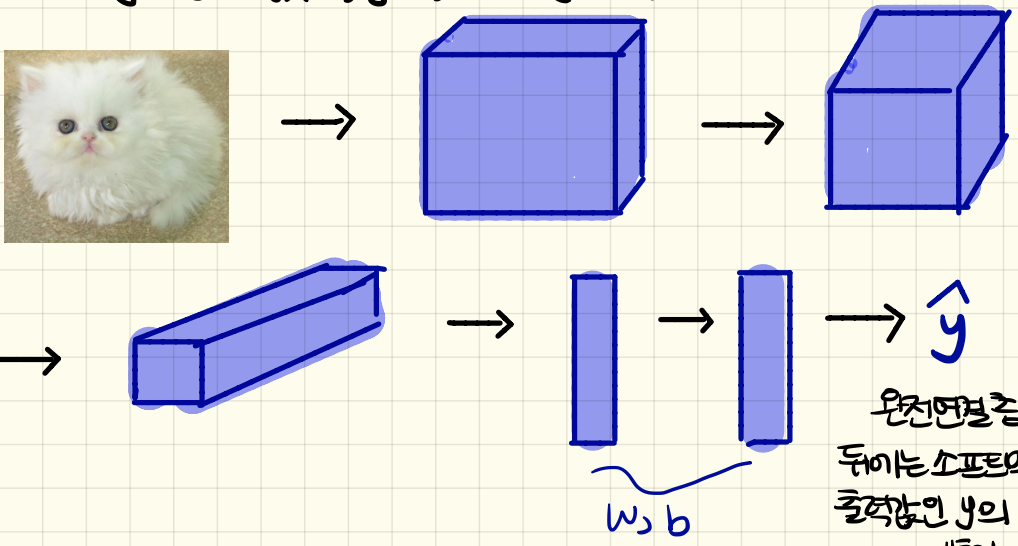
① FC3 하면 변수가 너무 많아진다.

Parameter sharing: A feature detector that's useful in one part of the image is probably useful in another part of the image.

② **sparsity of connections**: In each layer, each output value depends on a small number of inputs.

putting it together

Training set $(x^{(1)}, y^{(1)}) \cdots (x^{(m)}, y^{(m)})$

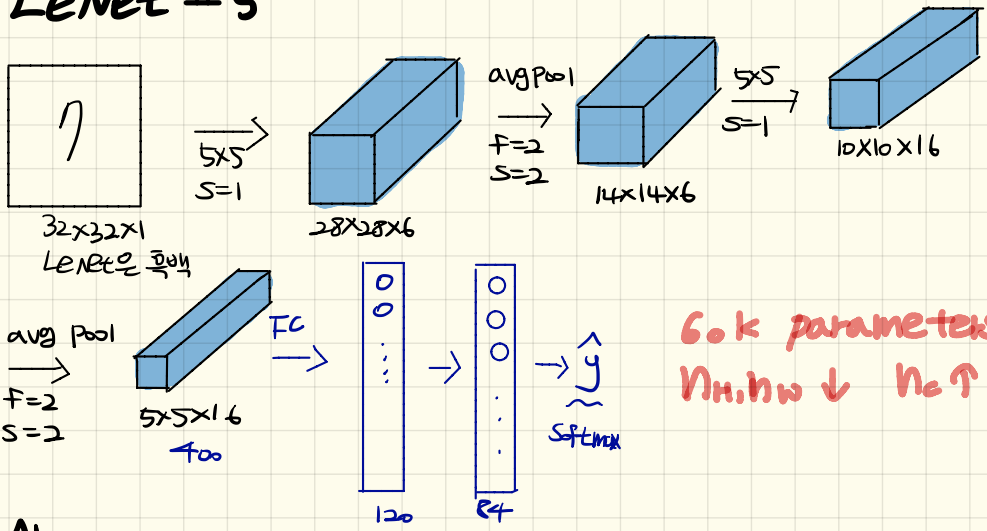


로짓변환을 통해 소프트맥스 출력값인 y의 예측값

$$\text{Cost } J = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$

Use gradient descent to optimize parameters to reduce J

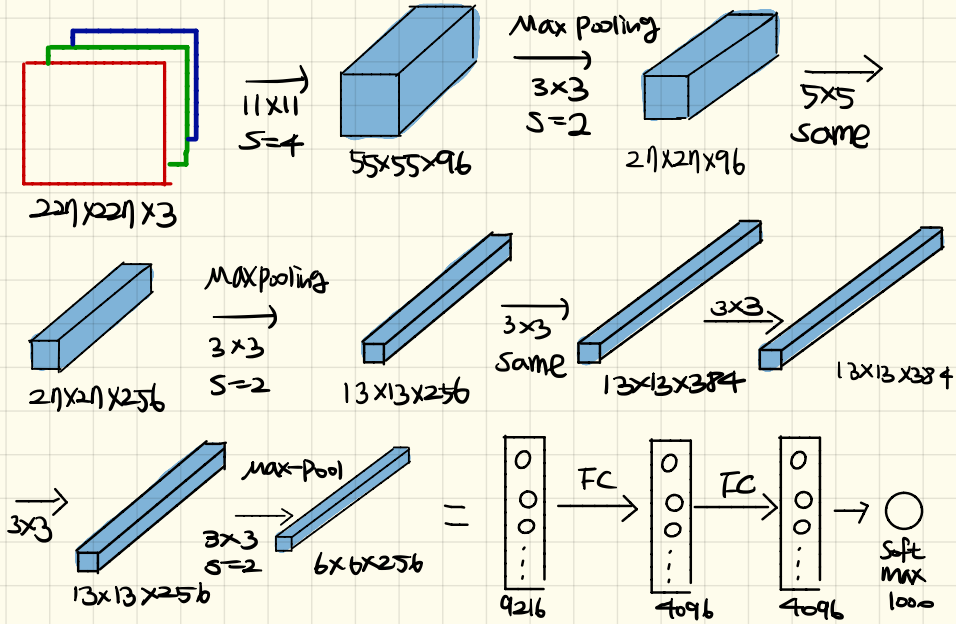
LeNet-5



60k parameters

$n_H, n_W \downarrow$ $n_C \uparrow$

AlexNet

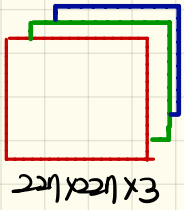


- Similar to LeNet, but much bigger
- ReLU 를 사용
- Multiple GPUs
- local Response normalization (LRN)

Less important

VGG-16 → 16개의 가중치를 가진 층이 있다.

CONV = 3x3 filter, s=1, same MaxPool = 2x2, s=2



→ 224x224x64 (CONV64 x2) → 112x112x64 (Pool) → 112x112x128 (CONV128 x2)

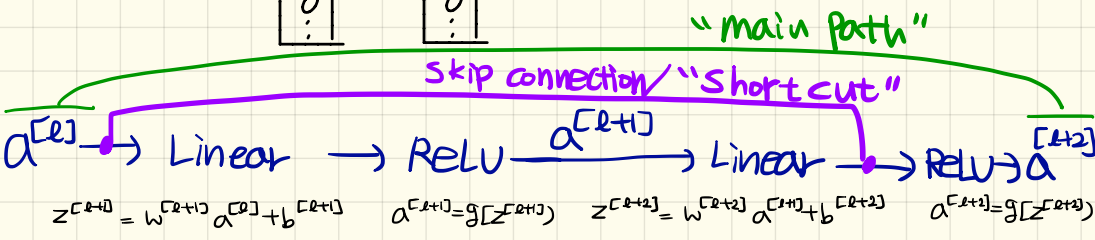
↳ 64개의 필터를 가진 두 개의 합성곱층을 사용한다.

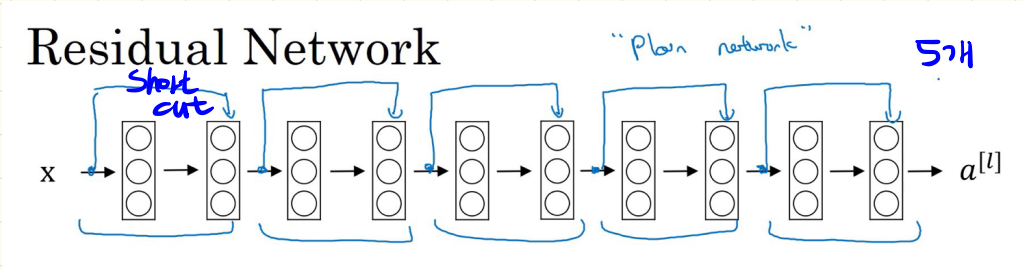
→ 56x56x128 (Pool) → (CONV256 x3) → ... → Softmax 1000

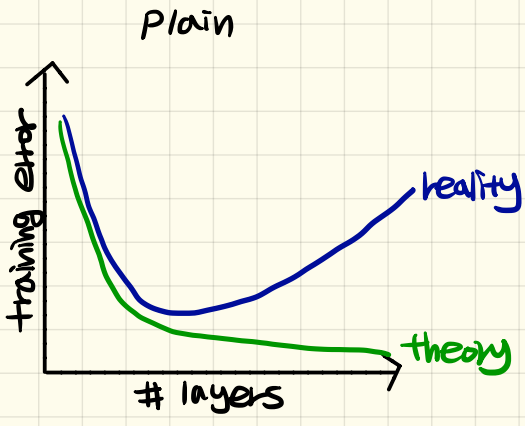
- 꼬내나 균일하고 간결하다.

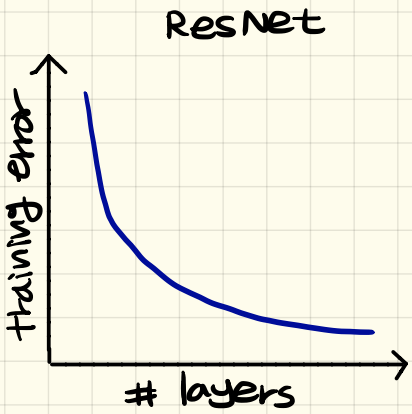
Residual block

$a^{[l]} \rightarrow \rightarrow \rightarrow a^{[l+2]}$









Why do residual networks work?

$X \rightarrow$ Big NN $\xrightarrow{a^{[l]}}$ $\rightarrow a^{[l+2]}$ "same"

ReLU $a \geq 0$

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

$z^{[l+2]}$ 와 $a^{[l]}$ 이 같은 차원을 가진다고 가정

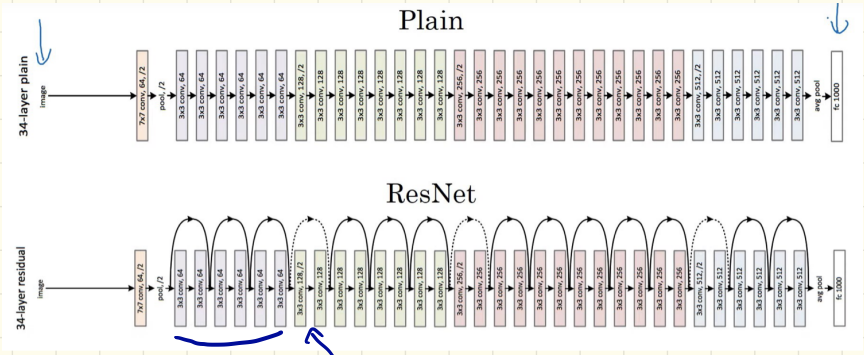
If 256

$$= g(w^{[l+2]} a^{[l+1]} + b^{[l+2]} + a^{[l]})$$

If $w^{[l+2]} = 0$ $b^{[l+2]} = 0$

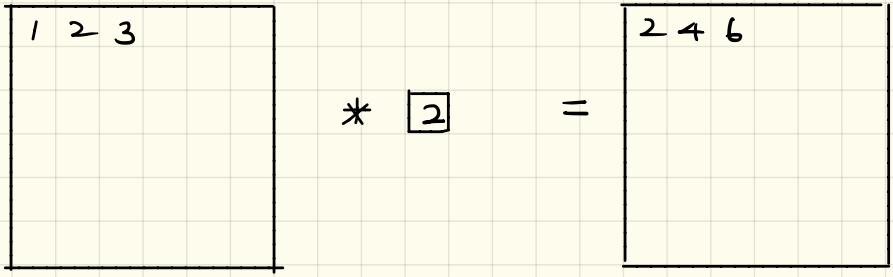
IR 항등함수 $= g(a^{[l]}) = a^{[l]}$

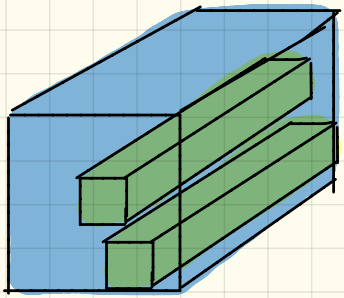
어느곳에 ResNet을 추가해도 성능저하가 없다.
경사하강법으로 더 나아질 뿐이다.

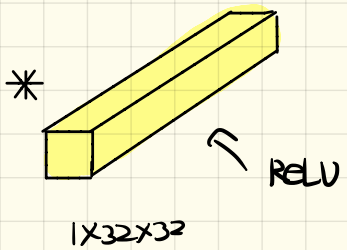


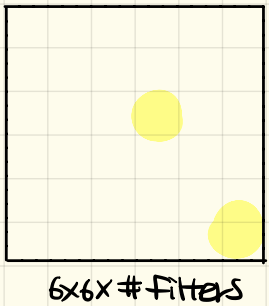
$z^{[l+1]} + a^{[l]}$

Why do a 1x1 Convolution do?





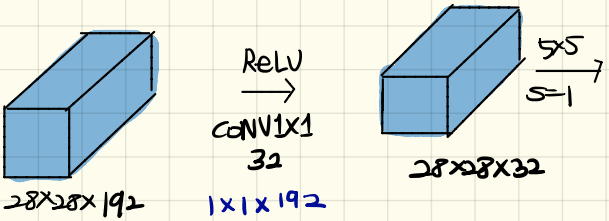




32 → # filters

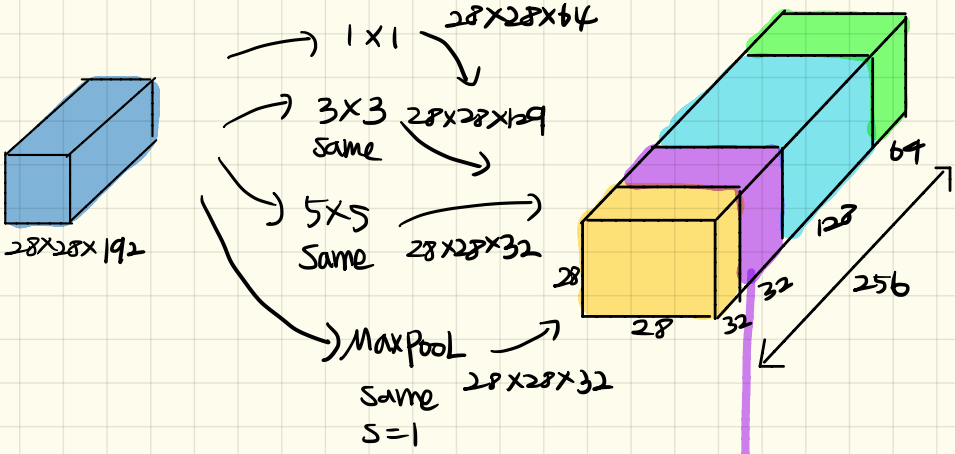
$n_c^{[l+1]}$

Using 1x1 convolutions



- 네트워크의 차원정상을 다루려고 채널수를 조절할수 있게 된다.

Motivation for inception network.



32filters filters are 5x5x192

28x28x32 x 5x5x192 = 120m

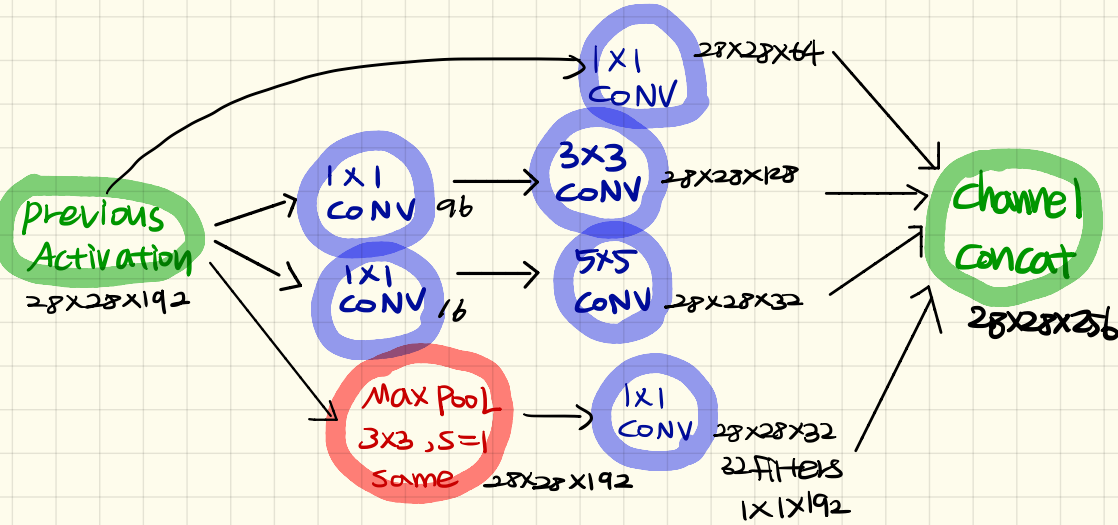
28X28X192 —CONV 1X1, 16 1X1X192→ 28X28X16 —CONV 5X5 32 5X5X16→ 28X28X32

비용 28X28X192X192 = 2.4M 28X28X32X5X5X16 = 10M

12.4M

필터가 1X1, 3X3, 5X5, 풀링층인지 고민하고 싶지 않다면 인셉션 모델은 그걸 다 썼은 것이다.

Inception module



Transfer Learning

- 기존 신경망을 다운로드 받을때 가중치도 받아야한다.
- 마지막 소프트맥스층만 수정.
- trainable parameter = 0, freeze = 1로 두면 훈련중은 그대로 유지된다.
- 데이터가 아주 많으면 네트워크를 다시 훈련.

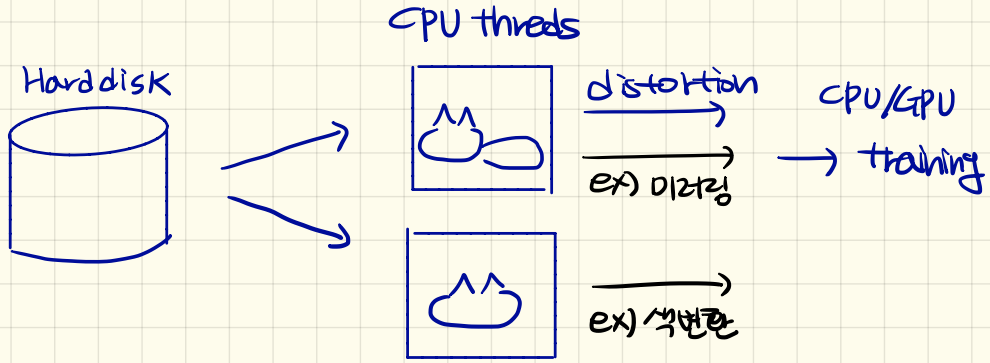
Common argumentation method

- mirroring
- Random cropping
- Rotation
- shearing
- Local wraping

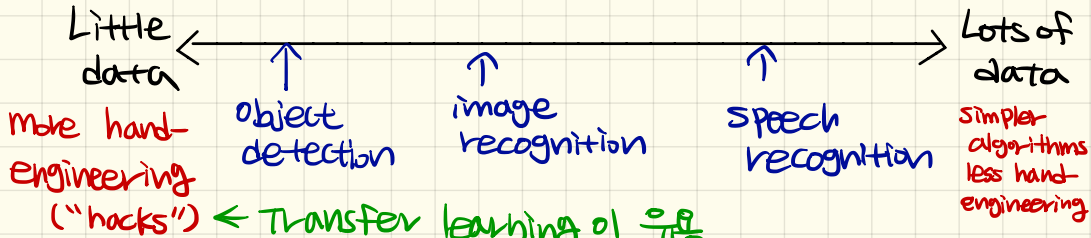
Color shifting

- RGB 조절
- PCA color argumentation.

Implementing distortions during training



Data vs hand-engineering



← Transfer learning 이 유용

Two sources knowledge

- labled data (x,y)
- Hand engineered ~~features~~/network architecture/ other components.

레이블 데이터가 충분하지 않으면 직접 설계에 의존해야 함

벤치마킹을 잘 하는 팁

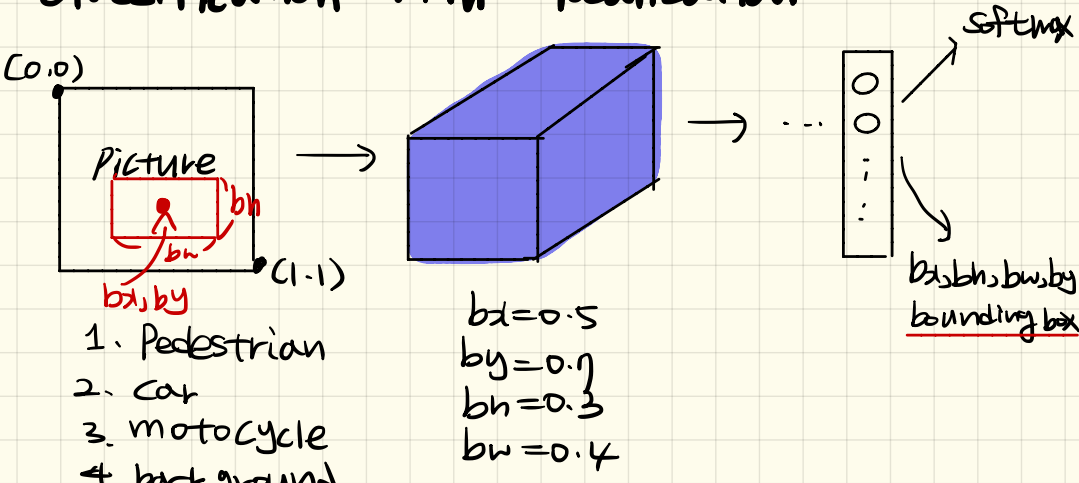
Ensembling

- 여러 신경망을 사용할지 결정하고, 몇개의 신경망을 독립적으로 훈련시킨 뒤 평균을 내는것.
  성능이 좋아 대회에선 쓰이지만 서비스용으로는 쓰이지않음.
  많은 양의 메모리가 필요.

Multi-crop at test time

- Run classifier on multiple versions of test images and average results.
  이것도 실제 제품 보다는 벤치마킹에 사용된다.
  네트워크 하나만으로 유지하고 돌릴 메모리를 많이 차지하진 않지만 여전히 실행 시간을 낮춘다.

classification with localization



Need to output b_x, b_y, b_h, b_w, class label (1–4)

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

is there any objelt?

$x =$

$$\begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

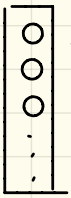
1. Pedestrian
2. car
3. motocycle
4. background

$$\mathcal{L}(\hat{y}, y) = \begin{cases} (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \cdots + (\hat{y}_8 - y_8)^2 & \text{if } y=1 \\ (\hat{y}_1 - y_1)^2 & \text{if } y_1 = 0 \end{cases}$$

Land mark detection.

만약 얼굴에 64개의 랜드마크가 있다면?

→ CONVNet →



face ?
l_{1x}
l_{1y}
$\vdots$
l_{64x}
l_{64y}

129
128+1
64x2+1

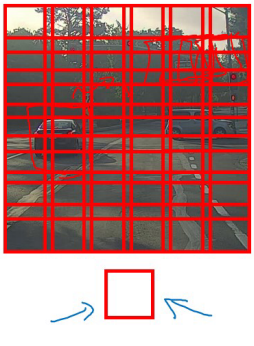
Car detection example

Training set:

x	y
	1
	1
	1
	0
	0

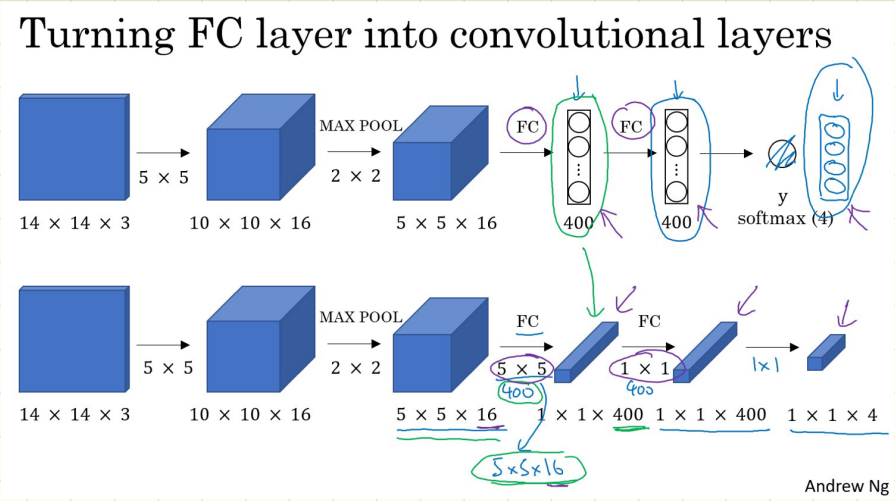
Andrew Ng

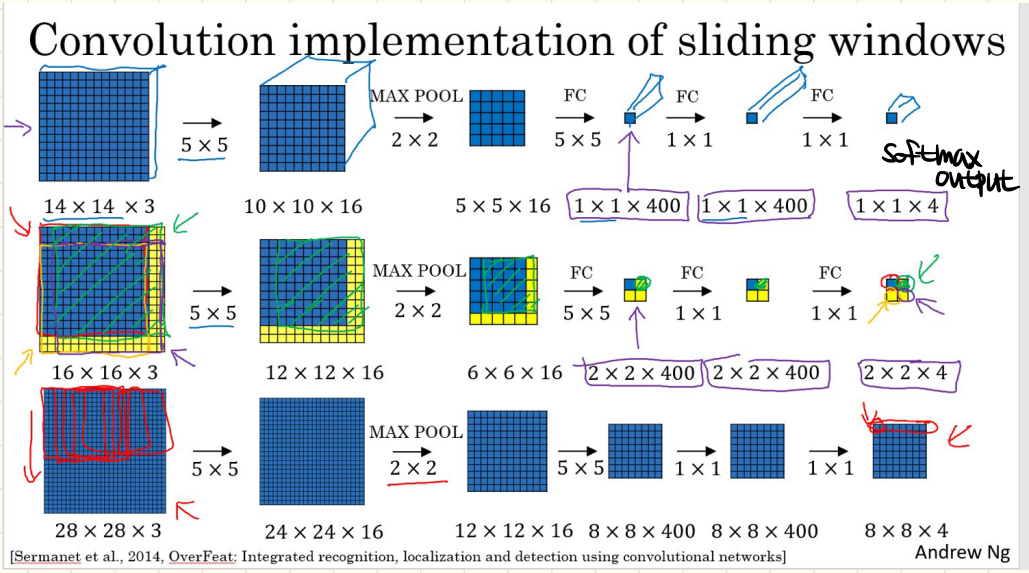
Sliding windows detection



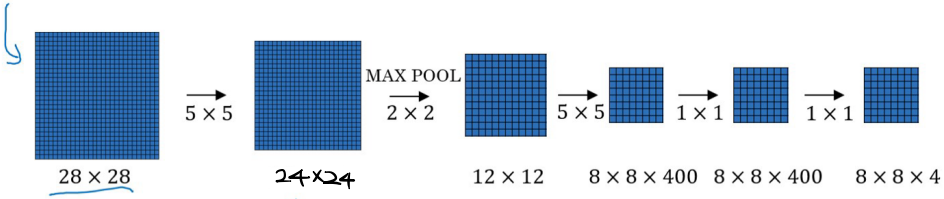
Andrew Ng

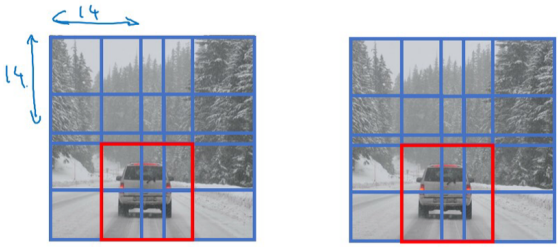
계산비용이 단점, 아주 큰 스트라이드는 정확도↓ 아주 작은 스트라이드는 아주 큰 계산비용.


Turning FC layer into convolutional layers
14 × 14 × 3
5 × 5
10 × 10 × 16
MAX POOL
2 × 2
5 × 5 × 16
FC
400
FC
400
y
softmax (4)
14 × 14 × 3
5 × 5
10 × 10 × 16
MAX POOL
2 × 2
5 × 5 × 16
FC
5 × 5
400
FC
1 × 1
400
1×1
1 × 1 × 400
1 × 1 × 400
1 × 1 × 4
5×5×16
Andrew Ng


Convolution implementation of sliding windows
14 × 14 × 3
5 × 5
10 × 10 × 16
MAX POOL
2 × 2
5 × 5 × 16
FC
5 × 5
FC
1 × 1
FC
1 × 1
Softmax output
1 × 1 × 400
1 × 1 × 400
1 × 1 × 4
16 × 16 × 3
5 × 5
12 × 12 × 16
MAX POOL
2 × 2
6 × 6 × 16
FC
5 × 5
FC
1 × 1
FC
1 × 1
2 × 2 × 400
2 × 2 × 400
2 × 2 × 4
28 × 28 × 3
5 × 5
24 × 24 × 16
MAX POOL
2 × 2
12 × 12 × 16
5 × 5
8 × 8 × 400
1 × 1
8 × 8 × 400
1 × 1
8 × 8 × 4
[Sermanet et al., 2014, OverFeat: Integrated recognition, localization and detection using convolutional networks]
Andrew Ng

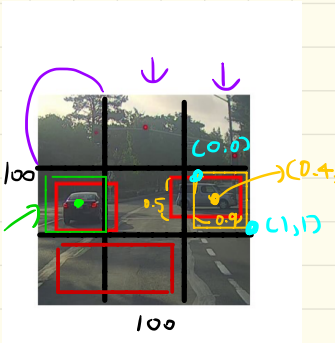
Convolution implementation of sliding windows





Andrew Ng

가장 큰 단점은 바운딩 박스의 위치가 그리 정확하지 않다는 것.



Labels for training

For each grid cell:

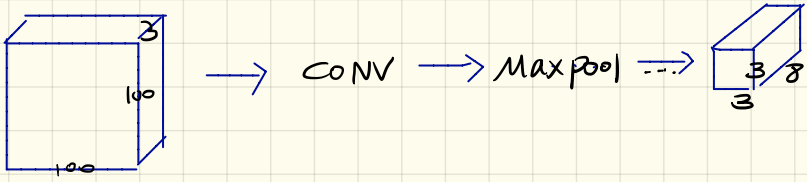
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \quad \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{matrix} \\ \\ 0.4 \\ 0.3 \\ 0.9 \\ 0.5 \\ \\ \\ \end{matrix}$$

between 0, 1

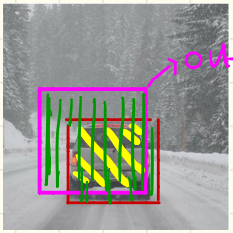
could be > 1

Target output

3×3×8



Evaluating object localization



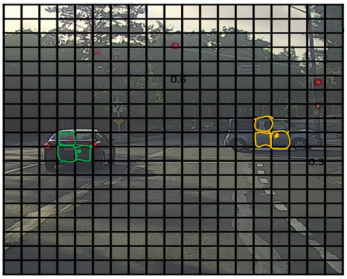
Intersection over Union (IoU)

$$\frac{\text{Size of } \square \text{ (yellow)}}{\text{Size of } \square \text{ (green)}}$$

"correct" if IoU ≥ 0.5

IoU↑ 바운딩박스정확

Non-max suppression example



19×19

Each output prediction is: $\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \end{bmatrix}$

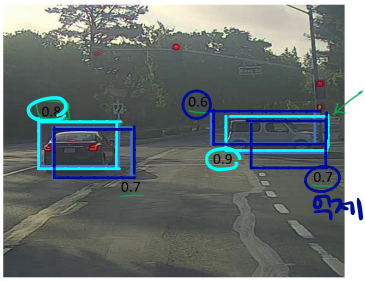
Discard all boxes with $p_c \leq 0.6$.

While there are any remaing boxes:

- 가장 높은 p_c를 가지고 있는 상자들을 반복적으로 골라서 예측
- 가장 높은 IoU 제외하고 어둡게하기

Andrew Ng

Non-max suppression example

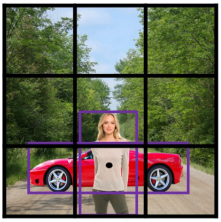


Non-max suppression은 이러한 탐지를 깨끗하게 정리하는것

Andrew Ng

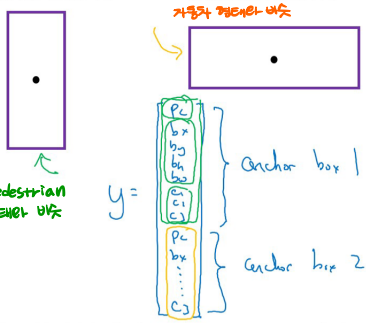
Anchor Boxes

Overlapping objects:



$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Anchor box 1: Anchor box 2:



[Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

Andrew Ng

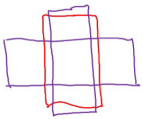
Anchor box algorithm

Previously:

Each object in training image is assigned to grid cell that contains that object's midpoint.

Output y:

$3 \times 3 \times 8$



With two anchor boxes:

Each object in training image is assigned to grid cell that contains object's midpoint and anchor box for the grid cell with highest IoU.

(grid cell, anchor box)

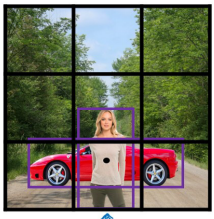
Output y:

$3 \times 3 \times 16$

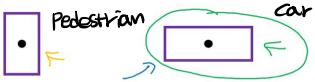
$3 \times 3 \times 2 \times 8$

Andrew Ng

Anchor box example



Anchor box 1: Anchor box 2:



$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

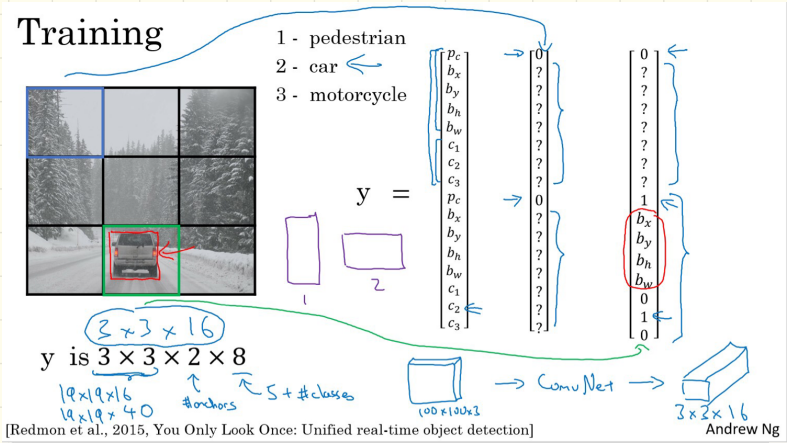
car only? no Pedestrian

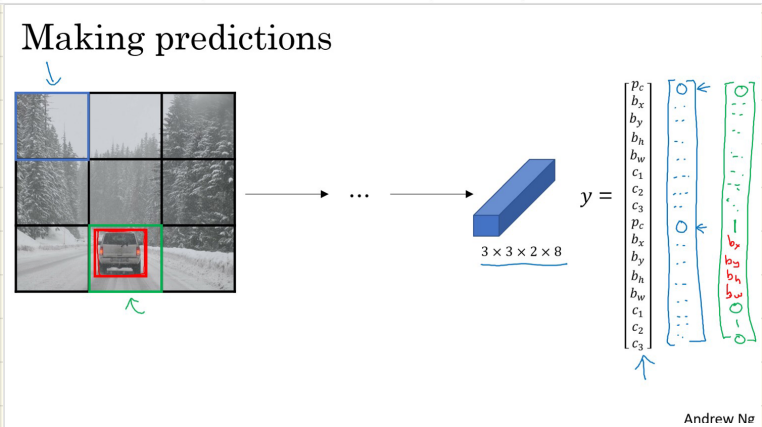
anchor box 1

anchor box 2

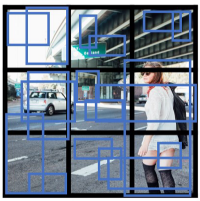
Andrew Ng

YOLO Algorithm





Outputting the non-max supressed outputs



- For each grid cell, get 2 predicted bounding boxes.
- Get rid of low probability predictions.
- For each class (pedestrian, car, motorcycle) use non-max suppression to generate final predictions.

Andrew Ng

Face verification vs face recognition

→ Verification 실제로 잘 작동하기 위해서는 99% 이상이어야함

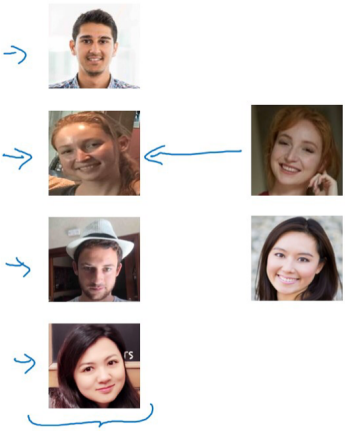
- Input image, name/ID
- output whether the input image is that of the claimed person.

→ Recognition

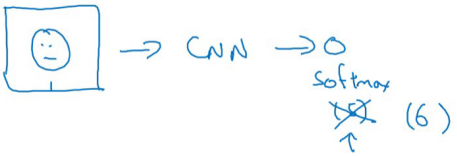
k<=100

- Has a database of k persons.
- Get an input image.
- output ID if the image is any of the k persons (or

One-shot learning



Learning from one example to recognize the person again



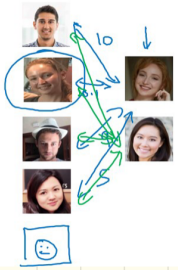
training set이 이렇게 작으면 훈련이 잘 안될것

Andrew Ng

Learning a "similarity" function

→ d(img1,img2) = degree of difference between images

If d(img1,img2) $\leq \tau$ "same"
$> \tau$ "different"
} Verification.

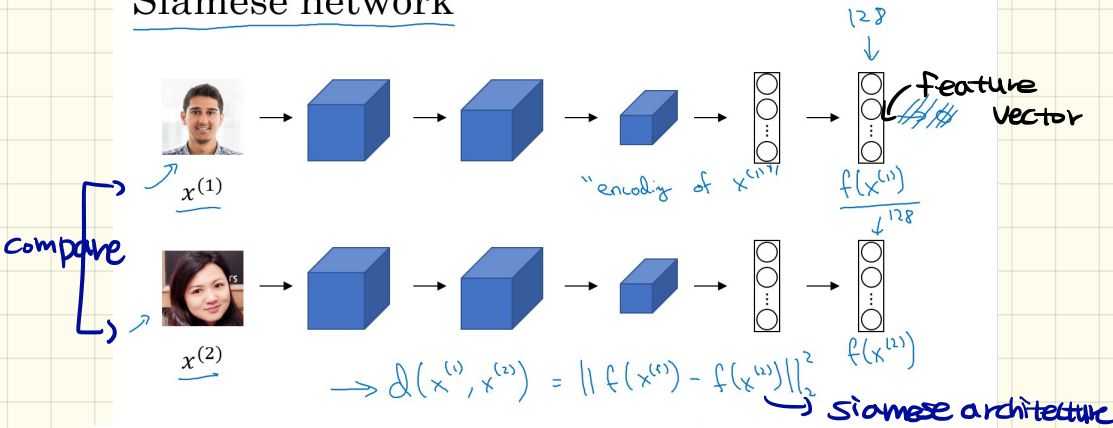


d(img1, img2)

두사진이 같으면 작은수 출력, 두 사진이 다르면 큰 수 출력하게 하여라

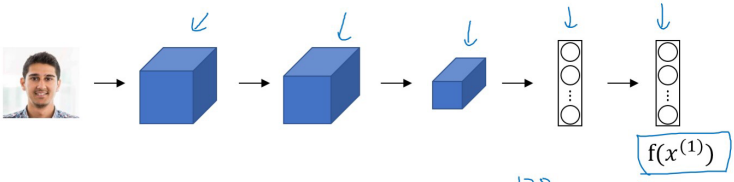
Andrew Ng

Siamese network



Andrew Ng

Goal of learning



Parameters of NN define an encoding $f(x^{(i)})$ — 128

Learn parameters so that:

If $x^{(i)}, x^{(j)}$ are the same person, $\|f(x^{(i)}) - f(x^{(j)})\|^2$ is small.

If $x^{(i)}, x^{(j)}$ are different persons, $\|f(x^{(i)}) - f(x^{(j)})\|^2$ is large.

}

Andrew Ng

Learning Objective

Anchor (A) Positive (P) Anchor (A) Negative (N)

$d(A,P) = 0.5$ $d(A,N) = 0.51 \rightarrow 0.7$

Want: $\underbrace{\|f(A) - f(P)\|^2}_{d(A,P)} + \alpha \le \underbrace{\|f(A) - f(N)\|^2}_{d(A,N)}$ (α: $\nearrow 0.2$)

$$\boxed{\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha \le 0}$$

(0, 0 under the two terms; α — margin; $f(\text{img}) = \vec{0}$)

[Schroff et al.,2015, FaceNet: A unified embedding for face recognition and clustering] Andrew Ng

Loss function

Given 3 images A, P, N :

$$\mathcal{L}(A,P,N) = \max(\underbrace{\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha}_{\le 0}, 0) = 0$$

$$J = \sum_{i=1}^{m} \mathcal{L}(A^{(i)}, P^{(i)}, N^{(i)})$$

> 0 =

A,P에 같은 사진이 있어야함.

training set : 10k pictures of 1k persons.

(사람 당 사진 하나는 훈련이 불가능하다)

Choosing the triplets A,P,N

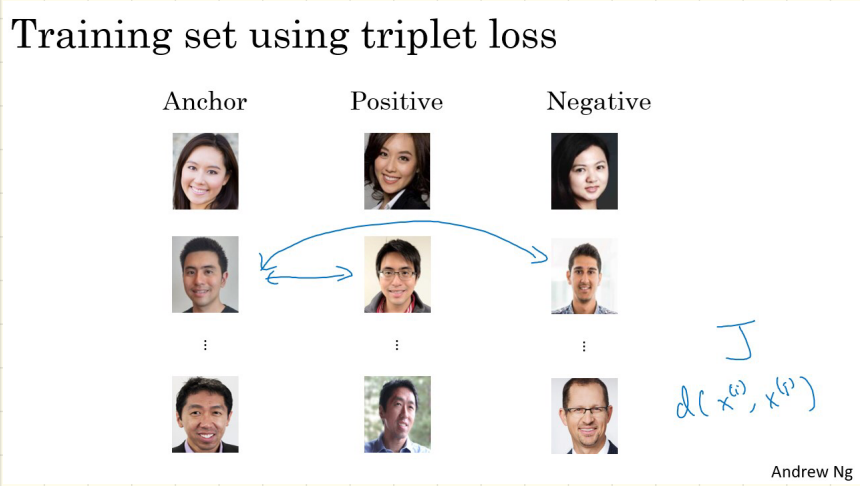
During training, if A,P,N are chosen randomly, $d(A,P) + \alpha \le d(A,N)$ is easily satisfied.

$\hookrightarrow \|f(A) - f(P)\|^2 + \alpha \le \|f(A) - f(N)\|^2$, 랜덤하게 선택되면 왼쪽의 오른쪽보다 훨씬 더 큰 확률을 가질 가능성이 매우 높다. 따라서 신경망으로 부터 배울수 있는것이 많이 없다.

choose triplets that're "hard" to train on.

$d(A,P) + \alpha \leq d(A,N)$

$d(A,P) \approx d(A,N)$ 해서 둘 사이의 α를 만들어줘야함

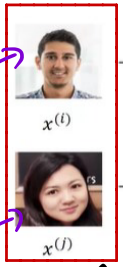


(J) 비용함수를 작게 만들기 위해 경사하강을 사용하라.
그러면 신경망의 모든 파라미터에 역전파 되는 효과가 있다.

$d(x^{(i)}, x^{(j)})$ ↓ "same" $d(x^{(i)}, x^{(j)})$ ↑ "different"

Learning the similarity function

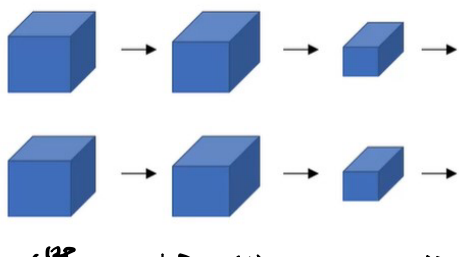
Siamese network



New

data set

training set



같은사람이면 1
다른사람이면 0

여기계산.

$$\hat{y} = \sigma\left(\sum_{k=1}^{128} w_k \left| f(x^{(i)})_k - f(x^{(j)})_k \right| + b\right)$$

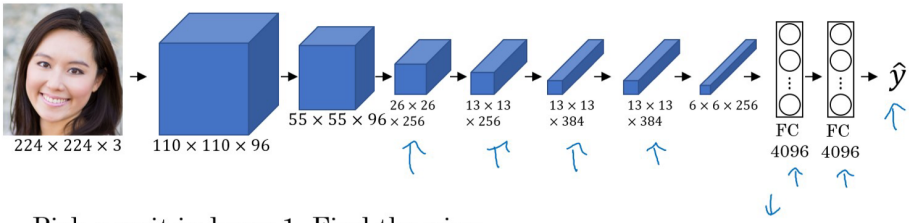
$$\frac{(f(x^{(i)})_k - f(x^{(j)})_k)^2}{f(x^{(i)})_k + f(x^{(j)})_k}$$ → χ^2 유사도

새로운 직원이 들어오면 위쪽 구성 요소를 사용하여 인코딩을 계산하고 사용하면 된다.

Face verification supervised learning

x	y	
	1	"Same"
	0	"Different"
	0	
	1	

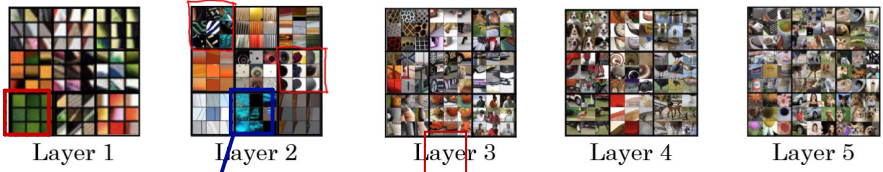
Visualizing what a deep network is learning



Pick a unit in layer 1. Find the nine image patches that maximize the unit's activation.

Repeat for other units.

Visualizing deep layers



대각선 텍스처

수직선이 많은 수직 텍스처를 찾고 있는것으로 보임

둥근 모양과 점이며 사물을 인식하려함

둥근이미지에 매우 활성화 된 것처럼 보임.

점점 점고해짐

Andrew Ng

Cost function.

Generated 된 Content 가 얼마나 비슷한지

$J(G) = \alpha J_{content}(C, G) + \beta J_{style}(S, G)$

Find the Generated image G.

1. Initiate G randomly G: 100 x 100 x 3
2. Use gradient descent to minimize J(G)

$G := G - \frac{\partial}{\partial G} J(G)$

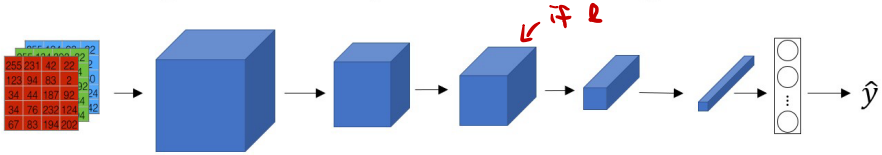
Content cost function

$J(G) = \alpha J_{content}(C, G) + \beta J_{style}(S, G)$

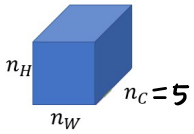
$\hookrightarrow \frac{1}{2}\| a^{[l](C)} - a^{[l](G)} \|^2$

- Say you use hidden layer l to compute content cost.
  너무 얕지도 깊지도 않게 히든레이어 설정.
- Use pre-trained ConvNet (E.g. VGG network)
- Let $a^{[l](C)}$ and $a^{[l](G)}$ be the activation of layer l on the images.
- If $a^{[l](C)}$ and $a^{[l](G)}$ are similar, both images have similar content.

Meaning of the "style" of an image



Say you are using layer l's activation to measure "style."
Define style as correlation between activations across channels.



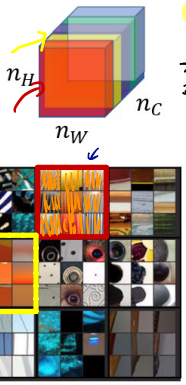
How correlated are the activations across different channels?

[Gatys et al., 2015. A neural algorithm of artistic style]

Andrew Ng

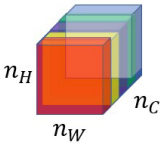
Intuition about style of an image

Style image



Correlated!
이세판 수직 텍스쳐 유형을 가진 있으면 이 이미지 내용도 주황빛 계열을 가지게 된다는 뜻.

Generated Image



[Gatys et al., 2015. A neural algorithm of artistic style]

Andrew Ng

Style matrix

Let $a^{[l]}_{i,j,k}$ = activation at (i,j,k). $G^{[l]}$ is $n_c^{[l]} \times n_c^{[l]}$

H W C

$$G^{[l](S)}_{kk'} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} a^{[l](S)}_{ijk} a^{[l](S)}_{ijk'}$$

이미지의 서로 다른 위치 에서 높이와 너비에 대한 합.
채널 k와 k'의 activation을 곱한다.

둘 다 커지게 되면 $G_{kk'}$도 증가
둘 중 하나가 상관이 없다면 $G_{kk'}$는 감소

$$G^{[l](G)}_{kk'} = \sum_i \sum_j a^{[l](G)}_{ijk} a^{[l](G)}_{ijk'}$$ → Generated (grand matrix)

$$J^{[l]}_{style}(S,G) = \|G^{[l](S)} - G^{[l](G)}\|_F^2$$

$$= \sum_k \sum_{k'} (G^{[l](S)}_{kk'} - G^{[l](G)}_{kk'})^2$$

[Gatys et al., 2015. A neural algorithm of artistic style]

Andrew Ng

Style cost function

$$J_{style}^{[l]}(S,G) = \frac{1}{\left(2n_H^{[l]}n_W^{[l]}n_C^{[l]}\right)^2}\sum_k\sum_{k'}\left(G_{kk'}^{[l](S)} - G_{kk'}^{[l](G)}\right)^2$$

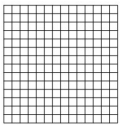
$$J_{style}(S,G) = \sum_l \lambda^{[l]} J_{style}^{[l]}(S,G)$$

$$J(G) = \alpha J_{content}(C,G) + \beta J_{style}(S,G)$$

[Gatys et al., 2015. A neural algorithm of artistic style]

Andrew Ng

Convolutions in 2D and 1D

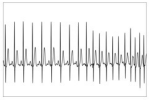


2D input image
14 × 14

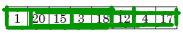
*

2D filter
5 × 5

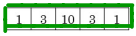
14×14×3 * 5×5×3
→ 10×10×16



*



14(1D)

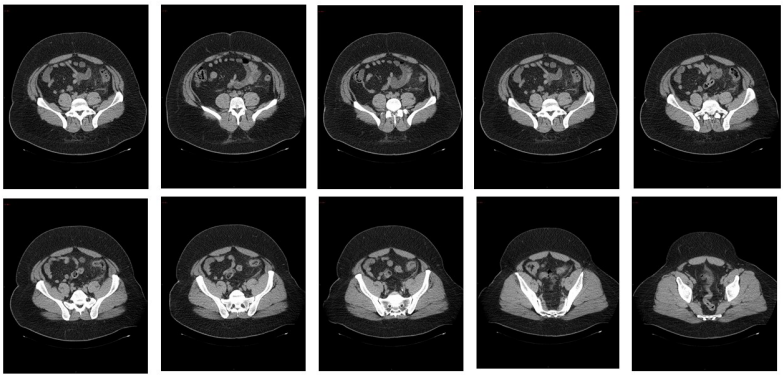


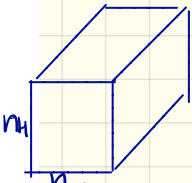
5

14 X1 * 5X1
→ 10×16
10×16 * 5*16
→ 6*32

Andrew Ng

3D data

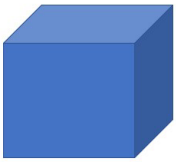




데이터의 높이, 너비, 깊이 같은지 확인하는것.

Andrew Ng

3D convolution



3D volume

*

3D filter

14x14x14x1

* 5x5x5x1 16filters

→ 10x10x10 x16

* 5x5x5x16 32filters

→ 6x6x6x32

Andrew Ng