Lecture 11:
Detection and Segmentation

Today: Segmentation, Localization, Detection

So far: Image Classification

This image is CC0 public domain

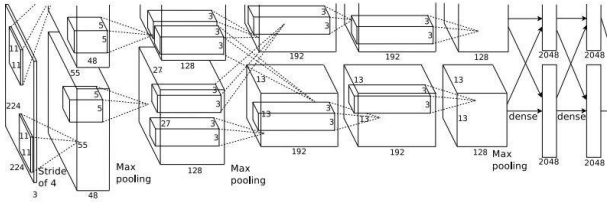


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Vector:
4096

Fully-Connected:
4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

So far: Image Classification

This image is CC0 public domain

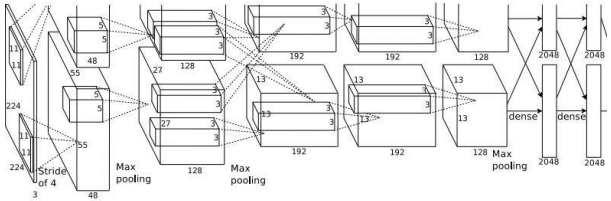


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Vector:
4096

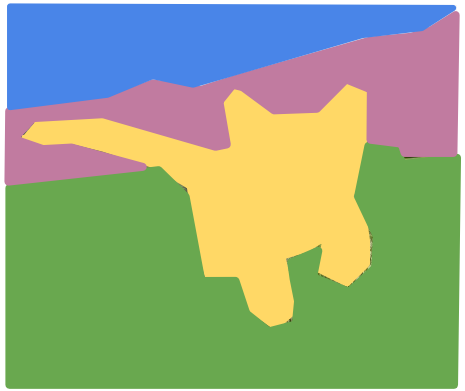
Fully-Connected:
4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Input: image → Deep Convolutional Network 통과 → Feature vector나옴

Other Computer Vision Tasks

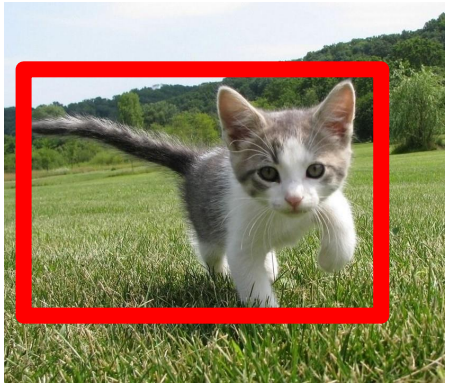
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

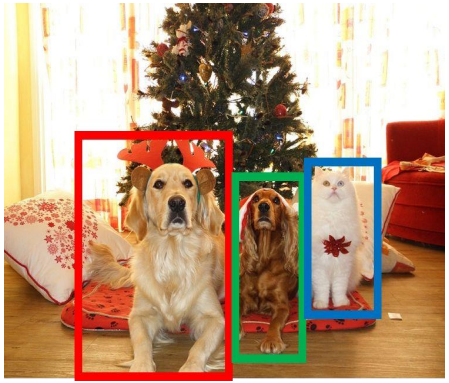
Classification + Localization



CAT

Single Object

Object Detection



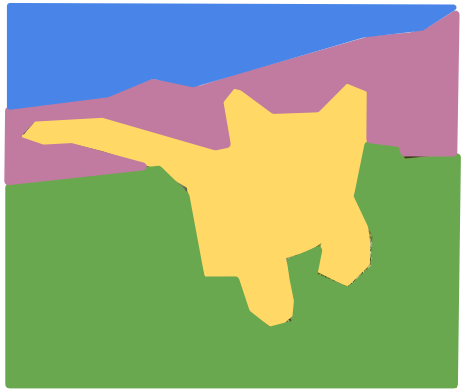
DOG, DOG, CAT

Instance Segmentation

DOG, DOG, CAT

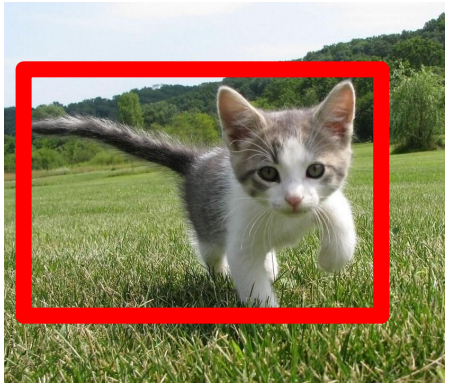
Multiple Object

Semantic Segmentation



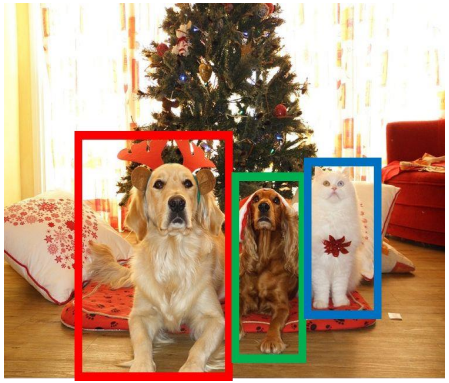
GRASS, CAT, TREE, SKY

No objects, just pixels



CAT

Single Object

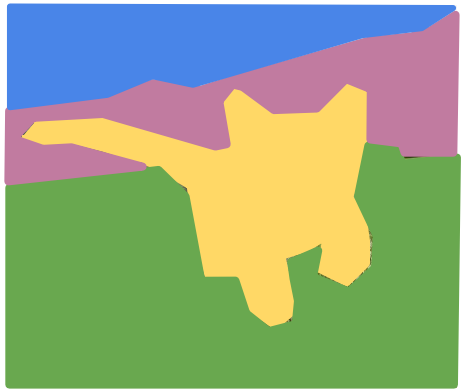


DOG, DOG, CAT

DOG, DOG, CAT

Multiple Object

Semantic Segmentation



<Input>
image

<Output>
모든 픽셀마다 카테고리가 지정

GRASS, CAT,
TREE, SKY

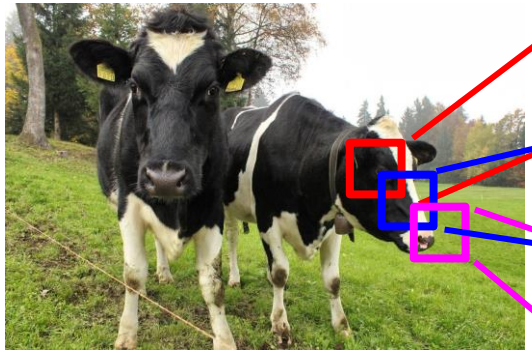
Semantic Segmentation

semantic segmentation은 단일 객체를 구별할 수 없다.

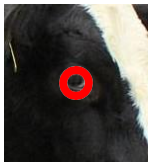
→ “빵 5개” 에 대한 구별x, 이 픽셀이 “빵”인 것만 알 수 있다

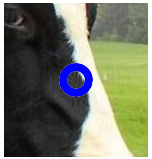
Semantic Segmentation Idea: Sliding Window

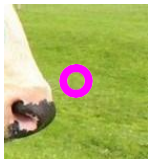
Full image



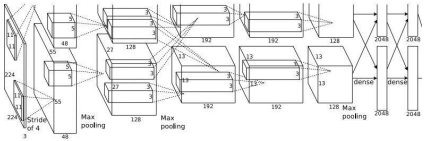
Extract patch



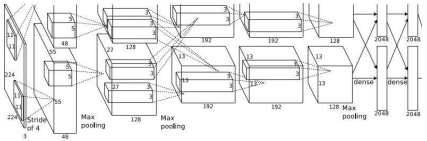




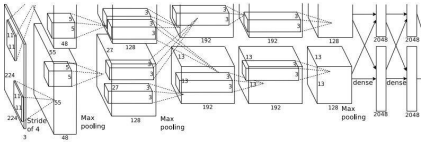
Classify center pixel with CNN



Cow



Cow



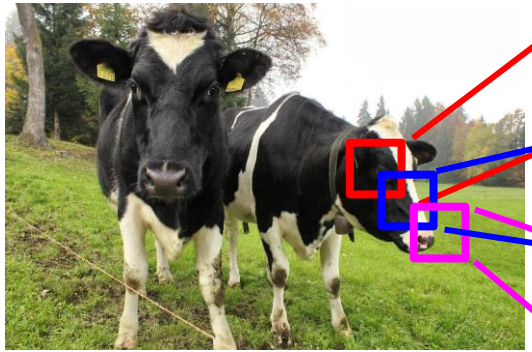
Grass

Farabet et al, “Learning Hierarchical Features for Scene Labeling,” TPAMI 2013

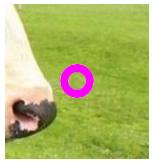
Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

Semantic Segmentation Idea: Sliding Window

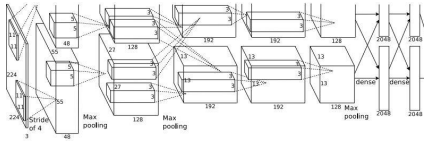
Full image



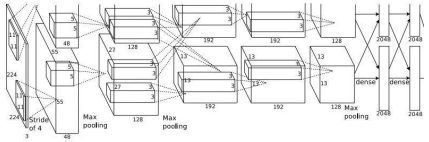
Extract patch



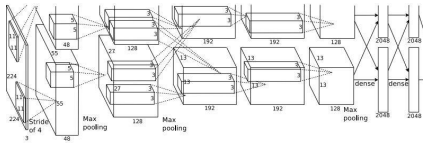
Classify center pixel with CNN



Cow



Cow

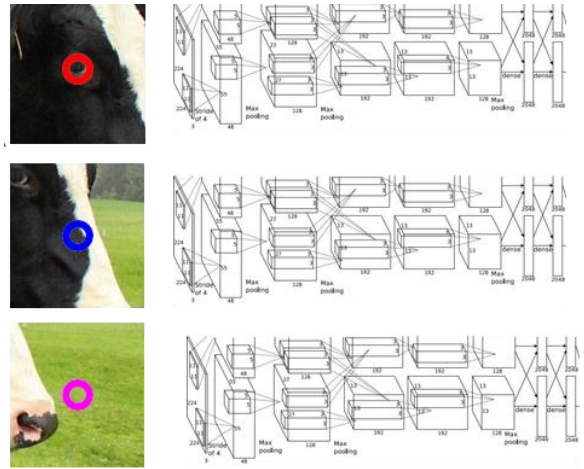


Grass

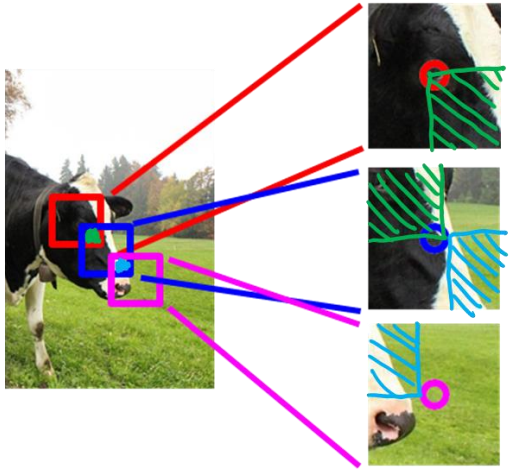
Input image를
작은 단위로 쪼갠다

분류 문제로 접근한다.

Semantic Segmentation Idea: Sliding Window



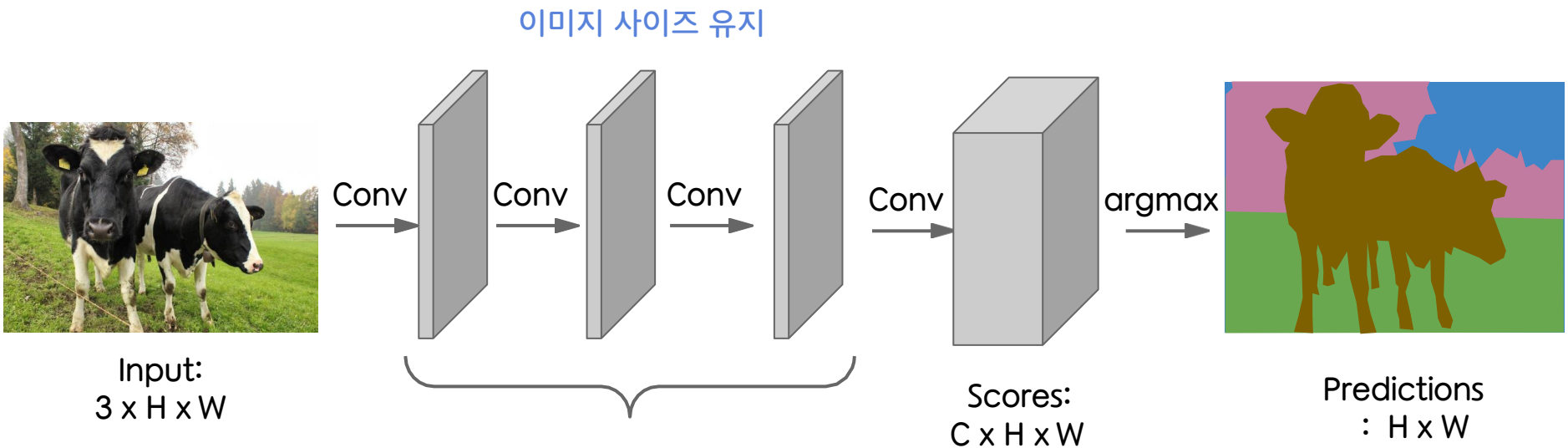
Image를 잘게 쪼개서 CNN에 넣음



공유되는 특성을 이용하지 않기때문에
비효율적임

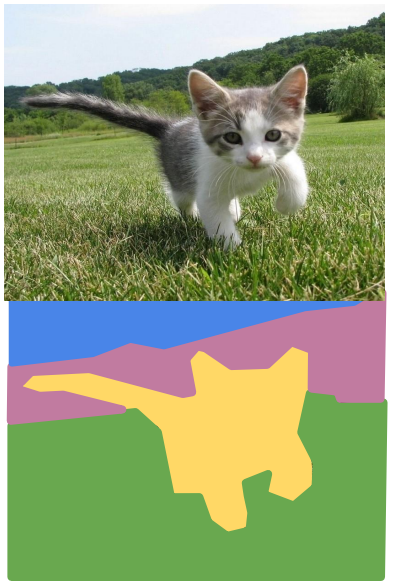
→ 비용이 매우 크다

Semantic Segmentation Idea: Fully Convolutional



- C: 예측해야하는 카테고리의 수
- Output tensor: input 이미지의 모든 픽셀간 classification score

Semantic Segmentation Idea: Fully Convolutional



학습 시킬 때

- Train data는 픽셀들이 어떤 class에 속하는지 알아야한다.
- Class가 정해져 있어야 한다.

Semantic Segmentation Idea: Fully Convolutional

output

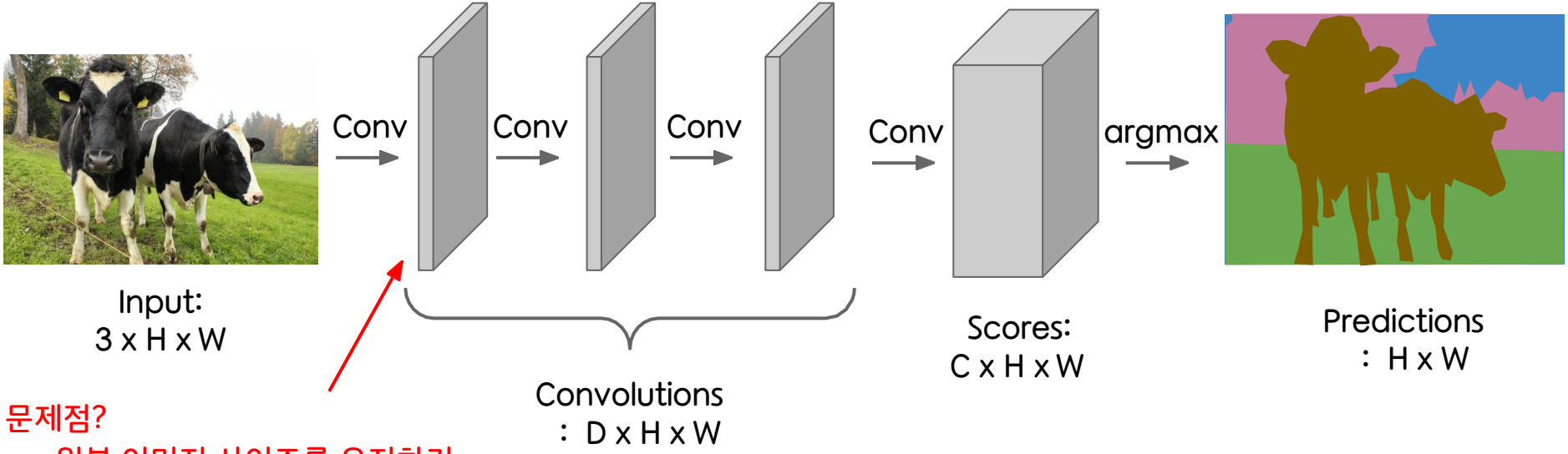
실제 정답

네트워크 학습 방법

- 모든 output 픽셀간 classification loss를 계산한다.
- Backpropagation을 수행한다.

- Loss의 경우 output의 모든 픽셀에 Cross Entropy를 적용하여 이 값들을 모두 혹은 mini-batch 단위로 더하거나 평균 내어 계산한다.

Semantic Segmentation Idea: Fully Convolutional



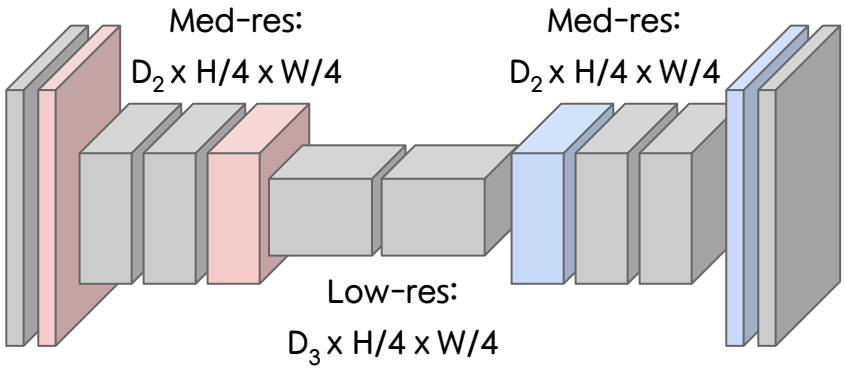
문제점?

- 원본 이미지 사이즈를 유지하기 때문에 비용이 많이 든다.

Semantic Segmentation Idea: Fully Convolutional

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Input:
3 x H x W



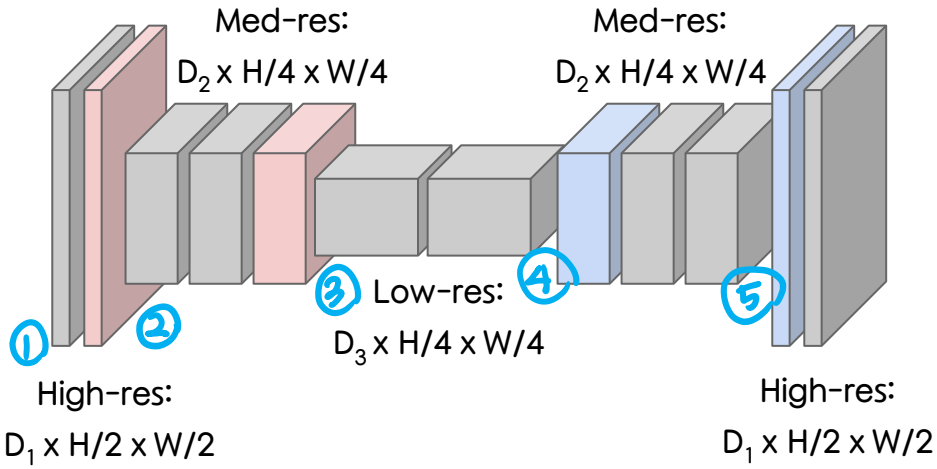
Predictions
: H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Semantic Segmentation Idea: Fully Convolutional

Design network as a bunch of convolutional layers, with downsampling and upsampling inside the network!

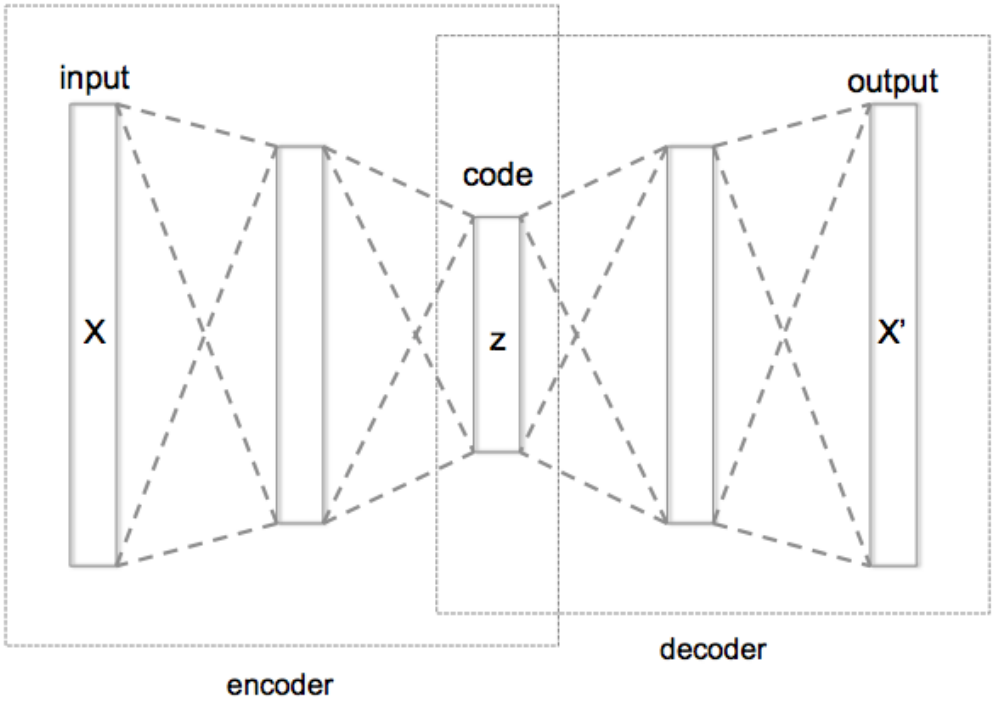
Input:
3 x H x W



Predictions
: H x W

- 2,3: downsampling
- 4,5: upsampling

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

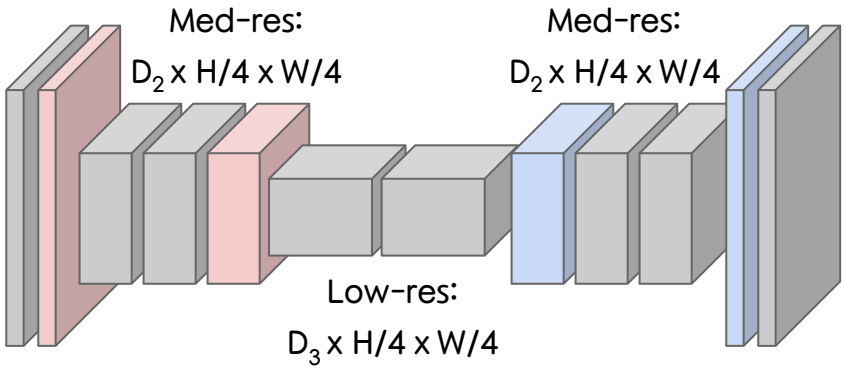


출처: https://www.datacamp.com/community/tutorials/autoencoder-keras-tutorial

Semantic Segmentation Idea: Fully Convolutional

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Input:
3 x H x W

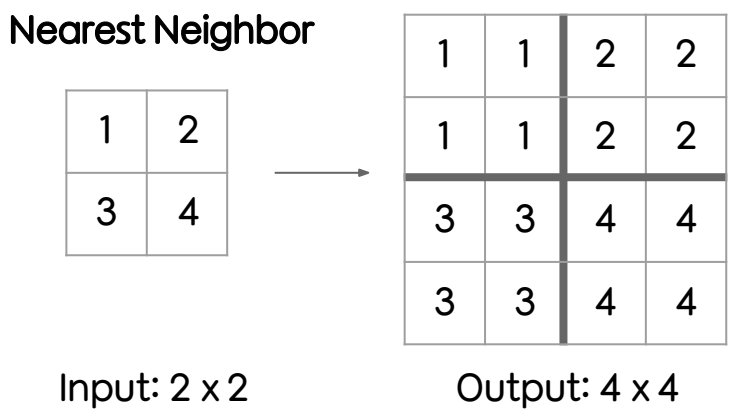


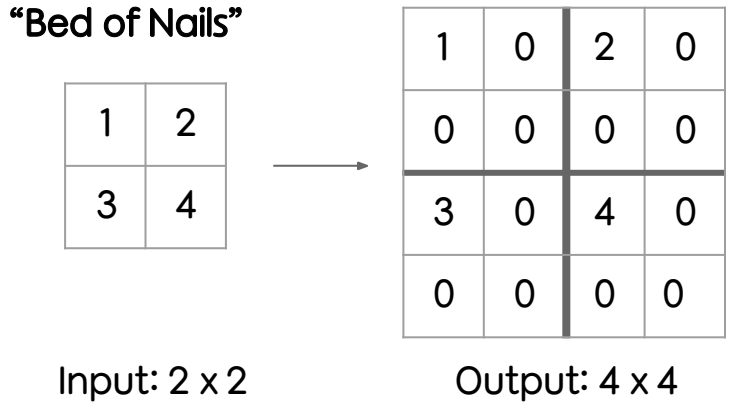
Predictions
: H x W

Downsampling
Pooling, strided convolution

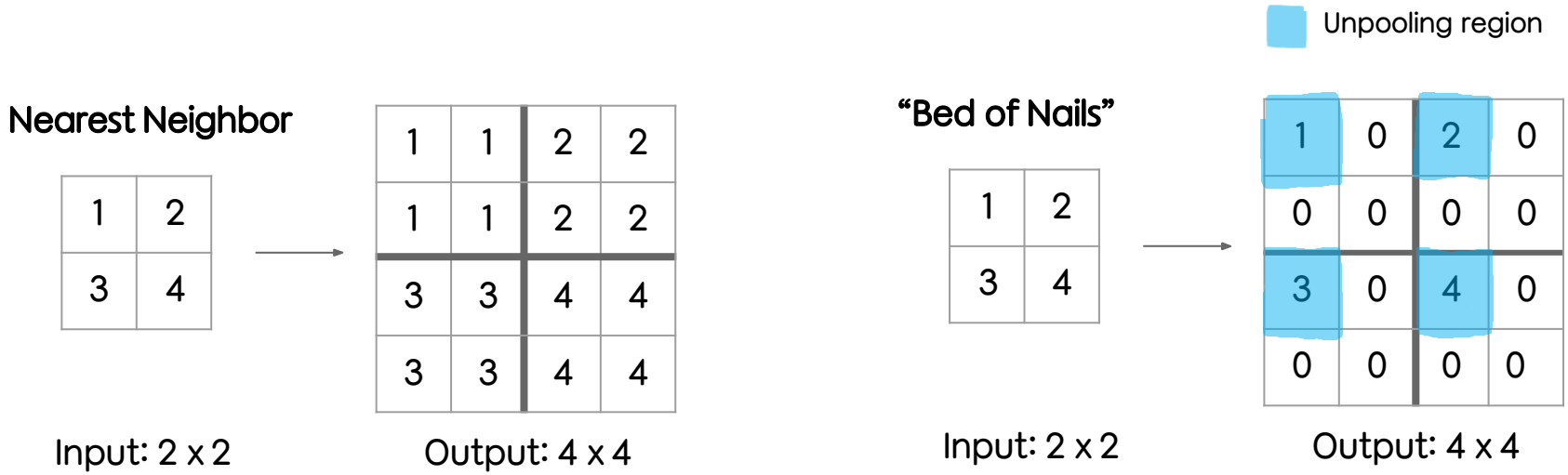
Upsampling:
???

In-Network upsampling: "Unpooling"





In-Network upsampling: "Unpooling"



In-Network upsampling: "Max Unpooling"

Max Pooling
Max 요소가 어디에 있었는지 기억한다.

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

Input: 4 x 4

5	6
7	8

Output: 2 x 2

Rest of the network

Max Unpooling
기억해 놓았던 위치에 넣고,
나머지 0으로 채운다.

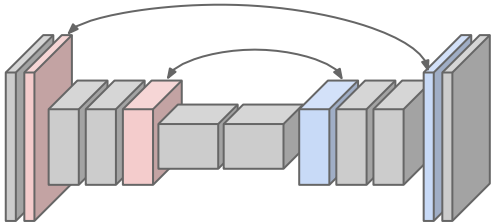
1	2
3	4

Input: 2 x 2

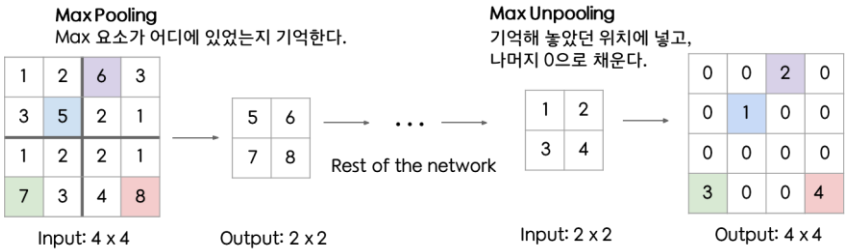
0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

Output: 4 x 4

대부분 network에서
downsampling과
upsampling이 대칭적!!



In-Network upsampling: “Max Unpooling”



이 방법이 좋은 이유?

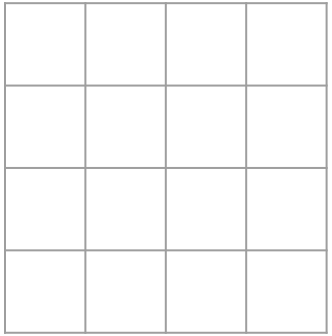
- Semantic Segmentation → object간의 경계를 디테일하게 잡아야함.
- 하지만 Max pooling시 이 data가 어디에서 왔는가를 모르면 공간 정보를 상실하게됨.

--> Max Unpooling 방법을 이용하면 공간 정보를 좀 더 상세하게 다룰 수 있다!

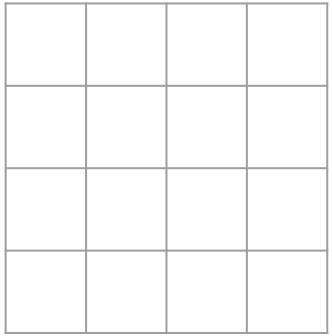
Learnable Upsampling: Transpose Convolution

복습> Stride Convolution

Recall: Typical 3 x 3 convolution, stride 1 pad 1



Input: 4 x 4

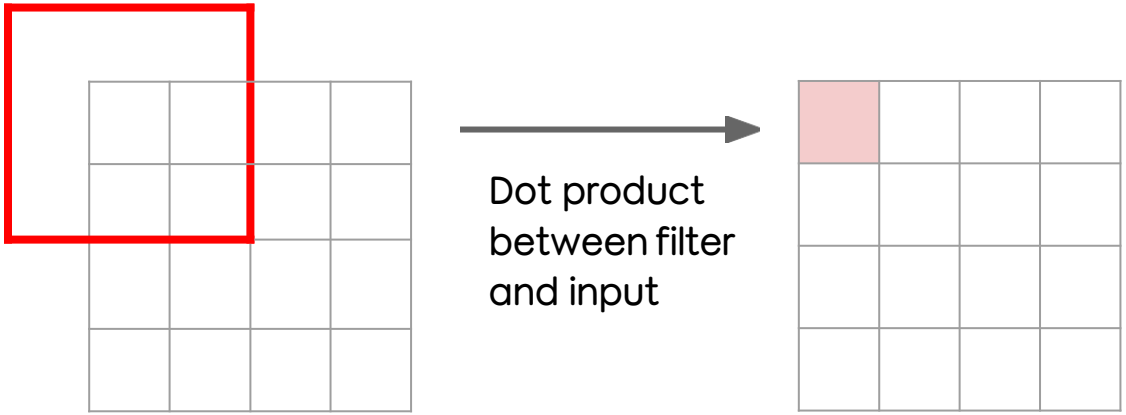


Output: 4 x 4

Learnable Upsampling: Transpose Convolution

복습> Stride Convolution

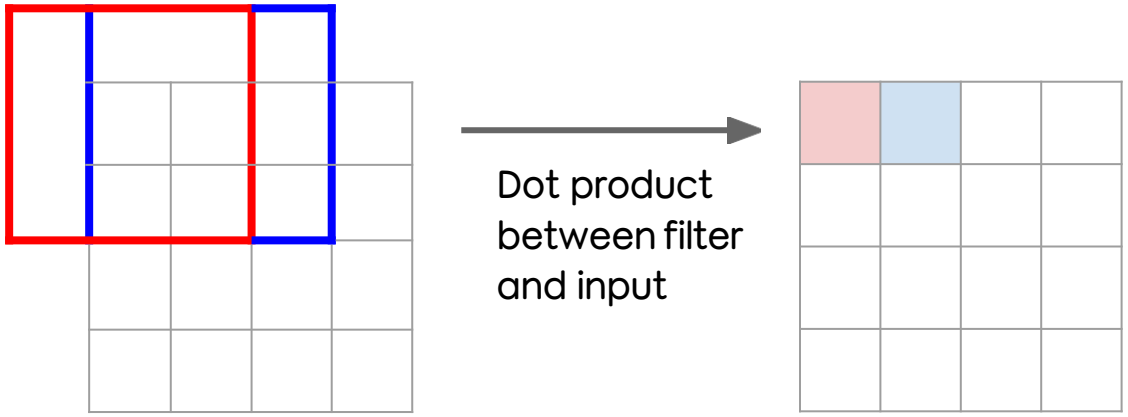
Recall: Normal 3 x 3 convolution, stride 1 pad 1



Learnable Upsampling: Transpose Convolution

복습> Stride Convolution

Recall: Normal 3 x 3 convolution, stride 1 pad 1



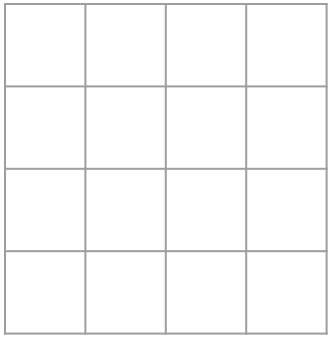
Input: 4 x 4

Output: 4 x 4

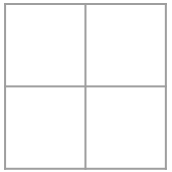
Learnable Upsampling: Transpose Convolution

복습> Stride Convolution

Recall: Normal 3 x 3 convolution, stride 2 pad 1



Input: 4 x 4

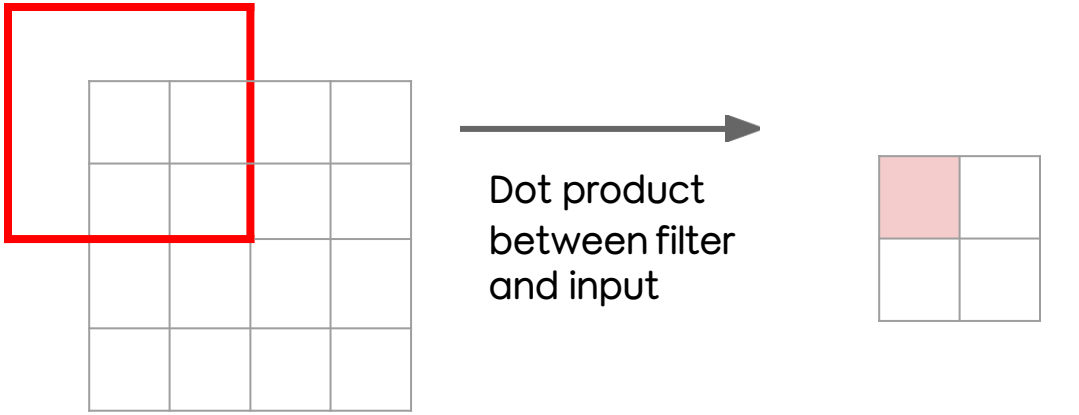


Output: 2 x 2

Learnable Upsampling: Transpose Convolution

복습> Stride Convolution

Recall: Normal 3 x 3 convolution, stride 2 pad 1



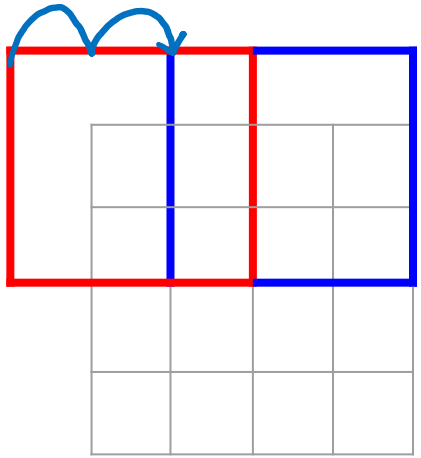
Input: 4 x 4

Output: 2 x 2

Learnable Upsampling: Transpose Convolution

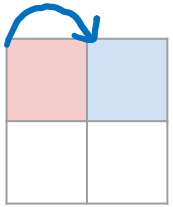
복습> Stride Convolution

Recall: Normal 3 x 3 convolution, stride 2 pad 1



Input: 4 x 4

Dot product between filter and input



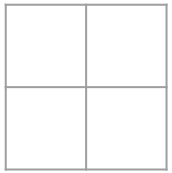
Output: 2 x 2

Stride 2=
input에서 움직이는 거리
/output에서 움직이는 거리

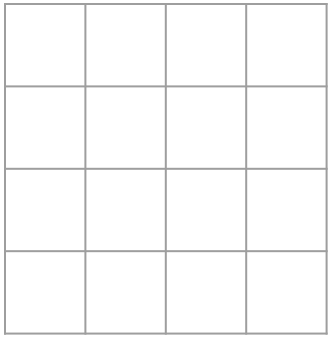
2/1=2

Learnable Upsampling: Transpose Convolution

3 x 3 **transpose** convolution, stride 2 pad 1



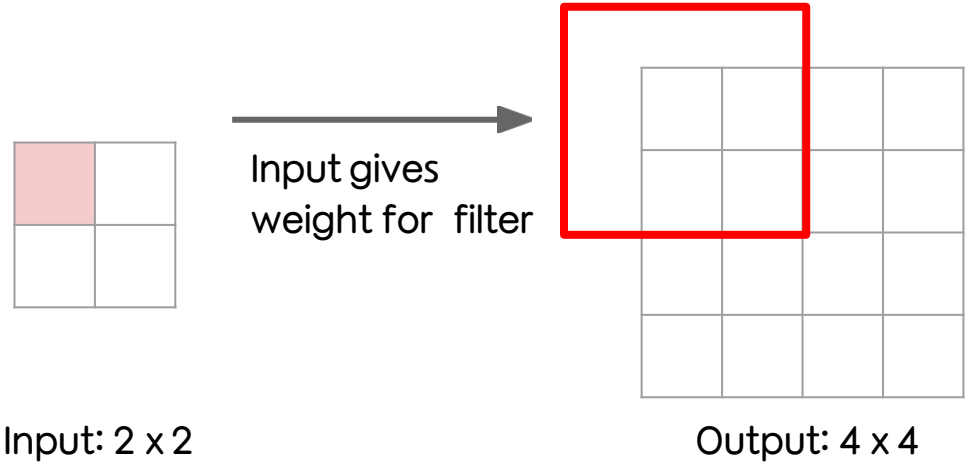
Input: 2 x 2



Output: 4 x 4

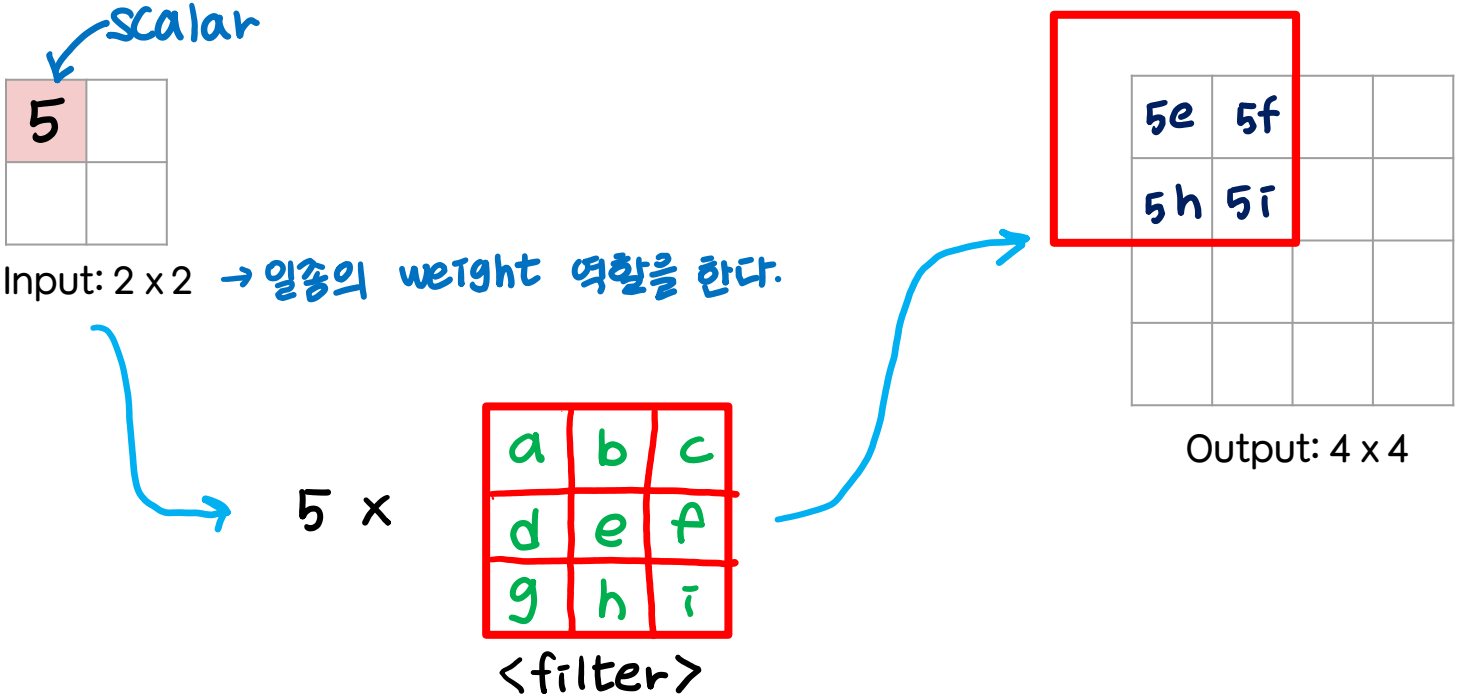
Learnable Upsampling: Transpose Convolution

3 x 3 **transpose** convolution, stride 2 pad 1

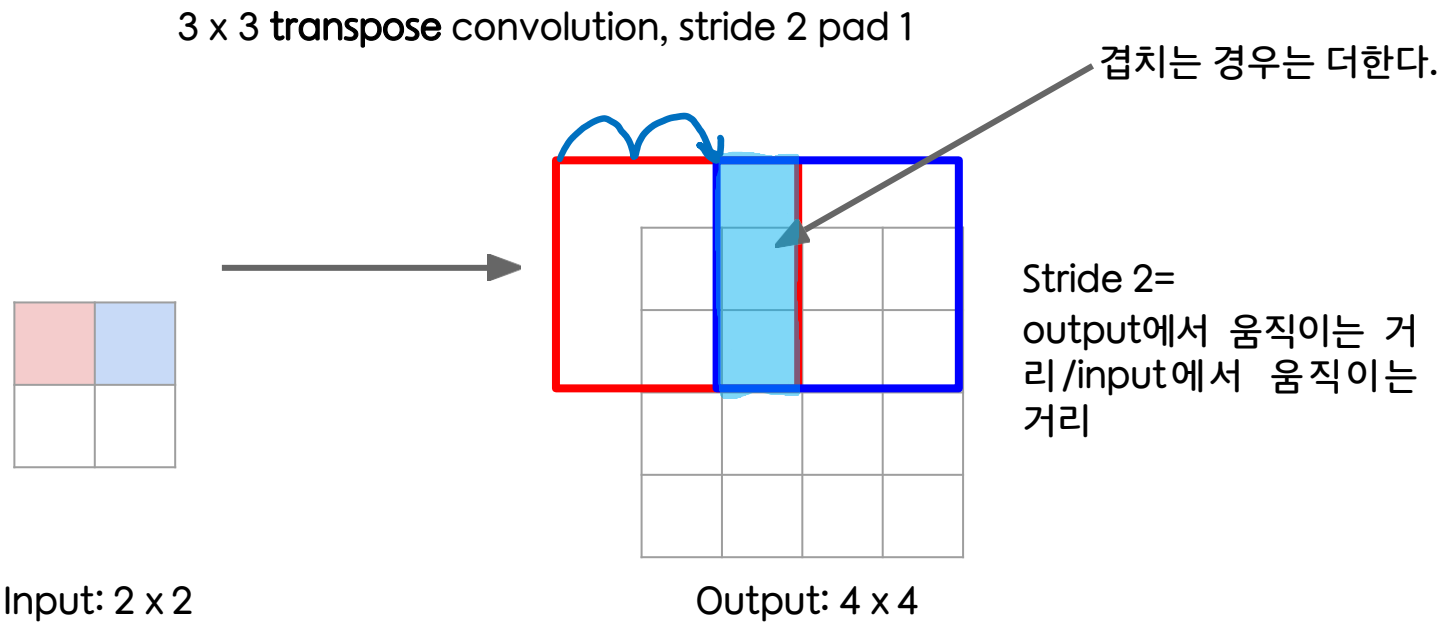


Learnable Upsampling: Transpose Convolution

3 x 3 **transpose** convolution, stride 2 pad 1



Learnable Upsampling: Transpose Convolution

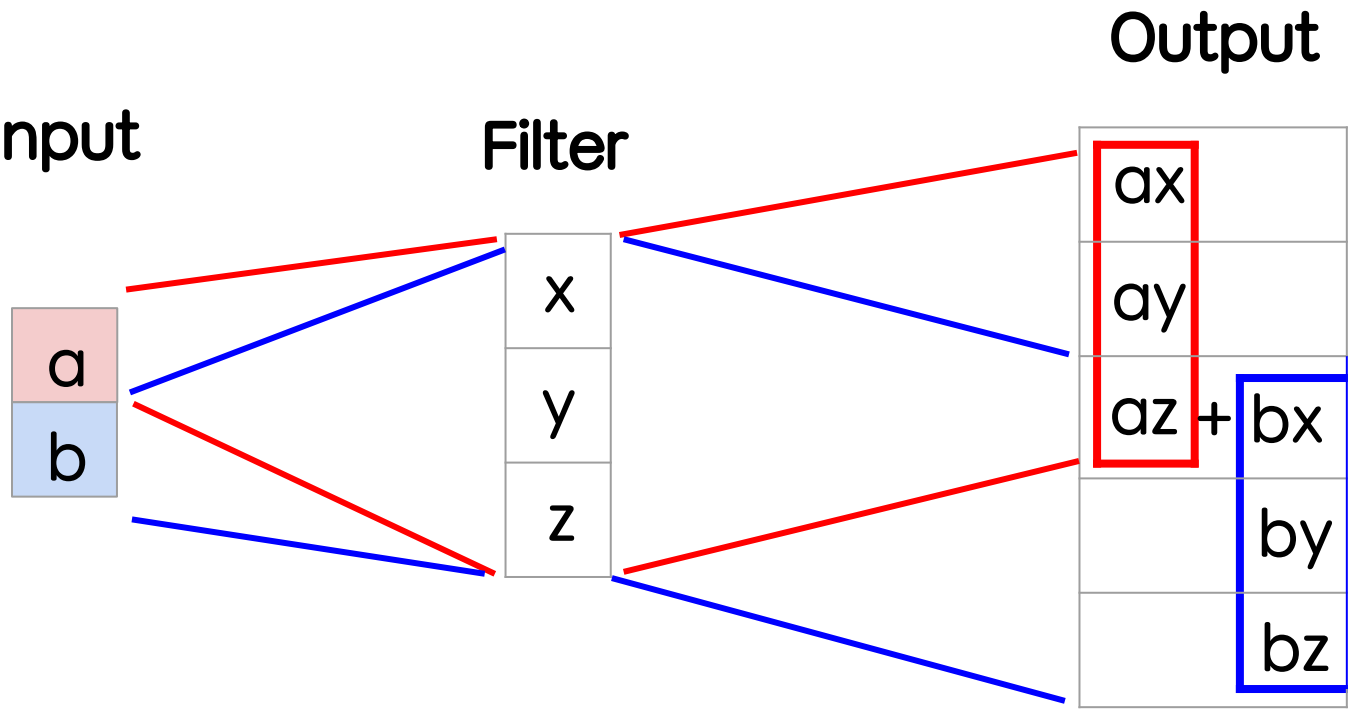


Learnable Upsampling: Transpose Convolution

Transpose Convolution의 다른 이름

- Deconvolution (bad)
  - “Deconvolution”이라는 뜻 자체가 convolution 연산의 역 연산 이라는 뜻인데 실제로 transpose convolution은 convolution 연산의 역 연산이 아니라서 안좋은 이름임
- Upconvolution
- Fractionally strided convolution
- Backward strided convolution

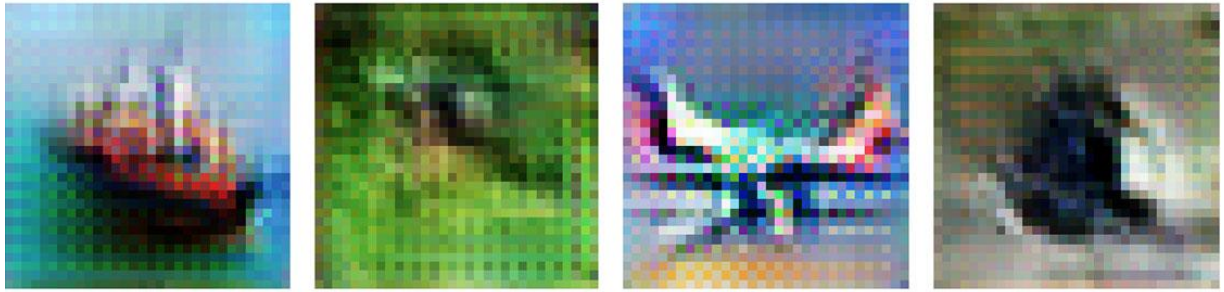
Transpose Convolution: 1D Example



< 겹치는 부분: 더한다

Learnable Upsampling: Transpose Convolution

최근 checkerboard 현상에 대한 문제가 많이 제기됨



→ 최근 논문에서는 4x4 filter, stride 2 사용

→ 또는 2x2 filter, stride 2 사용

https://distill.pub/2016/deconv-checkerboard/

Convolution as Matrix Multiplication (1D Example)

Convolution 연산은
행렬곱 연산으로 나타낼 수 있다.

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

벡터 x: 원소 3개

X: 컨볼루션 커널, 가중치 행렬

벡터 a: input vector, 원소 4개(a~d)

Example: 1D conv,
kernel size=3, stride=1, padding=1

Convolution as Matrix Multiplication (1D Example)

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv,
kernel size=3, stride=1, padding=1

Convolution transpose multiplies by t
he transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T\vec{a}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

Convolution as Matrix Multiplication (1D Example)

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv,
kernel size=3, stride=1, padding=1

$$\vec{x} *^T \vec{a} = X^T\vec{a}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

x, y, z는 훈련을 통해서 정해지는 값

Convolution as Matrix Multiplication (1D Example)

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv
kernel size=3, stride=2, padding=1

Convolution as Matrix Multiplication (1D Example)

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

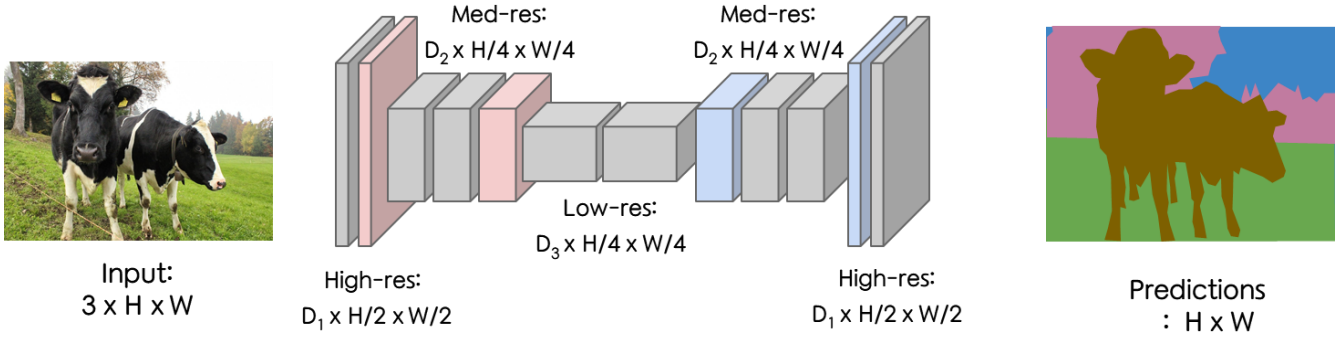
Example: 1D conv
kernel size=3, stride=2, padding=1

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

Stride>1인 경우 더 이상 같은 연산이라고 할 수 없다.

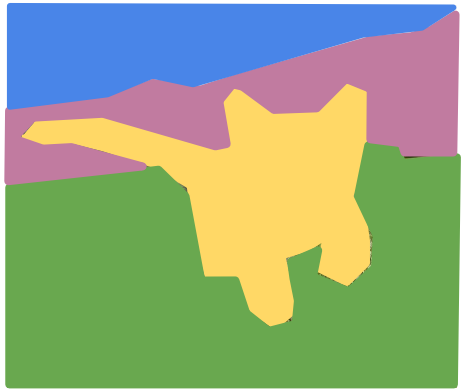
Semantic Segmentation Idea: Fully Convolutional



Fully Convolutional 정리>

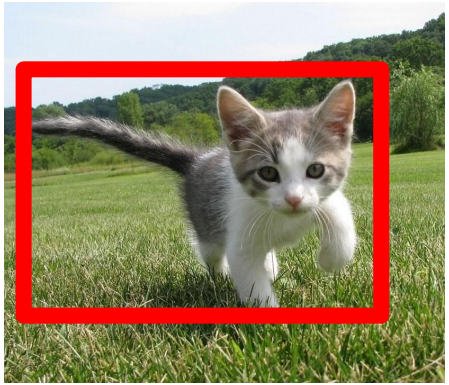
- 위의 사진이 일반적 구조
- 네트워크 내부에 downsampling과 upsampling하는 네트워크 존재
  - Downsampling: pooling, strided convolution
  - Upsampling: unpooling, strided transpose convolution
- 비용함수로는 cross entropy

Classification + Localization



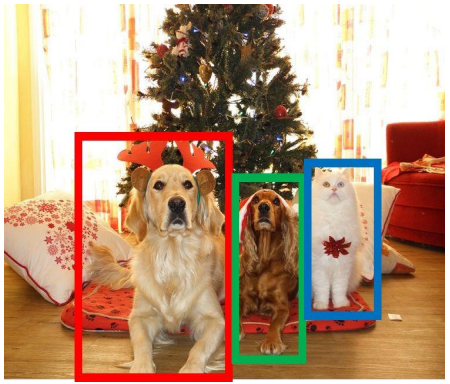
GRASS, CAT,
TREE, SKY

No objects, just pixels



CAT

Single Object

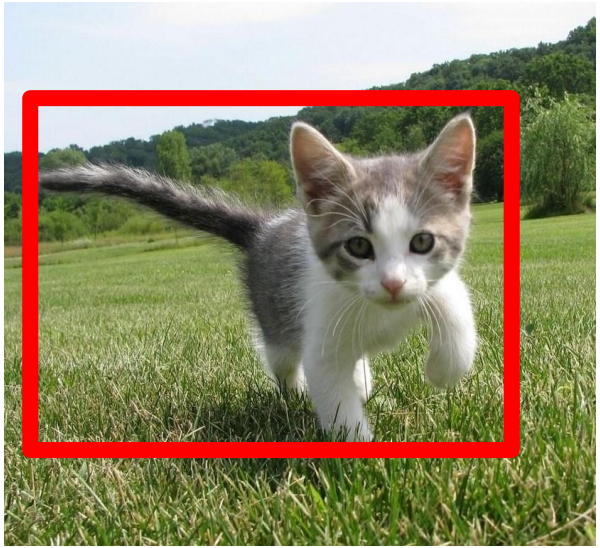


DOG, DOG, CAT

DOG, DOG, CAT

Multiple Object

Classification + Localization

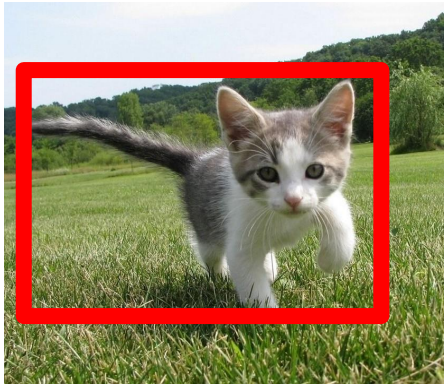


CAT

분류: 고양이이다.
위치 찾기:  에 있다.

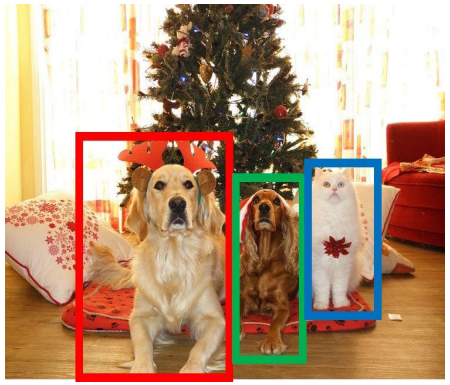
Classification + Localization

Classification
+ Localization



CAT

Object
Detection

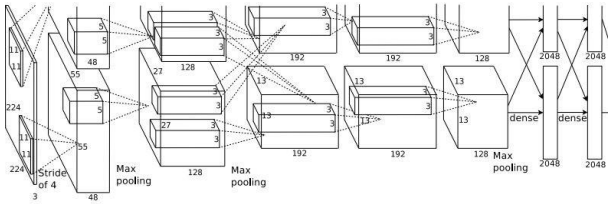


DOG, DOG, CAT

→ 내가 찾고자 하는 객체 “하나만” 찾는다는 점에서 object detection과 다르다.

Classification + Localization

This image is CC0 public domain



Fully Connected: 4096 to 1000 ①

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Vector: 4096

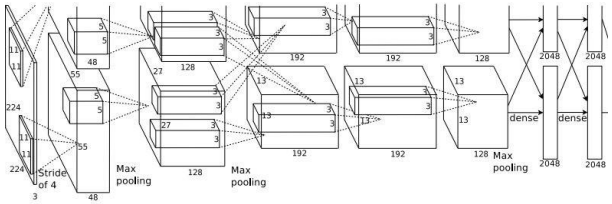
Fully Connected: 4096 to 4 ②

Box Coordinates
(x, y, w, h)

1. Class score → 1000개중 카테고리를 정해준다.
2. Bounding box 위치 결정(좌표)

Classification + Localization

This image is CC0 public domain



Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Correct label: Cat

Class score 예측

Vector: 4096

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

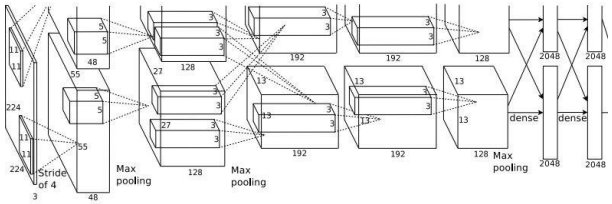
L2 Loss

Correct box: (x', y', w', h')

실제와 예측된 bounding box 사이의 차이 계산

Classification + Localization

This image is CC0 public domain



Correct label:
Cat

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Fully Connected:
4096 to 1000

Softmax Loss

Multitask Loss
: 2개의 loss를 합친 것

+ → Loss

Vector:
4096

Fully Connected:
4096 to 4

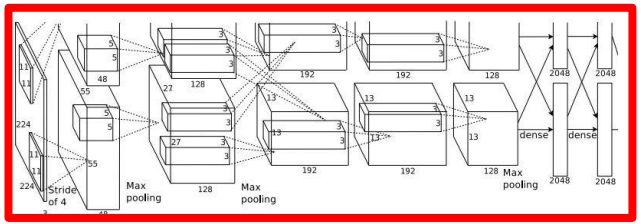
Box Coordinates
(x, y, w, h)

L2 Loss

Correct box:
(x', y', w', h')

- 합칠 때 가중치를 잘 설정해야한다.
- 가중치는 hyperparameter

Classification + Localization



Often pretrained on Image Net (Transfer learning)

Vector: 4096

Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Correct label: Cat

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

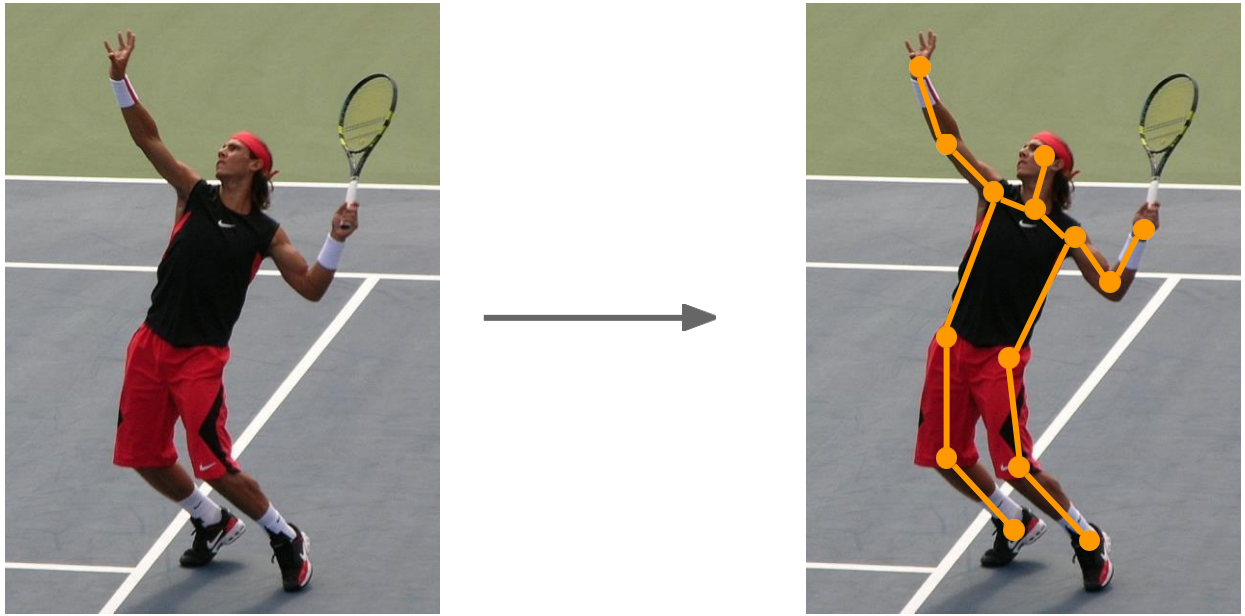
L2 Loss

Correct box: (x', y', w', h')

Multitask Loss

+ → Loss

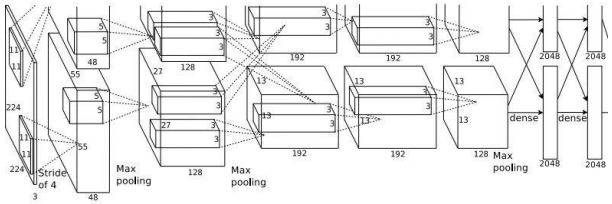
Aside: Human Pose Estimation



사람 관절 위치 예측 통한 pose 정의

Input: 사람 이미지
Output: 관절의 위치
→ 14개의 관절로 포즈를 정의

Johnson and Everingham, "Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation", BMVC 2010



Vector: 4096

Left foot: (x, y)

Right foot: (x, y)

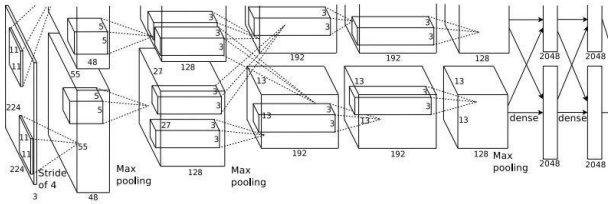
…

Head top: (x, y)

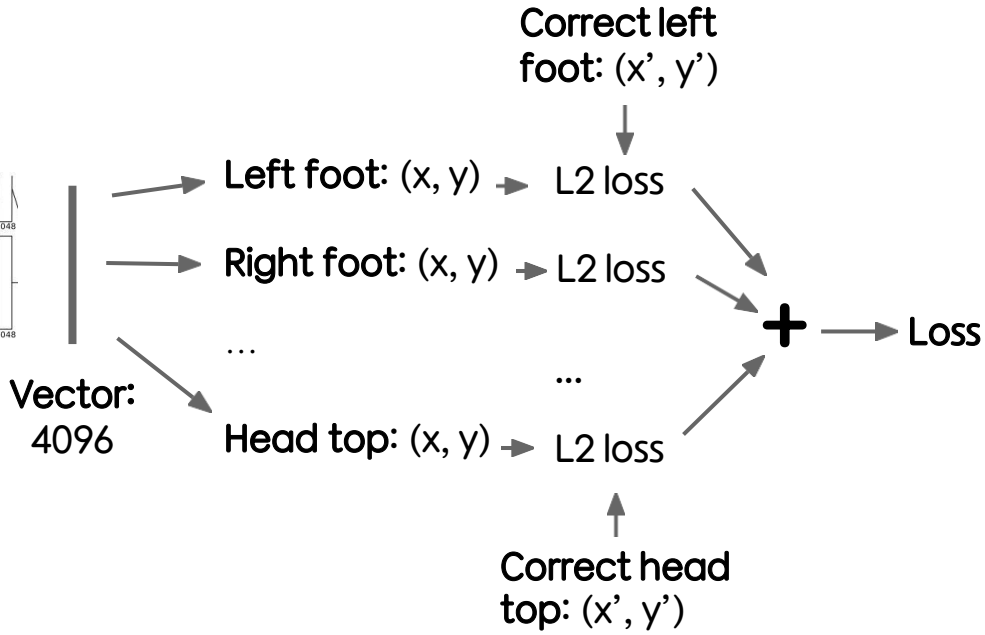
Toshev and Szegedy, “DeepPose: Human Pose Estimation via Deep Neural Networks”, CVPR 2014

결과는 좌표 값으로 나옴

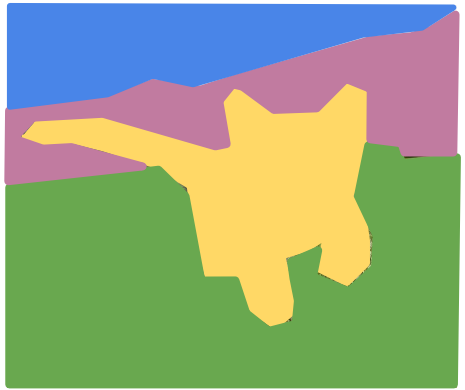
Aside: Human Pose Estimation



Toshev and Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks", CVPR 2014

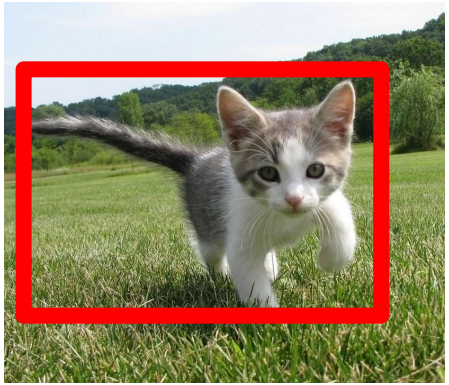


Object Detection



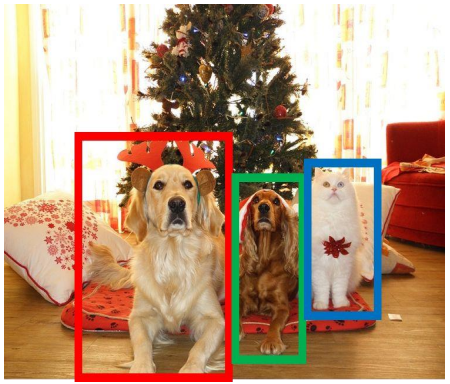
GRASS, CAT,
TREE, SKY

No objects, just pixels



CAT

Single Object

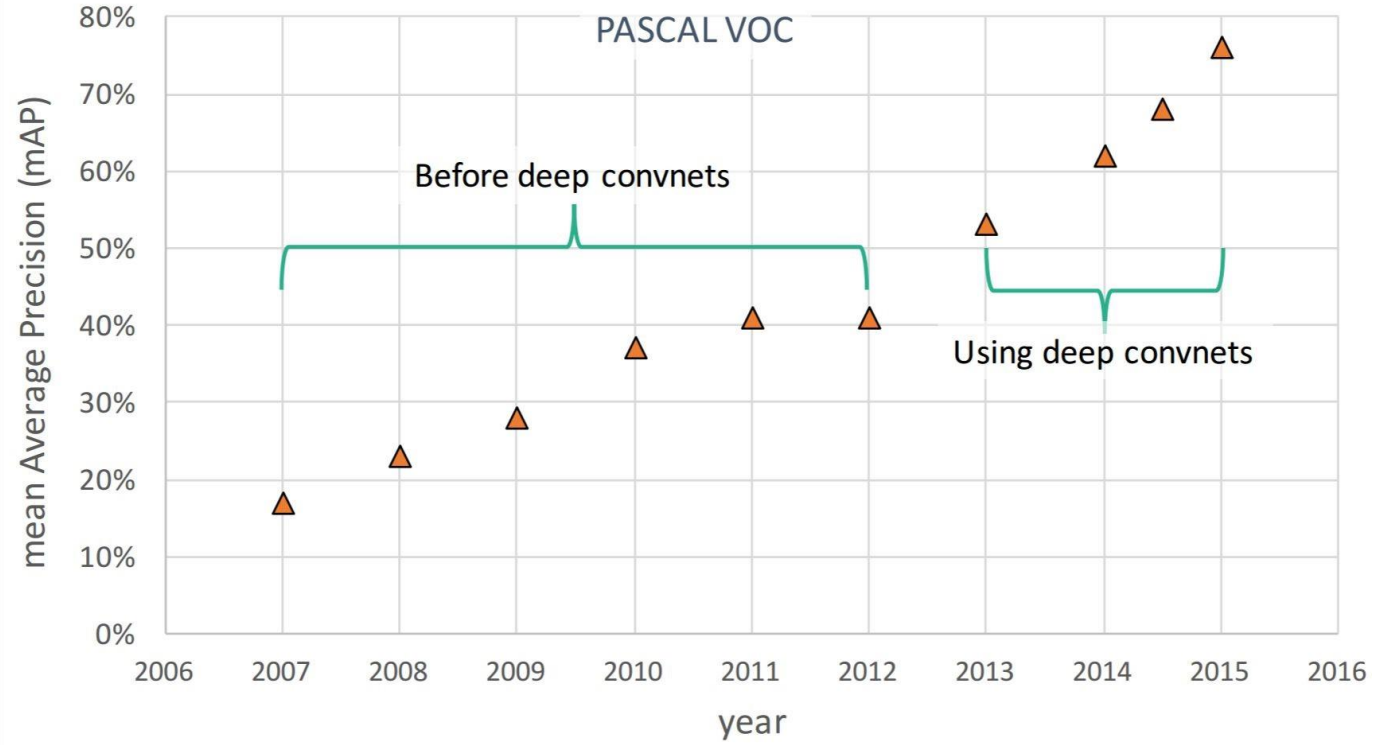


DOG, DOG, CAT

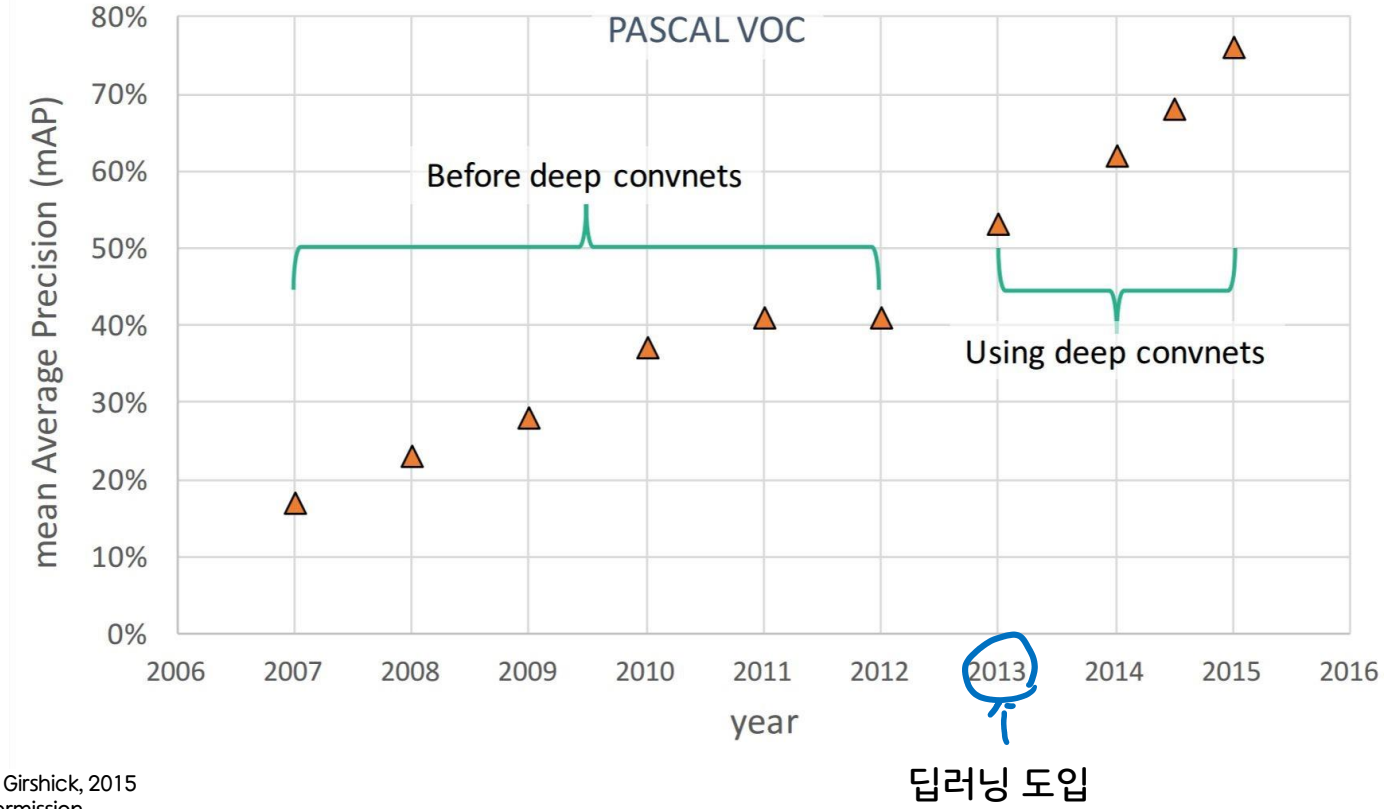
DOG, DOG, CAT

Multiple Object

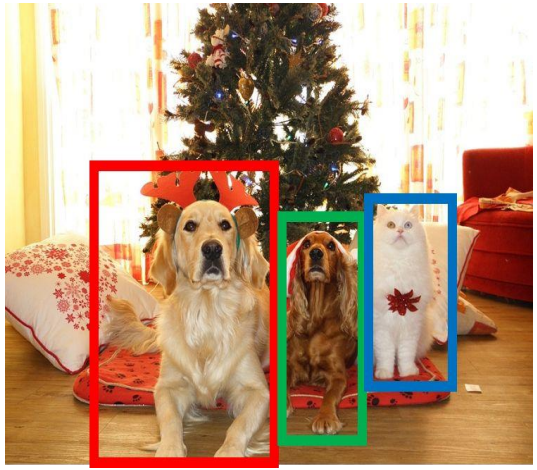
Object Detection: Impact of Deep Learning



Object Detection: Impact of Deep Learning



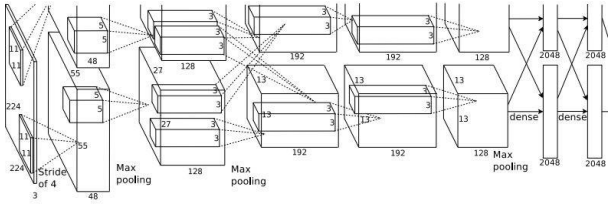
Object Detection



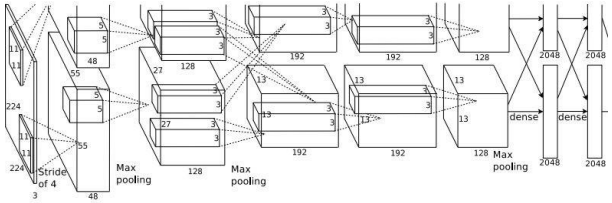
DOG, DOG, CAT

- 다룰 카테고리가 정해져있다.
- Input image에 bounding box를 그리고 카테고리를 예측
- Classification+localization과 다르다
  - Image에 따라서 bounding box의 개수가 다르다(몇 개 있는지 모름)

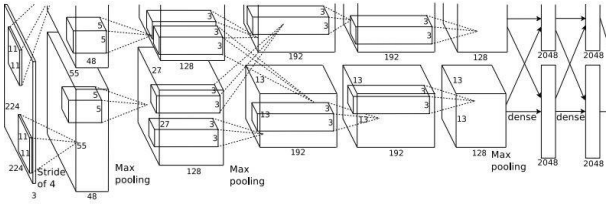
Object Detection



CAT: (x, y, w, h)

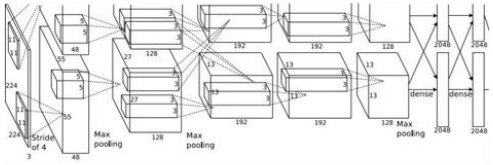


DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

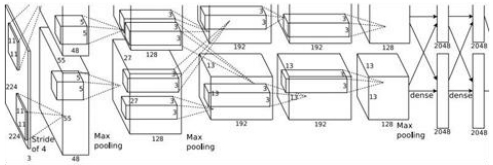


DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
….

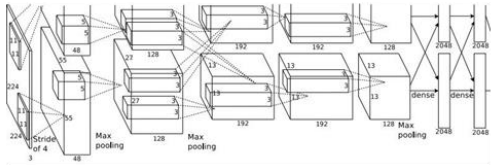
Object Detection



CAT: (x, y, w, h)



DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

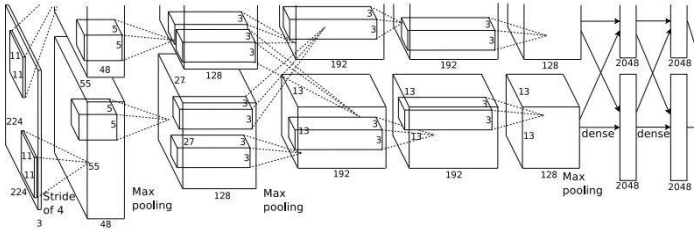


DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
….

"이미지마다 객체 수가 다르기 때문에 어렵다"

Object Detection as Classification: Sliding Window

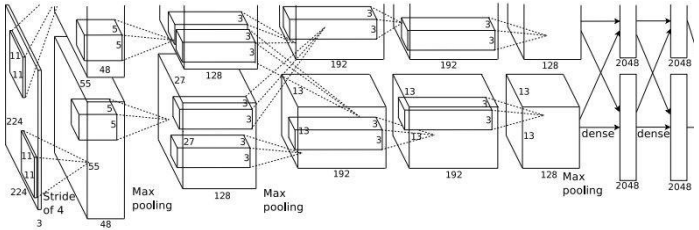
: 영역을 나눠서 그 영역을 CNN에 넣는 방법



Dog? NO
Cat? NO
Background? YES

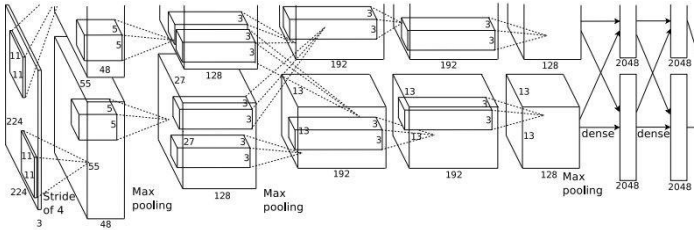
Background라는 카테고리 추가 ->
찾고자 하는 것(개, 고양이)이 없을 경우 배경으로 분류

Object Detection as Classification: Sliding Window

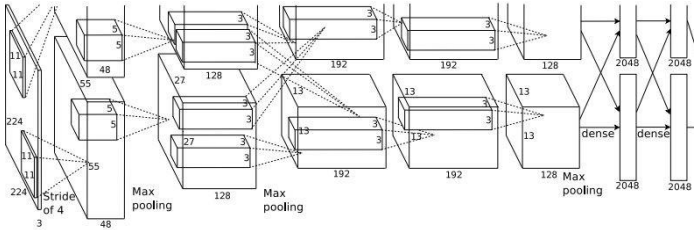


Dog? YES
Cat? NO
Background? NO

Object Detection as Classification: Sliding Window

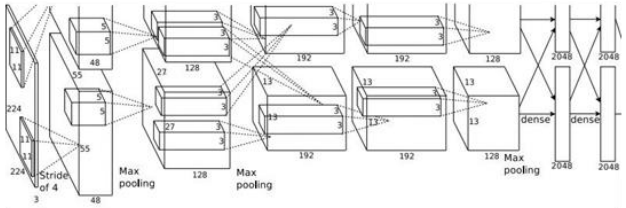


Object Detection as Classification: Sliding Window



Dog? NO
Cat? YES
Background? NO

Object Detection as Classification: Sliding Window



Dog? NO
Cat? YES
Background? NO

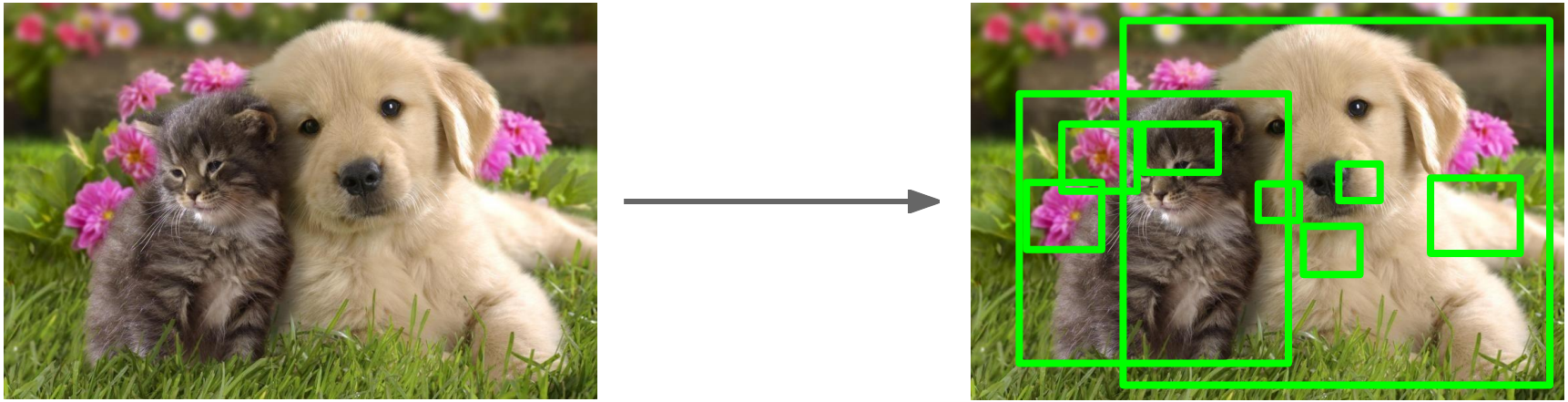
Bounding box 그리는 영역 추출 문제 존재!

- Object 몇 개?
- 어디에 box 그릴건지?
- Box의 size (크기, 비율 등)

→ 이걸 다 CNN에?

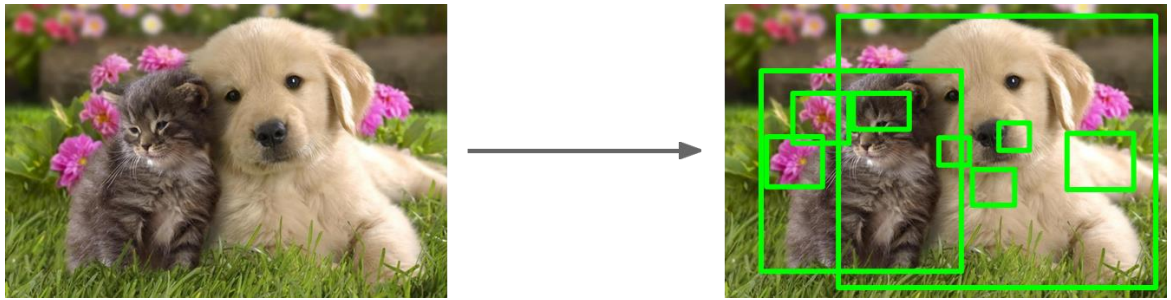
Region Proposals 딥러닝 사용X, 전통적인 신호처리를 사용하는 방법

- Object가 있을법한 1000개의 box를 제공



Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012
Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 201
4 Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

Region Proposals



Selective Search

- 2000개의 Region Proposal을 만듦
  - Noise가 심하다
  - 재현율이 높다

- 이 방법을 통해 얻은 영역을 CNN에 넣으면 Sliding window보다 계산량이 매우 줄어듦!

R-CNN

Input image

Girshick et al, “Rich feature hierarchies for accurate object detectio n and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permissio n.

R-CNN

- input 이미지에서 region proposal network를 통과해서 2000개의 RoI를 얻는다.

Region Proposal=Regions of Interest(RoI)

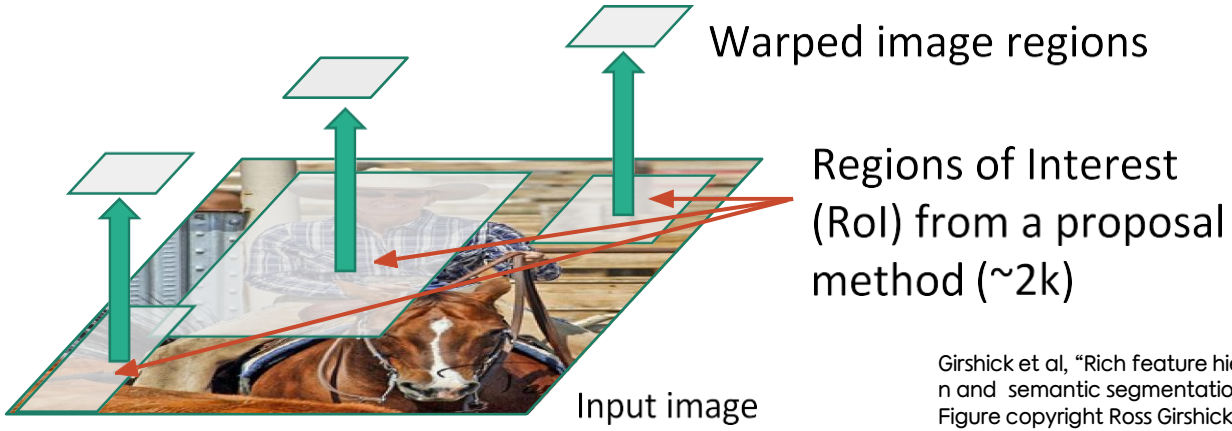
Input image

Regions of Interest (RoI) from a proposal method (~2k)

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN

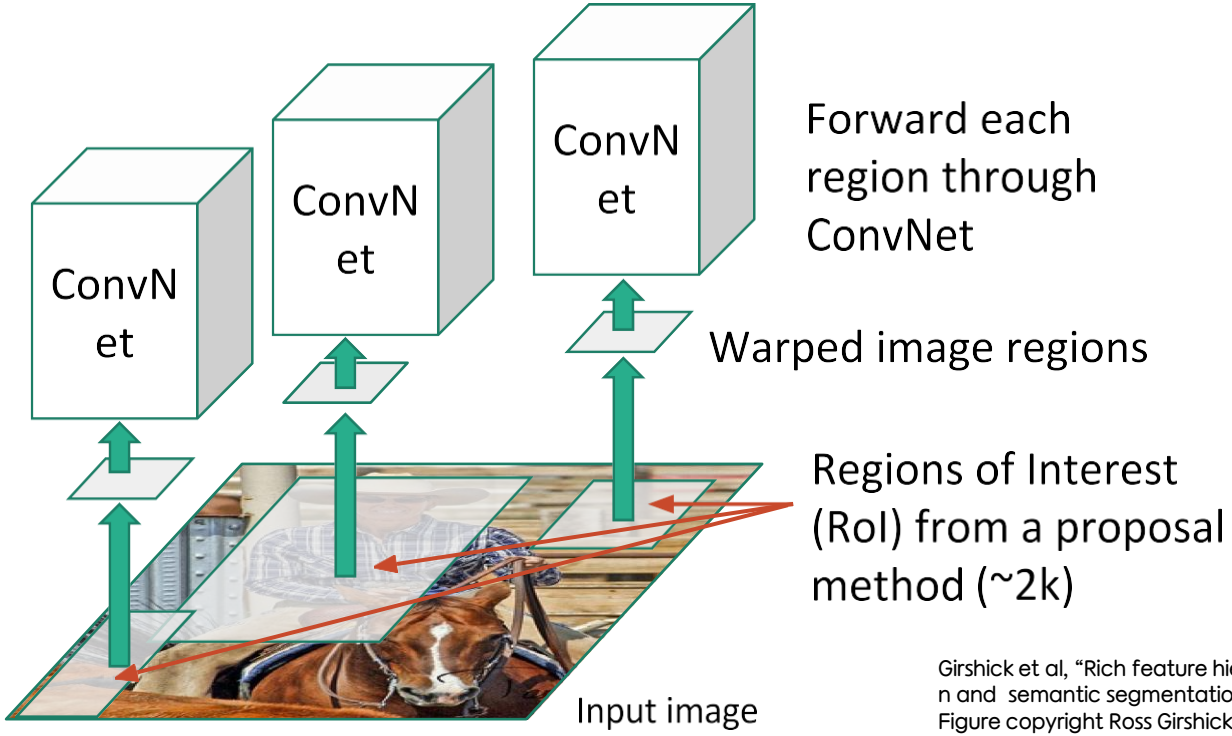
RoI 2000개 박스 사이즈가 다르다 → 고정된 크기로 바꿔준다.



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

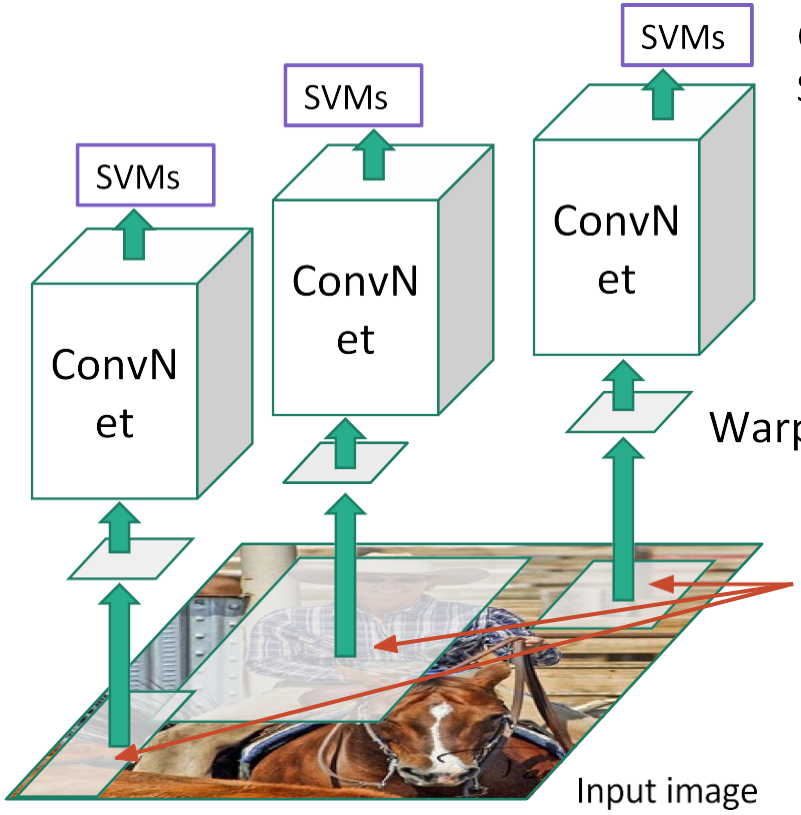
R-CNN

각각의 이미지를 CNN에 통과



Girshick et al, “Rich feature hierarchies for accurate object detectio
n and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permissio
n.

R-CNN

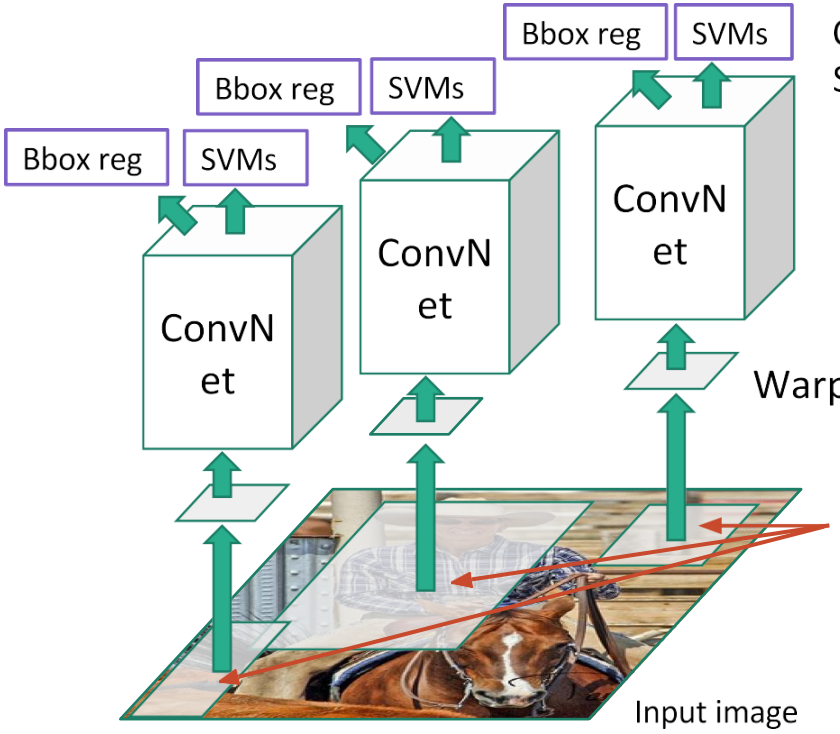


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN

카테고리 예측 말고도 bounding box의 값을 보정해주는 offset 4개를 제공

Linear Regression for bounding box offsets



Classify regions with SVMs

Forward each region through ConvNet

Warped image regions

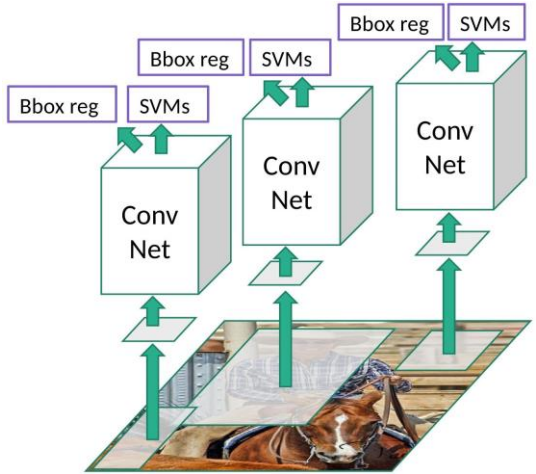
Regions of Interest (RoI) from a proposal method (~2k)

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN: Problems

- Training이 느리다.
  - 84시간 걸림
- Test도 느리다
  - 1장 이미지 test하는데 47초 걸림

- 계산 비용이 많이든다
  - 2000개의 Region Proposal들을 CNN에 넣어 feature 찾으므로..



Girshick et al, "Rich feature hierarchies for accurate object detectio
n and semantic segmentation", CVPR 2014.
Slide copyright Ross Girshick, 2015; source. Reproduced with permissio
n.

Fast R-CNN

Input image

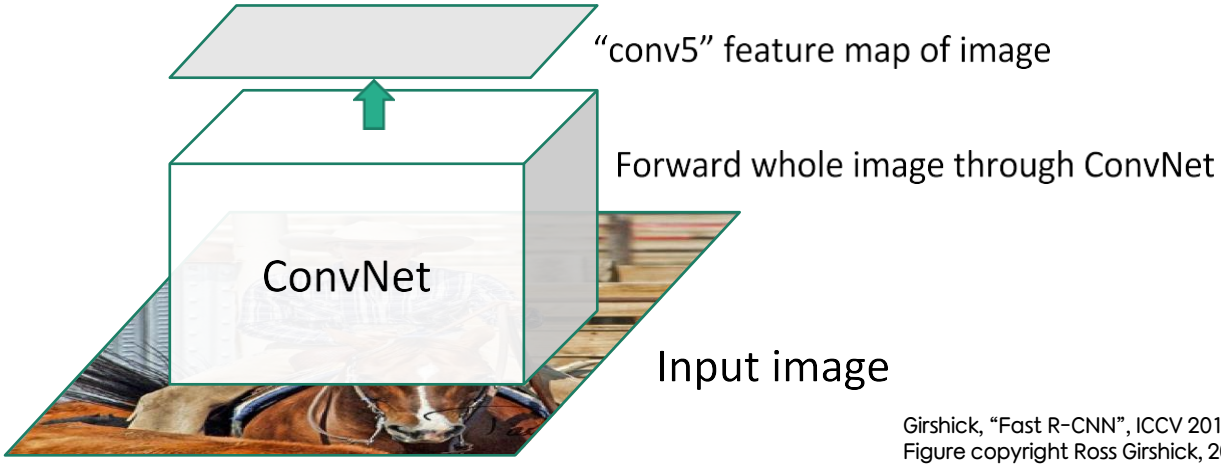
Girshick, “Fast R-CNN”, ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN

R-CNN과의 차이점!

- 전체 이미지에 CNN을 적용한다.

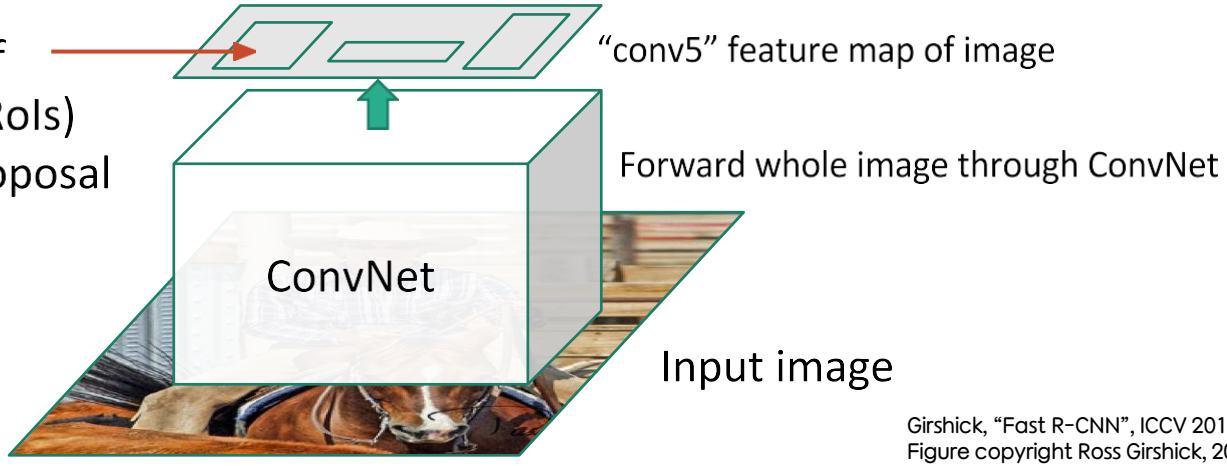
(RoI마다 CNN을 적용하지 않는다.)



Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN

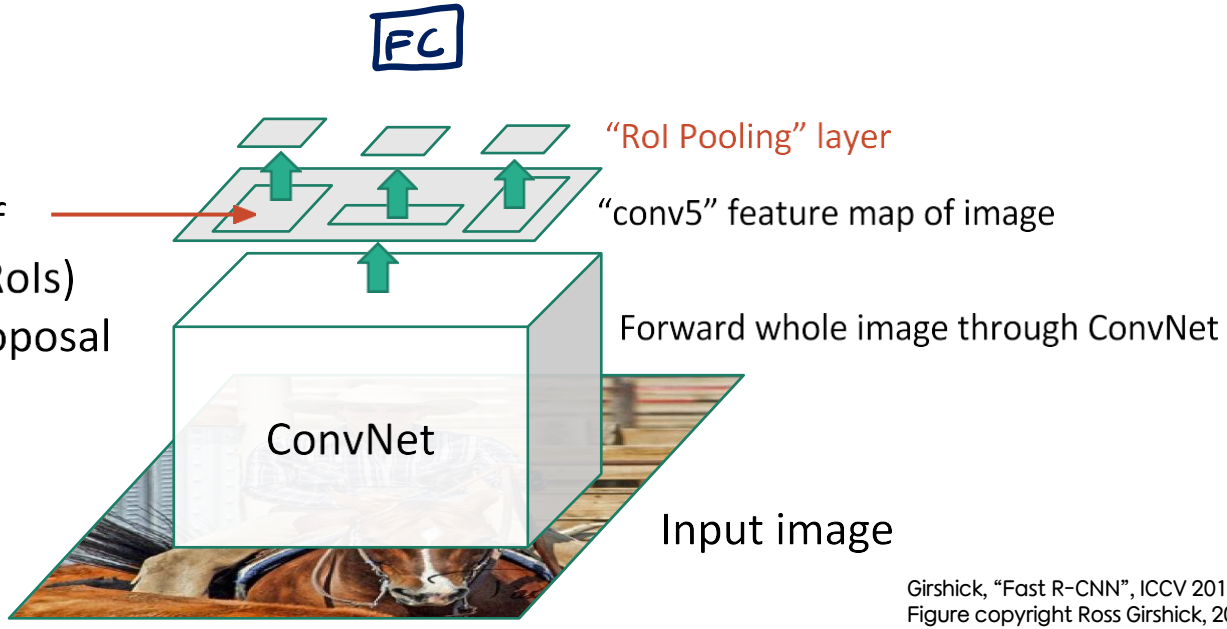
CNN을 통과해서 얻은 feature map에서 RoI를 찾는다.
- 아까는 이미지에서 RoI찾았지만 이번엔 feature map에서 찾는다



Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

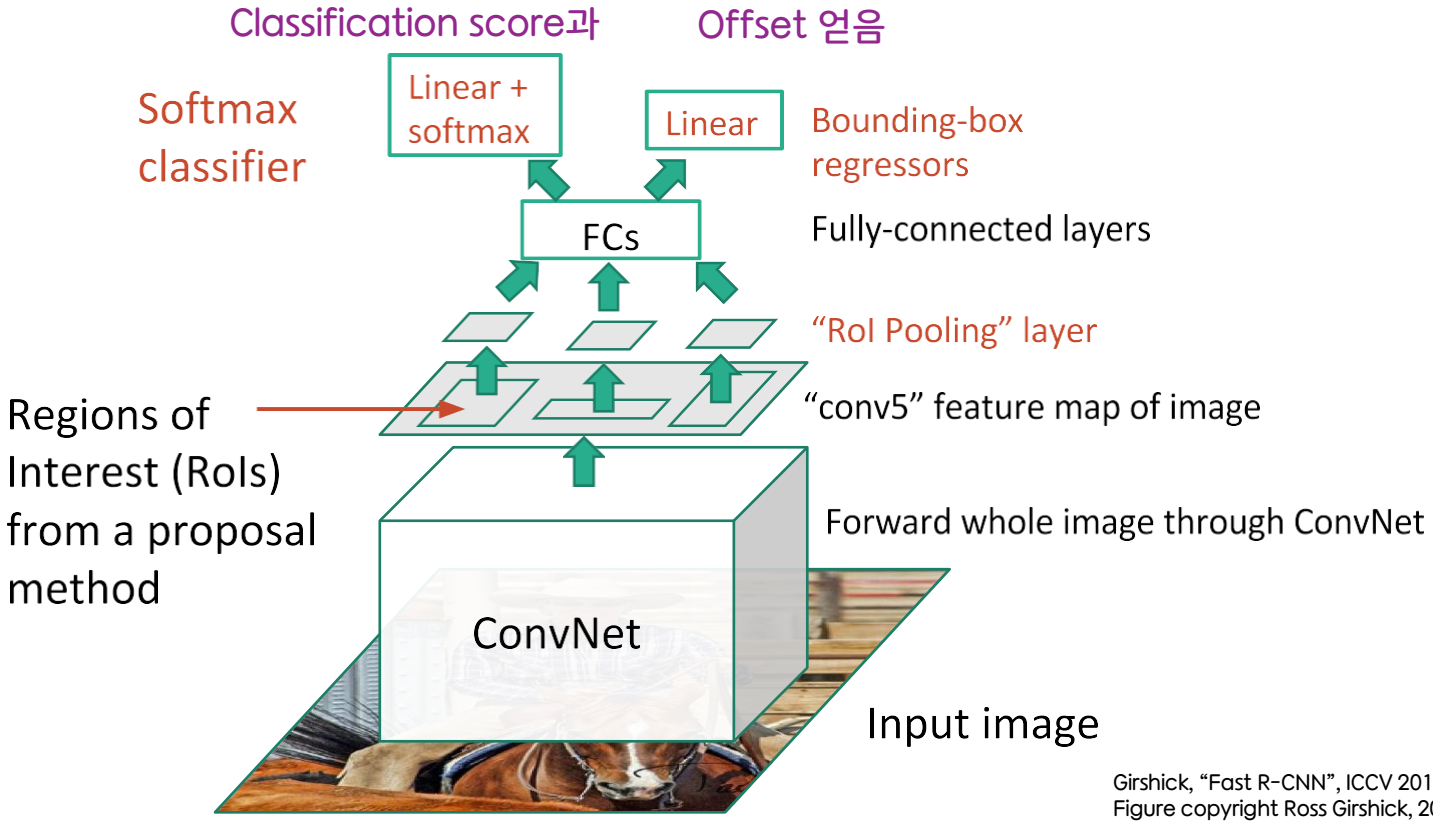
Fast R-CNN

FC는 고정된 size의 input을 받기 때문에 size를 맞춰준다.



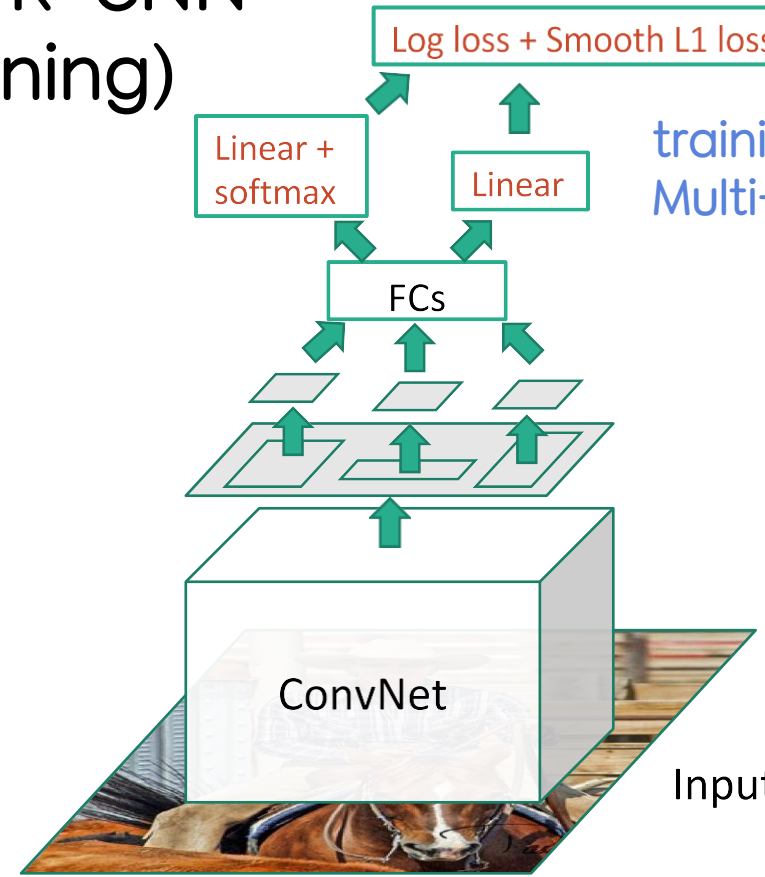
Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN



Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permissio n.

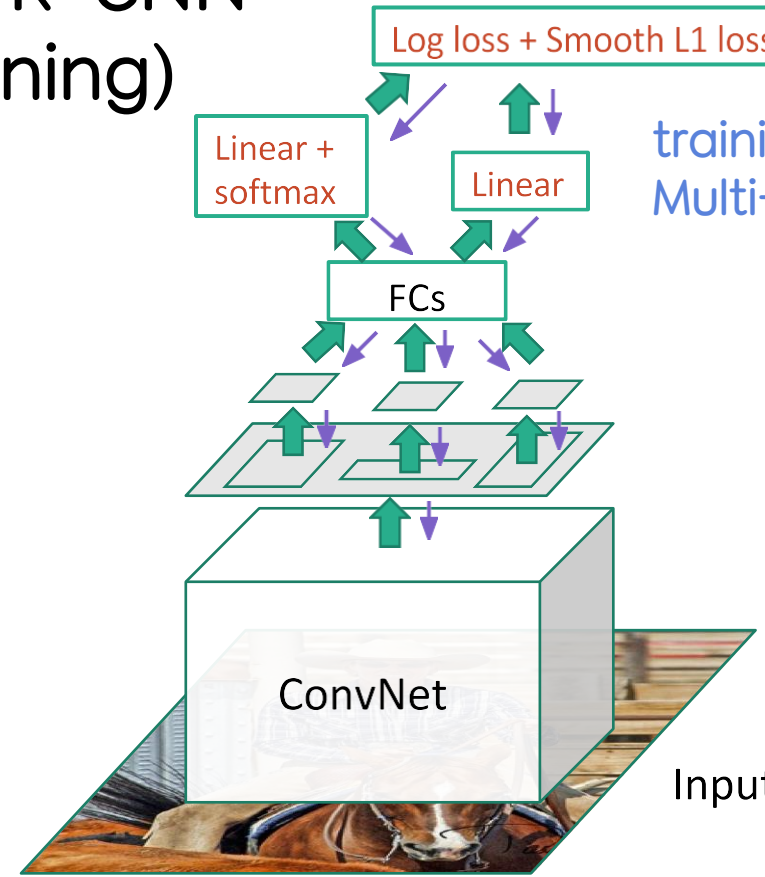
Fast R-CNN (Training)



training시에는
Multi-task loss로 backpropagation을 진행한다.

Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

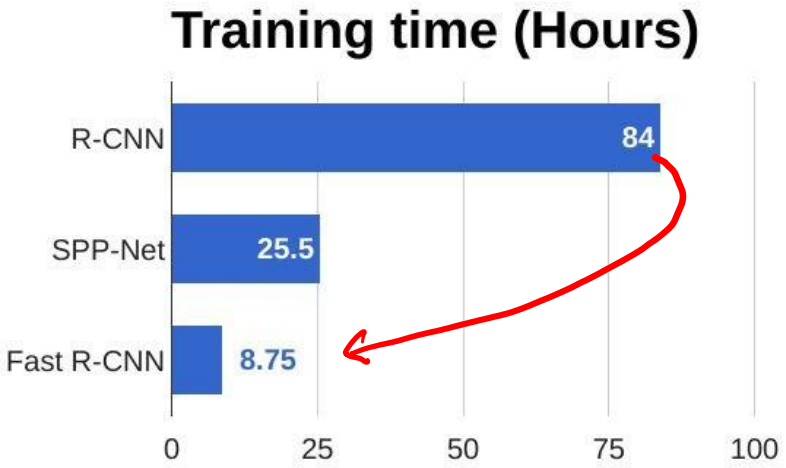
Fast R-CNN (Training)

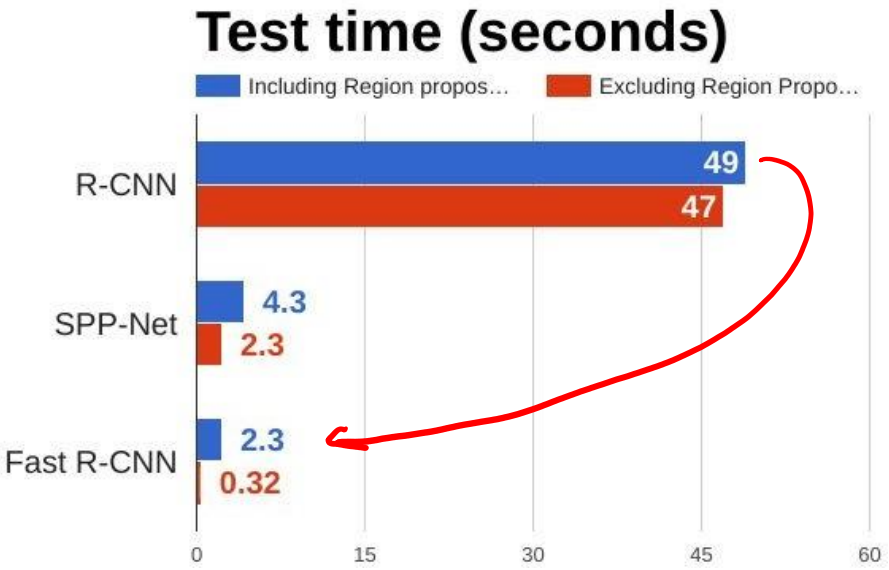


training시에는
Multi-task loss로 backpropagation을 진행한다.

Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

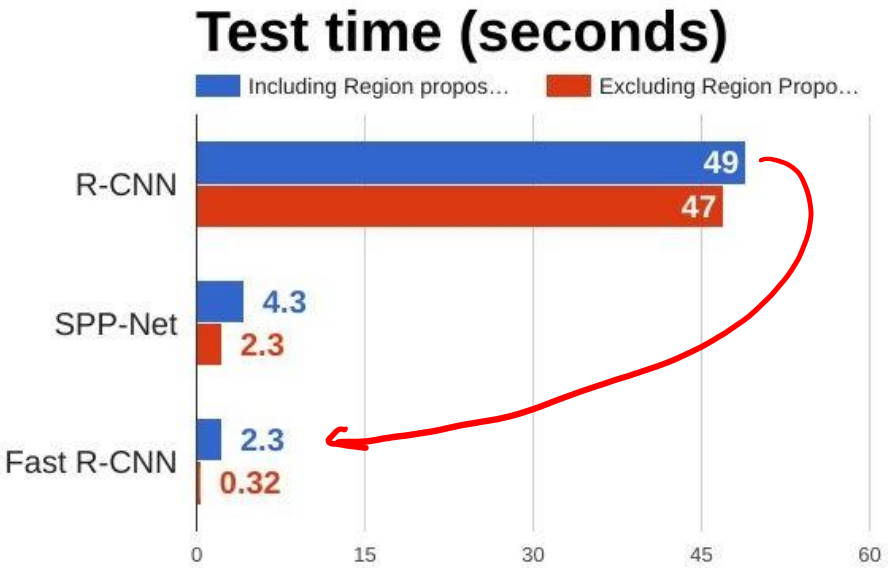
R-CNN vs SPP vs Fast R-CNN





Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVP R 2014. He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2 014
Girshick, "Fast R-CNN", ICCV 2015

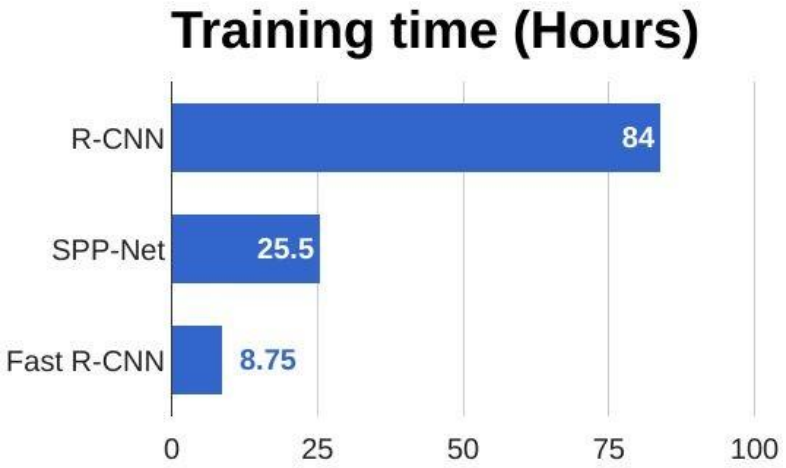
R-CNN vs SPP vs Fast R-CNN

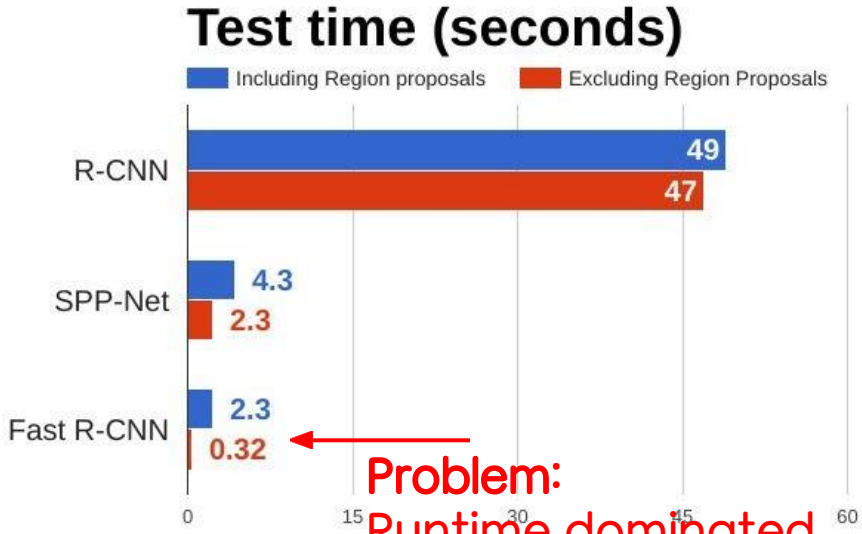


속도 줄어든 이유: CNN을 1번만 통과

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014. He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

R-CNN vs SPP vs Fast R-CNN





Problem:
Runtime dominated by region proposals!

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVP R 2014. He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2 014
Girshick, "Fast R-CNN", ICCV 2015

Faster R-CNN:

Make CNN do proposals!

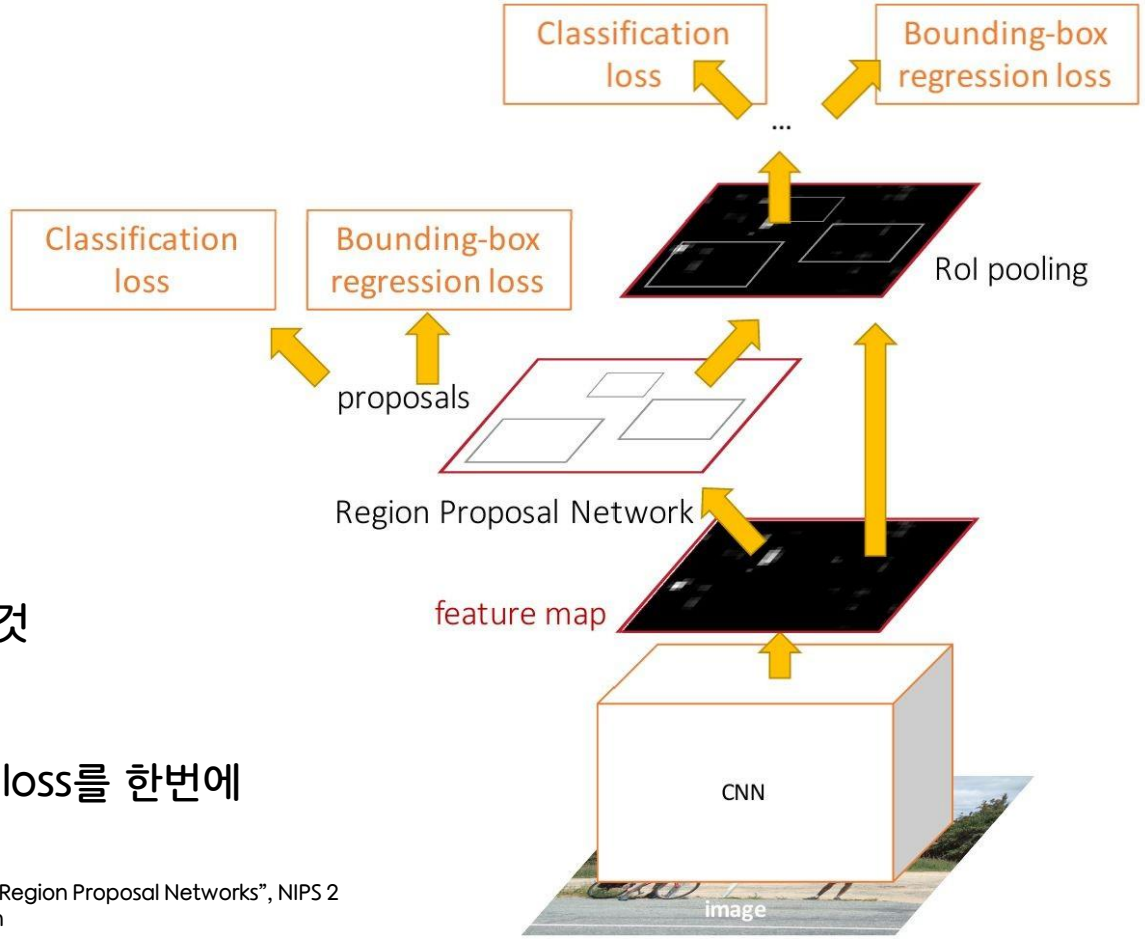
RoI를 찾기 위해

자체 Region Proposal Network(RPN)를 통과한다.

Loss가 4개:

- RPN에서 객체 유무 판단
- RPN에서 bounding box에 대한것
- 나머지 2개는 아까랑 같음

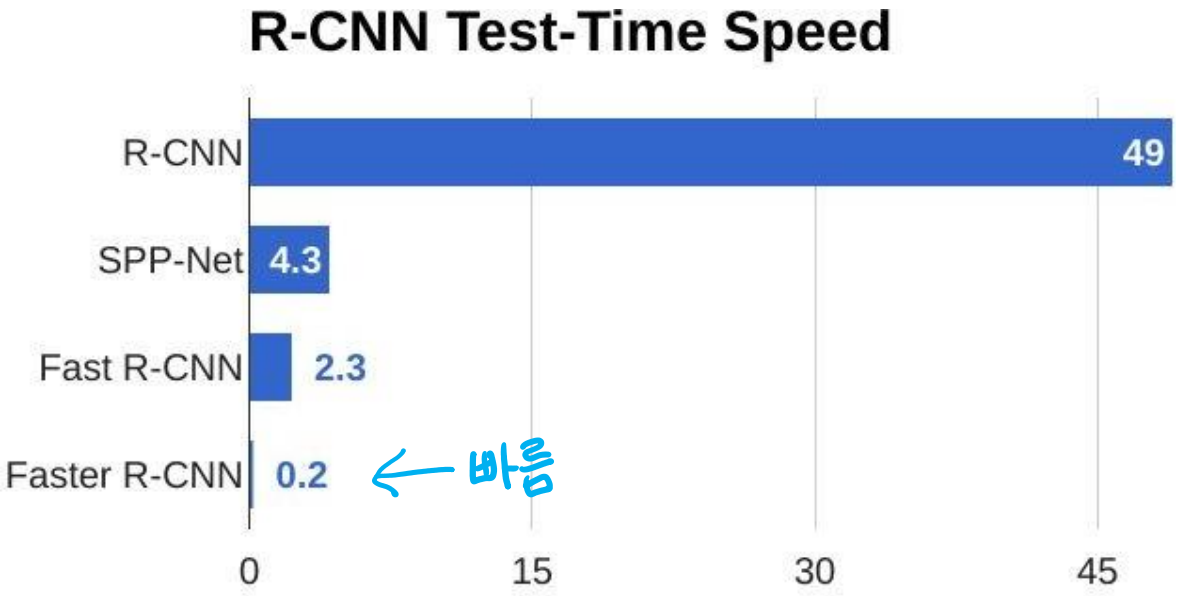
→ Multitask loss를 이용해서 4가지 loss를 한번에 학습



Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015 Figure copyright 2015, Ross Girshick; reproduced with permission

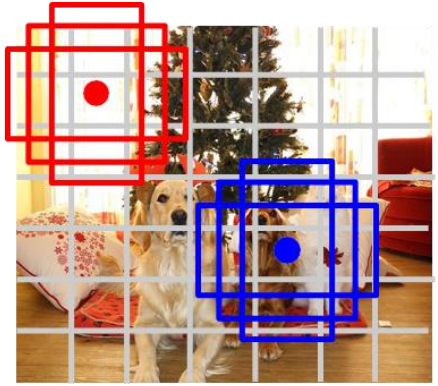
Faster R-CNN:

Make CNN do proposals!



Detection without Proposals: YOLO / SSD

Input image
3 x H x W



Divide image into grid
7 x 7

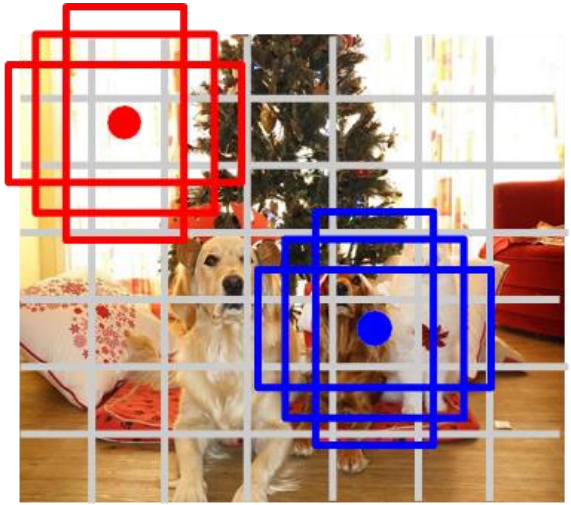
YOLO: You Only Look Once
SSD: Single Shot Detection

한번만에 image detection을 한다
➔ Single shot methods

Output:
7 x 7 x (5 * B + C)

Redmon et al, "You Only Look Once:
Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

Detection without Proposals: YOLO / SSD



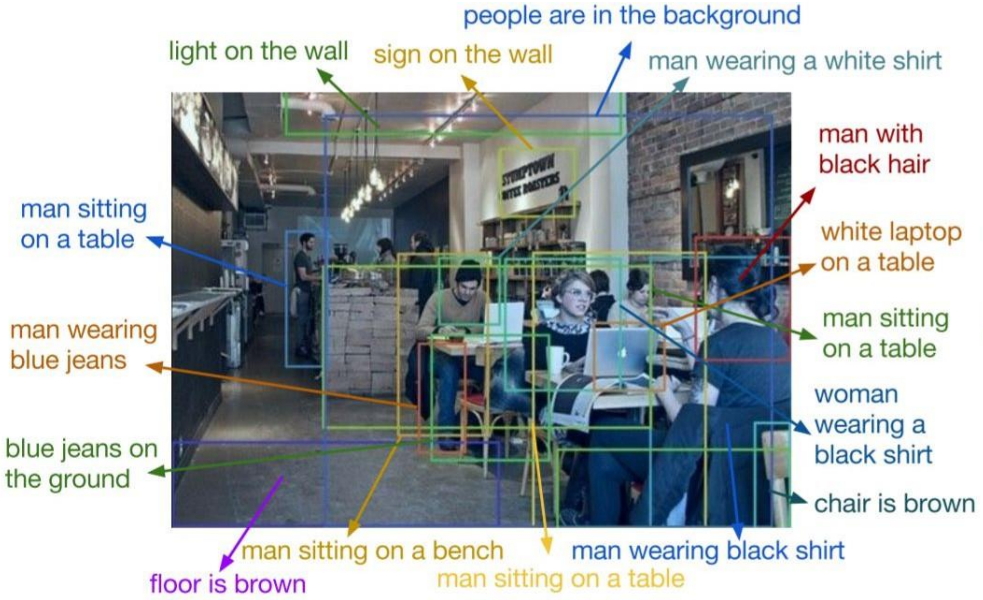
Divide image into grid
7 x 7

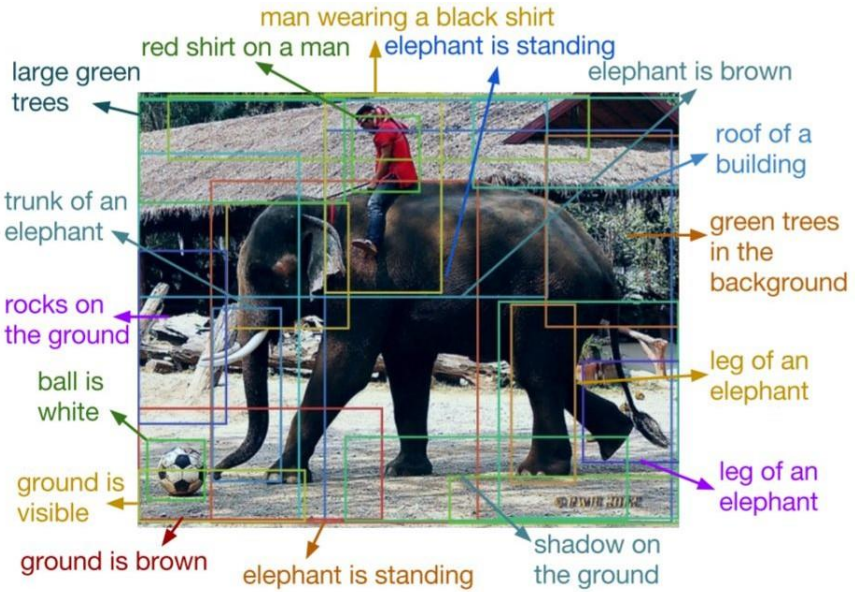
이미지를 7x7로 나눈다.
B=3이라고 했을 때 그림
(한 cell당 찾을 박스 개수)

찾고자 하는 것의
실제 위치로 가려면
얼마나 가야하는지
(offset) 예측

객체가 존재할 가능
성 (classification
score) 예측

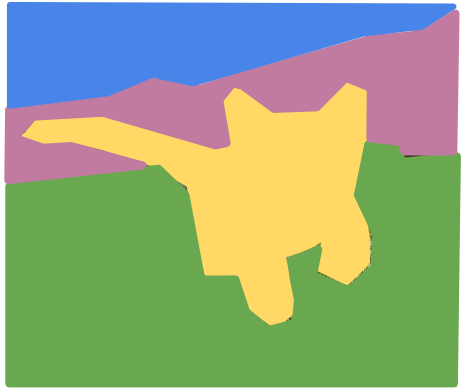
Aside: Object Detection + Captioning = Dense Captioning





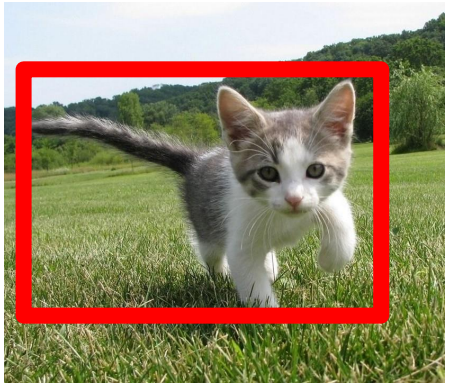
Johnson, Karpathy, and Fei-Fei, "DenseCap: Fully Convolutional Localization Networks for Dense Captioning", CVPR 2016 Figure copyright IEEE, 2016. Reproduced for educational purposes.

Instance Segmentation



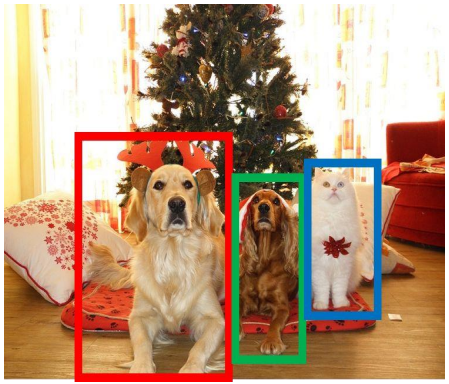
GRASS, CAT,
TREE, SKY

No objects, just pixels



CAT

Single Object



DOG, DOG, CAT

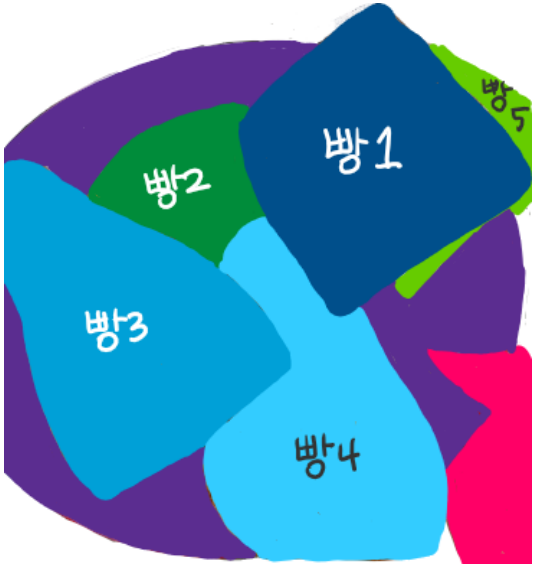
DOG, DOG, CAT

Multiple Object

Instance Segmentation

<Semantic Segmentation>

Instance Segmentation



<Instance Segmentation>

Instance Segmentation

Object detection+ Semantic segmentation

Instance Segmentation

Input image

Instance Segmentation

Input image

찾고자 하는 Class: 머랭쿠키

→

객체의 위치를 찾는 과정

: object detection

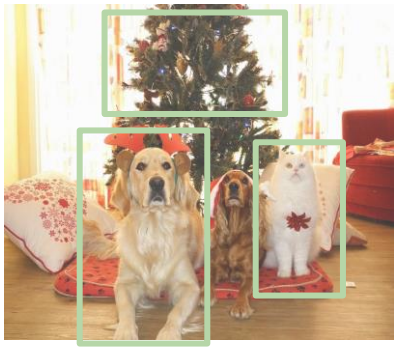
→

Instance Segmentation

Bounding box가 아니라
Segmentation mask를 예측한다.
→ Semantic segmentation

각각의 객체를 구분한다.

Mask R-CNN



He et al, “Mask R-CNN”, arXiv 2017

Faster R-CNN을 이용하여
객체가 있을만한 영역 탐지

CNN

탐지된 bounding box(RoI)

RoI Align

256 x 14 x 14

탐지한 영역 내에 어떤 category가
있을지 예측(classification)

Conv

256 x 14 x 14

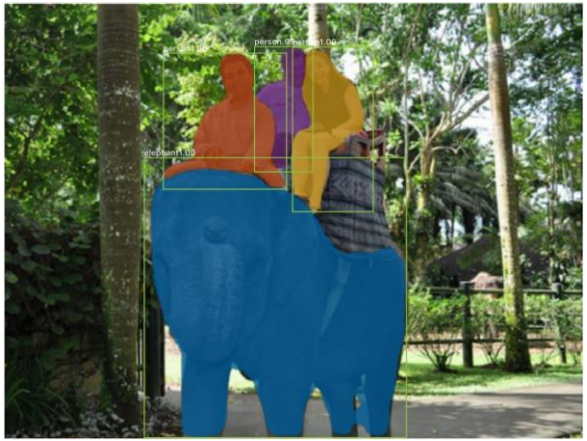
Conv

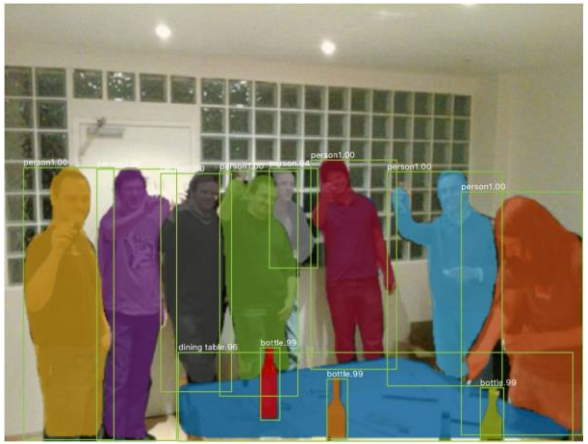
C x 14 x 14

(Semantic segmentation)
픽셀마다 object인지 분류

Class마다 mask 예측

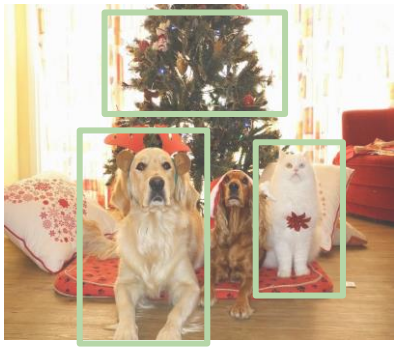
Mask R-CNN: Very Good Results!





He et al, "Mask R-CNN", arXiv 2017
Figures copyright Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, 2017
Reproduced with permission.

Mask R-CNN
Also does pose



He et al, “Mask R-CNN”, arXiv 2017

Faster R-CNN을 이용하여
객체가 있을만한 영역 탐지

CNN

RoI Align

256 x 14 x 14

탐지한 영역 내에 어떤 category가
있을지 예측(classification)
+ 관절 좌표 예측 부분 추가

Conv

256 x 14 x 14

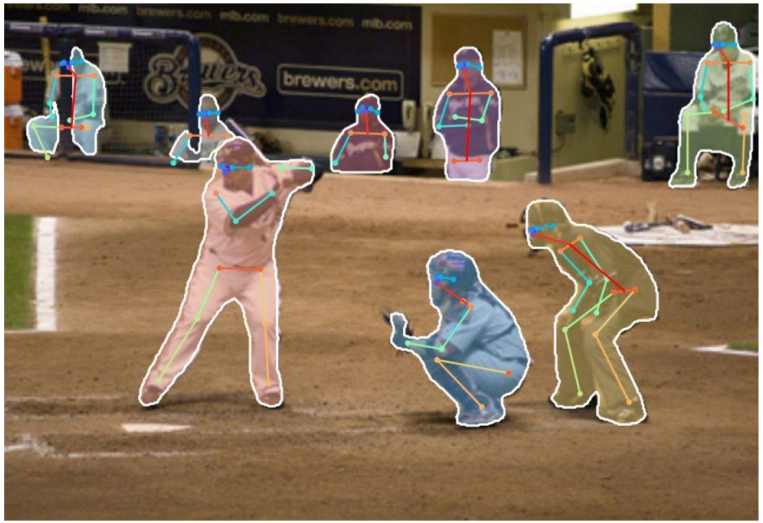
Conv

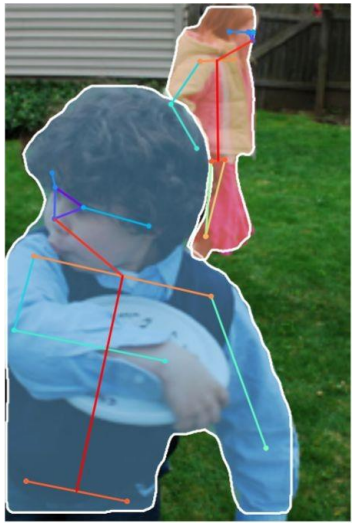
C x 14 x 14

(Semantic segmentation)
픽셀마다 object인지 분류

Class마다 mask 예측

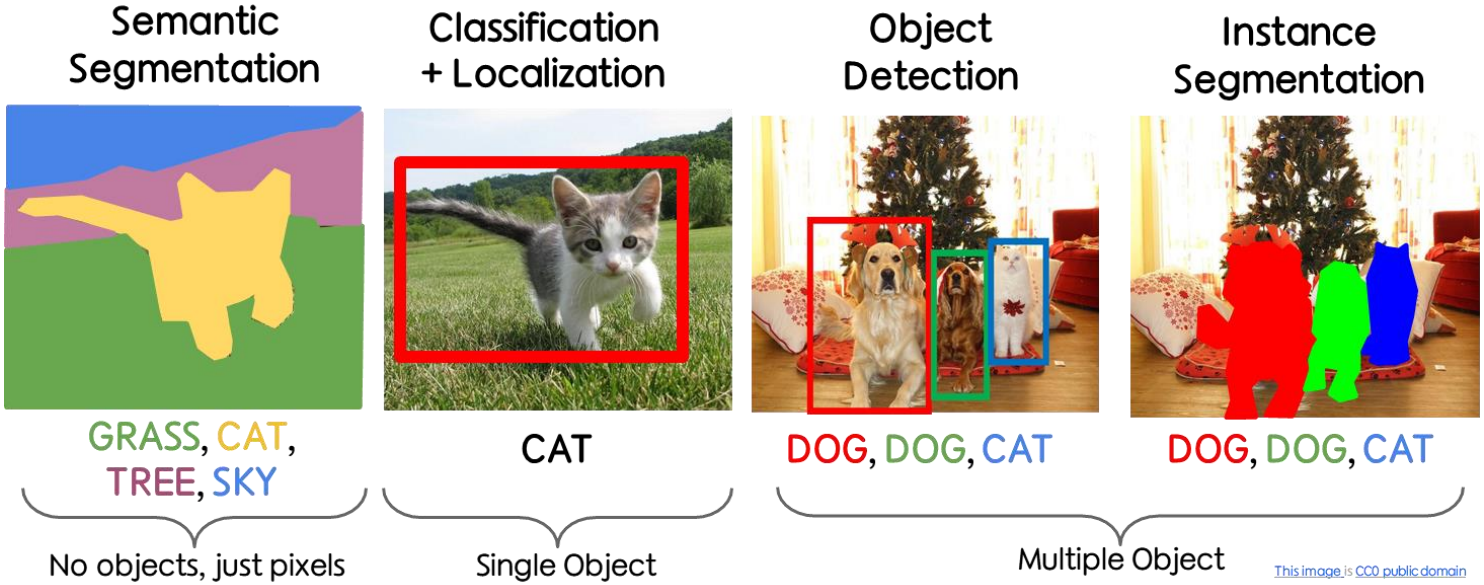
Mask R-CNN
Also does pose





He et al, “Mask R-CNN”, arXiv 2017
Figures copyright Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, 2017
. Reproduced with permission.

정리



Downsampling과 upsampling,

- Upsampling
  - Unpooling
  - Max unpooling
  - Transpose convolution

- Faster R-CNN
- Single Shot Methods(YOLO, SSD)

- Mask R-CNN
- YOLACT