
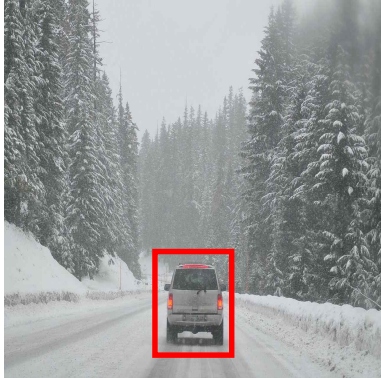

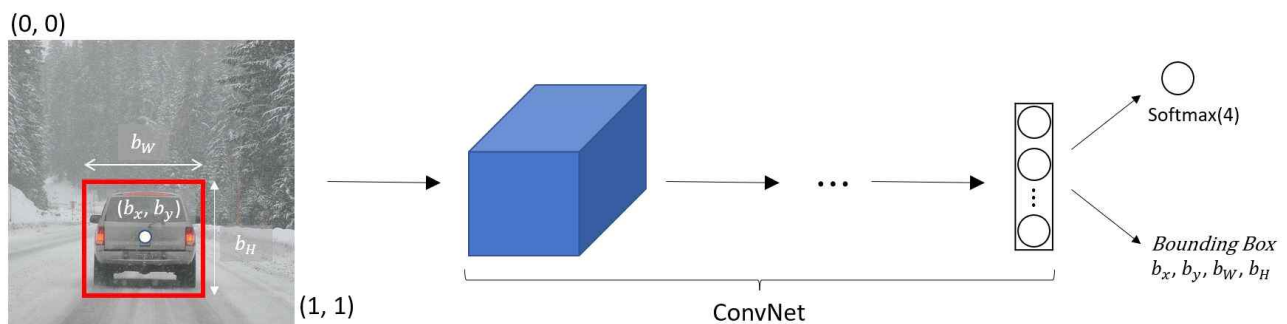


Object Detection

Object localization

Image classification	Classification with localization	Detection
		
	1 Object	Multiple Object

Classification with localization



1. Need to output b_x, b_y, b_H, b_W , class label(1-4)

2.

탐지한 객체 $y =$

$$\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_H \\ b_W \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

p_c : 탐지한 객체가 있는지
 b : 탐지한 객체의 위치
 c : 탐지한 객체의 라벨/클래스

ex1.

$$y = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_H \\ b_W \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

ex2.

$$y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

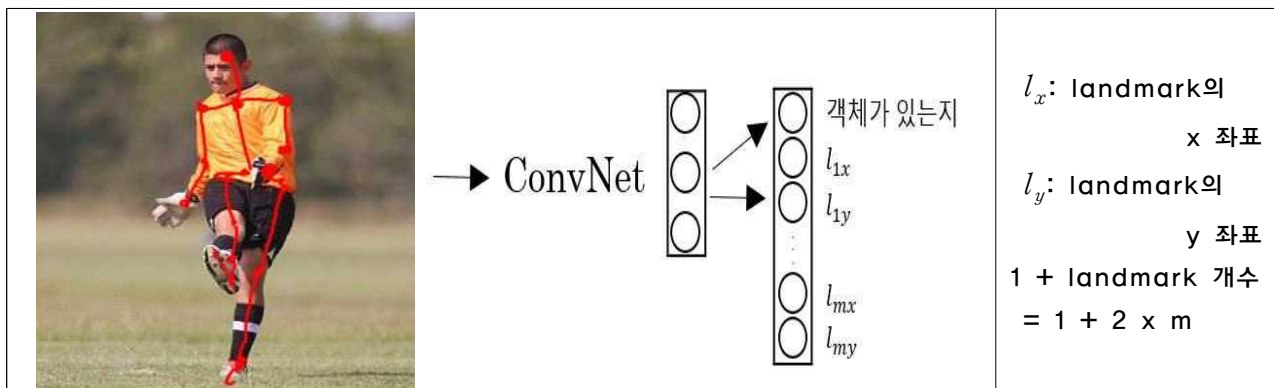
?: don't care

$$3. L(\hat{y}, y) = \begin{cases} (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \dots + (\hat{y}_8 - y_8)^2 & \text{if } y_1 = 1 \\ (\hat{y}_1 - y_1)^2 & \text{if } y_1 = 0 \end{cases}$$

오차 제곱으로 나타냈지만 실제에는 여러 방법을 사용할 수 있다. 예로 들어 c에는 log-likelihood loss와 softmax 출력값들을 사용하고 나머지 p와 b에는 오차 제곱을 사용할 수 있다.

Landmark detection

: 신경망이 이미지의 주요 지점을 (x, y) 좌표를 아웃풋으로 만든다.



Sliding window detection

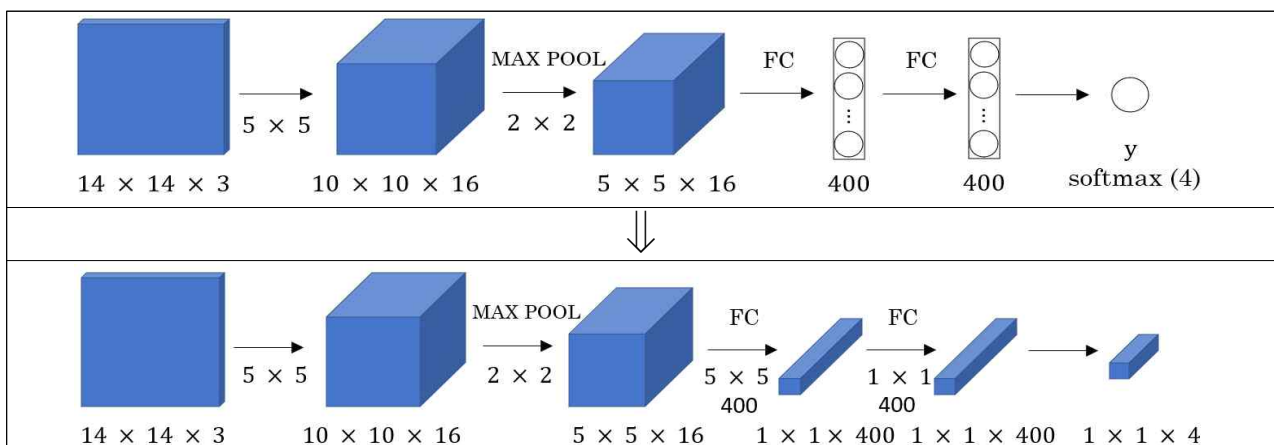


슬라이딩 윈도우의 단점:

아주 큰 computation cost. 이미지의 많은 사각형 영역의 잘라서 CONV를 통해 독립적으로 각각을 실행시키기 때문이다.

큰 윈도우를 사용할수록 슬라이딩 윈도우의 계산 비용을 줄어든다. 하지만 더 거친 gradnularity는 수행력에 안 좋은 영향을 준다. 반대로 아주 고운 gradnularity, 즉 아주 작은 스트라이드로 사용하면 아주 작은 영역으로 CONV를 지나치게 많이 하여 계산 비용이 높을 것이다.

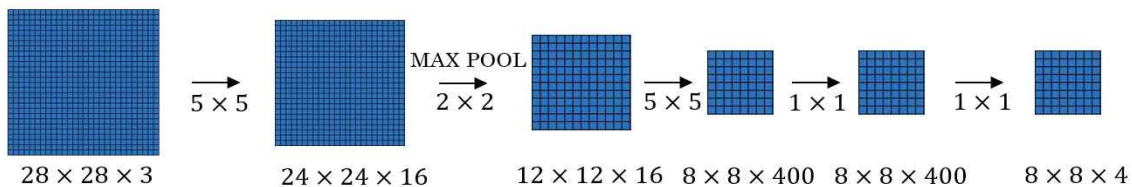
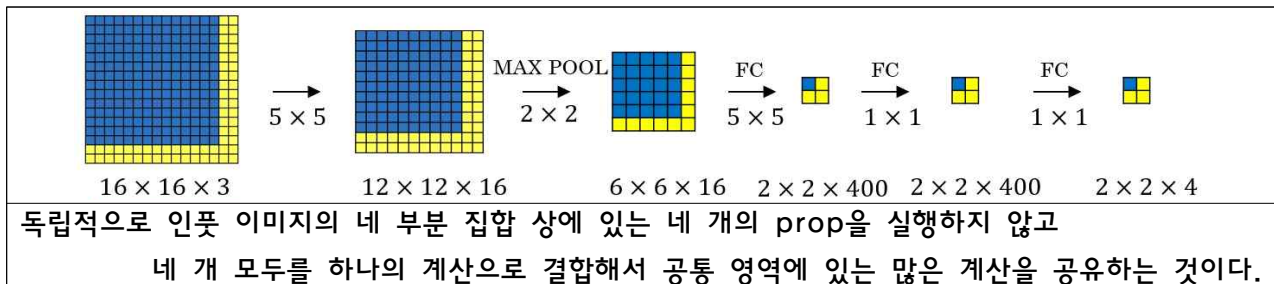
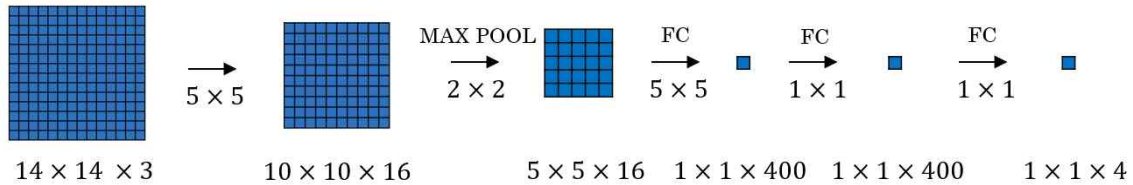
Turning FC layer into convolutional layers



Convolution implementation of sliding windows

[Sermanet et al., 2014, OverFeat:

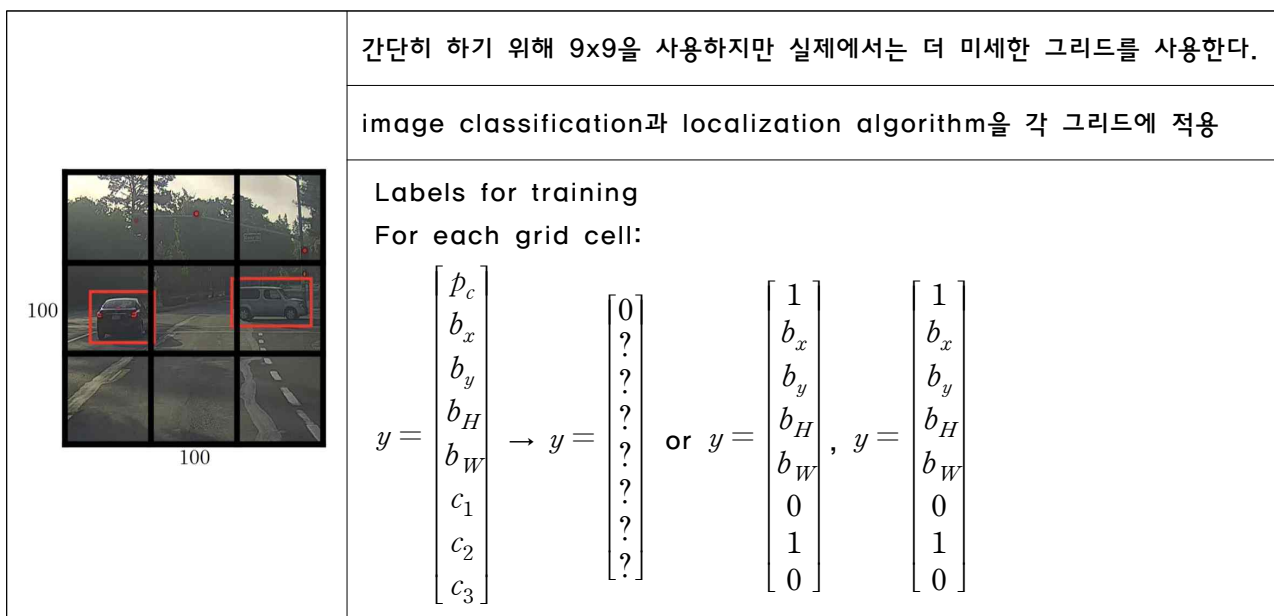
Integrated recognition, localization and detection using convolutional networks]



convolutional implmentation of sliding windows는 계산 비용을 낮춰준다는 장점이 있지만 bounding box의 위치가 정확하지 않다는 단점이 있다.

Yolo algorithm

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



1. image classification과 localization algorithm을 사용하여 객체의 라벨과 위치를 파악한다.
2. sliding window를 CONV로 사용하면 computation cost가 효율적이다.

back prop으로 input $x(\text{image})$ 에 output $y(\text{class, location})$ 를 매핑한다.

Class \Rightarrow 미세한 그리드로 설정하면 한 그리드 셀에 여러 객체가 할당될 가능성을 줄여준다.

즉, 한 그리드 셀에 객체의 중간점이 2개 이상이 될 확률을 줄여준다.

Location \Rightarrow Yolo algorithm은 output에 정확한 bounding box를 만든다.

Specify the bounding box

바운딩 박스 지정하는 방법: 해당 그리드 셀 기준으로

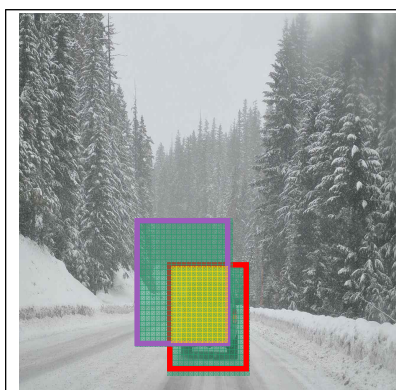
1. $b_x, b_y, b_H, b_W > 0$
2. b_x 와 b_y 를 찾고 (중간점을 지금 그리드 셀에서 찾았으므로 between 0 and 1)
3. b_H 와 b_W 를 찾는다 (객체가 지금 그리드 셀을 넘어갈 수 있으므로
could be greater than 1)

Intersection over Union(IoU)

evaluating object localization

More generally, IoU is a measure of the overlap

between two bounding boxes.



Intersection over Union(IoU)

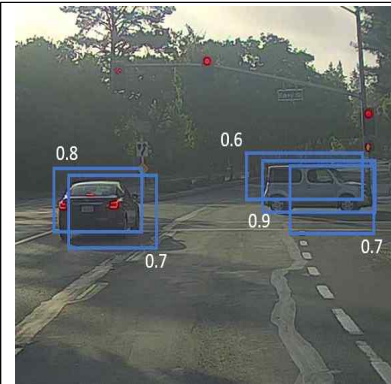
$$= \frac{\text{size of intersection}}{\text{size of union}}$$

ex) "Correct" if $\text{IoU} \geq 0.5$

한계치를 크게 두면 더 엄격하게 측정된다.

Non-max suppression

Object detection의 문제점 중 하나는 알고리즘이 같은 객체에 대해 여러 번 탐지를 할 수 있다는 것이다. Non-max suppression은 각 객체를 한 번만 탐지할 수 있게 한다.



먼저 가장 큰 확률을 가진 bounding box(0.9)를 선택한다.

non-max suppression는 남아있는 모든 bounding box을 보고 많이 겹쳐 있어 기준치 이상의 IoU를 가지고 있는 모든 bounding box을 suppression한다.(0.6, 0.7)

그리고 남은 bounding box을 확인하여 가장 큰 확률을 가진 bounding box(0.8)을 선택하고 non-max suppression으로 높은 IOU를 가진 bounding box를 지운다.

그리고 반복한다.

Each output prediction is:

$$\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_H \\ b_W \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Discard all boxes with $p_c \leq 0.6$

While there are any remaining boxes:

- Pick the box with the largest p_c

Output that as a prediction

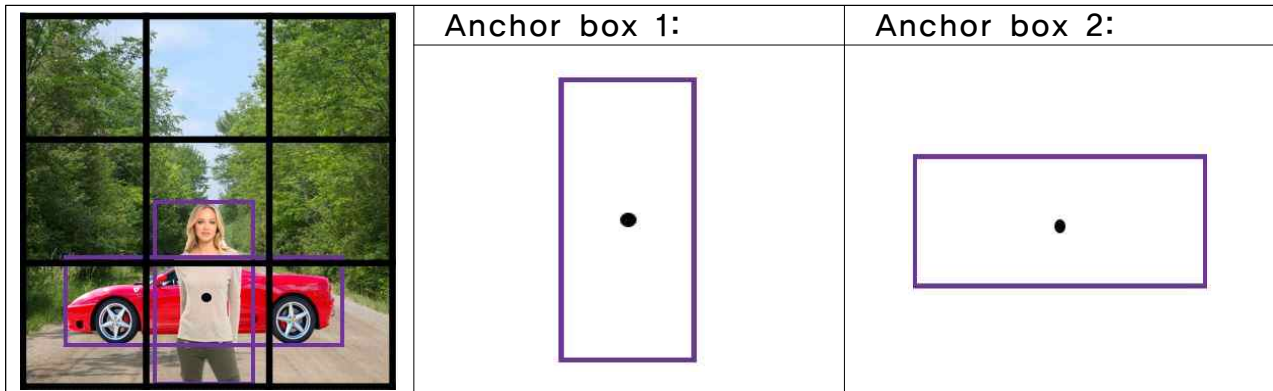
- Discard any remaining box with $\text{IoU} \geq 0.5$

with the box output in the previous step

output class 당 하나씩 독립적으로 non-max suppression을 수행한다.

Anchor boxes

object detection의 문제 중 하나는 각 그리드 셀이 하나의 객체만 감지할 수 있다는 것이다. 만약 한 그리드 셀에서 여러 객체를 감지할 수 있다면 어떨까?



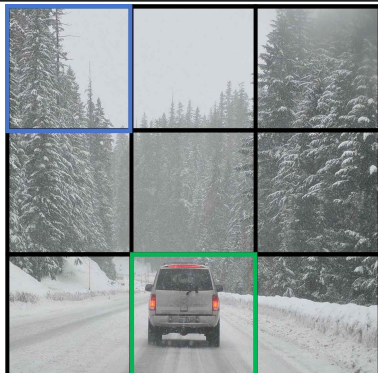
Previously:	With two anchor boxes:
$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_H \\ b_W \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$ <p>Each object in training image is assigned to grid cell that contains that object's midpoint.</p>	$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_H \\ b_W \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_H \\ b_W \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$ <div style="display: flex; align-items: center; justify-content: center;"> <div style="text-align: center;"> \Rightarrow </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div style="display: flex; align-items: center;"> <div style="text-align: center;"> $\left. \begin{matrix} p_c \\ b_x \\ b_y \\ b_H \\ b_W \\ c_1 \\ c_2 \\ c_3 \end{matrix} \right\}$ </div> <div style="margin: 0 10px;">}</div> <div>anchor box 1</div> </div> <div style="display: flex; align-items: center;"> <div style="text-align: center;"> $\left. \begin{matrix} p_c \\ b_x \\ b_y \\ b_H \\ b_W \\ c_1 \\ c_2 \\ c_3 \end{matrix} \right\}$ </div> <div style="margin: 0 10px;">}</div> <div>anchor box 2</div> </div> </div> <p>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.</p> </div>

단점, 문제점:

1. 만약 anchor box는 2개인데 같은 그리드 셀 안에 3개의 객체가 있으면 잘 처리하지 못한다. 이것을 처리하는 좋은 방법이 없다. 디폴트 브레이커 효과를 줘야 한다.
 2. 같은 그리드 셀에서 2개의 객체가 같은 anchor box 모양을 가지고 있을 때도 제대로 처리하지 못한다. 이 일이 생기면 타이브레이크의 효과를 디폴트로 줘야 한다.
- 그리드가 미세하게 촘촘할수록 중간점이 2개일 확률은 줄어든다. anchor box가 제공하는 motivation이나 결과들의 알고리즘 학습이 더 전문화되도록 해준다.

anchor box 선택 방법: 직접 선택하기도 하고 다양한 모양으로 여러 개 사용하기도 한다. 요즘 가장 좋다고 알려진 것은 k-mean algorithm을 사용하는 것이다.

Putting it together: Yolo algorithm

	$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_H \\ b_W \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_H \\ b_W \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$	blue grid cell: $y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$	green grid cell: $y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_x \\ b_y \\ b_H \\ b_W \\ 0 \\ 1 \\ 0 \end{bmatrix}$
y is $3 \times 3 \times 2 \times 8$			
class 1: pedestrian class 2: car class 3 : motorcycle			
y is $n_H \times n_W \times \#anchor\ boxes \times (p_C, b_x, b_y, b_H, b_W, \#classes)$			

Outputting the non-max suppressed output

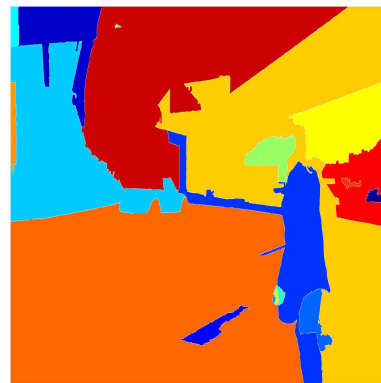
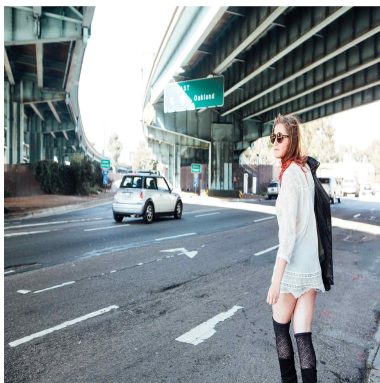
1. For each grid cell, get 2 predicted bounding boxes.
2. Get rid of low probability predictions.
3. For each class (pedestrian, car, motorcycle), use non-max suppression to generate final prediction.

Region proposals

R-CNN

일부 region만 선택하고 region끼리 crossfire하는 것이다. sliding window로 모든 것을 처리하지 말고 단지 일부 window만 선택해서 continent crossfire를 한다.

수행 방법: 어떤 객체인지 알아내기 위해 결과(오른쪽 output)를 내는 세분화 알고리즘 실행
ex) 세분화 알고리즘은 작은 blob을 찾고 그것을 골라서 "Let's run a crossfire on that blob"하는 것이라고 할 수 있다.

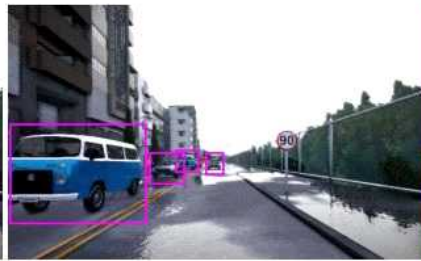


R-CNN	Propose regions. Classify proposed regions one at a time. Output label + bounding box
[Girshik et. al, 2013. Rich feature hierarchies for accurate object detection and semantic segmentation]	
Fast R-CNN	Propose regions. Use convolution implementation of sliding windows to classify all the proposed regions
[Girshik, 2015. Fast R-CNN]	
Faster R-CNN	Use convolutional network to propose regions.
[Ren et. al, 2016. Faster R-CNN: Towards real-time object detection with region proposal network]	

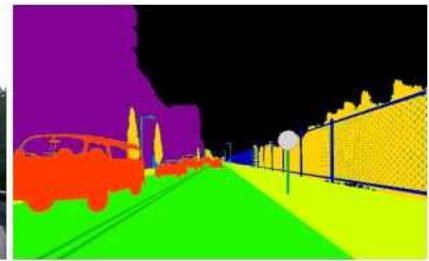
Sementic Segmentation with U-Net



Input image



Object Detection



Semantic Segmentation

[Novikov et al., 2017, Fully Convolutional Architectures]

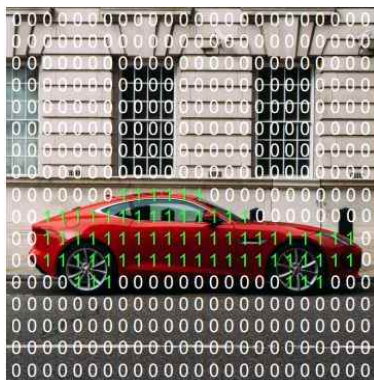
for Multi-Class Segmentation in Chest Radio graphs]

[Dong et al., 2017, Automatic Brain Tumor Detection and Segmentation

using U-Net Based Fully Convolutional Networks]

Per-pixel class labels

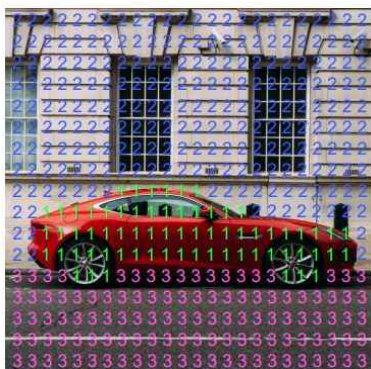
1. two classes



1. Car

0. Not Car

2. three classes



1. Car

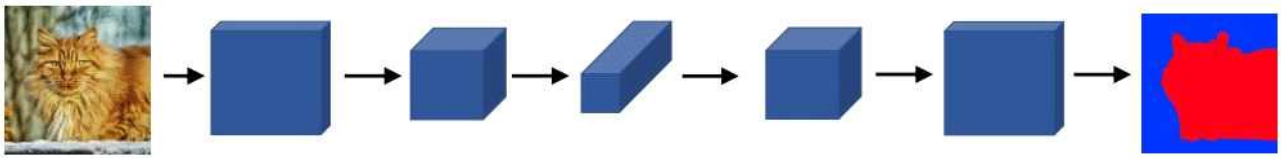
2. Building

3. Road

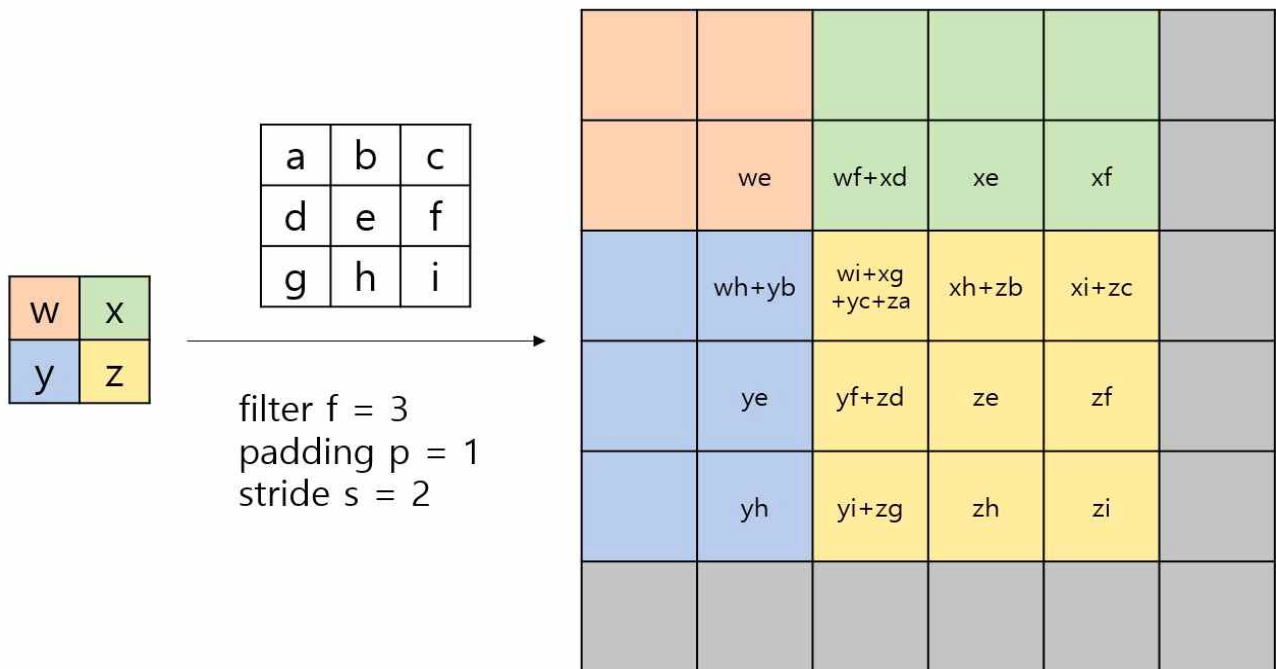
[illegible]

Segmentaion Map

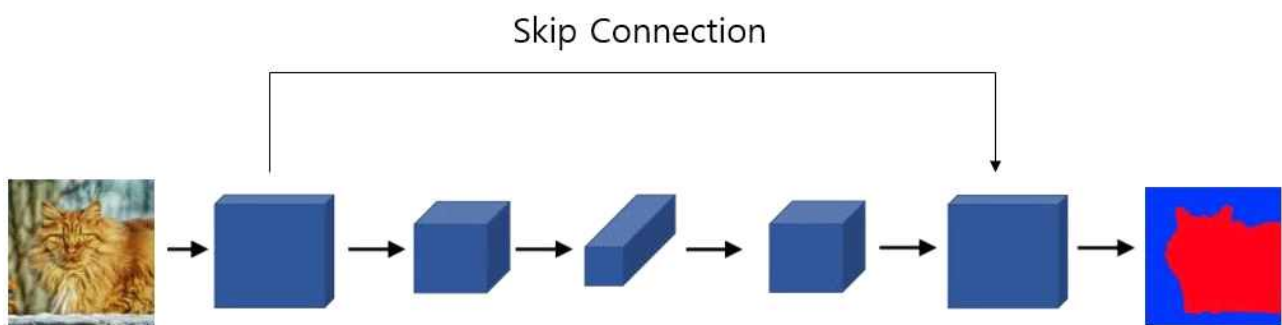
Deep learning for Segmentation



Transpose Convolution



Deep learning for Segmentation



장점: It turns out that next final layer to decide which region is a cat.

2가지 유용한 정보

1. lower resolution
2. high level, spatial, high level contextual information, as well as the low level

previous layer에서의 high level, spatial, high level contextual information 하지만 very detailed, fine grained spatial information이 사라졌다. activations 가 lower spatial resolution to heighten with is just lower를 가지기 때문이다. skip connection이 신경망이 high resolution(low level feature information where it could capture for every pixel position)을 유지하도록 한다. skip connection이 later layer과 직접 연결하면서 this layer가 lower resolution과 high level, spatial, high level contextual information, 그리고 low level contextual information을 모두 갖는다. 하지만 특정 픽셀이 고양이의 일부인지 아닌지를 결정하는 것은 more detailed texture와 같은 정보가 필요하다.

U-Net

[Ronnerberger et al., 2015, U-Net:

Convolutional Networks for Biomedical Image Segmentation]

