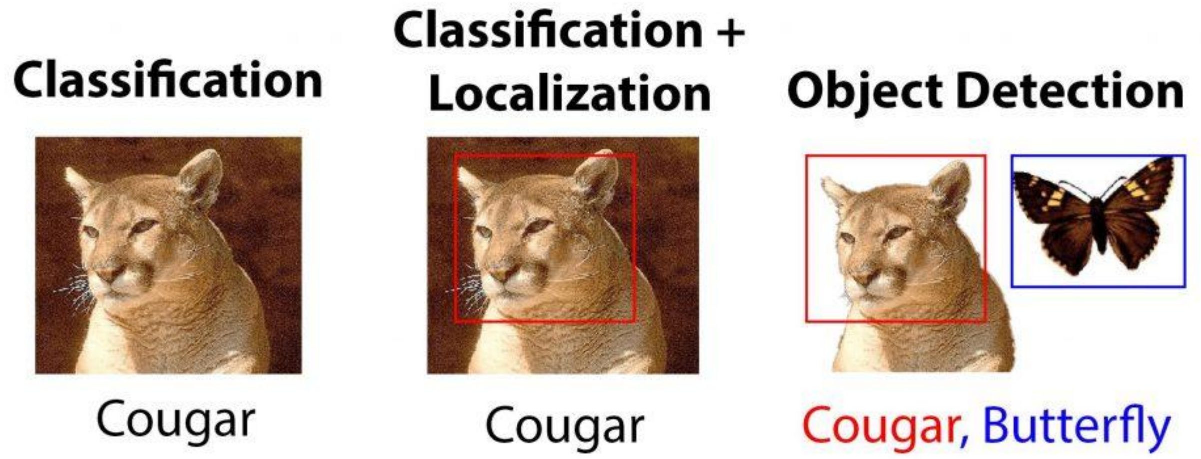
Object Detection

Arsham Golamzadeh Khoee
Alireza Javaheri

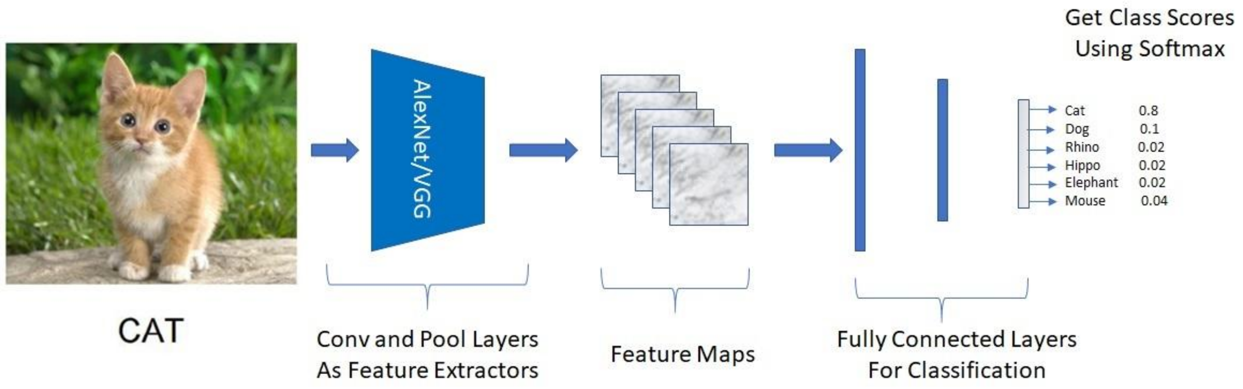
What are localization and detection ?



Classification with localization

Idea for Localization

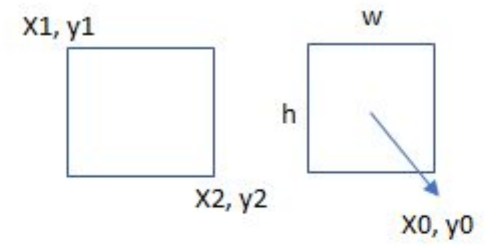
This is a task of locating in an image and There is a single object in the image . We have to draw bounding box around .**We act like image classification and extend it to do localization**



Classification with localization

Problem-(need 4 outputs per class)

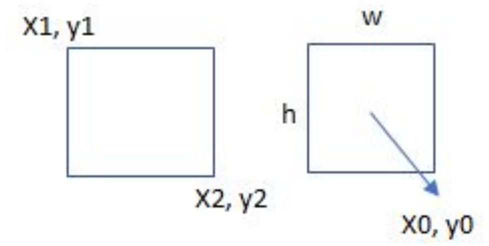
- To get the bounding box we need 4 outputs
- They point to the mid point of the box (x0, y0) and the height and width (h, w) of the box.



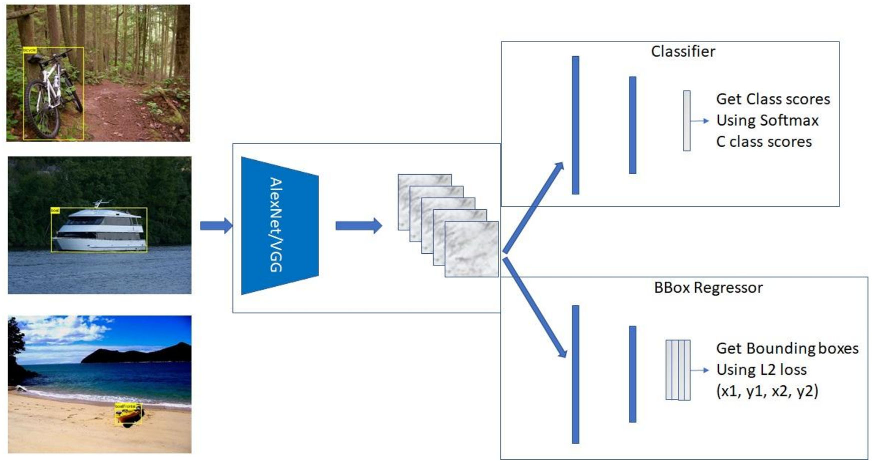
Classification with localization

Problem-(need 4 outputs per class)

- To get the bounding box we need 4 outputs
- They point to the mid point of the box (x0, y0) and the height and width (h, w) of the box.



Solution-(Modify last layer)



Classification with localization

Defining the target label y

1. Pedestrian
2. Car
3. Motorcycle
4. background

Output: b_x, b_y, b_w,b_h - class label

Classification with localization

Defining the target label y

1. Pedestrian
2. Car
3. Motorcycle
4. background

Output: b_x, b_y, b_w,b_h - class label (1 - 4)

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

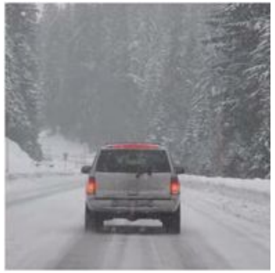
Classification with localization

Example

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

Classification with localization

Example



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

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

Classification with localization

Defining Loss

y = 1 **loss**

$$\begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \ldots + (\hat{y}_8 - y_8)^2$$

y = 0 **loss**

$$\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \quad (\hat{y}_1 - y_1)^2$$

Classification with localization

Defining Loss

y = 1 **loss**

$$\begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \ldots + (\hat{y}_8 - y_8)^2$$

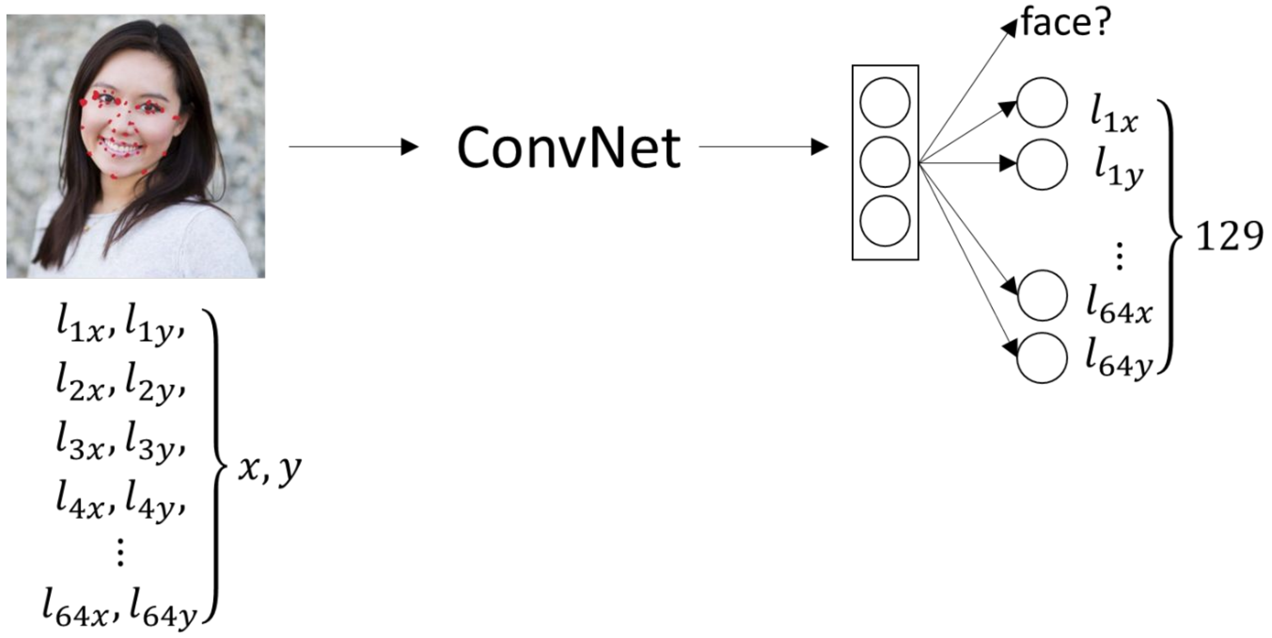
Put it together $\Rightarrow$

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

y = 0 **loss**

$$\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \quad (\hat{y}_1 - y_1)^2$$

Land mark detection



Ideas for Object Detection

We used the classification convNet as the base and extended it to do Localization.

On the same lines, can we extend the localization network to locate more than on object ? if so, how ?

Ideas for Object Detection

We used the classification convNet as the base and extended it to do Localization.

On the same lines, can we extend the localization network to locate more than on object ? if so, how ?

Single Object

We can just resize the image to the required dimension and use the same localization network

Ideas for Object Detection

We used the classification convNet as the base and extended it to do Localization.

On the same lines, can we extend the localization network to locate more than on object ? if so, how ?

Single Object

We can just resize the image to the required dimension and use the same localization network

Two Objects

Split the image into 2, so that the 2 objects are separated, and for each, we act as a single object manner.

Ideas for Object Detection

Problem

Any number of object

Ideas for Object Detection

Problem

Any number of object

Solution

Sliding Window

- We neither know the number of objects nor their location. The only option is to scan all possible locations in an image.
- Crops at all possible locations in the image and feed it to the localization network.

Ideas for Object Detection

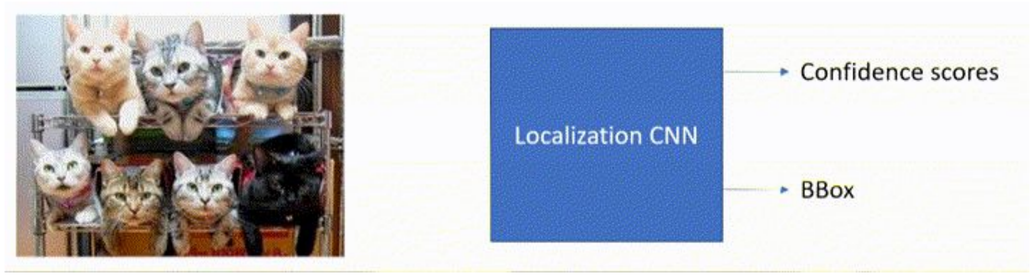
Problem

Any number of object

Solution

Sliding Window

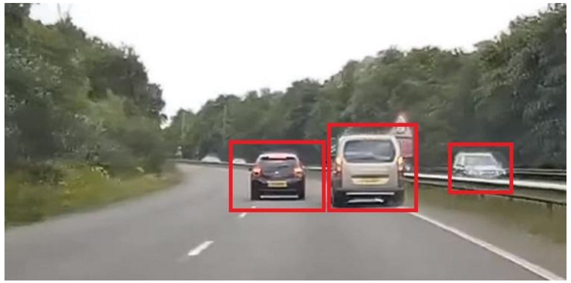
- We neither know the number of objects nor their location. The only option is to scan all possible locations in an image.
- Crops at all possible locations in the image and feed it to the localization network.



Ideas for Object Detection

Problem

Objects of Different Sizes



Ideas for Object Detection

Problem

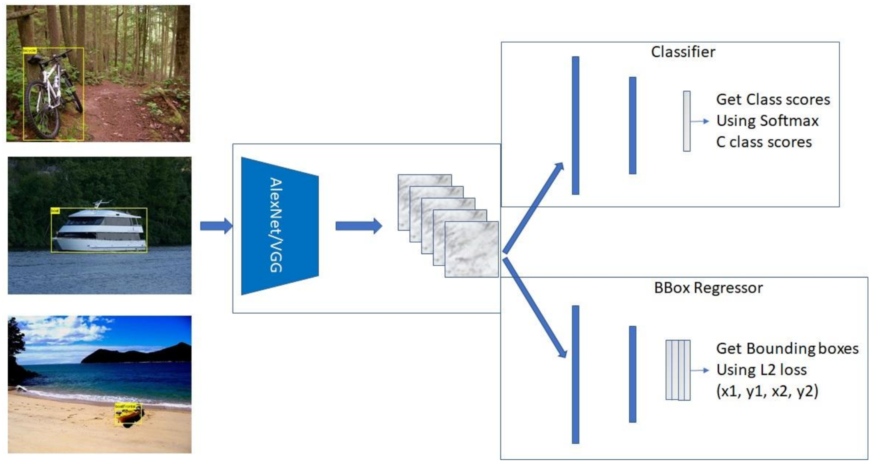
Objects of Different Sizes

Solution

1. Use sliding windows of different sizes like **sliding window detection**
2. Resize the main image itself, keeping the sliding window size constant.

Sliding Window Detection

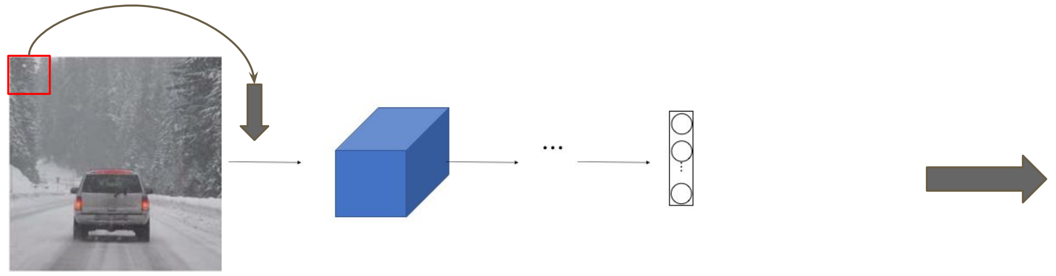
Classification with localization



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

Sliding Window Detection

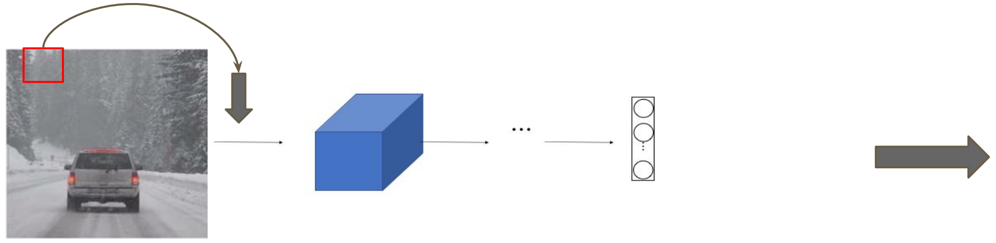
- Choose a grid cell of a specific size. Let us choose the grid cell of size 2*2.
- Pass grid cell through the image and convolute the part of image in the grid cell and predict the output.



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

Sliding Window Detection

- Slide the grid cell through stride and then convolute the next part of the image.

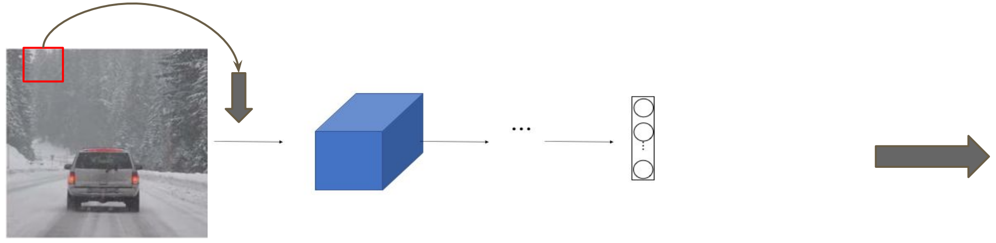


$$Y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

OBJ

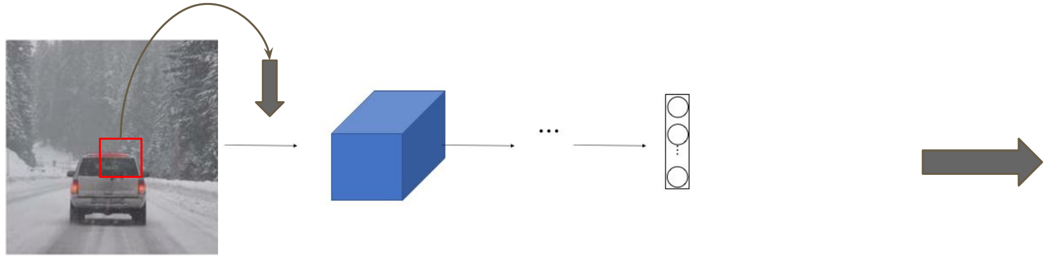
Sliding Window Detection

- Slide the grid cell through stride and then convolute the next part of the image.



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

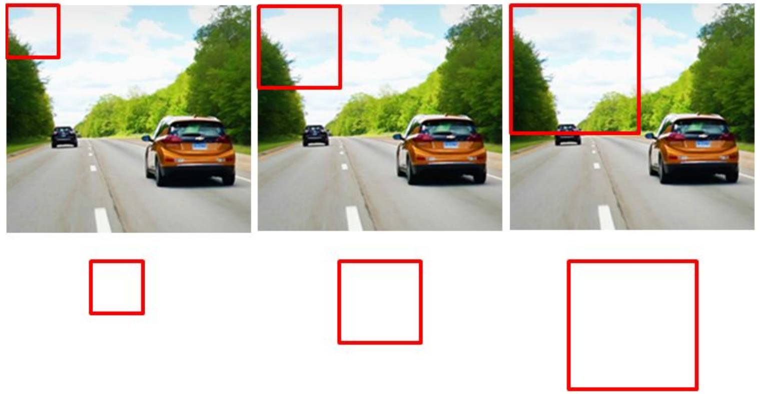
- Go cover the whole image.



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

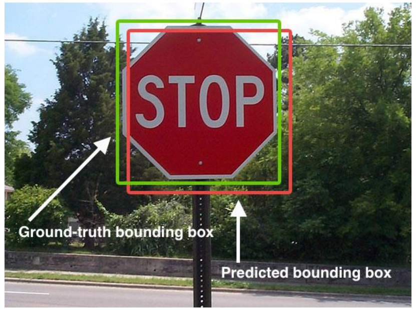
Sliding Window Detection

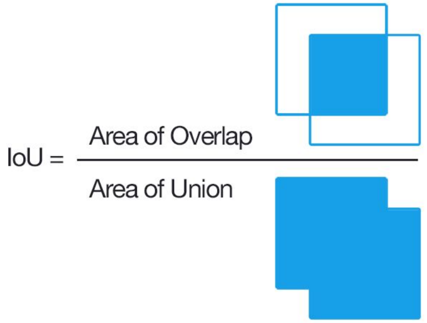
- Repeat the same procedure with different grid cells size.

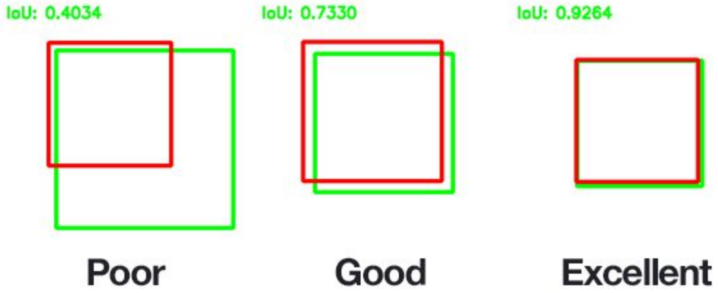


Evaluating Object localization

Intersection Over Union



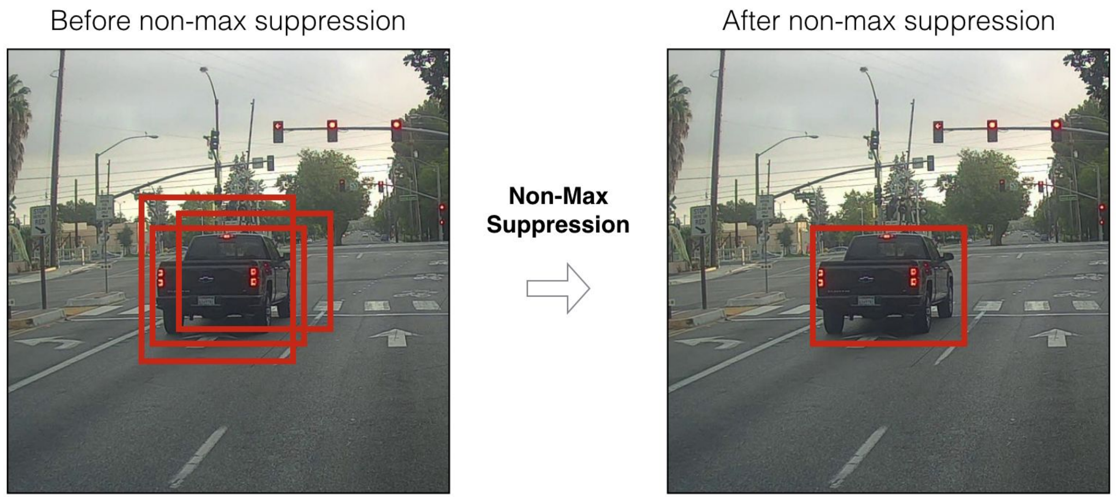




- When IoU is 1 that would imply that predicted and the ground-truth bounding boxes overlap perfectly

Non-max suppression

- Our objective with object detection is to detect an object just once with one bounding box. However, with object detection, we may find multiple detections for the same objects. Non-Max suppression ensures detection of an object only once.



Non-max suppression

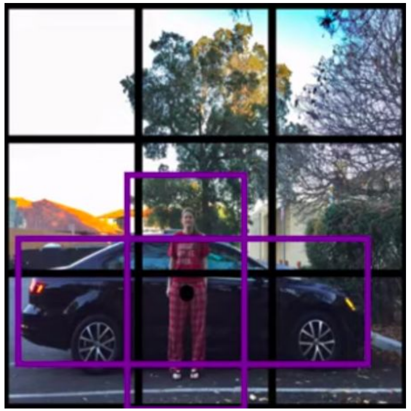
Each output prediction is : $\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \end{bmatrix}$

Discard all boxes with pc <= 0.6

While there are any remaining boxes:

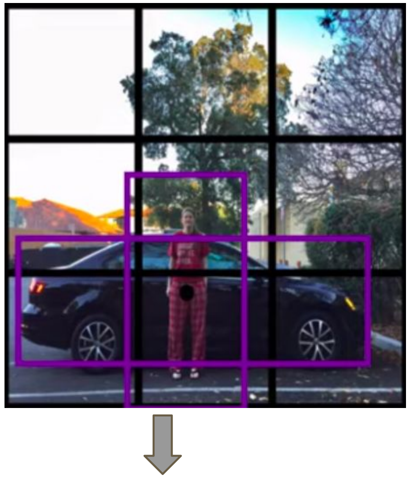
- Pick the box with the largest pc output that as a prediction
- Discard any remaining box with IoU >= 0.5 with the box output in the previous step

Anchor boxes

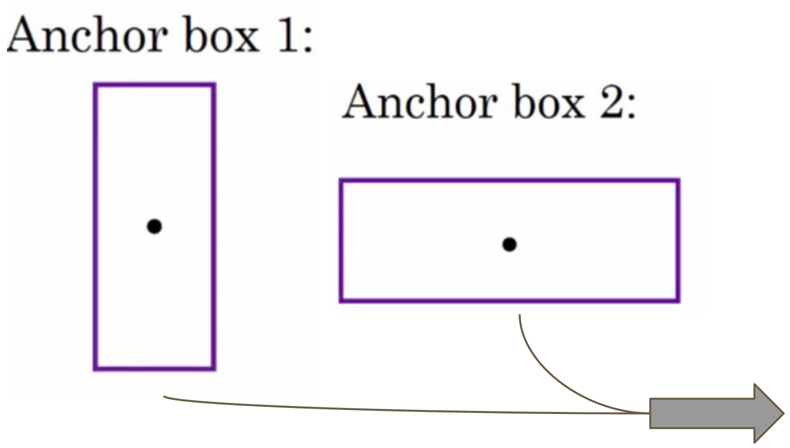


$$Y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Anchor boxes



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



$$y = \begin{bmatrix} p_{c1} \\ b_{x1} \\ b_{y1} \\ b_{w1} \\ b_{h1} \\ c_1 \\ c_2 \\ c_3 \\ p_{c2} \\ b_{x2} \\ b_{y2} \\ b_{w2} \\ b_{h2} \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Anchor box

Previously

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

Output: 3*3*8

With two anchor boxes

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

Output: 3*3*16

Anchor box

Previously

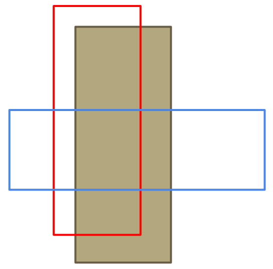
Each object in training image is assigned to grid cell that contains that object's midpoint

Output: 3*3*8

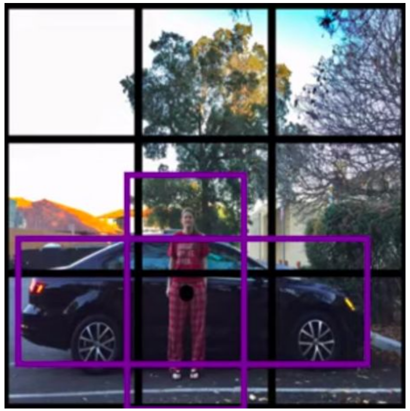
With two anchor boxes

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

Output: 3*3*16

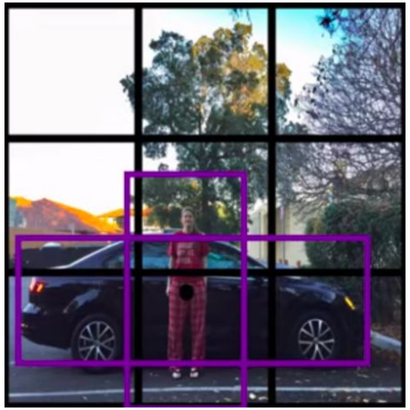


Anchor box example



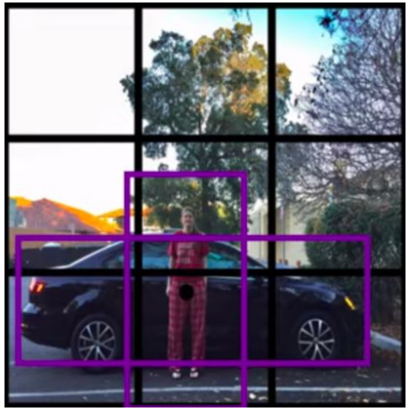
$$y = \begin{bmatrix} p_{c1} \\ b_{x1} \\ b_{y1} \\ b_{w1} \\ b_{h1} \\ c_1 \\ c_2 \\ c_3 \\ p_{c2} \\ b_{x2} \\ b_{y2} \\ b_{w2} \\ b_{h2} \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Anchor box example



$$y = \begin{bmatrix} p_{c1} \\ b_{x1} \\ b_{y1} \\ b_{w1} \\ b_{h1} \\ c_1 \\ c_2 \\ c_3 \\ p_{c2} \\ b_{x2} \\ b_{y2} \\ b_{w2} \\ b_{h2} \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ b_{x1} \\ b_{y1} \\ b_{w1} \\ b_{h1} \\ 1 \\ 0 \\ 0 \\ 1 \\ b_{x2} \\ b_{y2} \\ b_{w2} \\ b_{h2} \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

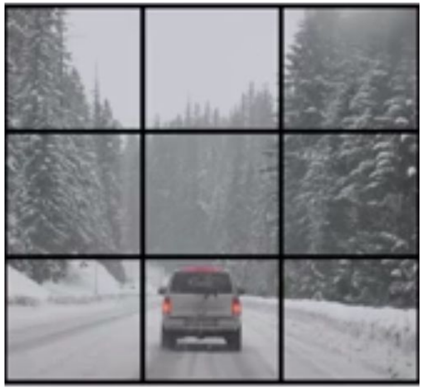
Anchor box example



$$y = \begin{bmatrix} p_{c1} \\ b_{x1} \\ b_{y1} \\ b_{w1} \\ b_{h1} \\ c_1 \\ c_2 \\ c_3 \\ p_{c2} \\ b_{x2} \\ b_{y2} \\ b_{w2} \\ b_{h2} \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ b_{x1} \\ b_{y1} \\ b_{w1} \\ b_{h1} \\ 1 \\ 0 \\ 0 \\ 1 \\ b_{x2} \\ b_{y2} \\ b_{w2} \\ b_{h2} \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad y = \begin{bmatrix} 0 \\ b_{x1} \\ b_{y1} \\ b_{w1} \\ b_{h1} \\ 0 \\ 0 \\ 0 \\ 1 \\ b_{x2} \\ b_{y2} \\ b_{w2} \\ b_{h2} \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Put it together:
YOLO algorithm

Training



Y is : 3*3*2*8

1. Pedestrian
2. Car
3. motorcycle

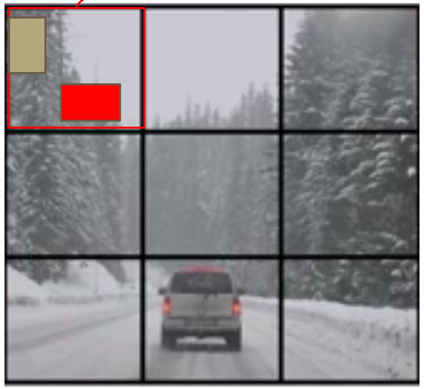
Training

Y is : 3*3*2*8

$$y = \begin{bmatrix} p_{c1} \\ b_{x1} \\ b_{y1} \\ b_{w1} \\ b_{h1} \\ c_1 \\ c_2 \\ c_3 \\ p_{c2} \\ b_{x2} \\ b_{y2} \\ b_{w2} \\ b_{h2} \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

1. Pedestrian
2. Car
3. motorcycle

Training

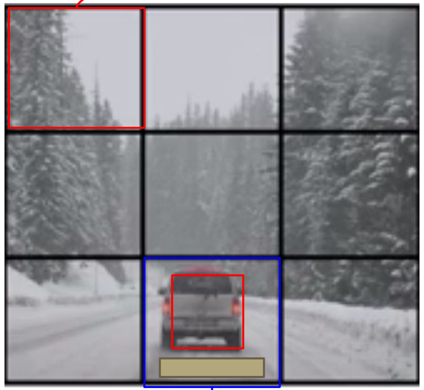


Y is : 3*3*2*8

1. Pedestrian
2. Car
3. motorcycle

$$y = \begin{bmatrix} p_{c1} \\ b_{x1} \\ b_{y1} \\ b_{w1} \\ b_{h1} \\ c_1 \\ c_2 \\ c_3 \\ p_{c2} \\ b_{x2} \\ b_{y2} \\ b_{w2} \\ b_{h2} \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \qquad y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

Training

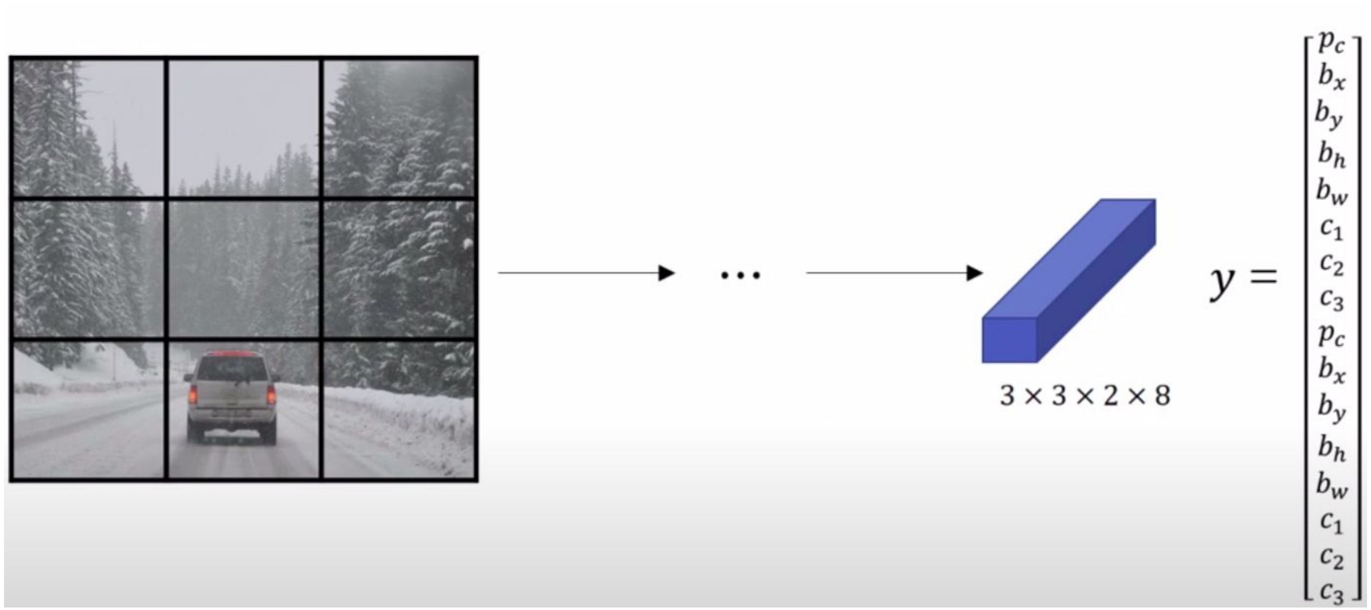


Y is : 3*3*2*8

$$y = \begin{bmatrix} p_{c1} \\ b_{x1} \\ b_{y1} \\ b_{w1} \\ b_{h1} \\ c_1 \\ c_2 \\ c_3 \\ p_{c2} \\ b_{x2} \\ b_{y2} \\ b_{w2} \\ b_{h2} \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \qquad y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \qquad y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_{x2} \\ b_{y2} \\ b_{w2} \\ b_{h2} \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

1. Pedestrian
2. Car
3. motorcycle

Making predictions

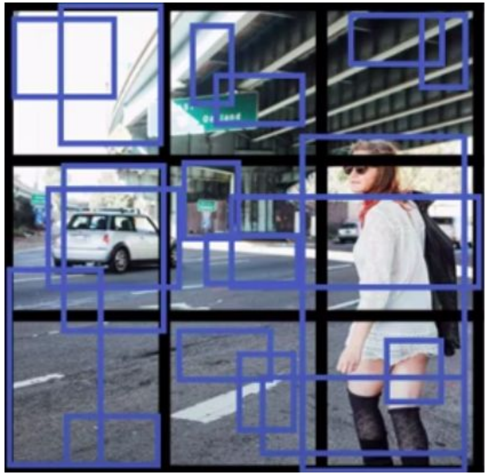


$3 \times 3 \times 2 \times 8$
$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$

Outputting the non-max suppressed outputs

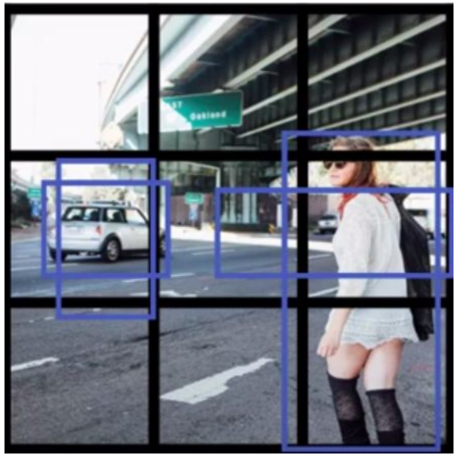
- For each grid call, get 2 predicted bounding boxes

Outputting the non-max suppressed outputs



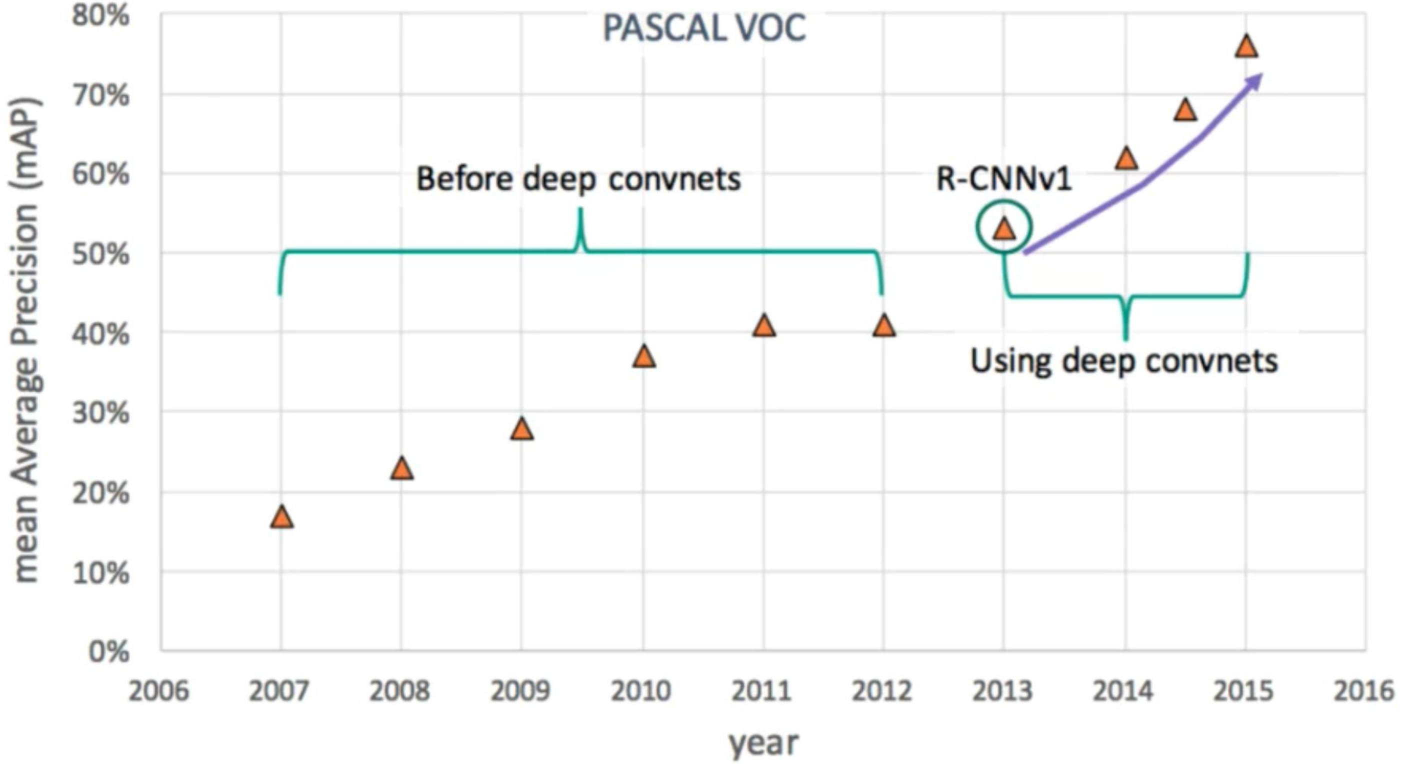
- For each grid call, get 2 predicted bounding boxes

Outputting the non-max suppressed outputs



- For each grid call, 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.

R-CNN

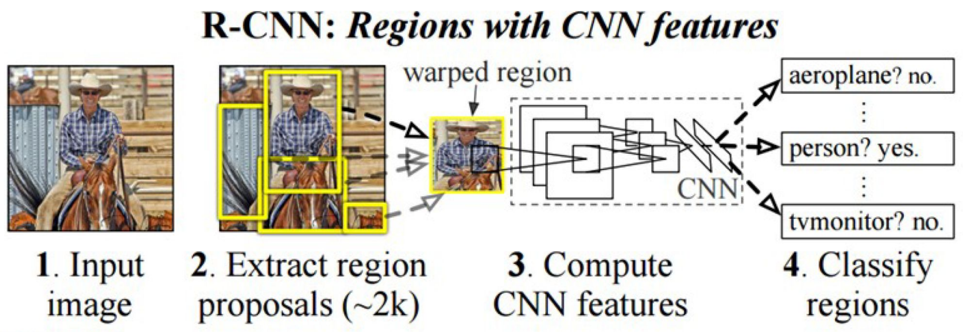

PASCAL VOC
mean Average Precision (mAP)
80%
70%
60%
50%
40%
30%
20%
10%
0%
Before deep convnets
R-CNNv1
Using deep convnets
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
year

R-CNN

- Introduced in 2014 by researcher at University College of Berkley, R-CNNs achieved the best performance in the PASCAL VOC Challenge (ImageNet for object detection testing).
- proposed a method where we use selective search to extract just 2000 regions from the image and he called them region proposals
- instead of trying to classify a huge number of regions, you can just work with 2000 regions.
- These 2000 region proposals are generated using the selective search algorithm

R-CNN

- The purpose of R-CNNs is to solve the problem of object detection. Given a certain image, we want to be able to draw bounding boxes over all of the objects. The process can be split into two general components, the region proposal (Selective Search) step and the classification step (Deep CNN).



R-CNN workflow

Selective Search

- Uses bottom-up grouping of image regions to generate a hierarchy of small to large regions
- The goal is to generate a small set of high-quality object locations
- Combines the best of the intuitions of segmentation and exhaustive search.
- Image segmentation exploits the structure of the image to generate object locations
- Exhaustive search aims to capture all possible object locations

Selective Search Algorithm

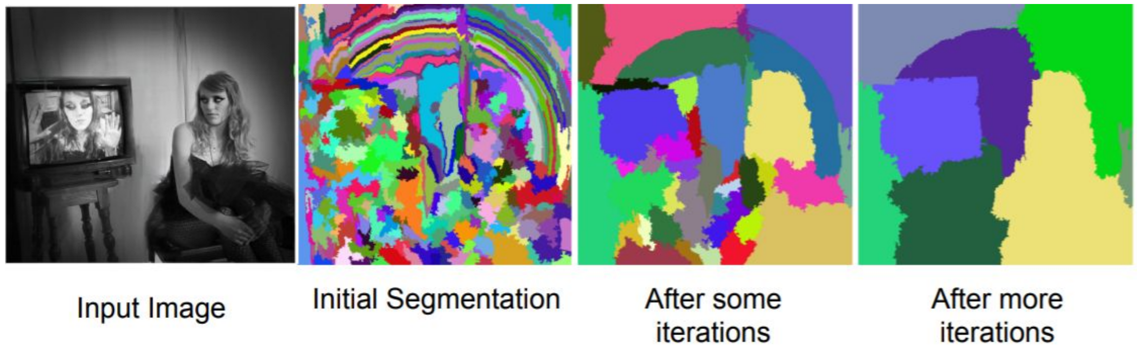
Selective Search with Exhaustive Search Step by Step working:

Step 1: Generate initial sub-segmentation. We generate as many regions, each of which belongs to at most one object.

Step 2: Recursively combine similar regions into larger ones. Here we use Greedy algorithm.

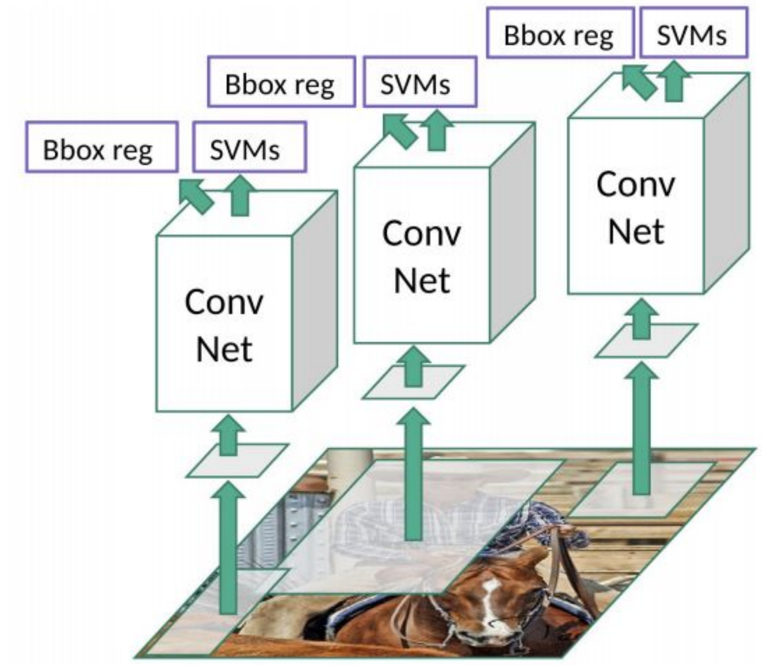
- From the set of regions, choose two regions that are most similar.
- Combine them into a single, larger region.
- Repeat until only one region remains.

Selective Search Algorithm

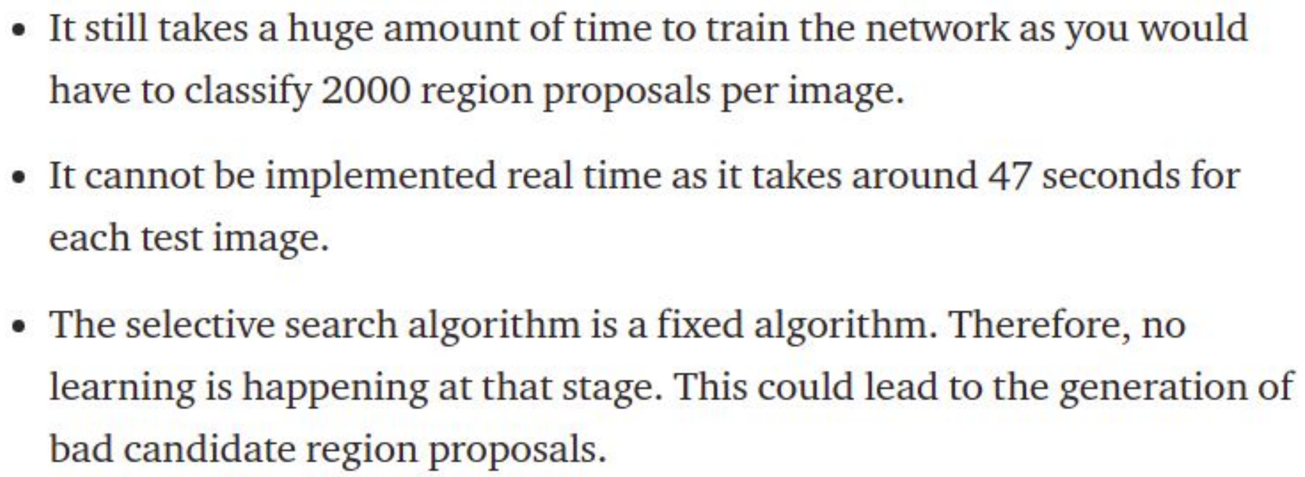


Step 3: Use the generated regions to produce candidate object locations.

R-CNN


Bbox reg
SVMs
Bbox reg
SVMs
Bbox reg
SVMs
Conv Net
Conv Net
Conv Net

Problems with R-CNN



- It still takes a huge amount of time to train the network as you would have to classify 2000 region proposals per image.
- It cannot be implemented real time as it takes around 47 seconds for each test image.
- The selective search algorithm is a fixed algorithm. Therefore, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals.