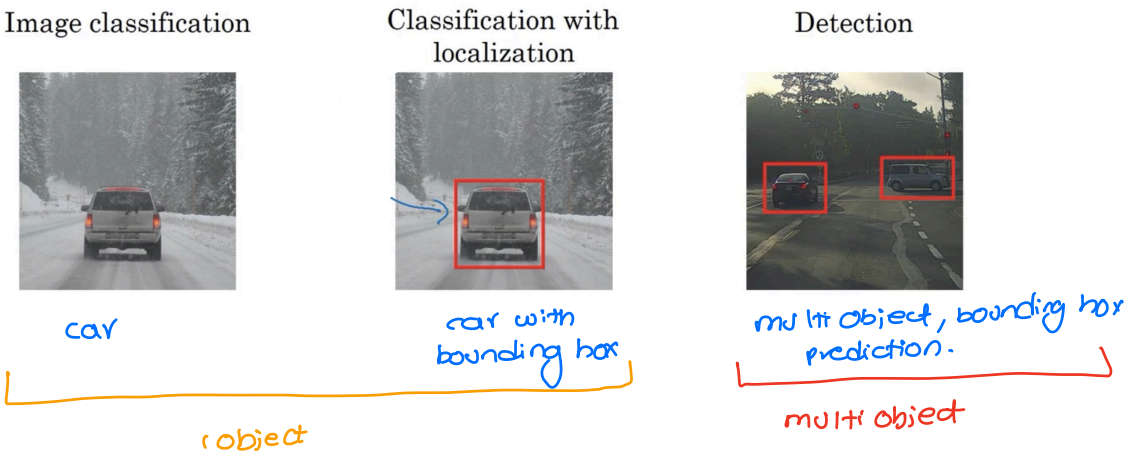
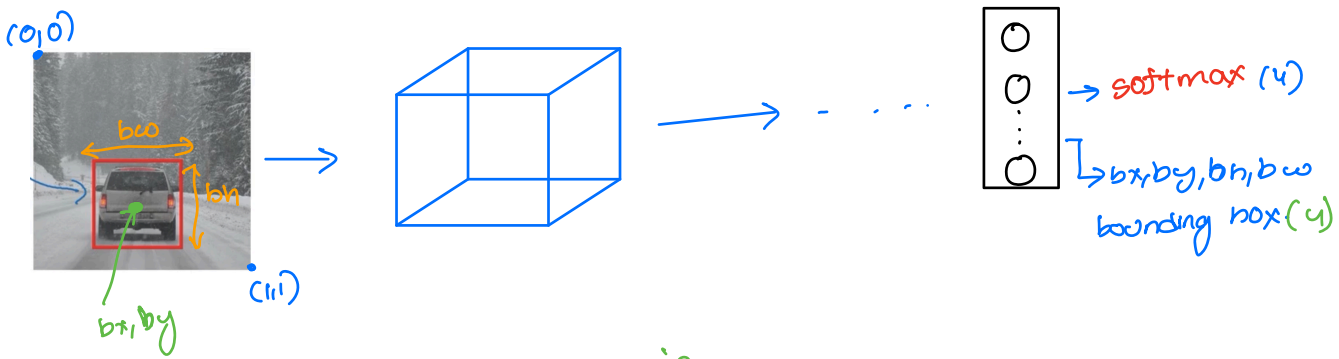
dec1: What are localization and detection:



Classification with localization:



bx = 0.5
by = 0.7
bh = 0.3
bw = 0.4

(+)

1. pedestrain
2. car
3. motorcycle
4. background

⇒ localization.

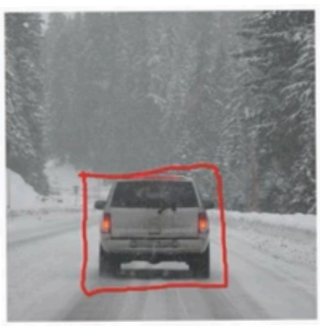
Target label:

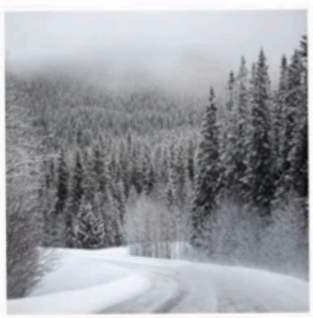
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

→ is there any object (1/0)

assume that image has one object

if x =





$y =$

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

1 → logistic regression loss.

b_x, b_y, b_h, b_w → squared error loss.

$0, 1, 0$ → log likelihood loss

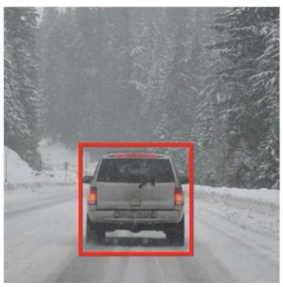
loss: if $y_1 = p_c = 1$

$$\alpha(\hat{y}, y) = \sum_{i=1}^{8} (\hat{y}_i - y)^2$$

loss if $y_1 = p_c = 0$

$$\alpha(\hat{y}, y) = (\hat{y}_1 - y_1)^2$$

dec2: dand mark detection:



b_x, b_y, b_h, b_w

output = 4

l_{1x}, l_{1y}

l_{2x}, l_{2y}

l_{3x}, l_{3y}

l_{4x}, l_{4y}

l_{68x}, l_{68y}.

(1-landmark)

output = 129

= conv 2+1

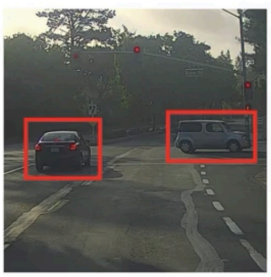
pose of person

l_{1x}, l_{1y}

l_{32x}, l_{32y}.

dec3: Object Detection:

Example: Car Detection:

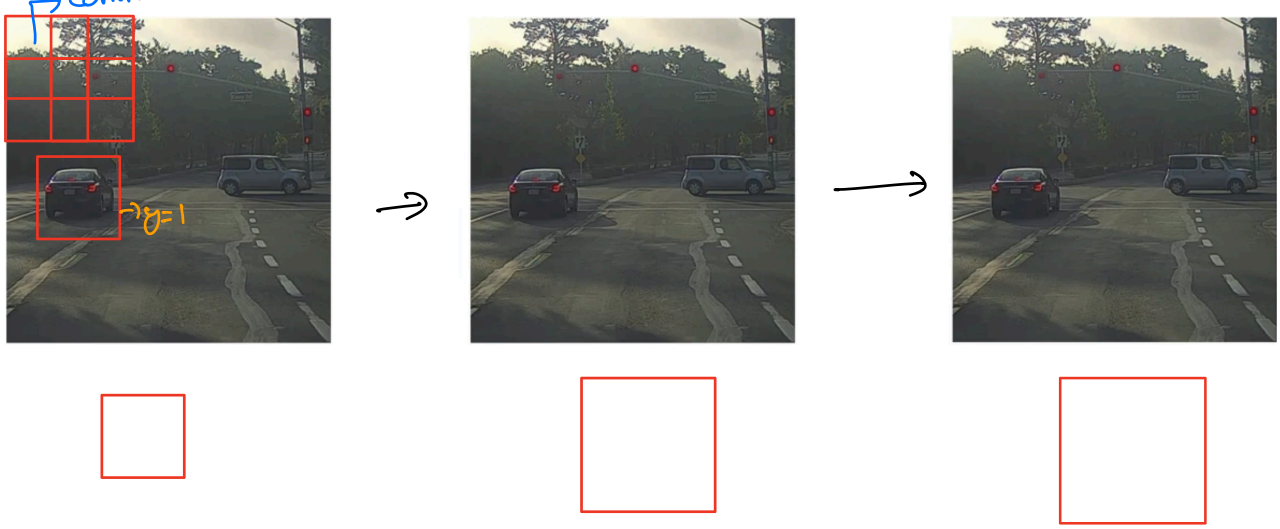


Training set:

x	y
	1
	1
	1
	0
	0

 → convnet → $y(1,0)$

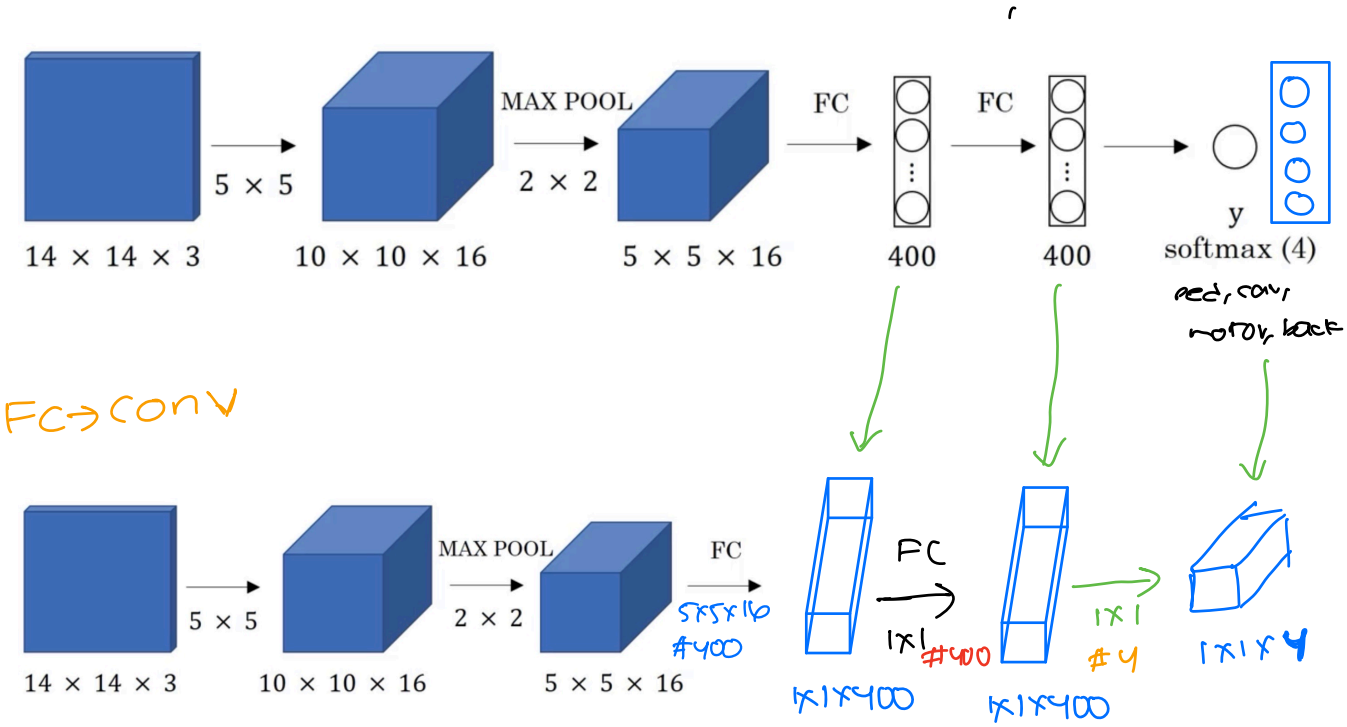
Sliding window detection



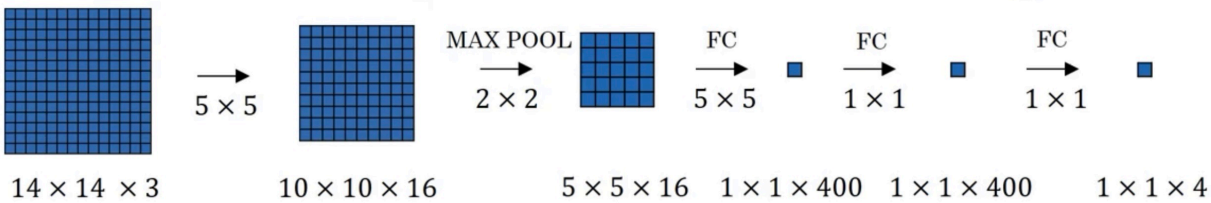
go through every region of input image and pass through convnet and increase the window size.

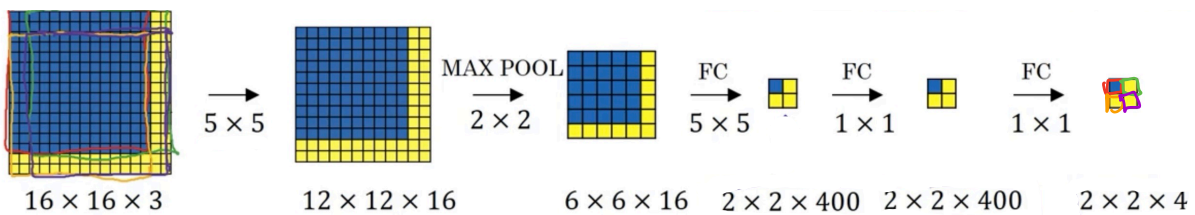
disadv: computational cost.

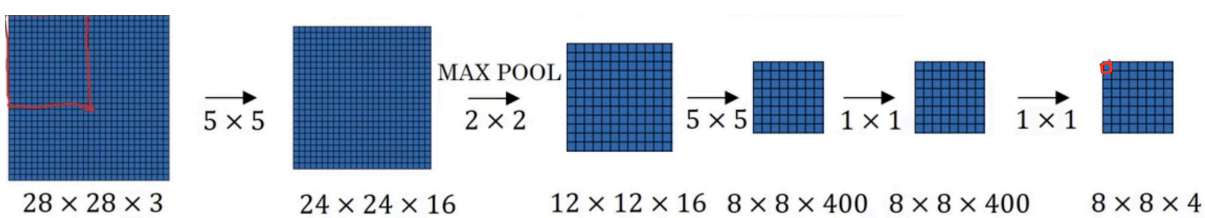
dec4: Turning fc layer into convolution layers:

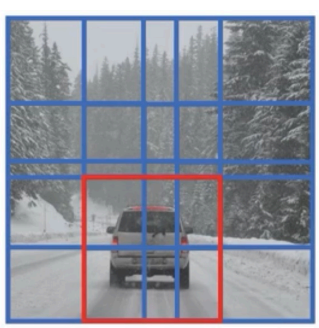


Convolution Implementation of sliding window:









pro:

sliding window problem can be solved by a single forward pass convolution network.

cons:

- bounding box position, not accurate.

dec5: Output accurate bounding boxes:

YOLO Algorithm: (2015)

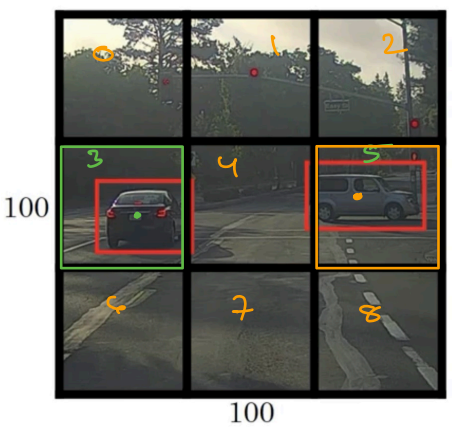
you only look once.

- real time object detection.

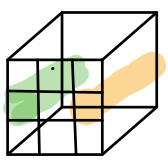
lables for training

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_w \\ b_h \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \quad (2) \; y = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_w \\ b_h \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad (4) \; y = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_w \\ b_h \\ 0 \\ 1 \\ \\ 0 \end{bmatrix}$$

(y)



$100 \times 100 \times 3$ → conv net. → $3 \times 3 \times 8$



$3 \times 3 \times 8$

outputs:

→ conv implementation, much less compute and efficient

→ precise bounding box instead realying on the stride and filter size.

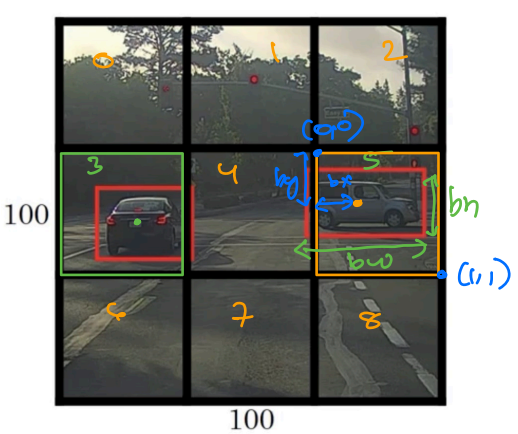
→ use bit finner grid (19x19)

→ outputs (19x19x8)

problem:

multiple object in a single grid. (use finner grid).

Specify the bounding boxes:



$$y = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.4 \\ 0.3 \\ 0.5 \\ 0.9 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

(fraction of cell grid height)

(fraction of cell grid weight)

$b_x, b_y < 1$

b_h, b_w could be > 1

dec 6: Intersection Over Union: (IoU)

- gt
- predicted

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

correct if IoU ≥ 0.5 (0.6)

→ measure of overlap between two bounding boxes

dec 7: Non Max Suppression:

19x19 grid

19x19x5

Each prediction: $\begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \end{bmatrix}$

if multiple objects (c_1, c_2, c_3) do non max suppression for all the classes.

① Discard all boxes with $P_c \leq 0.6$

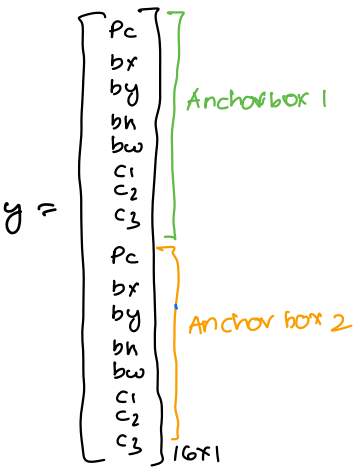
② while there are any remaining boxes:

- Pick the box with largest P_c output as prediction.
- Discard any remaining box with IoU ≥ 0.5 with the box output in the previous step.

Lec 8: Anchor Box

Anchor box 1:

Anchor box 2:



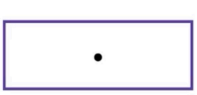
→ Each object in training image is assigned to grid cell that contains object's midpoint and anchor box for the grid cell with highest IOU.

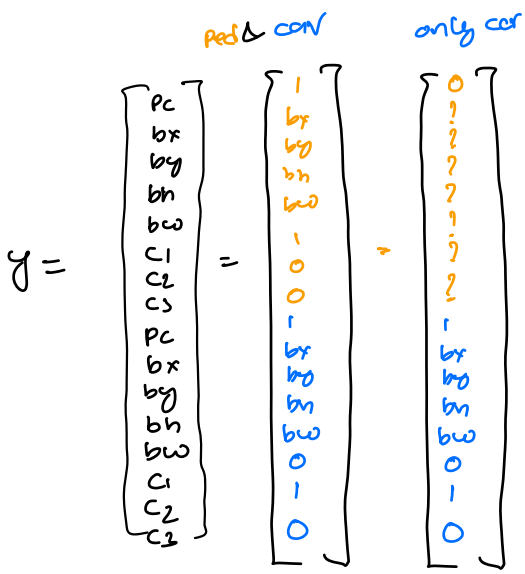
(grid cell, anchor box, object).

output : $(3 \times 3 \times 16) \rightarrow (3 \times 3 \times 2 \times 8)$

Anchor box 1:

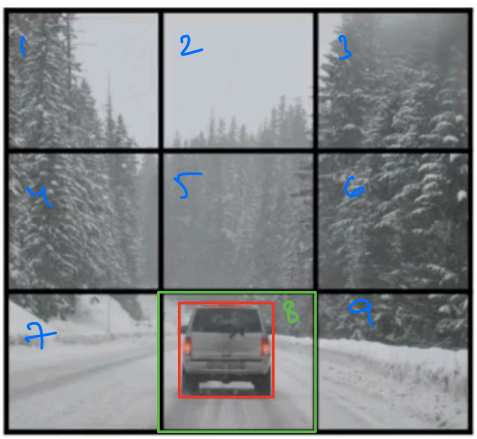
Anchor box 2:





deca: complete Yolo algorithm:

Training:



$X = 100 \times 100 \times 3$

classes:

1. pedestrian
2. car
3. motorcycle

Anchor box 1:

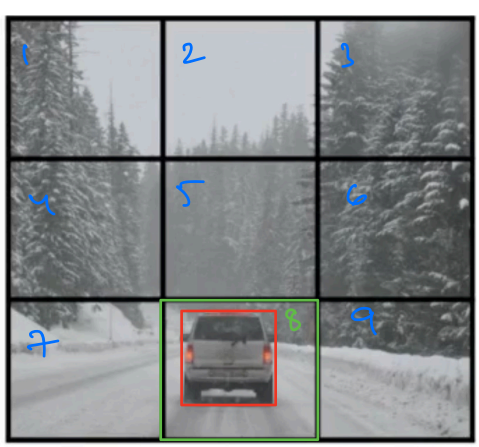
Anchor box 2:

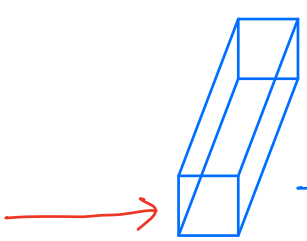
$y =$ [pc, bx, by, bh, bw, c_1, c_2, c_3, pc, bx, by, bh, bw, c_1, c_2, c_3] $=$ [1, 2, 3, 4, 5, 6, 7, 9] $=$ (8)

[0, ?, ?, ?, ?, ?, ?, ?, 1, bx, by, bh, bw, 0, 1, 0]

$y = 3 \times 3 \times 16$ output size

Predictions:



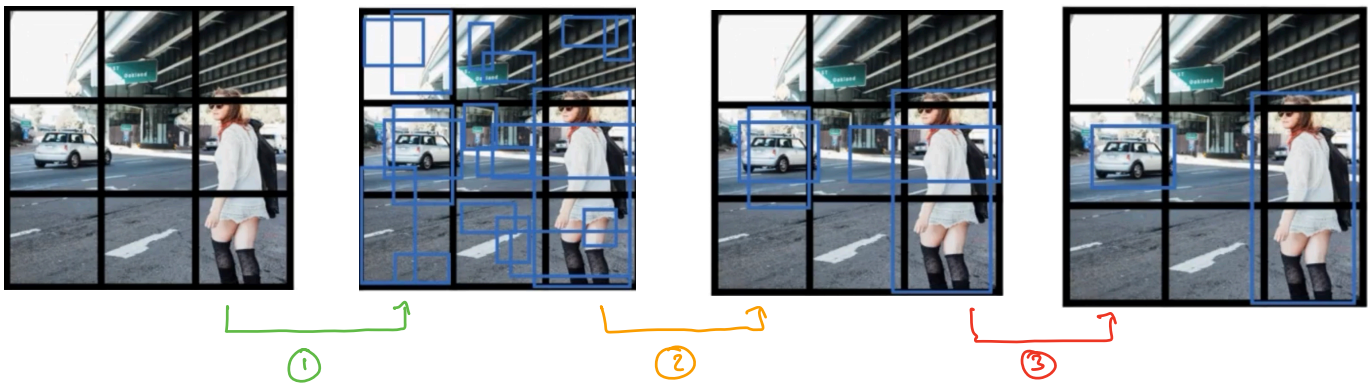


$3 \times 3 \times 16$

$y =$ [1, 2, 3, 4, 5, 6, 7, 9] (8)

[0, ..., 1, bx, by, bh, bw, 0, 1, 0]

Outputting the non-max suppressed outputs:

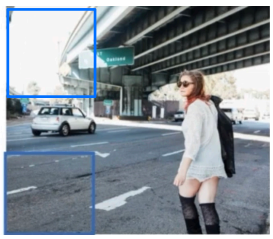


① For each grid cell, get 2 predictions bounding boxes

② Get rid of low probability predictions

③ for each class (ped, car, motorcycle) use non-max suppression to generate final prediction.

dec10: Region Proposals: (R-CNN).



unwanted regions

CNN run's on unwanted regions.

segmentation

run the classifier on segmentation output blobs.

RCNN: propose regions. classify proposed regions one at a time. output label + bounding box.

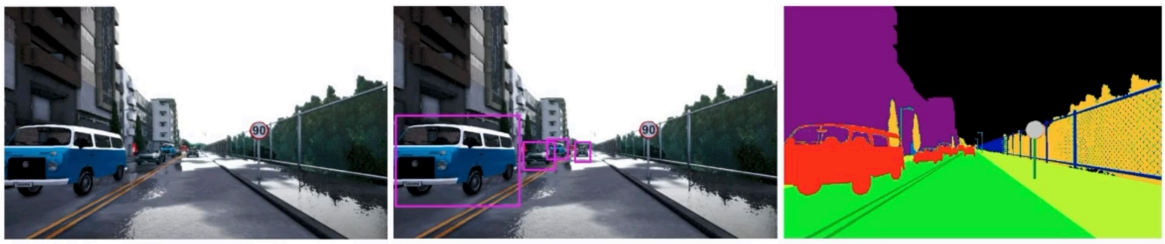
cons: quiet slow.

Fast R-CNN: propose regions. use convolution implementation of sliding windows to classify all the proposed regions.

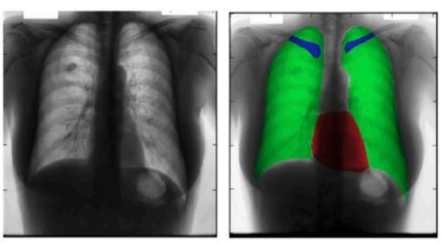
cons: clustering step in propose regions is slow.

Faster R-CNN: Use convolutional network to propose regions. (2016)

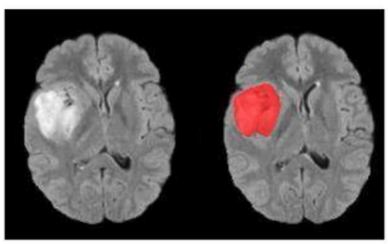
dec11: Semantic Segmentation with UNet:



Input image | Object Detection | Semantic Segmentation

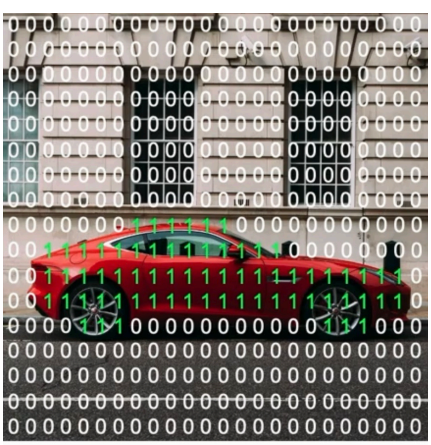


Chest X-Ray

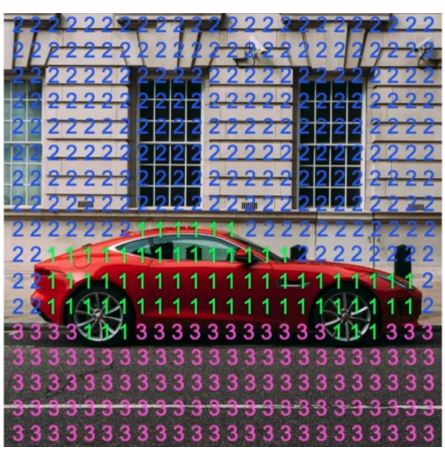


Brain MRI

Per Pixel Class labels: (segmentation Map)

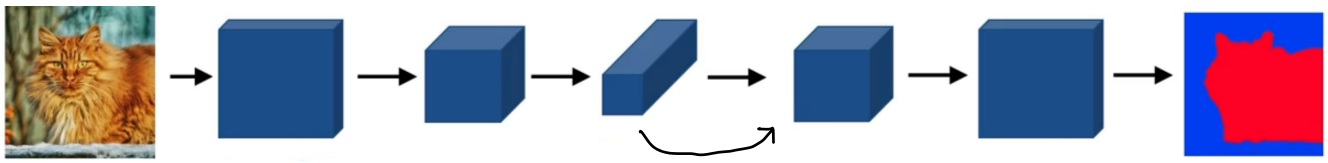


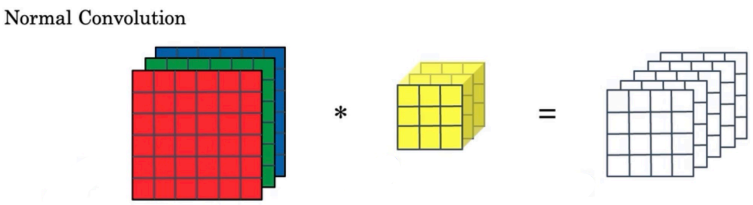
1. car
0. background



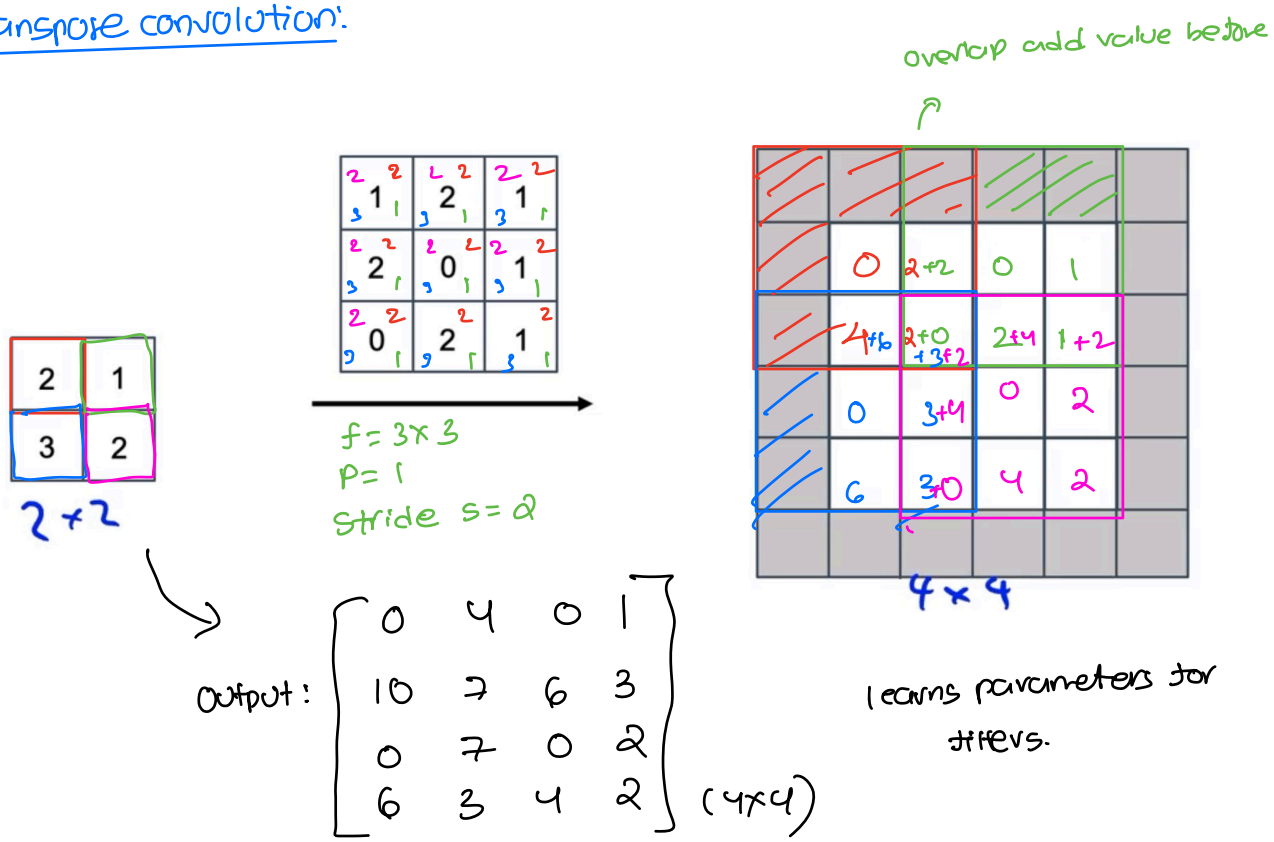
1. car
2. building
3. road

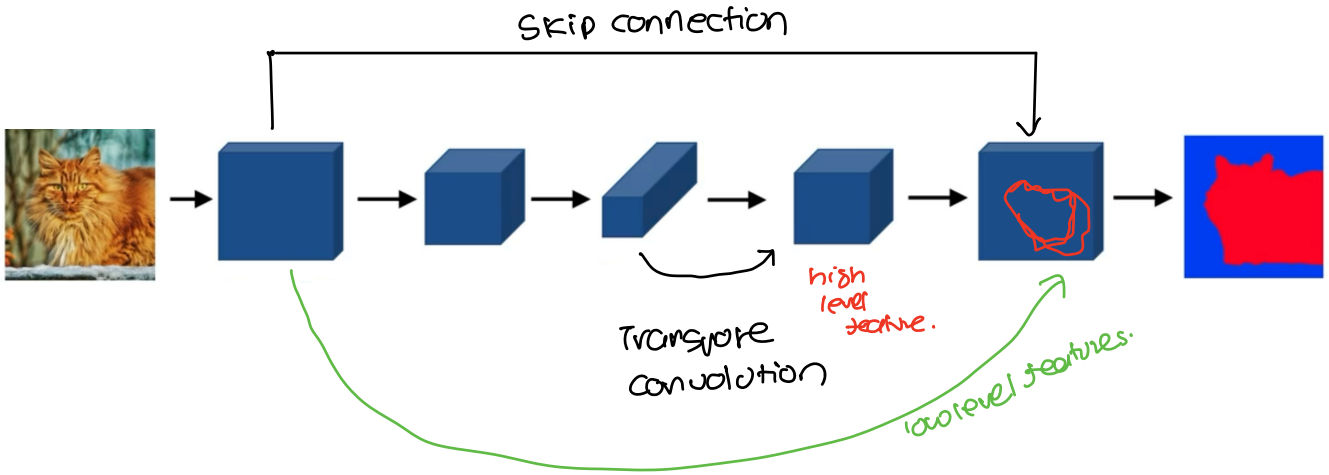
Architecture of Semantic Segmentation:





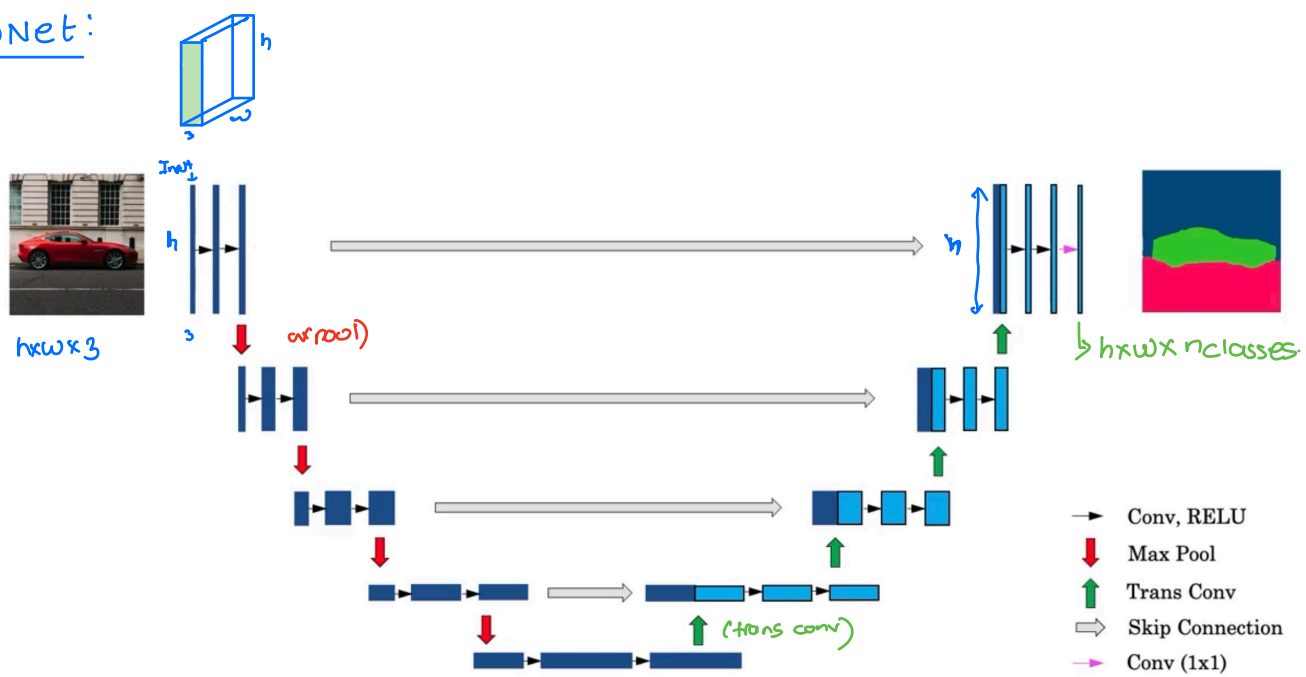
Transpose convolution:





gives the NN, low features into directly, low level features- textures
high level features- where the cat is.

UNet:



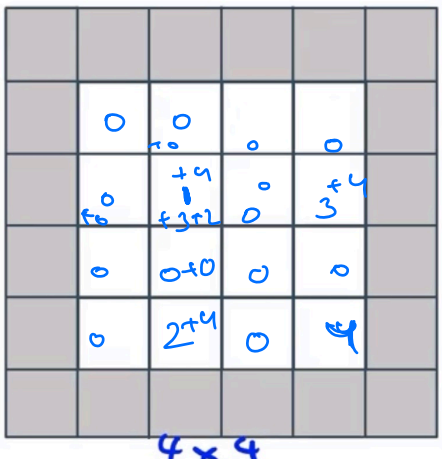
1 3

2 4

1 0 1

0 0 0

1 0 1



4 0 4

0 0 0

4 0 4

x=10 y=0 z=6

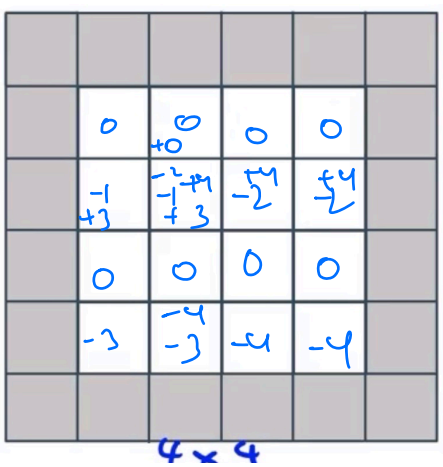
1 2

3 4

1 1 1

0 0 0

-1 -1 -1



z=-7

x=0

y=2

$\frac{12}{20+8} = \frac{12^{3}}{28_{7}} \quad \frac{4}{16+12}$

$\frac{4^{1}}{28_{7}}$