Copyright Notice

deeplearning.ai

Object Detection

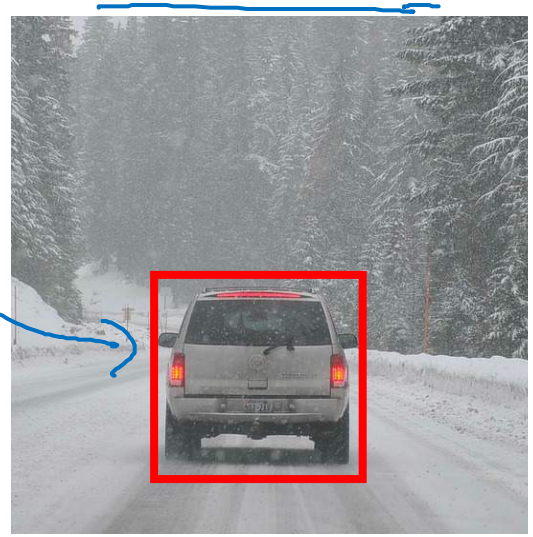
Object localization

What are localization and detection?

Image classification

"Car"

Classification with localization



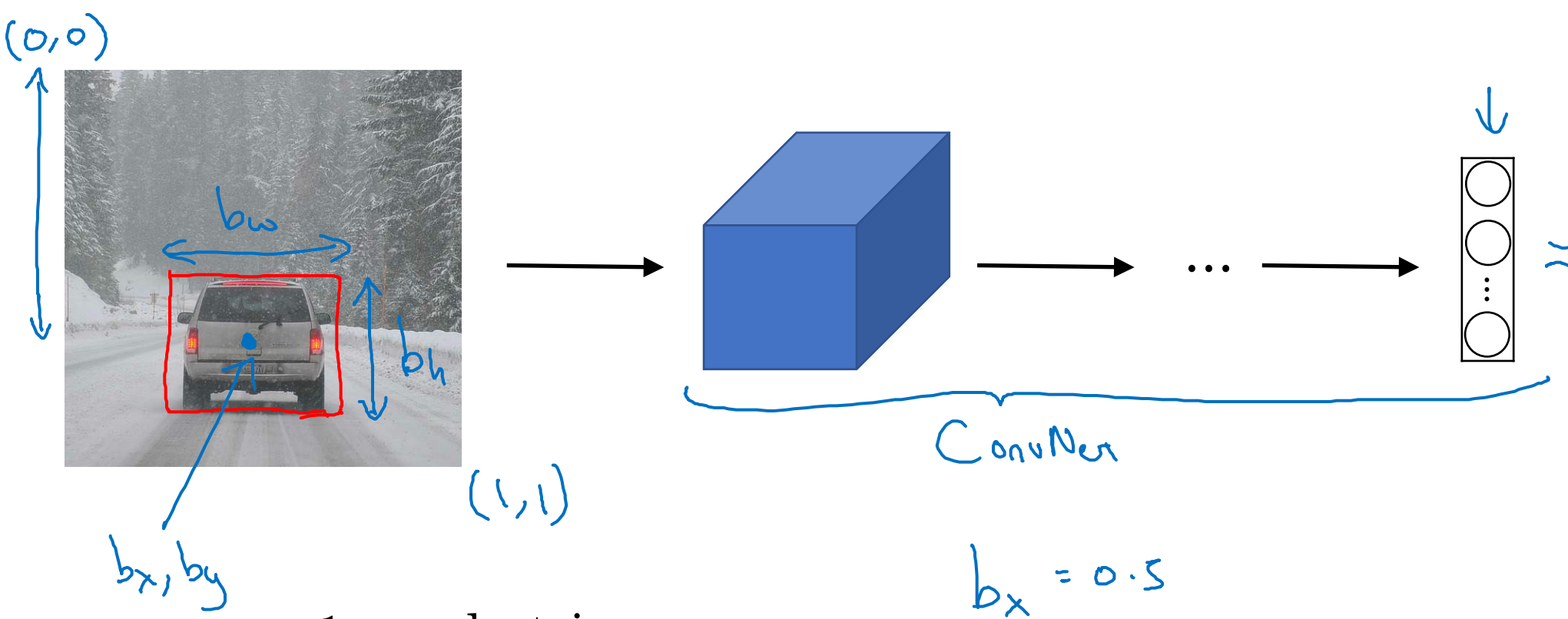
"Car"

1 object

Detection

multiple objects

Classification with localization



1 - pedestrian
2 - car
3 - motorcycle
4 - background

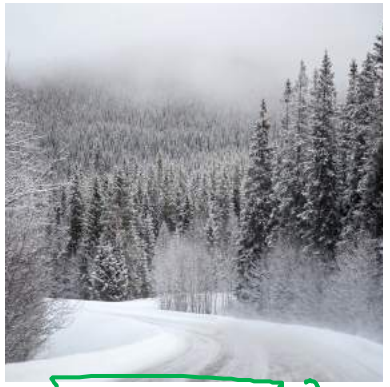
$b_x = 0.5$
$b_y = 0.7$
$b_h = 0.3$
$b_w = 0.4$

Defining the target label y

1 - pedestrian
2 - car
3 - motorcycle
4 - background

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

$x =$



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

p_c: is there an object?

$$(x, y) \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

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

p_c

← "don't care"

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

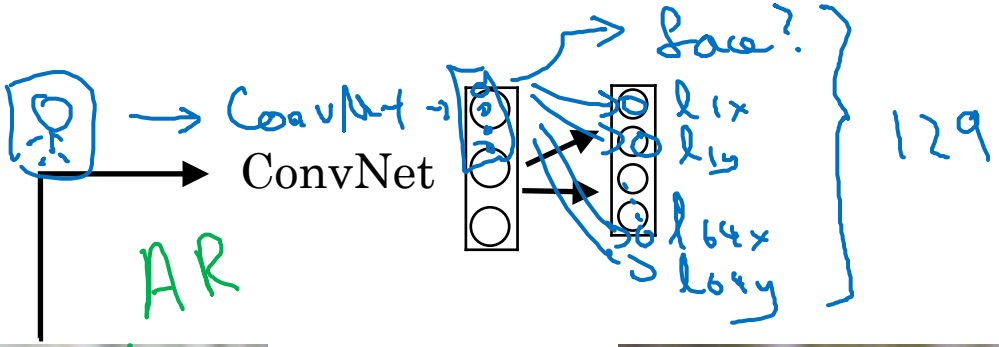
Andrew Ng

deeplearning.ai

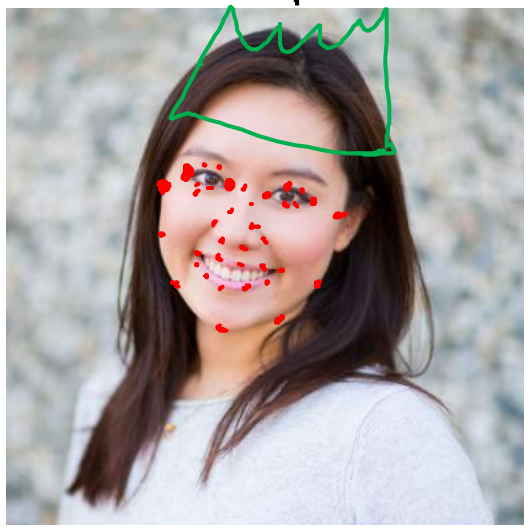
Object Detection

Landmark detection

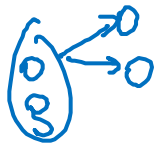
Landmark detection



b_x, b_y, b_h, b_w



$l_{1x}, l_{1y},$
$\vdots$
l_{32x} l_{32y}



$l_{1x}, l_{1y},$
$l_{2x}, l_{2y},$
$l_{3x}, l_{3y},$
l_{4x}, l_{4y}
$\vdots$
l_{64}, l_{64y}

x, y

deeplearning.ai

Object Detection

Object detection

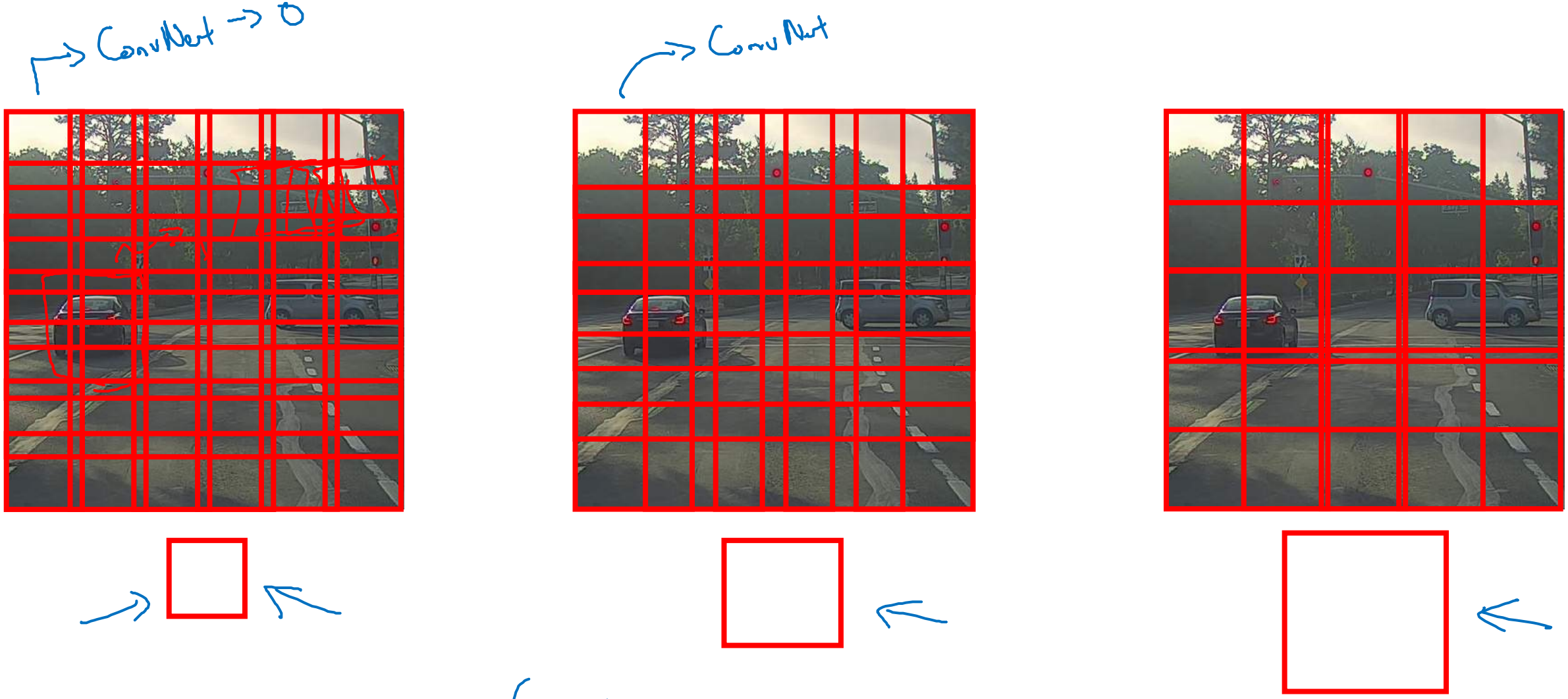
Car detection example

Training set:

x	y
	1
	1
	1
	0
	0

→ ConvNet → y

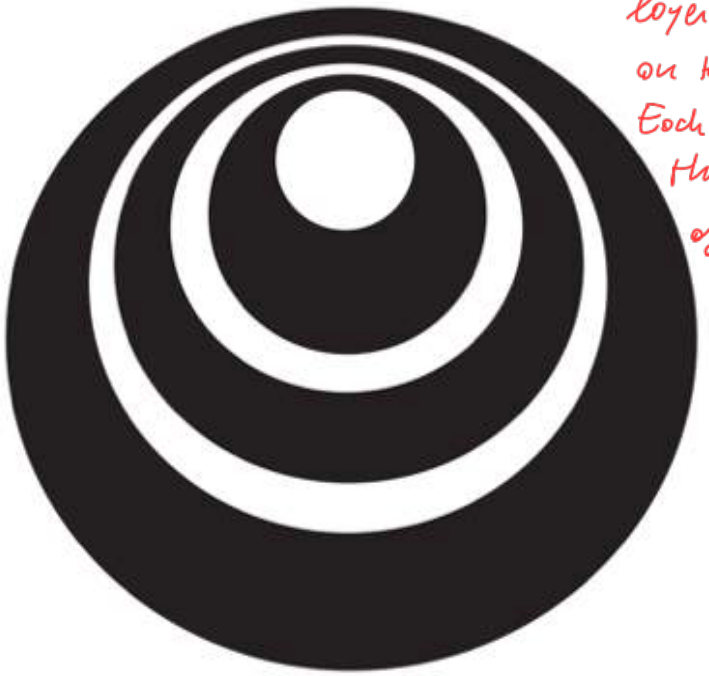
Sliding windows detection



Idea: instead of 1-training a CNN on images of cars precisely cropped and 2- at prediction time operate multiple predictions on many crops (sliding window) of a test image, opt for this alternative.

↳ Design the CNN with filters and strides of a specific size so that the final layers are not standard FC, but still 2D layers corresponding to various locations on the image. Each pixel on this final layer will correspond to a specific FC neuron due to the properties of the convolution.

↳ idea: conv 5,5 on a 5x5 layer corresponds to an FC operation.

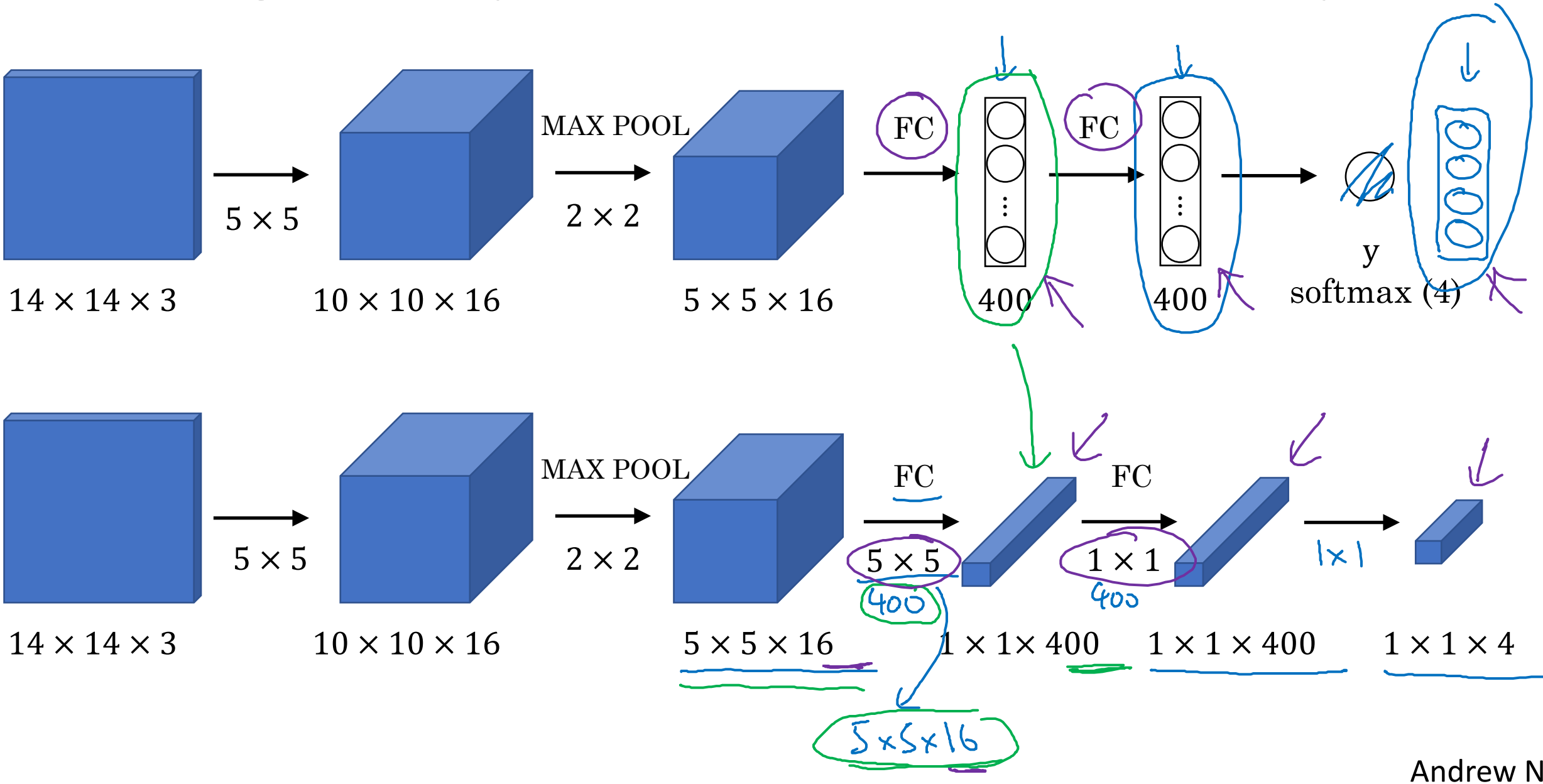


deeplearning.ai

Object Detection

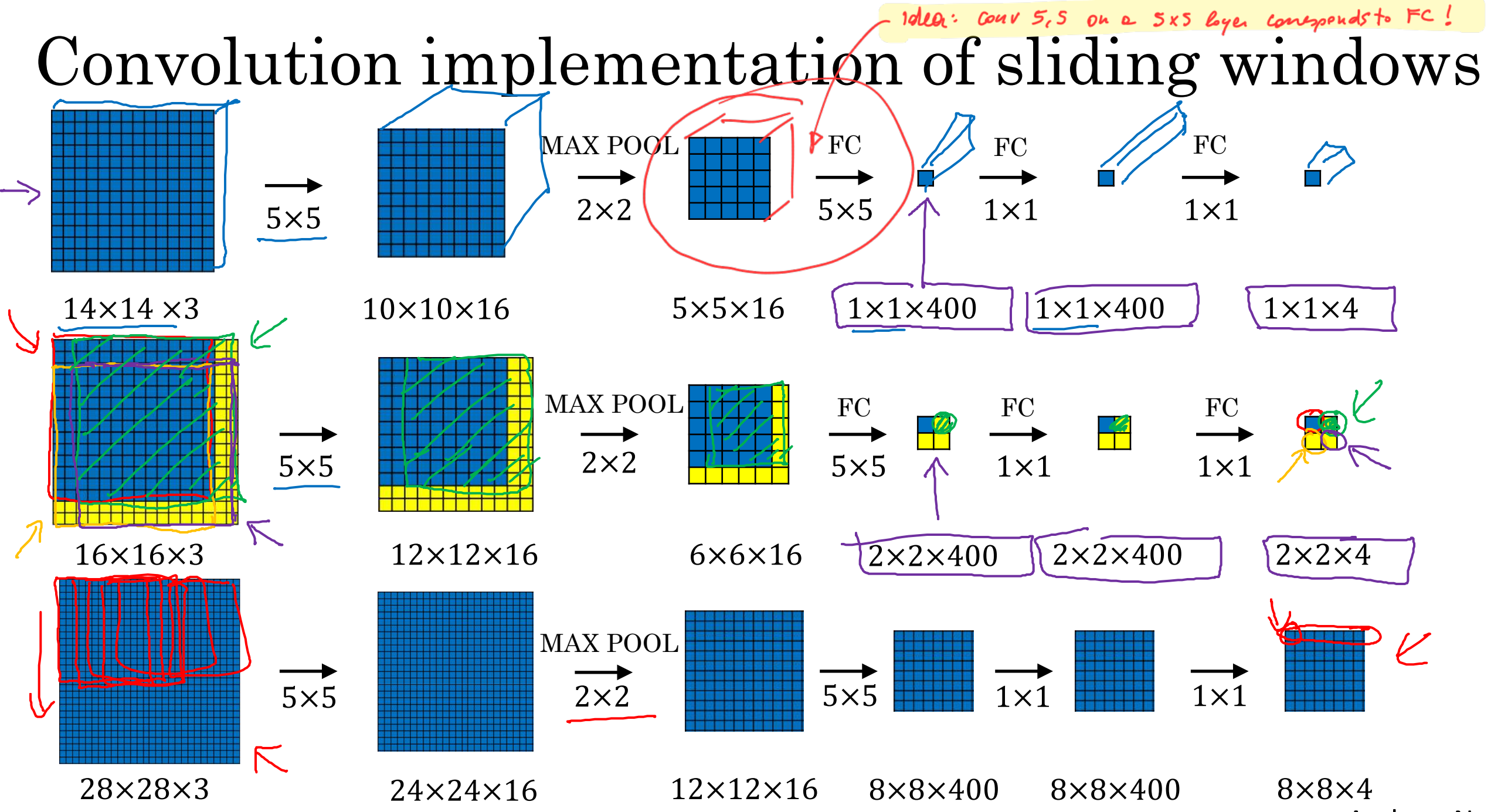
Convolutional implementation of sliding windows

Turning FC layer into convolutional layers



Andrew Ng

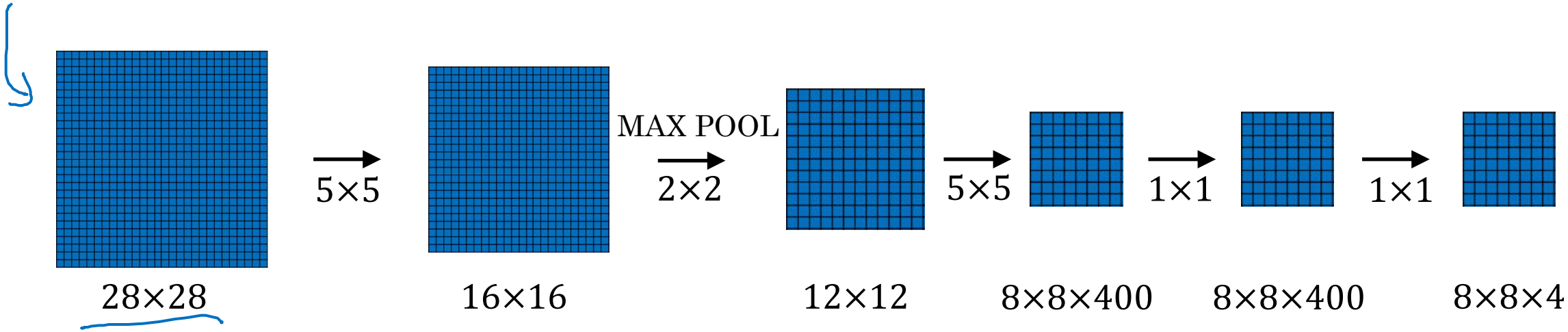
Convolution implementation of sliding windows

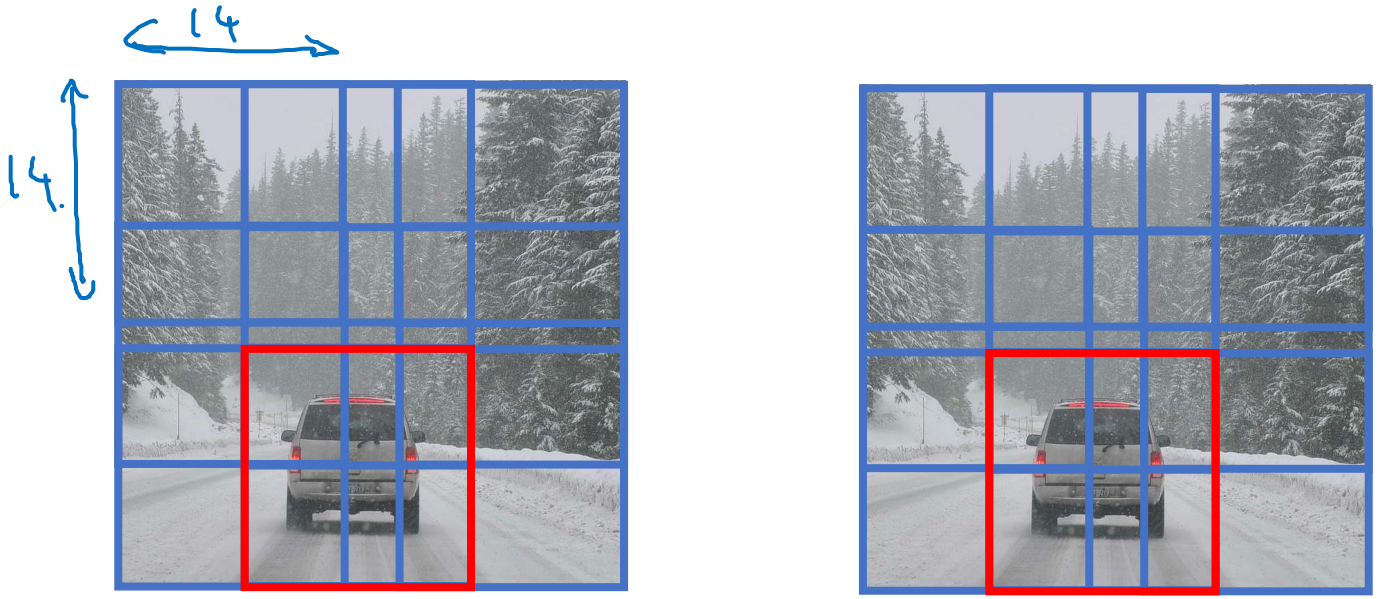


[Sermanet et al., 2014, OverFeat: Integrated recognition, localization and detection using convolutional networks]

Andrew Ng

Convolution implementation of sliding windows





The YOLO architecture allows for a pretty efficient object detection solution. The idea is to integrate the simultaneous object recognition and localization task described before ($\begin{bmatrix} 1 \\ BB_x \\ BB_y \\ BB_w \\ BB_h \\ CAT_0 \\ CAT_1 \\ CAT_2 \end{bmatrix}$) with the convolutional implementation of the sliding window approach. IN PRACTICE, this means designing an architecture that outputs an 3x3x8 output that implicitly carries out simultaneous object recognition and localization in 3x3 distinct non overlapping squares of the image.

deeplearning.ai

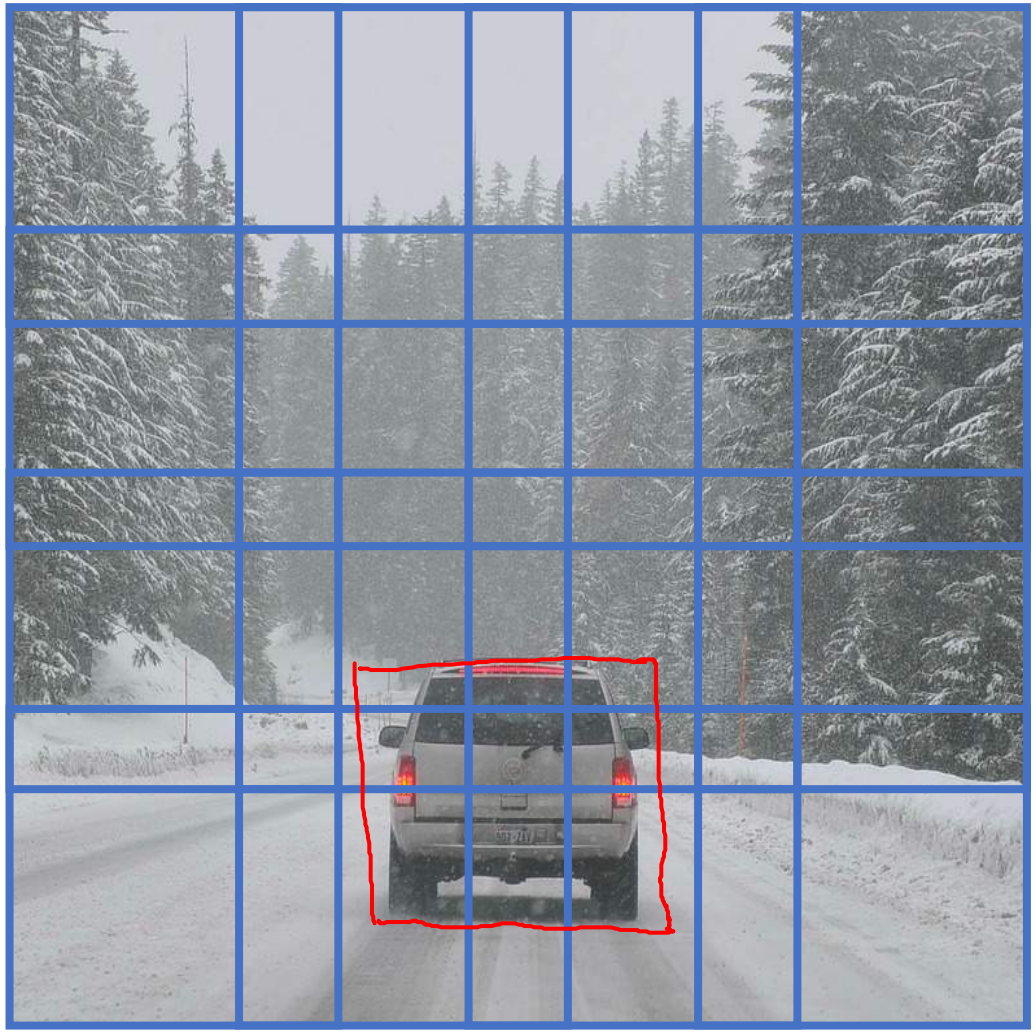
Object Detection

Bounding box predictions

OUTPUT is NOT FC or 8x1 but 3x3x1! (actually in YOLO is $19 \times 19 \times 8$)

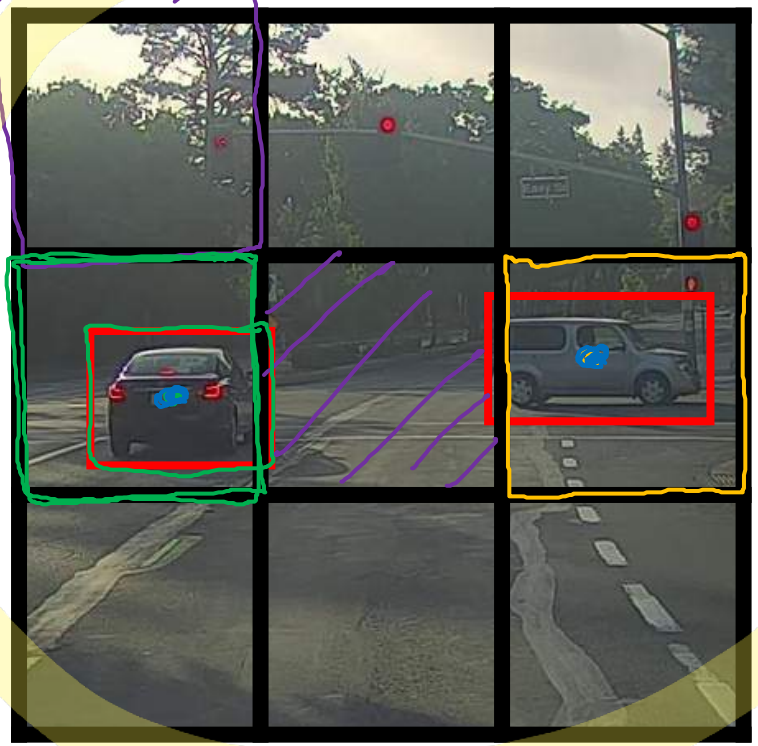
The point of dividing the image in these parts is to avoid that one object appears twice in one section.

Output accurate bounding boxes



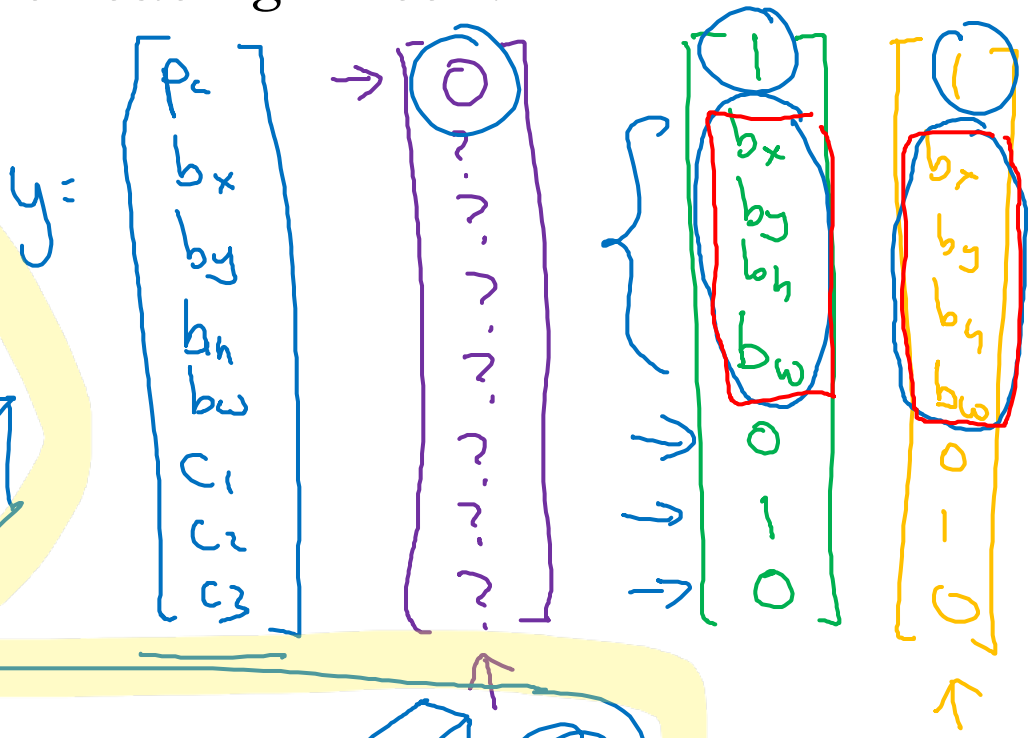
Andrew Ng

YOLO algorithm



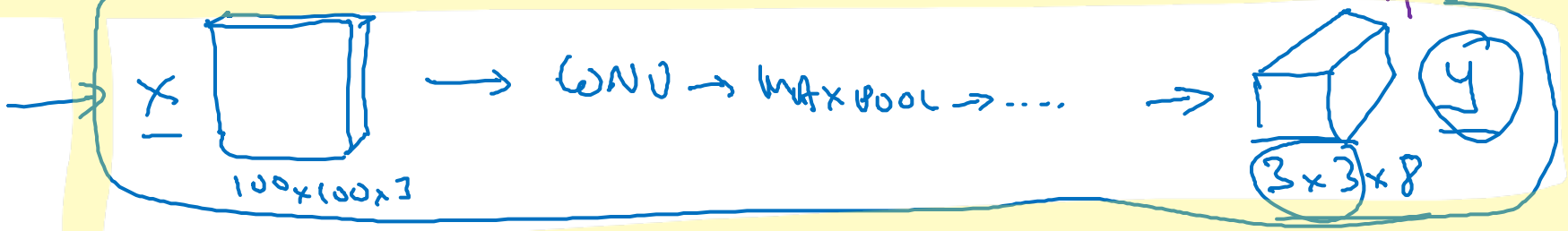
Labels for training

For each grid cell:



Target output:

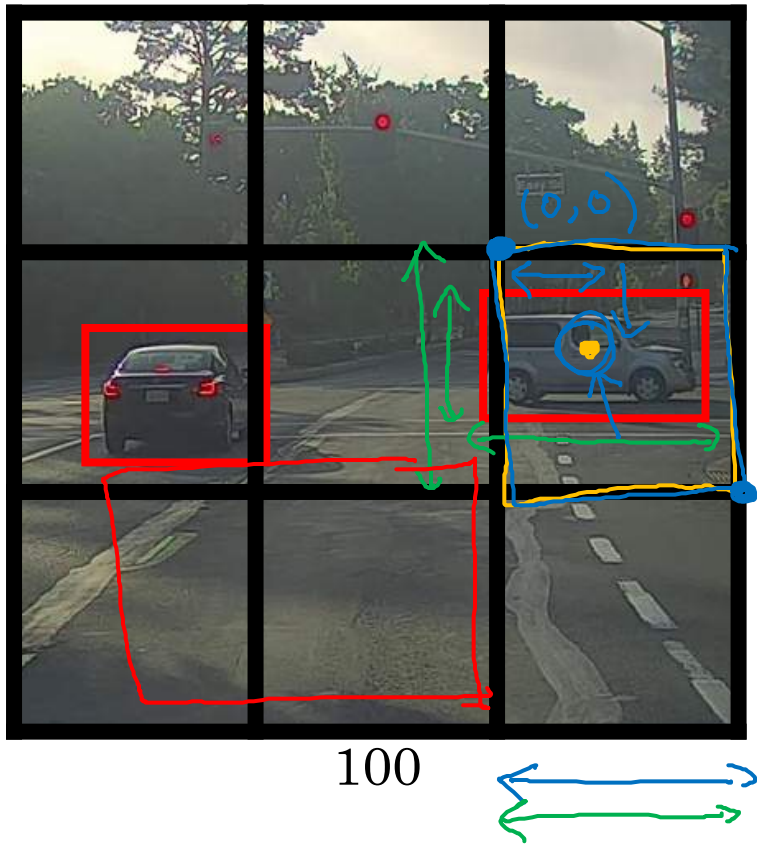
3x3x8



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

Andrew Ng

Specify the bounding boxes



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

0.4, 0.3 } between 0 and 1

0.5, 0.9 } could be > 1

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

How to evaluate performance of object detection?

By comparing true and predicted bounding boxes using the intersection over union function.

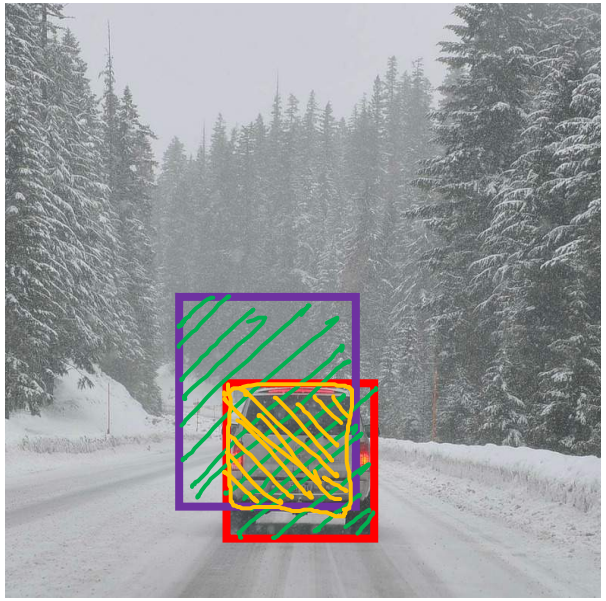
If IoU ≥ 0.5 is ok (heuristically...).

deeplearning.ai

Object Detection

Intersection over union

Evaluating object localization



Intersection over Union (IoU)

$$= \frac{\text{Size of } \square}{\text{Size of } \square}$$

"Correct" if IoU $\geq$ 0.5 ←

0.6 ←

More generally, IoU is a measure of the overlap between two bounding boxes.

Andrew Ng

HOW TO OBTAIN ONLY ONE BB FOR EACH OBJECT?

By analyzing the image in cells, the object detection algorithm may come up with multiple BBs for each object (one BB created for each grid cell that covers the object).

deeplearning.ai

Object Detection

The solution is a "post-processing" elimination of overlapping BB using the Non-max suppression.

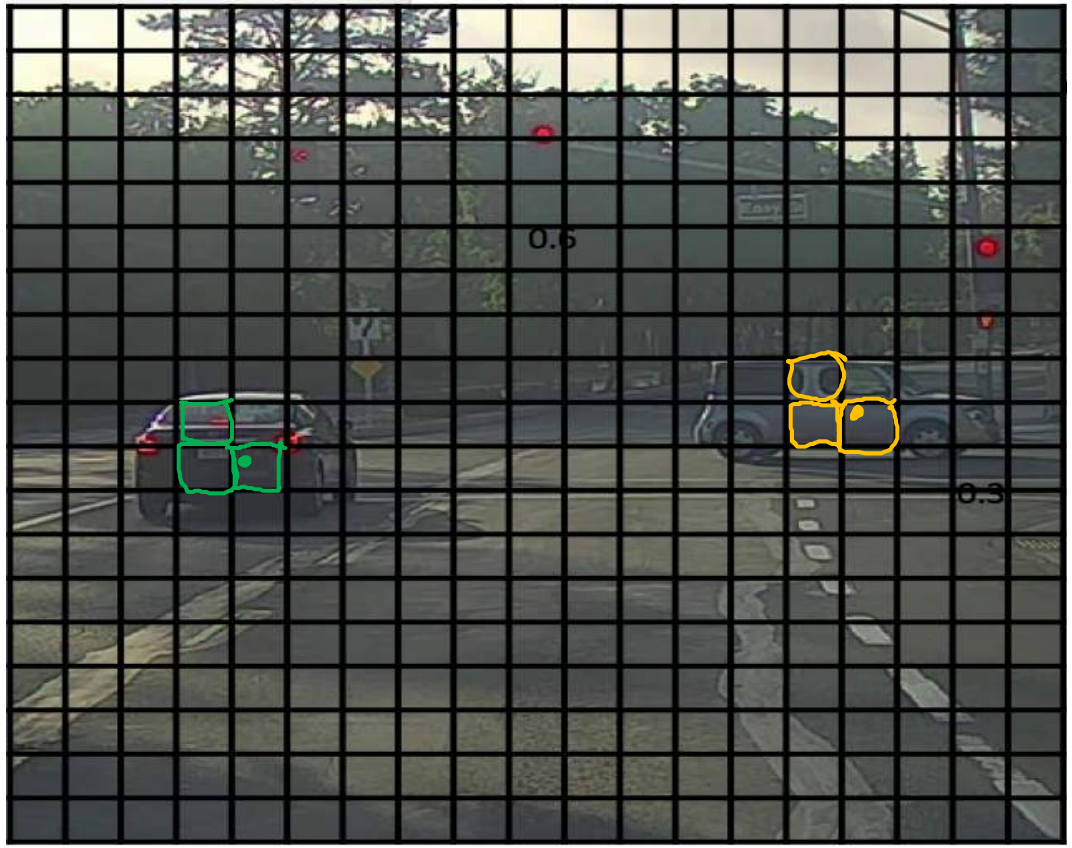
The idea is to take all the BB found in the image for a certain class and then iteratively

Non-max suppression

take the one with the most confident prediction and eliminate all the BBs overlapping with it; then continue with this process until all the overlaps are solved.

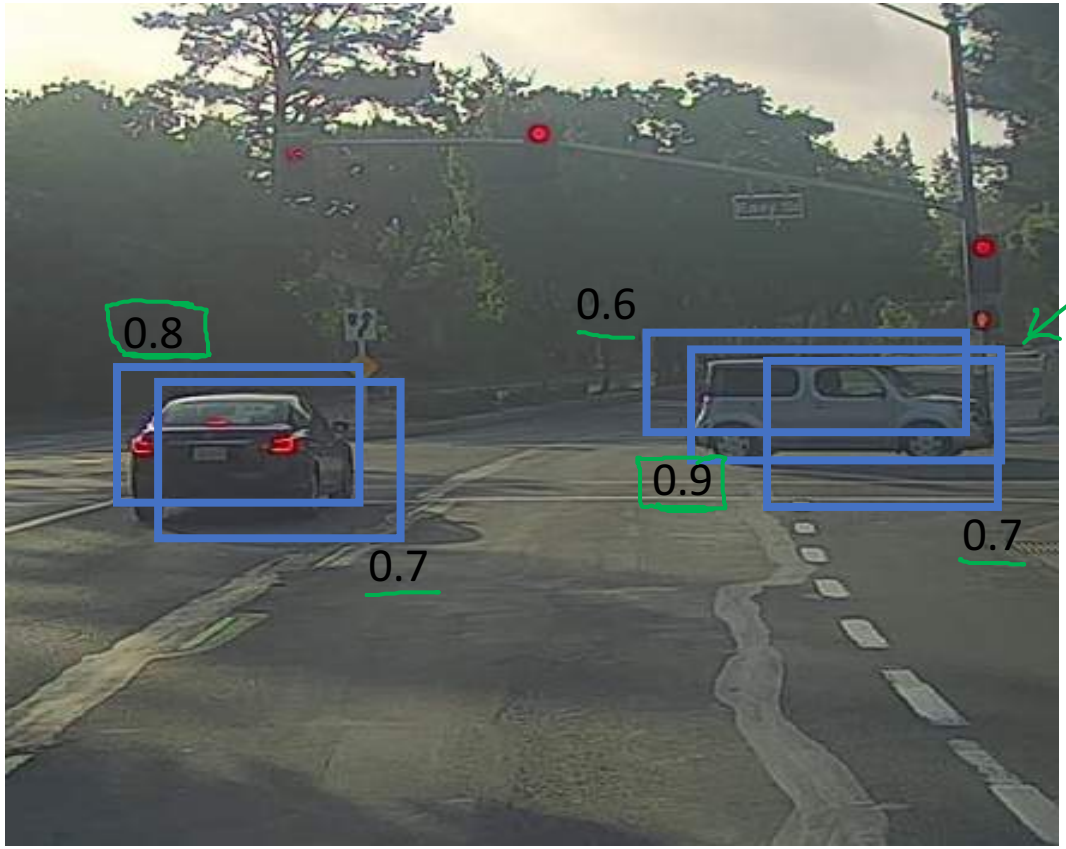
Non-max suppression example

Non-max suppression example



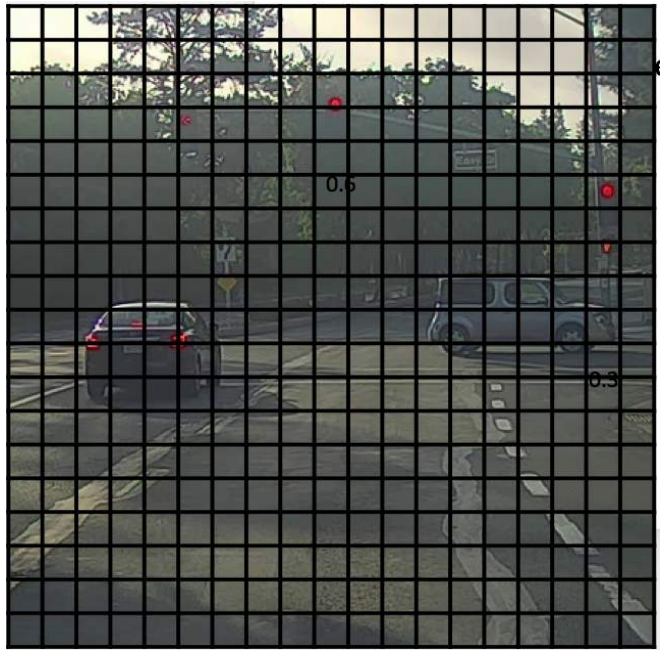
19×19

Non-max suppression example



p_c

Non-max suppression algorithm



19×19

Each output prediction is:

$$\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \end{bmatrix}$$

Discard all boxes with $p_c \leq 0.6$

While there are any remaining boxes:

- Pick the box with the largest p_c
  Output that as a prediction.
- Discard any remaining box with IoU ≥ 0.5 with the box output in the previous step

Andrew Ng

How to detect multiple objects in each grid cell?

The idea is to set the network to try identify n (2 in this example) distinct objects in each grid cell of the image. It is assumed that the n objects have very different shapes, and can therefore be distinguished from one another by setting up the network to look up for bounding boxes of very different shapes (referred to as anchor boxes) in each grid cell.

Object Detection

deeplearning.ai

Anchor boxes

? It remains unclear to me how these boxes of different shapes are parametrized within the network.

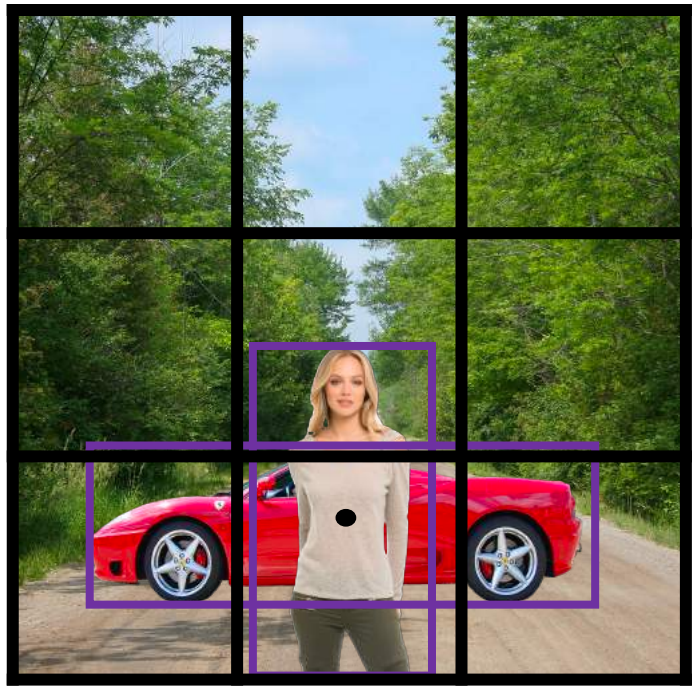
I guess that the distinction in anchor boxes shapes is encoded in the training labels and learned by the network. That is, the training data is labelled st

$$y = \begin{bmatrix} I \\ I \end{bmatrix}$$

← tall and skinny object reported here (if any)

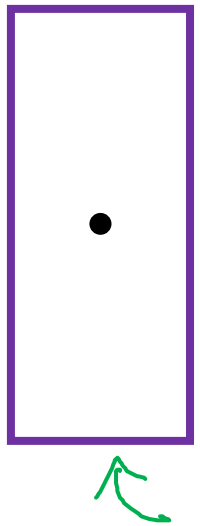
← short and wide object reported here (if any)

Overlapping objects:

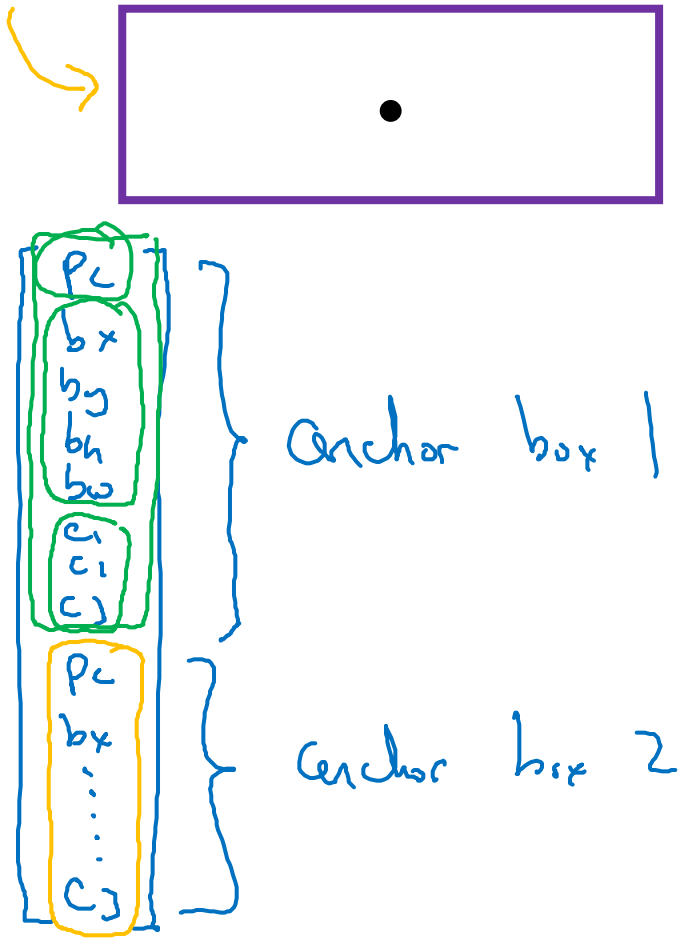


$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Anchor box 1:



Anchor box 2:



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

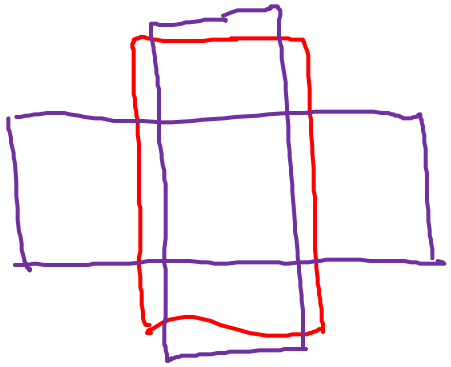
Anchor box algorithm

Previously:

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

Output y:

$3 \times 3 \times 8$



With two anchor boxes:

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

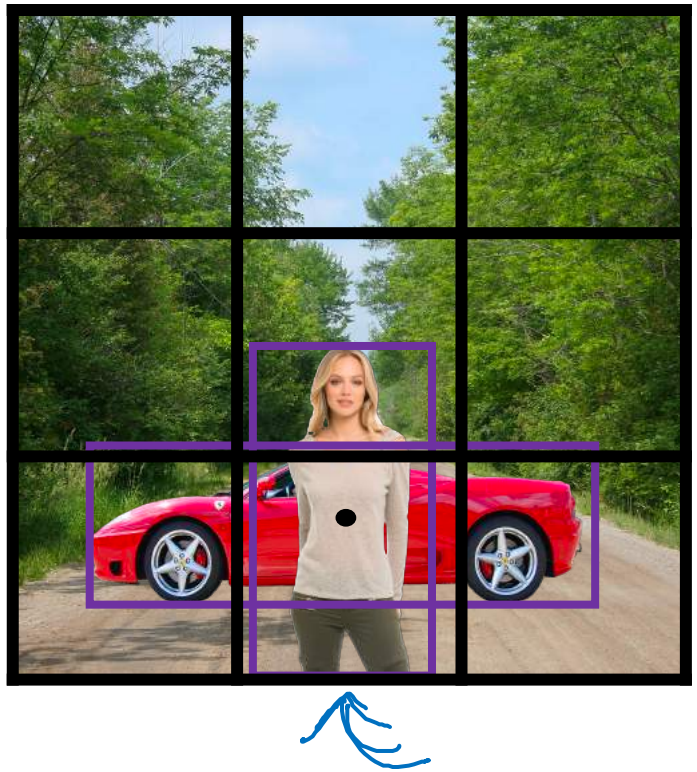
(grid cell, anchor box)

Output y:

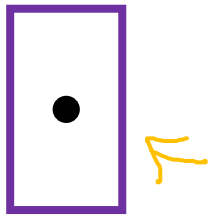
$3 \times 3 \times 16$

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

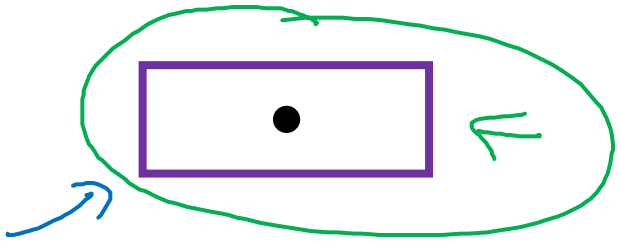
Anchor box example



Anchor box 1:



Anchor box 2:



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

deeplearning.ai

Object Detection

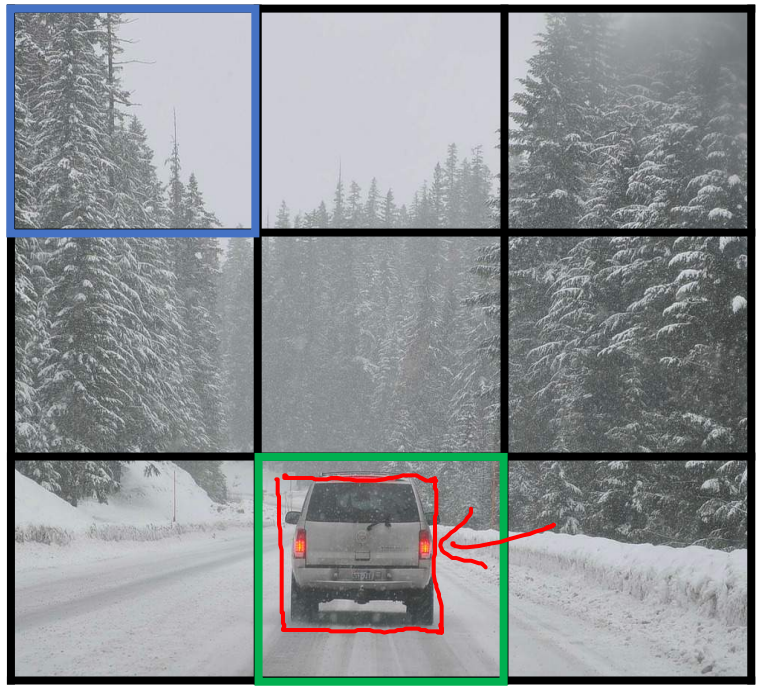
Putting it together: YOLO algorithm

Training

1 - pedestrian

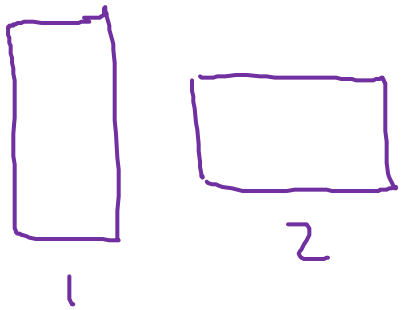
2 - car

3 - motorcycle



$y =$

$$\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$



3 × 3 × 16

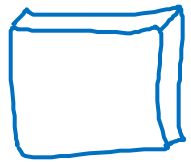
y is 3×3×2×8

19×19×16

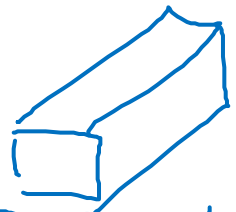
19×19×40

#anchors

5 + #classes

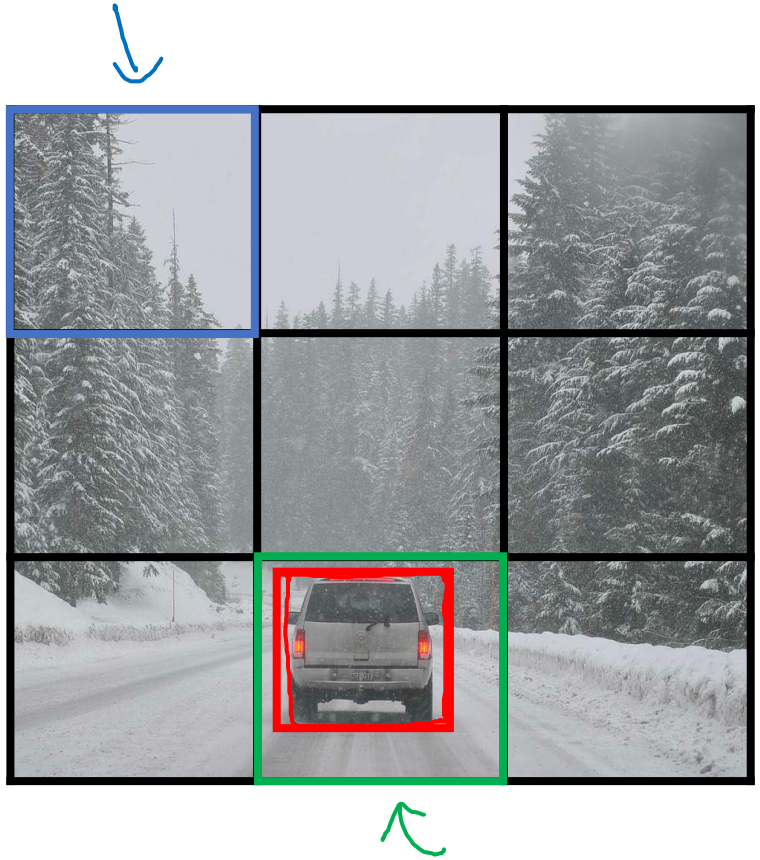


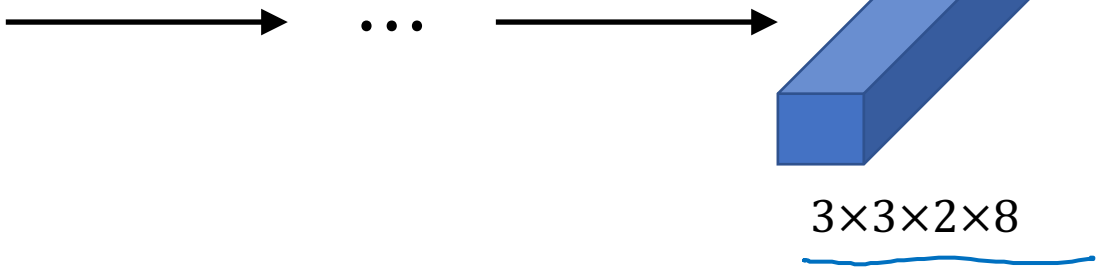
→ ConvNet →



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

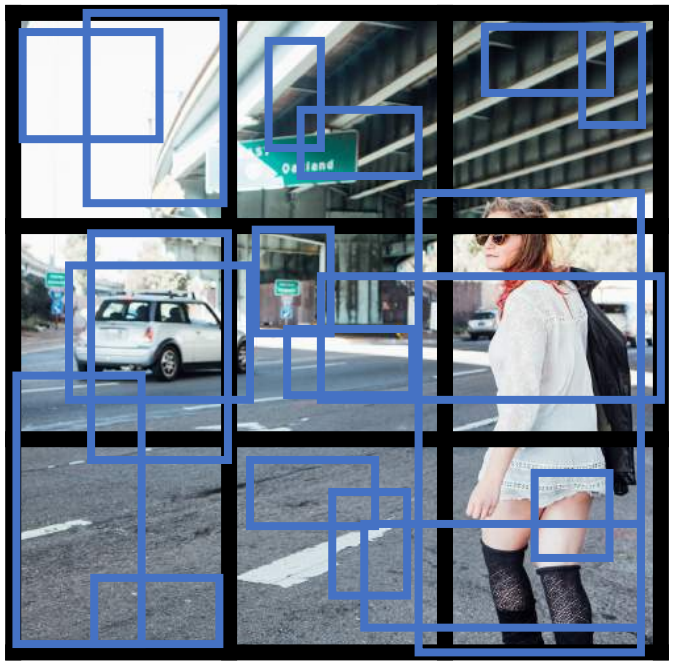
Making predictions





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

Outputting the non-max supressed 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.

In order to reduce the number of grids in which to check for objects, it was also proposed to use REGION PROPOSALS. The idea is to preliminary segment the image, then run the object recognition only in the regions where the segmentation found some shapes of interest.

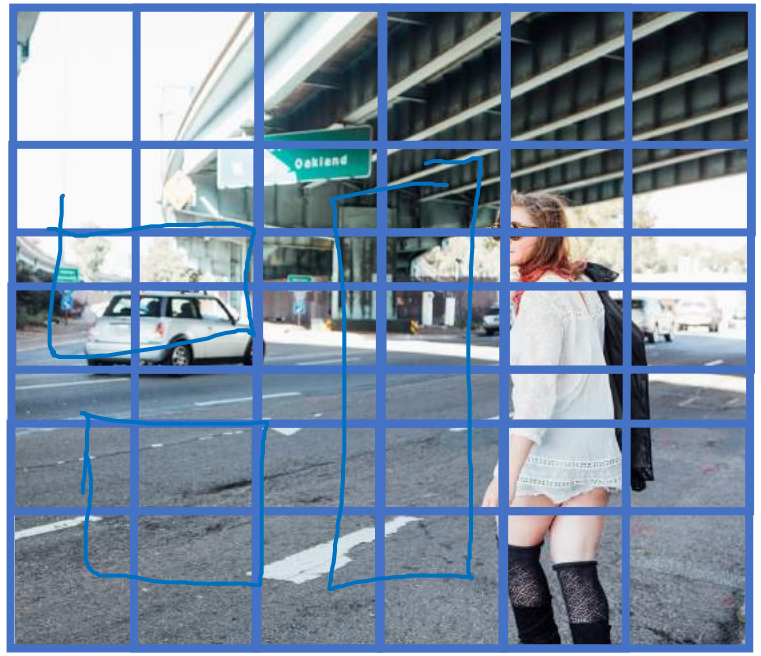
deeplearning.ai

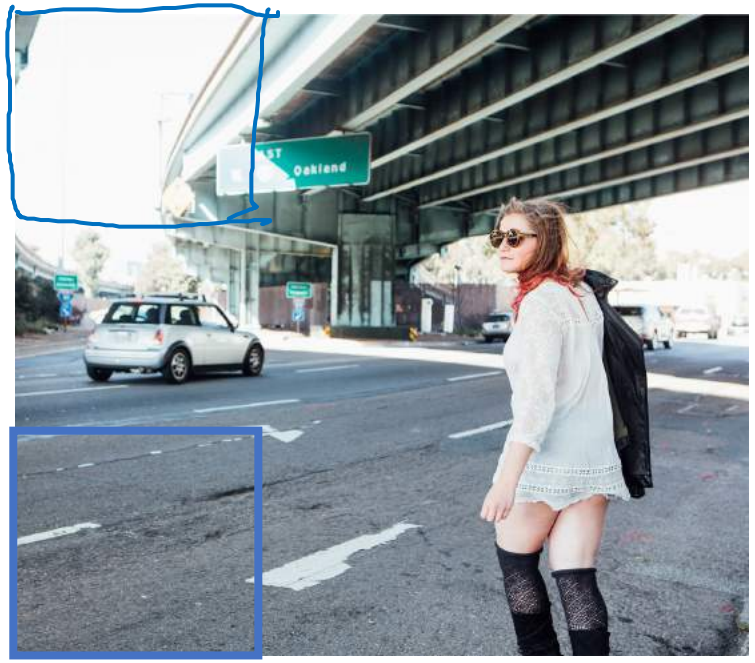
Object Detection

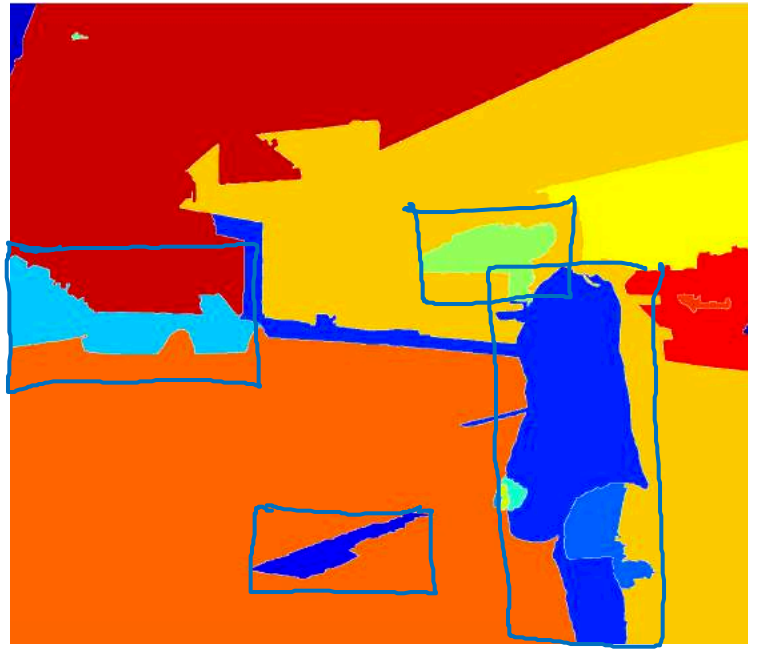
NG prefers "one step/one look" solutions such as YOLO

Region proposals (Optional)

Region proposal: R-CNN







Segmentation algorithm

~2,000

[Girshik et. al, 2013, Rich feature hierarchies for accurate object detection and semantic segmentation]

Faster algorithms

R-CNN: Propose regions. Classify proposed regions one at a time. Output label + bounding box.

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

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

↳ actually STILL SLOWER THAN YOLO

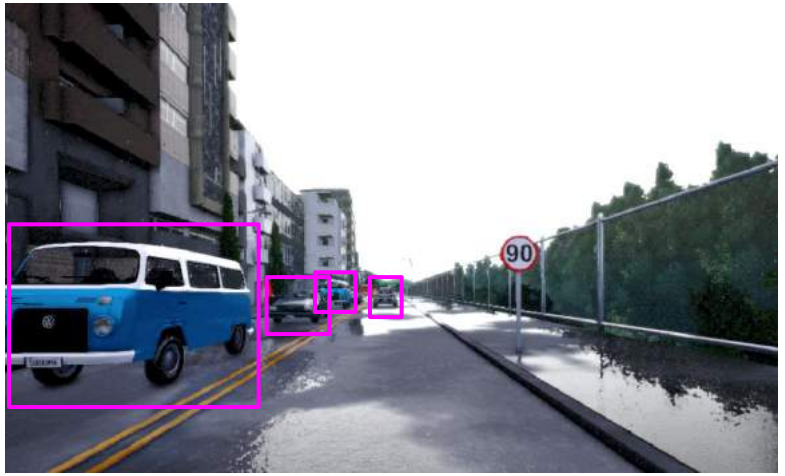
[Girshik et. al, 2013. Rich feature hierarchies for accurate object detection and semantic segmentation]
[Girshik, 2015. Fast R-CNN]
[Ren et. al, 2016. Faster R-CNN: Towards real-time object detection with region proposal networks]

Convolutional Neural Networks

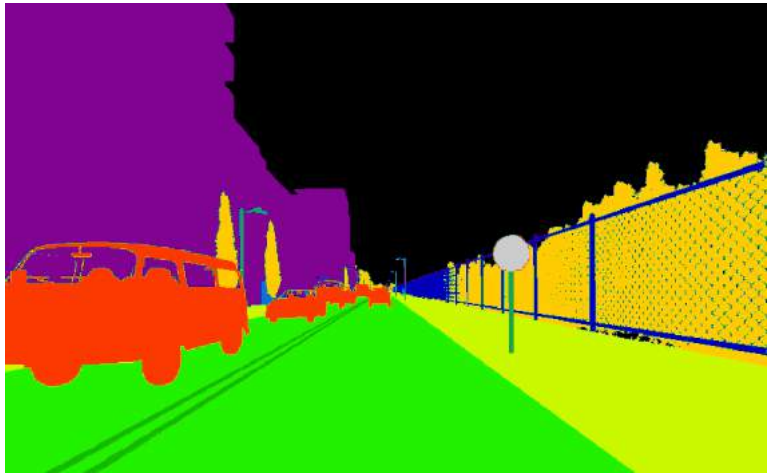
Semantic segmentation with U-Net

Object Detection vs. Semantic Segmentation

Input image

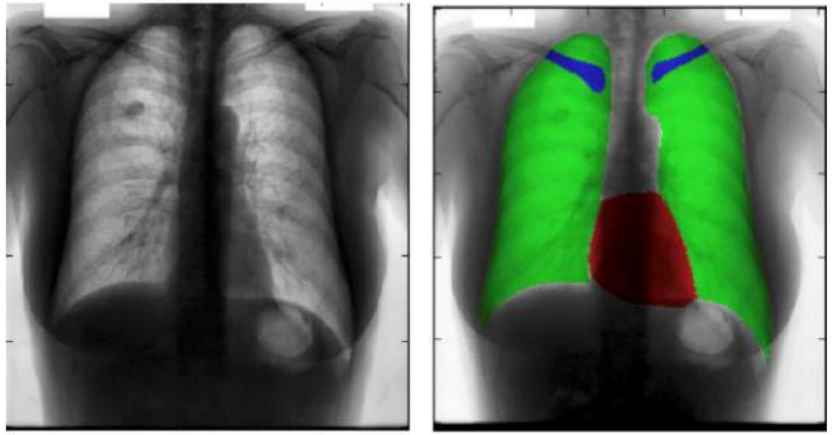


Object Detection

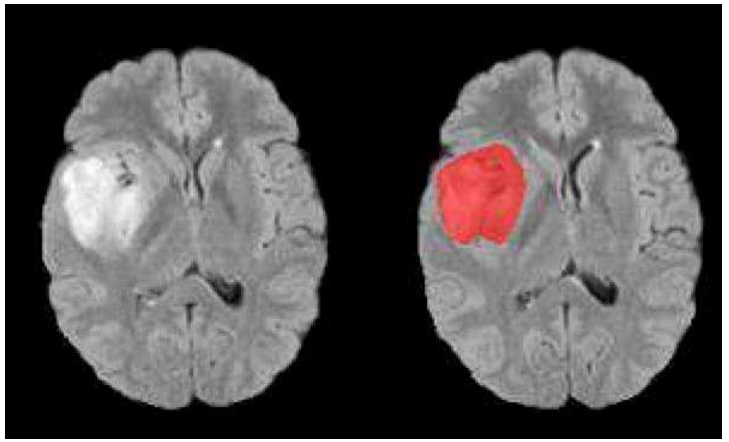


Semantic Segmentation

Motivation for U-Net



Chest X-Ray



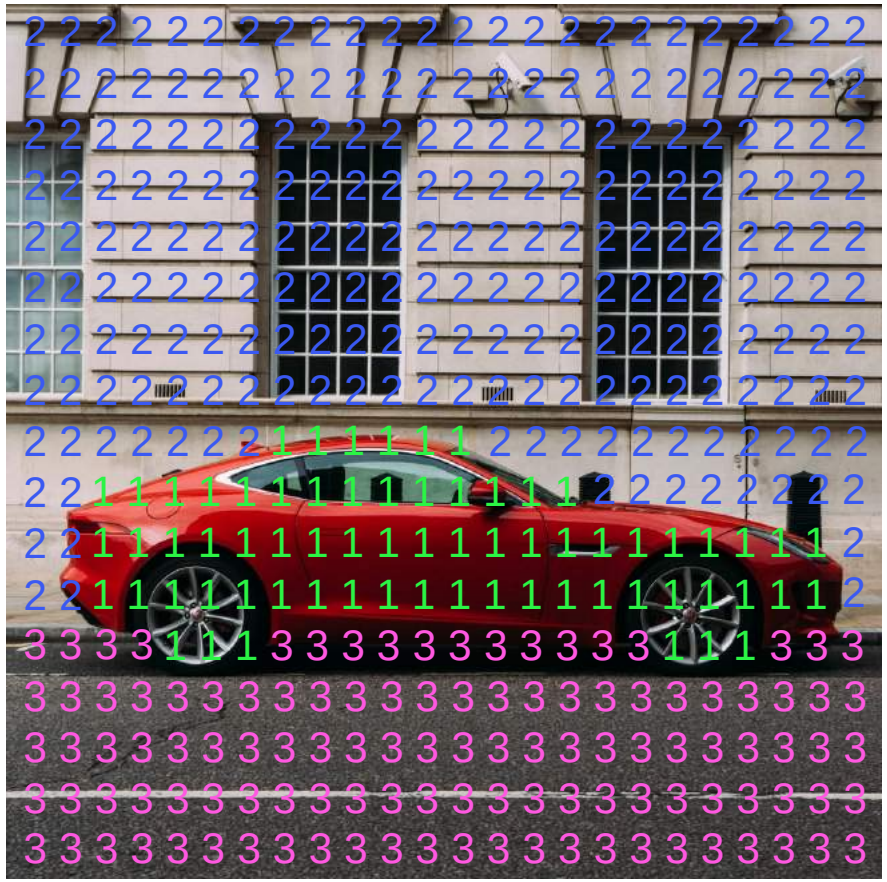
Brain MRI

[Novikov et al., 2017, Fully Convolutional Architectures for Multi-Class Segmentation in Chest Radiographs]
[Dong et al., 2017, Automatic Brain Tumor Detection and Segmentation Using U-Net Based Fully Convolutional Networks]

Per-pixel class labels

1. Car
0. Not Car

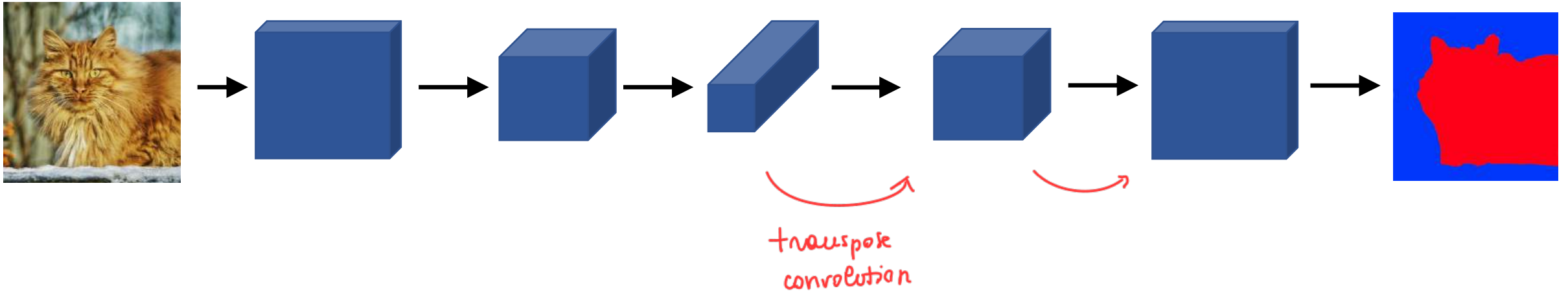
Per-pixel class labels



1. Car
2. Building
3. Road

Segmentation Map

Deep Learning for Semantic Segmentation

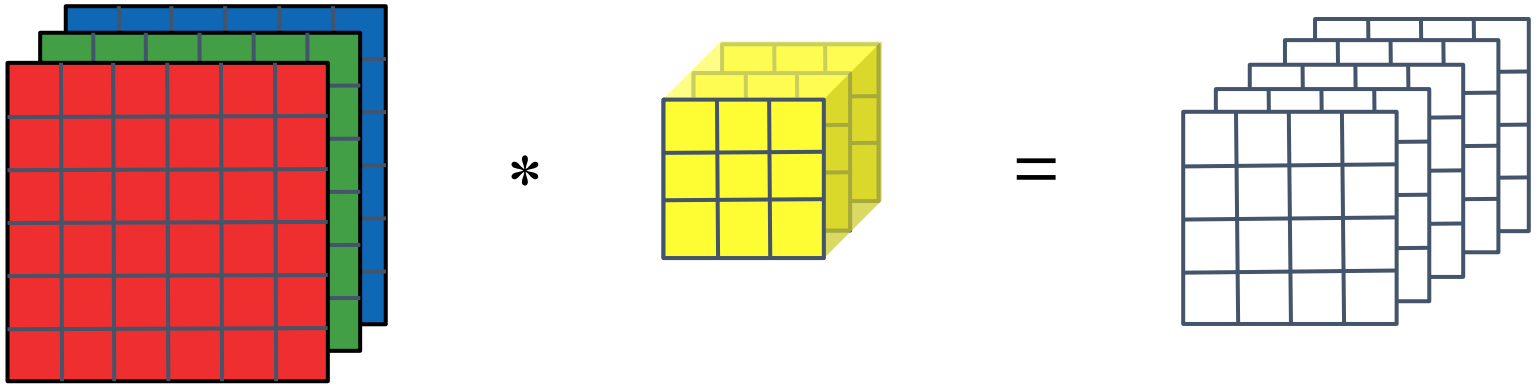


intuition: compress image to extract high-level semantic information, then reexpand that information attempting to combine it with low-level spatial information.

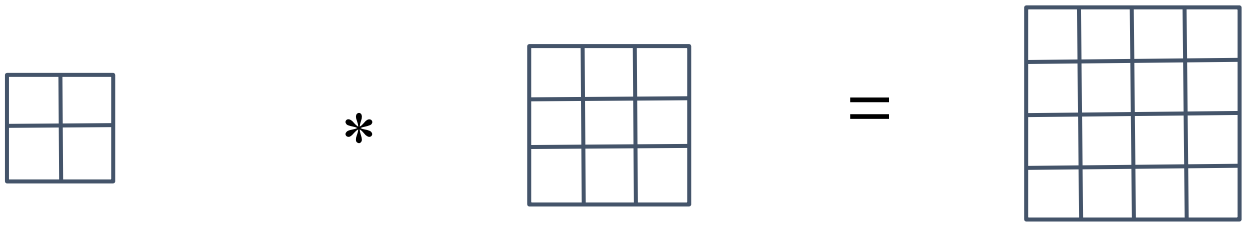
Andrew Ng

Transpose Convolution

Normal Convolution



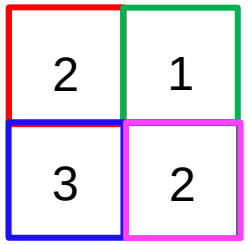
Transpose Convolution



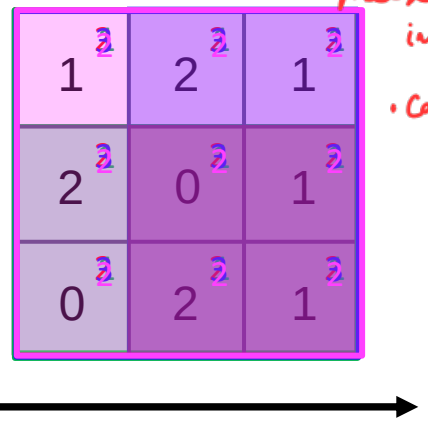
idea: place the filter on the output. • Define stride and padding, which will be applied on the output
• Take the first element in the input, multiply it by the filter, overlap the filter with the output (account for padding!), copy filter values.

Transpose Convolution

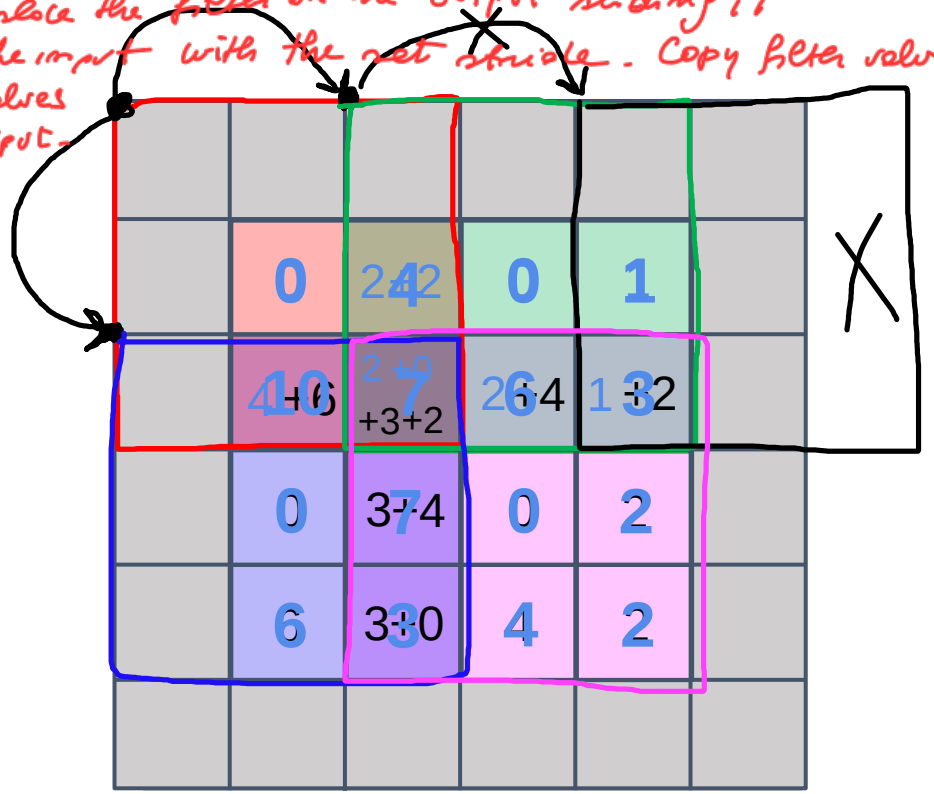
• Repeat for second element of the input, place the filter on the output sliding it in the same direction as in the input with the next stride. Copy filter values but add values to pre-existing values in the output.
• Continue.



2 x 2



weight filter

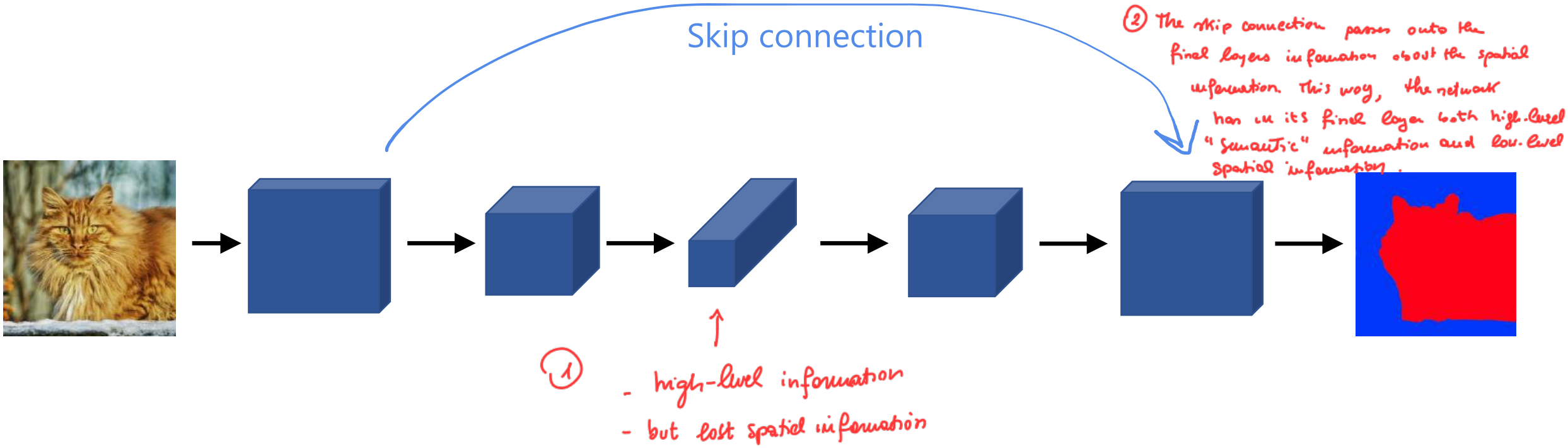


4 x 4

filter f x f = 3 x 3 padding p = 1 stride s = 2

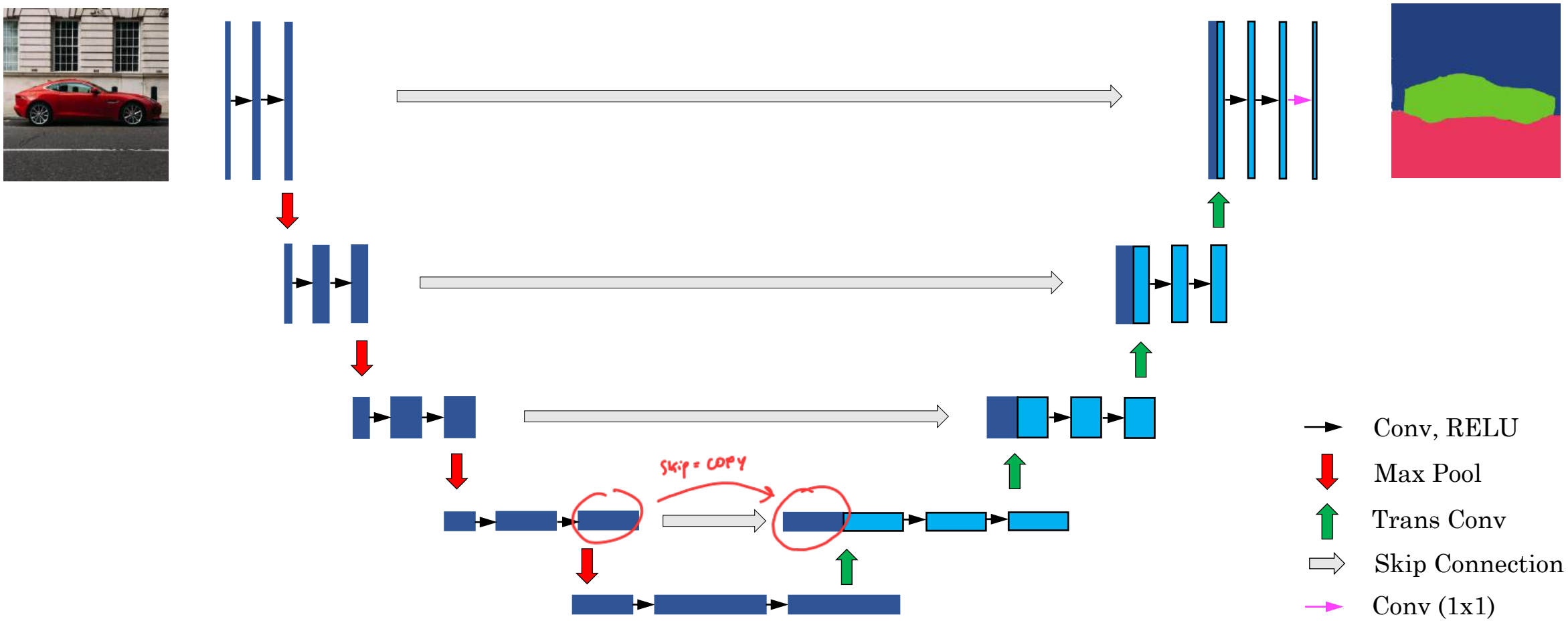
Andrew Ng

Deep Learning for Semantic Segmentation



Andrew Ng

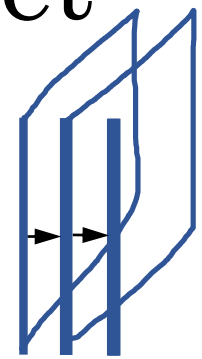
U-Net

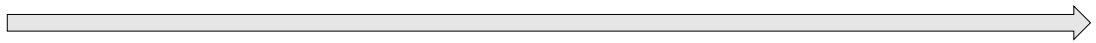


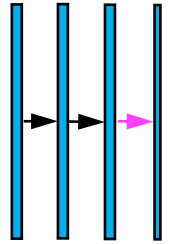
[Ronneberger et al., 2015, U-Net: Convolutional Networks for Biomedical Image Segmentation]

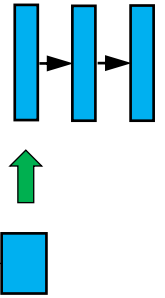
U-Net

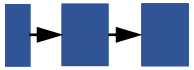
h x w x 3

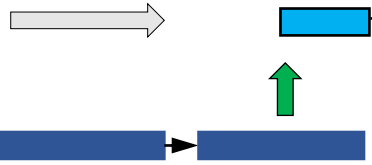


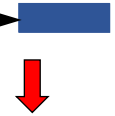


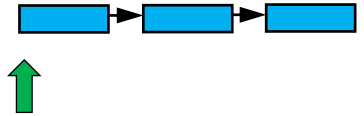




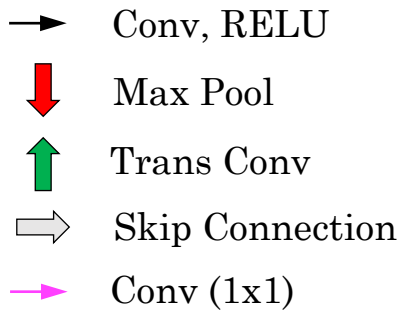








h x w x # classes



[Ronneberger et al., 2015, U-Net: Convolutional Networks for Biomedical Image Segmentation]

Andrew Ng