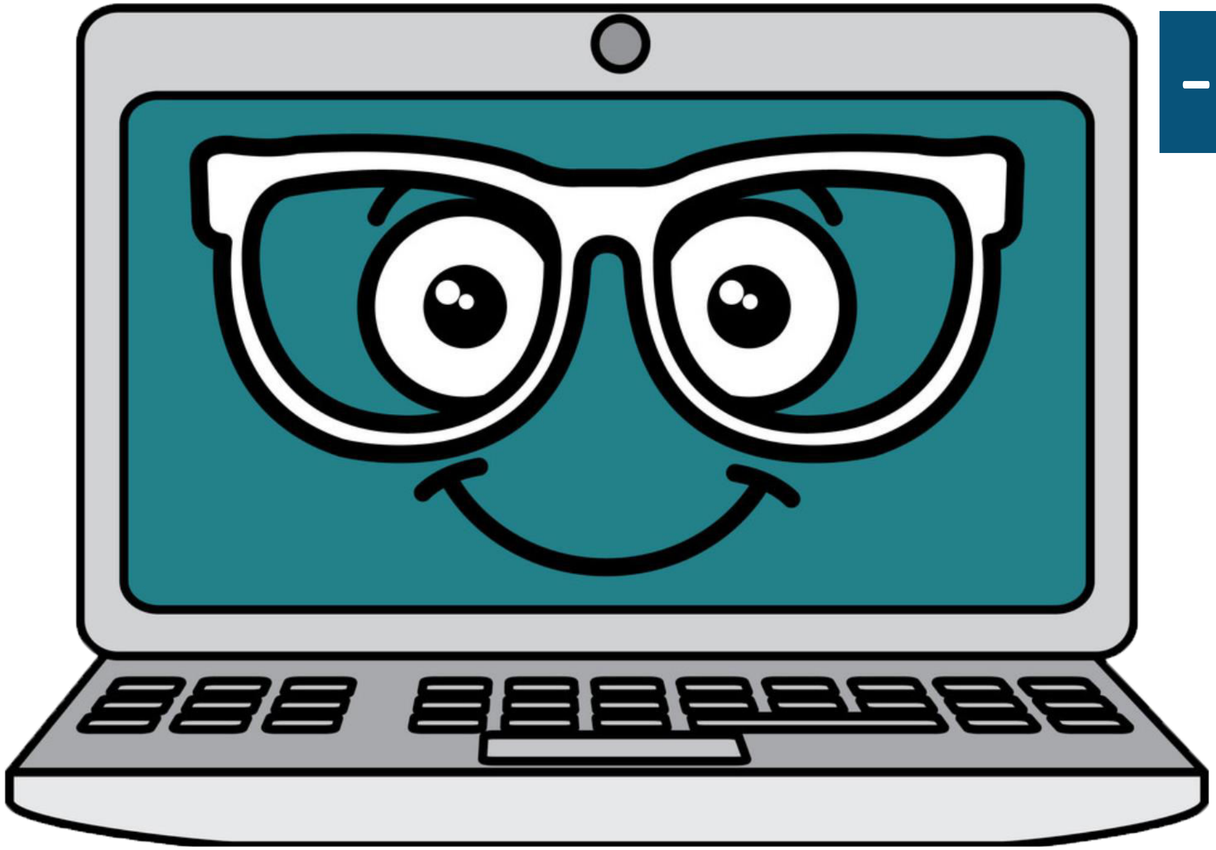
PyCon Pakistan 2017

Giving Computers the power of Sight

- Object Detection



Fahad Sarfraz

INTRODUCTION

Fahad Sarfraz

Software Engineer at Arbisoft - Machine

Machine Learning practitioner and an avid learner with professional experience in managing ML projects and providing ML solutions for image classification, text analytics and stock market trend prediction. I have expertise in numerical optimization, algorithm design and optimization for high precision and computational speed. I aim to lie somewhere in the middle of the spectrum between applied and research ML.

Email id: fahad.sarfraz@outlook.com | **Tel:** +92-341-4400551

LinkedIn: https://www.linkedin.com/in/fahadsarfraz/ | **GitHub:** https://github.com/fahad92virgo/

Agenda

1. Setting up the stage for object detection
2. Sliding Window Approach
3. YOLO Algorithm Components
4. Using pre-trained network to detect objects
5. Further discussion and Questions and Answers

Goal

While it took 25s for the car to make this journey, Getting those fancy labels with a bunch of numbers took researchers decades ☺

Image Classification and Localization

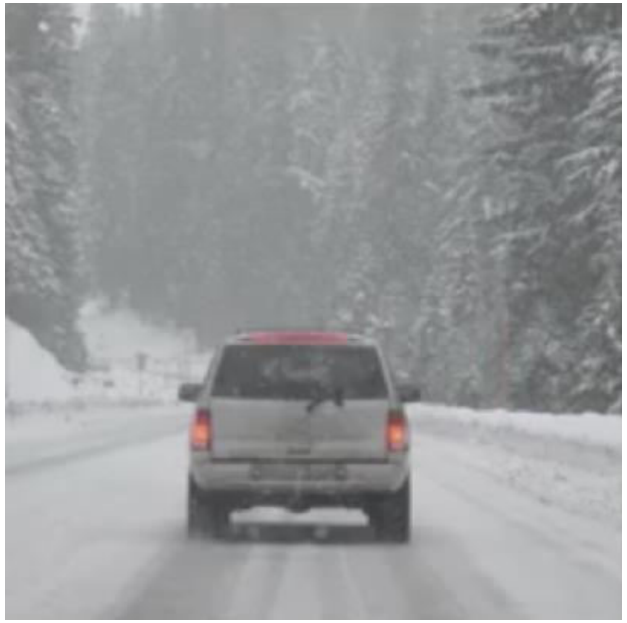


Image Classification

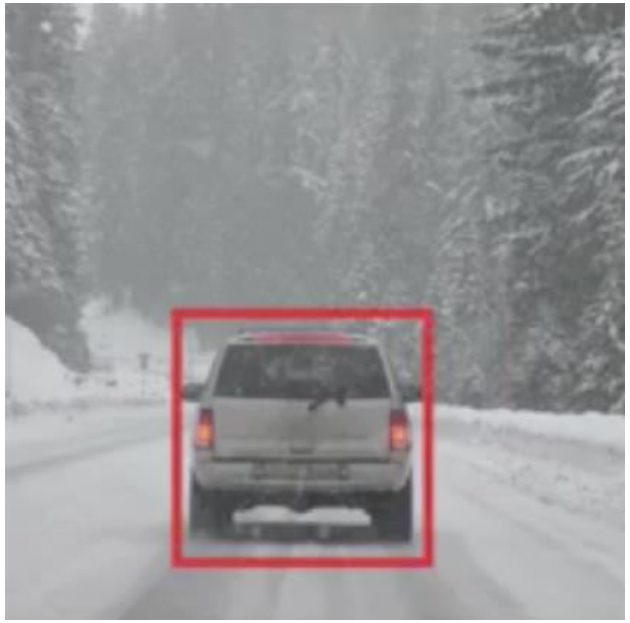
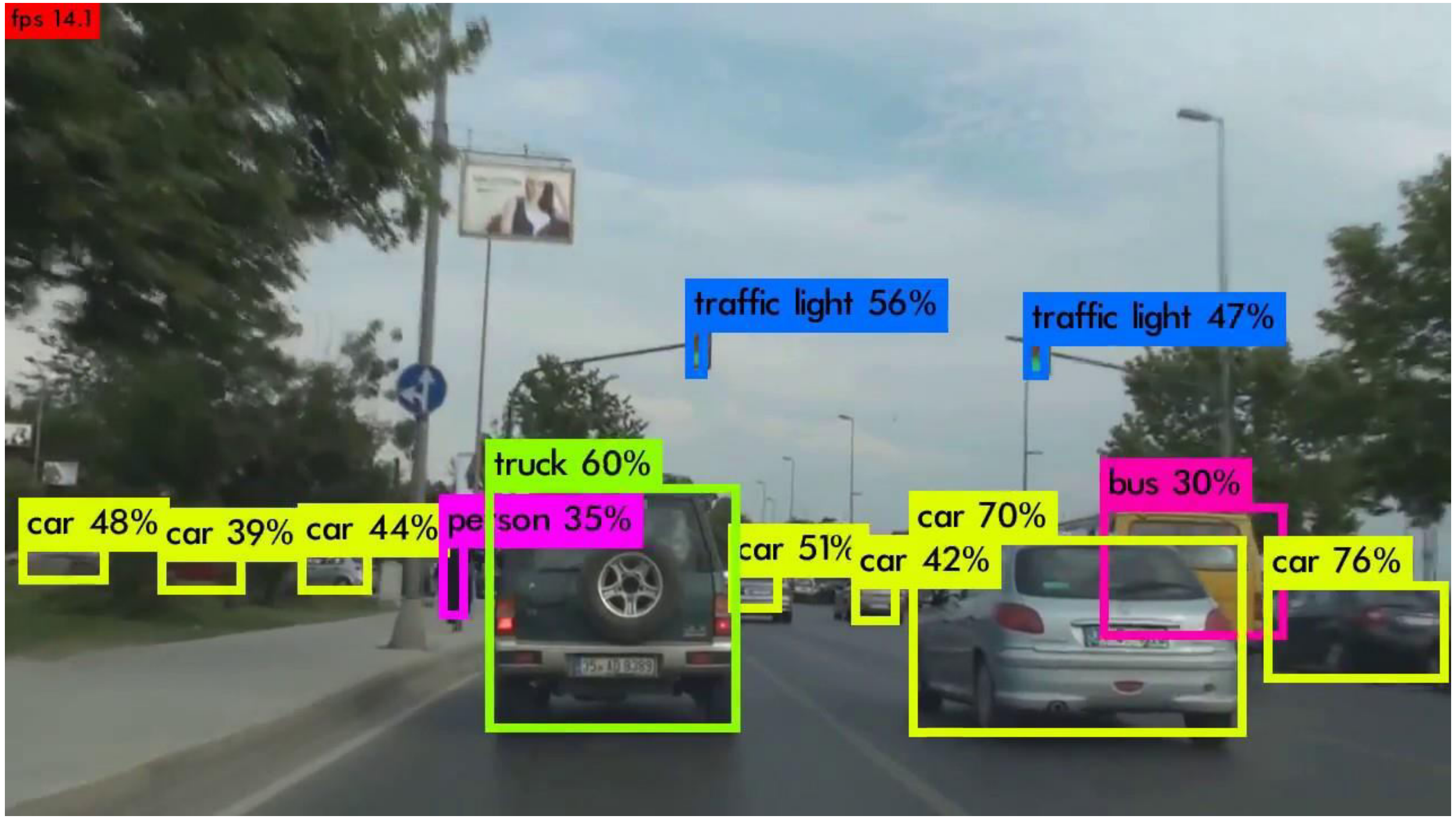
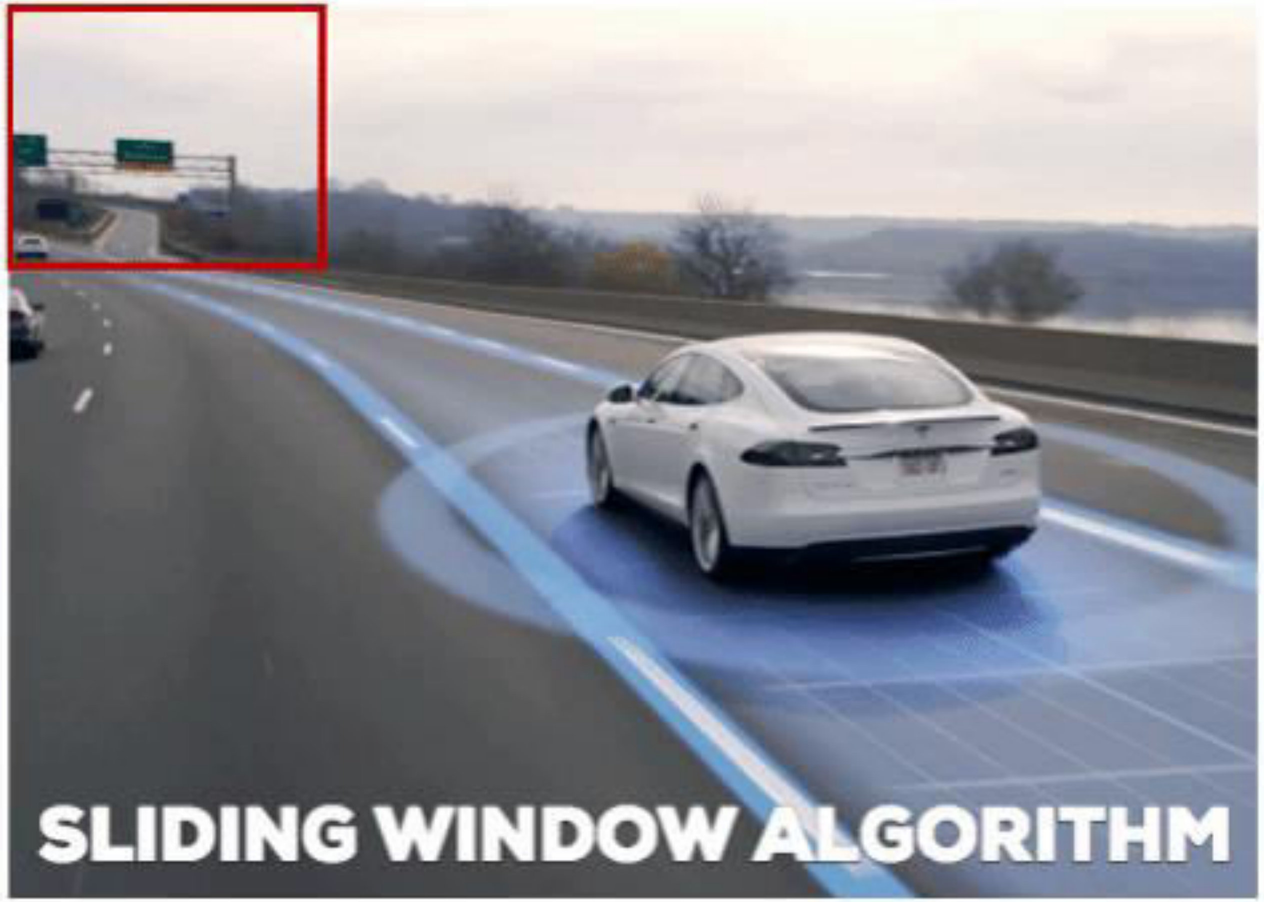


Image Classification with Localization

Object Detection



Sliding Window Approach



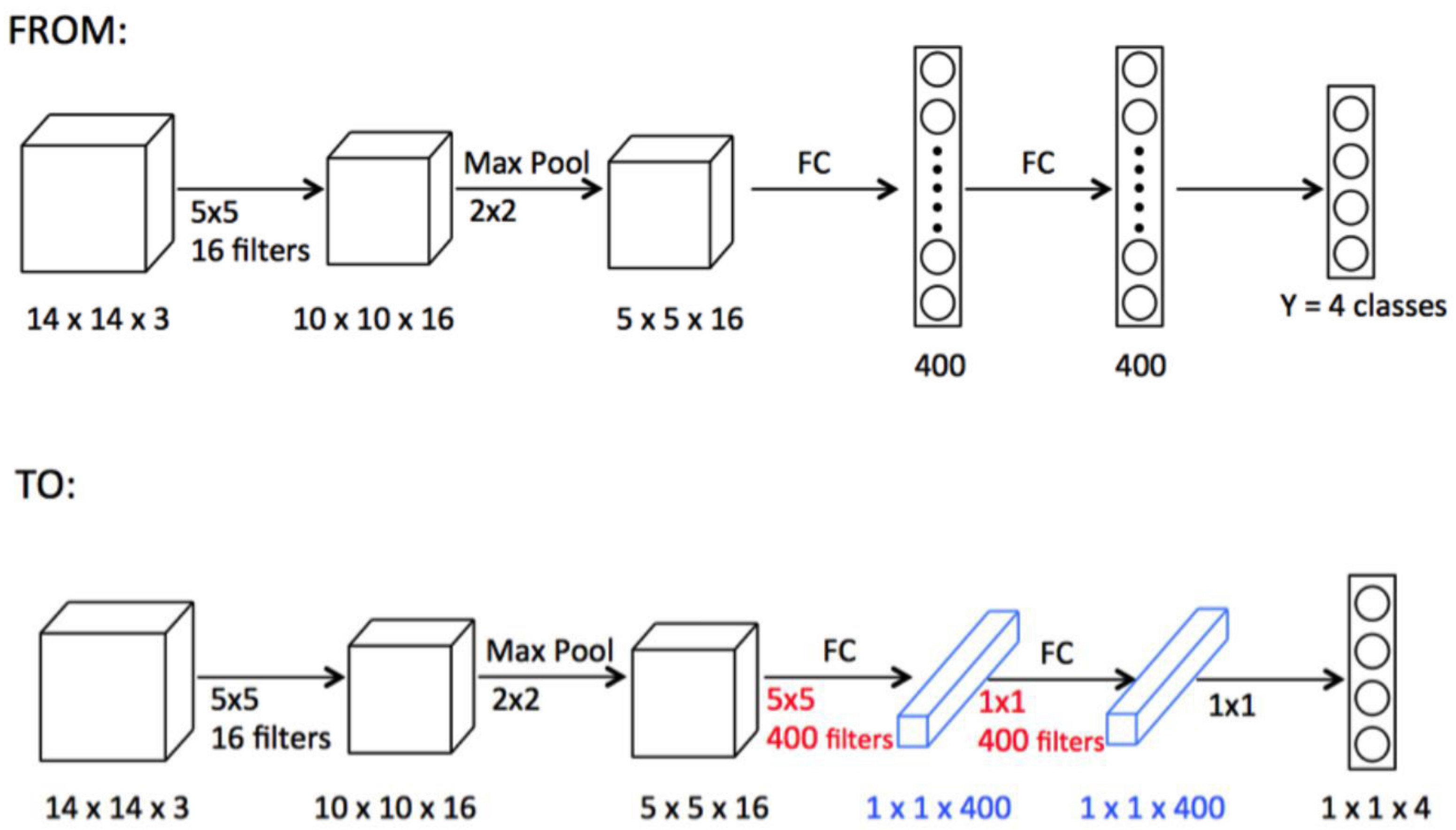
- Train a ConvNet to detect objects within an image and use windows of different sizes to slide on top of it.
- For each window, make a prediction.

Issue: It is computationally very expensive, since we can have a lot of windows

Solution: Convolutional Implementation of Sliding window

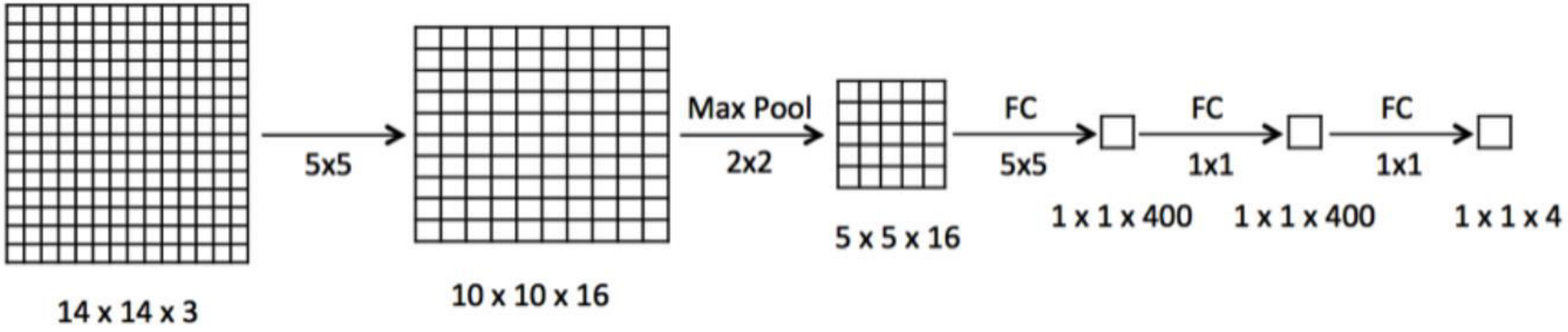
Sliding Window: Convolution Implementation

The first step to build up towards the convolutional implementation of sliding windows is to turn the Fully Connected layers in a neural network into convolutional layers.

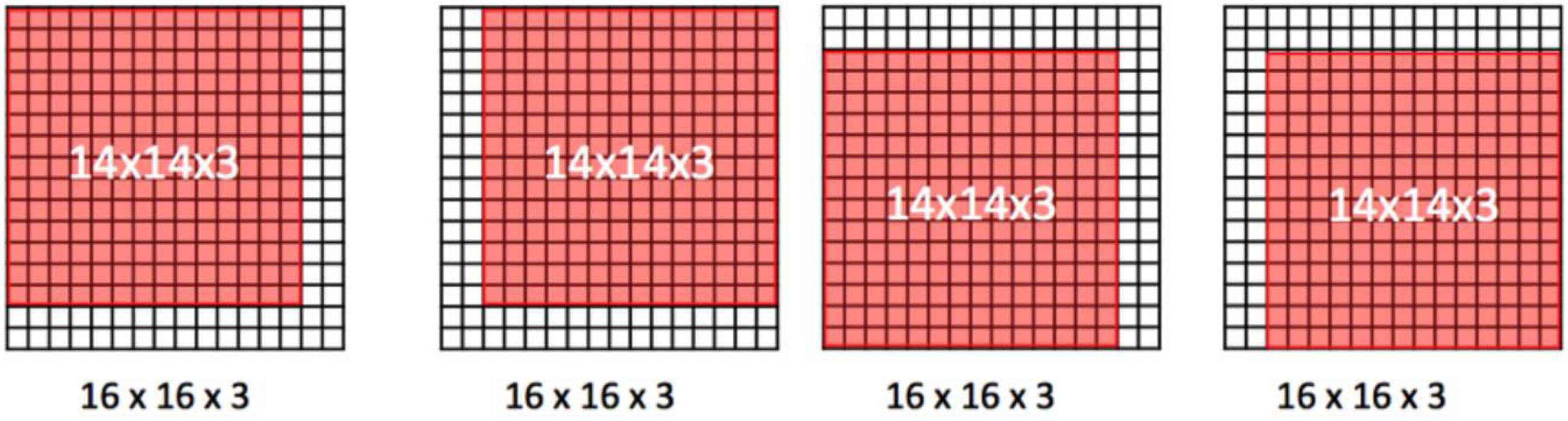


Sliding Window: Convolution Implementation cont..

2D Representation:

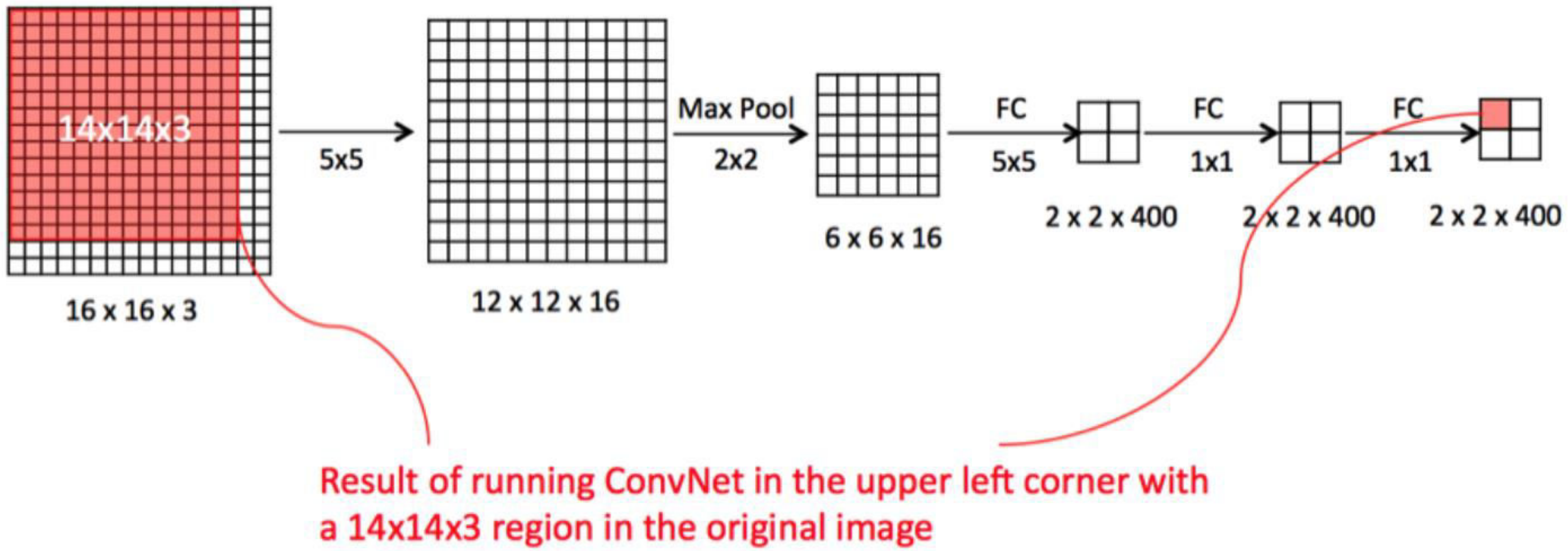


14x14x3 image crops of a 16x16x3 Image



Sliding Window: Convolution Implementation cont..

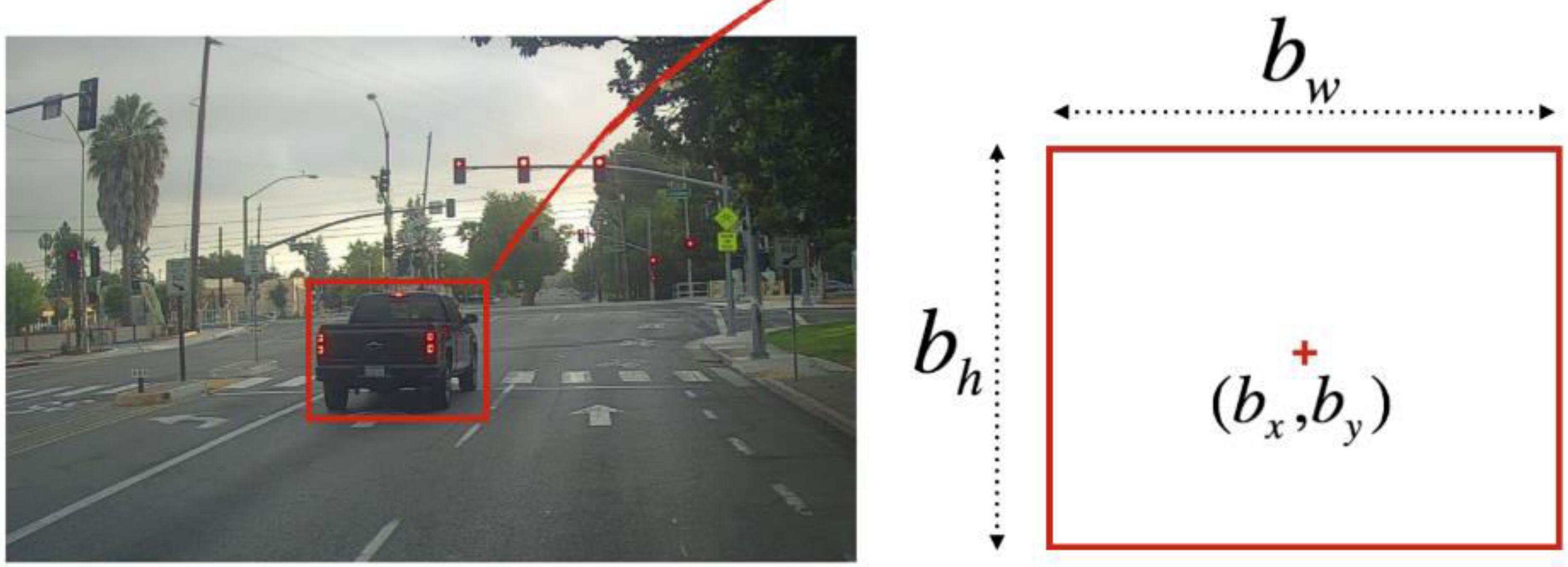
with the convolutional implementation of sliding windows we run the ConvNet, with the same parameters and same filters on the test image and this is what we get:



Each of the 4 subsets of the output unit is essentially the result of running the ConvNet with a 14x14x3 region in the four positions on the initial 16x16x3 image

YOLO Algorithm: Definition of a box

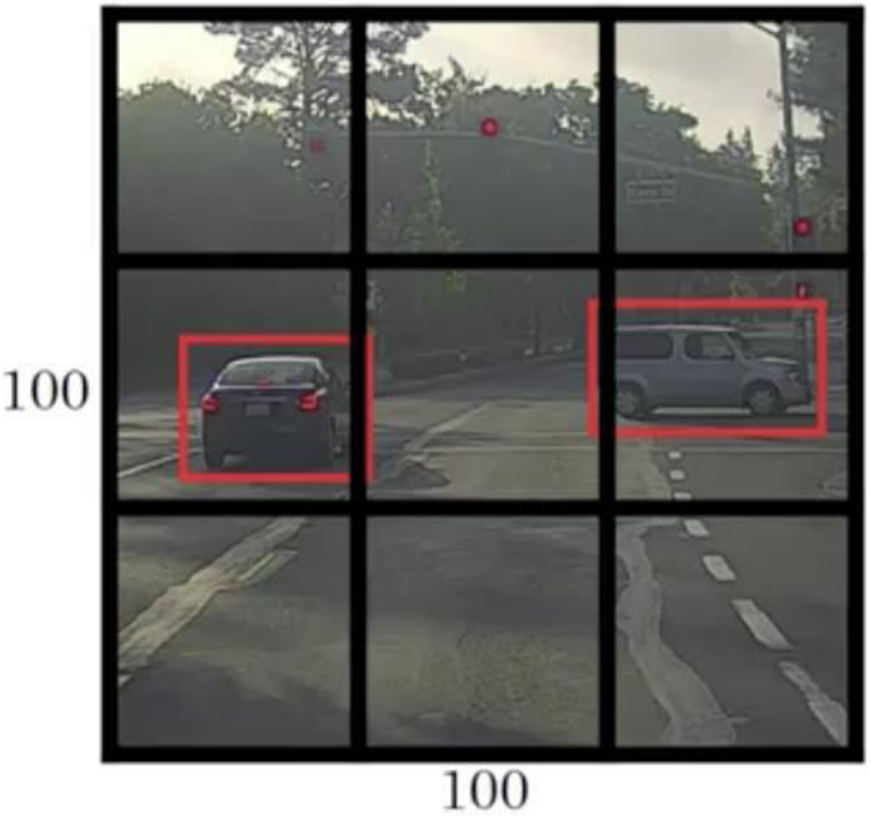
$$y = (p_c, b_x, b_y, b_h, b_w, c)$$



$p_c = 1$: confidence of an object being present in the bounding box

$c = 3$: class of the object being detected (here 3 for "car")

YOLO Algorithm: Division into grids



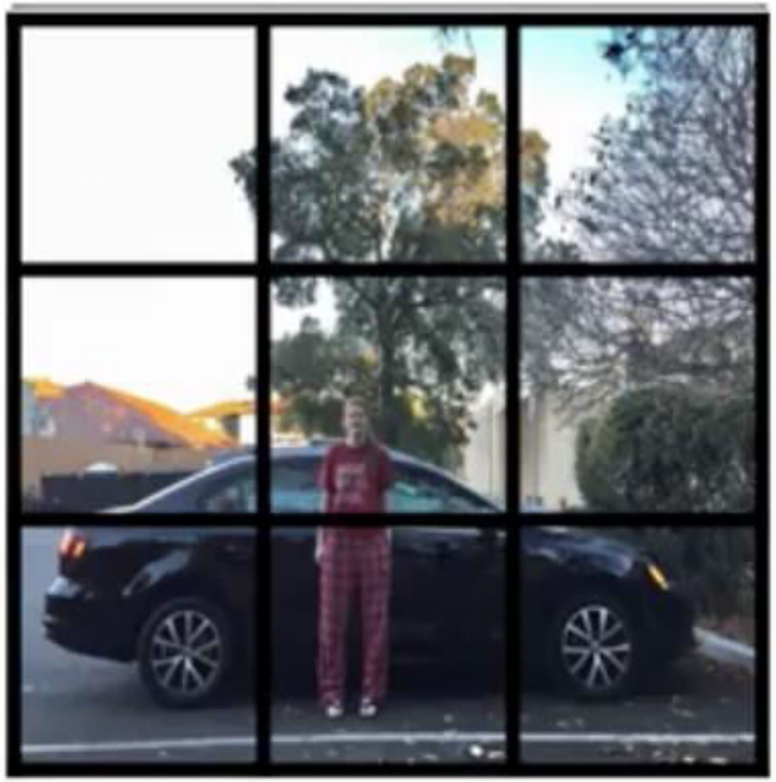
Labels for training
For each grid cell:

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

Object of class 2 (car) detected

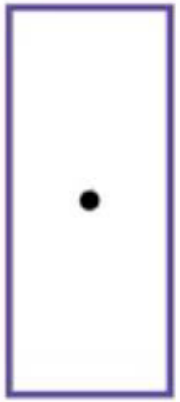
No object detected

YOLO Algorithm: Anchor boxes



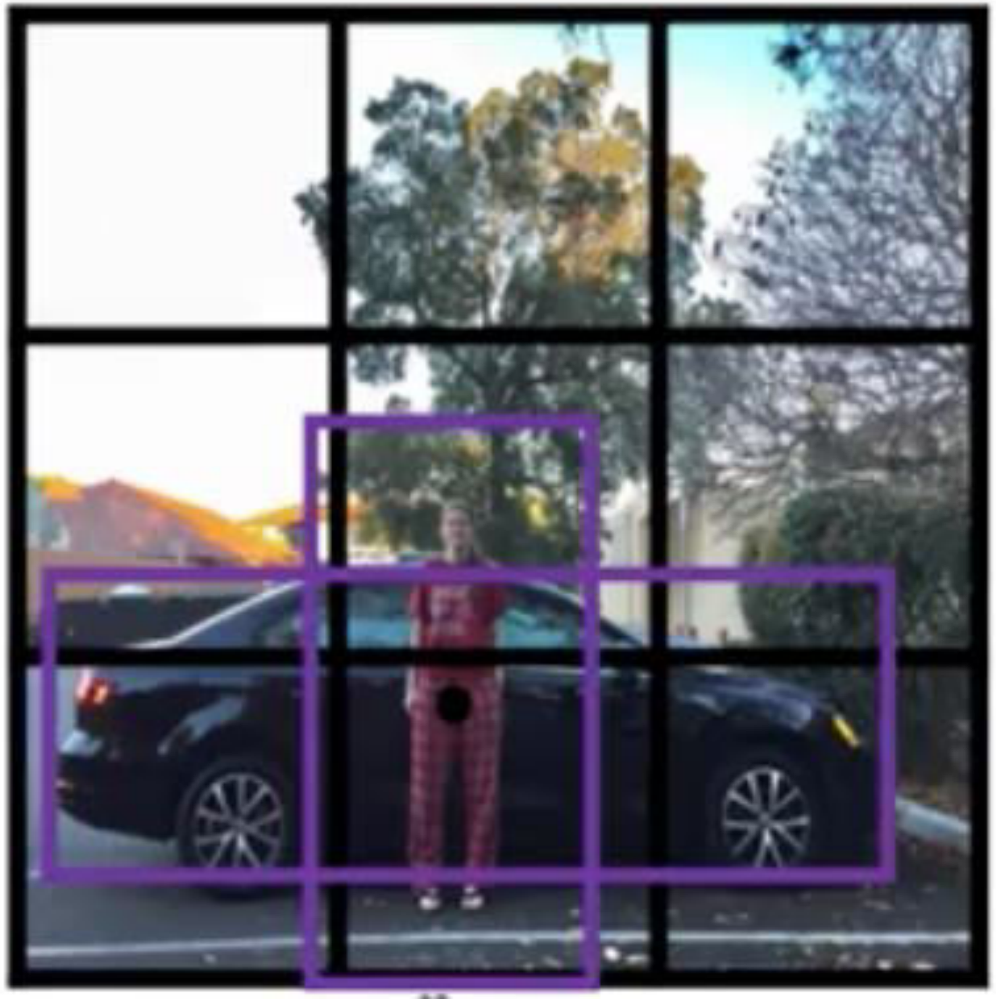
Overlapping Objects

Anchor box 1:

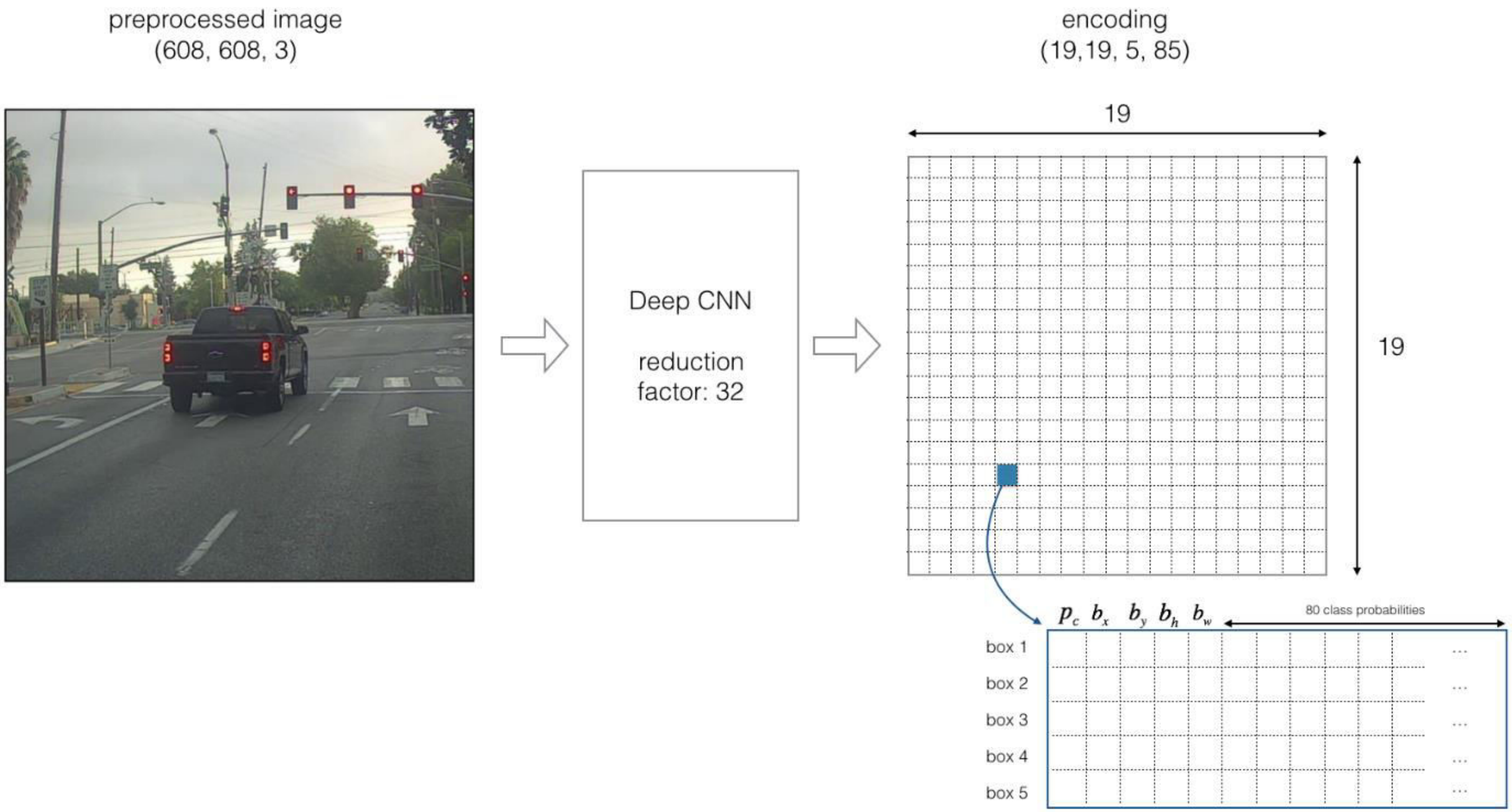


Anchor box 2:

YOLO Algorithm: Anchor boxes cont

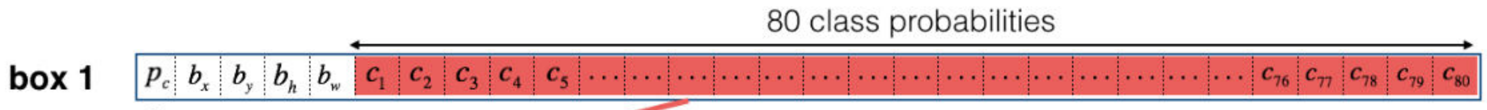


YOLO Algorithm: Encoding Architecture



YOLO Algorithm: Computing Class probability in a box

For each box of each grid cell, compute the following elementwise product and extract a probability that the box contains a certain class



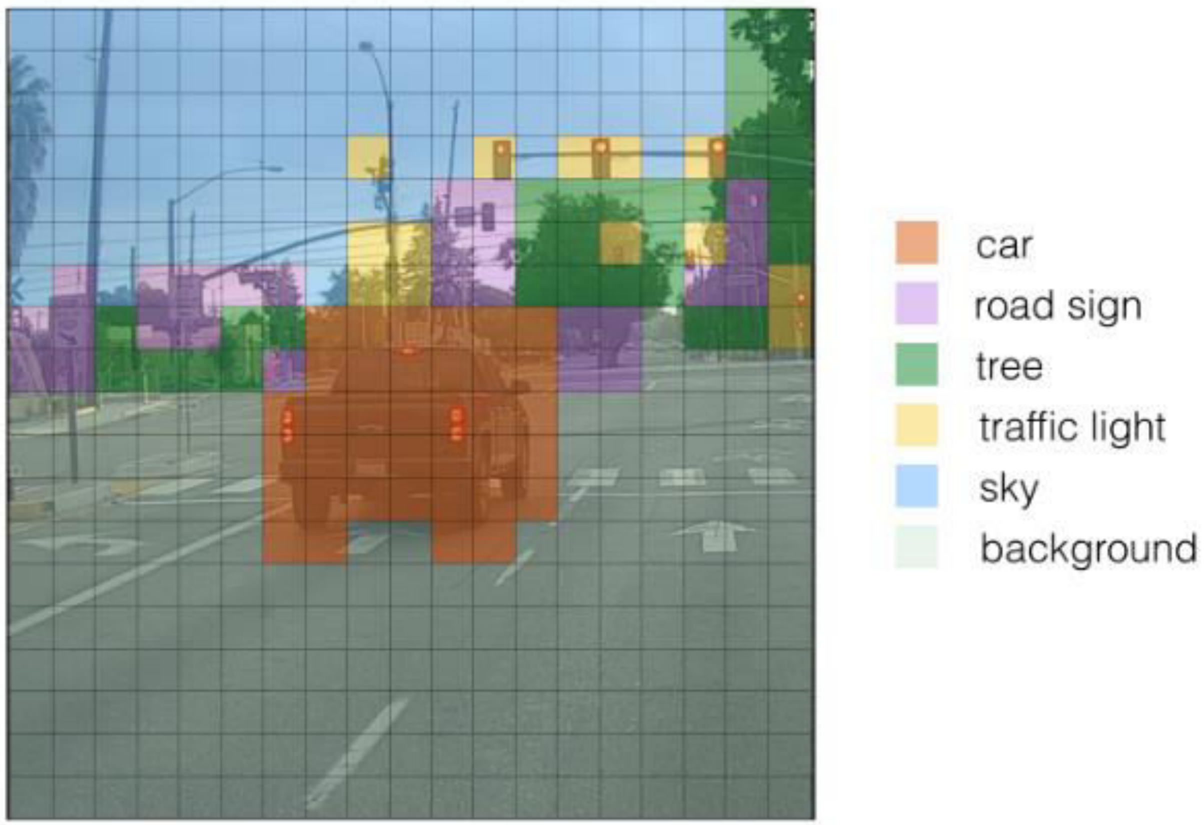
$$scores = p_c * \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{78} \\ c_{79} \\ c_{80} \end{pmatrix} = \begin{pmatrix} p_c c_1 \\ p_c c_2 \\ p_c c_3 \\ \vdots \\ p_c c_{78} \\ p_c c_{79} \\ p_c c_{80} \end{pmatrix} = \begin{pmatrix} 0.12 \\ 0.13 \\ 0.44 \\ \vdots \\ 0.07 \\ 0.01 \\ 0.09 \end{pmatrix} \xrightarrow{\text{find the max}}$$

score: 0.44
box: (b_x, b_y, b_h, b_w)
class: c = 3 (“car”)

the box (b_x, b_y, b_h, b_w) has detected c = 3 (“car”) with probability score: 0.44

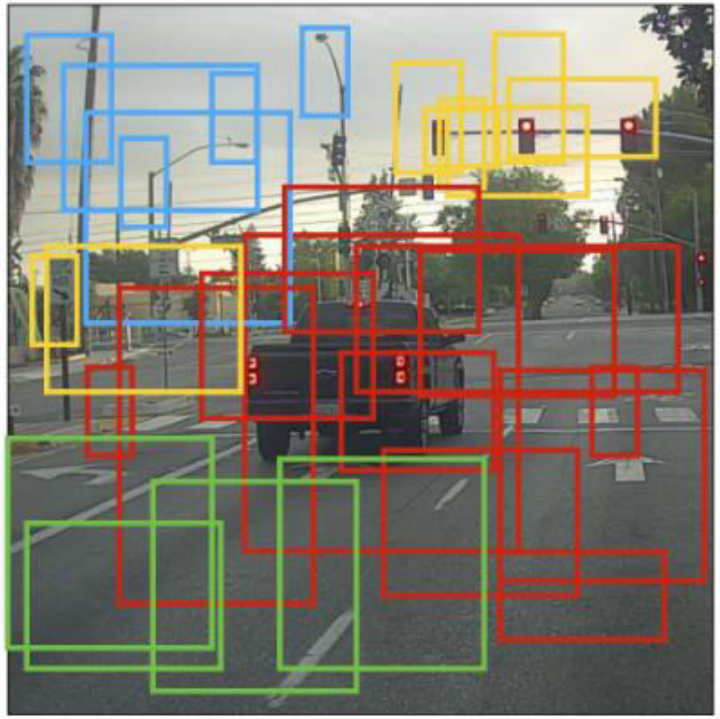
YOLO Algorithm: Computing Class probability in a Grid cell

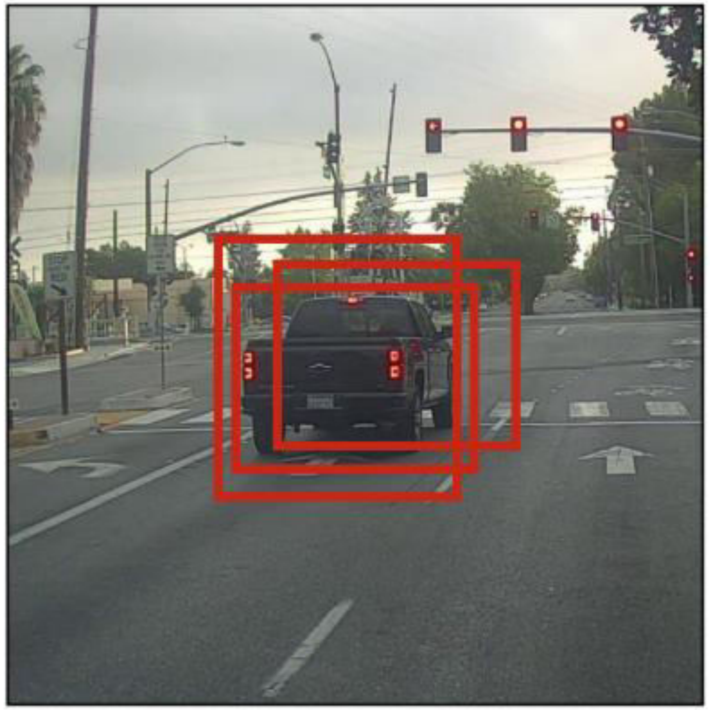
For each of the grid cells, find the maximum of the probability scores (taking a max across both the anchor boxes and across different classes)



YOLO Algorithm: Filtering with a threshold on class scores

Get rid of any box for which the class "score" is less than a chosen threshold

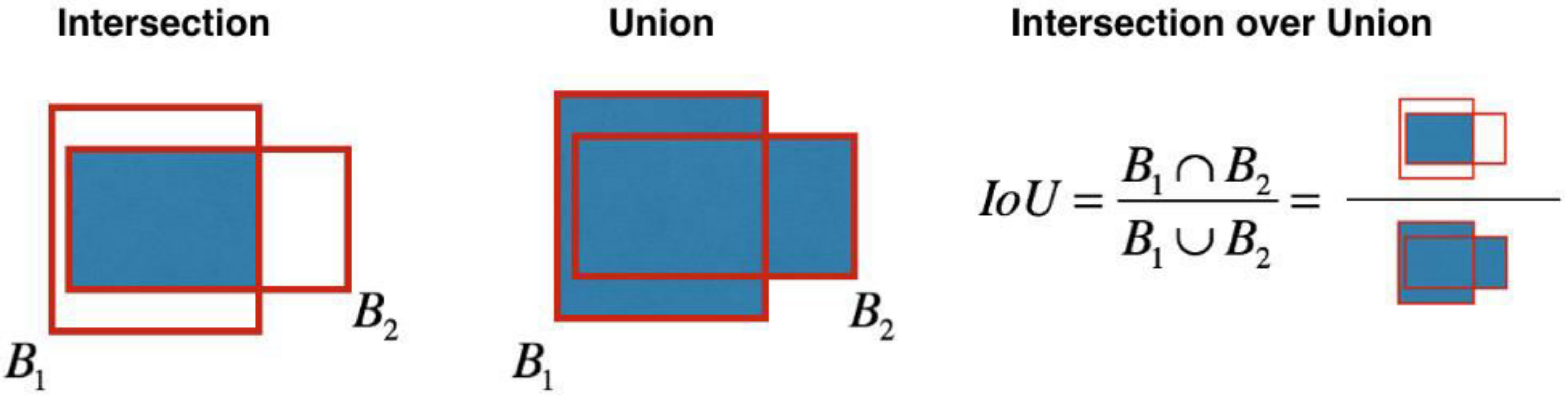




YOLO Algorithm: Non-max suppression - IoU

Even after filtering by thresholding over the classes scores, you still end up a lot of overlapping boxes.
A second filter for selecting the right boxes is called non-maximum suppression (NMS).

Non-max suppression uses a function called **"Intersection over Union"**, or IoU

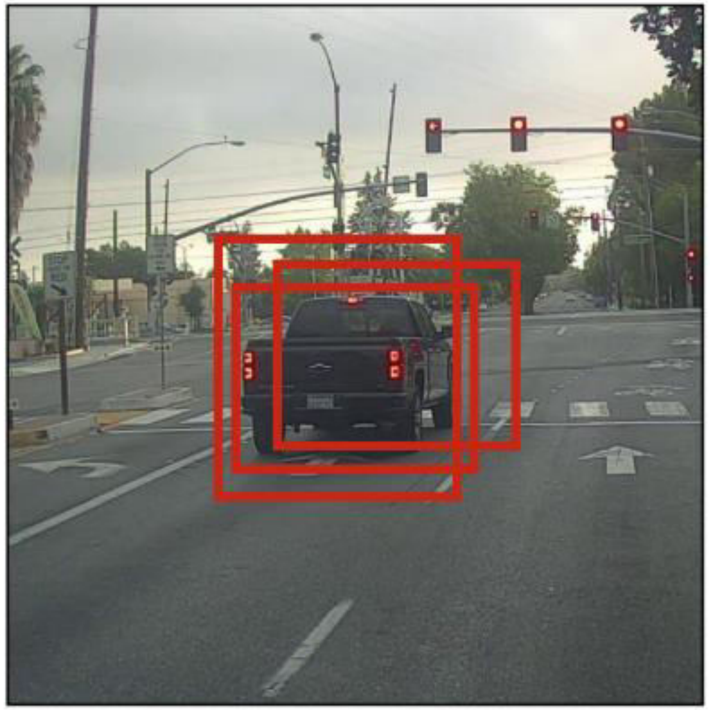


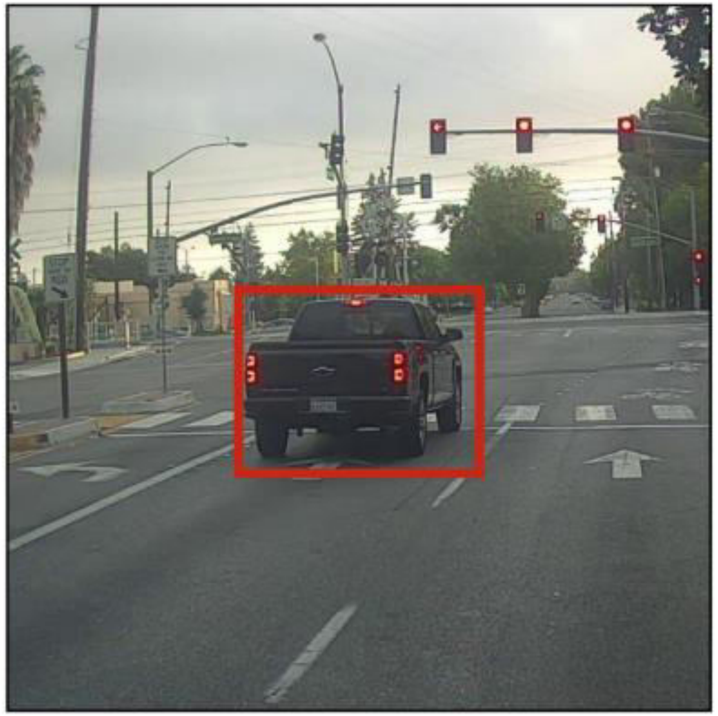
YOLO Algorithm: Non-max suppression algorithm

For each class:

- Select the box that has the highest score.
- Compute its overlap with all other boxes, and remove boxes that overlap it more than iou_threshold.
- Go back to step 1 and iterate until there's no more boxes with a lower score than the current selected box.

ave a large overla ɔest" boxes



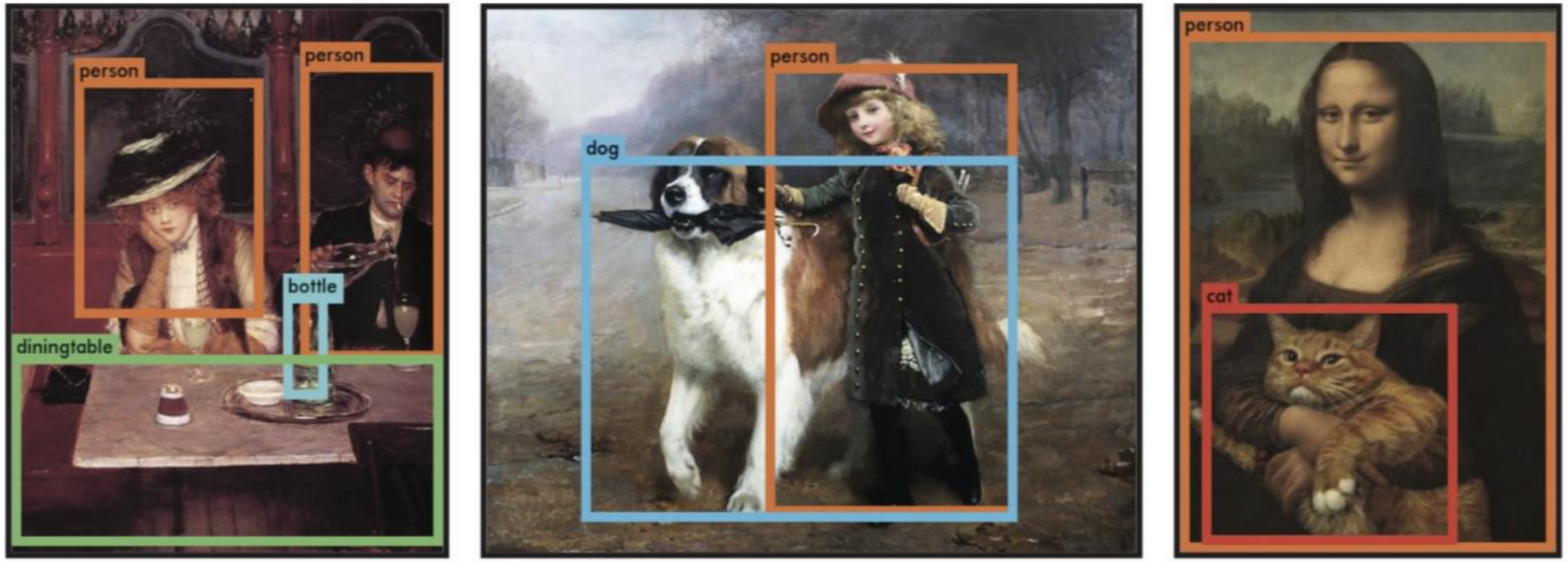


YOLO Algorithm: Training Phase

- Look which cell is near the center of the bounding box of the Ground truth. (Matching phase)
- Check from a particular cell which of it's bounding boxes overlaps more with the ground truth (IoU), then decrease the confidence of the bounding box that overlap less.
- Decrease the confidence of all bounding boxes from each cell that has no object. Also don't adjust the box coordinates or class probabilities from those cells.

YOLO Algorithm: Benefits

- Fast. Good for real-time processing.
- Predictions (object locations and classes) are made from one single network. Can be trained end-to-end to improve accuracy.
- YOLO is more generalized. It outperforms other methods when generalizing from natural images to other domains like artwork.



THANK YOU