**Week 3 – Notes**

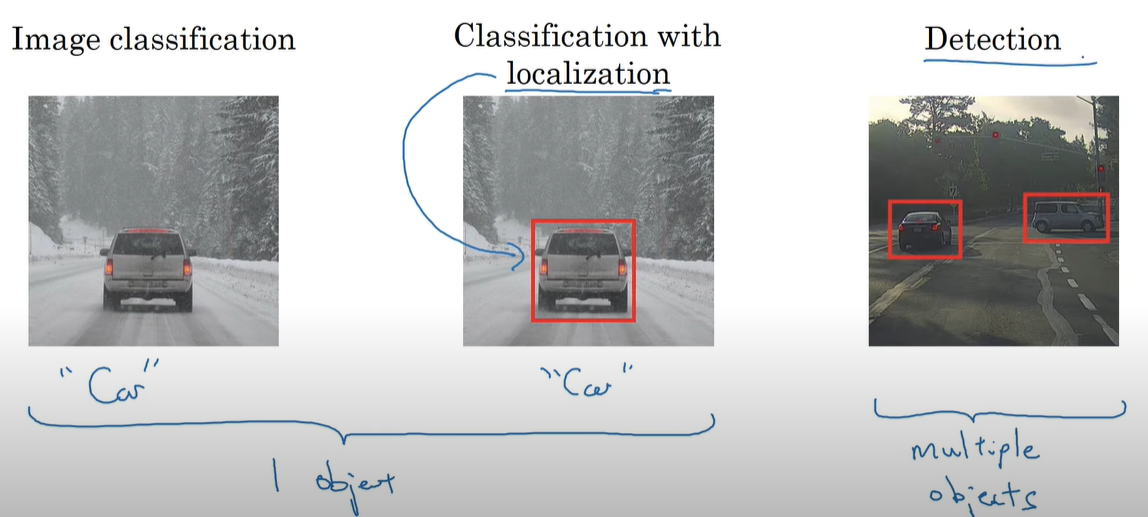
**Object Localization**

There are 3 different tasks:

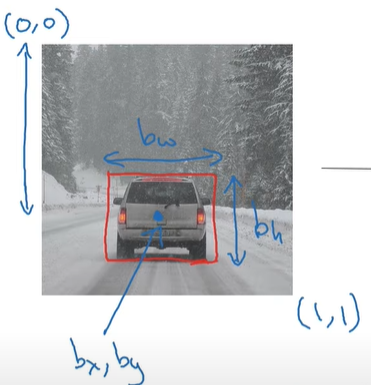
Image classification – predict what’s in the image (one object)

Classification with localization – predict what’s in the image and the bounding box coord (one object)

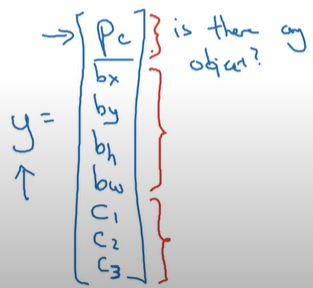
Detection – predict the classes and location of multiple objects in an image



For classification with localization, your CNN predicts the class with a softmax function and also the coordinates of the bounding box (bx, by, bh, bw) where bx and by represent the center of the object and bw, bh the width and height of the image; in addition, the left top corner of the image has the coordinates 0,0 and the bottom right 1,1 so the annotated coordinates are in relation to these ones

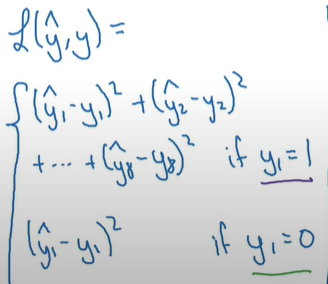


The CNN’s output is defined in the following way:



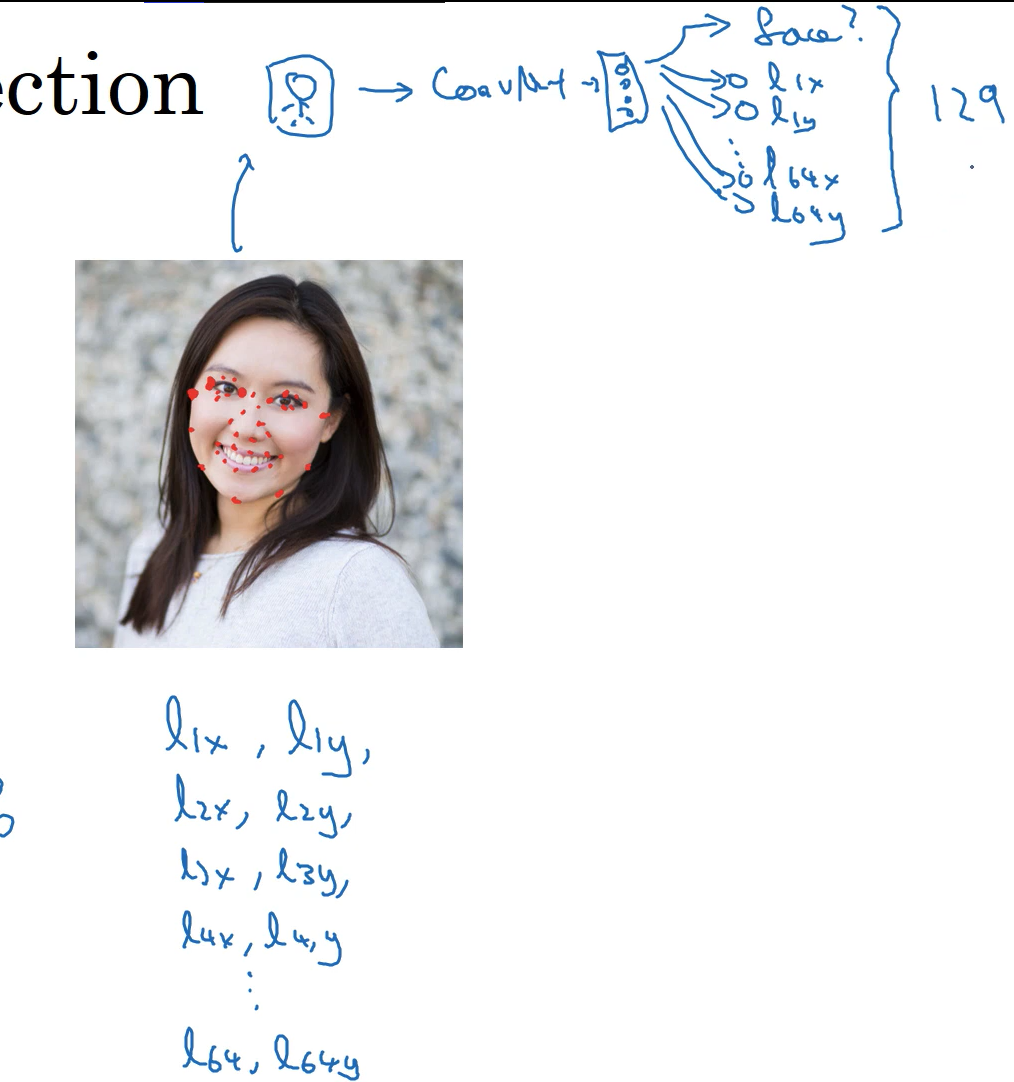
It’s important to have an output that specifies if we have or not an object of interest in the image because if we don’t, the loss isn’t computed on the rest of outputs (they are don’t care)

For the loss function, we can use:

, but in practice we use log likelihood loss for c1-3, square error for b and the logistic regression loss for pc

**Landmark Detection**

Exactly as you predict the bounding box of an object, exactly in the same way you can predict the position of landmarks (for example special points picked on the human face) + an output that predicts if a face is in the image or not



This landmark detector represents the baseline for snapchat filters (AR), sentiment detection and pose detection

It’s important to remain consistent across the labeled training set when placing landmarks

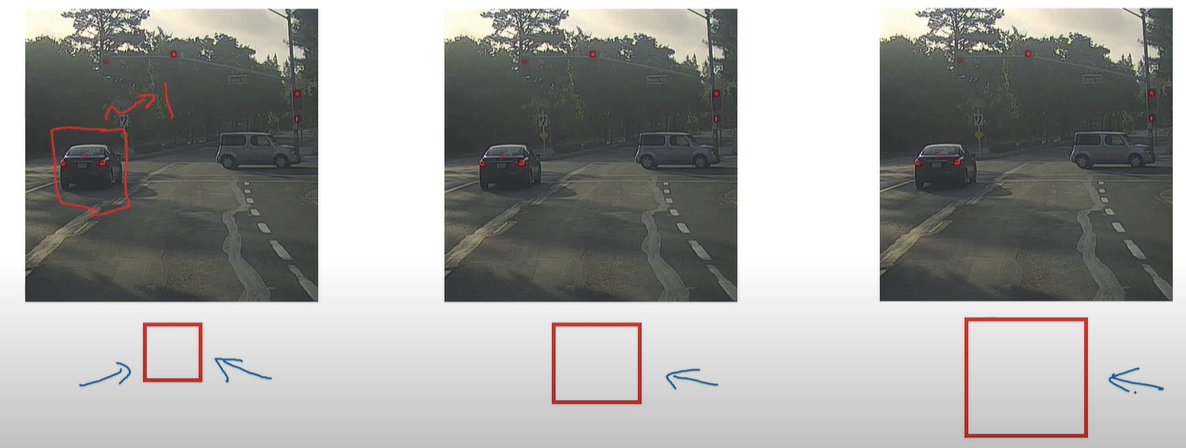
**Object Detection**

For this task we want to predict the class and bounding box of each object of interest from an image; the bounding boxes have to be placed so that they include as little as possible of the object surroundings



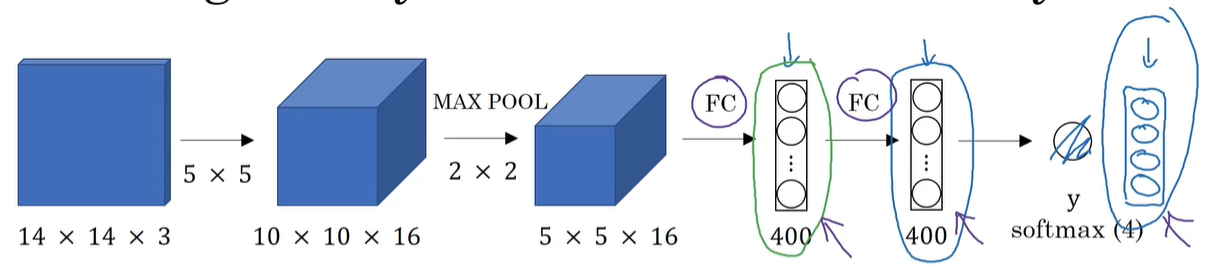
The most basic implementation is to create sliding windows of various sizes and to slide them on the image so that then you classify each region; this technique is called sliding windows detection

The problem is that the computational cost is extremely high but if you have a large stride (to reduce the number of classified regions) you can skip by mistake important image regions

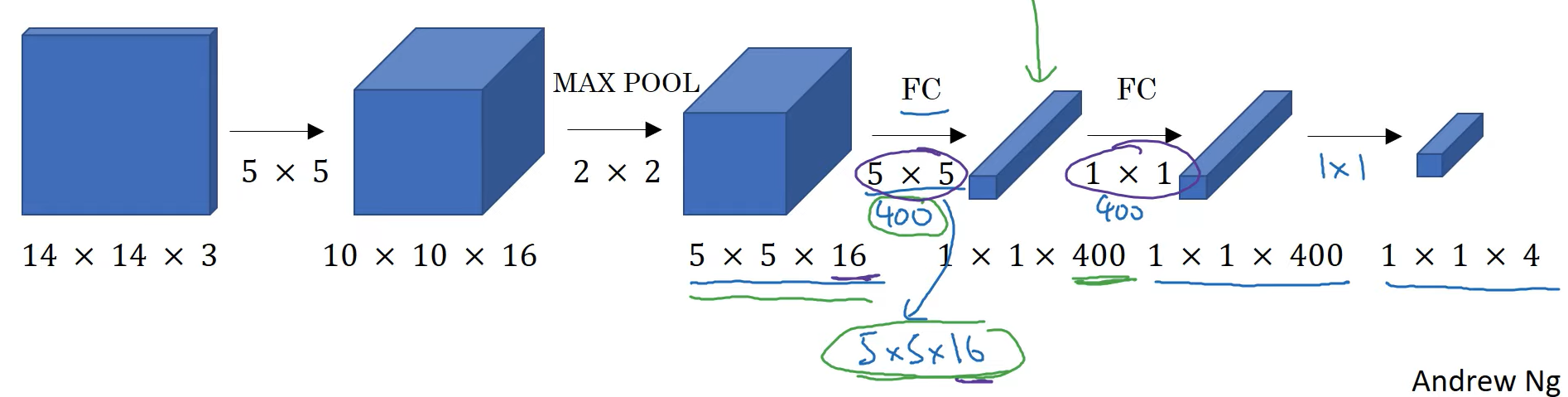


**Convolutional Implementation Sliding Windows**

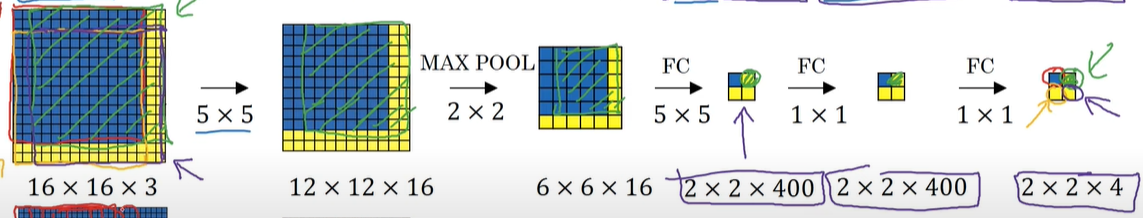
A fully connected layer can be implemented with a convolutional layer



For example, in order to implement the first FC layer, we can use instead a CONV2D that has as many filters as the number of neurons in the FC layer (400 filters) of size as the previous layer (5x5x16)



Instead of applying the whole CNN on each window, we can apply the CNN on the entire image because many input windows have common pixels and the extracted features can be shared



After applying the CNN on the whole image, each output position with its associated channels contains the predictions of the corresponding window

**Bounding Box Predictions**

The problem of the aforementioned implementation is that bounding boxes aren’t every time squared and, additionally, we can place, by design, the bounding box with a small offset

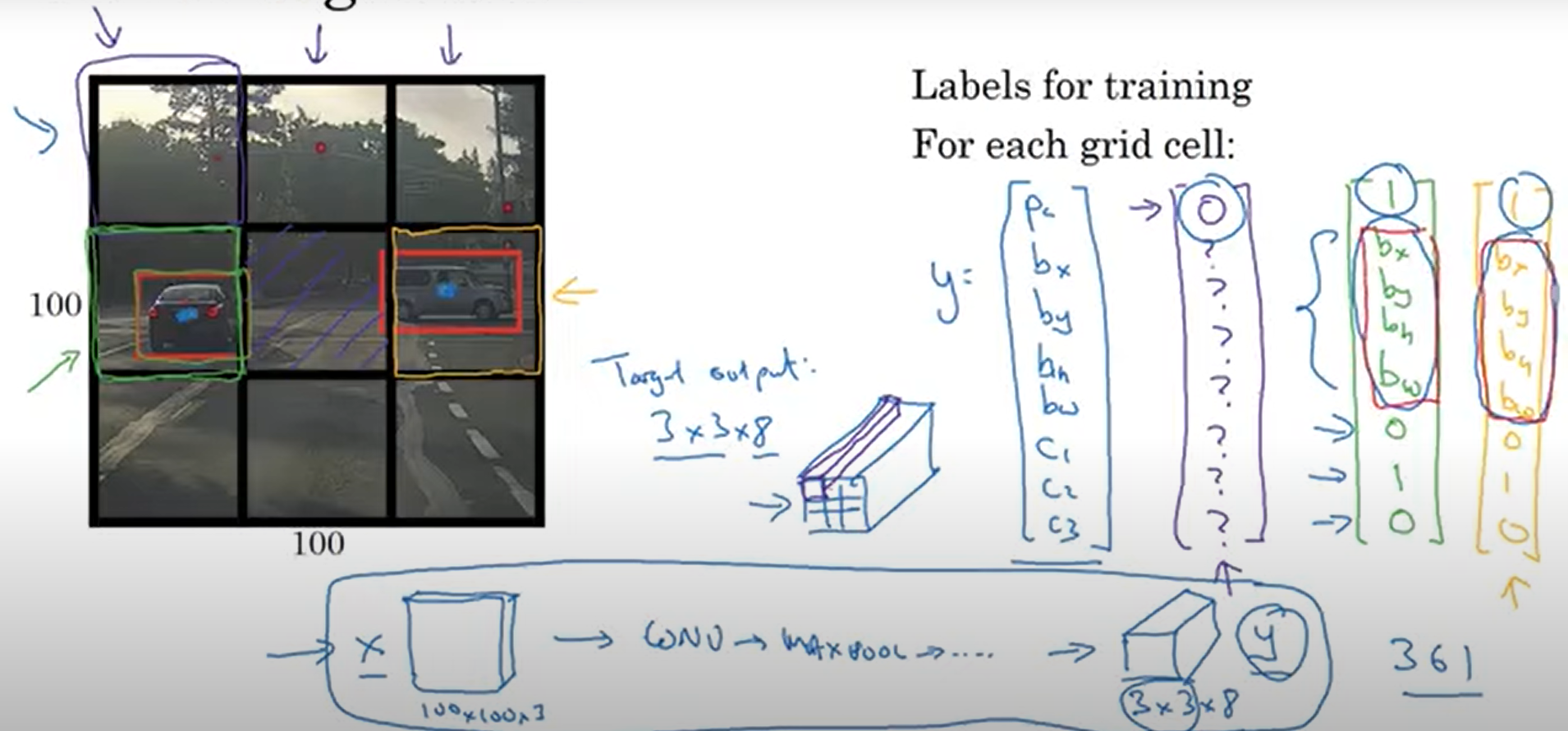
The YOLO algorithm, published in 2015, improves the convolutional implementation with sliding windows in the following way:

- each object is assigned to a window if its center falls inside it (one object only to one window)

- we take into consideration the bounding box of the object that will be predicted

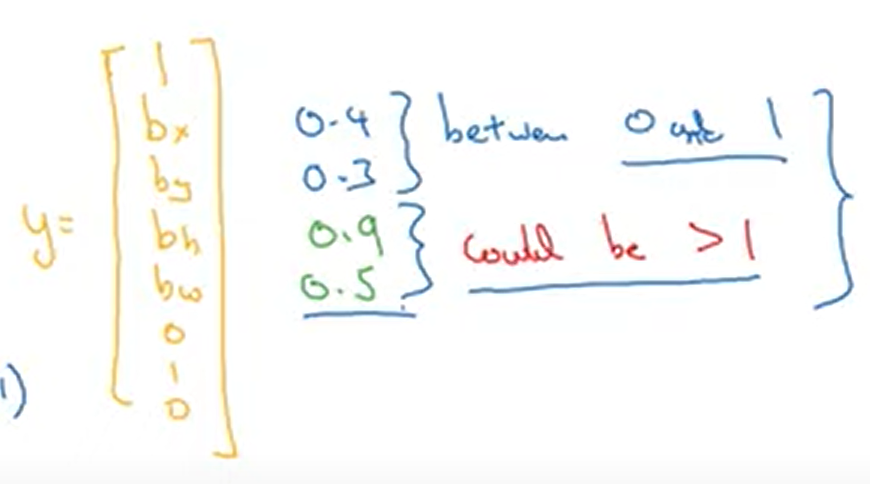
- the coordinates of each object are represented with respect to its associated window

- to avoid windows that have multiple object centers inside them, we use a fine grid (e.g.: 19 x 19)



YOLO algorithm is really fast because it detects all objects and classifies them in one shot

Because bounding boxes are defined in relation to the associated window, bx and by are between 0 and 1, but bh and bw can be bigger than 1 considering that one object can span to many windows



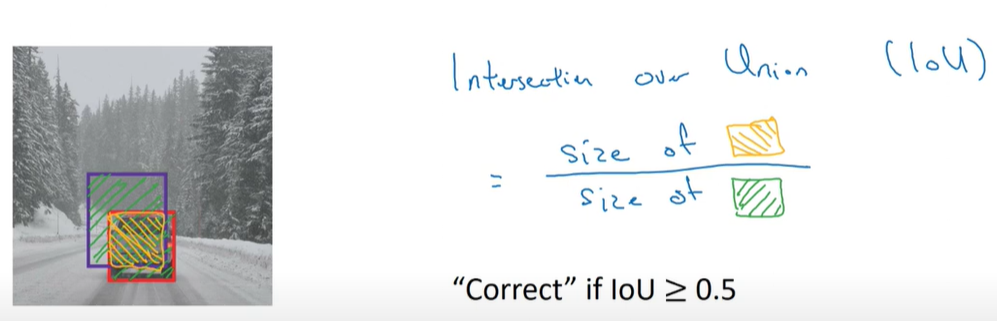
Each window has coordinates between 0 and 1

**Intersection Over Union**

In order to evaluate the object localization, we may want to compute the IoU, as the ration between intersection area of predicted bb and the ground truth bb and reunion of these 2 bb

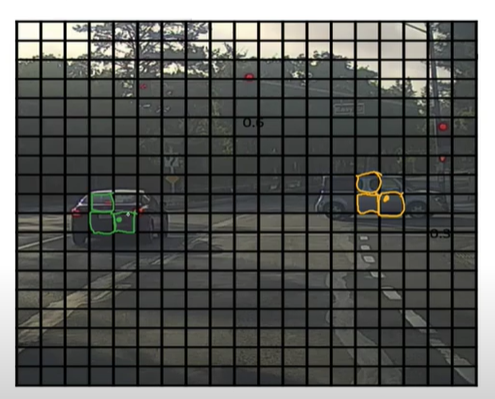
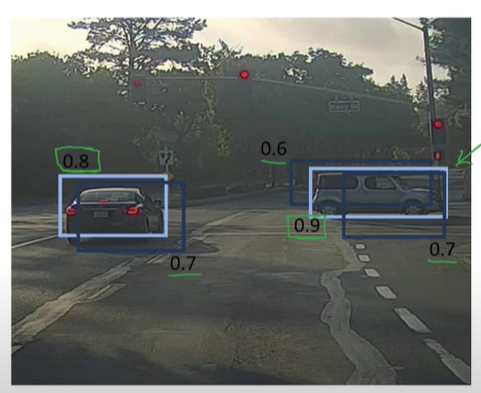
In practice, the usual minimum accepted IoU is 0.5

However, many researchers use 0.6 or even 0.7 for more overlapping, but they don’t use values lower than 0.5



**Non-max Suppression**

We want a way to get rid of multiple predictions for the same object, so that we end up with the best estimated



The non-max suppression algorithm can help us achieve that regardless of the number of objects present in the image

Algorithm:

discard all boxes with pc <= 0.6

while there are any remaining boxes:

pick the box with the largest pc and output it as a prediction

discard any remaining box with IoU >= 0.5 with the box output in the previous step

**Anchor boxes**

YOLO paper introduced the concept of anchor boxes that represent predefined boxes that are places in each cell, so that more objects that have the center in once cell can be detected (it’s solve the detection of overlapping objects)

The difference is that the CNN now outputs 8 values for each anchor box of each cell

A whiteboard with text and pictures

Description automatically generated

Previously: each object in training image is assigned to grid cell that contains that object’s midpoint (the output size is cells\_no \* outputs\_per\_cell)

With 2 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 (the output size is cells\_no \* outputs\_per\_cell \* anchors\_per\_cell)

Help the network to specialize in detecting specific objects (tall, wide etc)

If in one cell 2 objects have a midpoint and they are best represented by the same type of anchor box, in that cell only one object can be detected

Researchers define 5-10 anchor boxes, so in each cell can be detected maximum 5-10 objects

However, it’s fairy rare to find 2 objects in the same cell (considering we are using 19 x 19 cells)

Later YOLO version used KNN to find stereotypical shapes of anchor boxes

**YOLO Algorithm**

Training

Takes the image and splits it into a grid, and for each grid it output a number of values equal to number of anchors \* (1 + 4 points that determine the bounding box + number of classes)

For this algorithm, if you have 3 classes, you don’t need the fourth one, which is the “unknown” class

For each object from the training set, its predefined bounding box is assigned considering the highest IoU with the actual bounding box

While training, the CNN minimizes the loss for the outputs

Making predictions

It predicts the in one step all the detections from each cell

Outputting the non-max suppressed outputs

For each grid cell, get 2 predicted bounding boxes

Get rid of low probability predictions

For each class at a time use non-max suppression to generate final predictions

**Region Proposals**

R-CNN paper was published in 2013 and it’s based on the fact that for a sliding window approach, not each region has to be classified, so initially a segmentation algorithm is applied on the image to find blobs (regions of interest)

R-CNN stands for Region Based Convolutional Neural Networks

A person standing on a street

Description automatically generated

However, too many regions are found, and the model takes a lot of time because it sequentially classifies them; thus, some improvements were made

Fast R-CNN: propose regions and use convolution implementation of sliding windows to classify all the proposed regions (2015)

Faster R-CNN: like the previous version, but it uses a convolutional network to propose regions (2016)