深度学习-第九章-目标检测

黄海广 副教授

2021年05月

本章目录

1.目标定位

01 目标定位

02 目标检测算法

03 YOLO算法

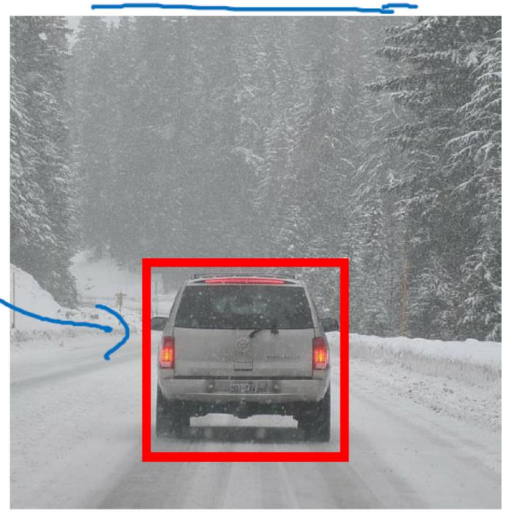
04 Faster RCNN算法

1.目标定位

Image classification

"Car"

Classification with localization



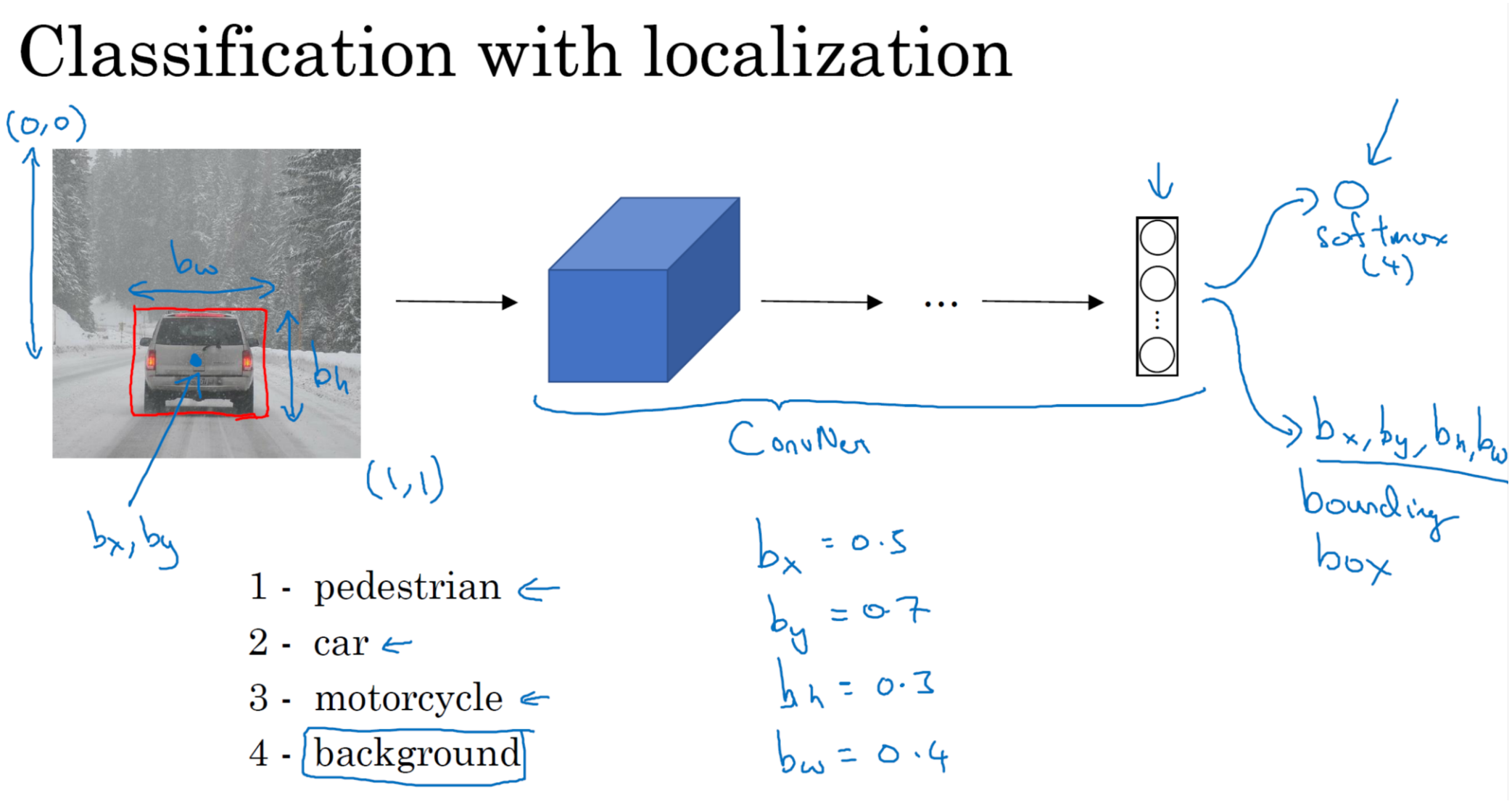
"Car"

1 object

Detection

multiple objects

1.目标定位


Classification with localization
(0,0)
bw
bh
bx, by
(1,1)
ConvNet
softmax
(4)
bx, by, bh, bw
bounding box
1 - pedestrian
2 - car
3 - motorcycle
4 - background
bx = 0.5
by = 0.7
bh = 0.3
bw = 0.4

1.目标定位

目标标签y的定义如下：

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

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

$\mathcal{L}(\hat{y}, y) =$

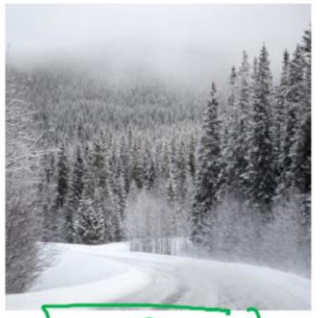
$\begin{cases} (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \cdots + (\hat{y}_8 - y_8)^2 & \text{if } y_1 = 1 \\ (\hat{y}_1 - y_1)^2 & \text{if } y_1 = 0 \end{cases}$

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

$x =$

$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$ Is there an object?

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



$\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$ p_c ← "don't care"

Andrew Ng

2.目标检测算法

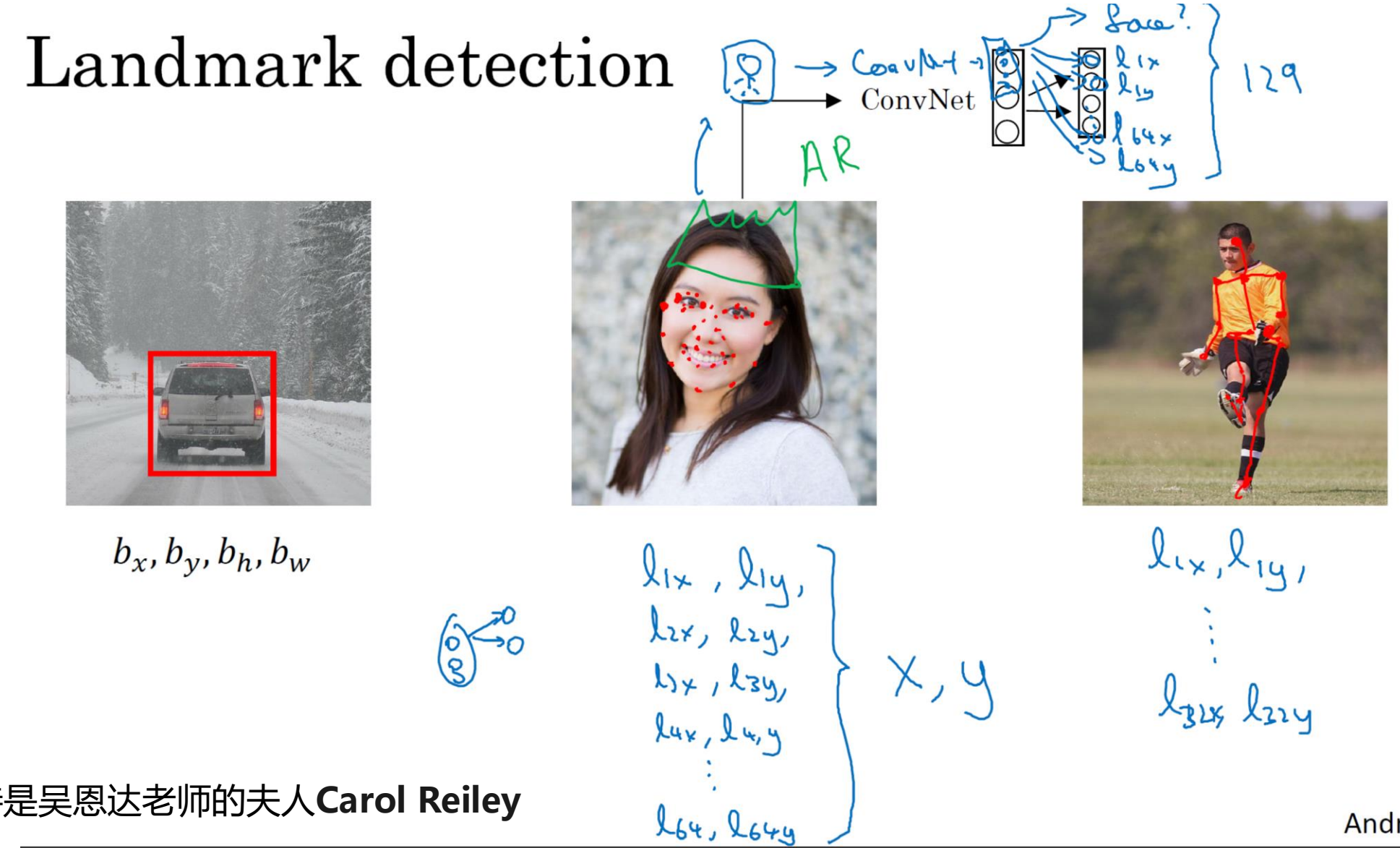
01 目标定位

02 目标检测算法

03 YOLO算法

04 Faster RCNN算法

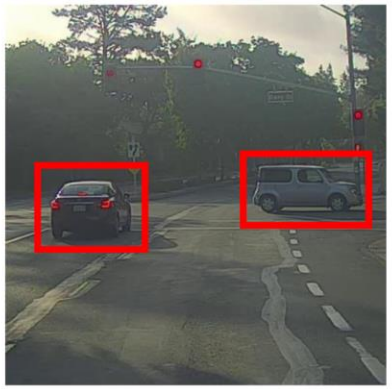
2.目标检测算法



备注：图中的模特是吴恩达老师的夫人**Carol Reiley**

2.目标检测算法

汽车检测案例



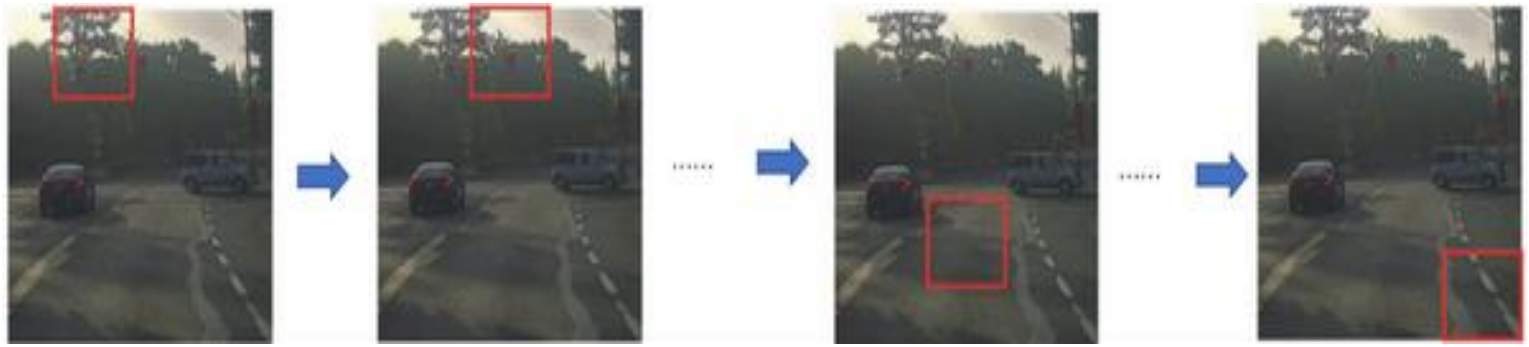
Training set:

x	y
	1
	1
	1
	0
	0

ConvNet → y

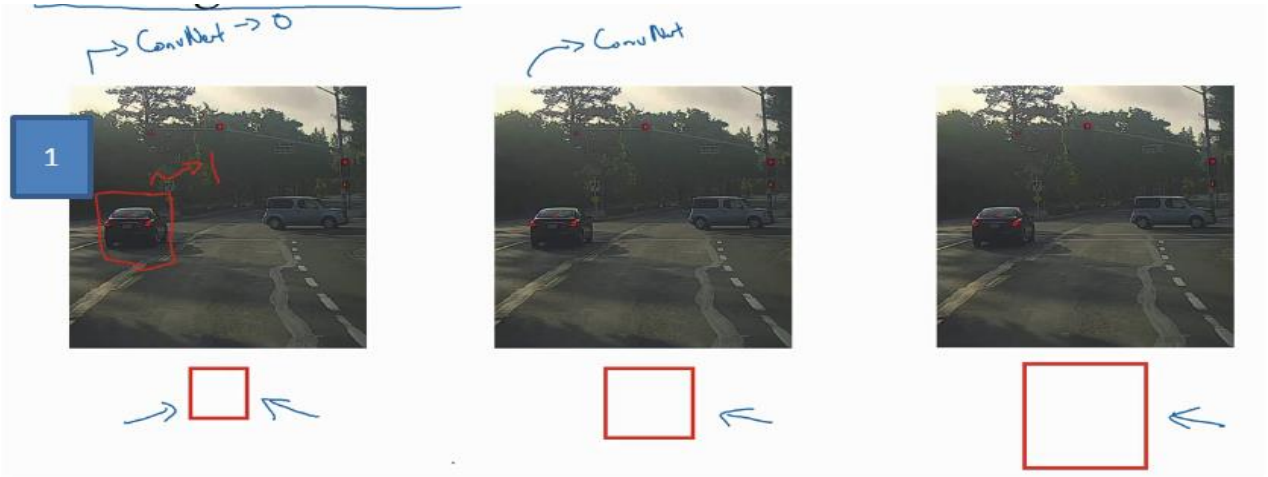
2.目标检测算法

滑动窗口检测



2.目标检测算法

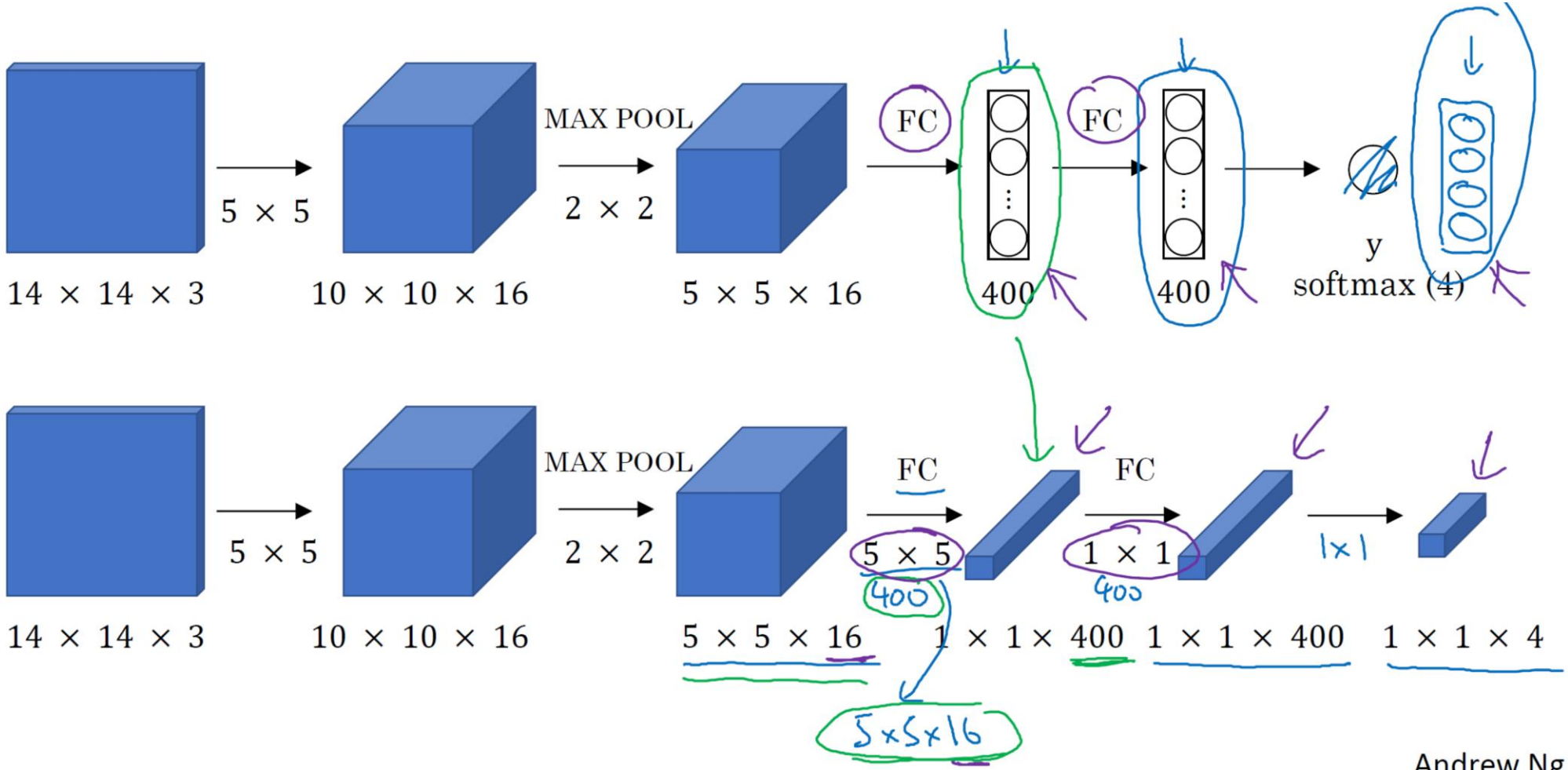
滑动窗口检测



滑动窗口目标检测算法也有很明显的缺点，就是计算成本，因为你在图片中剪切出太多小方块，卷积网络要一个个地处理。如果你选用的步幅很大，显然会减少输入卷积网络的窗口个数，但是粗糙间隔尺寸可能会影响性能。反之，如果采用小粒度或小步幅，传递给卷积网络的小窗口会特别多，这意味着超高的计算成本。

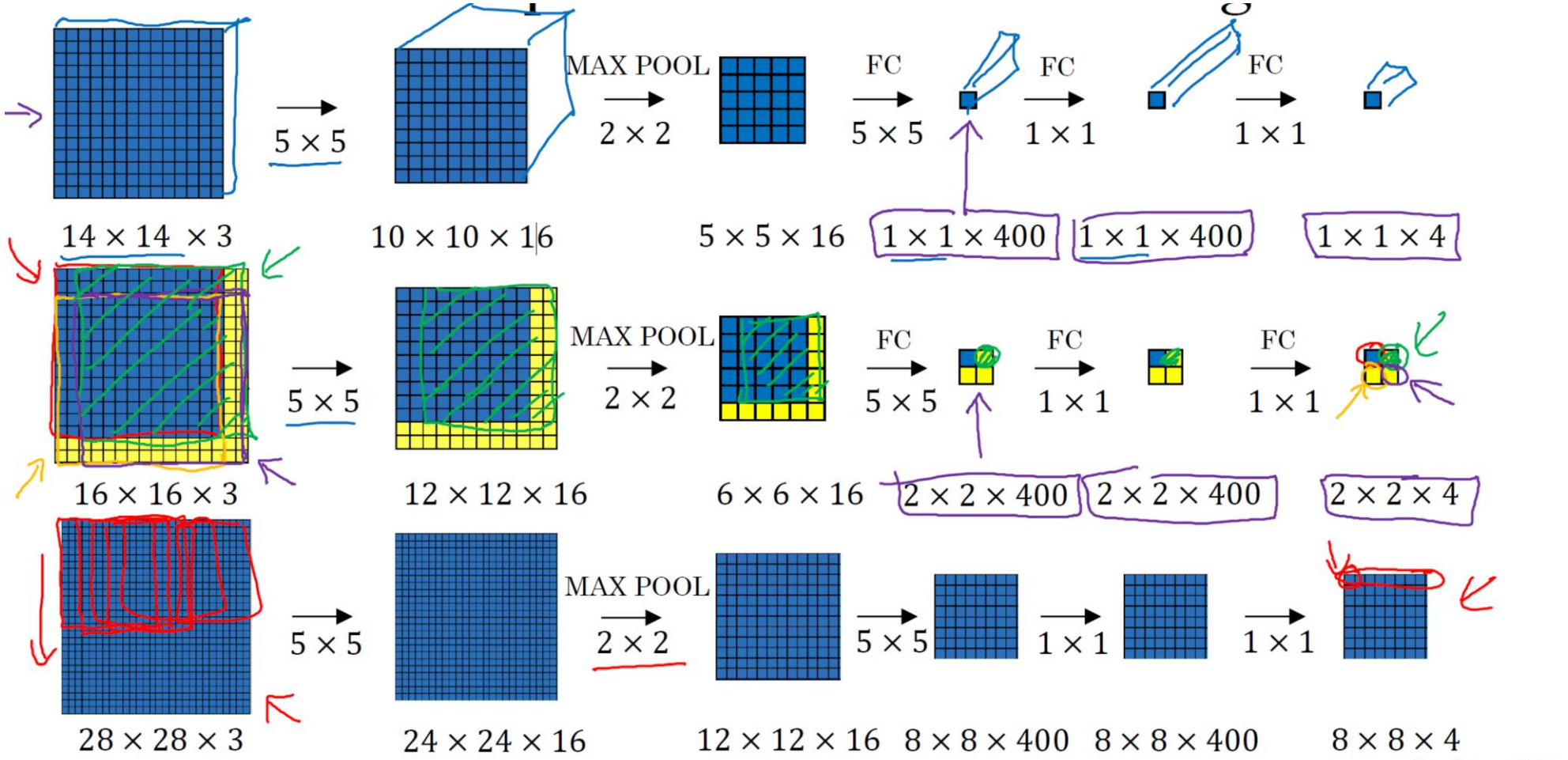
2.目标检测算法

滑动窗口的卷积实现



2.目标检测算法

滑动窗口的卷积实现

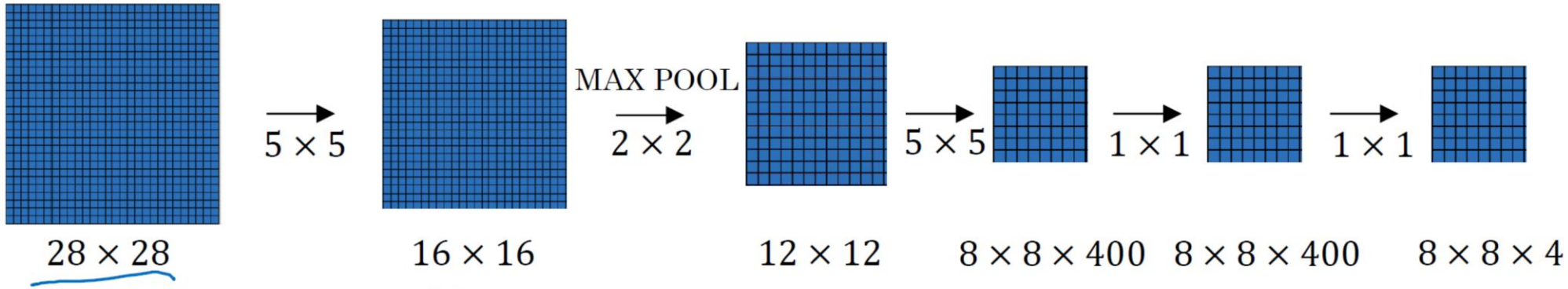


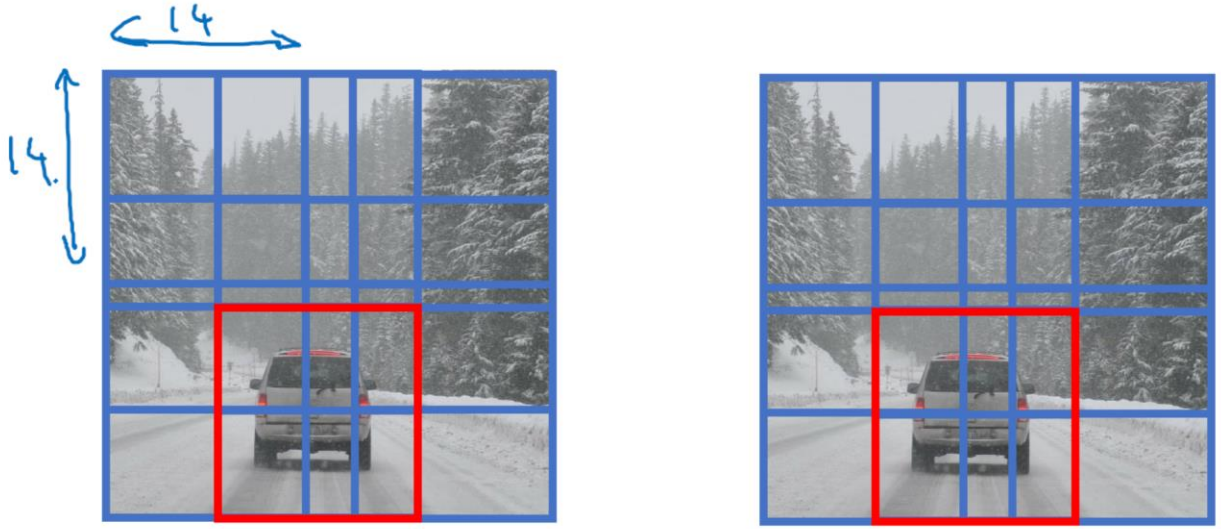
[Sermanet et al., 2014, OverFeat: Integrated recognition, localization and detection using convolutional networks]

Andrew Ng

2.目标检测算法

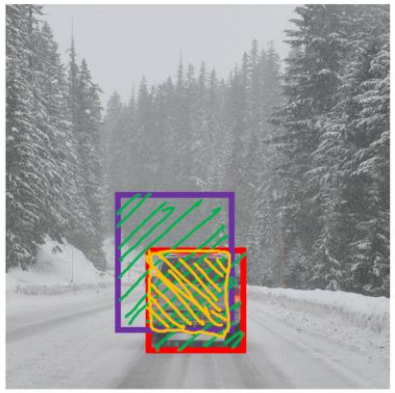
滑动窗口的卷积实现





2.目标检测算法

交并比

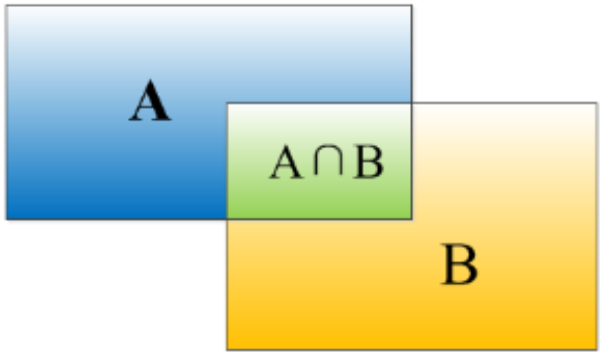


Intersection over Union (IoU)

$$= \frac{\text{size of}}{\text{size of}}$$

"Correct" if IoU ≥ 0.5 ←

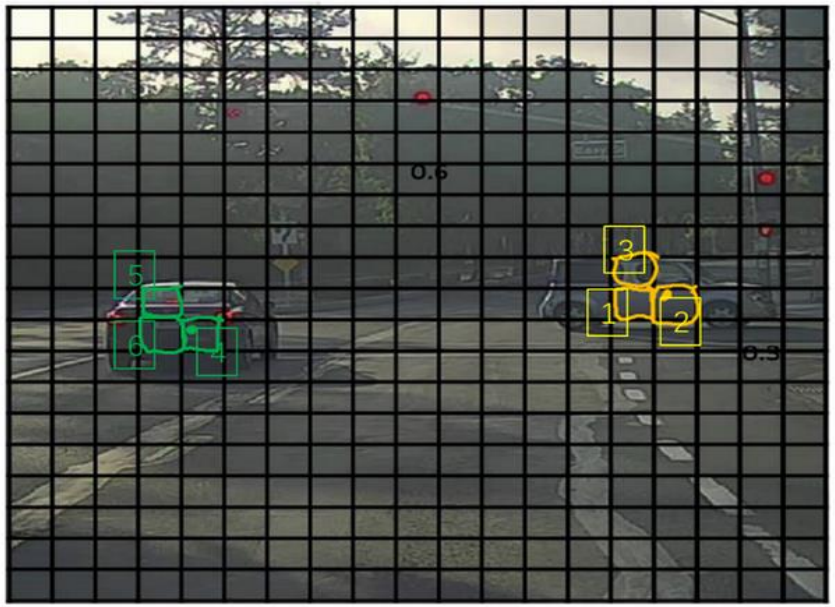
0.6 ←



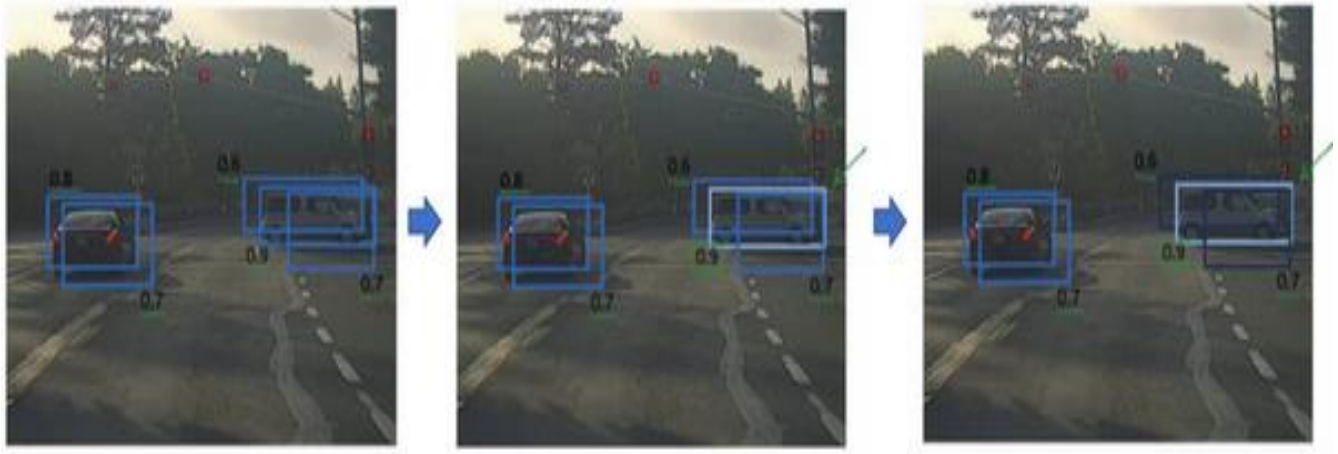
交并比：**IOU=(A∩B)/(A∪B)**

2.目标检测算法

非极大值抑制（Non-max suppression）



首先这个19×19网格上执行一下算法，你会得到19×19×8的输出尺寸。不过对于这个例子来说，我们简化一下，就说你只做汽车检测，我们就去掉c_1、c_2和c_3，然后假设这条线对于19×19的每一个输出，对于361个格子的每个输出，你会得到这样的输出预测，就是格子中有对象的概率（p_c），然后是边界框参数（b_x、b_y、b_h和b_w）。如果你只检测一种对象，那么就没有c_1、c_2和c_3这些预测分量。

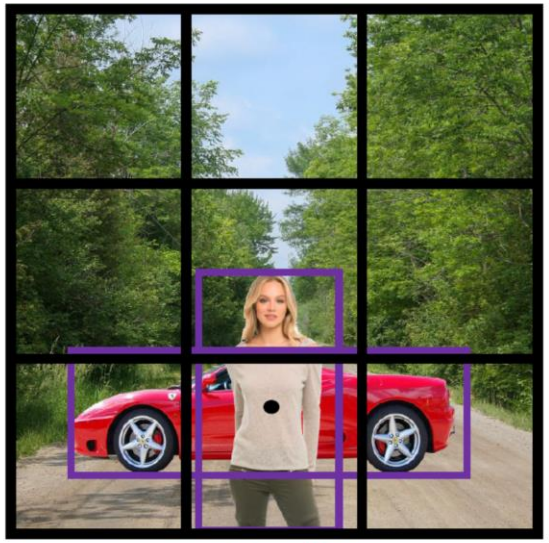


3.YOLO算法

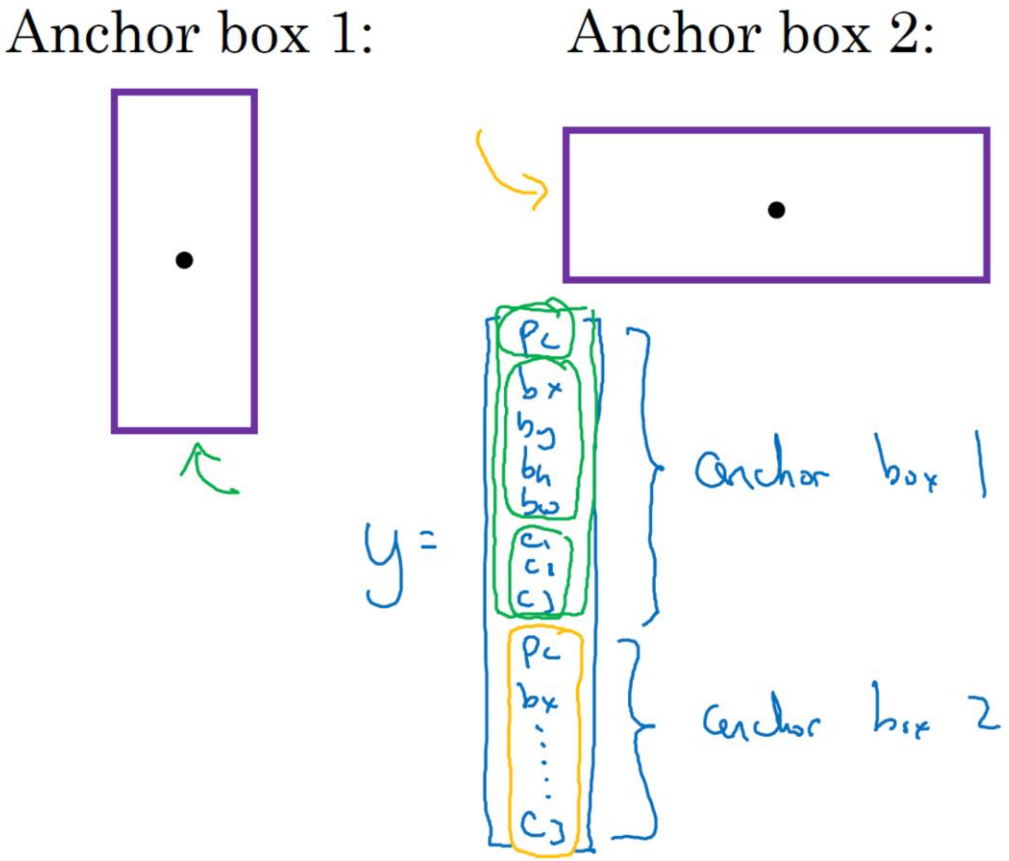
3.YOLO算法

3.YOLO算法

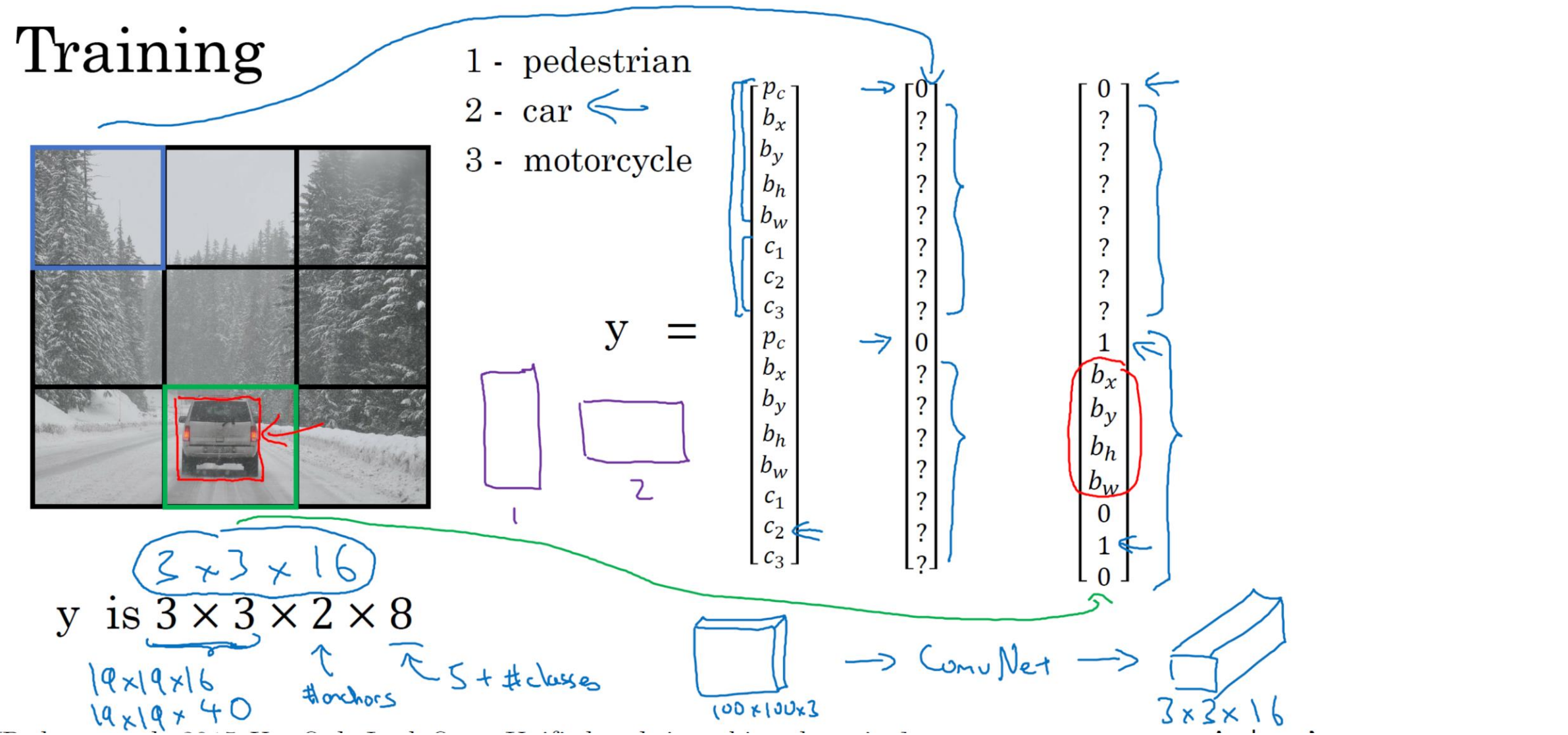
Anchor Boxes



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

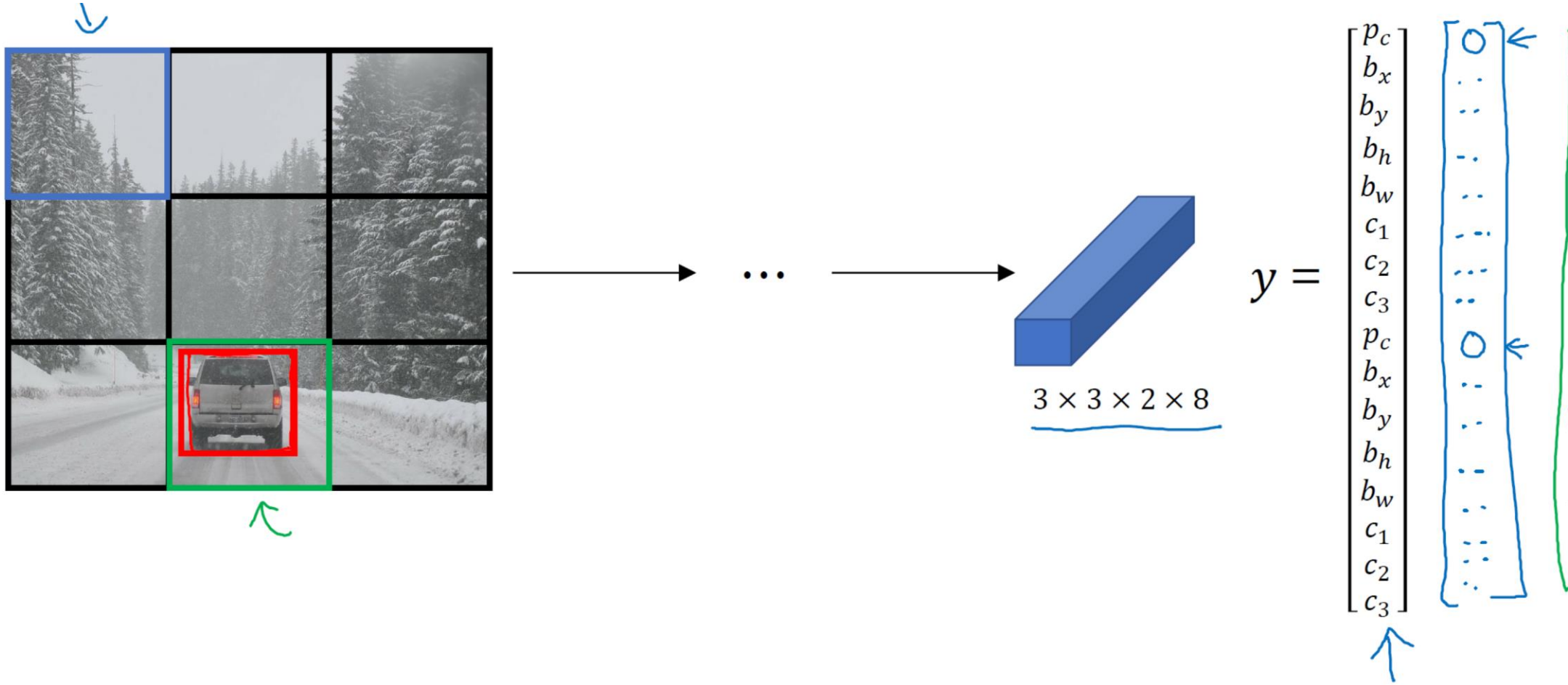


3.YOLO算法


Training
1 - pedestrian
2 - car
3 - motorcycle
y =
y is 3 × 3 × 2 × 8
3 × 3 × 16
19×19×16
19×19×40
#anchors
5 + #classes
100×100×3
ConvNet
3×3×16

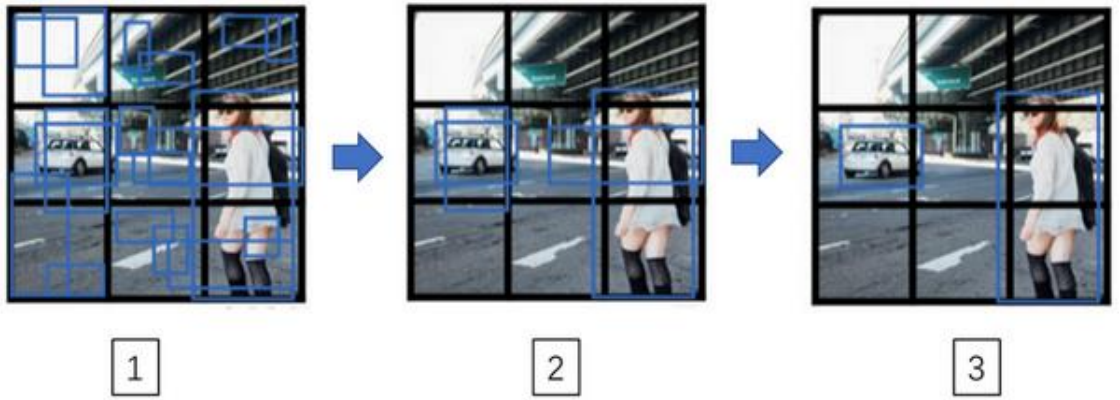
3.YOLO算法

进行预测



3.YOLO算法

输出非极大化抑制的结果



最后你要运行一下这个非极大值抑制，为了让内容更有趣一些，我们看看一张新的测试图像，这就是运行非极大值抑制的过程。如果你使用两个**anchor box**，那么对于9个格子中任何一个都会有两个预测的边界框，其中一个的概率p_c很低。但9个格子中，每个都有两个预测的边界框，比如说我们得到的边界框是是这样的，注意有一些边界框可以超出所在格子的高度和宽度（编号1所示）。接下来你抛弃概率很低的预测，去掉这些连神经网络都说，这里很可能什么都没有，所以你需要抛弃这些（编号2所示）。

4.Faster RCNN算法

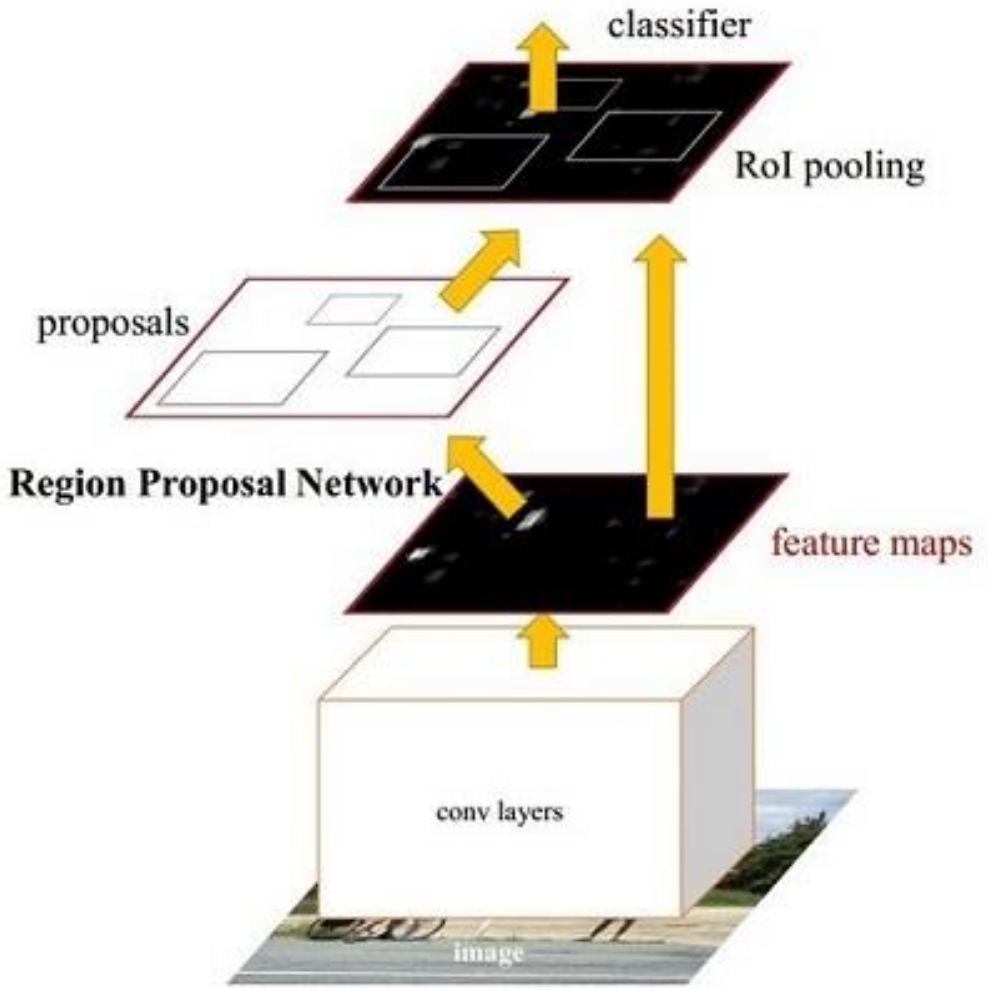
4.Faster RCNN算法

1.Conv layers

2.Region Proposal Networks

3.Roi Pooling

4.Classification

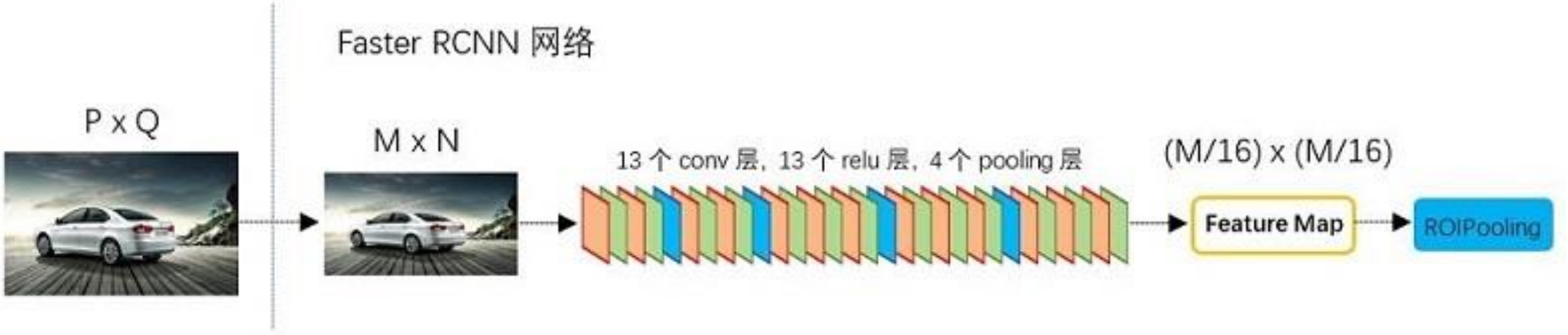


4.Faster RCNN算法

Faster RCNN训练步骤

- 第一步，训练RPN，该网络用ImageNet预训练的模型初始化，并端到端微调，用于生成region proposal；
- 第二步，训练Faster RCNN，由imageNet model初始化，利用第一步的RPN生成的region proposals作为输入数据，训练Fast R-CNN一个单独的检测网络，这时候两个网络还没有共享卷积层；
- 第三步，调优RPN，用第二步的Faster RCNN model初始化RPN再次进行训练，但固定共享的卷积层，并且只微调RPN独有的层，现在两个网络共享卷积层了；
- 第四步，调优Faster RCNN,由第三步的RPN model初始化Faster RCNN网络，输入数据为第三步生成的proposals。保持共享的卷积层固定，微调Faster RCNN的FC层。这样，两个网络共享相同的卷积层，构成一个统一的网络。

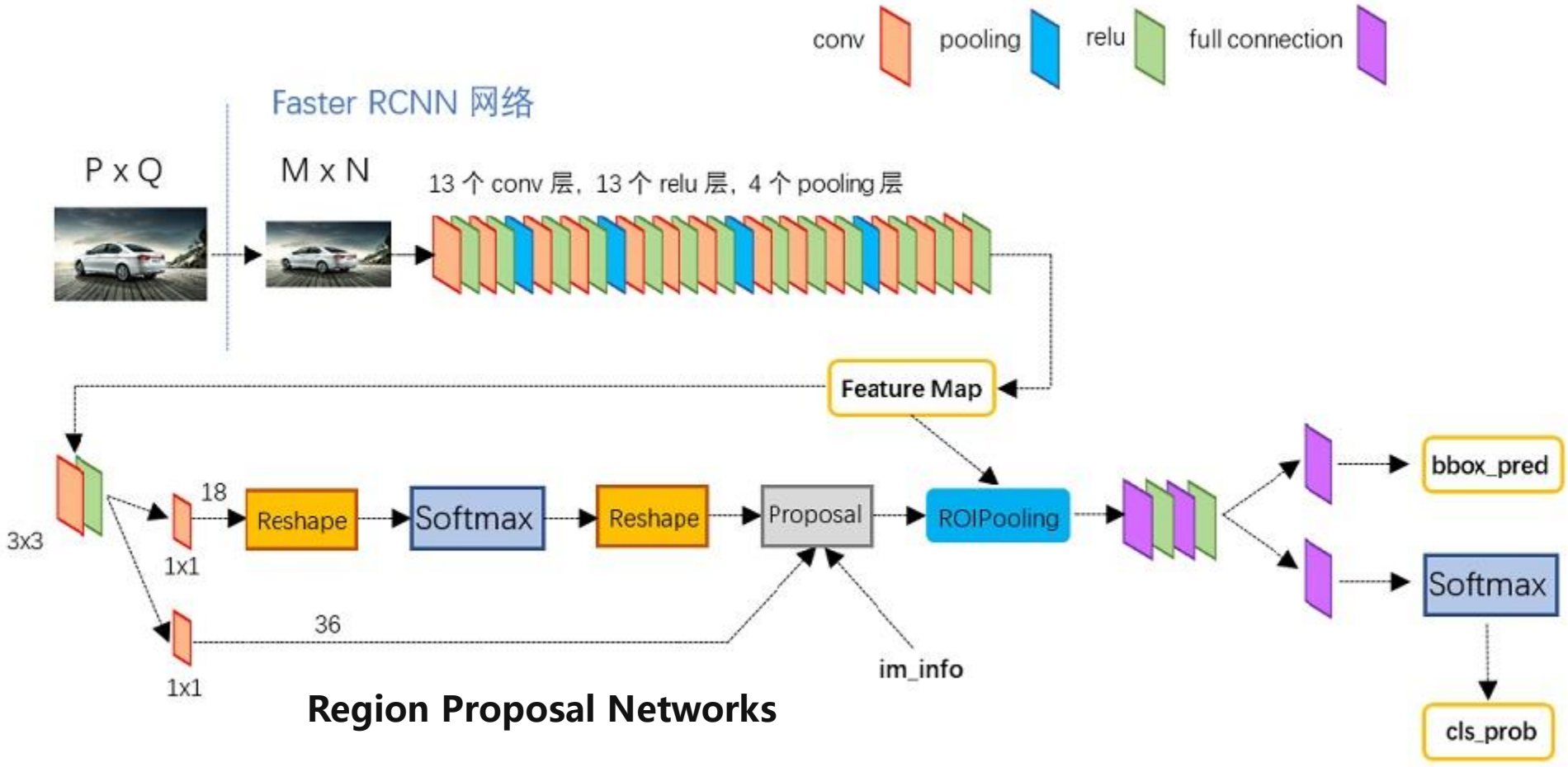
4.Faster RCNN算法



Conv layers

4.Faster RCNN算法

RPN网络的作用： RPN专门用来提取候选框，一方面RPN耗时少，另一方面RPN可以很容易结合到Fast RCNN中，成为一个整体。



参考文献

1. IAN GOODFELLOW等，《深度学习》，人民邮电出版社，2017
2. Andrew Ng，http://www.deeplearning.ai

课件、视频、代码地址

下载地址：

https://github.com/fengdu78/WZU-machine-learning-course

最新更新公布在公众号“机器学习初学者”