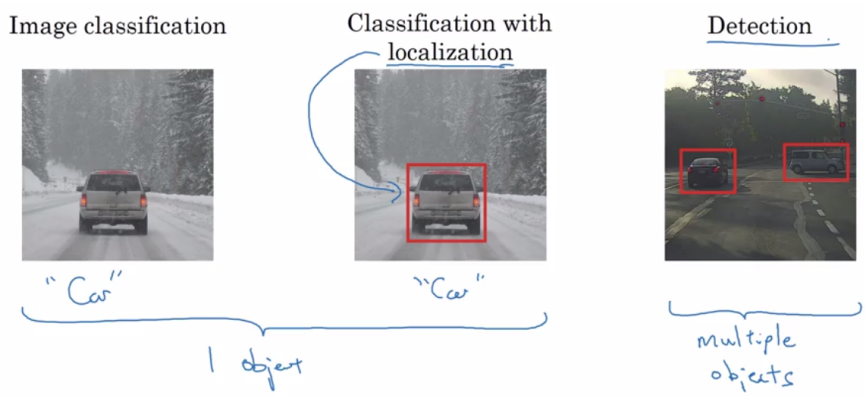
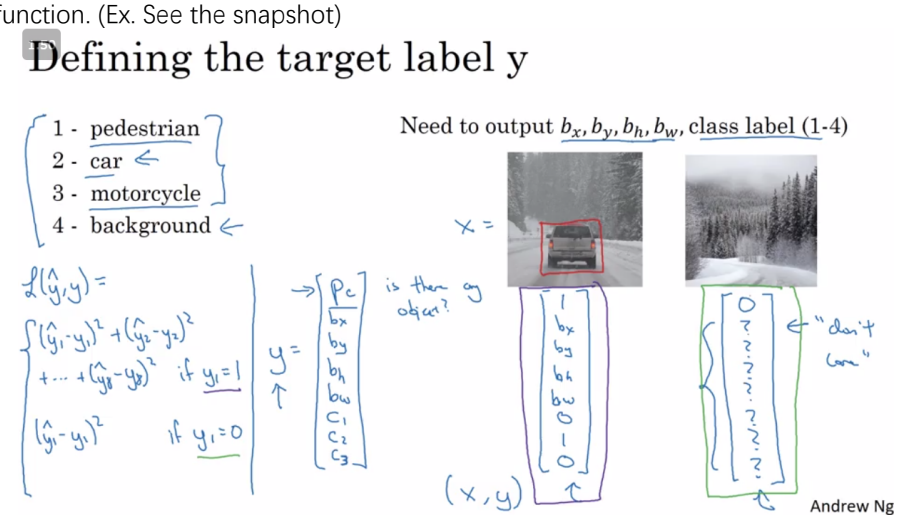
deep Learning_c4_week3

Created: 19/03/04 18:40 **Updated:** 19/03/08 17:13

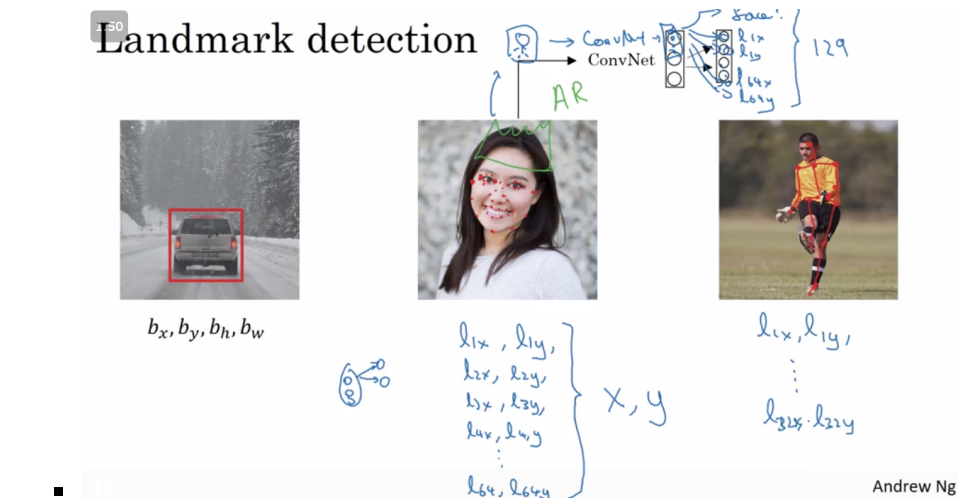
Detection algorithm

object classification -> object localization -> object detection

- 

- ## localization
  - assume **only one object** in giving sample, we want to make a classification and detect a **bounding box** of where the object is.
  - we add more output units to achieve localization. See the snapshot below.
  - annotation:
    - PC: value equal to 0/1, where 1 means there has object in image.
    - the series of b are used to describe the position of bounding box
    - the series of c are our classes
  - Loss function. (Ex. See the snapshot)
    - 

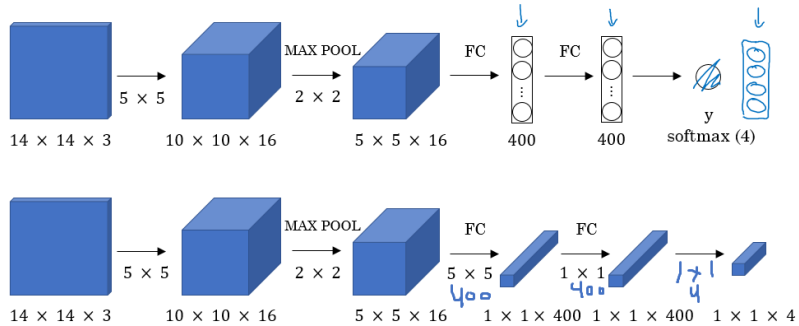
- ## Landmark
  - computer vision problem. You want to output some useful points in image.
  - EX. out line of face, pose detection, etc.



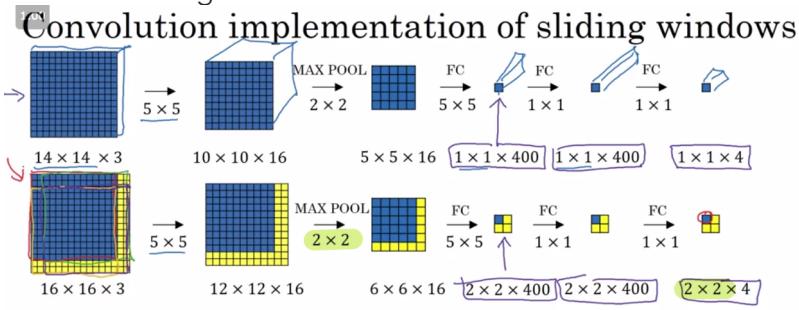
- be cautious, the meaning of output points should be **consistent**. Ex. if `l1x,l1y` is the left corner of left eye, all other `l1x,l1y` of the other examples has to be the same.
- 👆the data need people to manually label

object detection

- conventional way: **sliding windows**
  - design a **filter**
    - Conv net
    - other handcraft classifier
  - Sliding windows detection algorithm:
    - Decide a **rectangle size.**
    - Split image into rectangles of the size. Each region should be covered. You can use some strides. (if the strides are too large, it could save time but always lead to lower accuracy)
    - For each rectangle feed the image into a filter and decide if its a car or not.
    - Pick larger/smaller rectangles and **repeat** the process from 2 to 3.
    - Store the rectangles that contains the cars.
    - If two or more rectangles intersects choose the rectangle with the best accuracy.
    -
    - disadvantage: high computational cost
- improvement👉 convolutional implementation of sliding windows.
  - Turning FC layer into convolutional layers



- so, we use Conv in order to utilize the shared values contained in each small rectangle.



- weakness: the position of the rectangle **won't be so accurate**

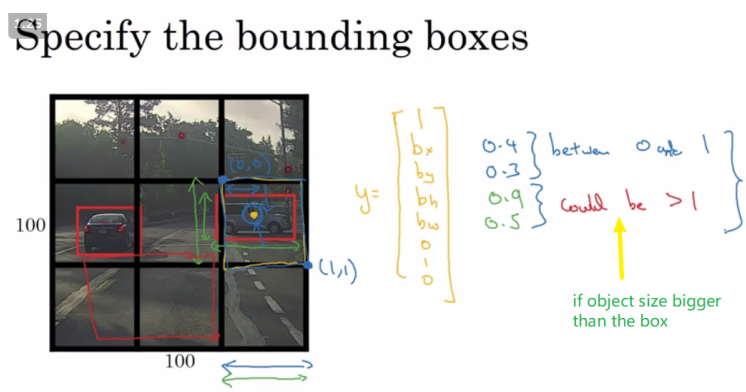
YOLO algorithm

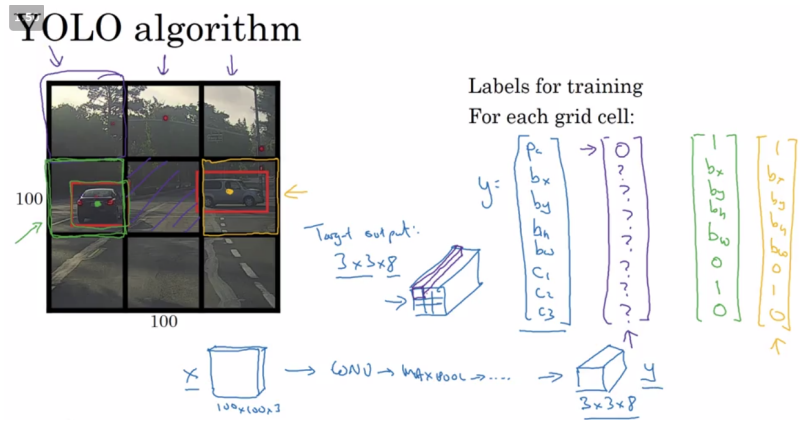
-

You Only Look Once: Unified, Real-Time Object Detection

bounding box detection

- split the image into several grids. Each grid make a detection, if their has object then gives out vector y (specify the bounding boxes).
- we assume each grid only contains **no more than one object.** (for multi-object, we will use **Archor boxes**, which will talk in below)
- **EX.**





- ## intersection over union
  - here we comes the problem: how to evaluate accuracy of a bounding box?
  - we use **IoU** (size of intersection divided by the union.)

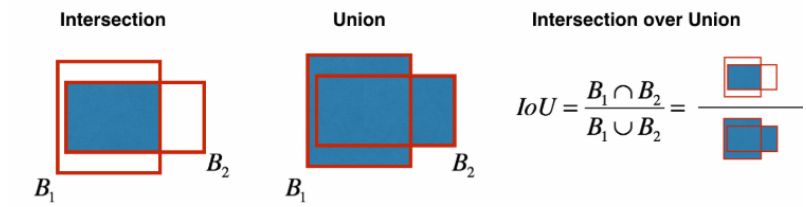
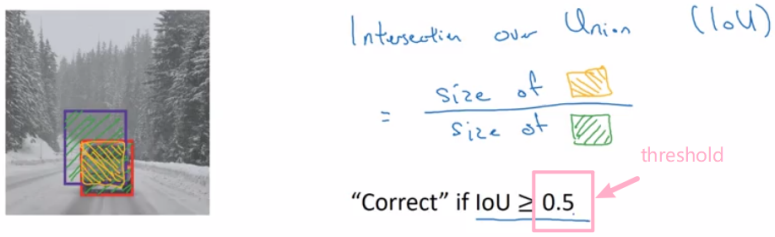
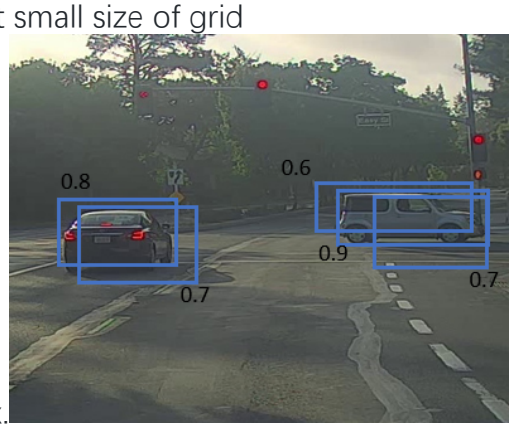


Figure 8 : Definition of "Intersection over Union".

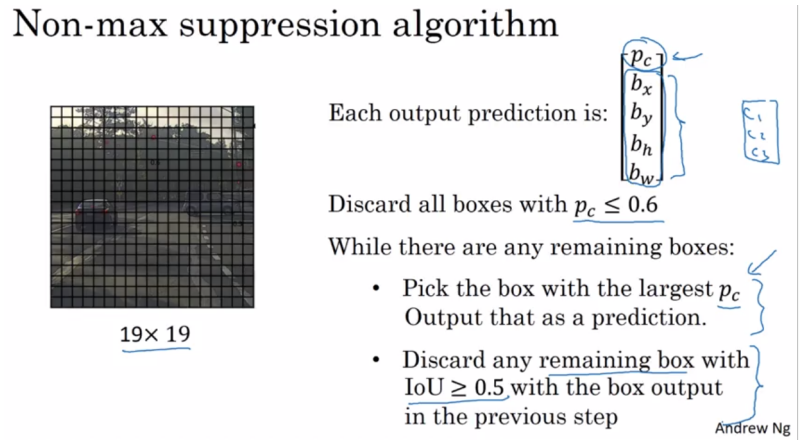
  - and we can set threshold to judge.
    - EX.



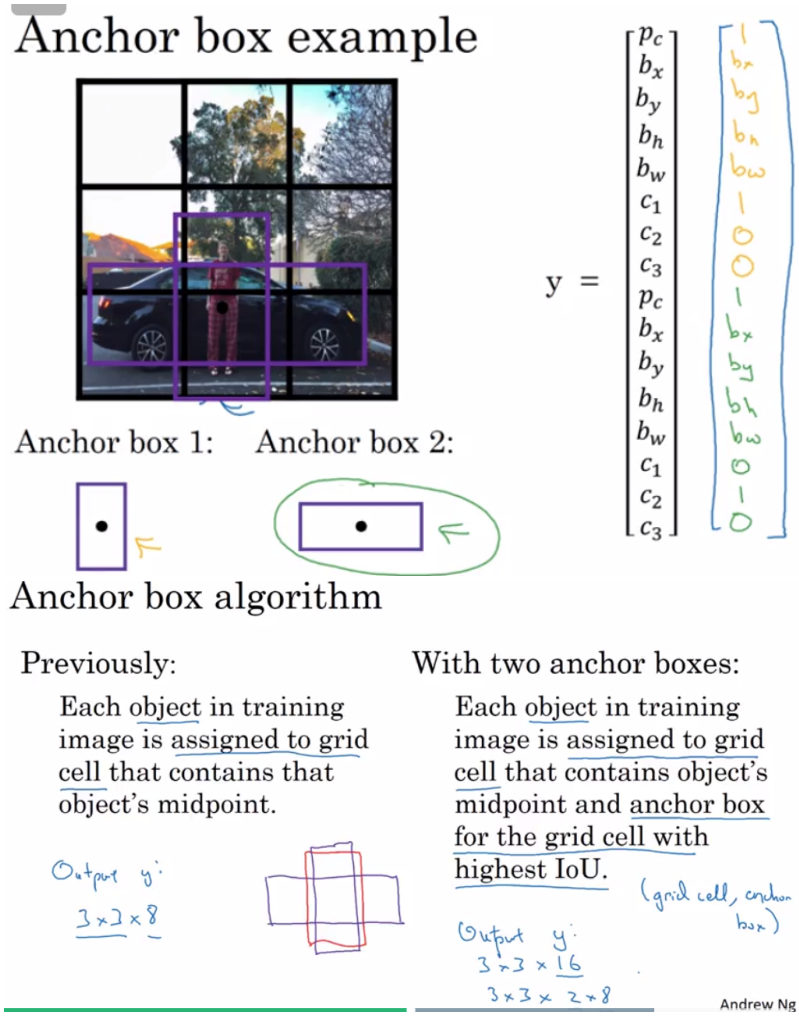
- ## Non-Max Suppression
  - Apparently, we want to detect each object for just one time. But we might get several bounding boxes for one object as we have set small size of grid
    - Ex.



  - So it's necessary for us to **clean the redundant boxes** and choose the best one. Here comes the non-max suppression.
  - briefly speaking, we only select the box with highest possibility (we can use IoU as the metric). Further more, we also suppress other boxes.
  - Algorithm with Ex.



- ## Anchor Boxes
  - **pre-design anchor boxes** to detect multiple classes in a grid
  - we can't handle the situation when number of objects > number of anchor boxes.



  - **K-Means** could help to make anchor boxes.
- summary:
  - YOLO is a state-of-the-art object detection model that is fast and accurate
  - use non-max suppression and anchor box to make multi-detection.
- what's more: Region proposal **R-CNN**
  - first select some regions by segmentation algorithm, then do obj-detection on those selective regions.

- for example the segmentation algorithm produces 2000 blob then we should run our classifier/CNN on top of these blobs.

Region proposal: R-CNN

-