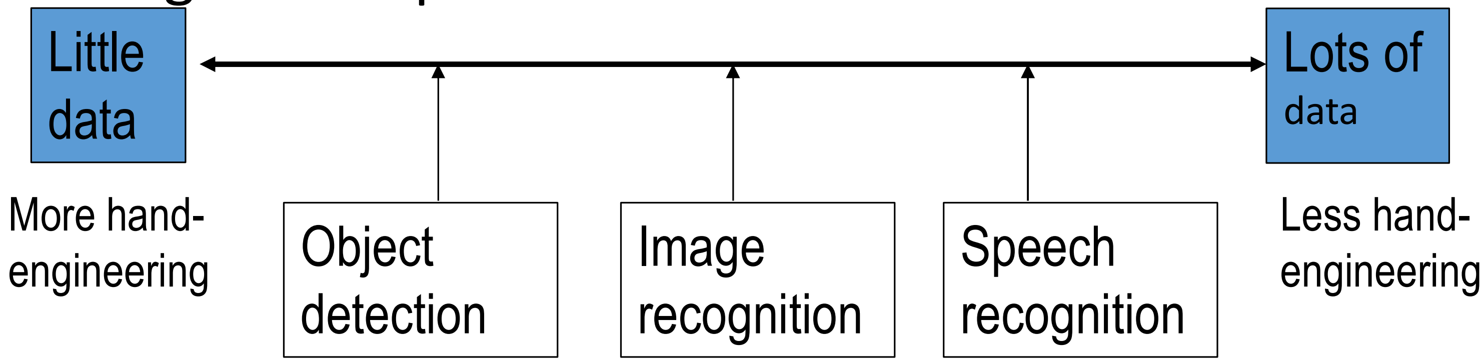
you only look once-Yolo

Outline

- Introduction
- Object localization
- Object detection
  - Sliding window detection
  - Convolutional implementation
- YOLO
  - Bounding box predictions

Introduction

- Deep learning has been successfully applied to computer vision, natural language processing, speech recognition, online advertising, etc.
- Deep learning for computer vision:



  - Two sources of knowledge:

  1) Labeled data; 2) Hand engineered features/network architecture

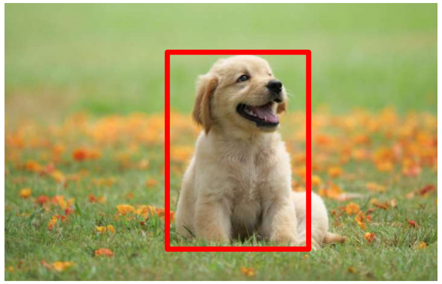
Object localization

- Localization vs Detection
  - Single object

Image classification

“dog”

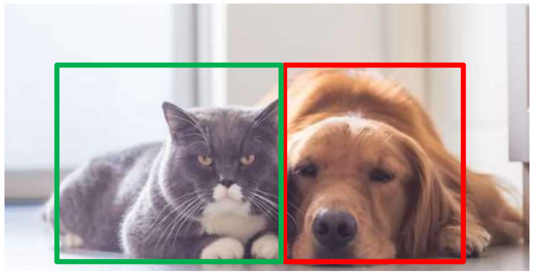
Classification with localization



“dog”

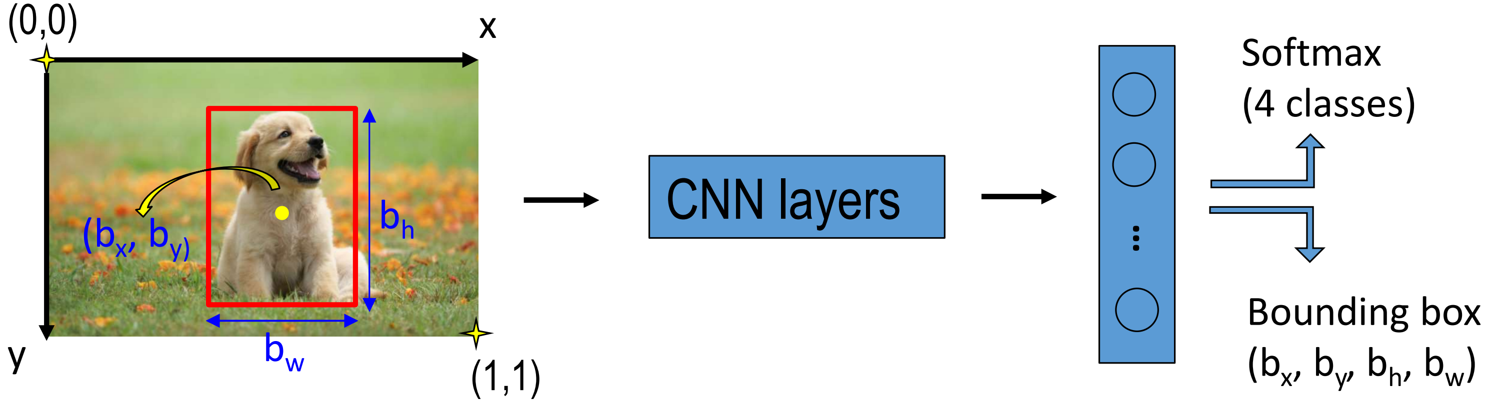
  - Multiple objects

Detection



Object localization

- Classification with localization

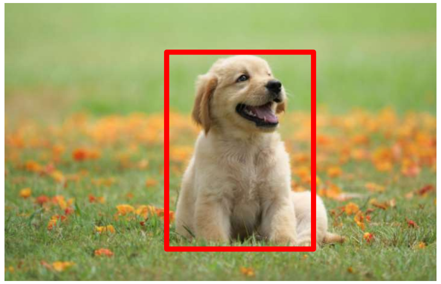


$b_x = 0.55$
$b_y = 0.53$
$b_h = 0.70$
$b_w = 0.35$

Class labels:
1. dog
2. cat
3. bird
4. background

Object localization

- Defining the target label y
    - Need to output b_x, b_y, b_h, b_w, and class label (1-4)

x = 

Classes (C):

1 - dog
2 - cat
3 - bird

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \end{bmatrix}$$

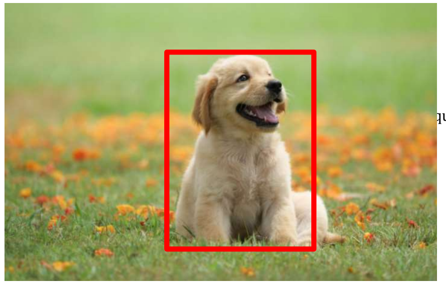
0/1 flag indicating "Is there any object?"

Bounding box information

The object is of Class 1, or Class 2, or Class 3

Object localization

- Defining the target label y
  - Need to output b_x, b_y, b_h, b_w, and class label (1-4)

x = 

quation here.

Classes (C):

1 - dog
2 - cat
3 - bird

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \end{bmatrix}$$

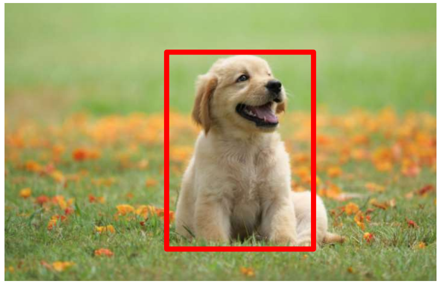
$$y = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

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

"Don't care."

Object localization

- Loss function

x = 

Classes (C):

1 - dog

2 - cat

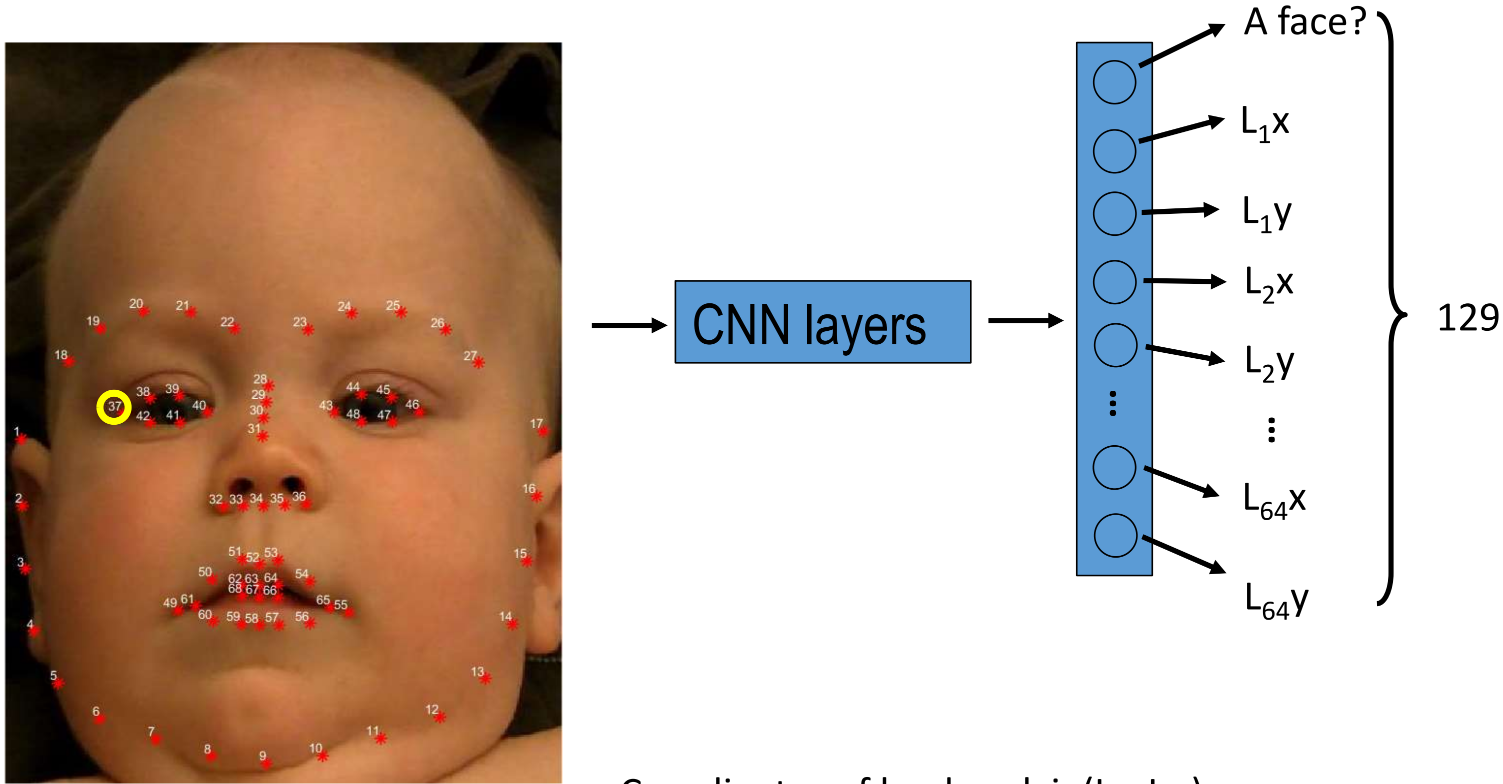
3 - bird

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \end{bmatrix}$$

$$L(\hat{y}, y) = \begin{cases} (\widehat{y_1} - y_1)^2 + (\widehat{y_2} - y_2)^2 \\ + \dots + (\widehat{y_8} - y_8)^2, \quad if\ y_1 = 1 \\ (\widehat{y_1} - y_1)^2, \quad if\ y_1 = 0 \end{cases}$$

Landmark detection

- Facial landmarks



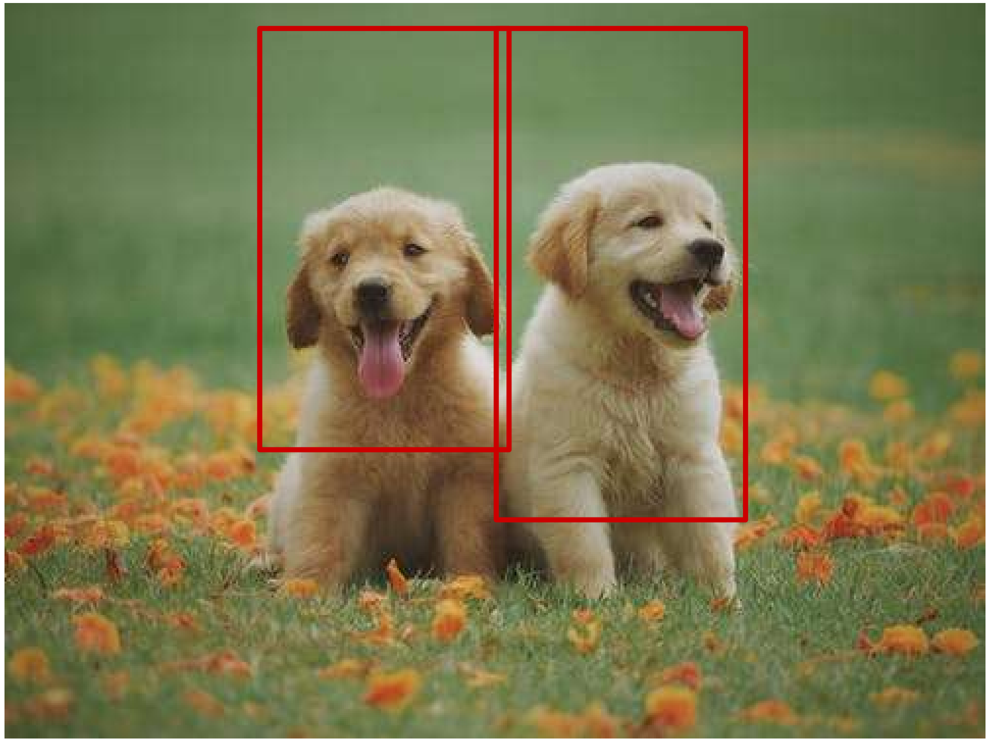
Coordinates of landmark i: (L_ix, L_iy)

Landmark detection

- Pose estimation

Object detection

- Dog detection example



Object detection

- Training set :

x　　　　　　　　　y

1

1

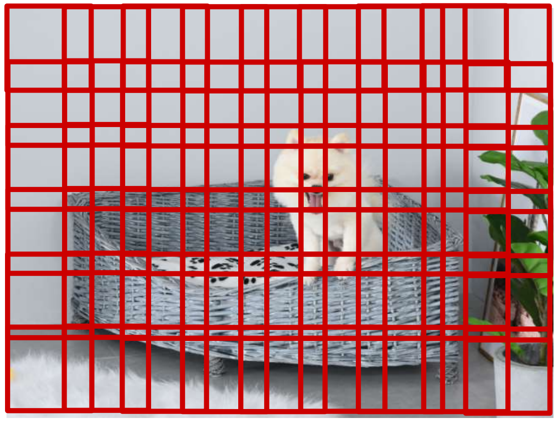
0

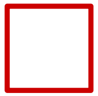
1

⋮　　　　　　　　　⋮

Object detection

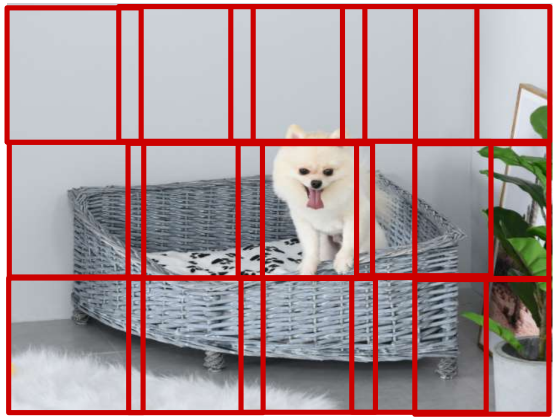
- Sliding window detection

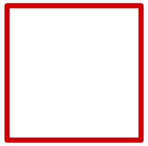




Object detection

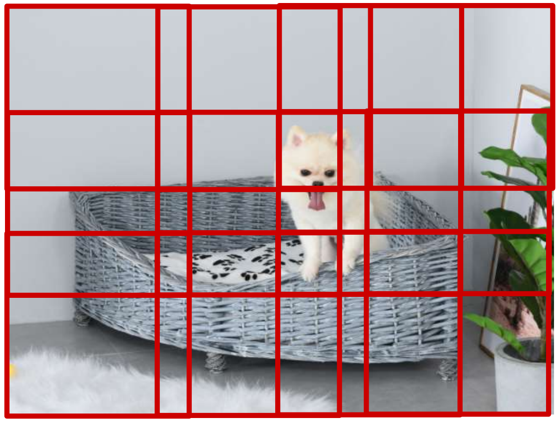
- Sliding window detection

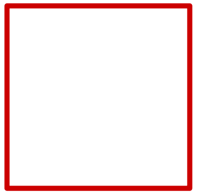




Object detection

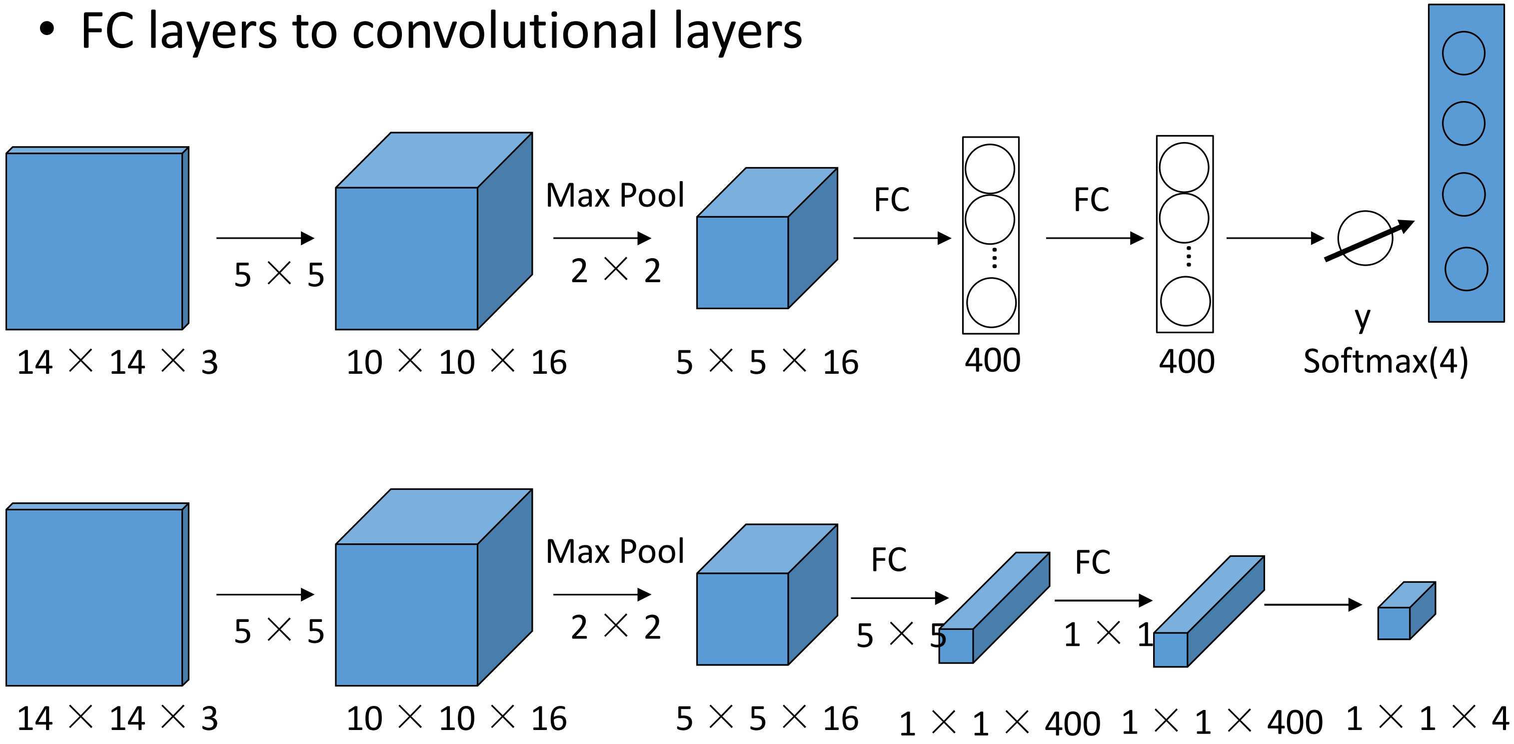
- Sliding window detection





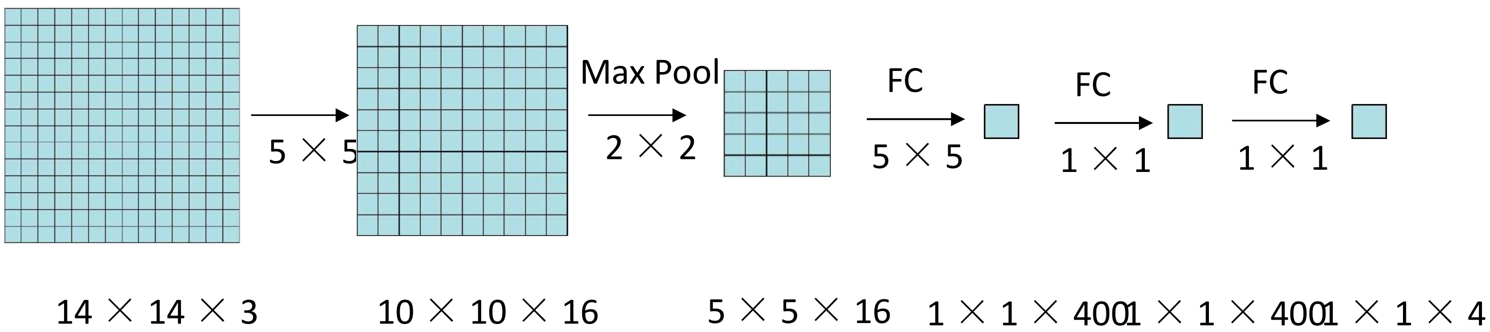
Object detection

- Convolutional implementation
  - FC layers to convolutional layers



Object detection

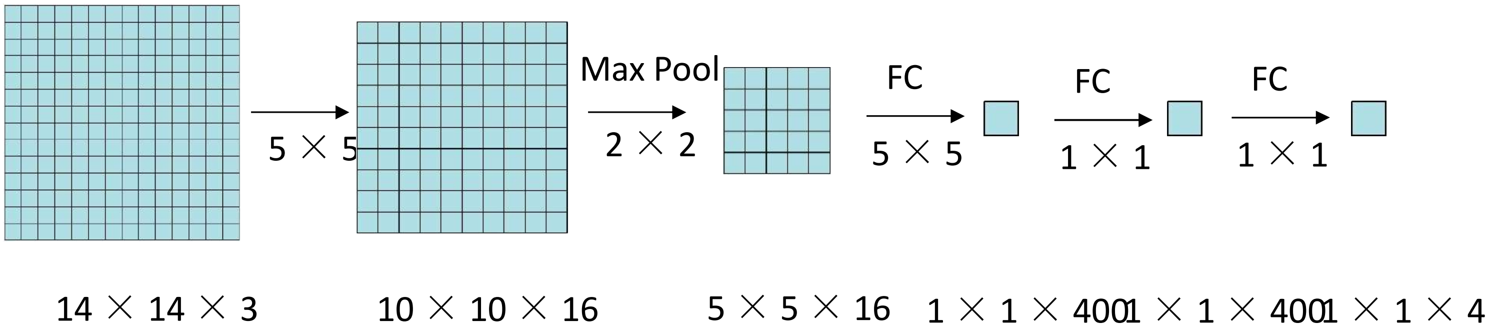
- Convolutional implementation of sliding window

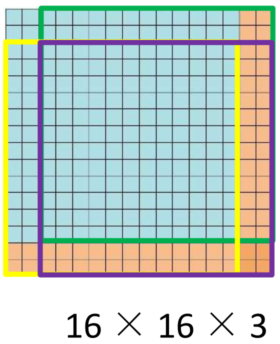


Sermanet, Pierre, et al. "Overfeat: Integrated recognition, localization and detection using convolutional networks." (2013).

Object detection

- Convolutional implementation of sliding window

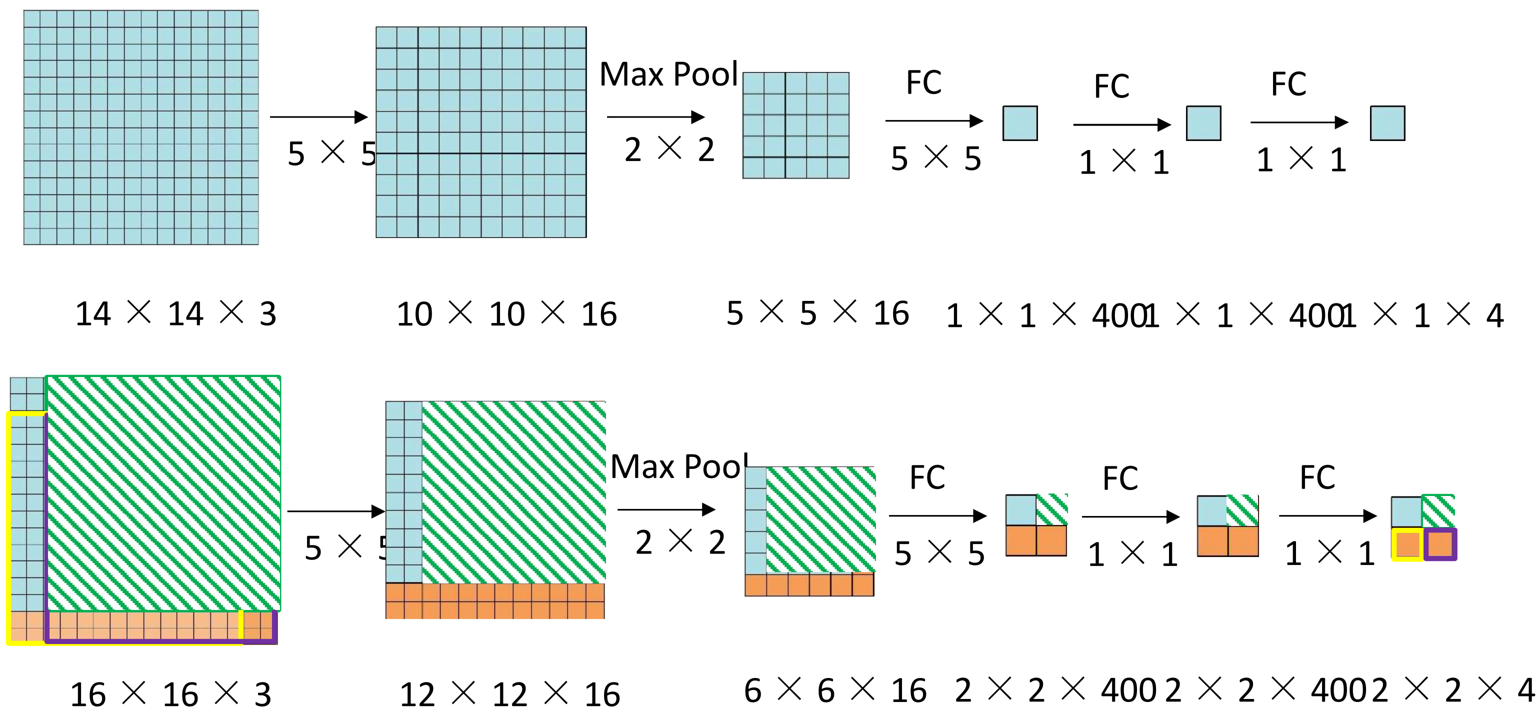




Sermanet, Pierre, et al. "Overfeat: Integrated recognition, localization and detection using convolutional networks." (2013).

Object detection

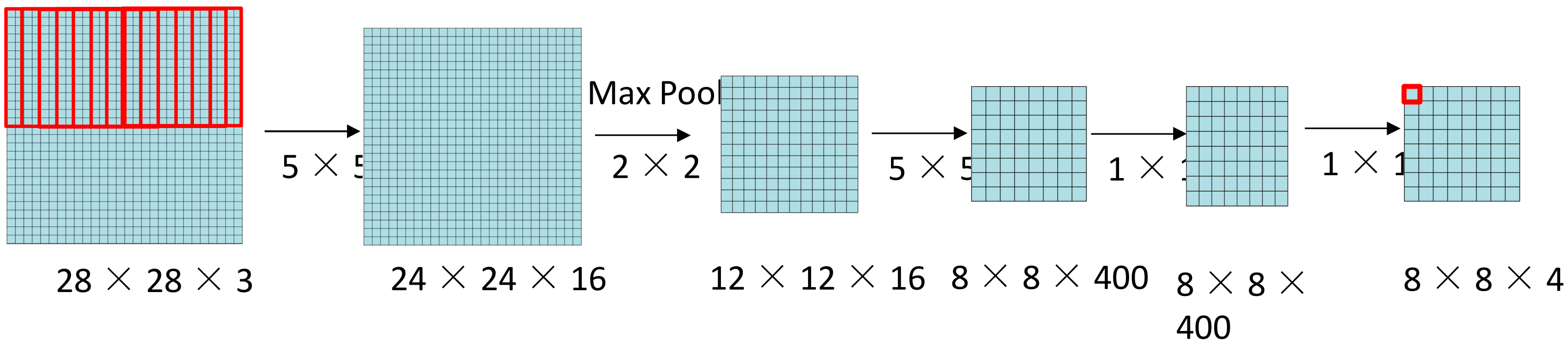
- Convolutional implementation of sliding window



Sermanet, Pierre, et al. "Overfeat: Integrated recognition, localization and detection using convolutional networks." (2013).

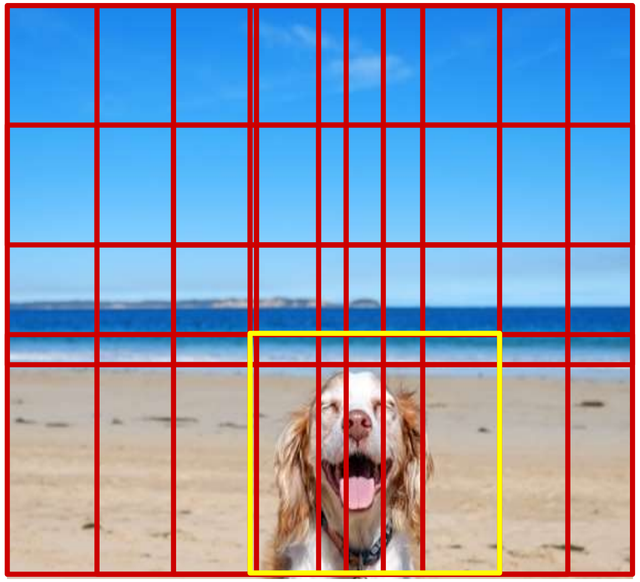
Object detection

- Convolutional implementation of sliding window



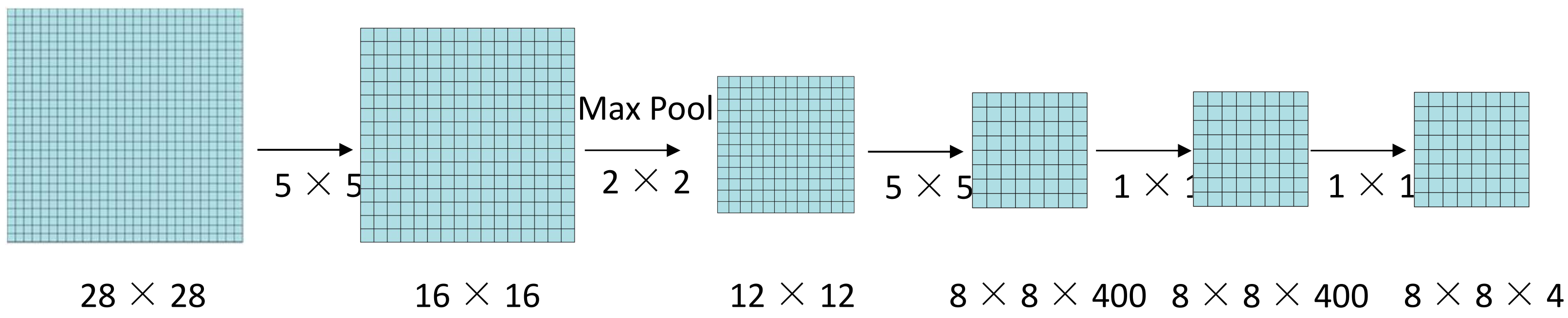
Object detection

- Convolutional implement of sliding window



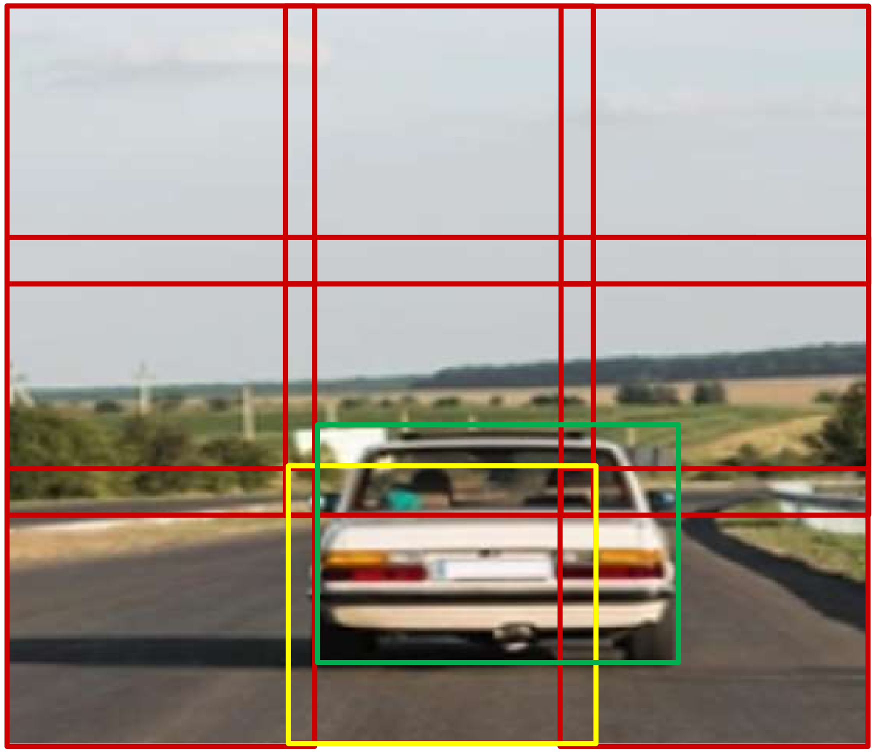
Object detection

- Convolutional implementation of sliding window



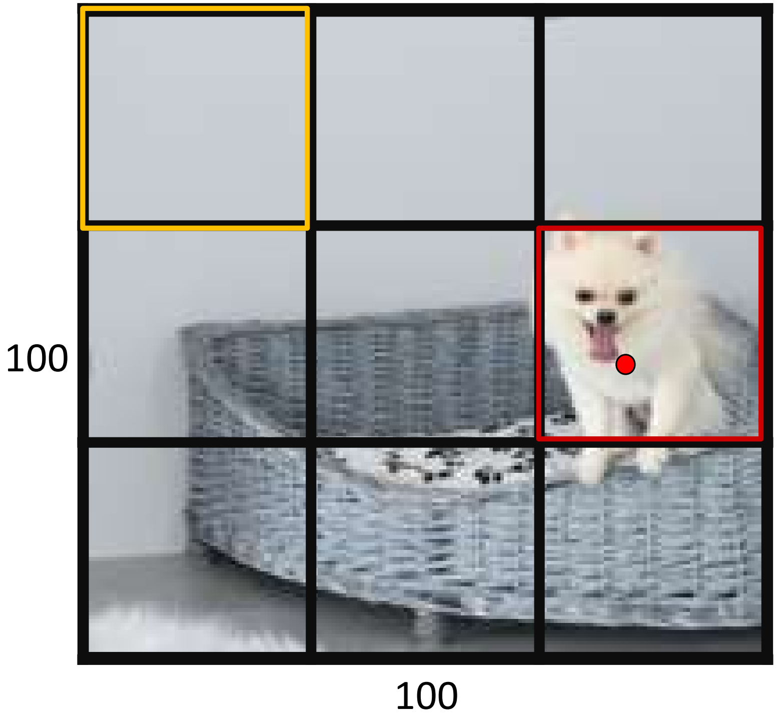
Bounding box prediction

- Accurate bounding boxes



YOLO

- You Only Look Once



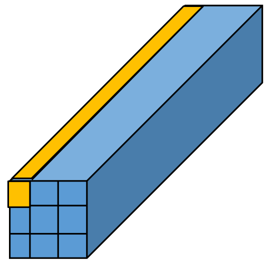
Labels for training (for each cell):

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} \qquad y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Classes (C):

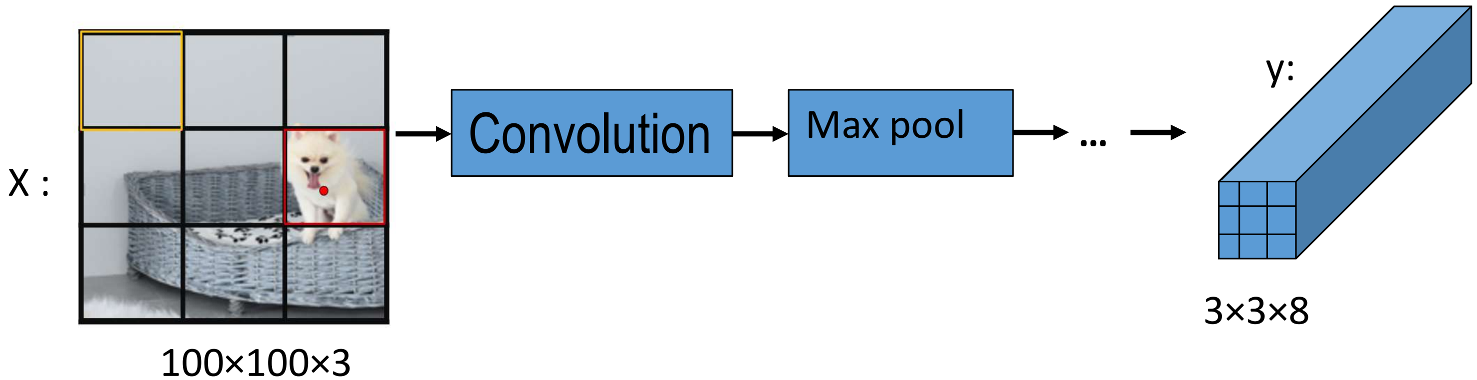
1 - dog
2 - cat
3 - bird

Target output: 3×3×8



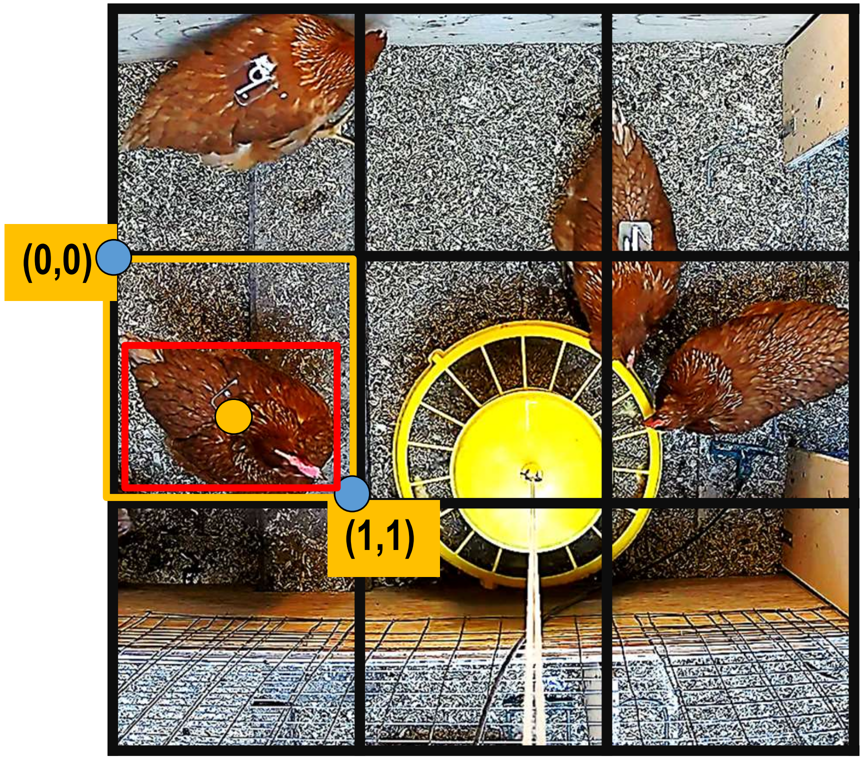
YOLO

- You Only Look Once



YOLO - Bounding box prediction

- Specify output labels



Labels for training (for each cell):

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} \rightarrow \boxed{y = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 0 \\ 1 \end{bmatrix}}$$

An object is present.

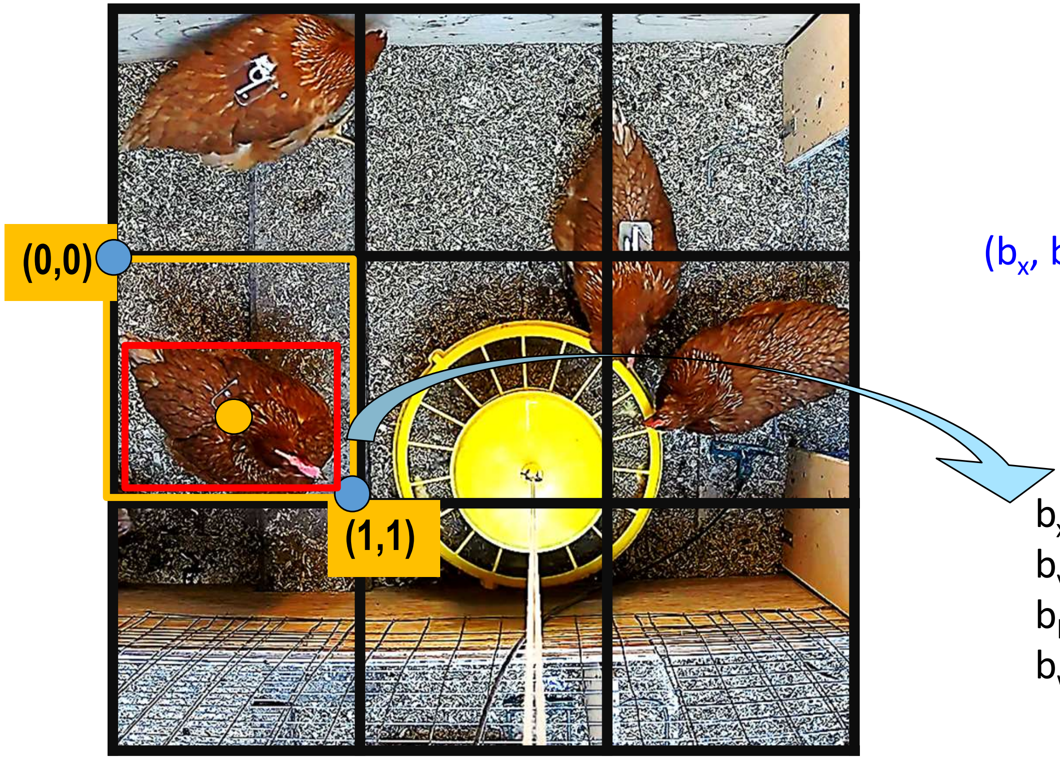
Bounding box information.

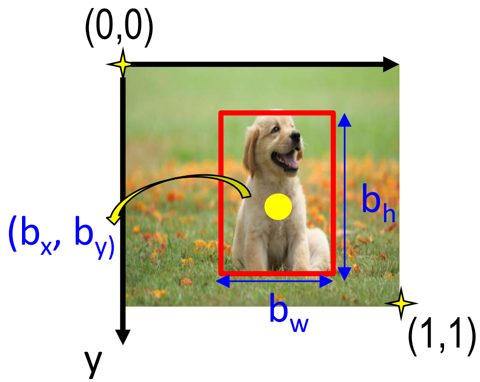
The object is of C_3 – chicken.

Classes (C):
C_1 - dog
C_2 - cat
C_3 - chicken

YOLO - Bounding box prediction

- Specify bounding boxes for objects



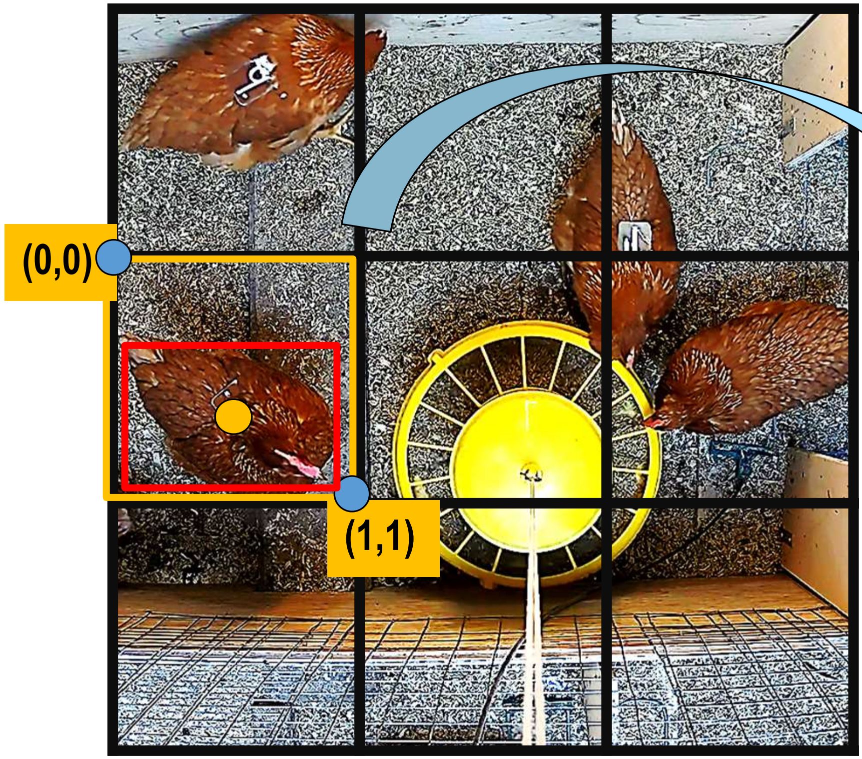


$b_x = 0.5$
$b_y = 0.7$
$b_h = 0.6$
$b_w = 0.9$

b_x, b_y, b_h, and b_w are specified relative to the grid cell.

YOLO - Bounding box prediction

- Output labels



$$y = \begin{bmatrix}1\\0.5\\0.7\\0.6\\0.9\\0\\0\\1\end{bmatrix}$$

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

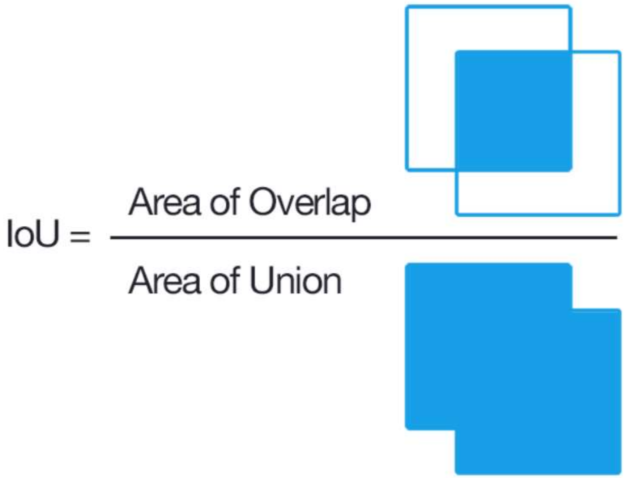
b_x, b_y: between 0 and 1

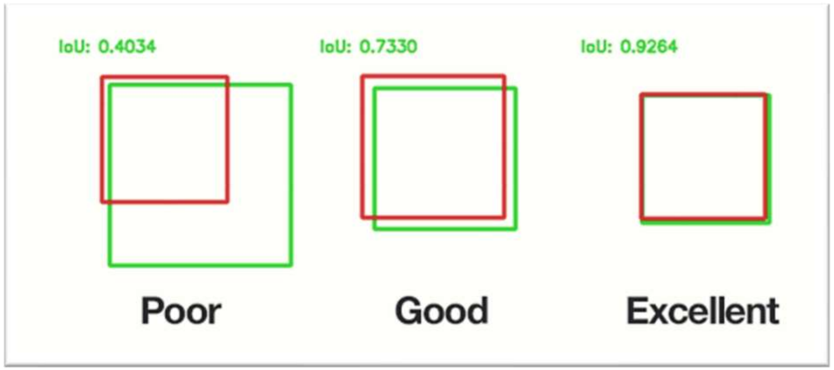
b_h, b_w: could be greater than 1

Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

Intersection over Union (IoU)

- Evaluation of object localization
  - IoU is a measure of the overlap between two bounding boxes

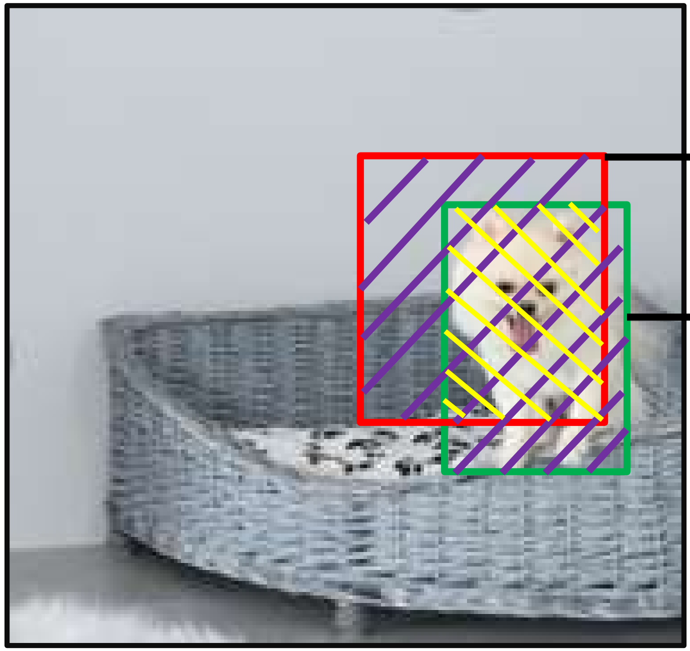




Examples of computing Intersection over Unions for various bounding boxes

Intersection over Union (IoU)

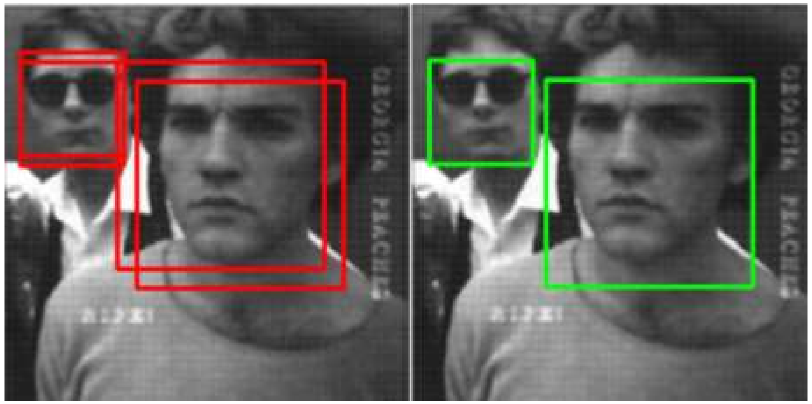
- Evaluation of object localization



- Object detection is correct, if IoU ≥ 0.5.
- To be more stringent, the threshold can be a number greater than 0.5.

YOLO - Non-max suppression

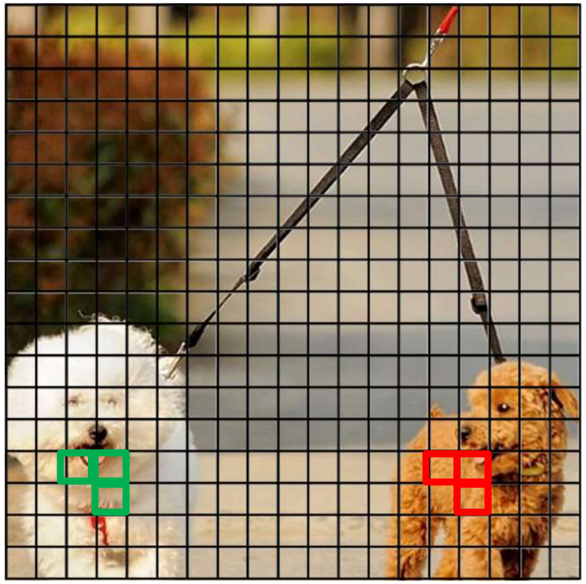
- Object detection algorithm may find multiple detections of the same object.
- Non-max suppression helps to ensure each object is only detected once.



Initial bounding boxes   After non-max suppression

YOLO - Non-max suppression

- Object detection algorithm may find multiple detections of the same object.

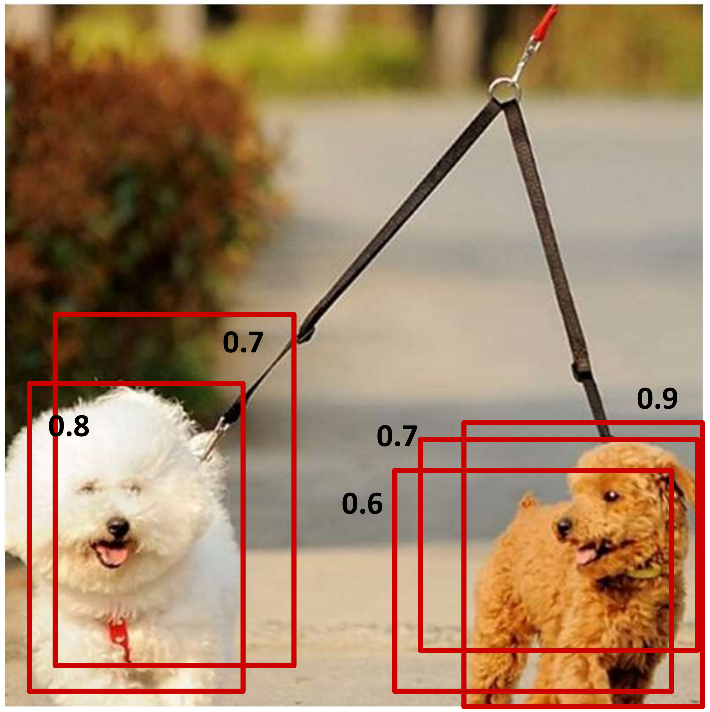


19×
19

- Technically, only one of those grid cells should predict that there is an object.
- However, in practice, an adjacent cell may also think the center of an object is inside itself.

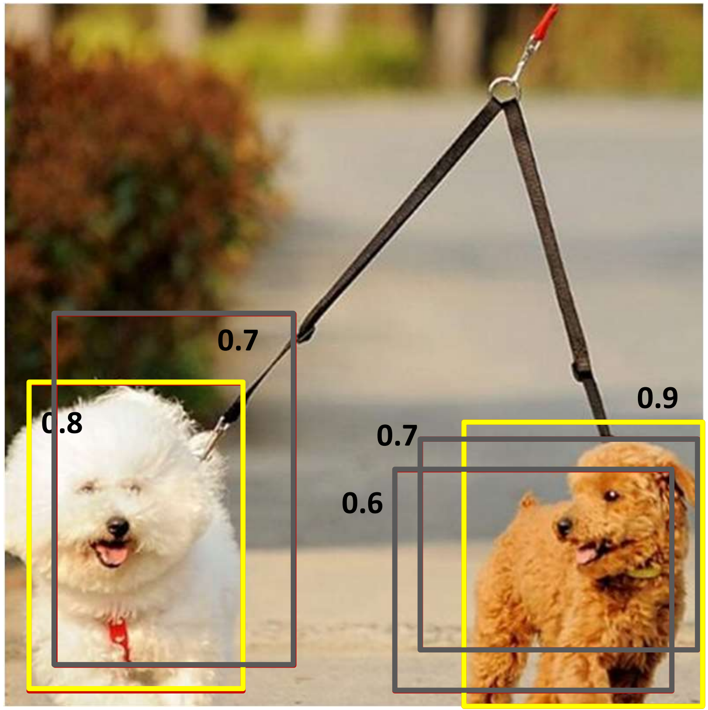
YOLO - Non-max suppression

- Multiple detections per object



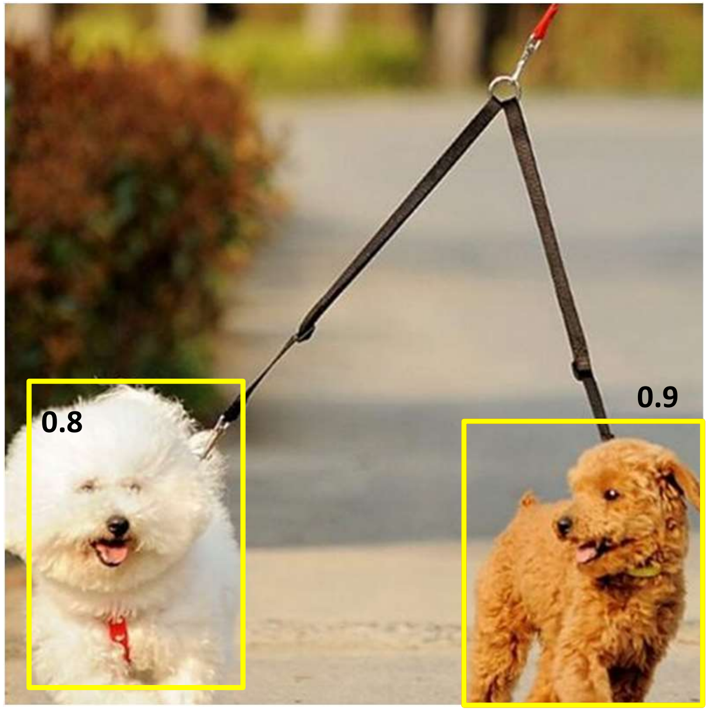
YOLO - Non-max suppression

- Non-max suppression example



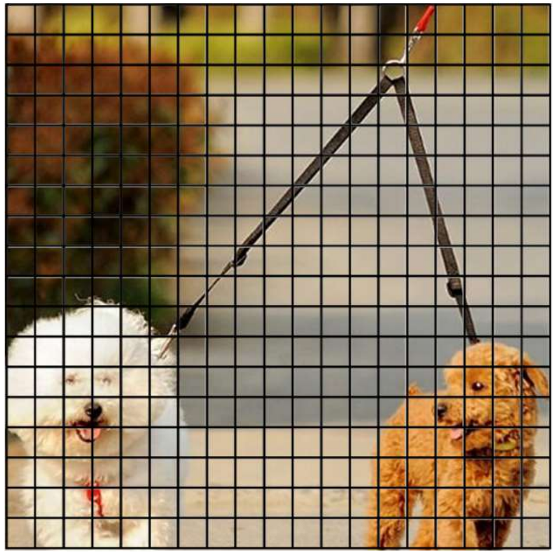
YOLO - Non-max suppression

- After non-max suppression



YOLO - Non-max suppression

- Non-max suppression algorithm



19×19

For each of the 19×19 positions, the output prediction is:

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

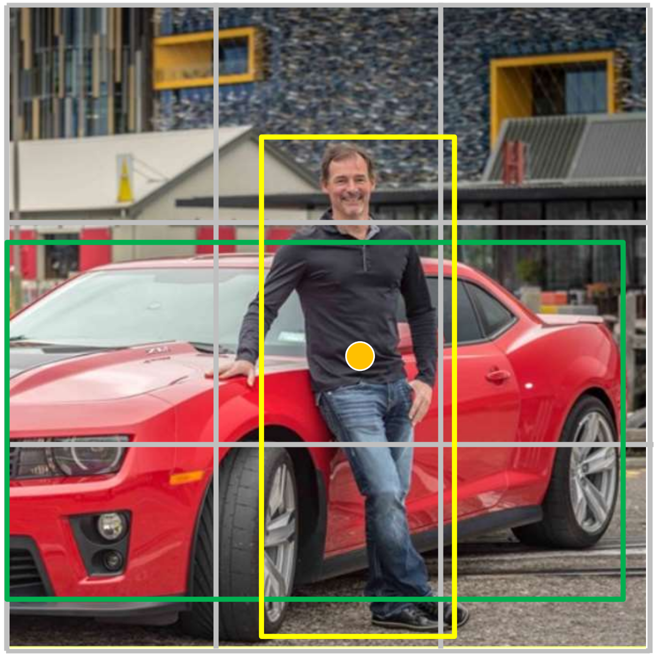
Discard all boxes with $p_c \leq Threshold$ (e.g. 0.6)

While there are any remaining boxes:

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

Anchor boxes

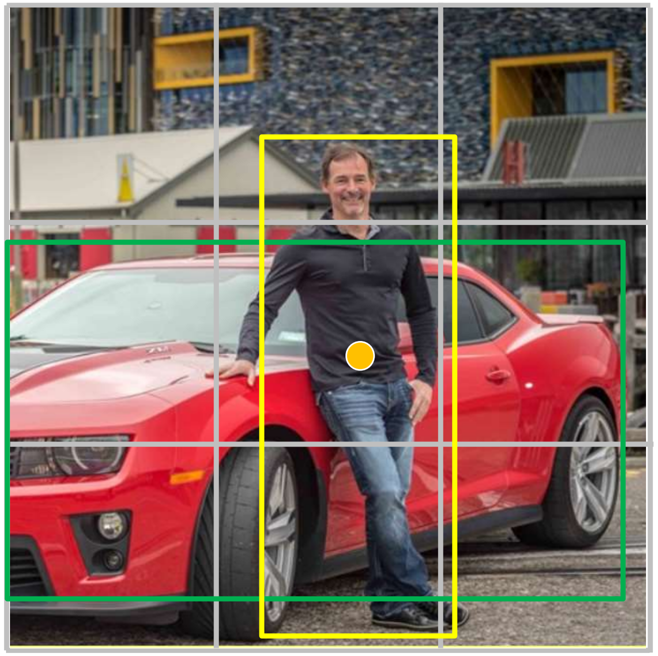
- Overlapping objects
  - One grid cell has the midpoints of two objects



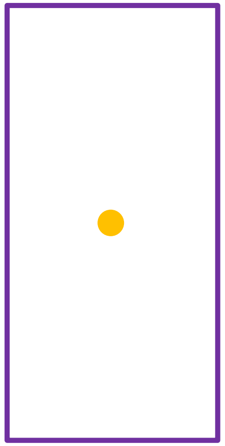
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \end{bmatrix}$$ is not able to output multiple detections.

Anchor boxes

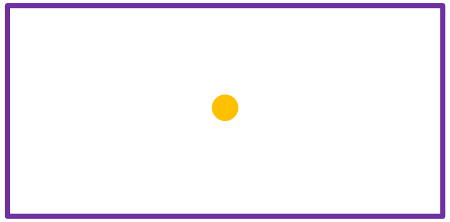
- Predefine different shapes – anchor boxes



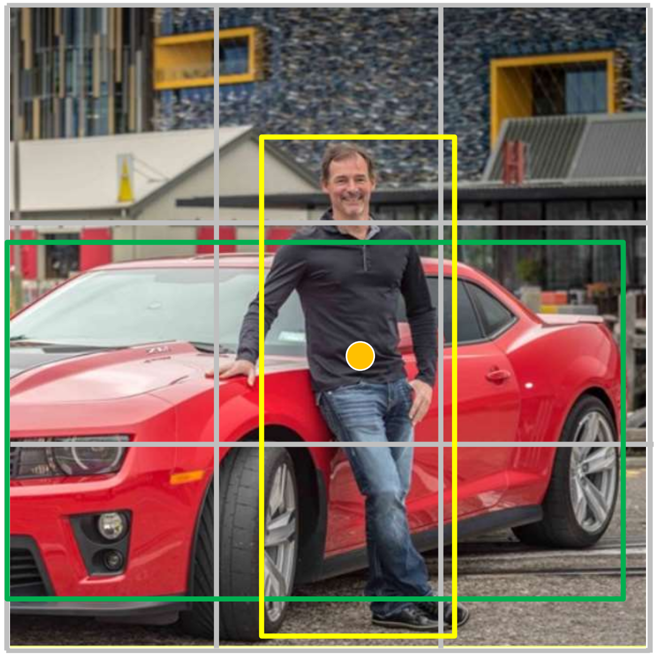
Anchor box 1:



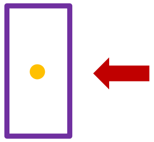
Anchor box 2:



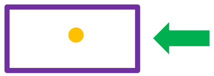
Anchor boxes



Anchor box 1:



Anchor box 2:



Output label:

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \\ P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} \qquad \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 1 \\ 0 \\ 0 \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Anchor box 1

Anchor box 2

Classes (C):

C_1 - person

C_2 - car

C_3 - bike

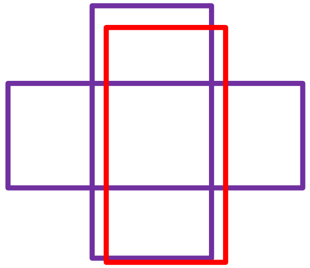
Anchor boxes

- Anchor box algorithm

Previously:
Each object in training image is assigned to a grid cell that contains that object's midpoint

With two anchor boxes:
Each object in training image is assigned to grid cell that contains object's midpoint and an anchor box for the gird cell with highest IoU

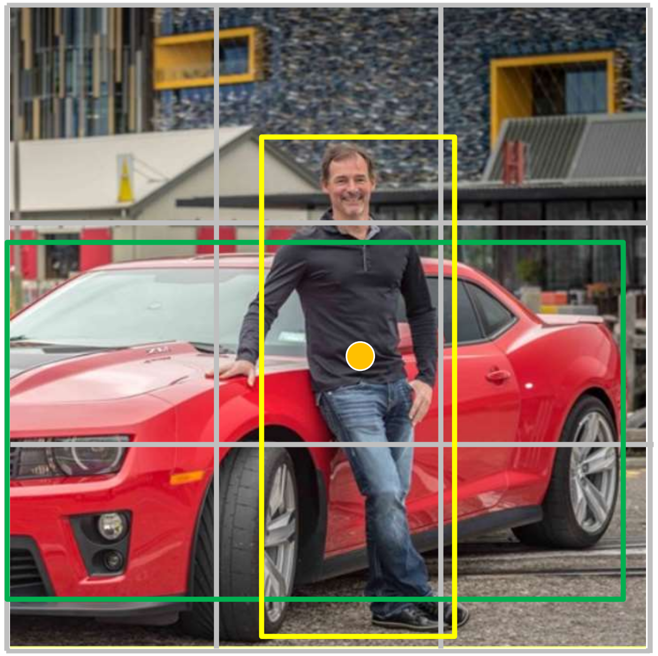
Output y: $3 \times 3 \times 8$



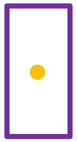
Output y: $3 \times 3 \times 16$

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

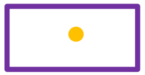
Anchor boxes



Anchor box 1:



Anchor box 2:



Output label of the center cell:

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \\ P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} \qquad \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 1 \\ 0 \\ 0 \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

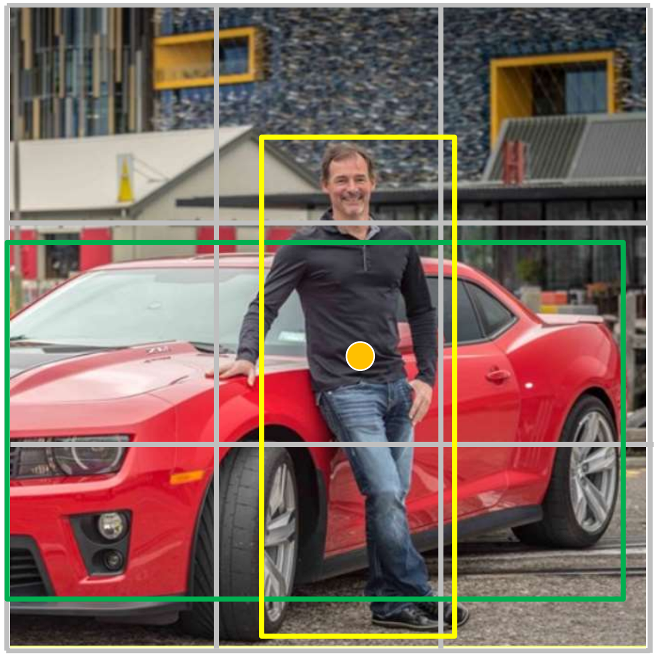
Classes (C):

C_1 - person

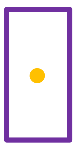
C_2 - car

C_3 - bike

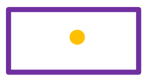
Anchor boxes



Anchor box 1:



Anchor box 2:



Output label of a grid cell only having part of a car:

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \\ P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Classes (C):

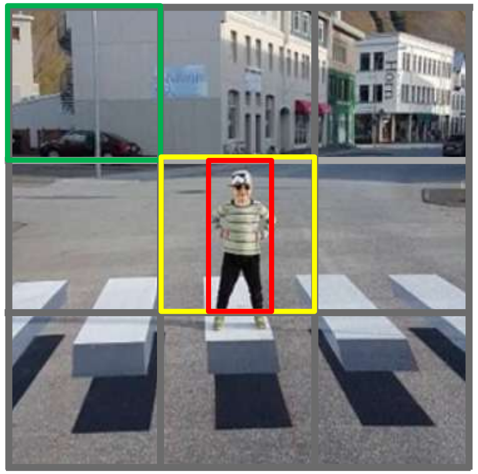
C_1 - person

C_2 - car

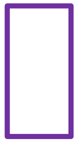
C_3 - bike

YOLO algorithm

- Construct a training set



Anchor box 1:



Anchor box 2:

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \\ P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \qquad \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 1 \\ 0 \\ 0 \\ 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

Anchor boxes

Size of output y is $3 \times 3 \times 2 \times 8$ or $3 \times 3 \times 16$

Classes (C):

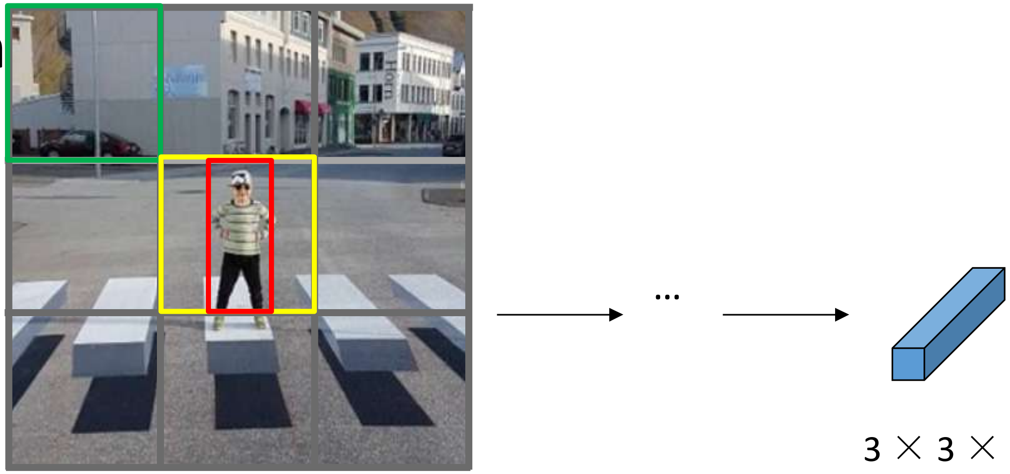
C_1 - person

C_2 - car

C_3 - bike

YOLO algorithm

- Makin



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

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \\ P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} \quad \begin{bmatrix} 0 \\ .. \\ .. \\ .. \\ .. \\ .. \\ .. \\ .. \\ 0 \\ .. \\ .. \\ .. \\ .. \\ .. \\ .. \\ .. \end{bmatrix} \quad \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 1 \\ 0 \\ 0 \\ 0 \\ .. \\ .. \\ .. \\ .. \\ .. \\ .. \\ .. \end{bmatrix}$$

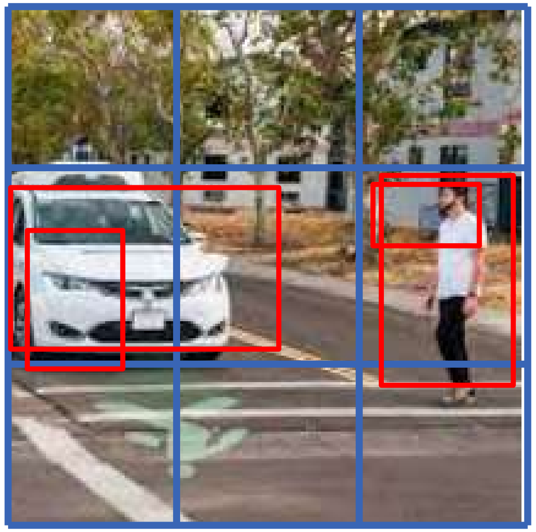
YOLO algorithm

- Outputting the non-max suppressed outputs

- For each grid cell, get 2 predicted bounding boxes

YOLO algorithm

- Outputting the non-max suppressed outputs



- For each grid cell, get 2 predicted bounding boxes
- Get rid of low probability predictions

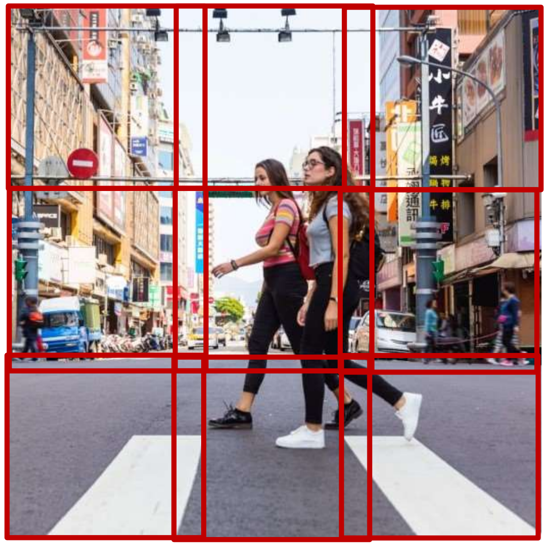
YOLO algorithm

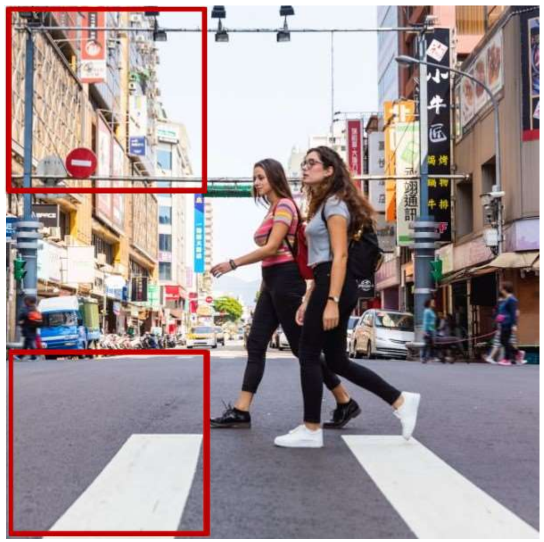
- Outputting the non-max suppressed outputs

- For each grid cell, get 2 predicted bounding boxes
- Get rid of low probability predictions
- For each class (person, car, bike) use non-max suppression to generate final perditions

Region proposal

- Region proposal : R-CNN

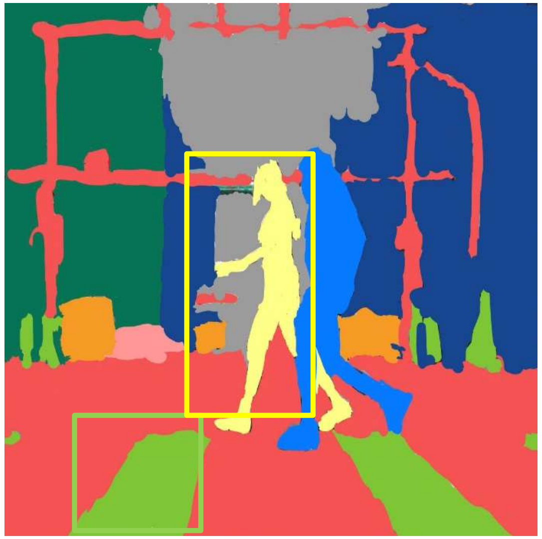




Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

Region proposal

- Region proposal : R-CNN



Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

Region proposal

- Faster algorithm

Propose regions. Classify proposed regions one at a time. Output label + bounding box

R-CNN:

Propose regions. Use convolution implementation of sliding windows to classify all the proposed regions

Fast R-CNN:

Use convolution network to propose regions

Faster R-CNN:

Yolo

- YOLO solves the object detection as a regression problem to obtain the positions of all the objects in the image , their categories and the corresponding confidence probabilities.
- YOLO uses the convolutional layer to extract image features, the fully connected layer predicts the image position and category probability value. YOLO divides the input image into S * S grids. If the coordinates of the central position of an object fall into a grid, the grid is responsible for detecting the object. Each grid outputs a plurality of bounding box information (including the central position coordinates of the predicted object bounding box, the width and height of the bounding box, the accuracy of the object and the object position) and the probability information that the multiple objects belong to a certain category. YOLO uses sum of mean squares error as a loss function .

- The YOLO training can be divided into two steps: pre-training and using the top 20 convolutional layer network parameters obtained from the pre-training to initialize the network parameters of the top 20 convolutional layers of the YOLO model, and then annotate the data for model training.

Conclusions

- Object localization
- Object detection
  - Sliding window detection
  - Convolutional implementation
- YOLO
  - Bounding box predictions
  - Non-max suppression
  - Anchor boxes
- Region proposal for object detection

Training your own yolo

- https://blog.paperspace.com/train-yolov5-custom-data/