Week 6 – Object Detection

ELEC0144 Machine Learning for Robotics

Dr. Chow Yin Lai

Email: uceecyl@ucl.ac.uk

Schedule

Week	Lecture	Workshop	Assignment Deadlines
1	Introduction; Image Processing	Image Processing	
2	Camera and Robot Calibration	Camera and Robot Calibration	
3	Introduction to Neural Networks	Camera and Robot Calibration	Friday: Camera and Robot Calibration
4	MLP and Backpropagation	MLP and Backpropagation	
5	CNN and Image Classification	MLP and Backpropagation	
6	Object Detection	MLP and Backpropagation	Friday: MLP and Backpropagation
7	Path Planning	Path Planning	
8	Kalman Filter SLAM	Path Planning	
9	Extended Kalman Filter SLAM	Path Planning	
10	Particle Filter SLAM	Path Planning	Friday: Path Planning

Content

- Introduction to Object Detection
- Detecting Single Object
- Detecting Multiple Objects
  - Sliding Window
  - Region Proposal Methods
  - YOLO Algorithm

Content

- Introduction to Object Detection
- Detecting Single Object
- Detecting Multiple Objects
  - Sliding Window
  - Region Proposal Methods
  - YOLO Algorithm

Introduction to Object Detection (1)

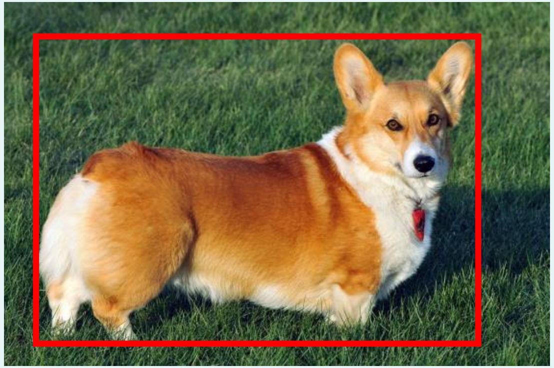
- So far, we have looked at image classification i.e. classifying an object from image.

This Photo by Unknown Author is licensed under CC BY

P(Cat) = 0.3
P(Dog) = 0.9
P(Car) = 0.1
P(Truck) = 0.1

Introduction to Object Detection (2)

- If we now additionally need to know the location of object in the image, it is called object detection.

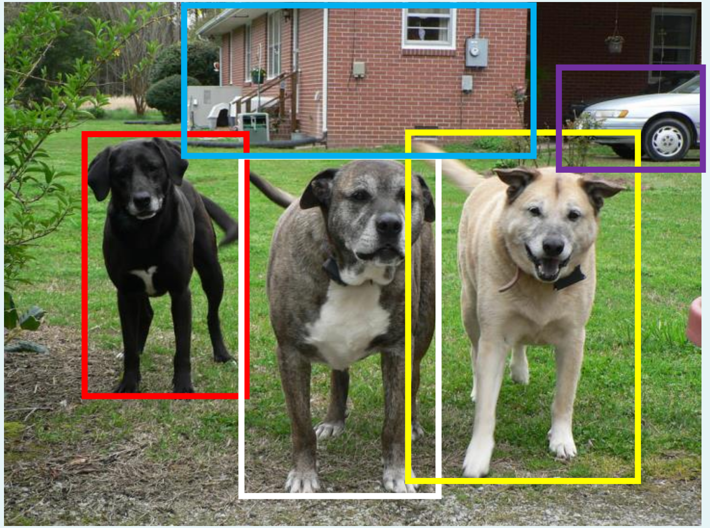


This Photo by Unknown Author is licensed under CC BY

P(Cat) = 0.3
P(Dog) = 0.9
P(Car) = 0.1
P(Truck) = 0.1
+
Location of object

Introduction to Object Detection (3)

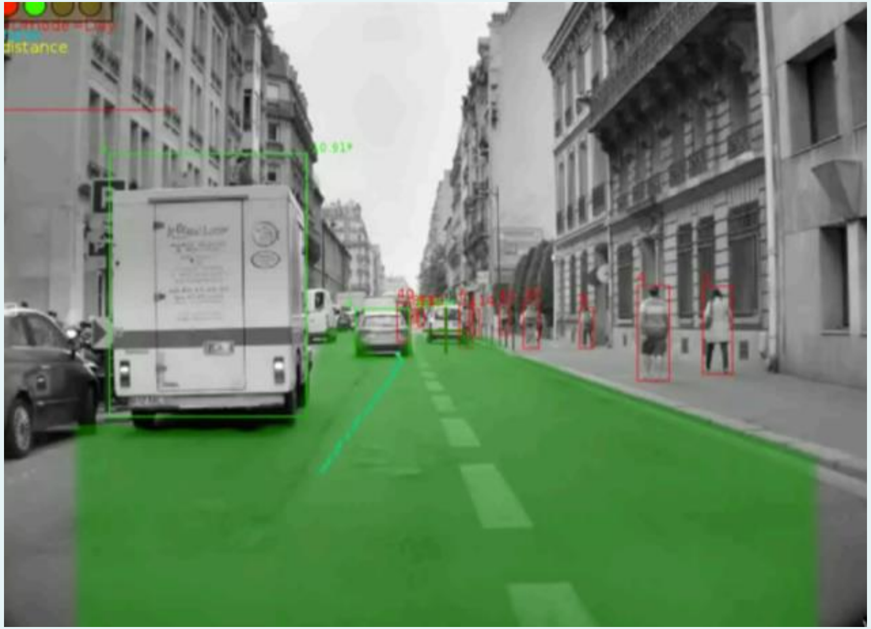
- This includes the cases where we have multiple objects, potentially from different classes.



This Photo by Unknown Author is licensed under CC BY

Introduction to Object Detection (4)

- Object detection is an important task in computer vision.
- E.g. self-driving cars need to detect other cars, pedestrians, traffic signs etc.



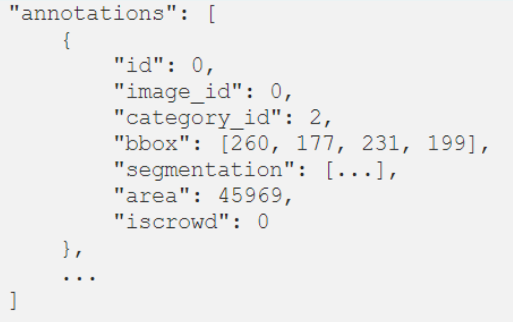
This Photo by Unknown Author is licensed under CC BY-SA-NC

Dataset for Object Detection (1)

- MNIST, CIFAR and ImageNet were used for image classification, but there were no bounding boxes given.
- You may take those images and manually (and painstakingly) draw bounding boxes and parameterize these.
- Luckily, there are other datasets which provide information of classes as well as bounding boxes.

Dataset for Object Detection (2)

- A good dataset for this purpose is the COCO (Common Objects in Context) dataset.
  - 80 classes.



```
"annotations": [
    {
        "id": 0,
        "image_id": 0,
        "category_id": 2,
        "bbox": [260, 177, 231, 199],
        "segmentation": [...],
        "area": 45969,
        "iscrowd": 0
    },
    ...
]
```

Sample data, by Eric Hofesmann

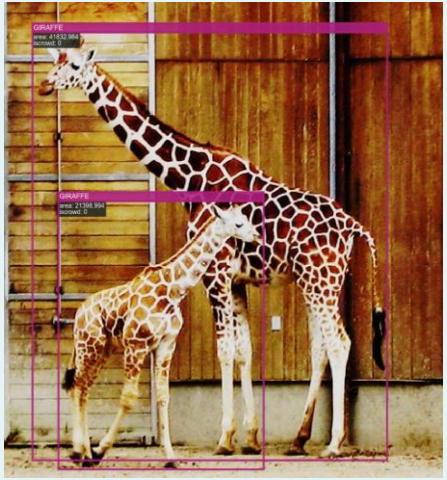


Image 001298.jpg from COCO, by Eric Hofesmann

Both images come from https://towardsdatascience.com/how-to-work-with-object-detection-datasets-in-coco-format-9bf4fb5848a4

Dataset for Object Detection (3)

- PASCAL VOC2007 (Visual Object Classes) is another famous dataset for object detection.
  - 20 classes.

http://lear.inrialpes.fr/RecogWorkshop08/documents/everingham.pdf

Evaluation of Object Detection (1)

- How we do evaluate the performance of the object detection algorithm?
- We need to understand a few terms.

Evaluation of Object Detection (2)

$$\boxed{\text{Precision} = \frac{\#\ \text{True positive prediction}}{\#\ \text{All positive prediction}}}$$

- This is regardless of the ground truth.
- E.g. ground truth = 120 persons. Predicted 100 persons. But out of the 100 predicted persons, only 80 are indeed persons at the correct location. Thus the precision is 80/100 = 80%.
- Not only do we care about the class (person), but also the location of the bounding box with respect to the ground truth.
- If the overlap between predicted bounding box and ground truth is small, it is considered incorrect as well – We will explain a measure of the overlap later.

Evaluation of Object Detection (3)

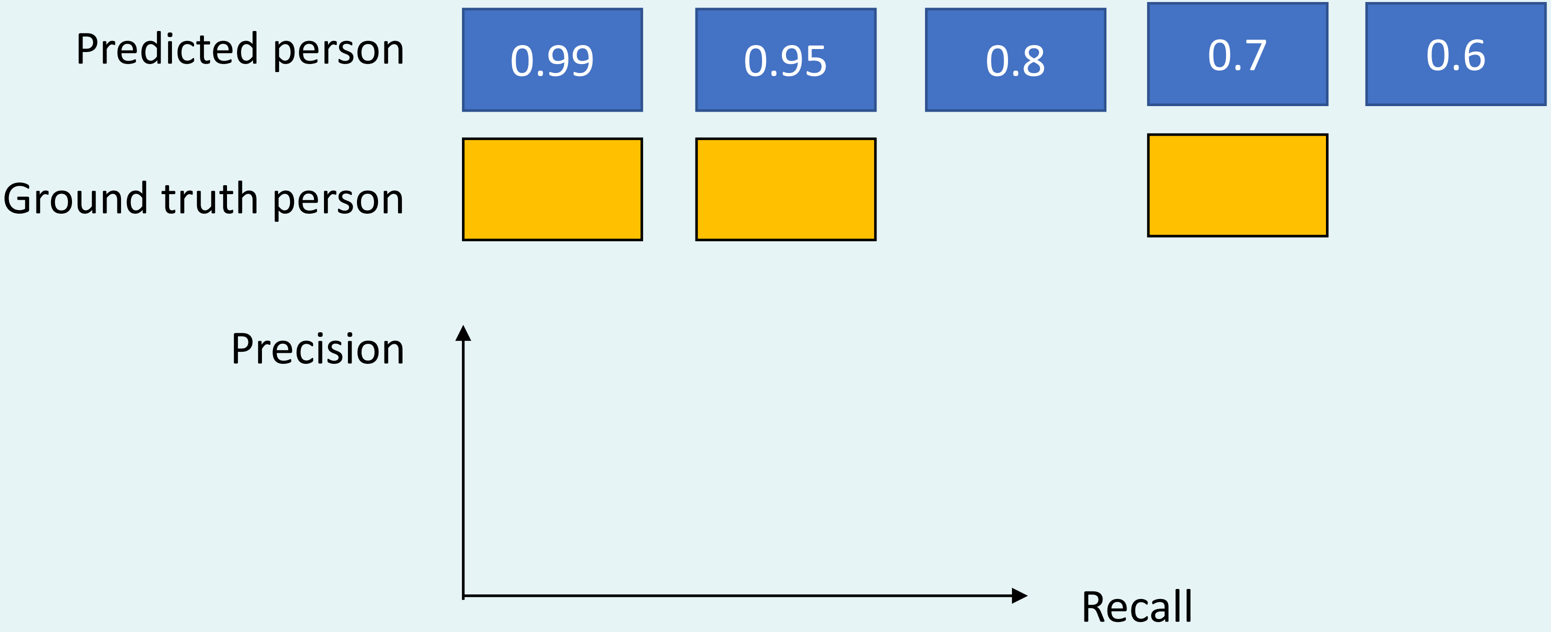
$$\boxed{\text{Recall} = \frac{\text{\# True positive prediction}}{\text{\# All ground truth positives}}}$$

- This takes into account the ground truth.
- E.g. ground truth = 120 persons. Predicted 100 persons. But out of the 100 predicted persons, only 80 are indeed persons at the correct location. Thus the recall is 80/120 = 66.67%.

Evaluation of Object Detection (4)

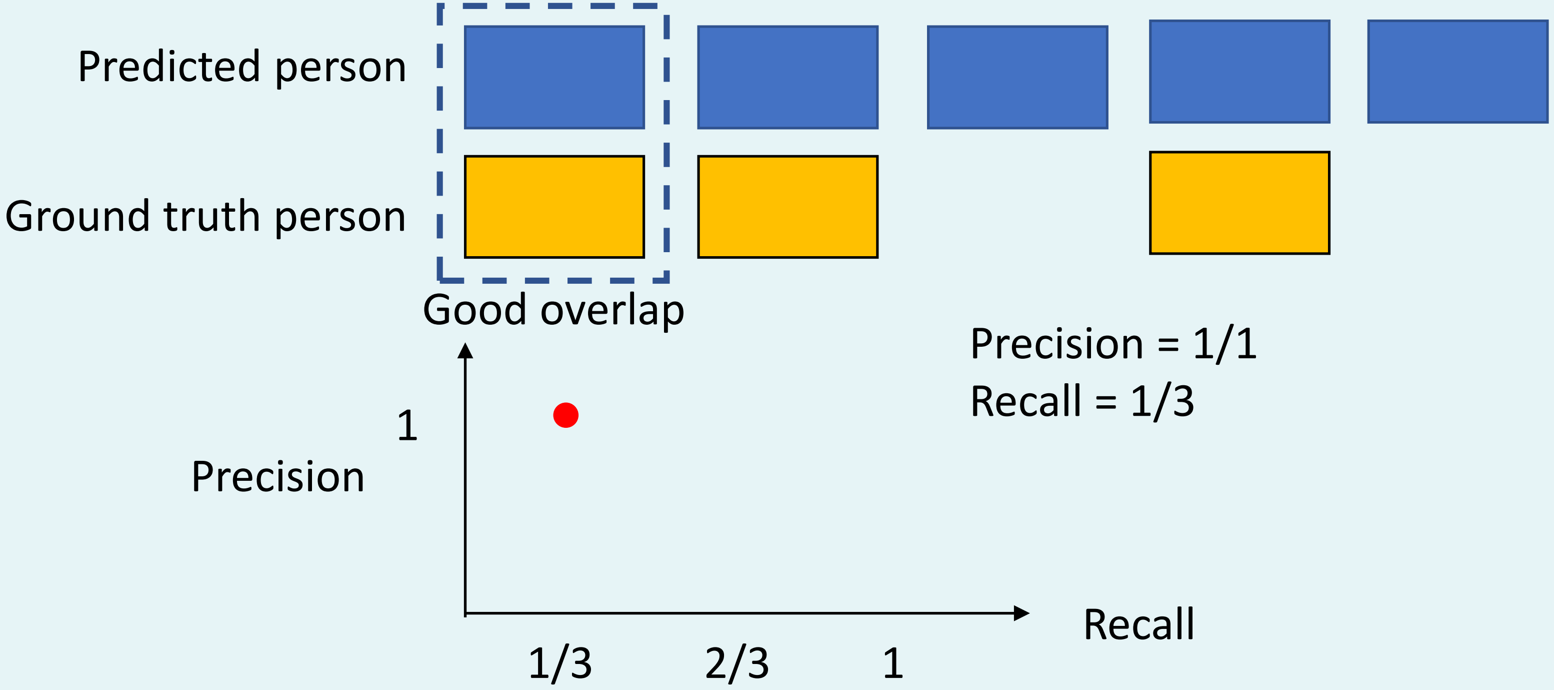
- Mean Average Precision:
  - The area under the precision-recall curve.

*Sorted by probability



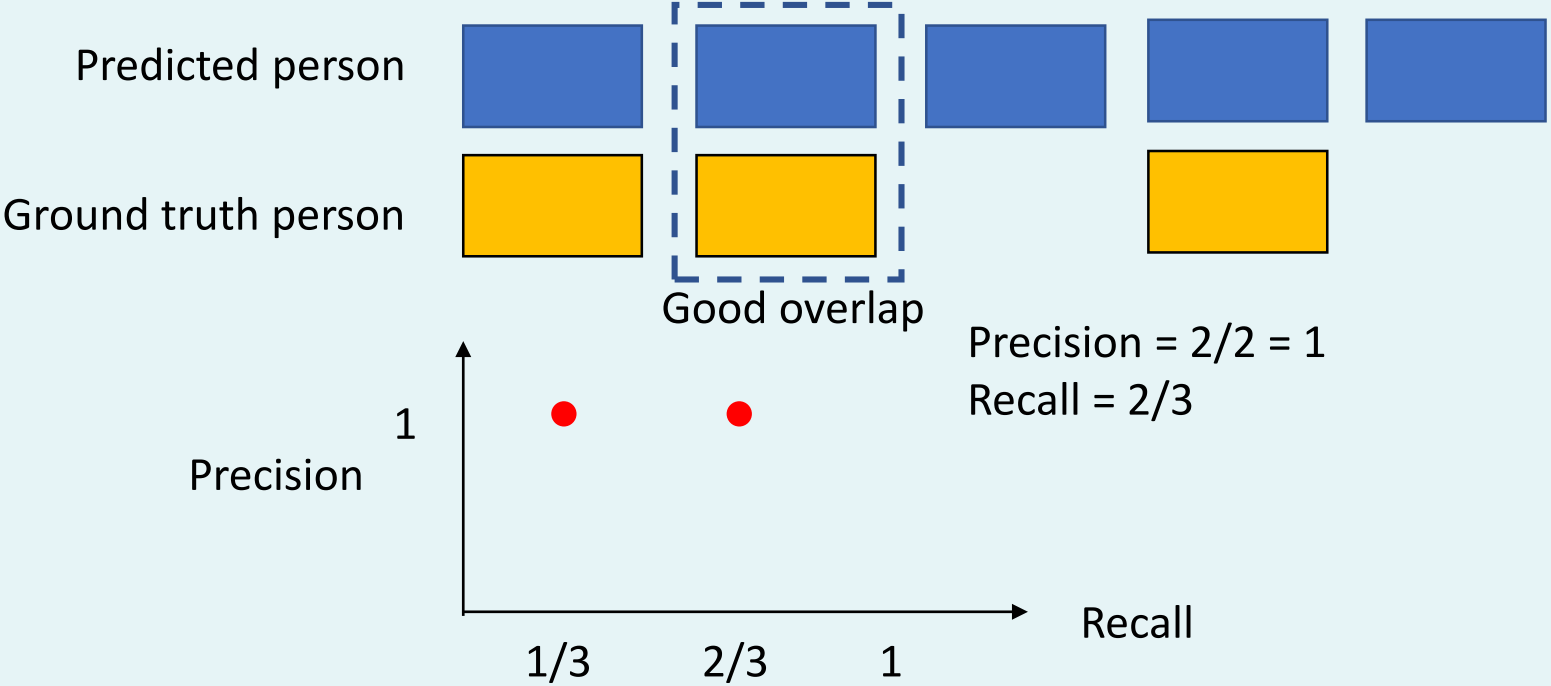
Evaluation of Object Detection (5)

- Mean Average Precision:
  - The area under the precision-recall curve.



Evaluation of Object Detection (6)

- Mean Average Precision:
  - The area under the precision-recall curve.

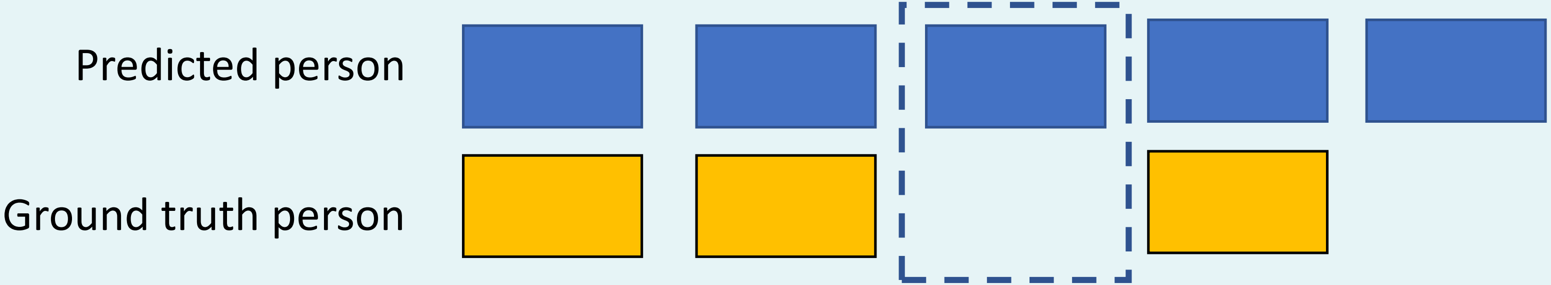


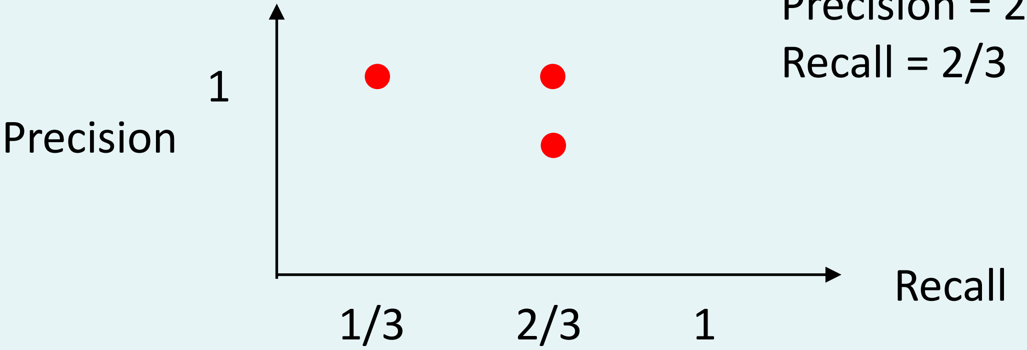
Precision = 2/2 = 1

Recall = 2/3

Evaluation of Object Detection (7)

- Mean Average Precision:
  - The area under the precision-recall curve.

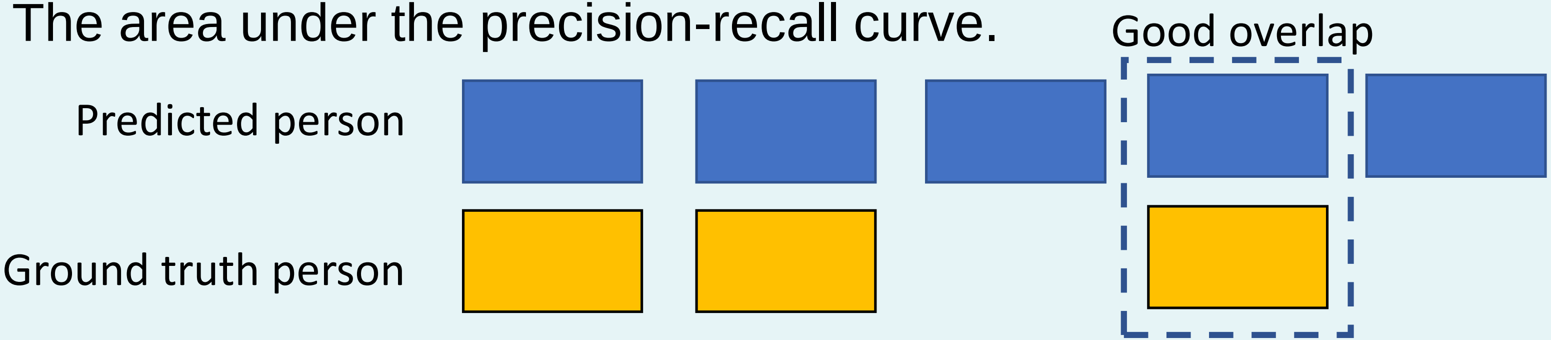


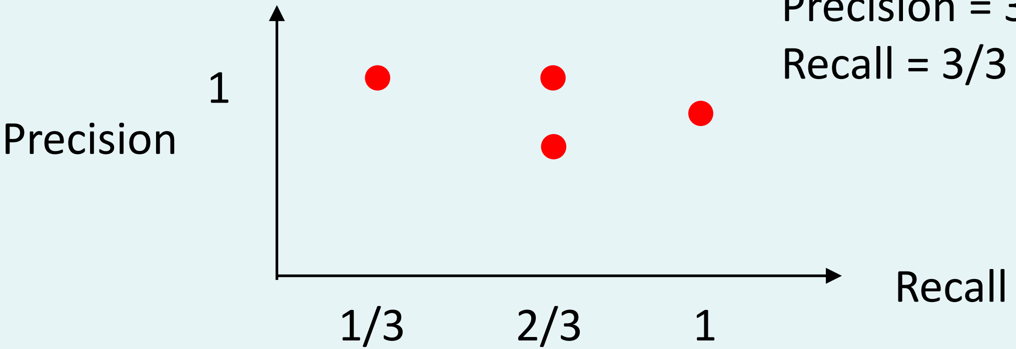


Precision = 2/3
Recall = 2/3

Evaluation of Object Detection (8)

- Mean Average Precision:
  - The area under the precision-recall curve.

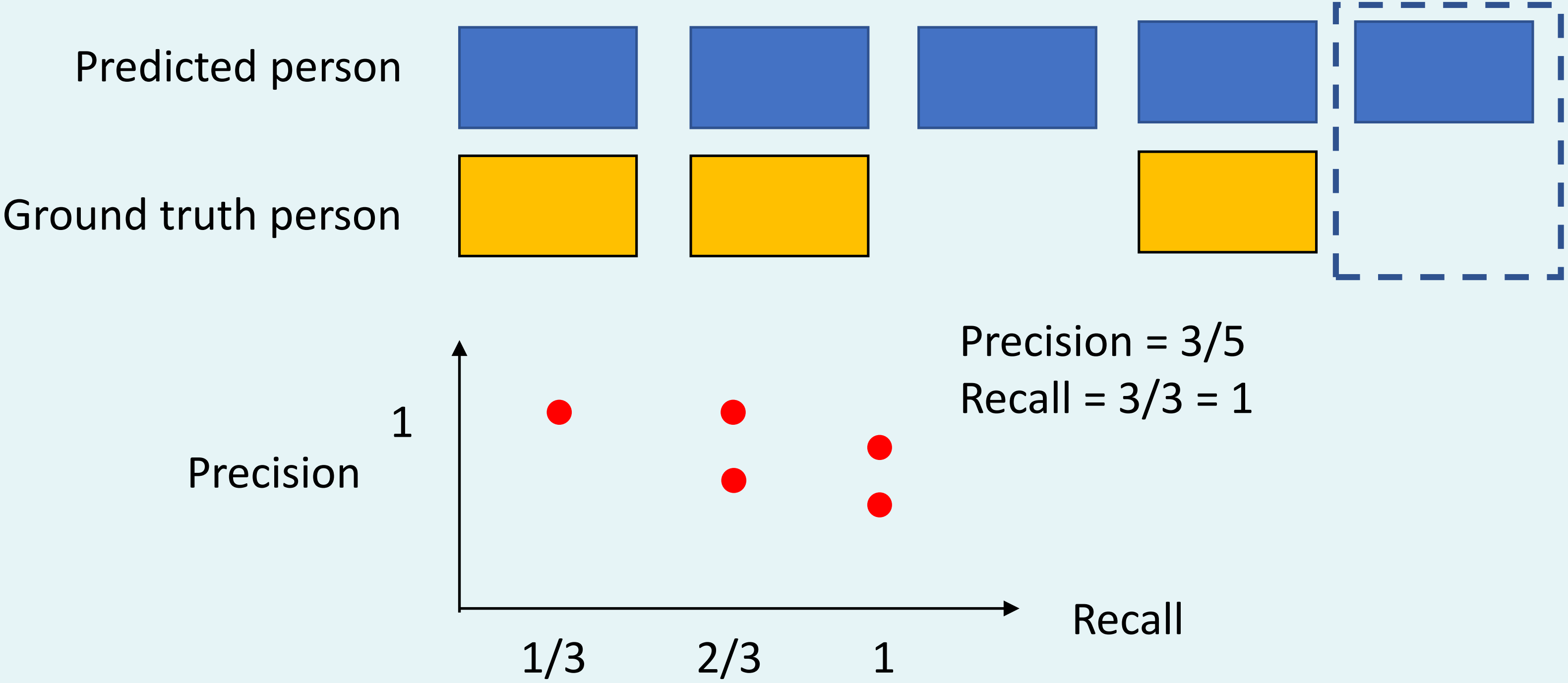




Precision = 3/4
Recall = 3/3 = 1

Evaluation of Object Detection (9)

- Mean Average Precision:
  - The area under the precision-recall curve.

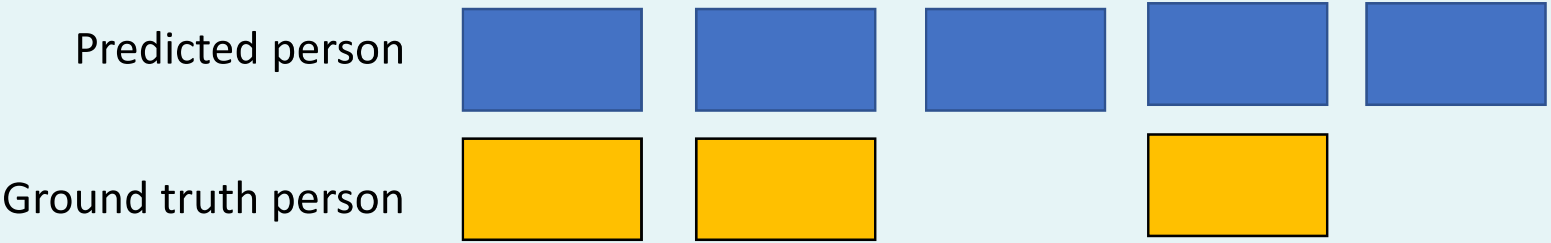


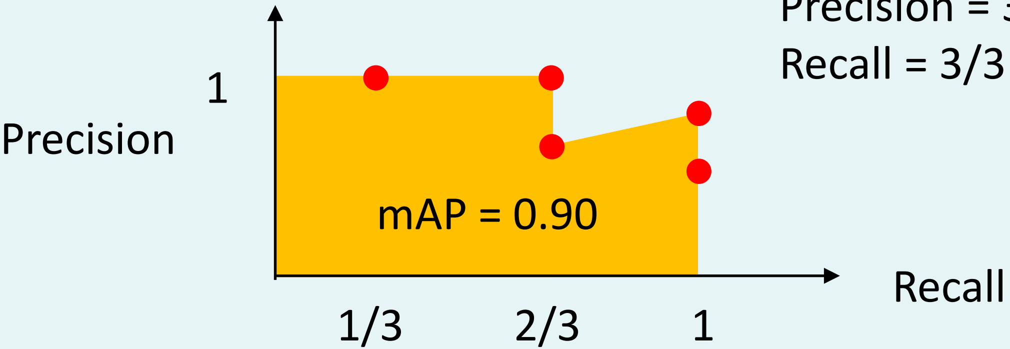
Precision = 3/5

Recall = 3/3 = 1

Evaluation of Object Detection (10)

- Mean Average Precision:
  - The area under the precision-recall curve.





Precision = 3/5

Recall = 3/3 = 1

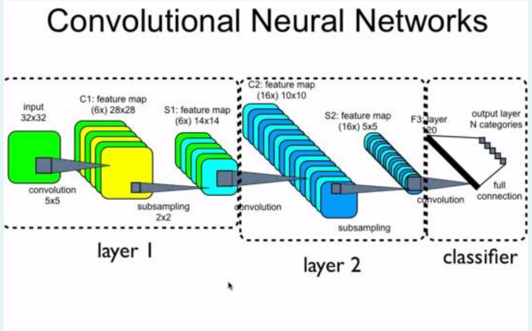
Content

- Introduction to Object Detection
- Detecting Single Object
- Detecting Multiple Objects
  - Sliding Window
  - Region Proposal Methods
  - YOLO Algorithm

Detecting Single Object (1)

- If an image only has one foreground object, the solution is quite straightforward.
- Previously for image classification, the outputs of our CNN only contains the probability of a certain class.

→



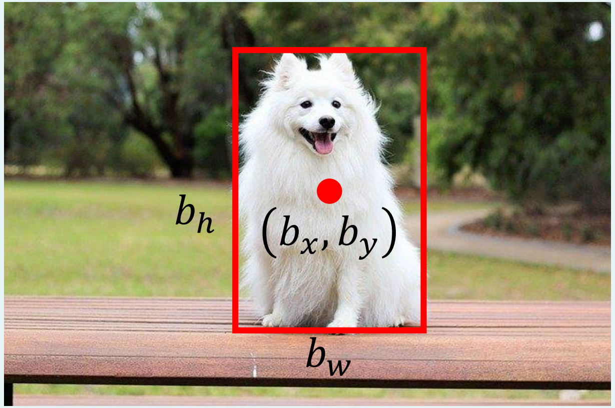
This Photo by Unknown Author is licensed under CC BY-SA

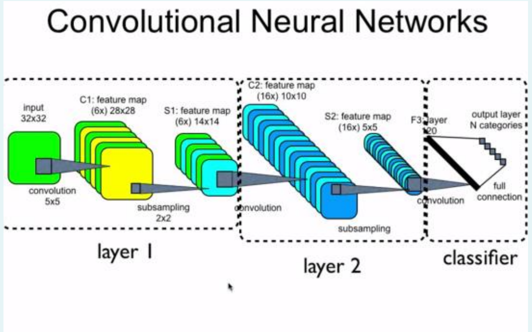
→

$$y = \begin{bmatrix} p(\text{cat}) \\ p(\text{dog}) \\ p(\text{car}) \\ p(\text{truck}) \end{bmatrix}$$

Detecting Single Object (2)

- We now add four more outputs corresponding to:
  - X-position of the center of bounding box;
  - Y-position of the center of bounding box;
  - Width of bounding box; and
  - Height of bounding box.





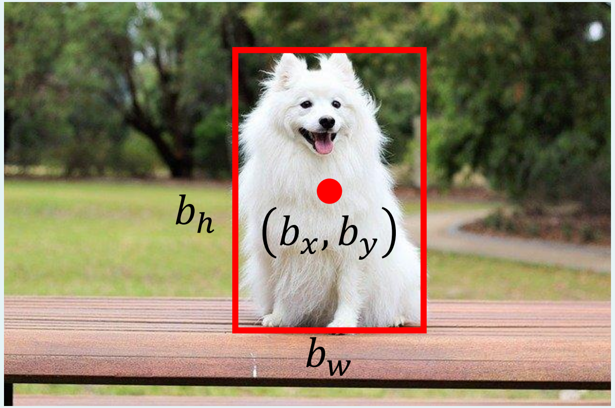
This Photo by Unknown Author is licensed under CC BY-SA

$$y = \begin{bmatrix} p(\text{cat}) \\ p(\text{dog}) \\ p(\text{car}) \\ p(\text{truck}) \\ b_x \\ b_y \\ b_w \\ b_h \end{bmatrix}$$

Detecting Single Object (3)

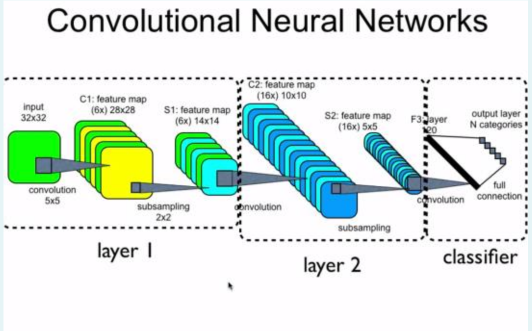
- The values of b_x, b_y, b_w, b_h are usually specified between 0 and 1:
  - Top left corner of image is $(0{,}0)$.
  - Bottom right corner of image is normalized to $(1{,}1)$.
  - b_w and b_h specified as ratio to the full width and height.

$(0{,}0)$



$(1{,}1)$

$\longrightarrow$

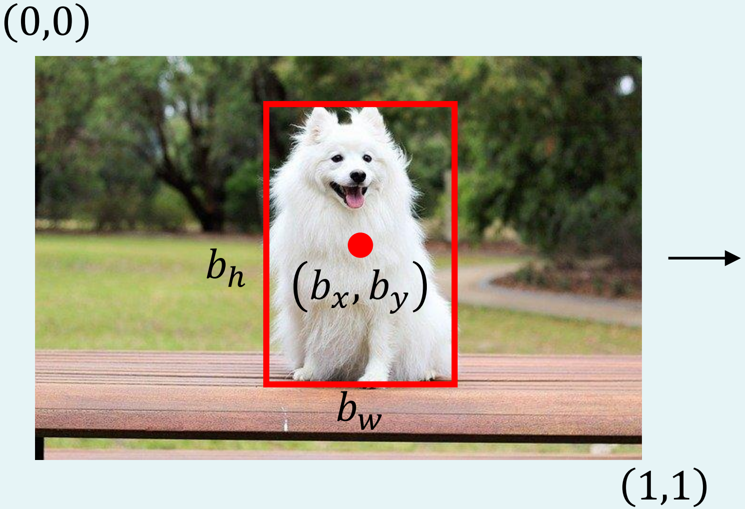

Convolutional Neural Networks
input 32x32
C1: feature map (6x) 28x28
S1: feature map (6x) 14x14
C2: feature map (16x) 10x10
S2: feature map (16x) 5x5
F3: layer 120
output layer N categories
convolution 5x5
subsampling 2x2
convolution
subsampling
convolution
full connection
layer 1
layer 2
classifier

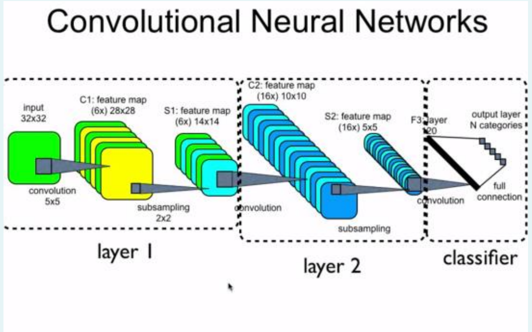
$\longrightarrow$

$$y = \begin{bmatrix} p(\text{cat}) \\ p(\text{dog}) \\ p(\text{car}) \\ p(\text{truck}) \\ b_x \\ b_y \\ b_w \\ b_h \end{bmatrix}$$

Detecting Single Object (4)

- Additionally, we add one more output p_o which specifies the probability that there is indeed a foreground object.
  - $p_o \to 1$ if there is an object.
  - $p_o \to 0$ if there is no object.



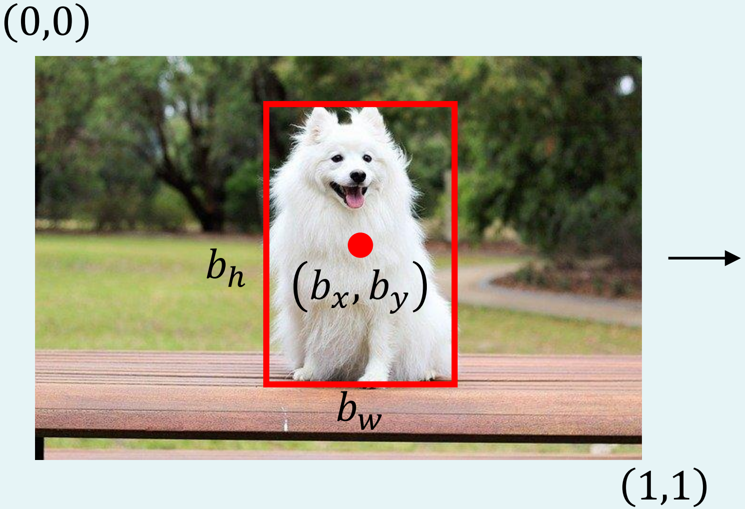


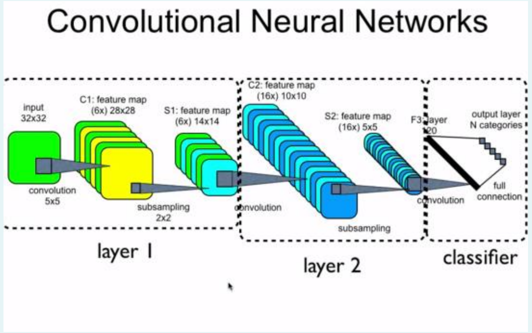
This Photo by Unknown Author is licensed under CC BY-SA

$$y = \begin{bmatrix} p_o \\ p(\text{cat}) \\ p(\text{dog}) \\ p(\text{car}) \\ p(\text{truck}) \\ b_x \\ b_y \\ b_w \\ b_h \end{bmatrix}$$

Target Output (1)

- So in this example, the target output (used in training, validation and testing) would be:





Convolutional Neural Networks
input 32x32
C1: feature map (6x) 28x28
S1: feature map (6x) 14x14
C2: feature map (16x) 10x10
S2: feature map (16x) 5x5
F3 layer 120
output layer N categories
convolution 5x5
subsampling 2x2
convolution
subsampling
convolution
full connection
layer 1
layer 2
classifier

This Photo by Unknown Author is licensed under CC BY-SA

$$d = \begin{bmatrix} p_o \\ p(\text{cat}) \\ p(\text{dog}) \\ p(\text{car}) \\ p(\text{truck}) \\ b_x \\ b_y \\ b_w \\ b_h \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0.55 \\ 0.45 \\ 0.3 \\ 0.7 \end{bmatrix}$$

Target Output (2)

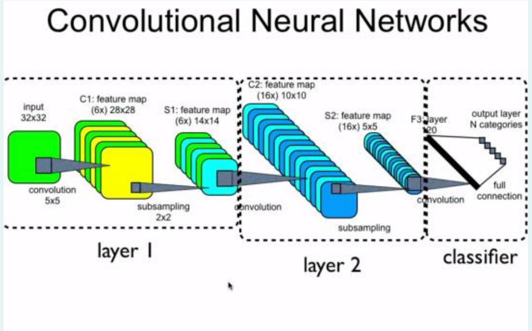
- For another image where there is no foreground object, the target output would be $p_o = 0$ followed by $*$ which means “don’t care”.

$(0,0)$

[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)

$(1,1)$

$\longrightarrow$


Convolutional Neural Networks
input 32x32
C1: feature map (6x) 28x28
S1: feature map (6x) 14x14
C2: feature map (16x) 10x10
S2: feature map (16x) 5x5
F3 layer 120
output layer N categories
convolution 5x5
subsampling 2x2
convolution
subsampling
convolution
full connection
layer 1
layer 2
classifier

[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)

$\longrightarrow$

$$d = \begin{bmatrix} p_o \\ p(\text{cat}) \\ p(\text{dog}) \\ p(\text{car}) \\ p(\text{truck}) \\ b_x \\ b_y \\ b_w \\ b_h \end{bmatrix} = \begin{bmatrix} 0 \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \end{bmatrix}$$

Training of CNN (1)

- After you ☹ (or someone else ☺) labeled all the images with the correct target outputs, we then train our CNN to predict the outputs!
- But how should the loss function be?
  - It should reflect the cases where there is foreground object or not:

$$L = \begin{cases} (p_o - \hat{p}_o)^2 + (p_{c1} - \hat{p}_{c1})^2 + (p_{c2} - \hat{p}_{c2})^2 + \cdots + (p_{cn} - \hat{p}_{cn})^2 \\ \quad + \left(b_x - \hat{b}_x\right)^2 + \left(b_y - \hat{b}_y\right)^2 + \left(b_w - \hat{b}_w\right)^2 + \left(b_h - \hat{b}_h\right)^2 & \text{if } p_o = 1 \\ (p_o - \hat{p}_o)^2 & \text{if } p_o = 0 \end{cases}$$

Training of CNN (2)

- A few notes about the loss function:

$$\boxed{L = \begin{cases} (p_o - \hat{p}_o)^2 + (p_{c1} - \hat{p}_{c1})^2 + (p_{c2} - \hat{p}_{c2})^2 + \cdots + (p_{cn} - \hat{p}_{cn})^2 \\ \quad + \left(b_x - \hat{b}_x\right)^2 + \left(b_y - \hat{b}_y\right)^2 + \left(b_w - \hat{b}_w\right)^2 + \left(b_h - \hat{b}_h\right)^2 & \text{if } p_o = 1 \\ \\ (p_o - \hat{p}_o)^2 & \text{if } p_o = 0 \end{cases}}$$

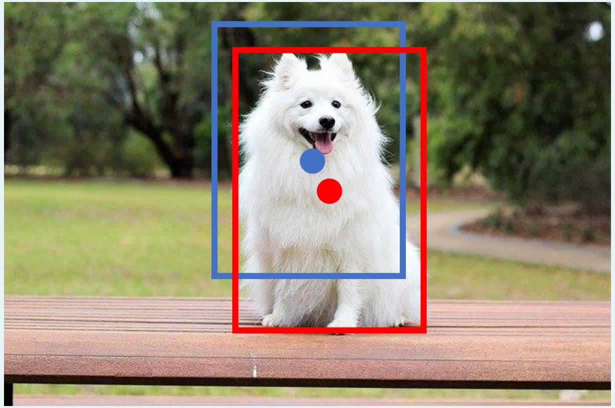
  - The loss function for the “classes” (p_{c1} to p_{cn}) can be replaced by cross entropy if preferred, but the overall sum might have to be changed to weighted sum.
  - The loss function when $p_o = 0$ reflects the fact that we “don’t care” about all other outputs.

Training of CNN (3)

- After training, CNN would hopefully give good prediction for foreground objects, for example:

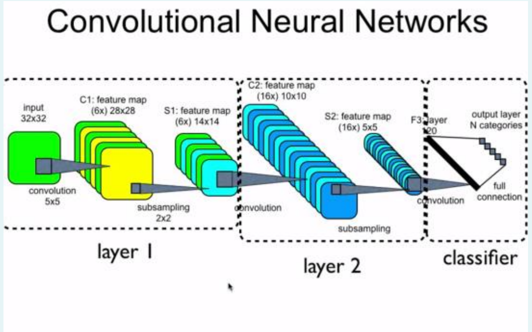
Red = ground truth;
Blue = predicted

(0,0)



(1,1)

→



This Photo by Unknown Author is licensed under CC BY-SA

→

$$y = \begin{bmatrix} p_o \\ p(\text{cat}) \\ p(\text{dog}) \\ p(\text{car}) \\ p(\text{truck}) \\ b_x \\ b_y \\ b_w \\ b_h \end{bmatrix} = \begin{bmatrix} 0.9 \\ 0.2 \\ 0.9 \\ 0.1 \\ 0.1 \\ 0.55 \\ 0.45 \\ 0.3 \\ 0.7 \end{bmatrix}$$

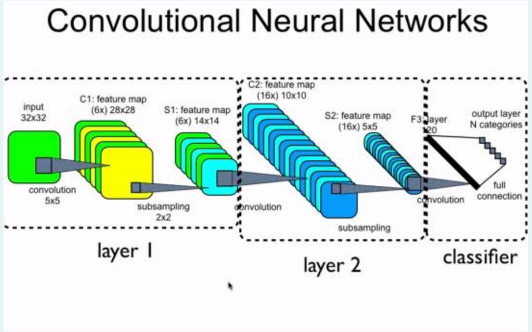
Training of CNN (4)

- If there is no foreground object, p_o should hopefully be close to zero.
  - The other output nodes will still chunk out some values, but they will be irrelevant.

(0,0)

(1,1)

→


Convolutional Neural Networks
input 32x32
C1: feature map (6x) 28x28
S1: feature map (6x) 14x14
C2: feature map (16x) 10x10
S2: feature map (16x) 5x5
F3: layer 120
output layer N categories
convolution 5x5
subsampling 2x2
convolution
subsampling
convolution
full connection
layer 1
layer 2
classifier

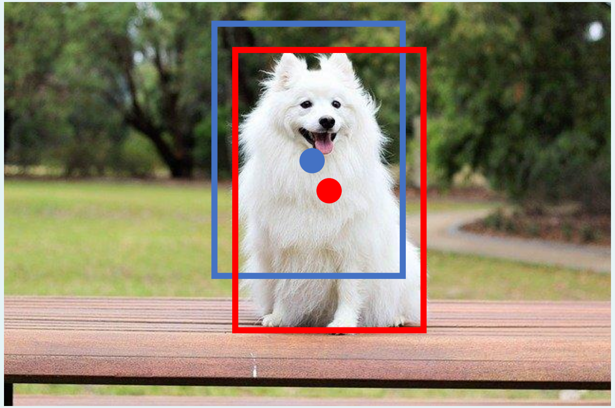
This Photo by Unknown Author is licensed under CC BY-SA

→

$$y = \begin{bmatrix} p_o \\ p(\text{cat}) \\ p(\text{dog}) \\ p(\text{car}) \\ p(\text{truck}) \\ b_x \\ b_y \\ b_w \\ b_h \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.1 \\ 0.1 \\ 0.55 \\ 0.45 \\ 0.3 \\ 0.7 \end{bmatrix}$$

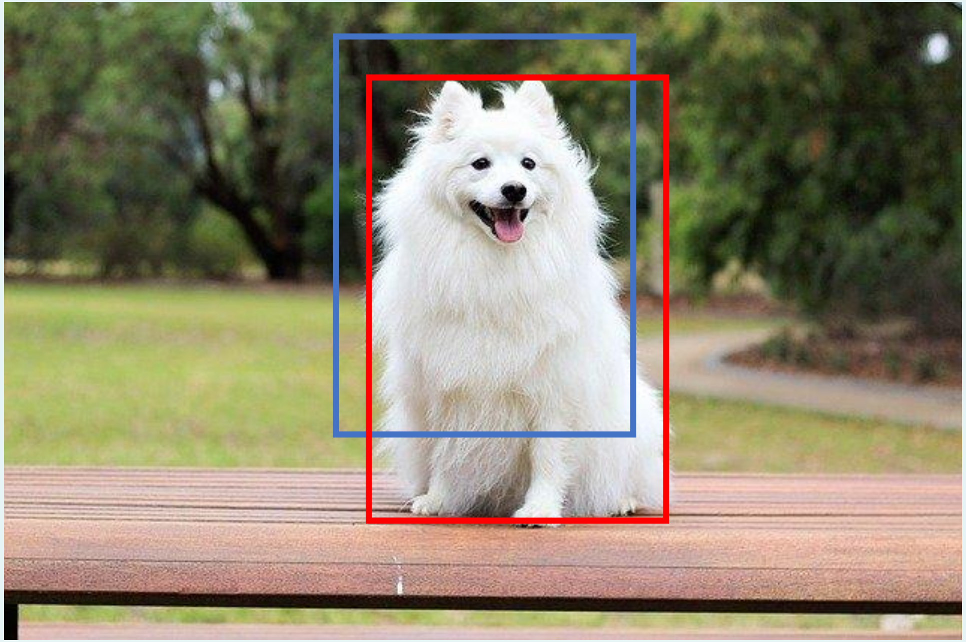
Intersection over Union (1)

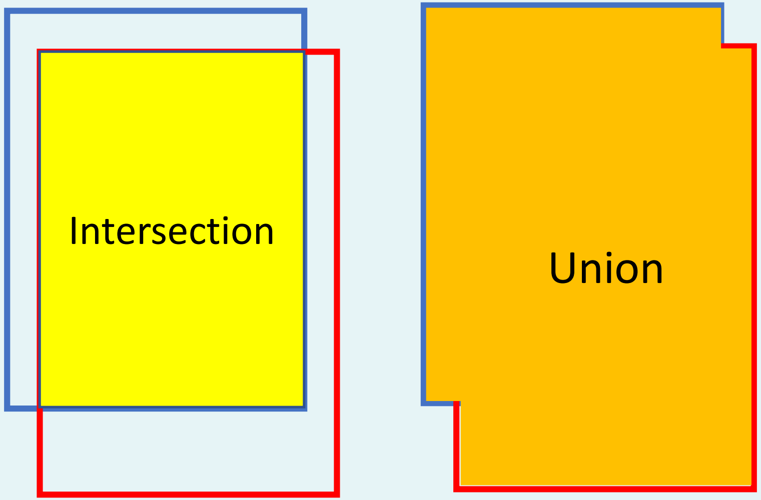
- In the example with foreground object, we saw that the predicted bounding box may not coincide exactly with the ground truth.
- How do we evaluate how good the prediction is?



Intersection over Union (2)

- A commonly used quantity is called “Intersection over Union (IOU)”:

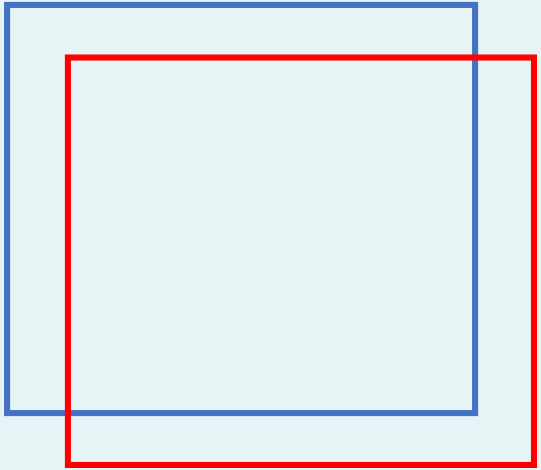




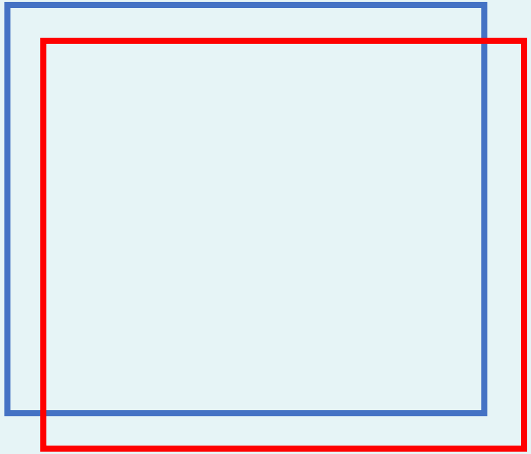
$$\text{IOU} = \frac{\text{Area of Intersection}}{\text{Area of Union}}$$

Intersection over Union (3)

- IOU of > 0.5 is considered decent.
- IOU of > 0.7 is considered good.
- IOU of > 0.9 is considered almost perfect!



IOU = 0.62



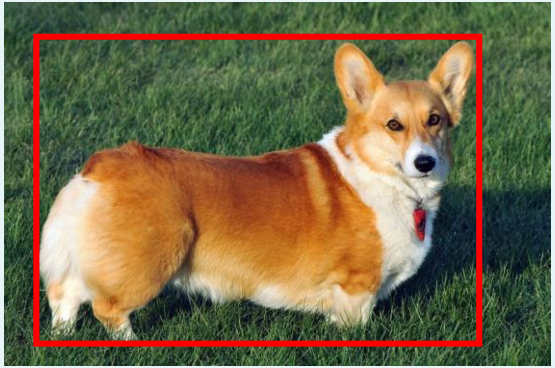
IOU = 0.7

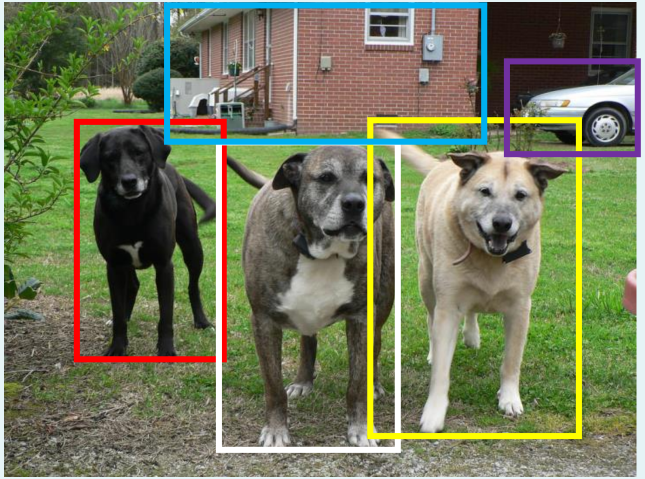
Content

- Introduction to Object Detection
- Detecting Single Object
- Detecting Multiple Objects
  - Sliding Window
  - Region Proposal Methods
  - YOLO Algorithm

Detecting Multiple Objects (1)

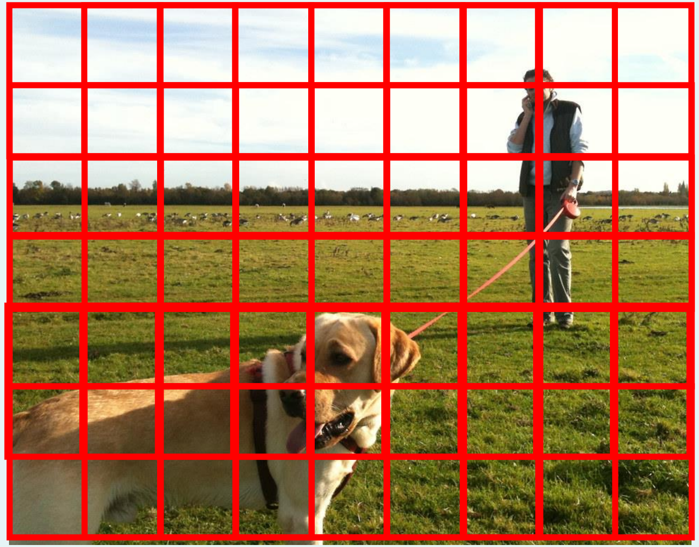
- Unfortunately, the same method cannot be applied directly to images with multiple objects!
- One reason is that the number of objects vary, so we can’t pre-determine how many bounding boxes (and CNN outputs) are needed!





Sliding Window (1)

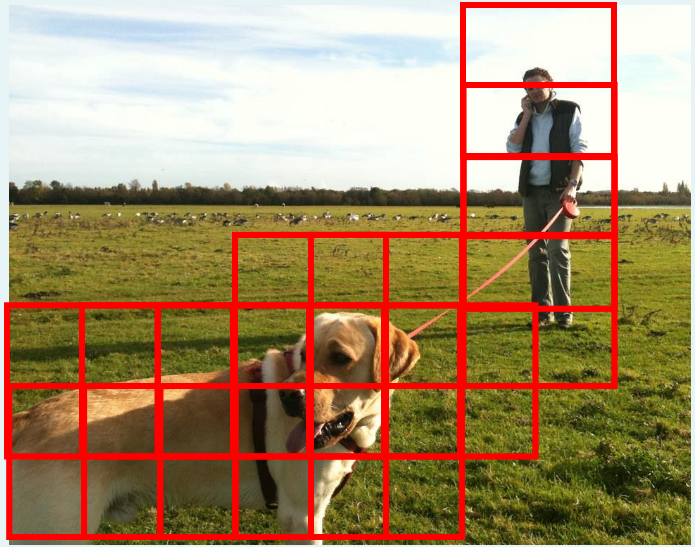
- One simple solution is to use sliding windows to focus on one area of the image at a time.
- Treat each area as an "individual" image and run through CNN for object detection.
  - Object or no object;
  - If object, then class and bounding box parameters wrt the sliding window.



This Photo by Unknown Author is licensed under CC BY-SA-NC

Sliding Window (2)

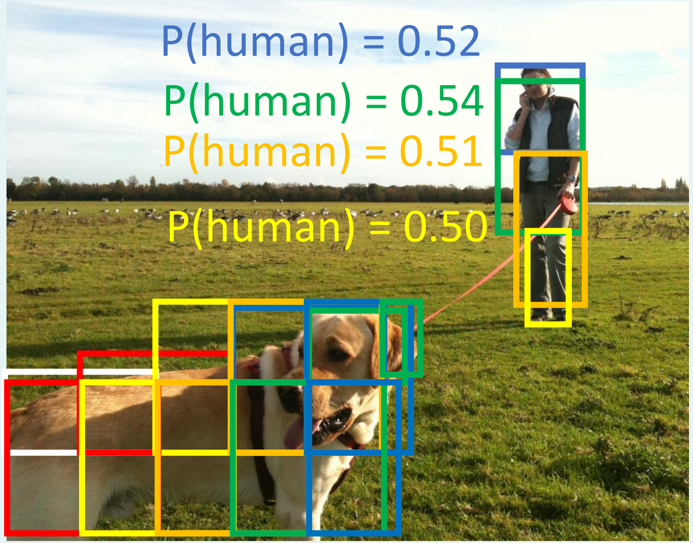
- We may end up with several sliding windows which the CNN determines as having foreground objects ($p_o \rightarrow 1$).



This Photo by Unknown Author is licensed under CC BY-SA-NC

Sliding Window (3)

- For these sliding windows, the class probability and the bounding box are given and recorded.
  - Some objects may be detected several times, but that’s ok. We will deal with this issue later.

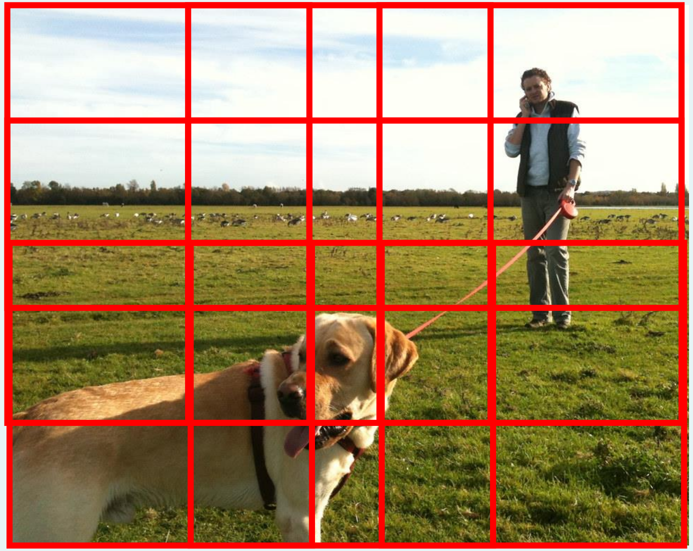


p/s: To avoid crowding the image, the original sliding windows are not shown.

This Photo by Unknown Author is licensed under CC BY-SA-NC

Sliding Window (4)

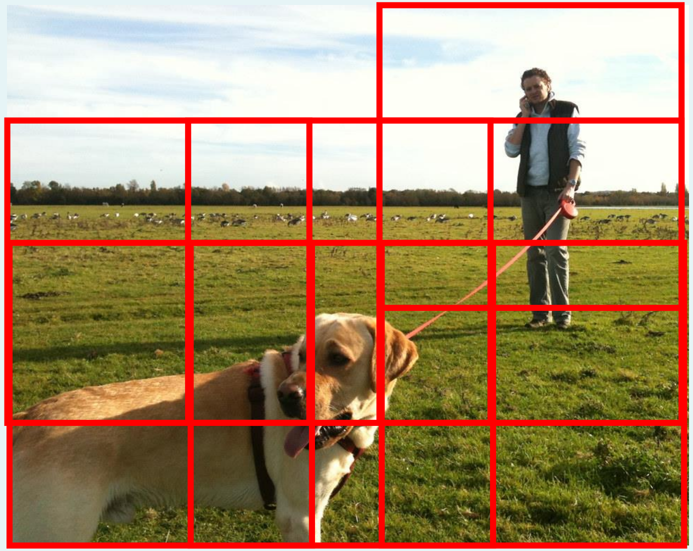
- We don't have to stop here. Instead, we can continue using larger sliding windows.



This Photo by Unknown Author is licensed under CC BY-SA-NC

Sliding Window (5)

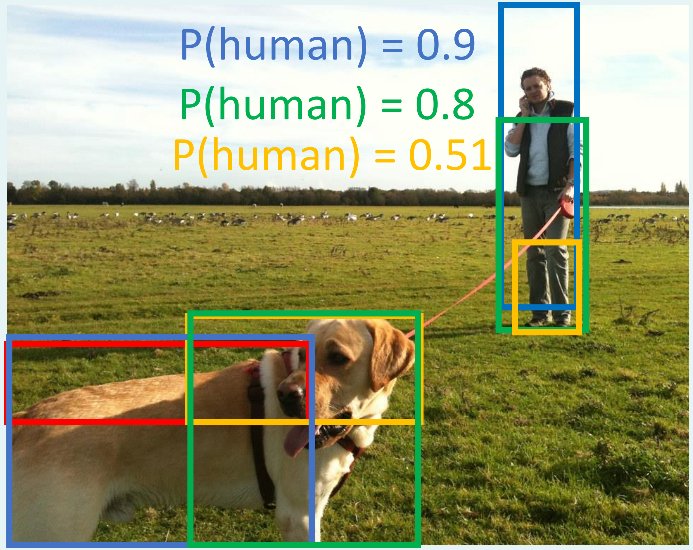
- Again, some of the sliding windows would be identified as having foreground objects.



This Photo by Unknown Author is licensed under CC BY-SA-NC

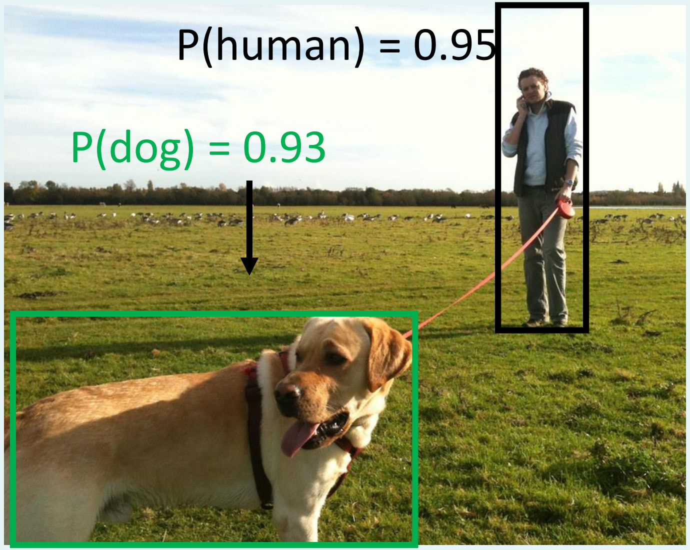
Sliding Window (6)

- For these sliding windows, the class probability and the bounding box are given and recorded.
- The probability of a class should differ from earlier when using smaller windows.



Sliding Window (7)

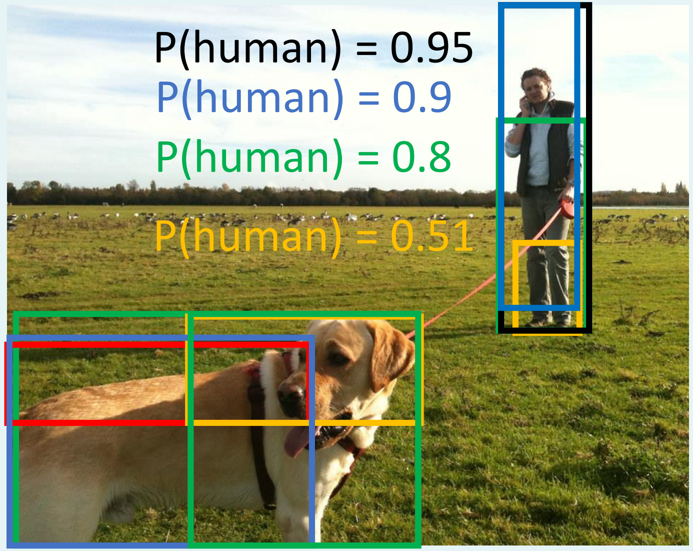
- Continue with even larger windows, and record the class probability and bounding box if foreground objects are detected.
- End the process when the pre-defined largest window has been used.



This Photo by Unknown Author is licensed under CC BY-SA-NC

Non-Max Suppression (1)

- We have so far recorded all the bounding boxes, when different sizes of sliding windows were used.
- There are many overlaps!
- How do we deal with this?

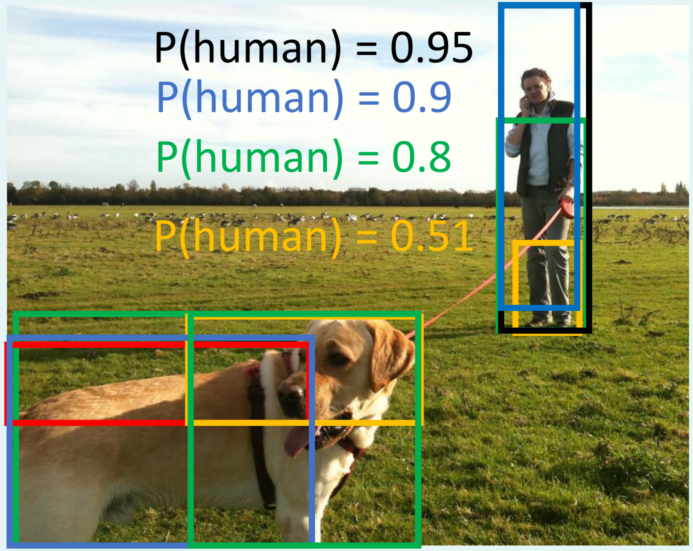


p/s: For the picture on the right, the boxes found when using small sliding windows are not shown to avoid overcrowding the image.

Non-Max Suppression (2)

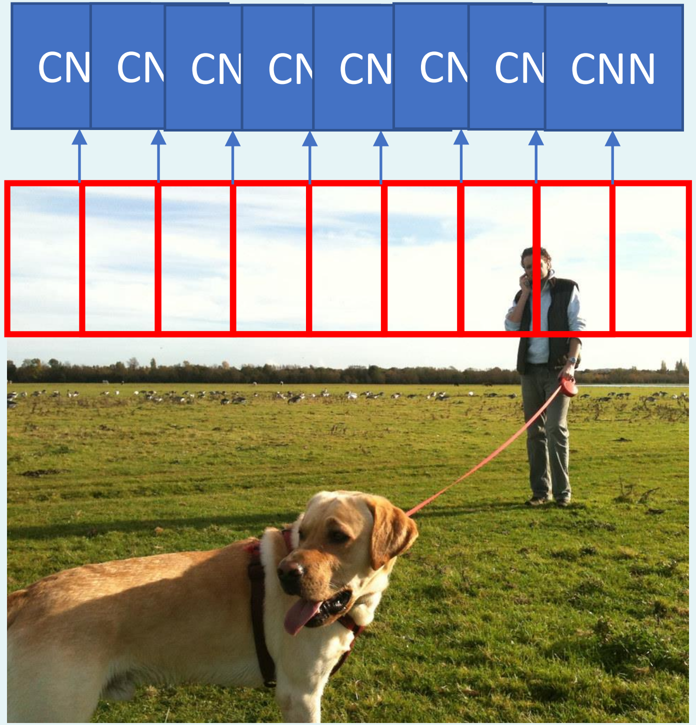
- The idea is as follows: For each class:

1. Pick the box with the highest probability. Output that as an accepted answer.
2. Discard other boxes with a large overlap (e.g. IOU > 0.5).
3. If there are any remaining boxes, go to 1 with next highest-scoring box.



Problem w. Sliding Window Method (1)

- The problem with the sliding window method is high computational cost!
  - Need to go through many steps, and need to explore different window sizes!
  - Also, with each sliding window, the image needs to go through the CNN once!



This Photo by Unknown Author is licensed under CC BY-SA-NC

Problem w. Sliding Window Method (2)

- Researchers have therefore proposed better / more advanced approach for multiple object detection.
- Two main groups:
  - Two-stage approach with region proposal
  - One-stage method e.g. YOLO algorithm
- We will discuss these in the subsequent sections.

Content

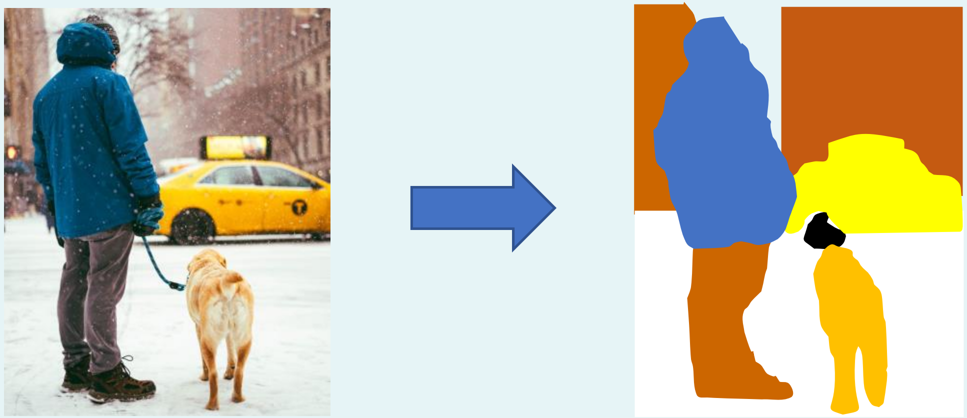
- Introduction to Object Detection
- Detecting Single Object
- Detecting Multiple Objects
  - Sliding Window
  - Region Proposal Methods
  - YOLO Algorithm

Region-Based CNN (R-CNN) (1)

- Instead of trying, in a “brute force” manner, different window sizes across the whole image, R-CNN uses region proposal methods to first find some (~2000) regions of interest ROI.
  - The method is proposed in Girshick et al. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”, CVPR 2014.

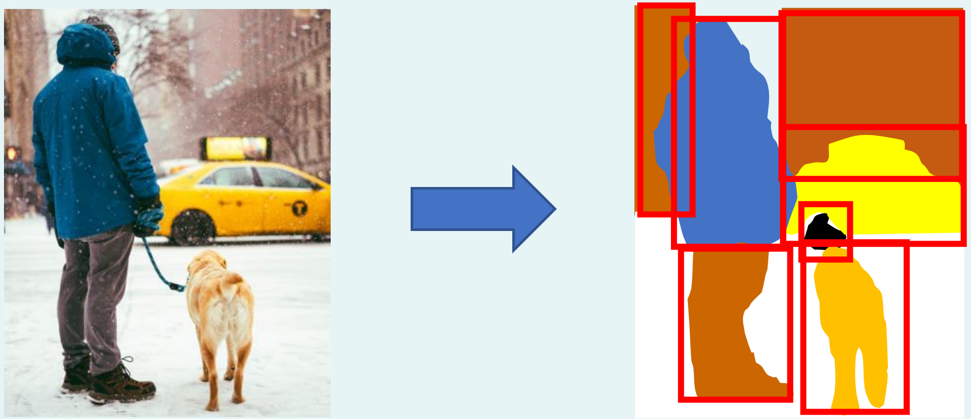
Region-Based CNN (R-CNN) (2)

- First, a segmentation algorithm is used to segment objects according to color, texture, similarity etc.



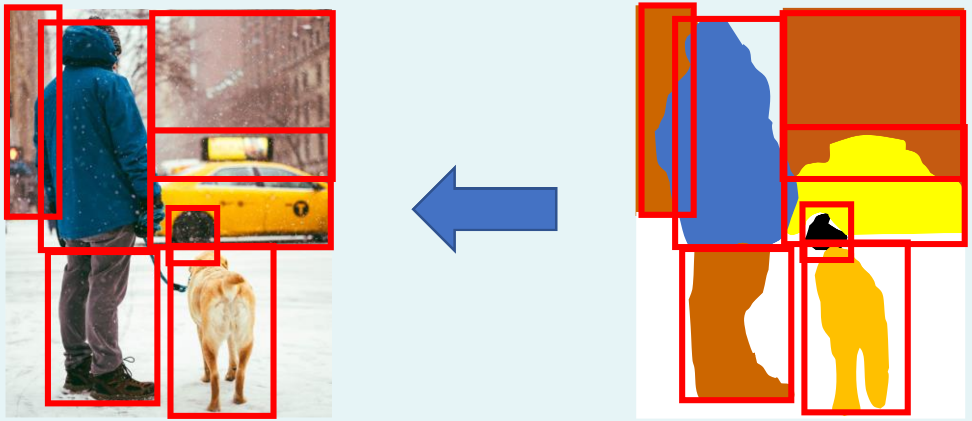
Region-Based CNN (R-CNN) (3)

- After segmentation, a region proposal method called "selective search" by hierarchical grouping is used to identify blobs with high probability of being an object, i.e. ROI.



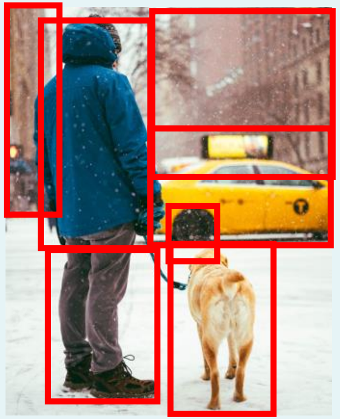
Region-Based CNN (R-CNN) (4)

- These regions will then be superimposed to the original image.



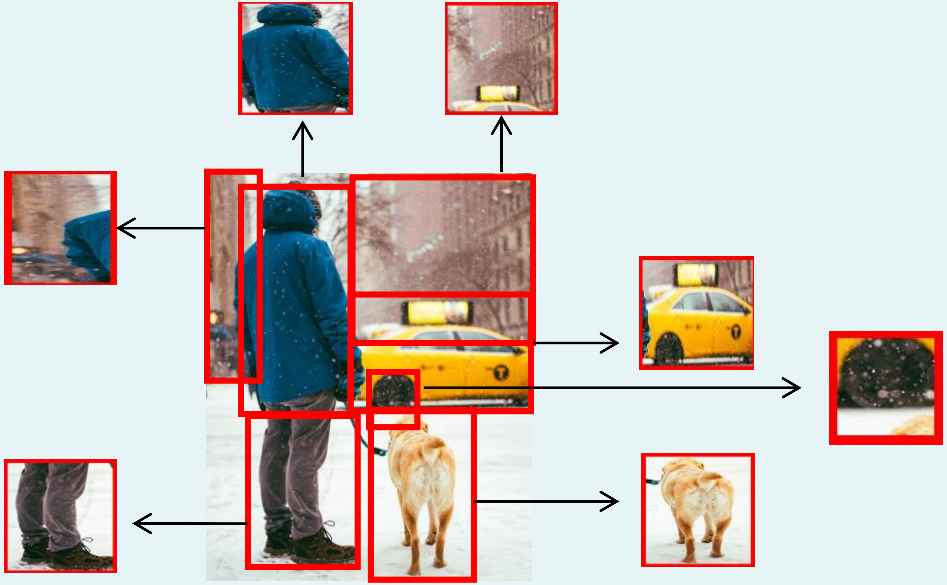
Region-Based CNN (R-CNN) (5)

- You might have guessed that we can now pass each region into a CNN to classify and locate the object.
- But there is one caveat – The regions have different dimensions!



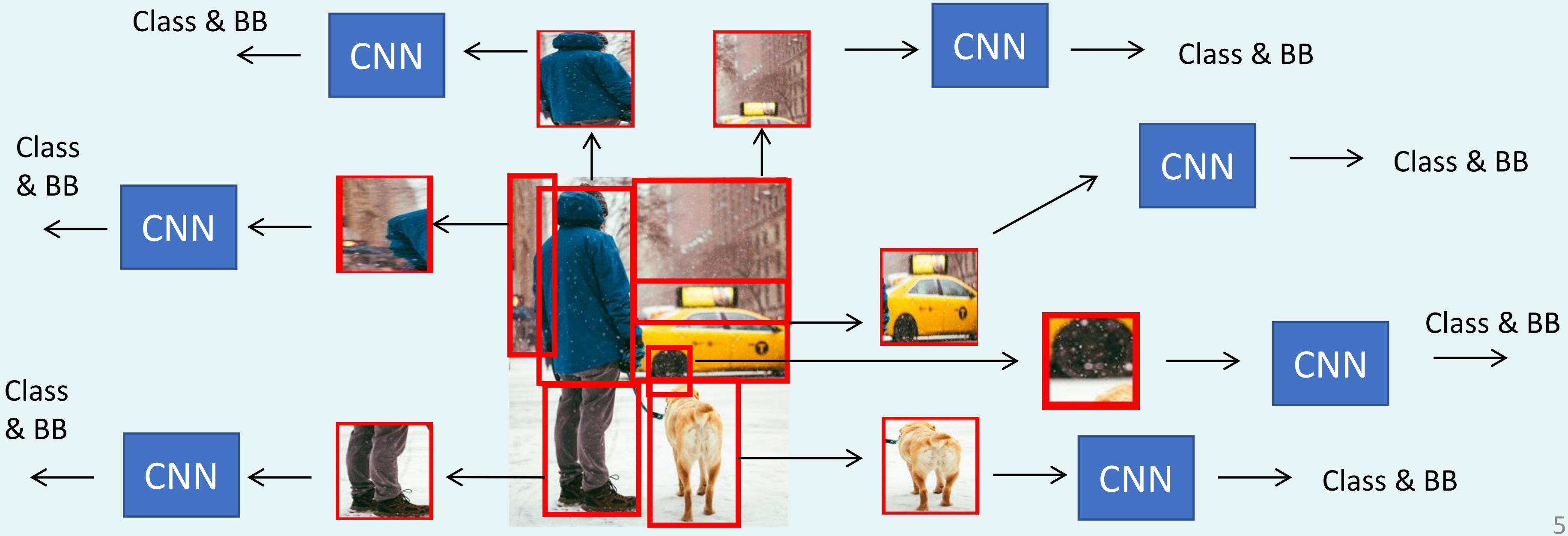
Region-Based CNN (R-CNN) (6)

- The solution is thus to warp the image regions into the same sizes, e.g. 224 x 224.



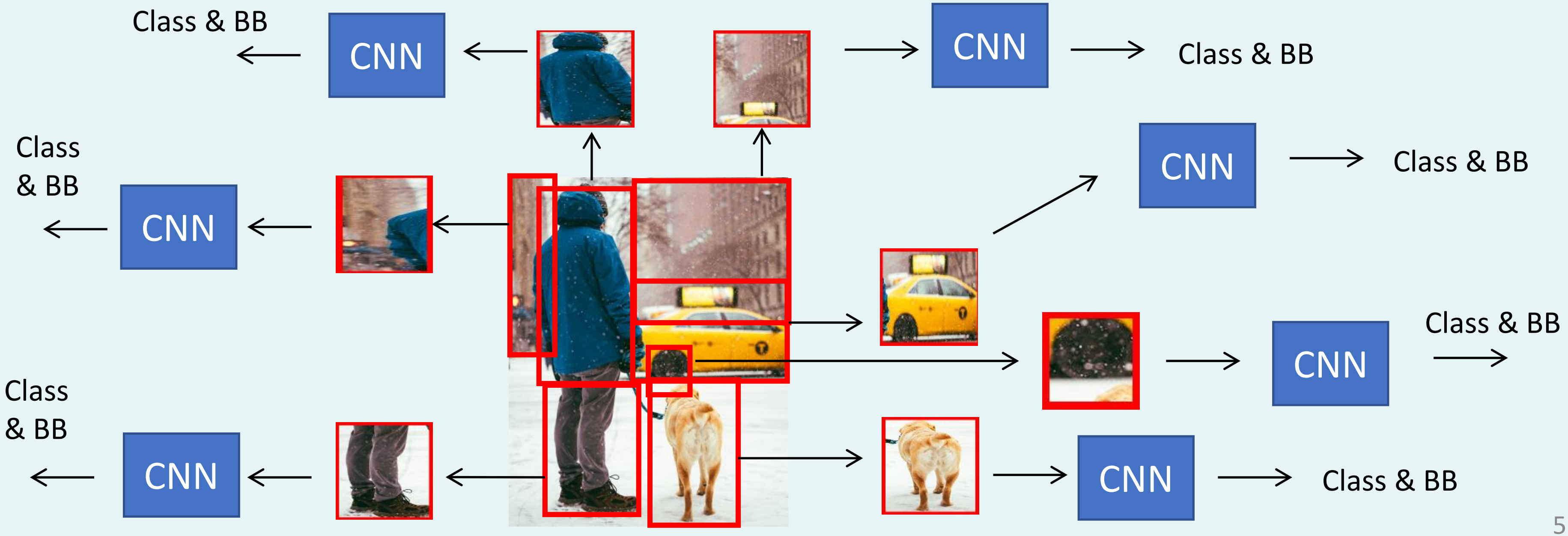
Region-Based CNN (R-CNN) (7)

- We then feed these warped images to CNN for object detection.



Region-Based CNN (R-CNN) (8)

- A regression layer will make the coordinates of the bounding box more precise.

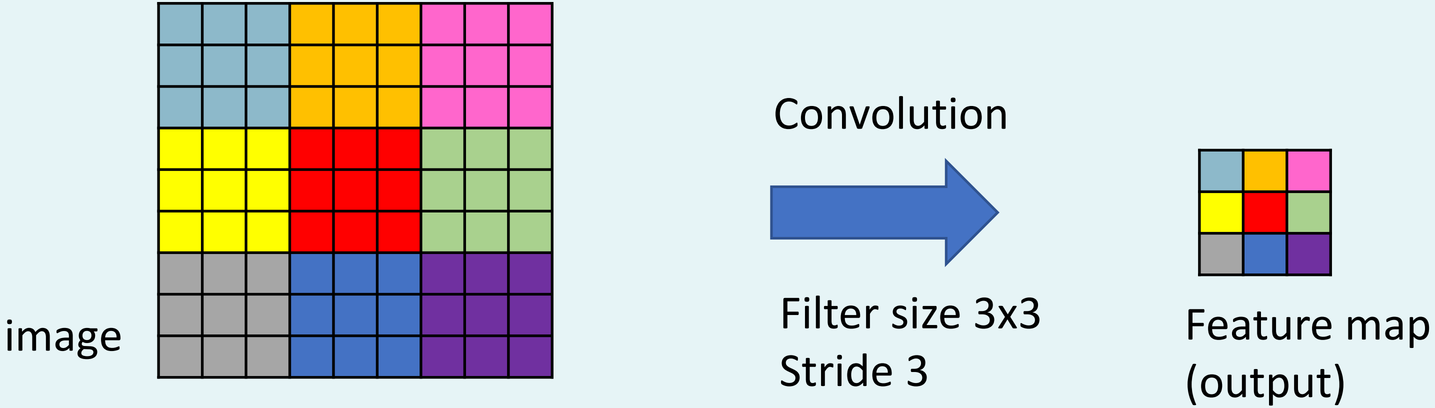


Fast R-CNN (1)

- While R-CNN reduced the number of windows by using the selective search method, it still ends up with ~2000 regions, which needs to go through CNN each.
    - Still high computation cost!
    - E.g. object detection in one image during testing: ~ 2s for region proposal + ~47s for all CNN.

Fast R-CNN (2)

- Girshick then proposed the “Fast R-CNN” in ICCV 2015.
- The idea behind this paper is that when we perform convolution, the spacial location of a region in image is directly correlated with the spacial location of a region in the output (feature map).



image

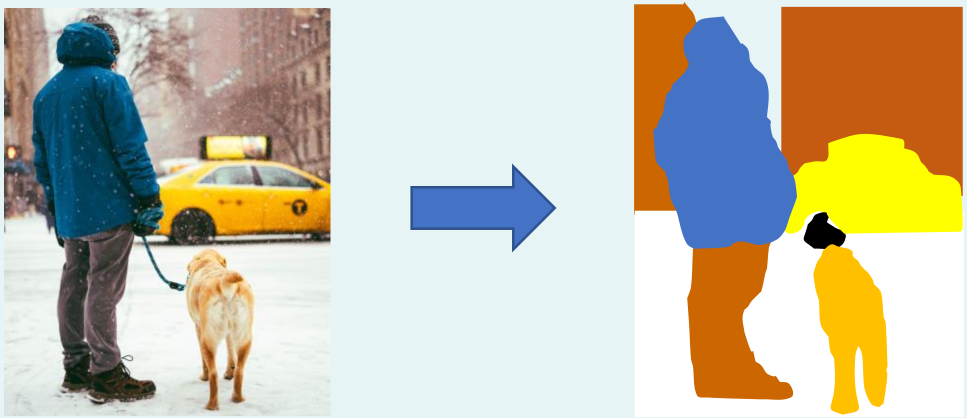
Convolution

Filter size 3x3
Stride 3

Feature map
(output)

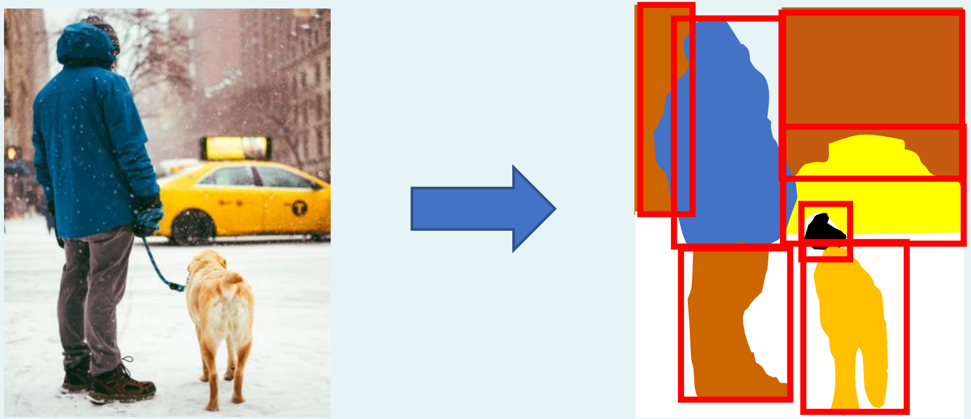
Fast R-CNN (3)

- The first two steps of Fast R-CNN are the same as R-CNN.
- A segmentation method is run to segment objects according to color, texture, size and shape.



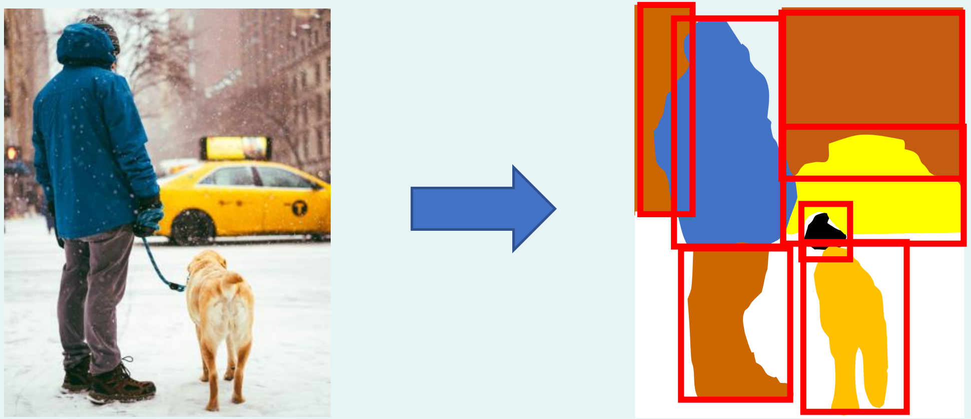
Fast R-CNN (4)

- After segmentation, a region proposal method called “selective search” is used to identify blobs with high probability of being an object, i.e. ROI.



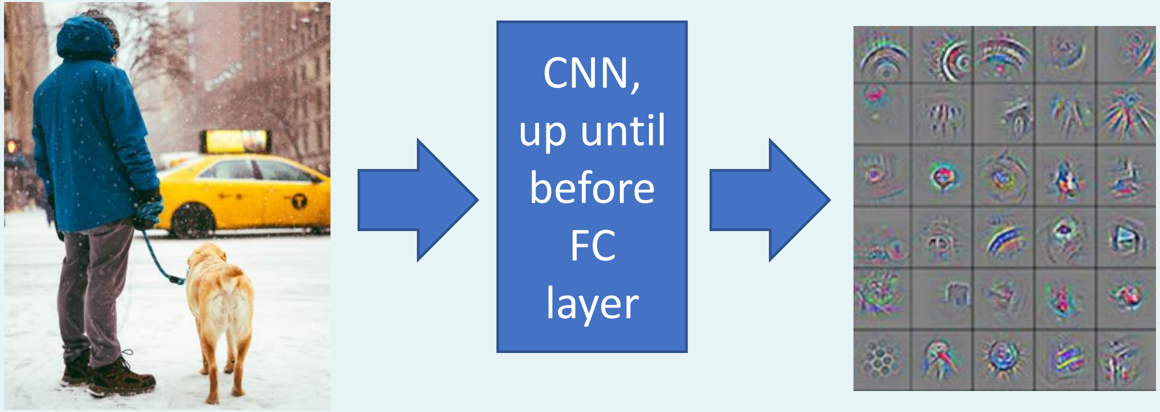
Fast R-CNN (5)

- But now, there is no need to impose the ROI on the original image.
  - We merely keep a record of where the ROIs are.



Fast R-CNN (6)

- We now run this original image through a CNN, up until right before FC layer, to extract features.
  - The CNN can be AlexNet, VGG, ResNet etc.



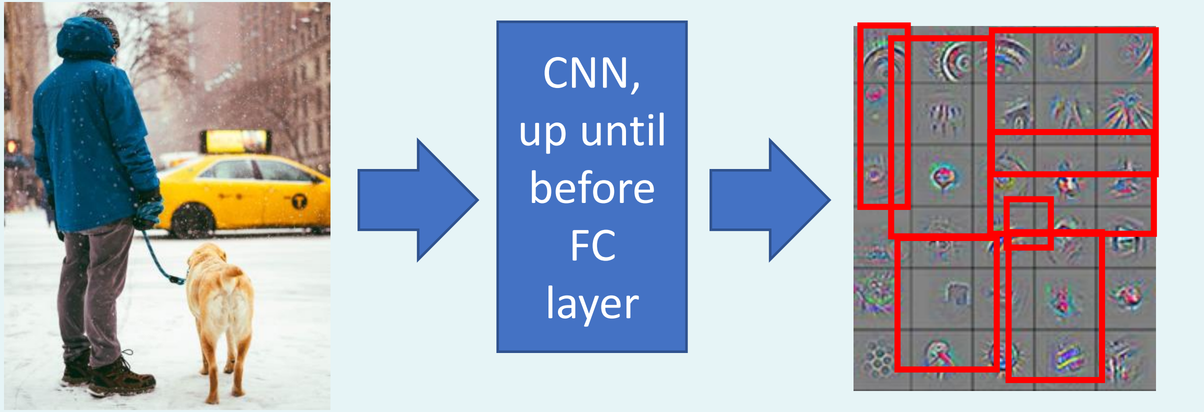
CNN, up until before FC layer

image

Feature map

Fast R-CNN (7)

- Here, we project the ROIs onto the feature map.
  - The spacial locations are maintained, though sizes are different from original which can be calculated based on convolution operation.



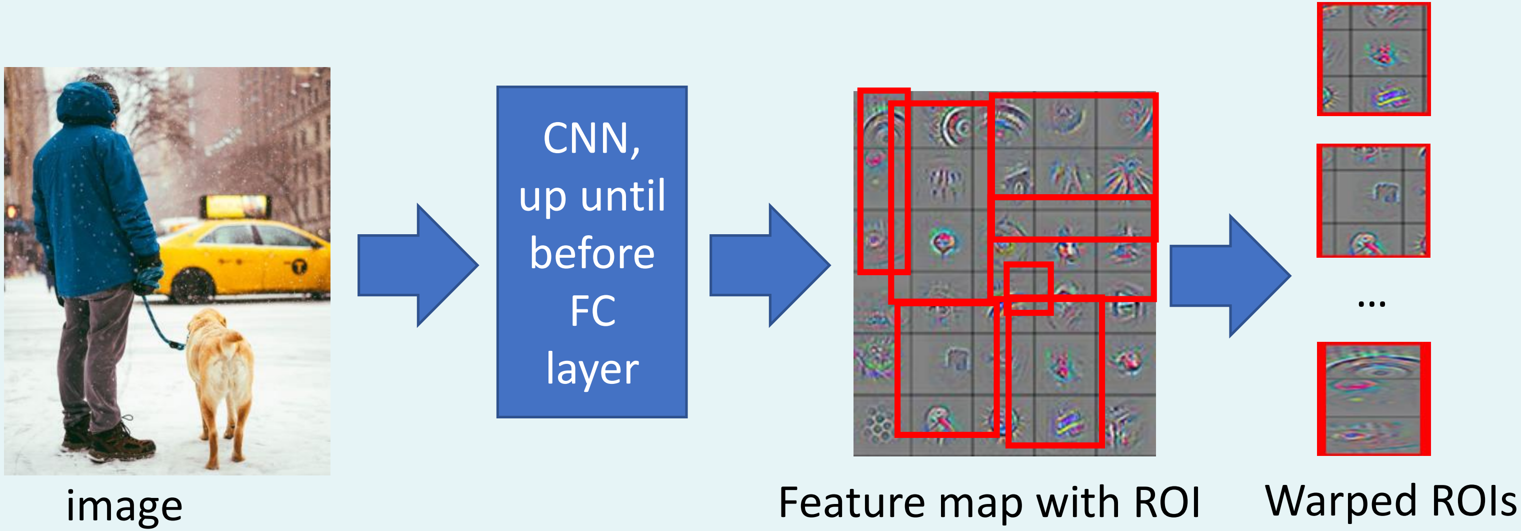
CNN, up until before FC layer

image

Feature map with ROI

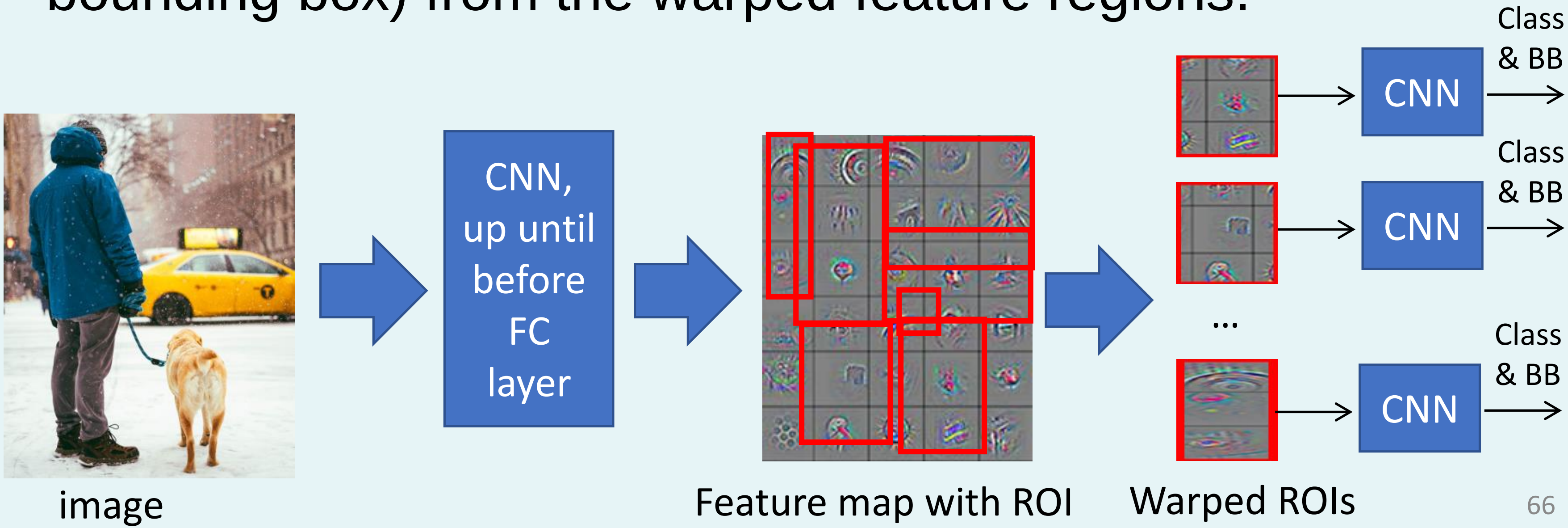
Fast R-CNN (8)

- Next, we warp those feature ROIs into fixed dimensions.



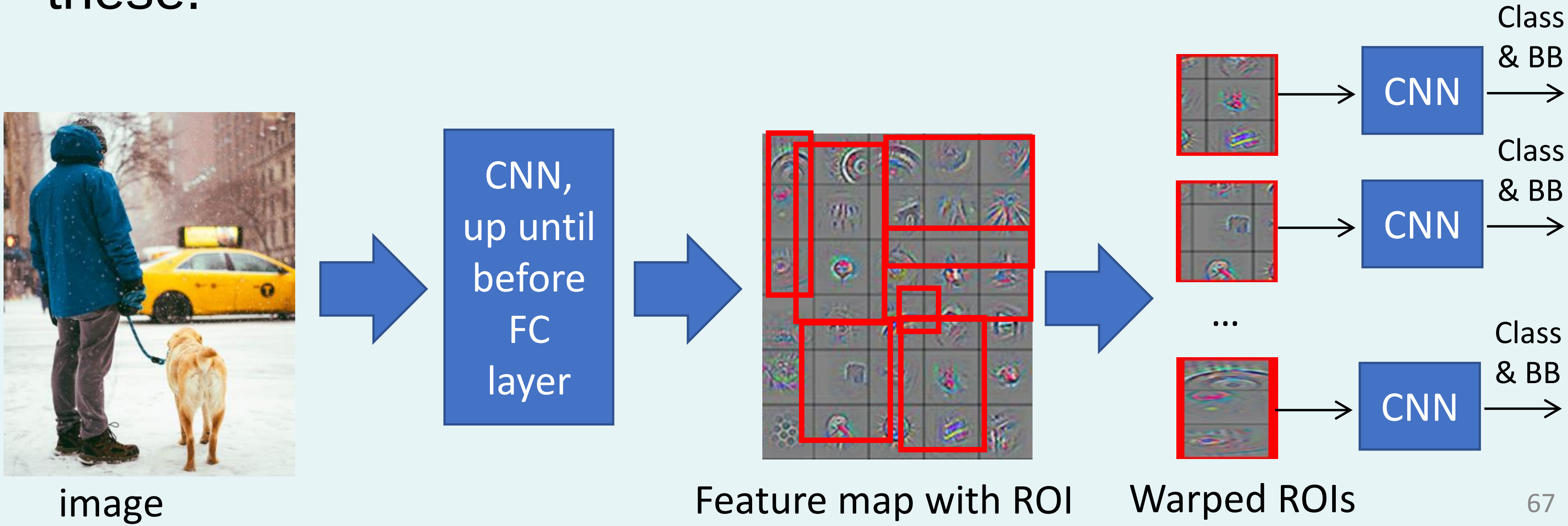
Fast R-CNN (9)

- Finally, we use some lightweight “Per-Region Networks” to perform object detection (classification and improving bounding box) from the warped feature regions.



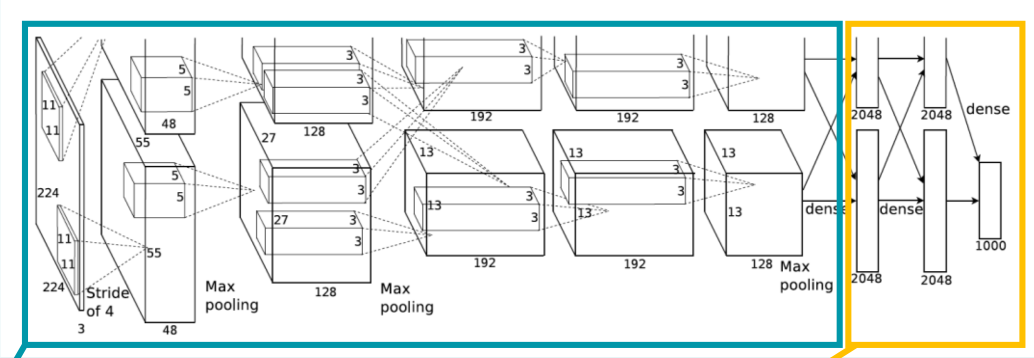
Fast R-CNN (10)

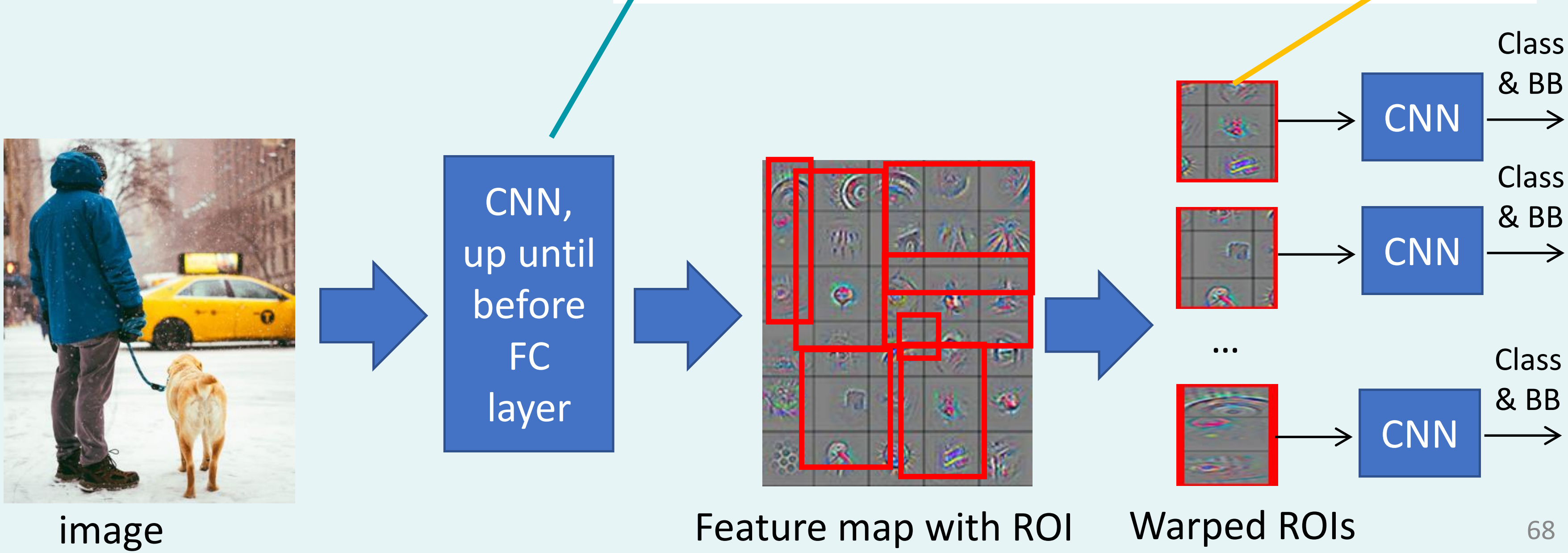
- The “per-region network” can be quite shallow and thus not computational intensive, even if we need ~2000 of these.



Fast R-CNN (11)

- E.g. AlexNet:





CNN, up until before FC layer

image

Feature map with ROI

Warped ROIs

CNN

Class & BB

CNN

Class & BB

...

CNN

Class & BB

Fast R-CNN (12)

- The changes done in Fast R-CNN (swapping convolution with cropping of regions) made it much faster than R-CNN.
  - E.g. object detection in one image during testing: ~ 2s for region proposal + ~0.3s for all CNN.
  - The latter was down from 47s!

Faster R-CNN (1)

- Let’s look at the computation time for R-CNN and Fast R-CNN:

	Region Proposal	Object Detection
R-CNN	2s	47s
Fast R-CNN	2s	0.3s

- As can be seen, the bottleneck is at the region proposal now.
- We need a way to reduce the region-proposal time!

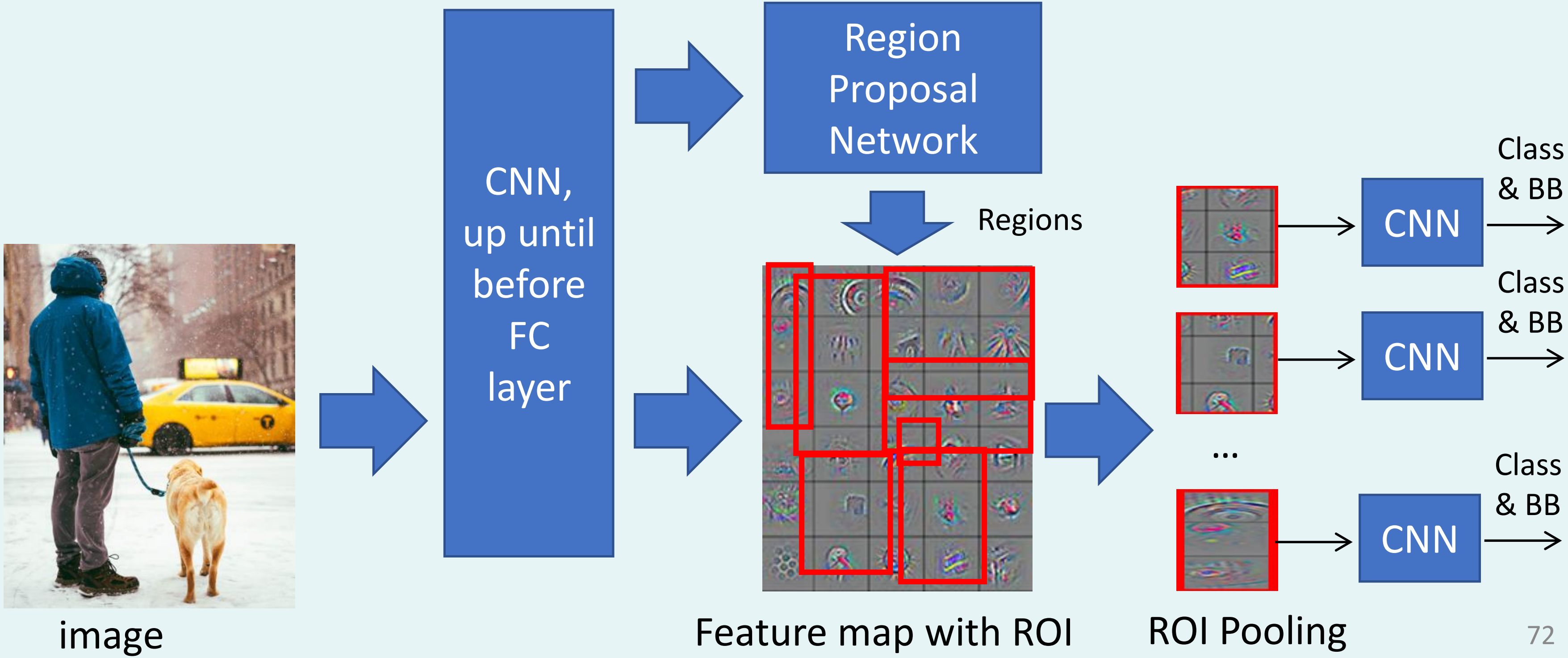
Faster R-CNN (2)

- In 2015, Ren et al. proposed using a “Region Proposal Network” to replace the selective search algorithm, making the whole model much faster than fast R-CNN.
  - They call it the Faster R-CNN.

	Region Proposal	Object Detection
R-CNN	2s	47s
Fast R-CNN	2s	0.3s
Faster R-CNN	0.2s	

Faster R-CNN (3)

- The overall pipeline of Faster R-CNN is as follows:



Faster R-CNN (4)

- Why is Faster R-CNN faster than Fast R-CNN?
  - One reason the Region Proposal Network (RPN) works faster than Selective Search, is that Selective Search cannot be implemented on GPU, while RPN can.
  - Another reason is that the region proposal stage shares layer with the object detection stage.

Region Proposal Network (1)

- Let’s look at the RPN in details.
- We already knew that the spacial location of the image is preserved in the feature map when we perform convolution:

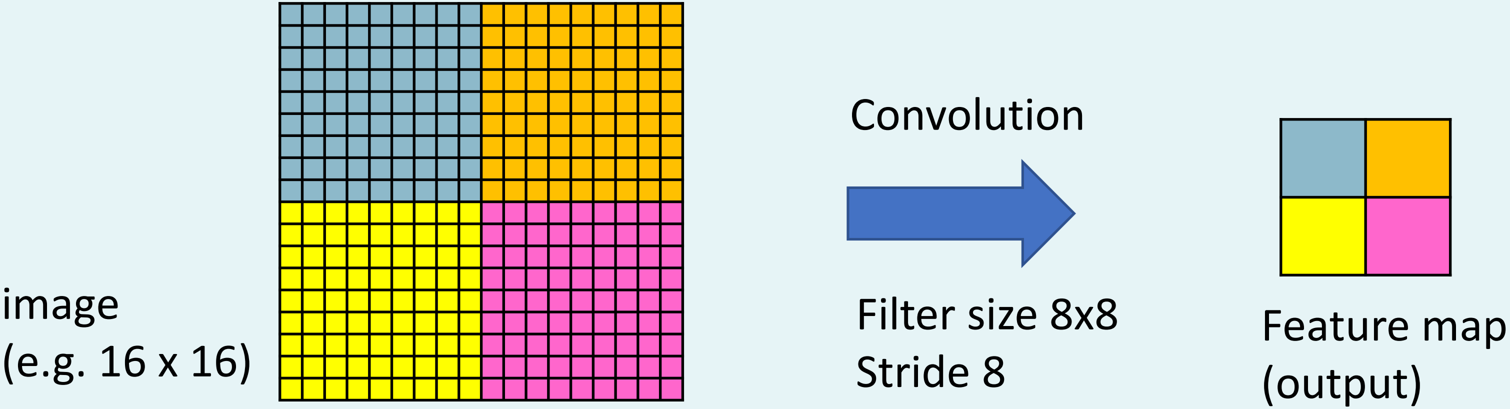


image
(e.g. 16 x 16)

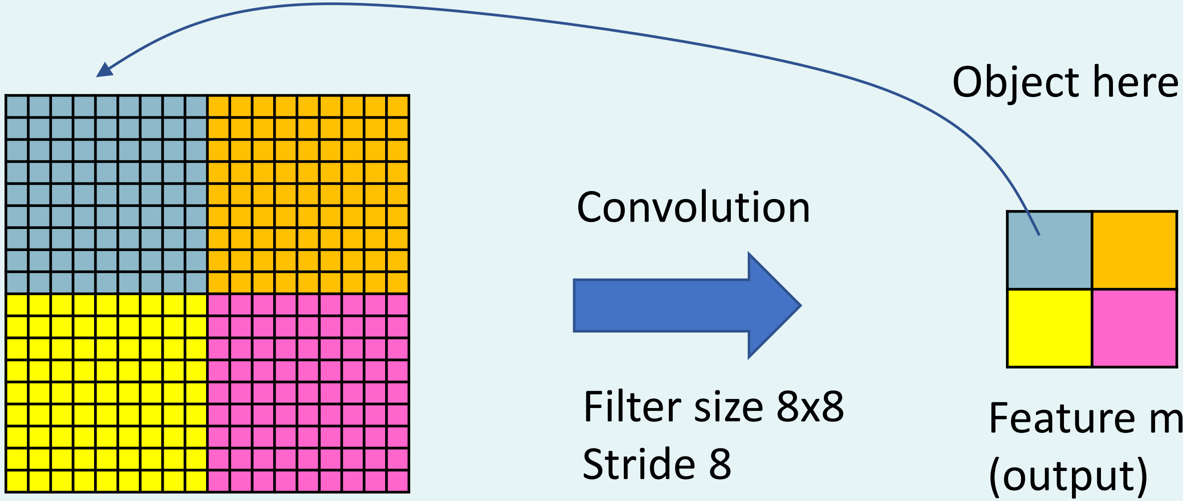
Convolution

Filter size 8x8
Stride 8

Feature map
(output)

Region Proposal Network (2)

- For each pixel in the feature map, we need to train the RPN to figure out whether there is an object in the original image location, and if yes, its location.



Object here?

Convolution

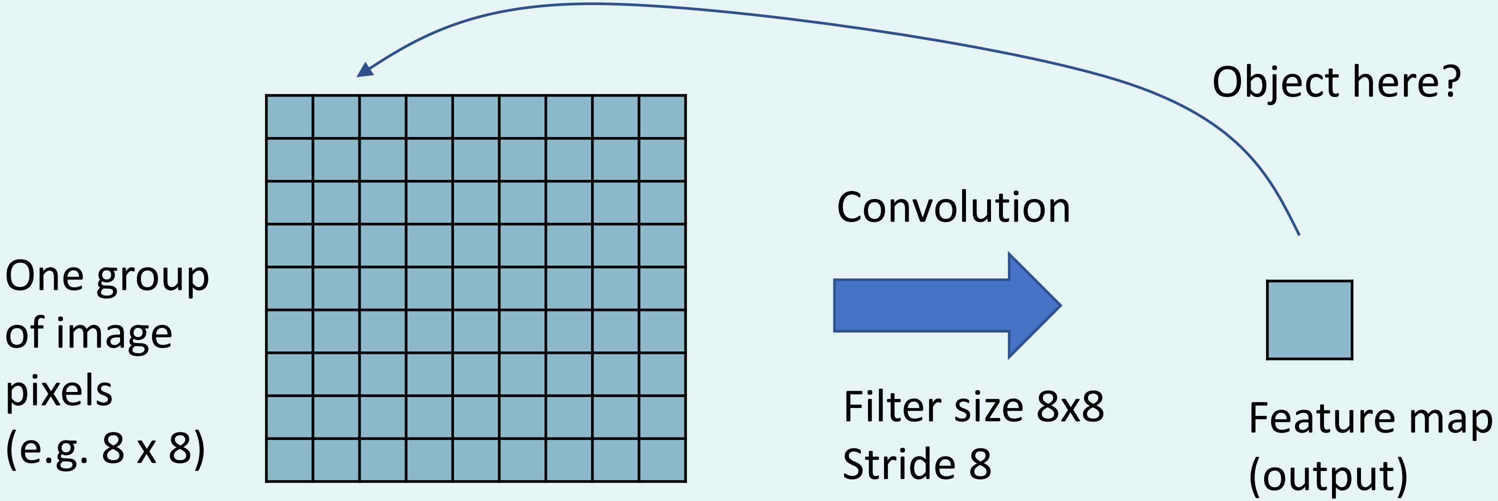
Filter size 8x8
Stride 8

Feature map
(output)

image
(e.g. 16 x 16)

Region Proposal Network (3)

- How should the target training label be?
- Let’s focus on the top left group of pixel:



Object here?

One group of image pixels (e.g. 8 x 8)

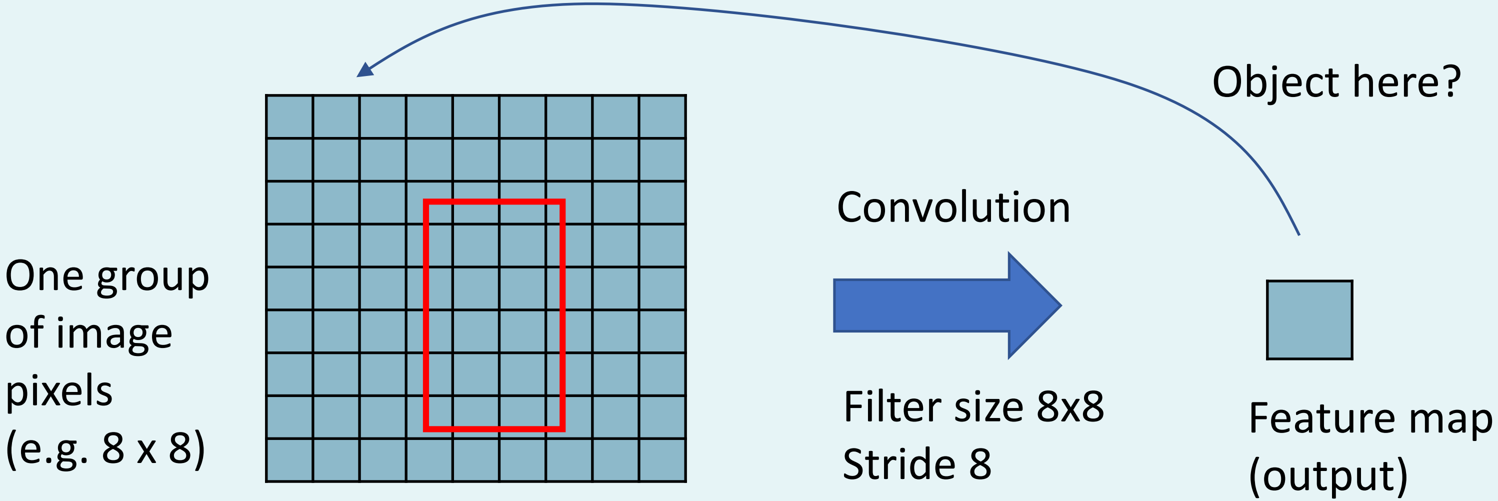
Convolution

Filter size 8x8
Stride 8

Feature map (output)

Region Proposal Network (4)

- Firstly, we need a ground truth (red box).



Object here?

One group of image pixels (e.g. 8 x 8)

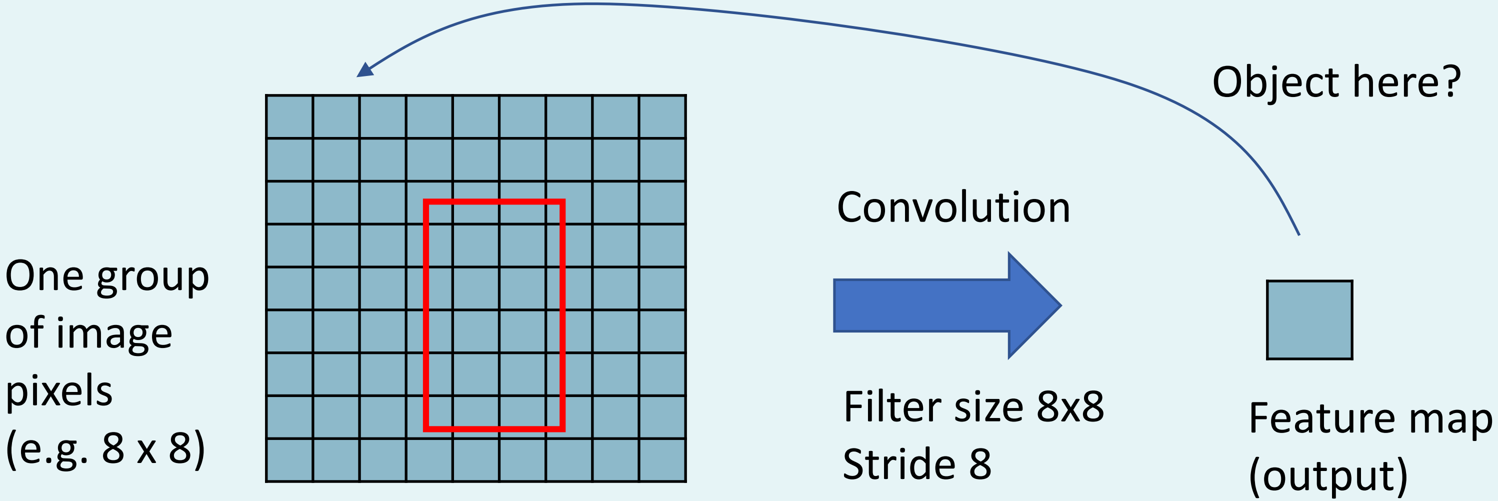
Convolution

Filter size 8x8
Stride 8

Feature map (output)

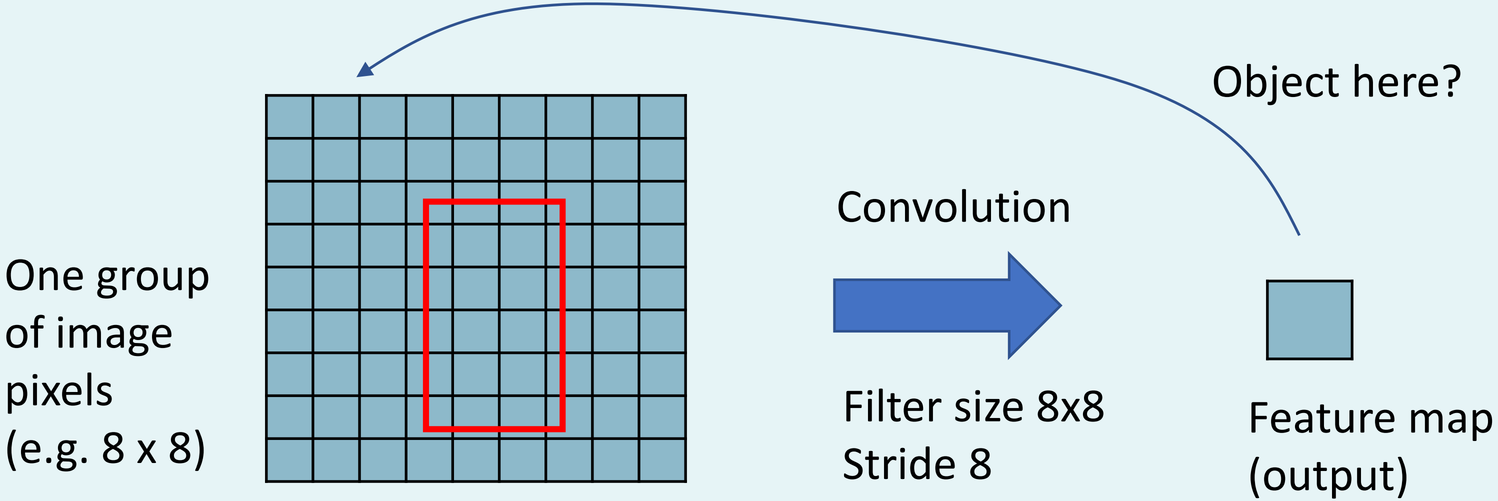
Region Proposal Network (5)

- It may be tempting to right away train the RPN to predict the bounding box (“absolute values”) of this ground truth.



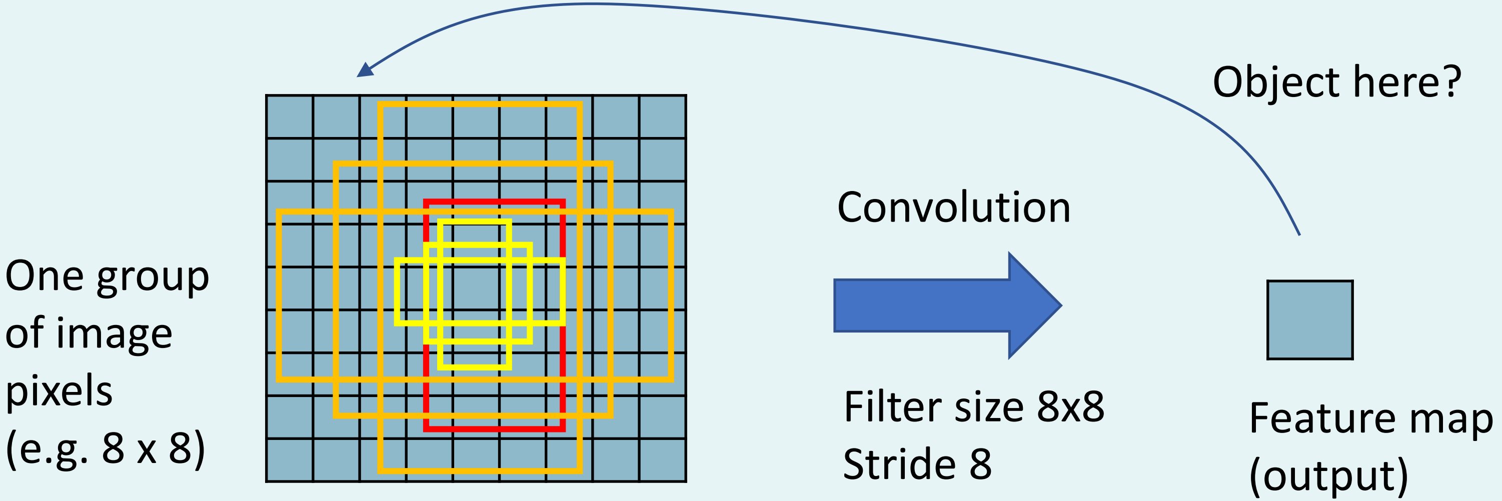
Region Proposal Network (6)

- It turns out that it is better to learn "small relative adjustments" from some known values instead.



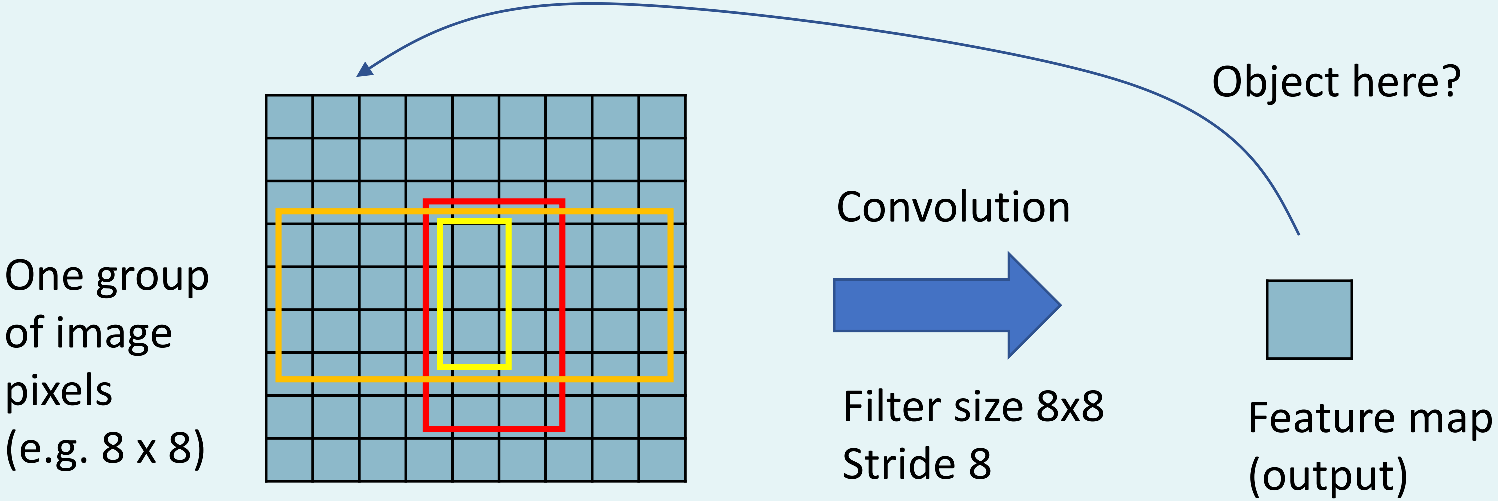
Region Proposal Network (7)

- Therefore, Faster R-CNN proposes a concept called “Anchor box” – a series of boxes with different sizes and aspect ratio, placed at the center of pixel groups.



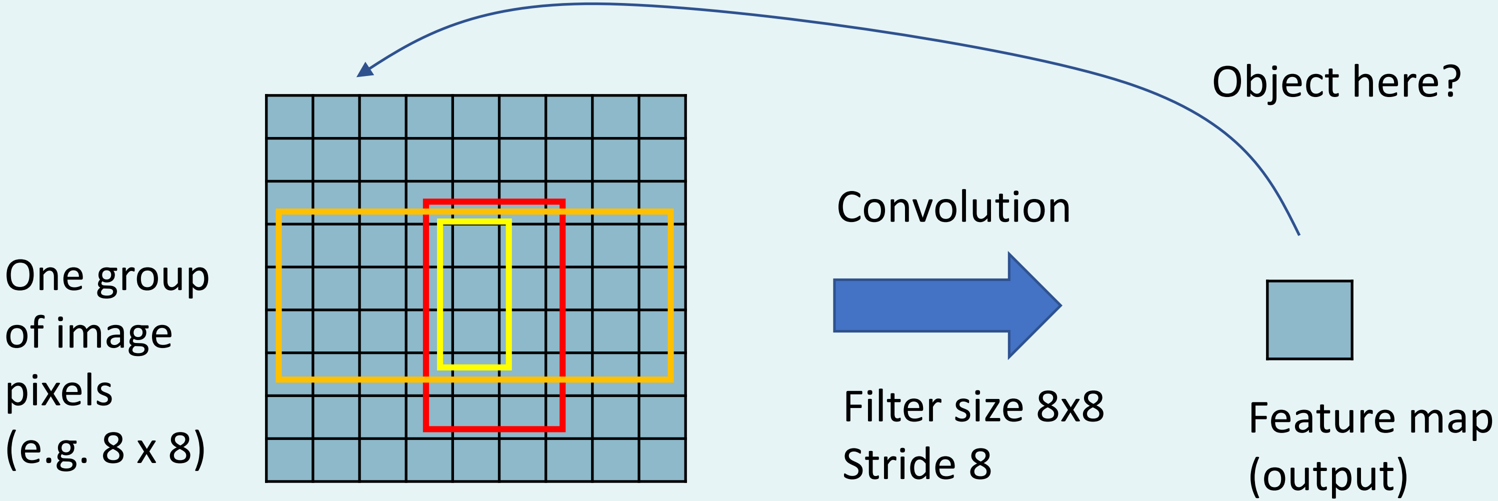
Region Proposal Network (8)

- We compare the IOU of each anchor box with GT:
  - IOU > 0.7 → Positive samples (object) – e.g. yellow anchor below
  - IOU < 0.3 → Negative samples (no object) – e.g. orange anchor
  - Disregard if IOU between 0.3 and 0.7



Region Proposal Network (9)

- For every image, there will be far more negative samples than positive samples.
  - So we randomly choose the same number (e.g. 128) of positive samples and negative samples for training.



Region Proposal Network (10)

- At this stage, we have:
  - Some positive samples $(p_i^* = 1)$ of anchor boxes, with known centers (x_a, y_a), widths (w_a) and heights (h_a).
  - Some negative samples $(p_i^* = 0)$ of anchor boxes.
  - The ground truth, with known center (x^*, y^*), width (w^*) and height (h^*).
- We calculate the target shifts from anchor box to ground truth:

$$\boxed{t_x^* = \frac{x^* - x_a}{w_a}}, \boxed{t_y^* = \frac{y^* - y_a}{y_a}}, \boxed{t_w^* = \log\left(\frac{w^*}{w_a}\right)}, \boxed{t_h^* = \log\left(\frac{h^*}{h_a}\right)}$$

Region Proposal Network (11)

- So with the class label (p_i^*) and the target shifts (t_i^*) , we can train the RPN by minimizing the following loss function:

$$\boxed{L = \frac{1}{N_{class}} \sum_i L_{class}(p_i, p_i^*) + \frac{\lambda}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)}$$

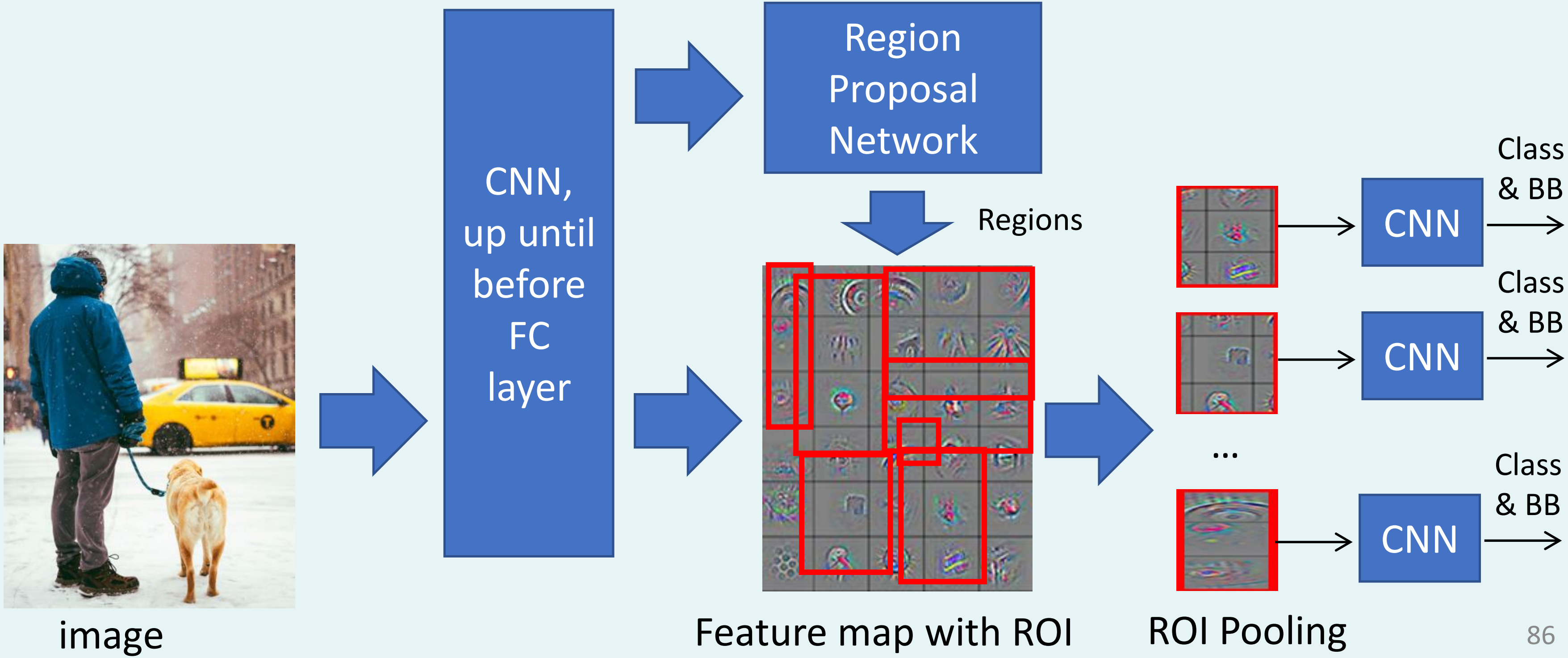
  - $L_{class}(p_i, p_i^*)$ is the classification loss, for e.g. binary cross-entropy.
  - The regression loss for bounding box is active only if $p_i^* = 1$.

Region Proposal Network (12)

- After training, the RPN will be able to predict if an anchor box contains an object (p_i), as well as the shift (t_i) from anchor box to region proposal box.
  - From the p_i values, typically choose top 300 as region proposals.

Faster R-CNN Summary (1)

- The overall pipeline of Faster R-CNN is as follows:



Faster R-CNN Summary (2)

- The region proposal network outputs:
  - Anchor box is object / not object.
  - Shift from anchor box to the region proposal box.

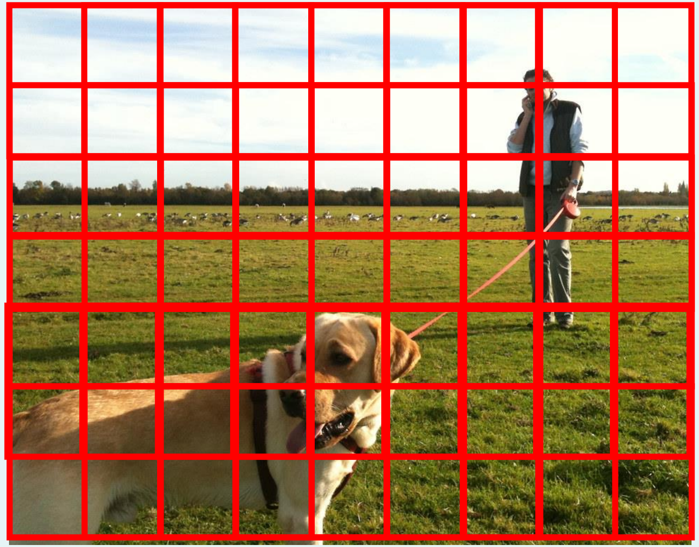
- The per-region network outputs:
  - The class of an object, including background.
  - Transform from region proposal box to bounding box.

Content

- Introduction to Object Detection
- Detecting Single Object
- Detecting Multiple Objects
  - Sliding Window
  - Region Proposal Methods
  - YOLO Algorithm

Recap (1)

- The sliding window method has high computational cost, because we need to step windows of different sizes across the entire image.



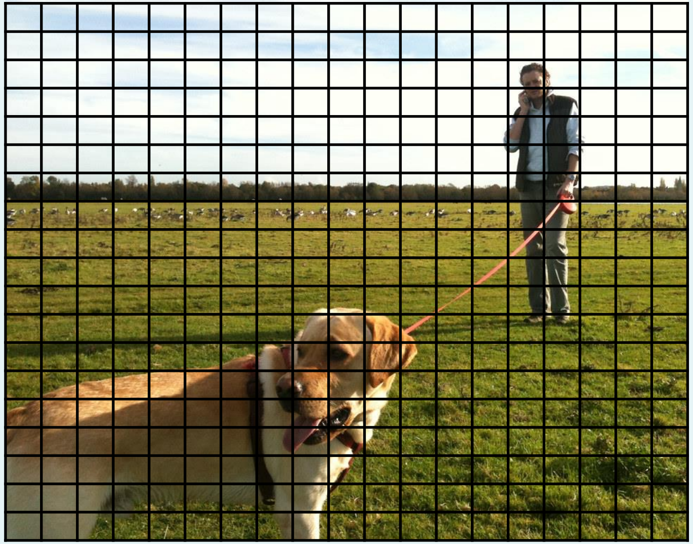
This Photo by Unknown Author is licensed under CC BY-SA-NC

Recap (2)

- The region proposal method reduced the computational cost by selecting a smaller number of regions with higher probability of containing an object.
- Here, we will learn another method of avoiding stepping through windows of different sizes.

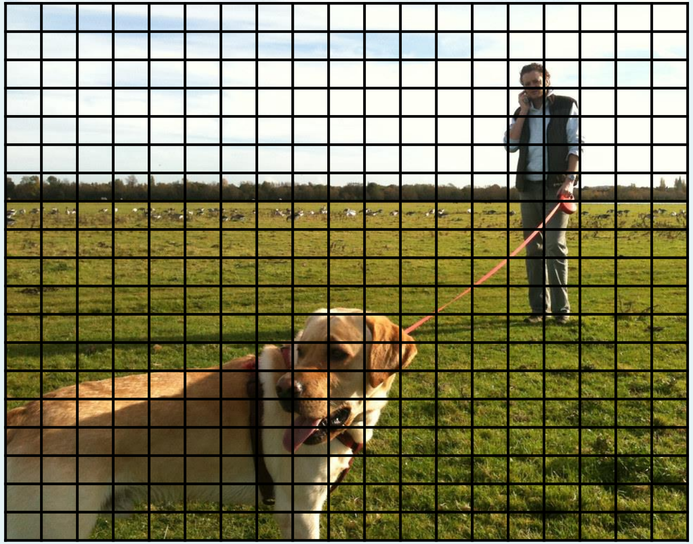
Basic Idea (1)

- The idea here is to just fix a certain number of cells, instead of trying out different windows.
- Commonly used number of cells: 19 x 19.
- Then detect object in each cell using CNN.



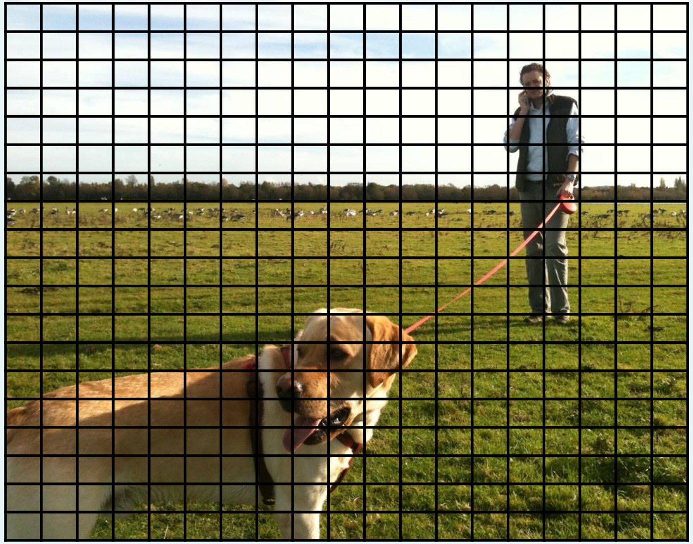
Basic Idea (2)

- But there are several questions:
  - Wouldn’t CNN need to run once for each cell? For e.g. 19 x 19 cells, that would still mean running CNN for 361 times! – Still high computational cost.
  - How should we deal with objects which span across several cells?



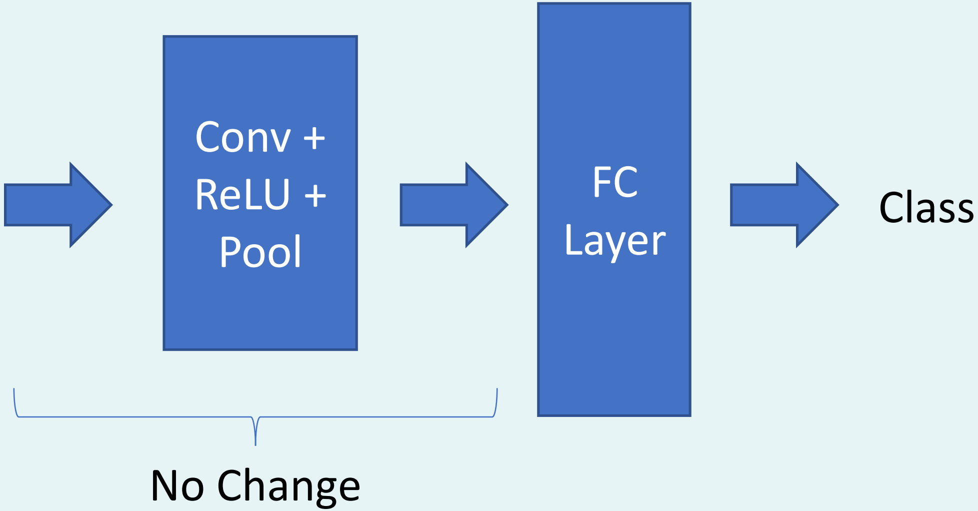
CNN for Each Cell? (1)

- To answer the first question:
  - Rather than running CNN 361 times, each time for one cell and with one output vector,
  - We create one single CNN for the entire image, but with 361 output vectors instead!



CNN for Each Cell? (2)

- The Conv-ReLU-Pool layers of the CNN is exactly the same as what you have learnt before, for classification.



Conv + ReLU + Pool

FC Layer

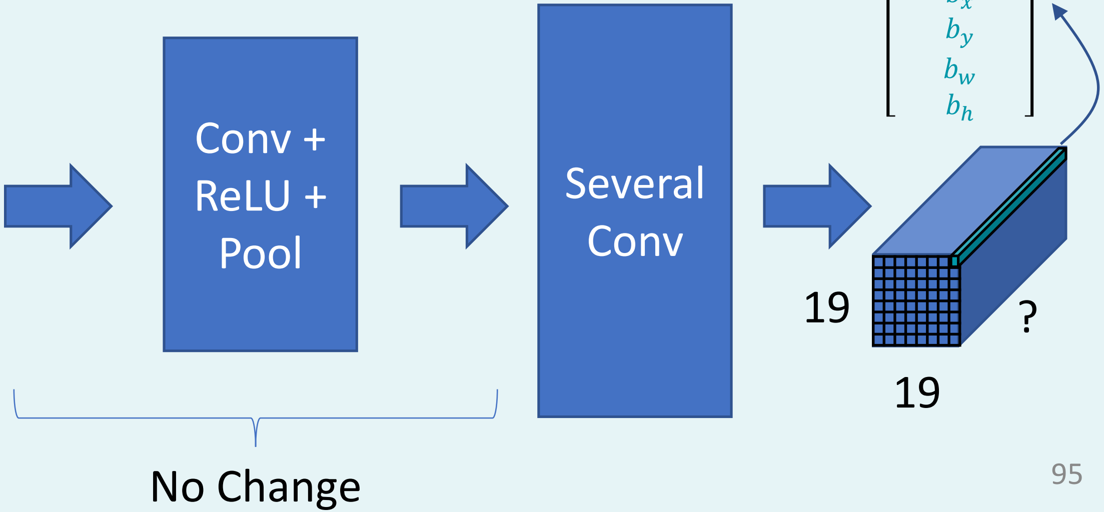
Class

No Change

CNN for Each Cell? (3)

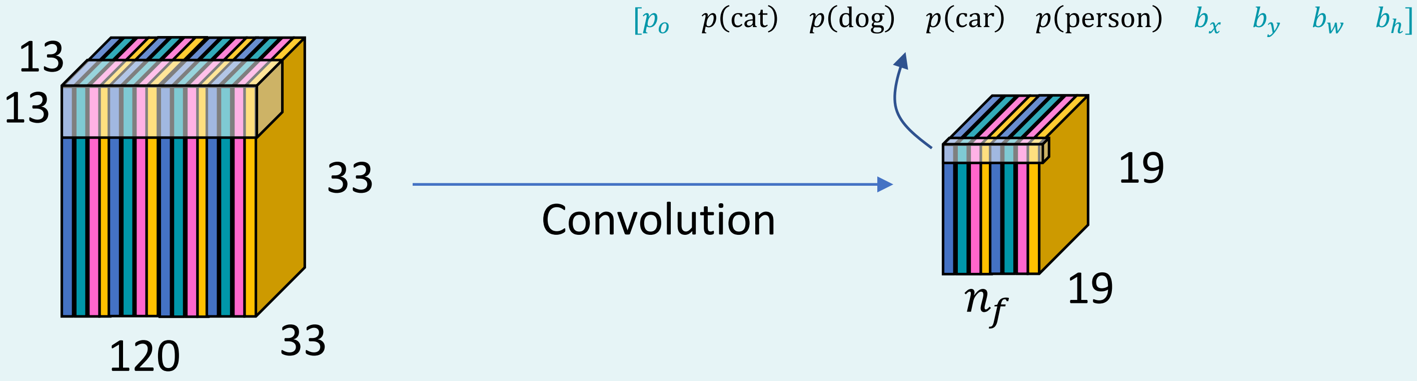
- But we replace the FC layer(s) with several convolution, such that the output has the required dimension.
  - Each “stripe” then represents one cell, with the outputs being for e.g.

$$\begin{bmatrix} p_o \\ p(\text{cat}) \\ p(\text{dog}) \\ p(\text{car}) \\ p(\text{person}) \\ b_x \\ b_y \\ b_w \\ b_h \end{bmatrix}$$



CNN for Each Cell? (4)

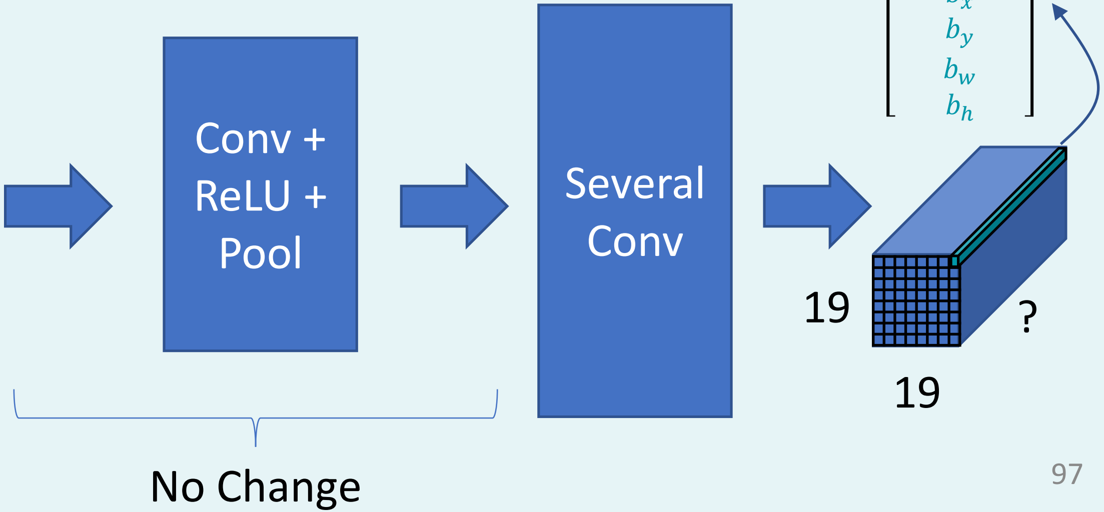
- E.g. at the end of Conv+ReLU+Pool, the matrix is of size 120 x 33 x 33.
  - Convolve with n_f filters of size (120 x 13 x 13), no padding, stride 1.
  - The output will be (n_f x 19 x 19).


13
13
33
33
120
Convolution
$[p_o \quad p(\text{cat}) \quad p(\text{dog}) \quad p(\text{car}) \quad p(\text{person}) \quad b_x \quad b_y \quad b_w \quad b_h]$
19
19
n_f

Target Labels? (1)

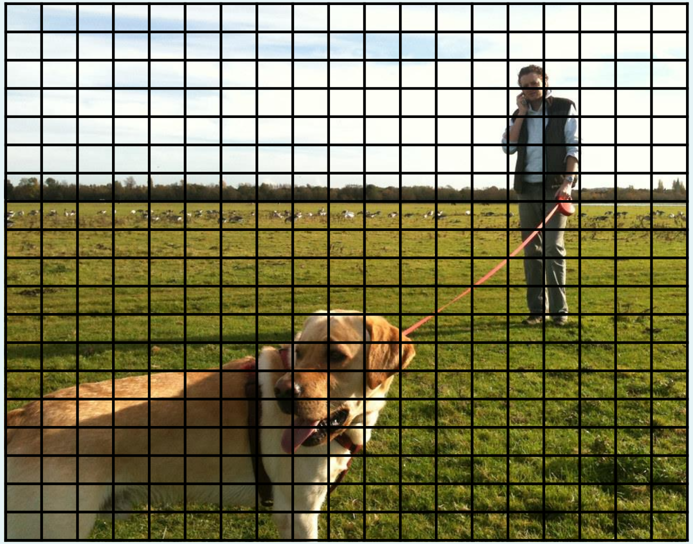
- It should be quite obvious now that to train the CNN, the target labels will be 19 x 19 of the output vectors.

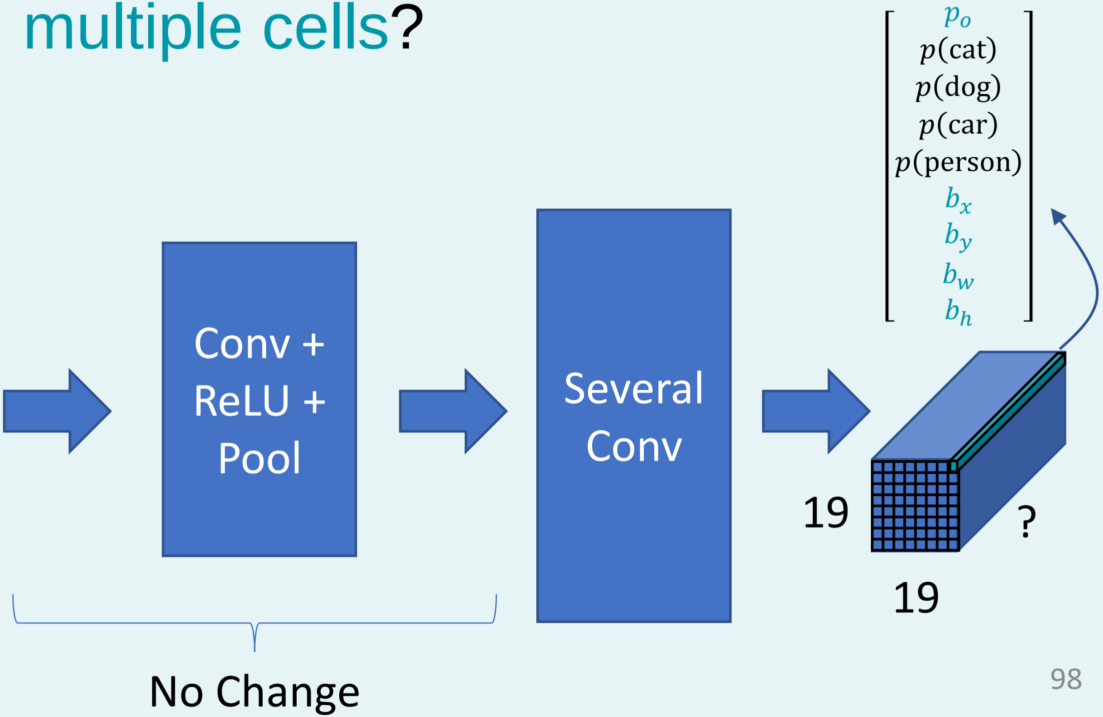
$$\begin{bmatrix} p_o \\ p(\text{cat}) \\ p(\text{dog}) \\ p(\text{car}) \\ p(\text{person}) \\ b_x \\ b_y \\ b_w \\ b_h \end{bmatrix}$$



Target Labels? (2)

- But here comes our 2nd question: How should we deal with objects spanning multiple cells?

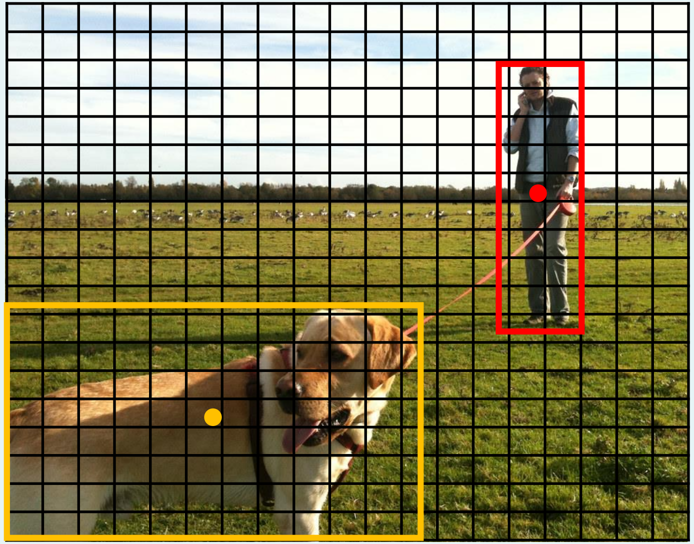




$$\begin{bmatrix} p_o \\ p(\text{cat}) \\ p(\text{dog}) \\ p(\text{car}) \\ p(\text{person}) \\ b_x \\ b_y \\ b_w \\ b_h \end{bmatrix}$$

Target Labels? (3)

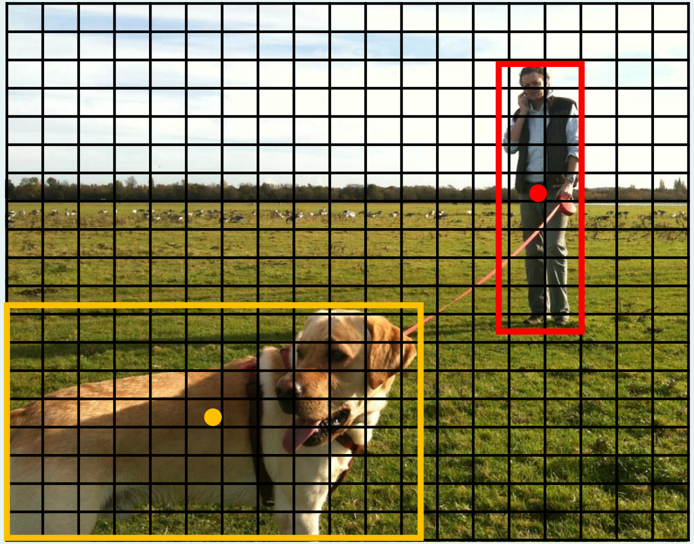
- To handle this situation, the training label is set as follows: Use the cell containing the center of bounding box as the “correct” cell.



Target Labels? (4)

- For the "correct" cells, the target labels will be:

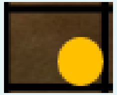
$$\begin{bmatrix} p_o \\ p(\text{cat}) \\ p(\text{dog}) \\ p(\text{car}) \\ p(\text{person}) \\ b_x \\ b_y \\ b_w \\ b_h \end{bmatrix}$$



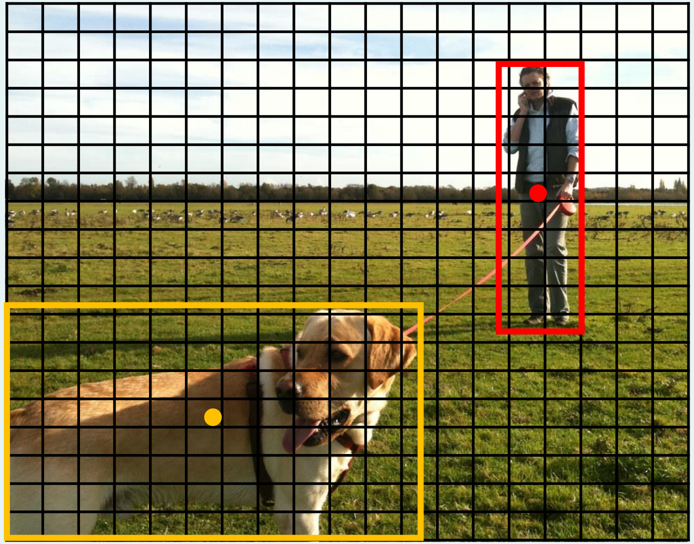
  - Where b_x, b_y are between 0 and 1, with respect to the chosen cell.
  - b_w, b_h can be more than 1!

Target Labels? (5)

- E.g.

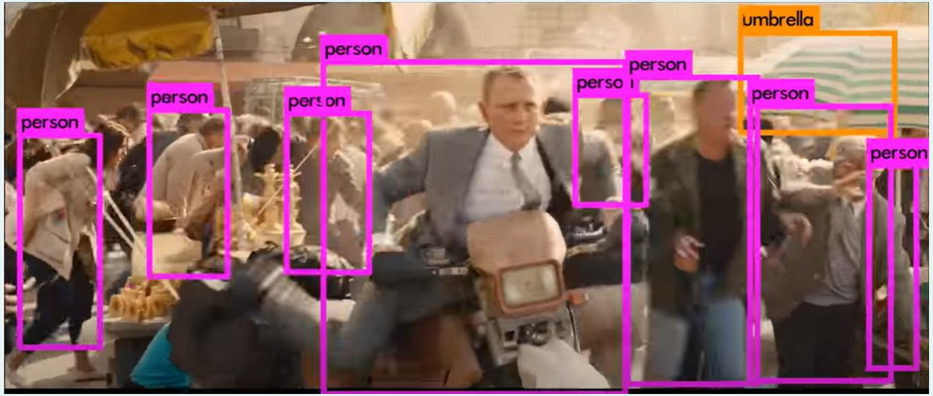


$$\begin{bmatrix} p_o \\ p(\text{cat}) \\ p(\text{dog}) \\ p(\text{car}) \\ p(\text{person}) \\ b_x \\ b_y \\ b_w \\ b_h \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0.8 \\ 0.9 \\ 2.2 \\ 9.5 \end{bmatrix}, \begin{bmatrix} p_o \\ p(\text{cat}) \\ p(\text{dog}) \\ p(\text{car}) \\ p(\text{person}) \\ b_x \\ b_y \\ b_w \\ b_h \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0.8 \\ 0.8 \\ 11.5 \\ 8.2 \end{bmatrix}$$



YOLO is Fast

- YOLO is a single-stage method, as it doesn't require segmentation as in region-proposal methods.
- It runs very fast and can be used in real-time!
  - https://www.youtube.com/watch?v=VOC3huqHrss



Thank you for your attention!

Any questions?