Deep Learning in Computer Vision

June 12th, 2019

Janus Nørtoft Jensen, jnje@dtu.dk

Topics of today

- What is localisation and detection
- Object localisation
- Object detection with sliding windows
- Sliding windows done smartly

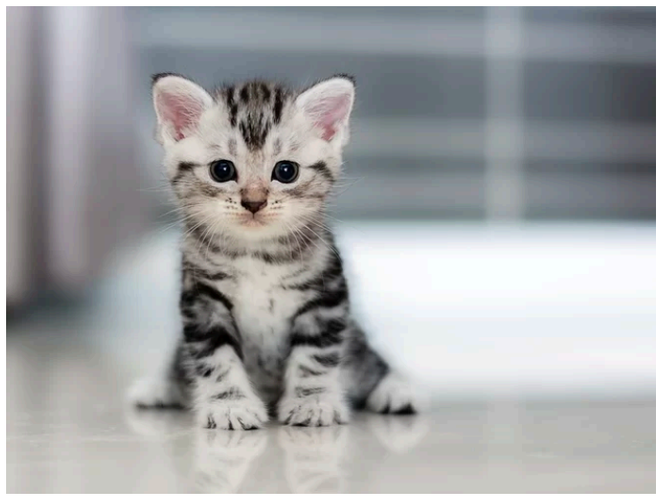
Object localisation

- Classification
- Classification with localisation
    - One object
- Detection
    - Classify and localize multiple objects

Object Localization

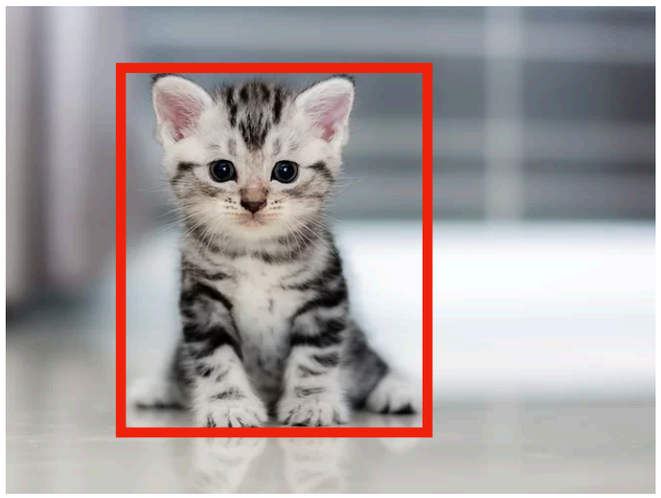
What are localisation and detection?

Image classification



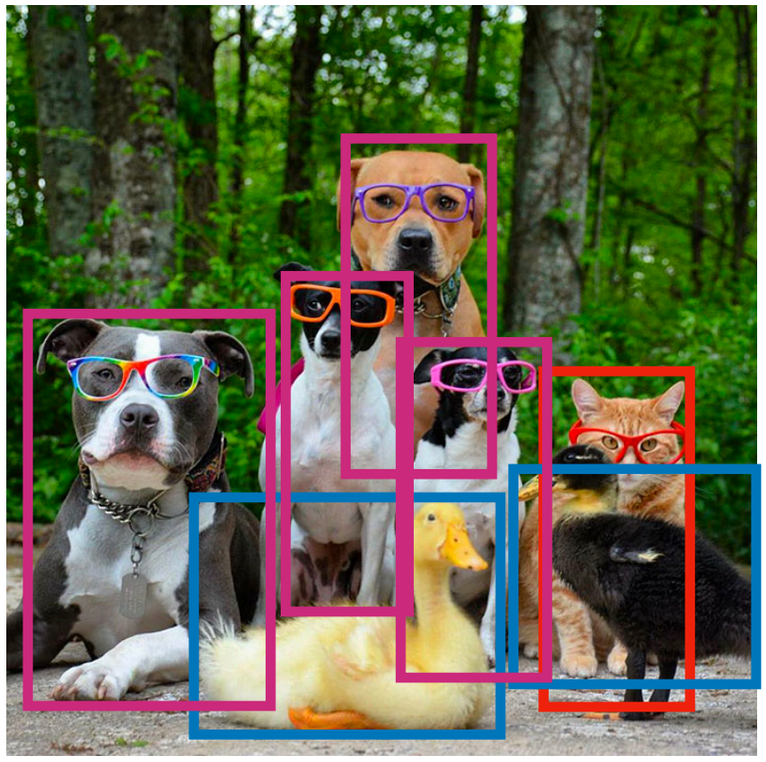
Class: Cat

Image classification + localisation



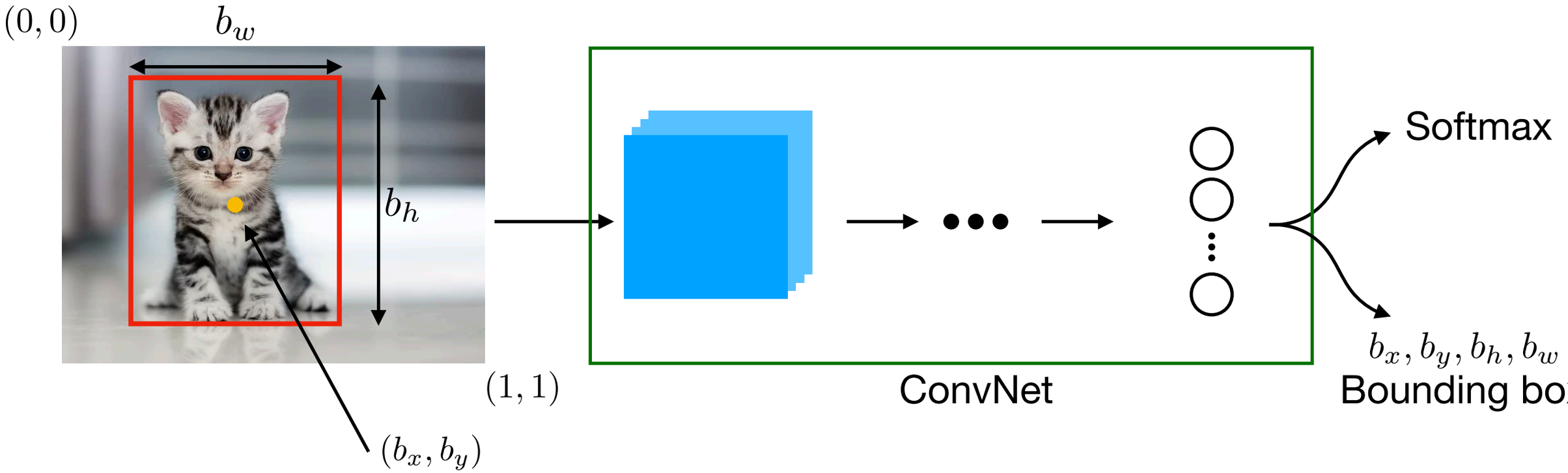
Class: Cat
+ Bounding Box

Detection



Multiple objects

Classification + Localization



Parrot

Dog

Cat

Duck

Background

$$b_x = 0.4$$
$$b_y = 0.45$$
$$b_h = 0.7$$
$$b_w = 0.5$$

Defining the target label

- We need to output: class label and b_x, b_y, b_h, b_w

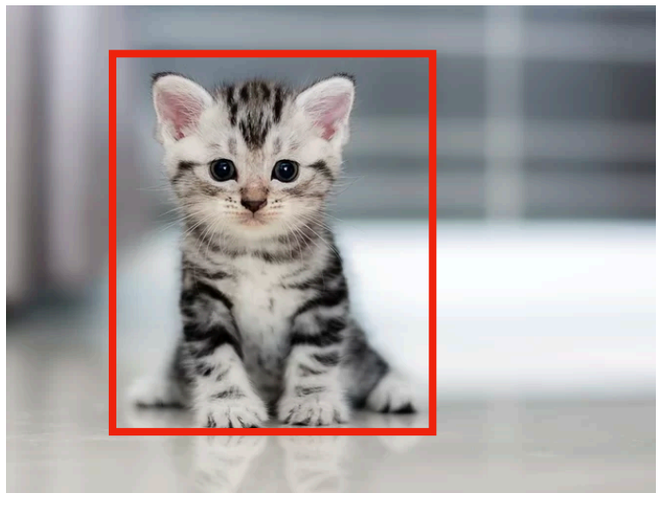
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

← Is there an object?

← Bounding box

← Which class?

Parrot
Dog
Cat
Duck
Background



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

Defining the target label

- We need to output: class label and b_x, b_y, b_h, b_w

Parrot
Dog
Cat
Duck
Background

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

p_c ← Is there an object?

b_x, b_y, b_h, b_w ← Bounding box

c_1, c_2, c_3, c_4 ← Which class?

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

Loss function

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

$$\mathcal{L}(y, \hat{y}) = \begin{cases} \sum_i (y_i - \hat{y}_i)^2 & \text{if } y_1 = 1 \\ (y_1 - \hat{y}_1)^2 & \text{if } y_1 = 0 \end{cases}$$

Sum of squared errors on the terms that matter

Loss function

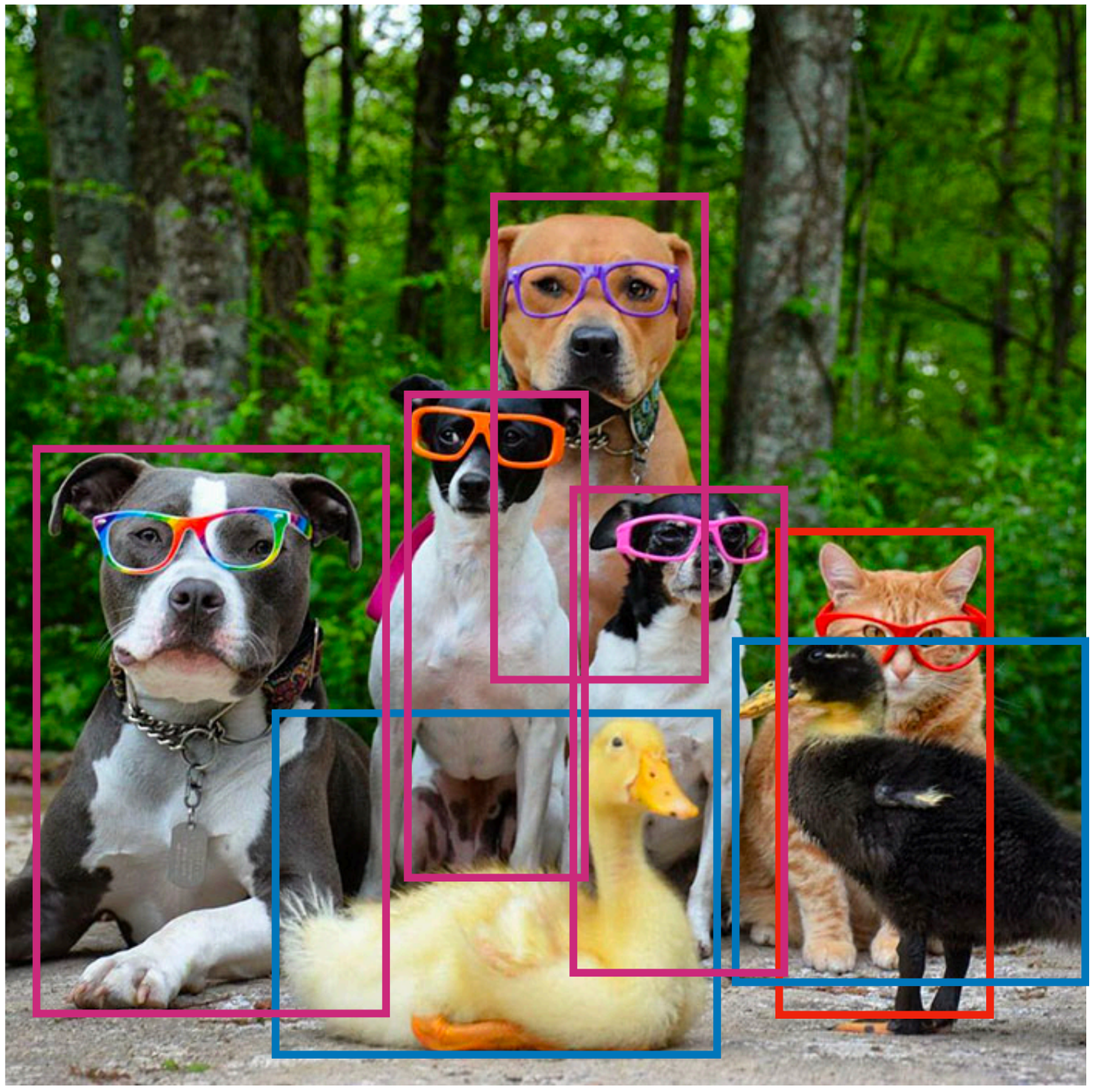
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

Sum of squared errors

Cross-entropy loss

Object Detection

Object detection

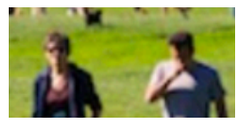
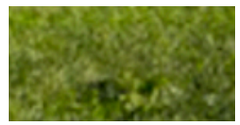


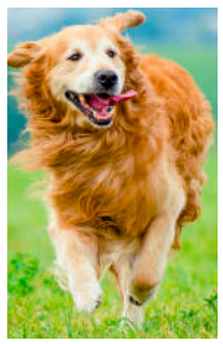
Object detection

- Train a network to classify closely cropped images
- Use the network to do object detection based on sliding windows
- Implement the sliding windows algorithm using convolutions

Object detection

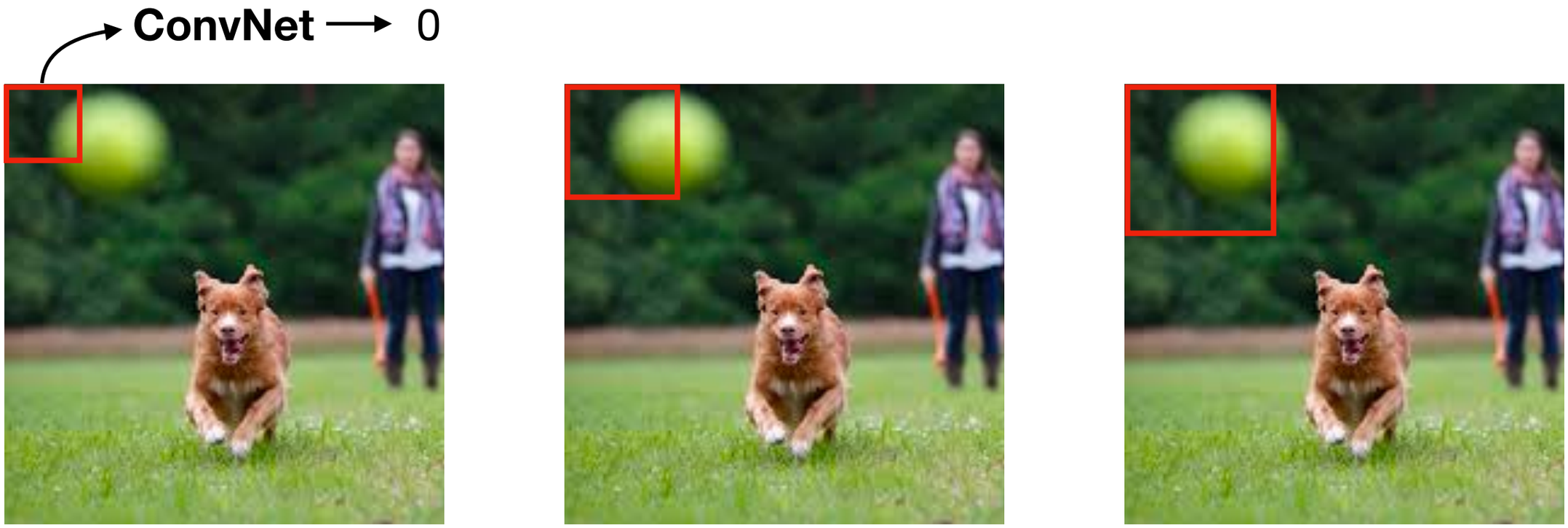
Training set

x	y
	1
	1
	0
	0



→ **ConvNet** → **y**

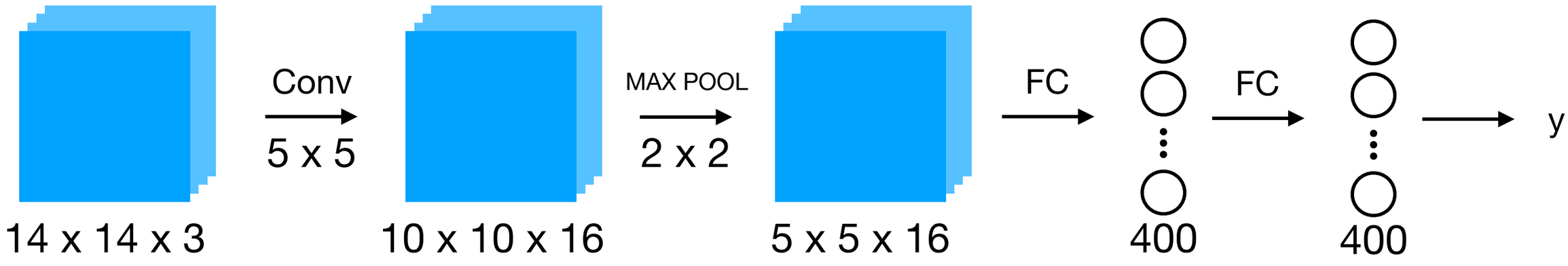
Sliding windows detection

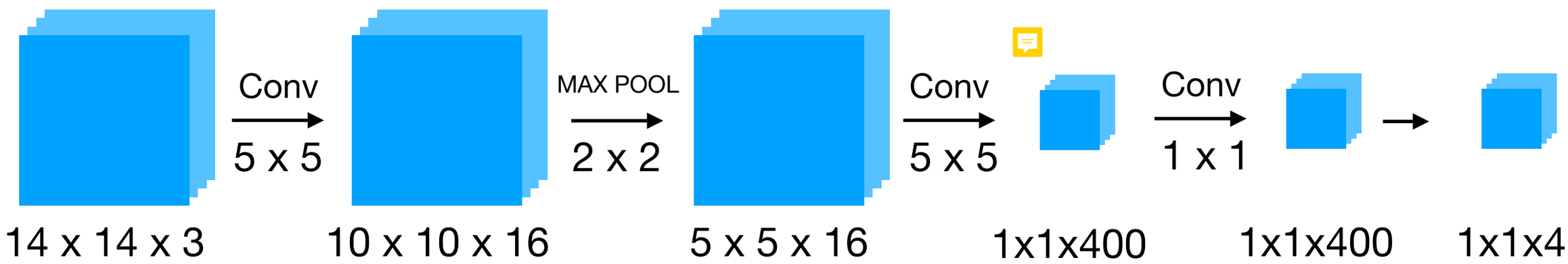


Very high computational cost!

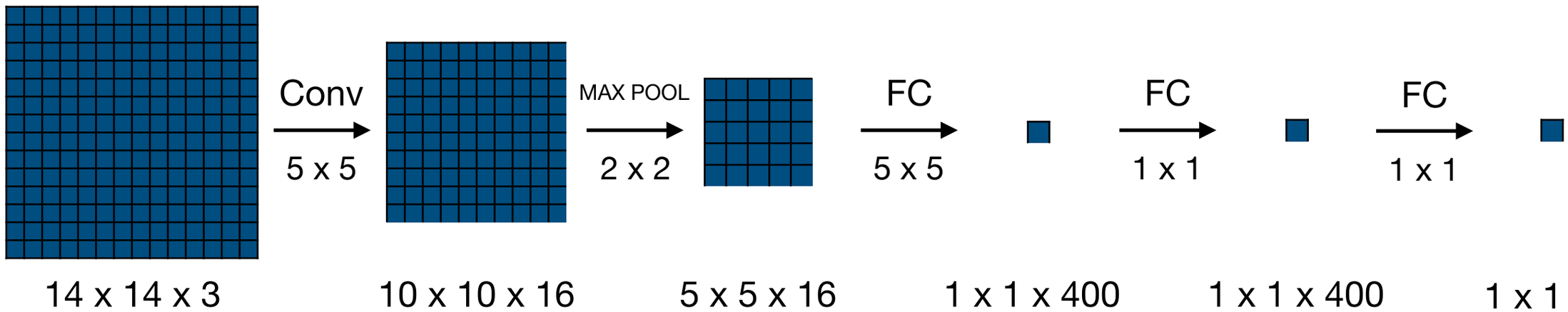
Convolutional implementation of sliding window

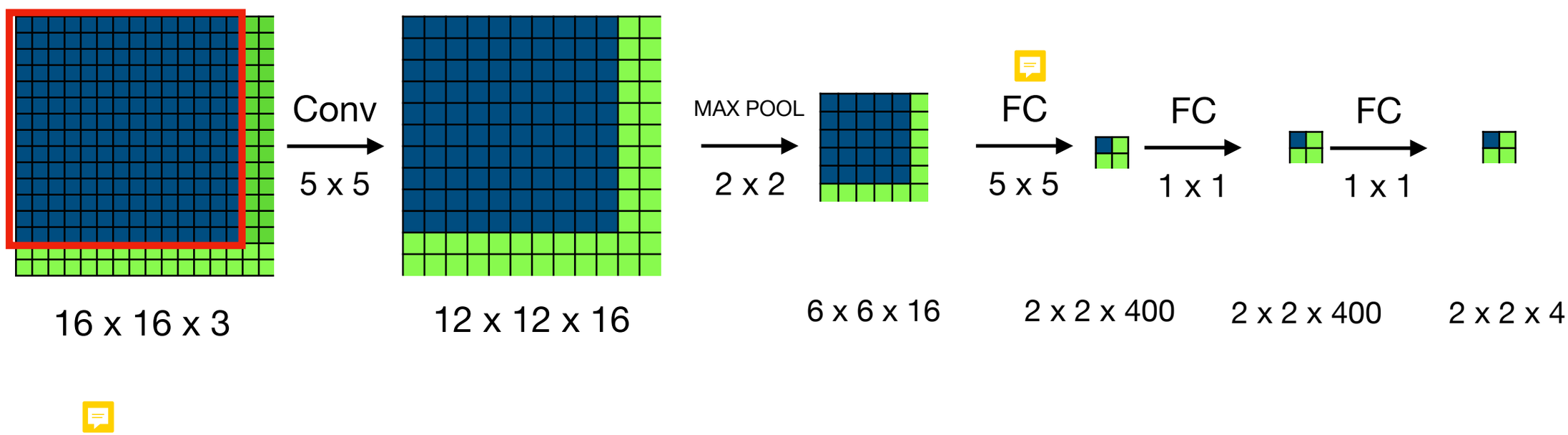
Fully connected layers as convolutional layers





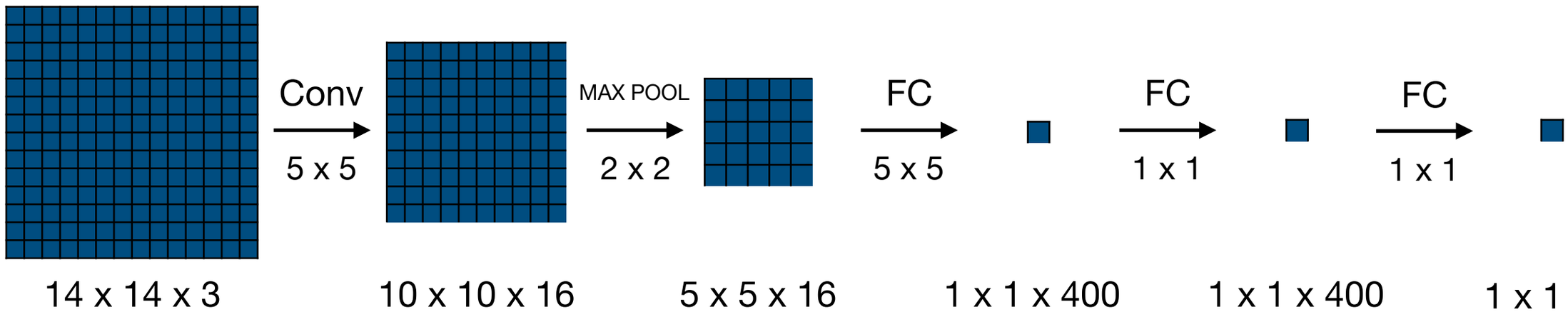
Convolutional implementation of sliding windows

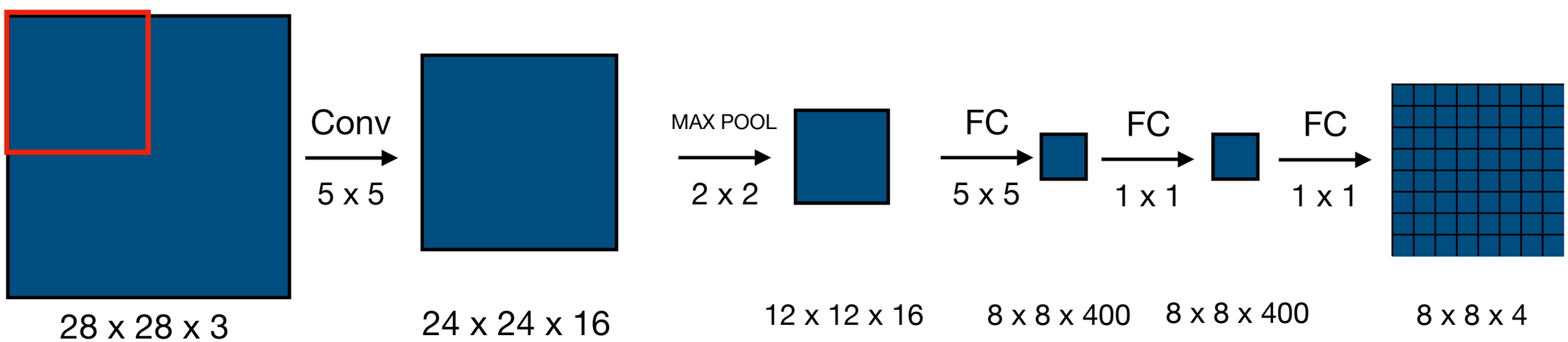




Sermanet et al., 2014, OverFeat: Integrated Recognition, localisation and detection using convolutional neural networks

Convolutional implementation of sliding windows



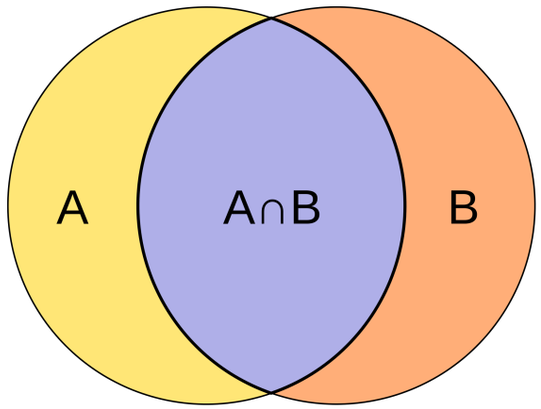


Sermanet et al., 2014, OverFeat: Integrated Recognition, localisation and detection using convolutional neural networks

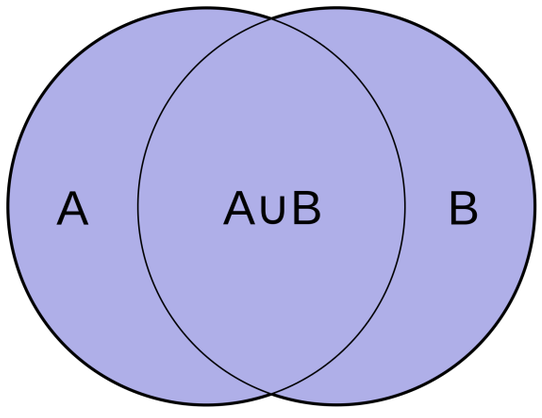
Intersection over Union

- Also known as the Jaccard index

$$IoU(A, B) = \frac{\text{Area}(A \cap B)}{\text{Area}(A \cup B)}$$



Intersection

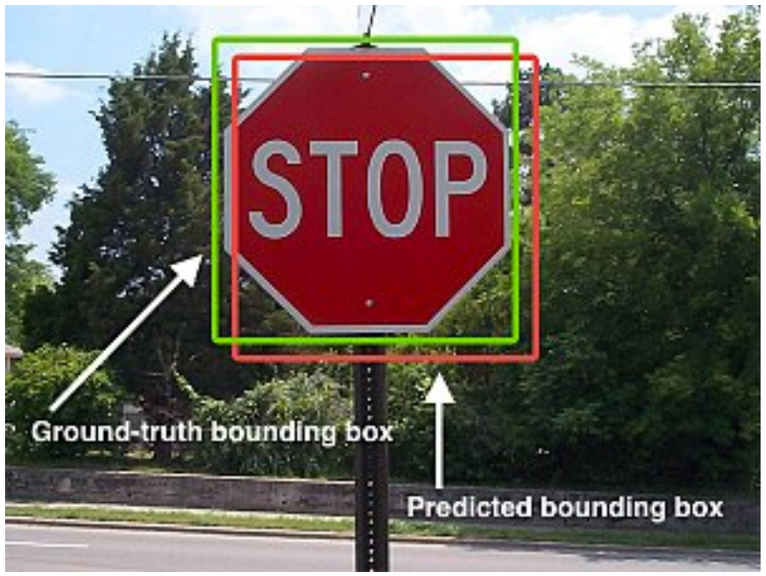


Union

Intersection over Union

- Also known as the Jaccard index

$$IoU(A, B) = \frac{\text{Area}(A \cap B)}{\text{Area}(A \cup B)}$$

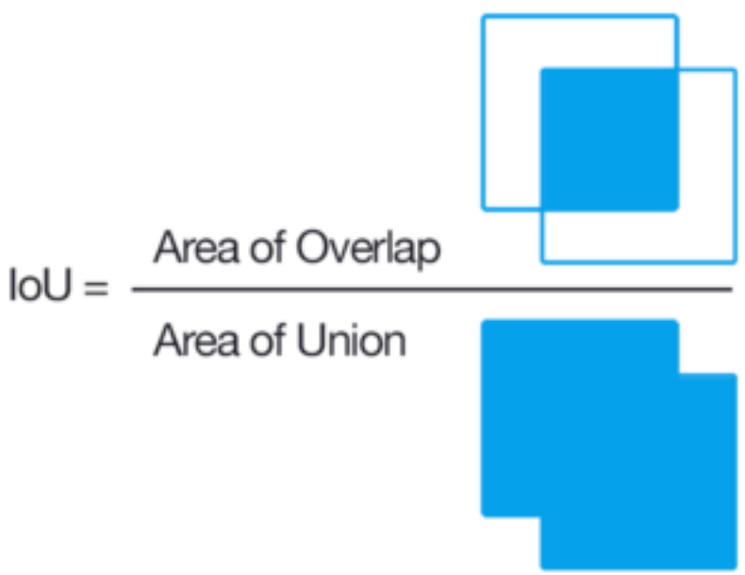


- An object is found correctly if

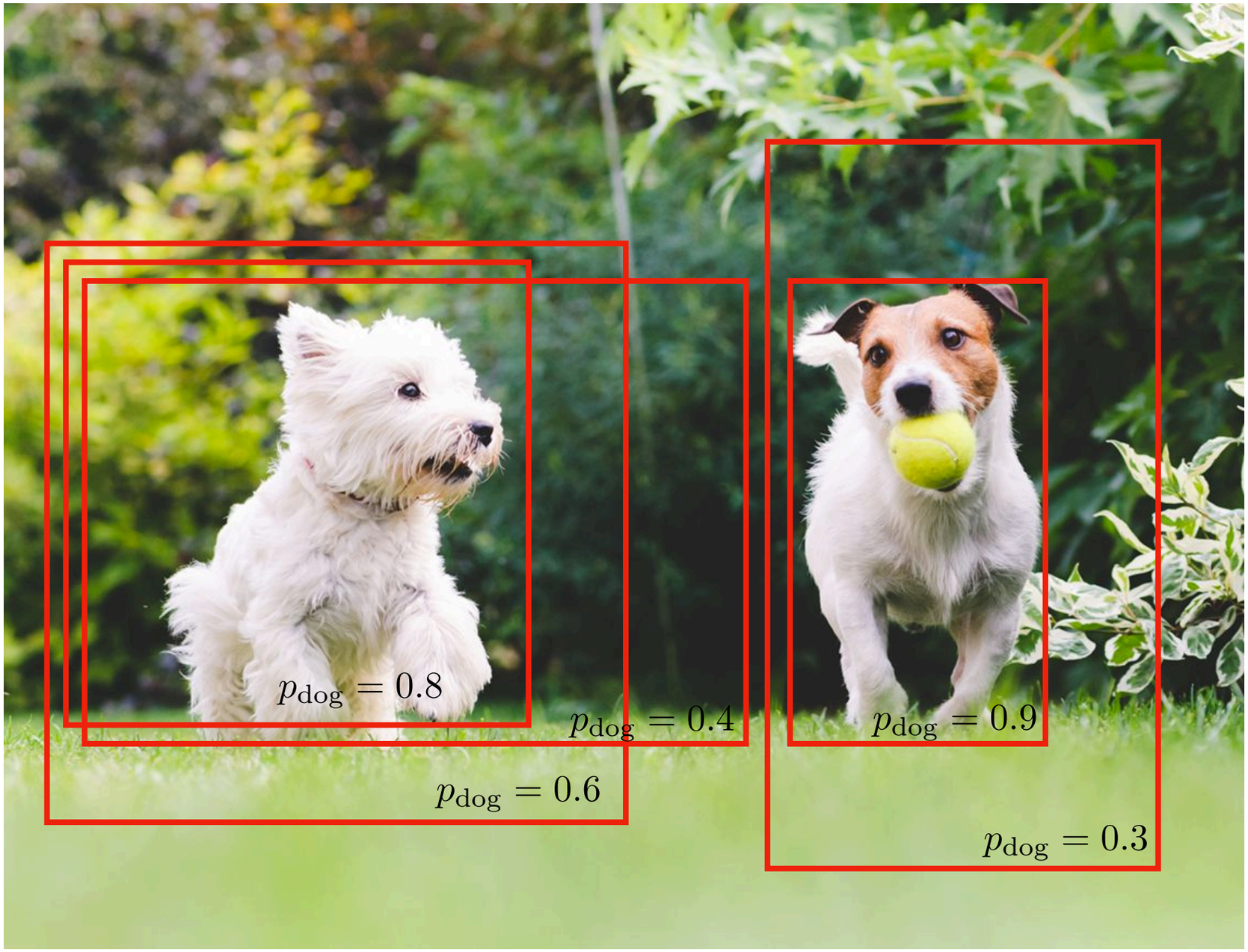
$$IoU(A, B) \geq T$$

- Where T is a threshold e.g.

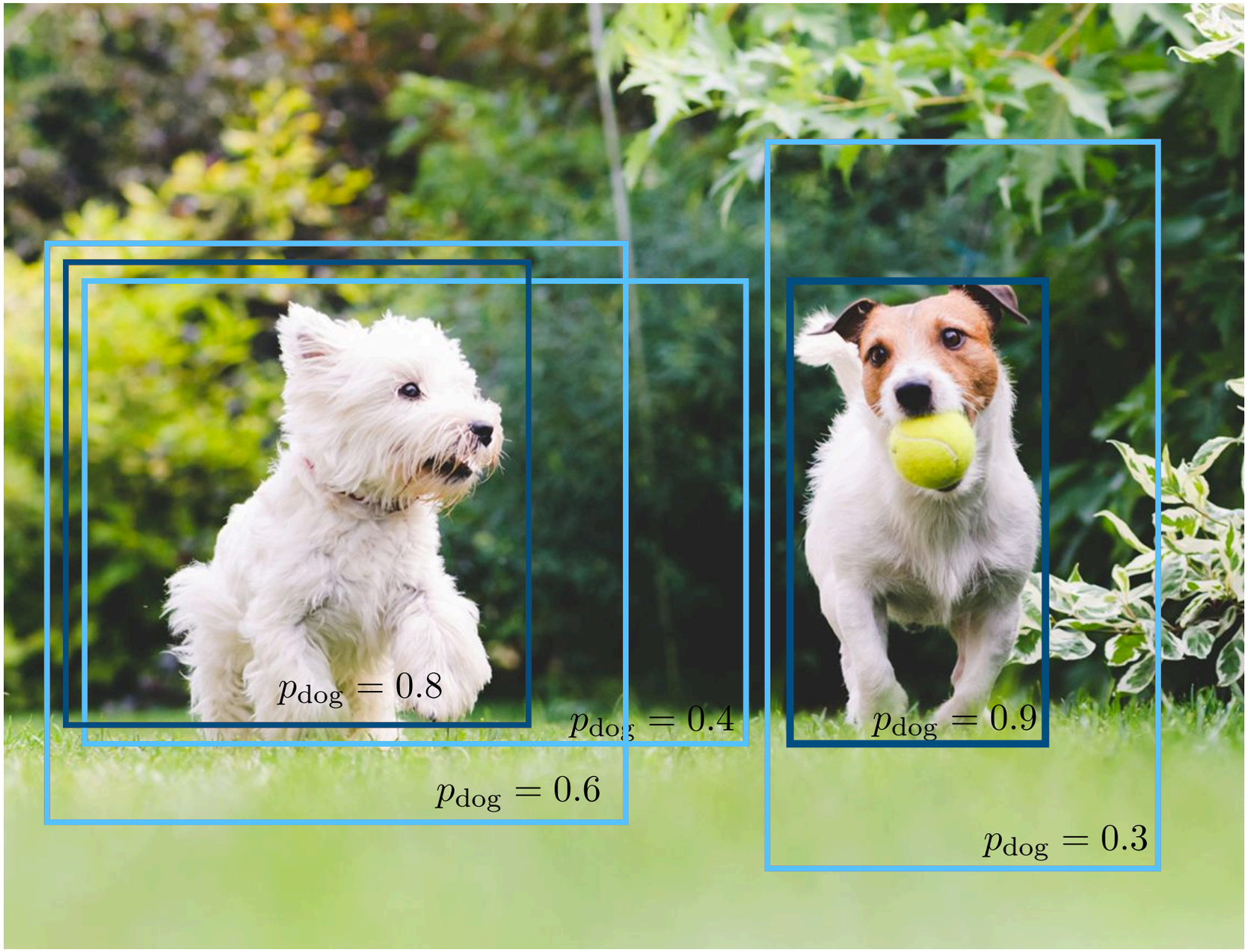
$$T = 0.5$$



Non-maximum suppression



Non-maximum suppression



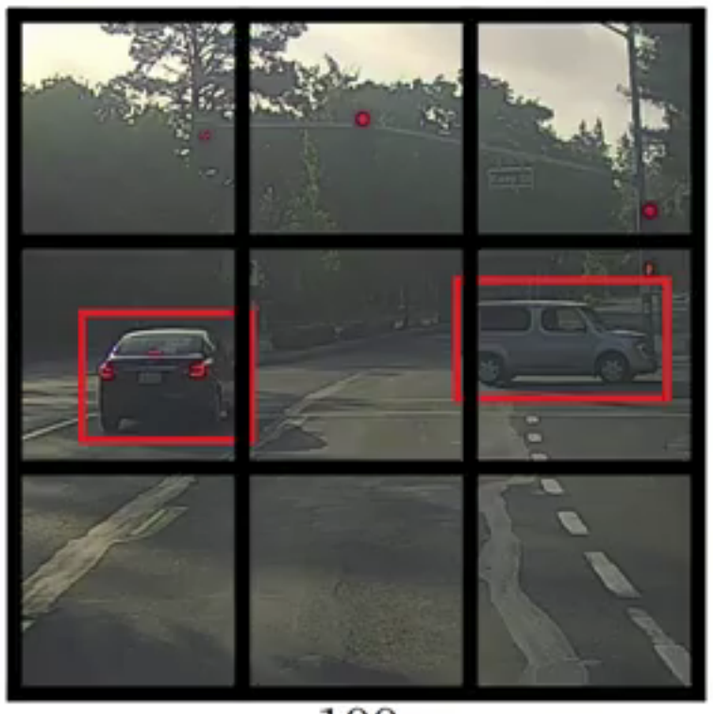
Non-maximum suppression

Non-maximum suppression

- Discard all boxes with $p_{\text{class}} \leq 0.6$
- For all classes:
  - While there are remaining boxes:
    - Pick the box with the largest p_{class}
    - Output as a prediction
  - Discard any boxes with $IoU \geq 0.5$ with the box from the previous step

YOLO

- You Only Look Once



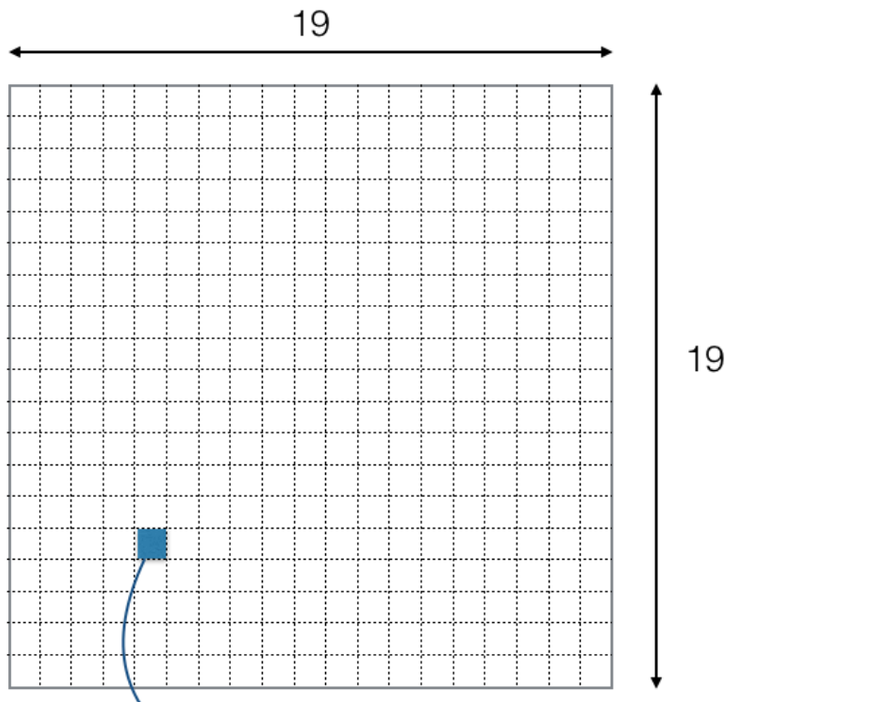
YOLO

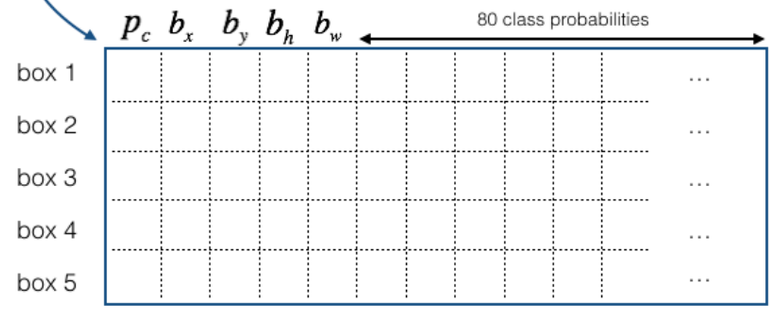
preprocessed image
(608, 608, 3)

Deep CNN

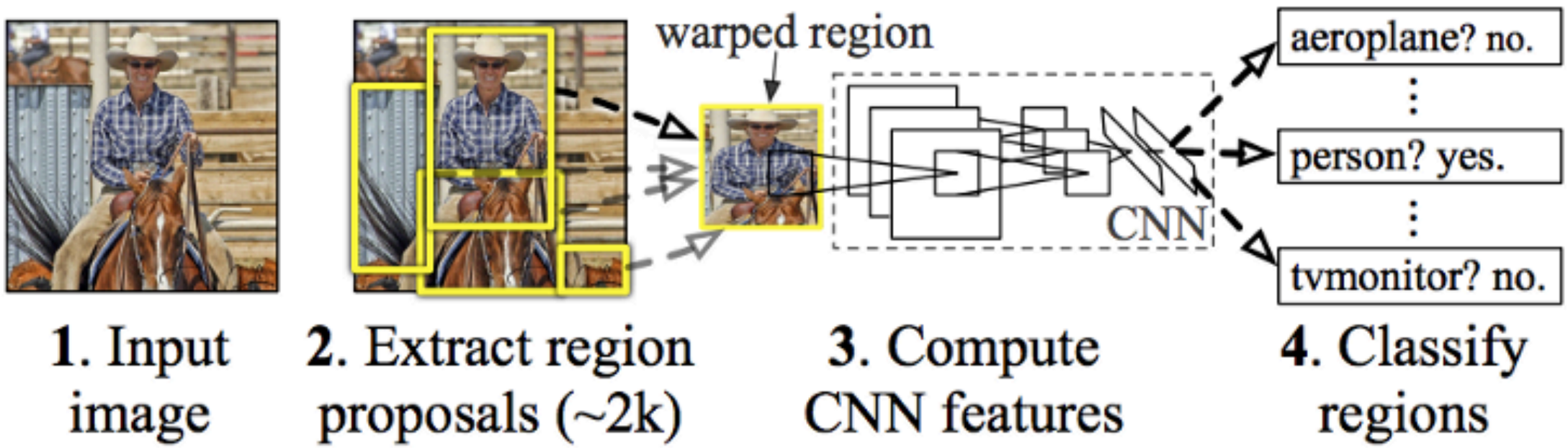
reduction
factor: 32

encoding
(19,19, 5, 85)

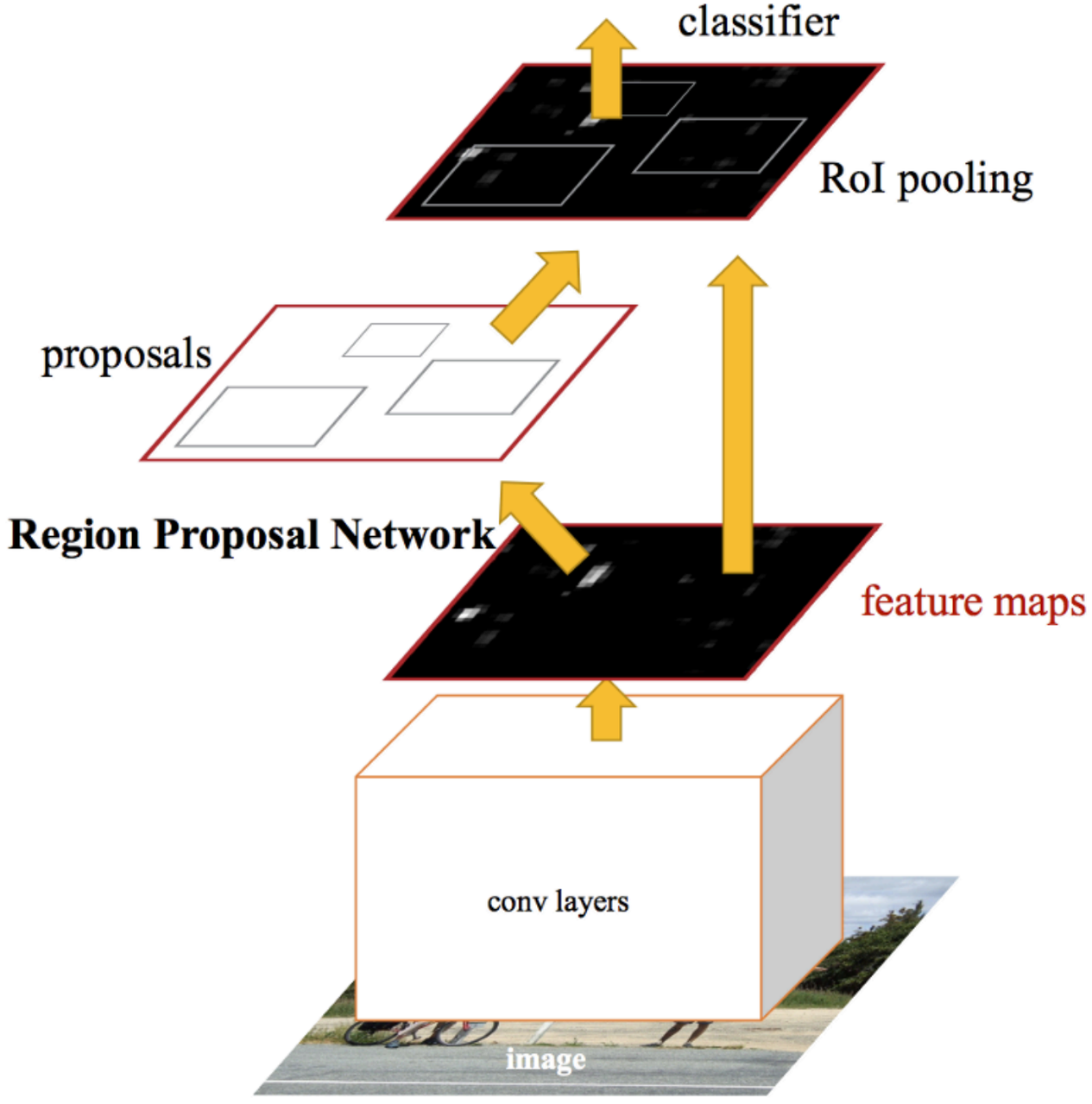




R-CNN


R-CNN: *Regions with CNN features*
warped region
aeroplane? no.
person? yes.
tvmonitor? no.
CNN
1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

Faster R-CNN


classifier
RoI pooling
proposals
Region Proposal Network
feature maps
conv layers
image