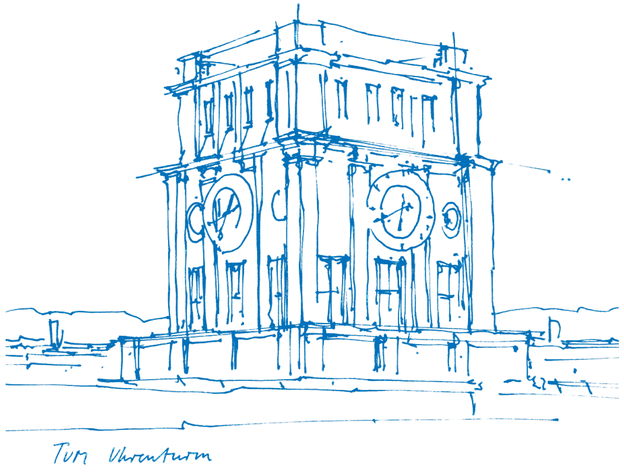
Computer Vision III:

Two-stage object detectors

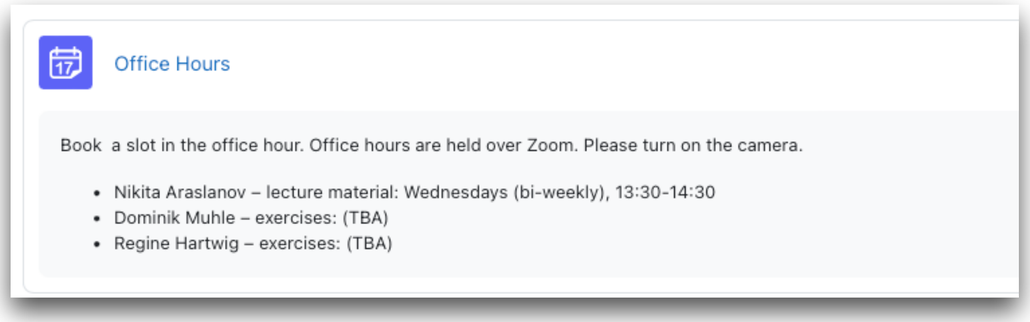
Dr. Nikita Araslanov
24.10.2023

Content credit:
Prof. Laura Leal-Taixé
https://dvl.in.tum.de



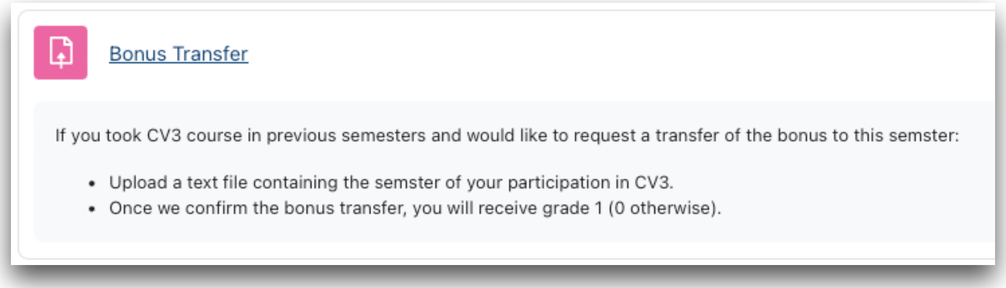
Announcements

- My office hours:
  - bi-weekly, Zoom, Wednesdays 13:30 (starting tomorrow);
  - over Zoom (please turn the camera on).
  - Please register in advance (use Moodle).



Announcements

- Bonus transfer – if you completed the assignments last year:
  - submit a text file over Moodle (see “Bonus Transfer”).
  - In the text file: the semester of participation in CV3
  - You’ll receive feedback/grade “1” if we confirm, “0” otherwise.



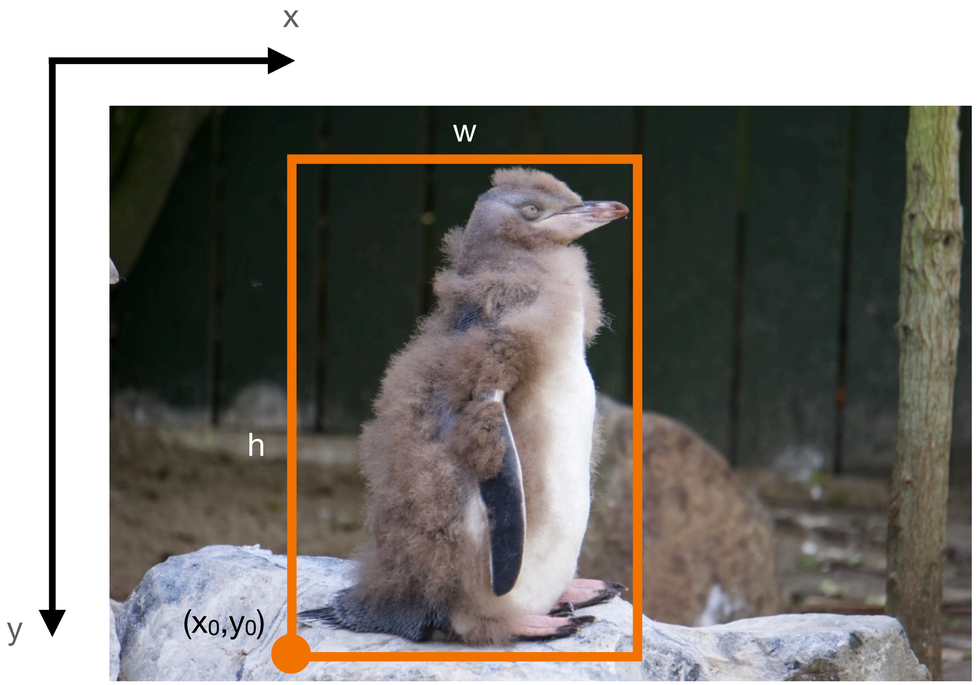
Announcements

- Quiz bonus – not this semester yet
  - no clear policy, probably hard to make fair.

- But...

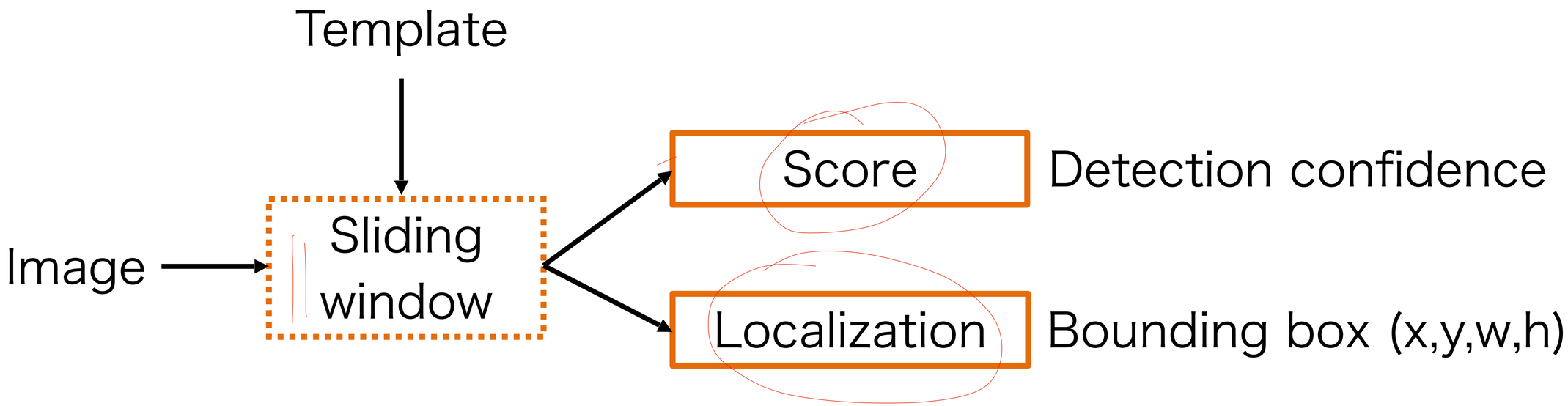
Recap: Last lecture

Object detection



Types of object detectors

- One-stage detectors (“old” times):



Template matching with sliding window

Template

Template matching with sliding window

For every position you measure the distance (or correlation) between the template and the image region:

$$L(x_0, y_0) = d(I_{(x_0,y_0)}, T)$$

distance metric

template

image region

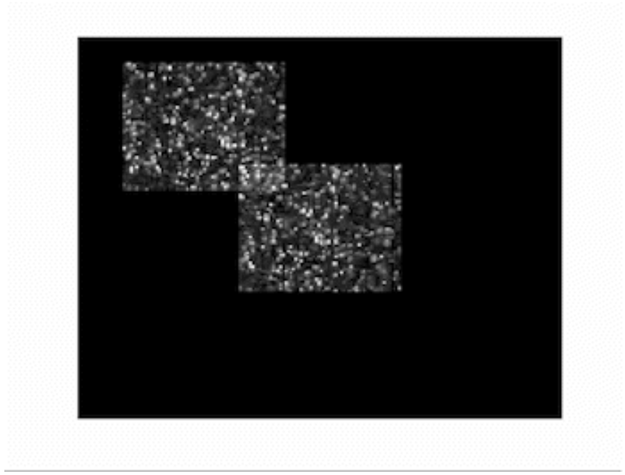
Template matching distance

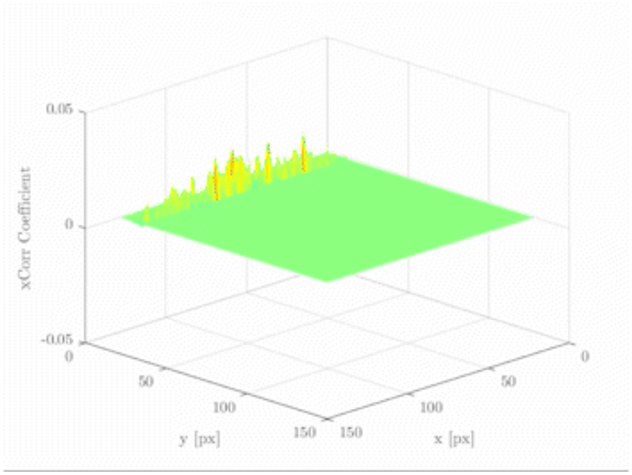
- Normalised cross-correlation (NCC):

$$d(I_{(x_0,y_0)}, T) = \frac{1}{n} \sum_{x,y} \frac{1}{\sigma_I \sigma_T} I_{(x_0,y_0)}(x,y) T(x,y)$$

- Zero-normalised cross-correlation (ZNCC):

$$d(I_{(x_0,y_0)}, T) = \frac{1}{n} \sum_{x,y} \frac{1}{\sigma_I \sigma_T} (I_{(x_0,y_0)}(x,y) - \mu_I)(T(x,y) - \mu_T)$$





Source: https://en.wikipedia.org/wiki/Cross-correlation

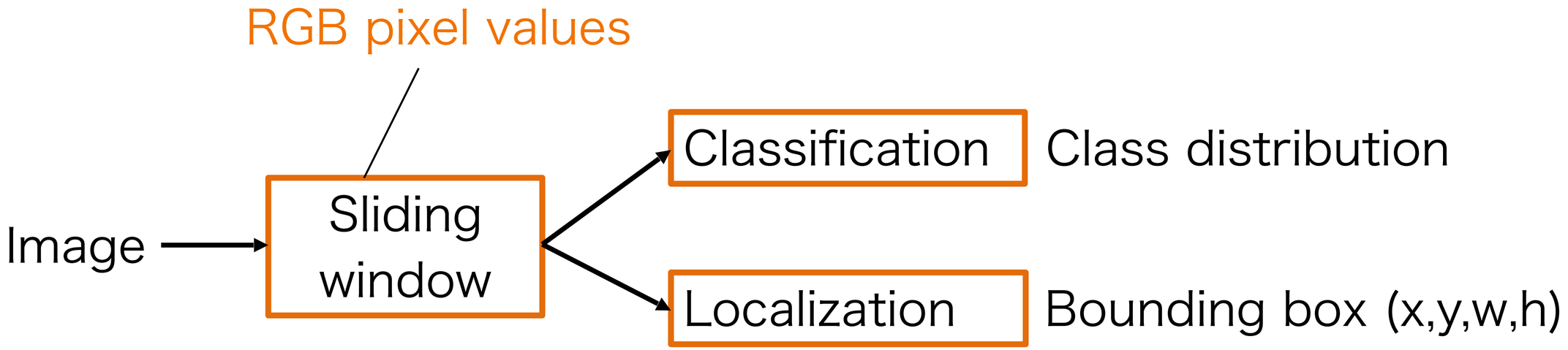
Template matching: disadvantages

- (Self-)occlusions (e.g. due to pose changes)

- Changes in appearance

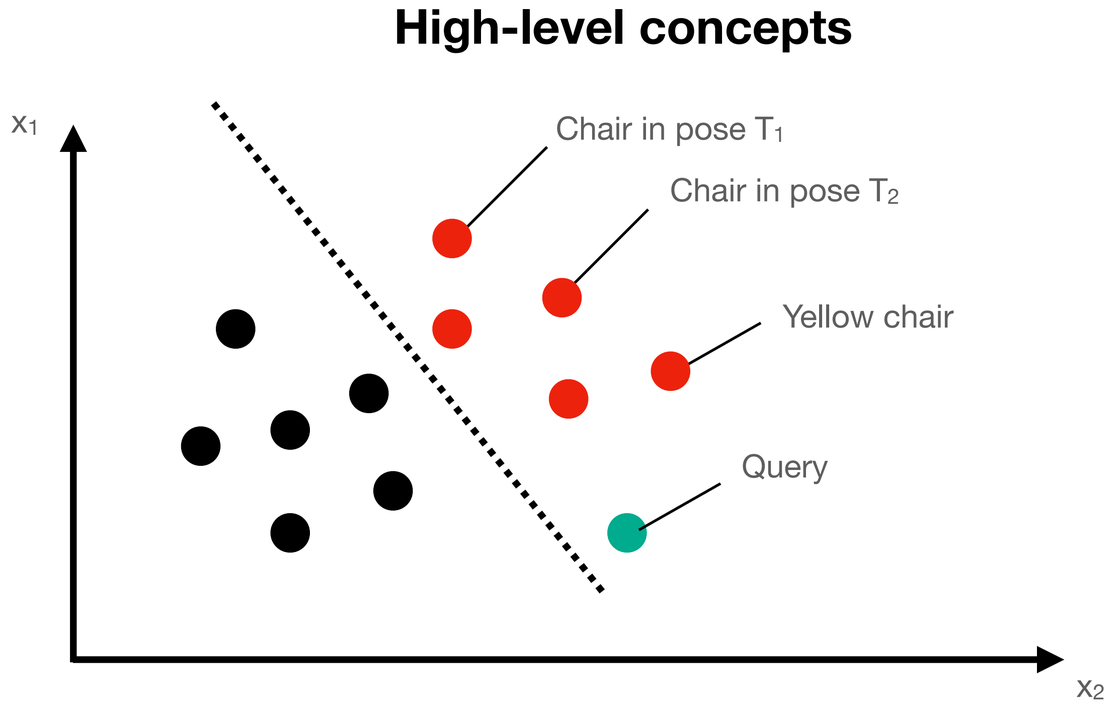
Types of object detectors

- One-stage detectors (new):

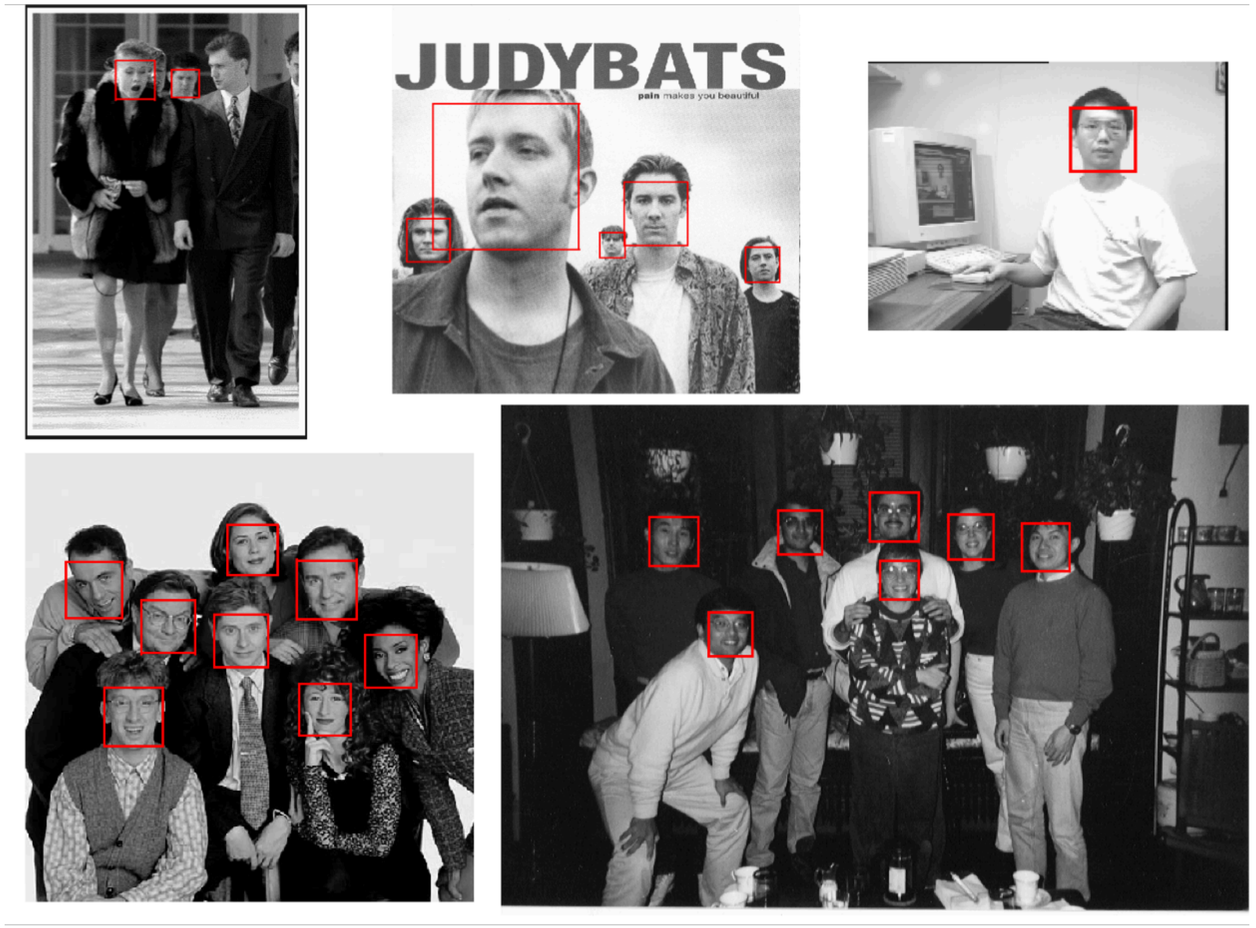


Feature-based detection

Idea: Learn feature based classifiers invariant to natural object changes



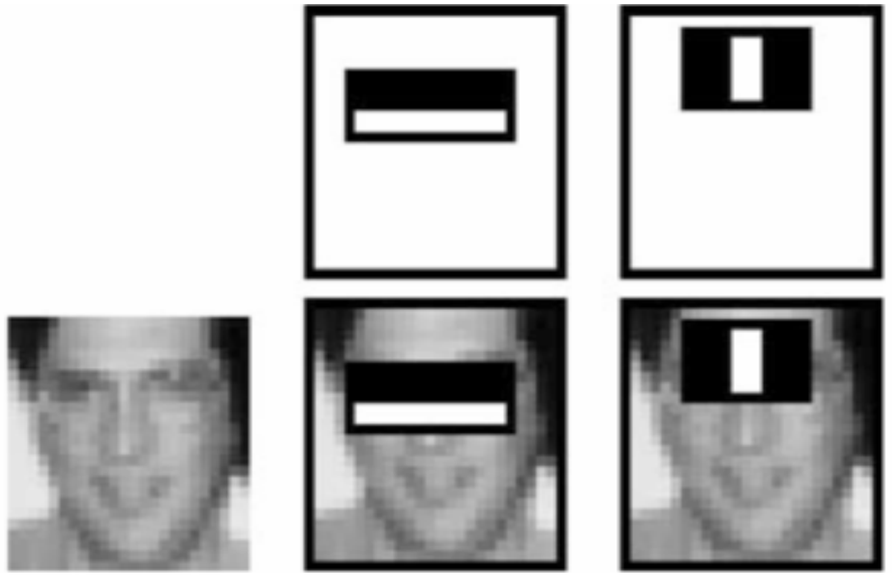
Viola-Jones detector



Viola and Jones. Rapid object detection using a boosted cascade of simple features. CVPR 2001.

Viola-Jones detector

Haar-like features



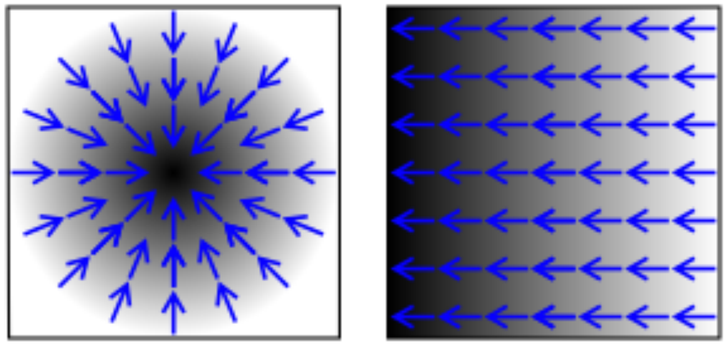
...

$h_1(\cdot)$ $h_2(\cdot)$

“weak” learners

Viola and Jones. Rapid object detection using a boosted cascade of simple features. CVPR 2001.

Histogram of Oriented Gradients

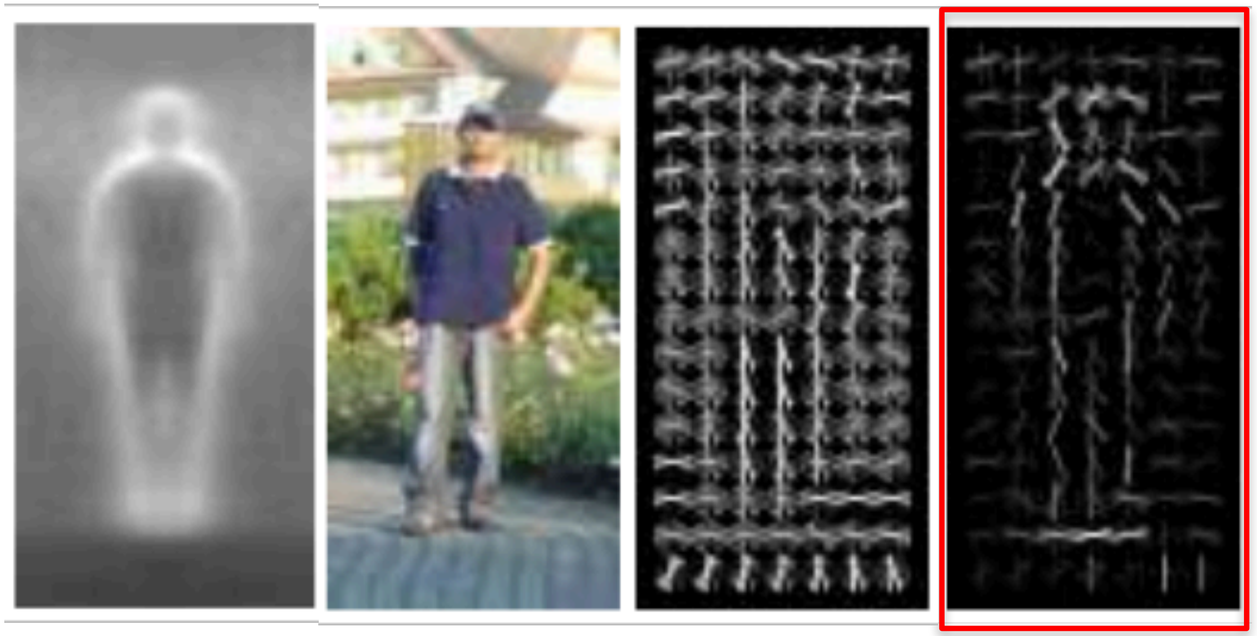


Gradient: blue arrows show the gradient, i.e., the direction of greatest change of the image.

Average gradient image over training samples → gradients provide shape information.

Dalal and Triggs. Histogram of oriented gradients for human detection. CVPR 2005.

Histogram of Oriented Gradients

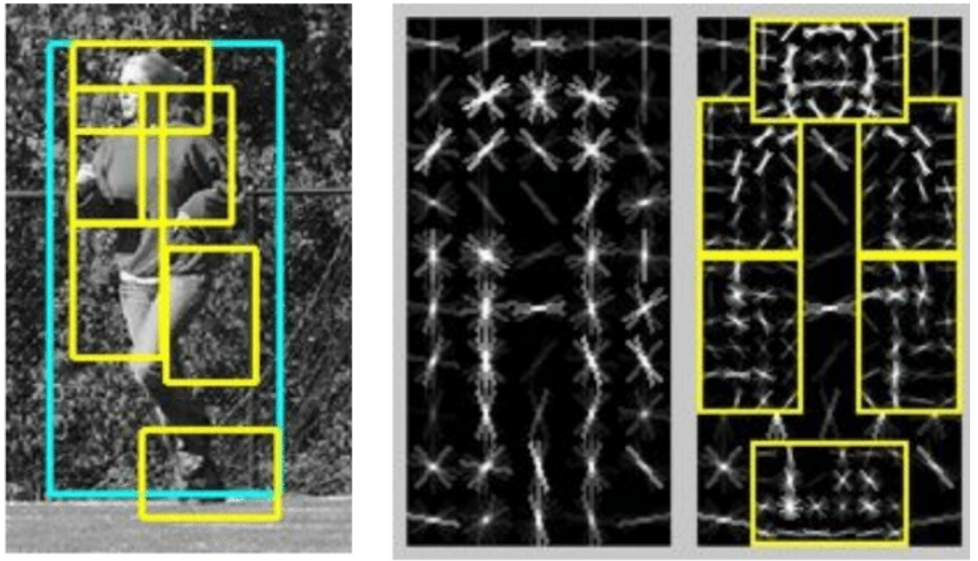


HOG features weighted by the positive SVM weights – the ones used for the pedestrian object classifier.

Dalal and Triggs. Histogram of oriented gradients for human detection. CVPR 2005.

Deformable Part Model

- Many objects are not rigid.
- Bottom-up approach:
  - detect body parts
  - detect “person” if the body parts are in correct arrangement

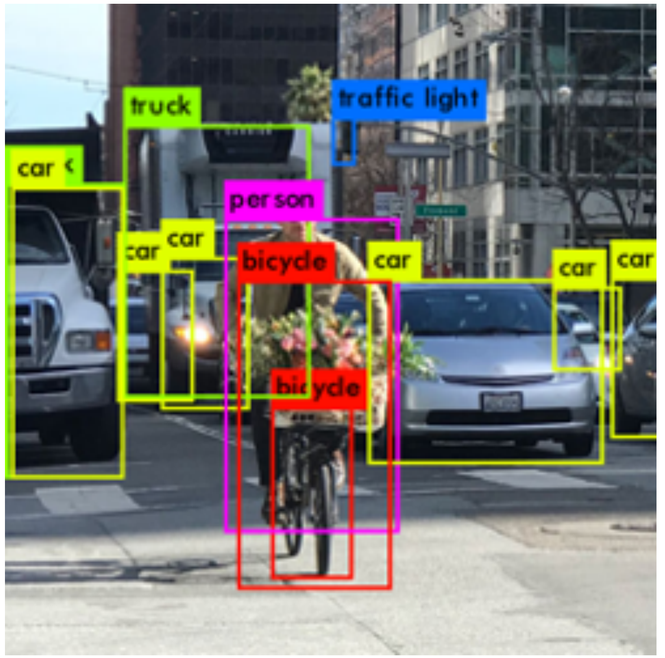


Felzenszwalb et al. A discriminatively trained, multiscale, deformable part model. CVPR 2008.

- Note: The amount of work for each RoI may grow significantly.
- Does it make sense to run it for every sliding window?

Towards real-world object detection

Can we design features that work everywhere?



Probably not by hand, but we can learn them from data.

↓

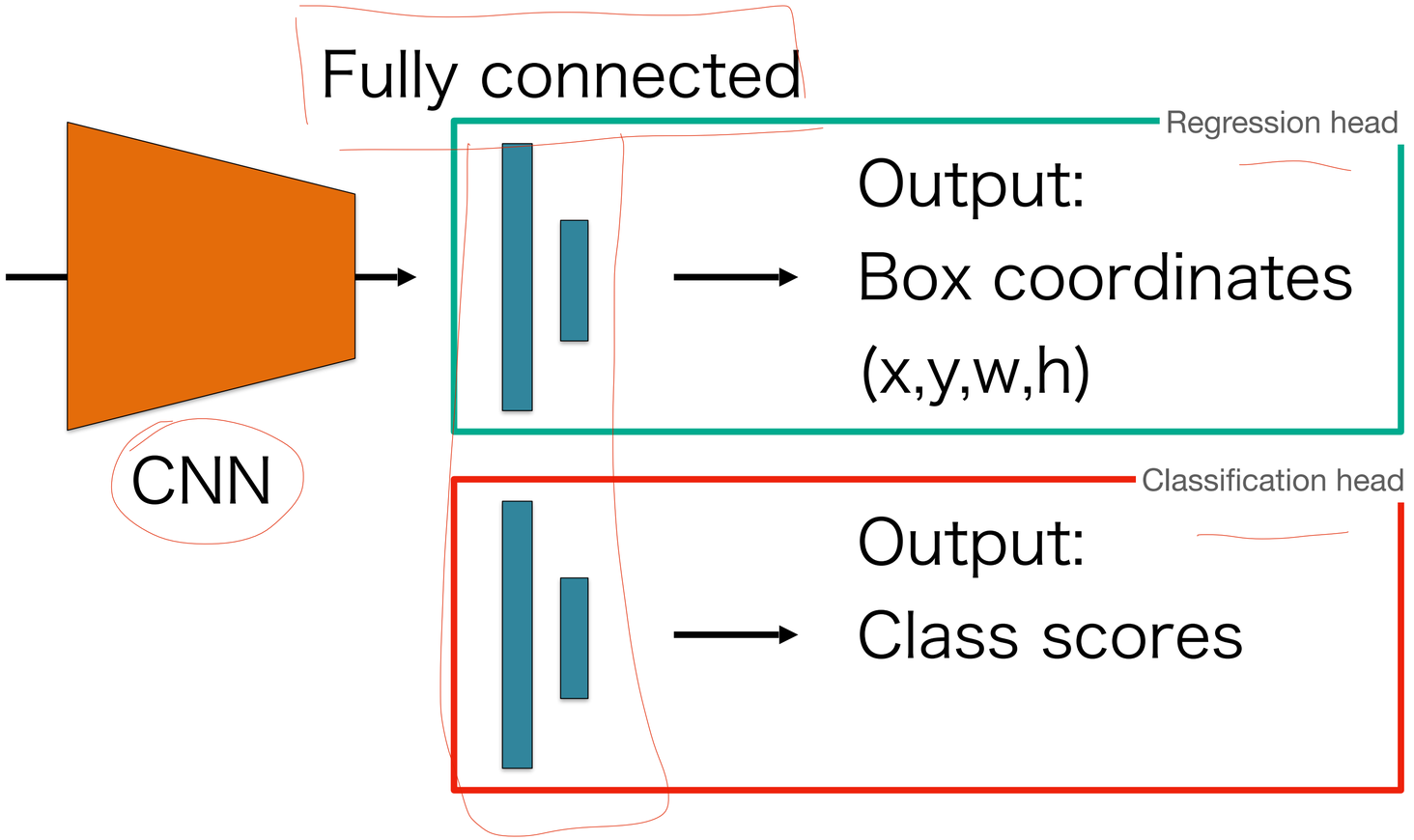
Deep Learning (starting today!)

Object detection with deep networks

Localisation and classification

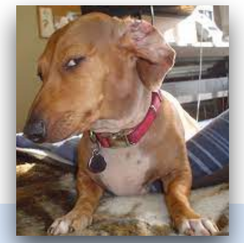
- Bounding box regression

Image



Overfeat

- Train the classification head first, freeze the layers.
- Then train the regression head.
- At test time, we use both.

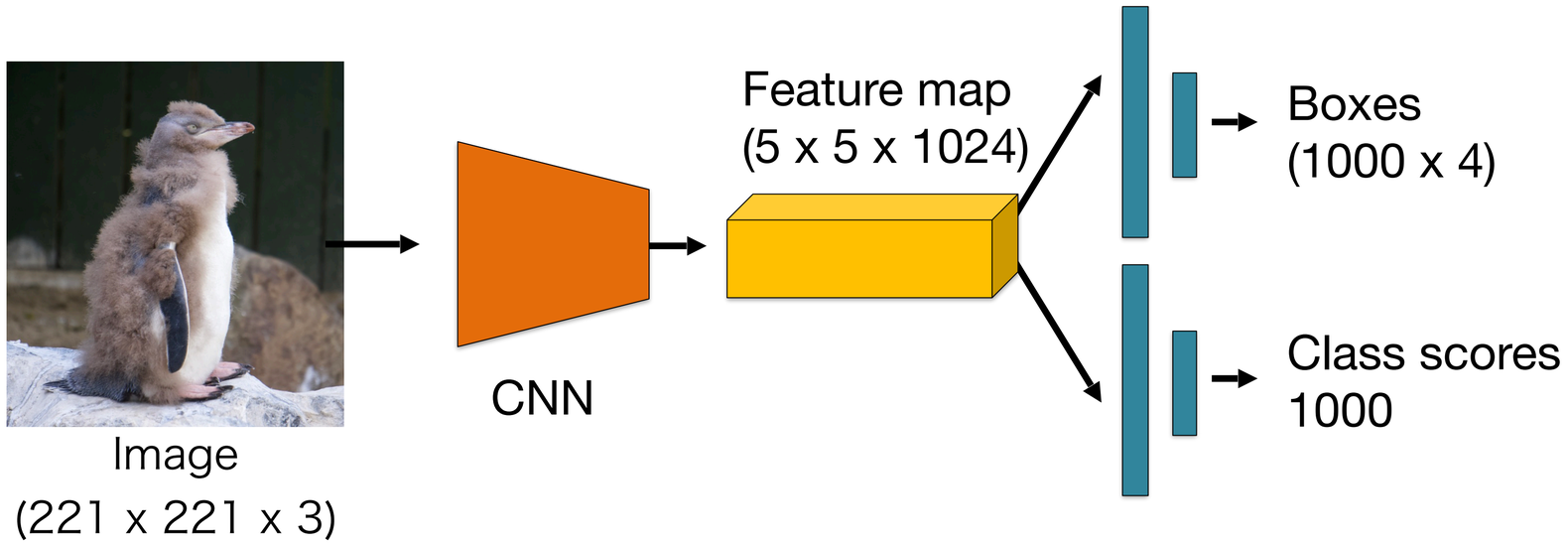


Quiz:
Why not train both tasks jointly?

Sermanet et al, “Overfeat: Integrated Recognition, Localization and Detection using Convolutional Networks”, ICLR 2014

Overfeat

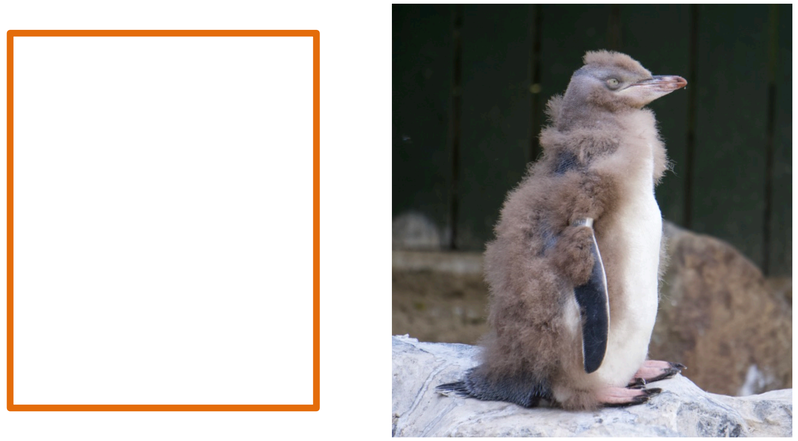
- Sliding window + box regression + classification



Sermanet et al, “Overfeat: Integrated Recognition, Localization and Detection using Convolutional Networks”, ICLR 2014

Overfeat

- Sliding window + box regression + classification



Sermanet et al, “Overfeat: Integrated Recognition, Localization and Detection using Convolutional Networks”, ICLR 2014

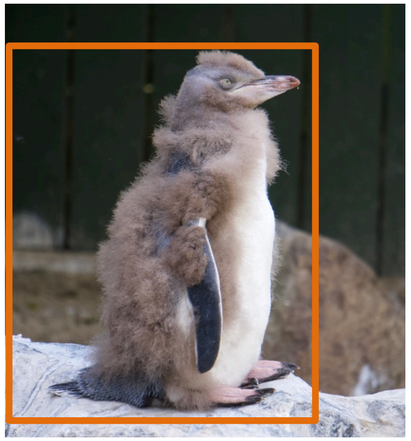
Overfeat

- Sliding window + box regression + classification

Sermanet et al, “Overfeat: Integrated Recognition, Localization and Detection using Convolutional Networks”, ICLR 2014

Overfeat

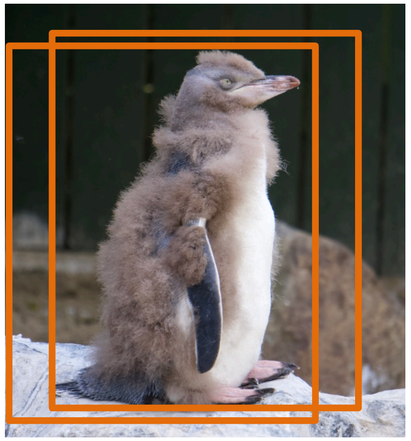
- Sliding window + box regression + classification



Sermanet et al, “Overfeat: Integrated Recognition, Localization and Detection using Convolutional Networks”, ICLR 2014

Overfeat

- Sliding window + box regression + classification

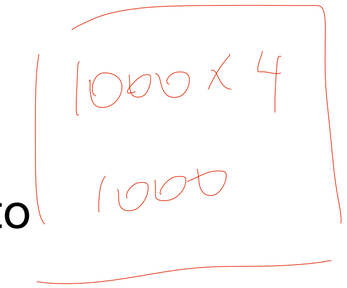


Sermanet et al, “Overfeat: Integrated Recognition, Localization and Detection using Convolutional Networks”, ICLR 2014

Overfeat

- Sliding window + box regression + classification

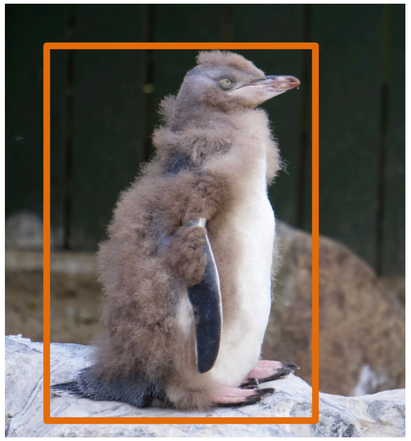
We end up with many predictions and we have to combine them for a final detection



Sermanet et al, “Overfeat: Integrated Recognition, Localization and Detection using Convolutional Networks”, ICLR 2014

Overfeat

- Sliding window + box regression + classification



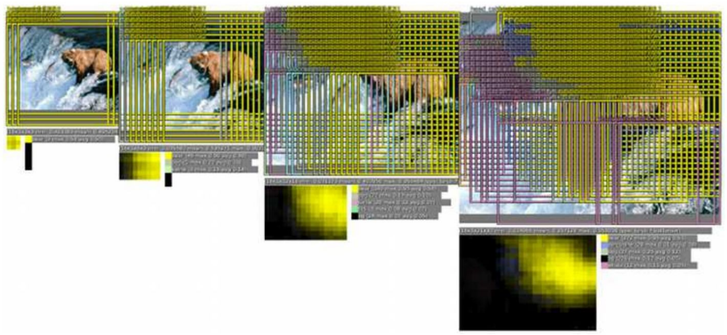
We end up with many predictions and we have to combine them for a final detection

Sermanet et al, “Overfeat: Integrated Recognition, Localization and Detection using Convolutional Networks”, ICLR 2014

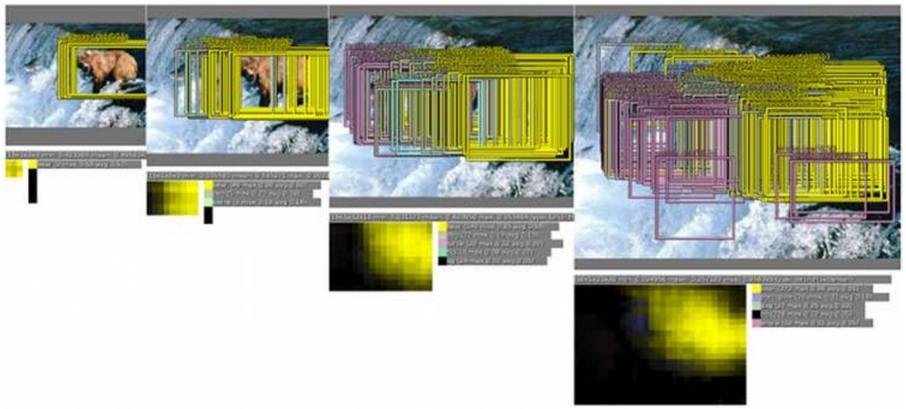
Overfeat

- In practice: use many sliding window locations and multiple scales

Window positions + score maps

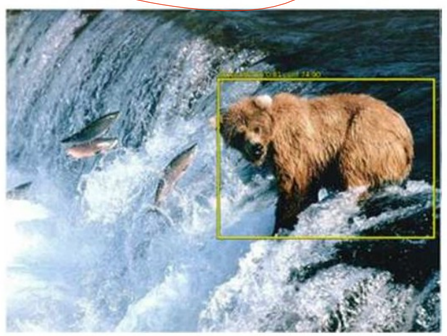


Box regression outputs



How?

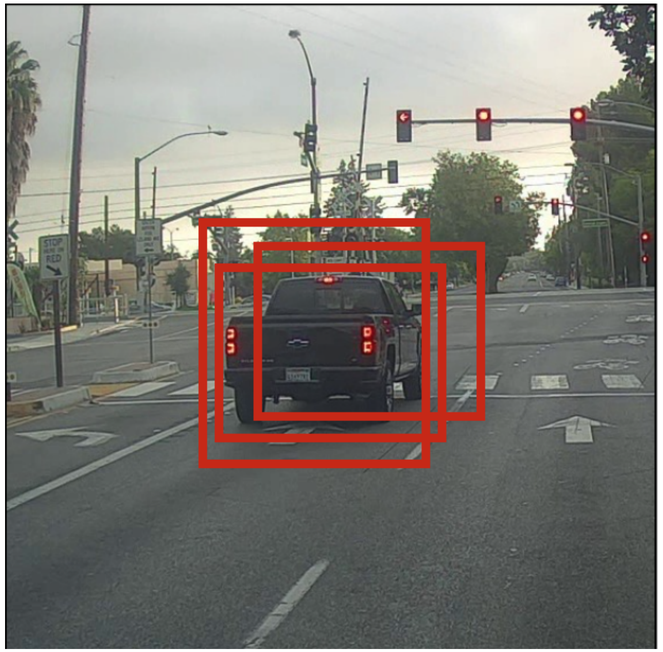
Final Prediction



Sermanet et al, “Overfeat: Integrated Recognition, Localization and Detection using Convolutional Networks”, ICLR 2014

Do we want all “proposals”?

- Nothing wrong with the classifier – it’s doing its job well.
- We need an “external” method to keep only the “best” boxes



choose one box

NMS

Non-Maximum Suppression (NMS)

- Nothing wrong with the classifier – it's doing its job well.
- We need an "external" method to keep only the "best" boxes

Non-Maximum Suppression (NMS)

Algorithm 1 Non-Max Suppression

```
1:  procedure NMS(B,c)
2:      B_nms ← ∅
3:      for b_i ∈ B do
4:          discard ← False
5:          for b_j ∈ B do
6:              if same(b_i, b_j) > λ_nms then
7:                  if score(c, b_j) > score(c, b_i) then
8:                      discard ← True
9:          if not discard then
10:             B_nms ← B_nms ∪ b_i
11:     return B_nms
```

Start with anchor box i

For another box j

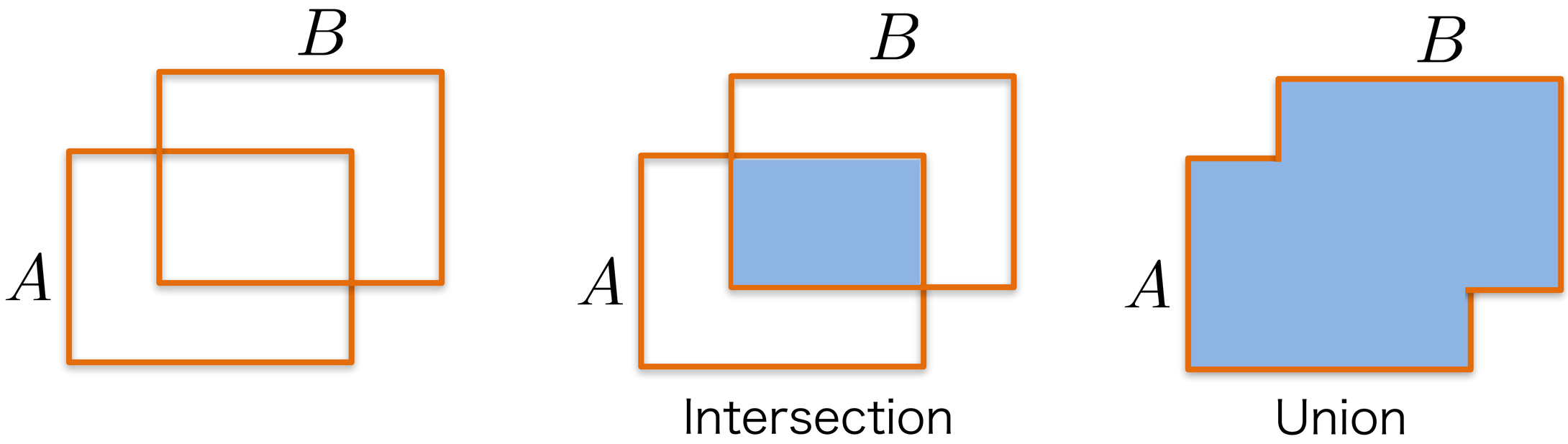
If they overlap

Discard box i if the score is lower than the score of j

Region overlap

- We measure region overlap with the **Intersection over Union (IoU)** or **Jaccard Index**:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$



Region overlap

Algorithm 1 Non-Max Suppression

```
1:  procedure NMS(B,c)
2:      B_nms ← ∅
3:      for b_i ∈ B do
4:          discard ← False
5:          for b_j ∈ B do
6:              if same(b_i, b_j) > λ_nms then
7:                  if score(c, b_j) > score(c, b_i) then
8:                      discard ← True
9:          if not discard then
10:             B_nms ← B_nms ∪ b_i
11:     return B_nms
```

Start with anchor box i

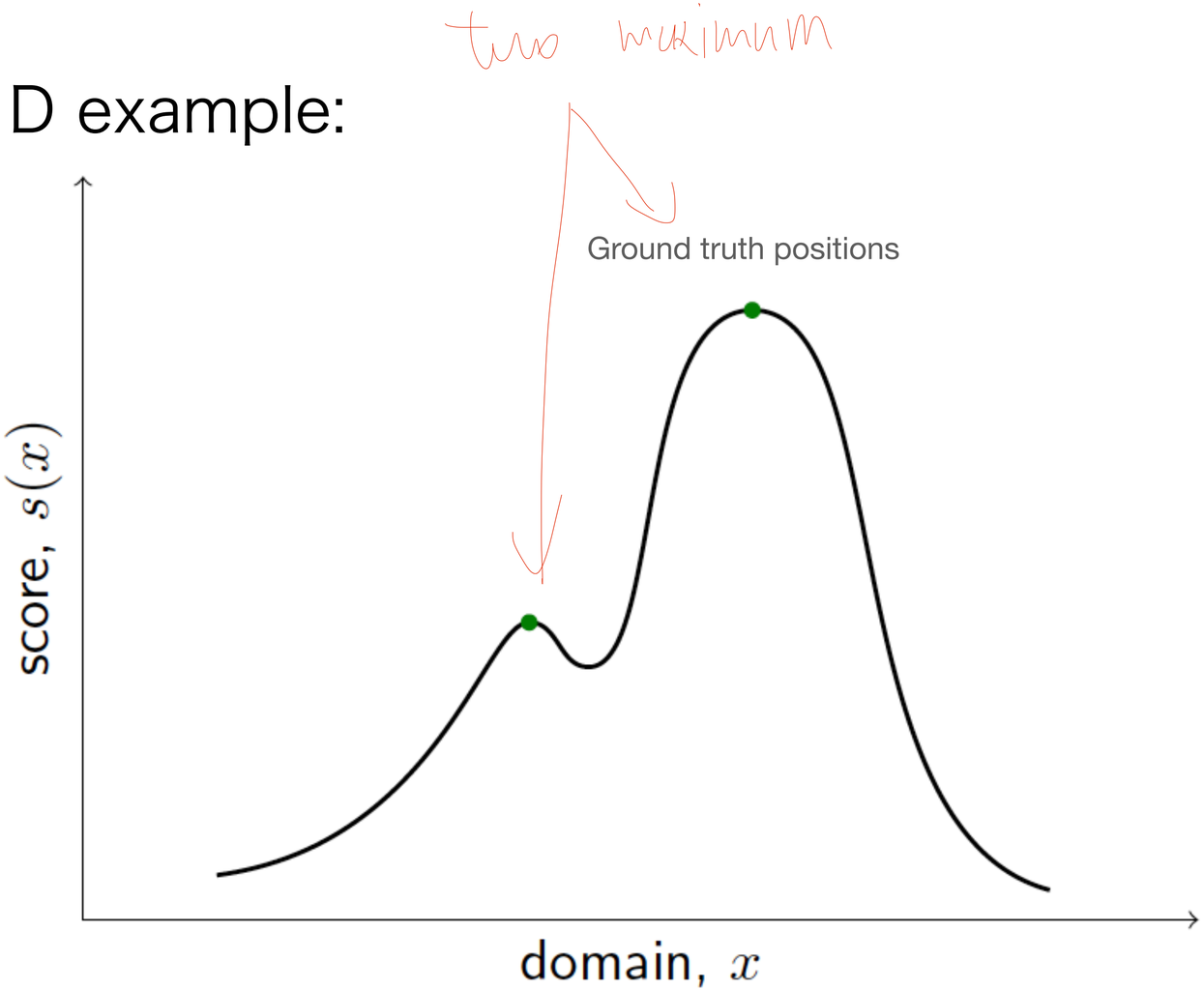
For another box j

If they overlap

Discard box i if the score is lower than the score of j

NMS: The problem

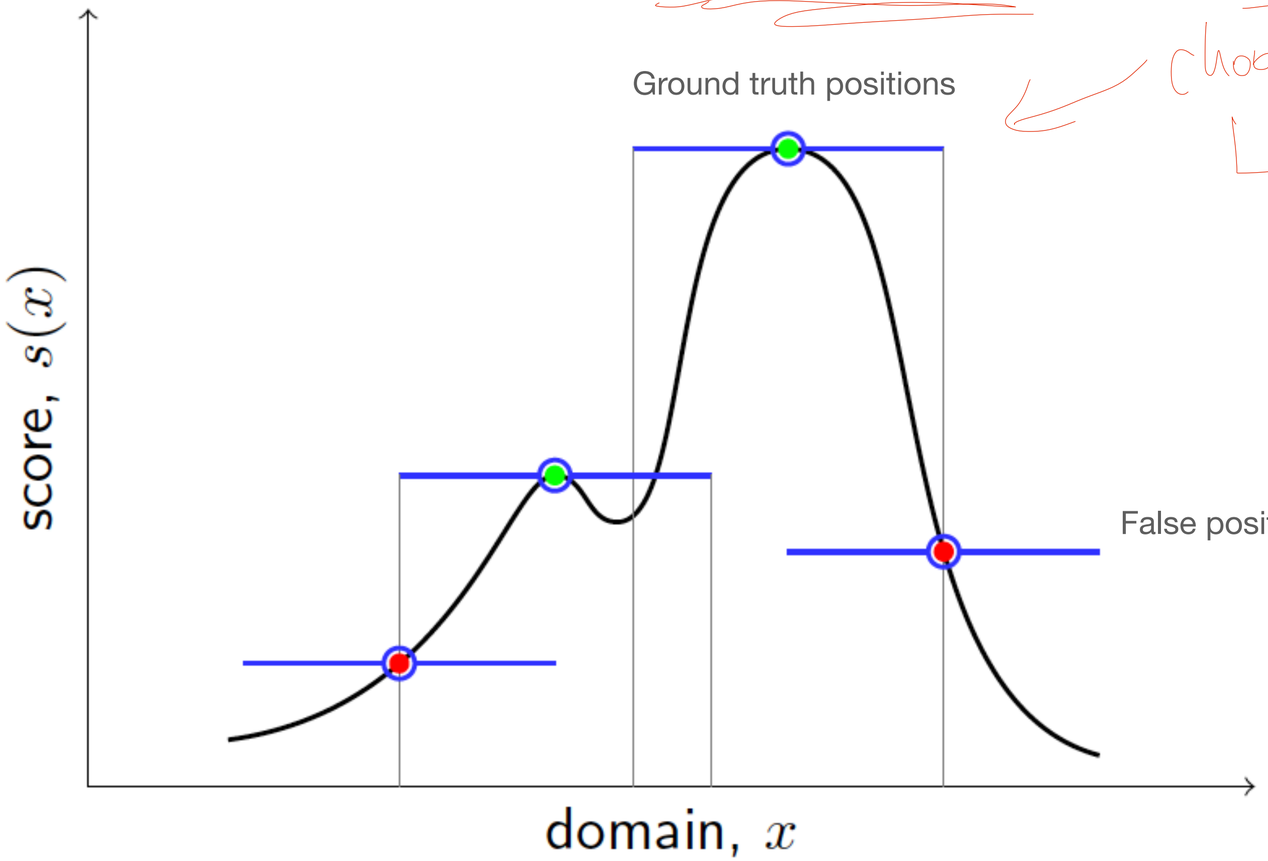
- Consider a 1D example:



Hosang, Benenson and Schiele. A ConvNet for Non-Maximum Suppression. GCPR 2015.

NMS: The problem

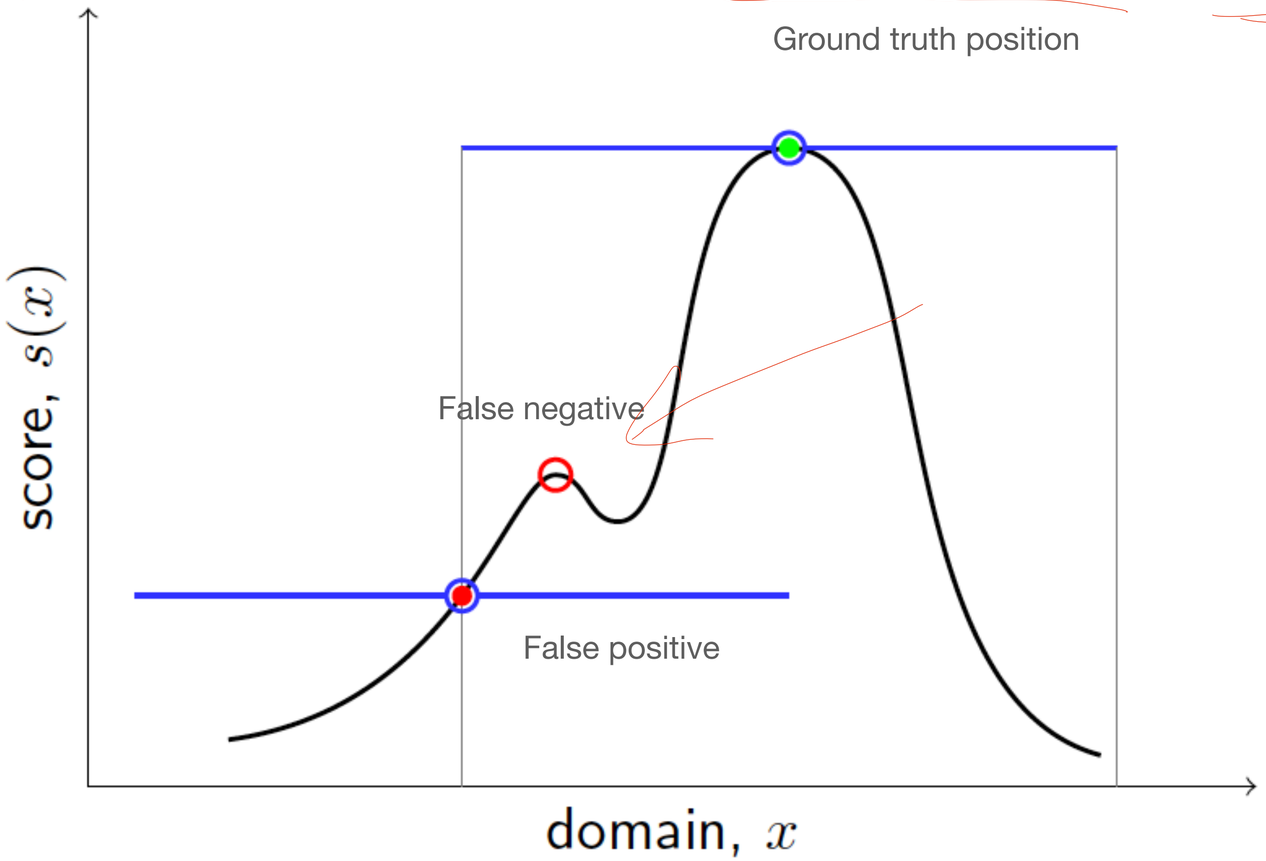
- Choosing a high threshold – more false positives, low precision



Hosang, Benenson and Schiele. A ConvNet for Non-Maximum Suppression. GCPR 2015.

NMS: The problem

- Choosing a low threshold – more false negatives, low recall



Hosang, Benenson and Schiele. A ConvNet for Non-Maximum Suppression. GCPR 2015.

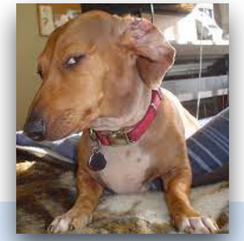
Non-Maximum Suppression (NMS)

- NMS will be used at test time. Most detection methods (even Deep Learning ones) use NMS!

Region overlap

Algorithm 1 Non-Max Suppression

```
procedure NMS(B,c)
    B_nms ← ∅
    for b_i ∈ B do
        discard ← False
        for b_j ∈ B do
            if same(b_i, b_j) > λ_nms then
                if score(c, b_j) > score(c, b_i) then
                    discard ← True
        if not discard then
            B_nms ← B_nms ∪ b_i
    return B_nms
```

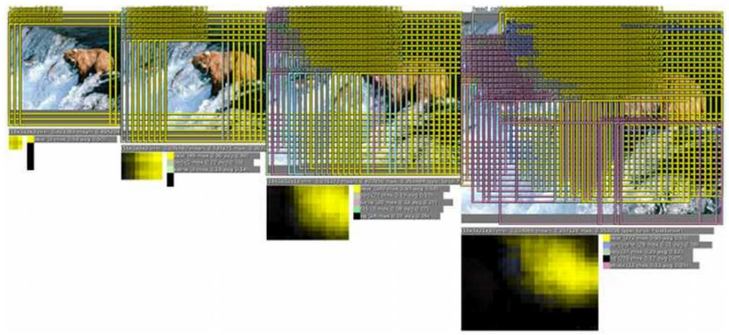


Quiz:
What's the runtime complexity?

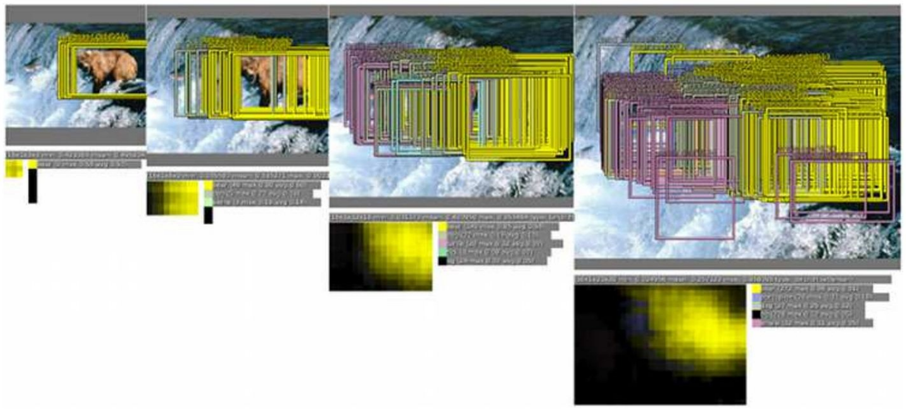
Overfeat

- In practice: use many sliding window locations and multiple scales

Window positions + score maps



Box regression outputs



NMS

Final Prediction

Sermanet et al, “Overfeat: Integrated Recognition, Localization and Detection using Convolutional Networks”, ICLR 2014

Overfeat

- Improved detection accuracy
  - largely due to feature representation learned with deep nets.

- Cons:
  - Expensive to try all possible positions, scales and aspect ratios.
  - Network works on a fixed input.

Sliding a window is slow

- How can we improve the runtime?

“amount of work”

```
0. for each sliding window s in S:
1.   detect_and_classify(S)
```

```
O(N := |S|)
O(D)
```

Quiz:
What is the runtime complexity?

- Let’s “pre-filter” the sliding windows with a cheap method:

```
0. for each sliding window s in S:
1.   if fast_check(s): add(S*, s)
2. for each proposal in S*:
3.   detect_and_classify(S)
```

```
O(N := |S|)
O(d)
O(n := |S*|)
O(D)
```

Quiz:
What is the runtime complexity?

Sliding a window is slow

- When does “pre-filtering” pay off?

```
O(ND) > O(Nd + nD)
```

- Let’s assume the constant factors are comparable / negligible.
- Then:

Region-of-Interest (RoI) ratio efficiency of RoI generator

$$\frac{n}{N} + \frac{d}{D} < 1$$

Quiz:
We filter out half of sliding windows, using with d = D/2 method.
Is it worth it?

=1

Sliding a window is slow

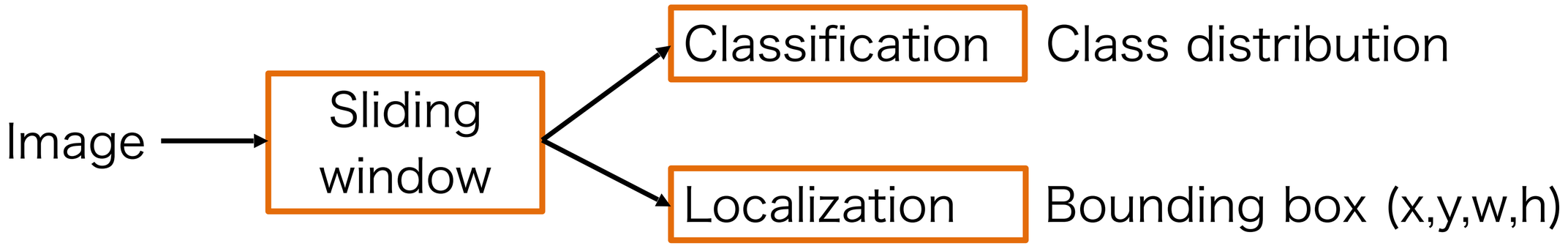
- Let's assume the constant factors are comparable. Then,

$$\frac{n}{N} + \frac{d}{D} < 1$$

- In practice, there is a delicate balance between `n` and `d`:
  - Reducing `d` increases `n`.
- Quiz: What's a (tight) lower bound of $\frac{n}{N} + \frac{d}{D}$?

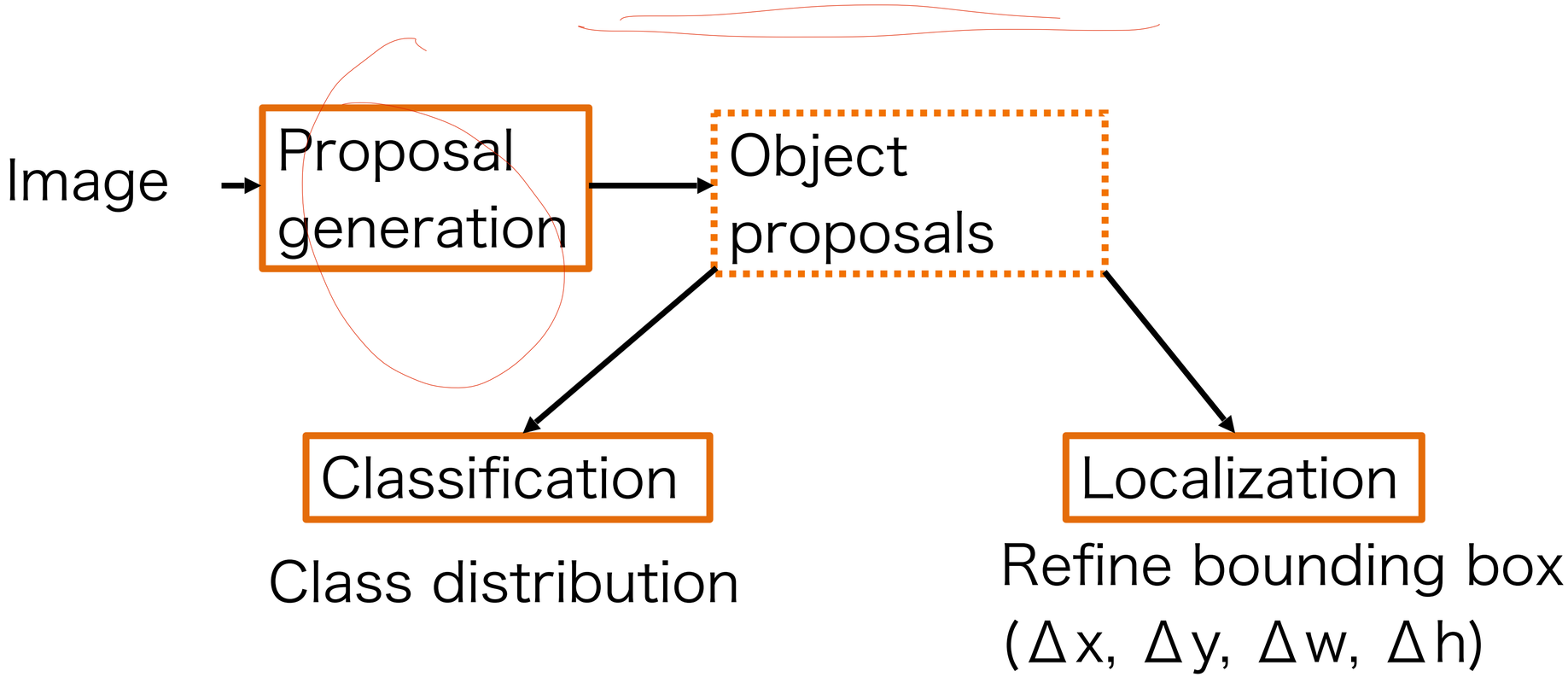
Types of object detectors

- One-stage detectors (before):



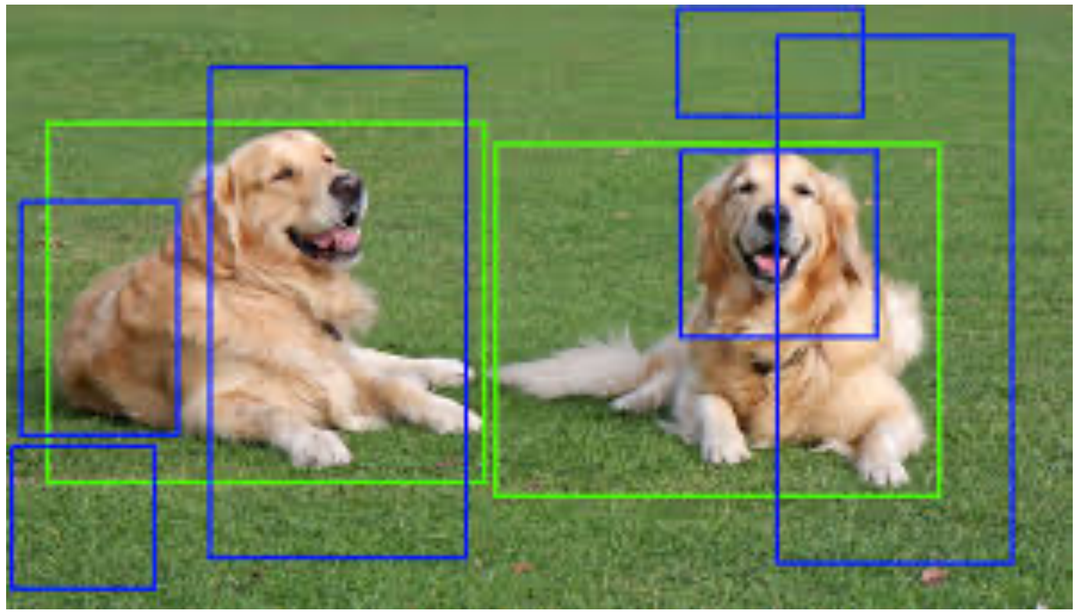
Two-stage detectors

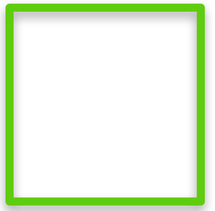
- “Pre-filtering” → generating object proposals



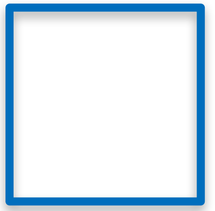
What defines an object?

- We need a generic, **class-agnostic objectness** measure: how likely it is for an image region to contain an object





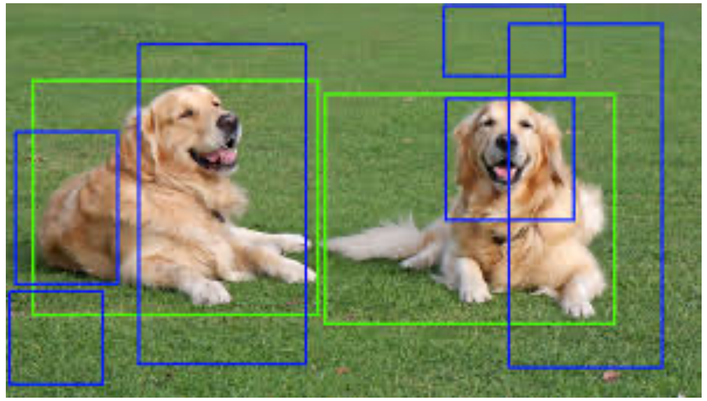
Very likely to be an object



Maybe it is an object

What defines an object?

- We need a generic, **class-agnostic objectness** measure: how likely it is for an image region to contain an object
- Using this measure yields a number of candidate **object proposals** or **regions of interest (RoI)** where to focus.

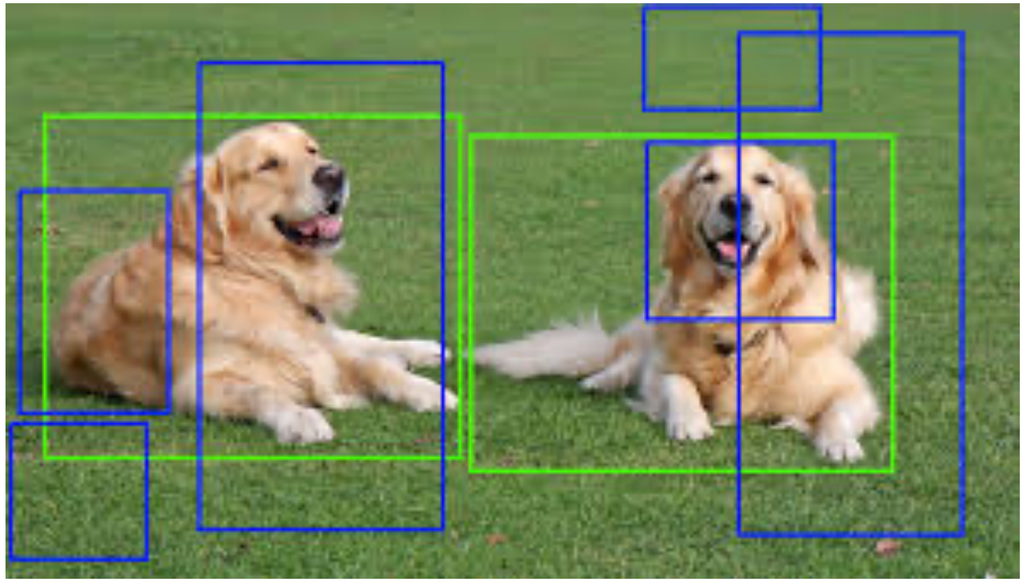


+ classifier

Region proposals

We can use heuristic-based methods producing “interesting” regions

1. Obtain region proposals.
2. Classify & refine them.



Uijlings et al. Selective Search for Object Recognition. IJCV 2013

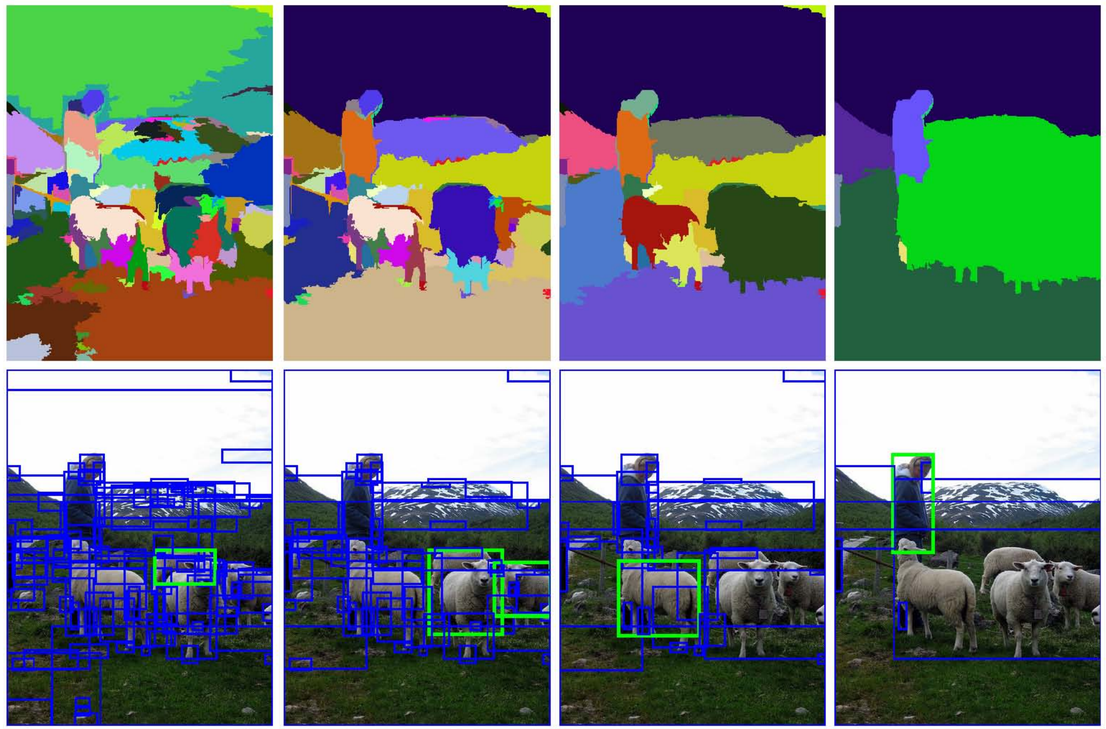
Object proposal methods

- Selective search: van de Sande et al. Segmentation as selective search for object recognition. ICCV 2011.
  - Using class-agnostic segmentation at multiple scales.

- Edge boxes: Zitnick and Dollar. Edge boxes: Locating object proposals from edges. ECCV 2014.
  - Bounding boxes that wholly enclose detected contours.

Object proposal methods

- **Selective search**: van de Sande et al. Segmentation as selective search for object recognition. ICCV 2011.



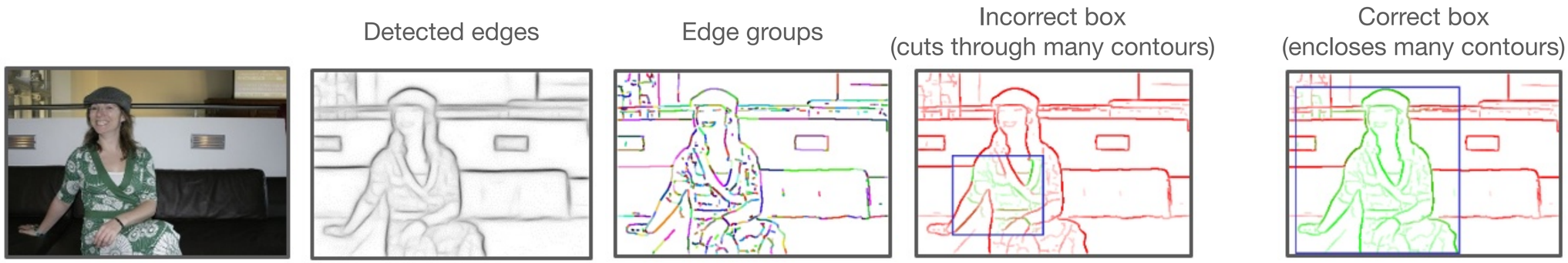
- Hierarchical segmentation:
  - Start with oversegmentation
  - Iteratively group two most similar segments (greedy)
  - Continue until the whole image is a single segment

Object proposal methods

- **Edge boxes**: Zitnick and Dollar. Edge boxes: Locating object proposals from edges. ECCV 2014.

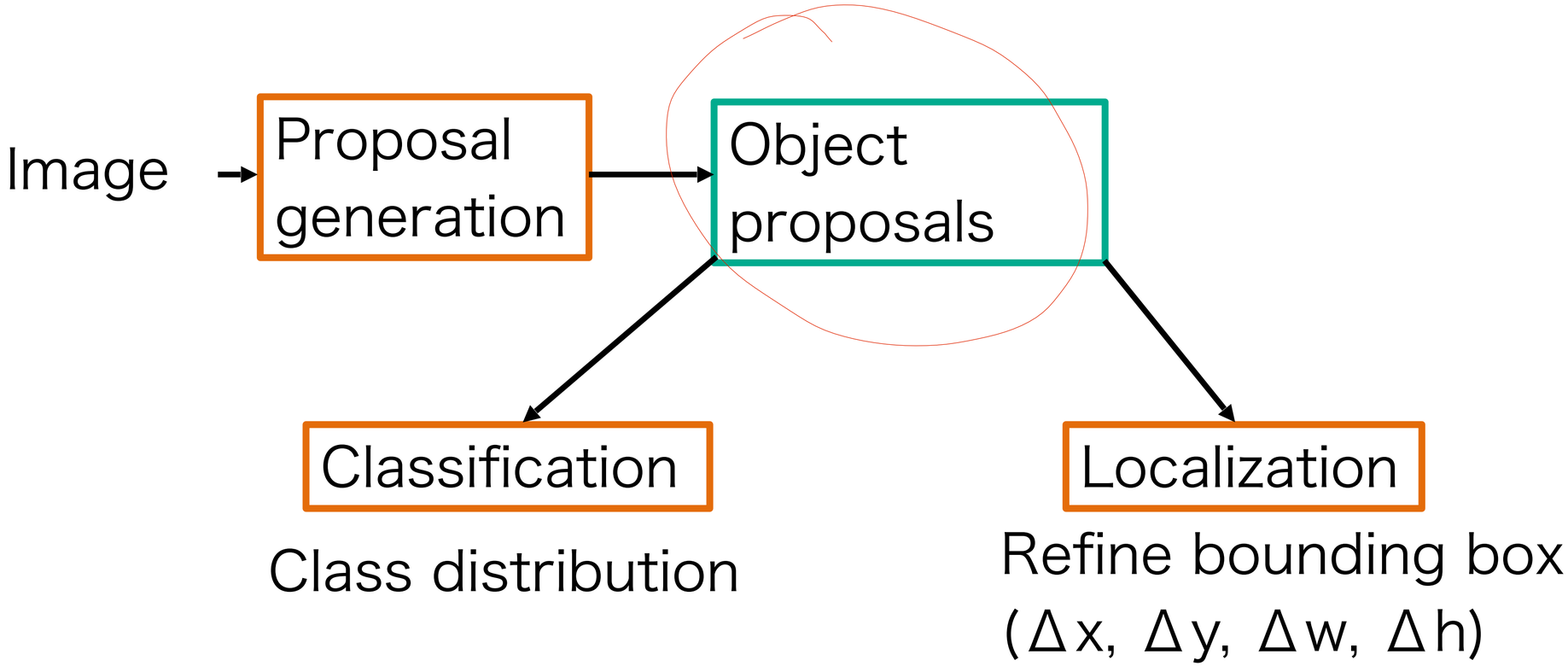
Main observation: “the number of contours wholly enclosed by a bounding box is indicative of the likelihood of the box containing an object.”

被边界框完全包围的轮廓线的数量表明边界框中包含物体的可能性。



Types of object detectors

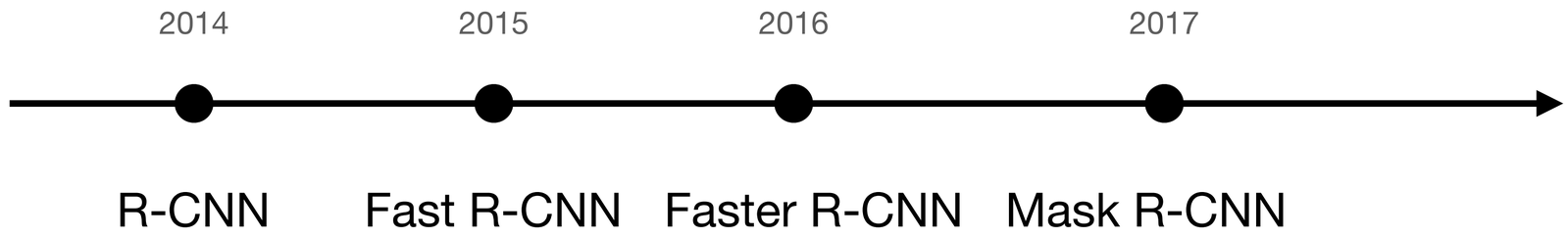
- Two-stage detectors



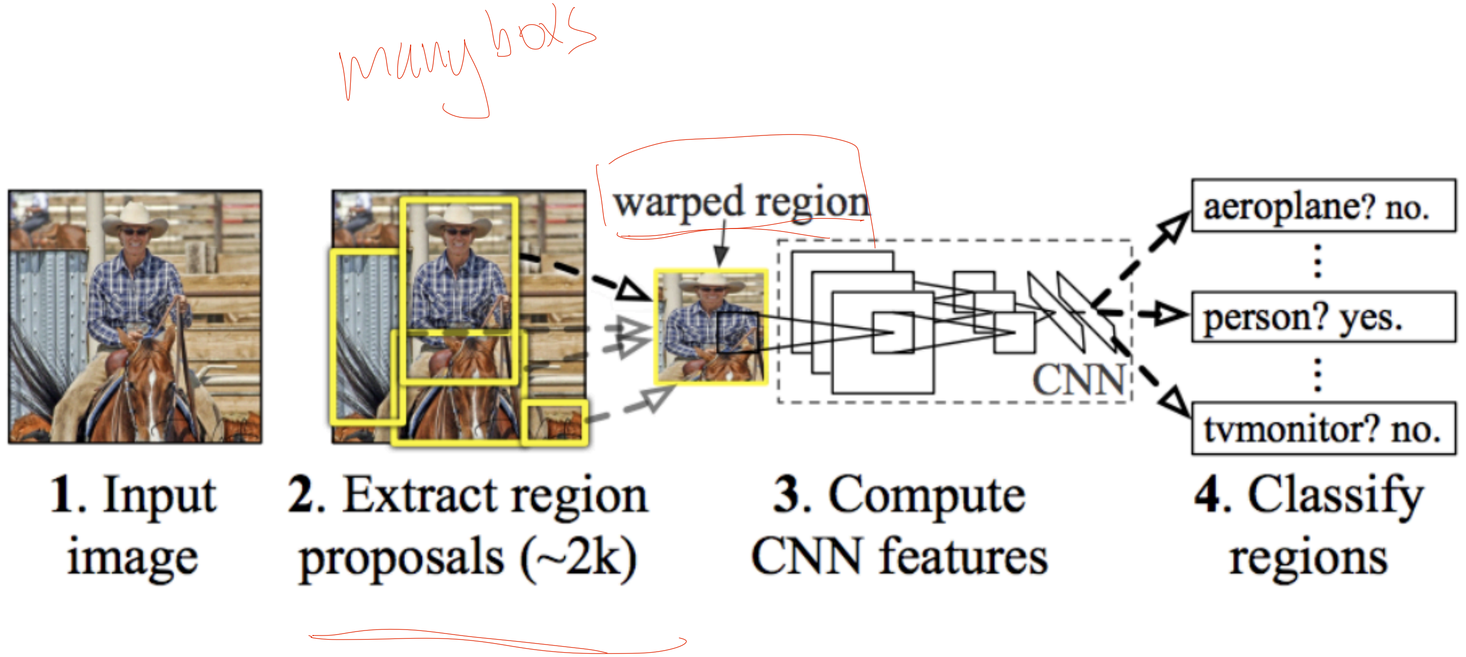
R-CNN family

R-CNN family

- “Regions with CNN features”
- One of the most impactful lines of work on multi-object detection and segmentation
- … and in CV
  - >120K total citations: R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN



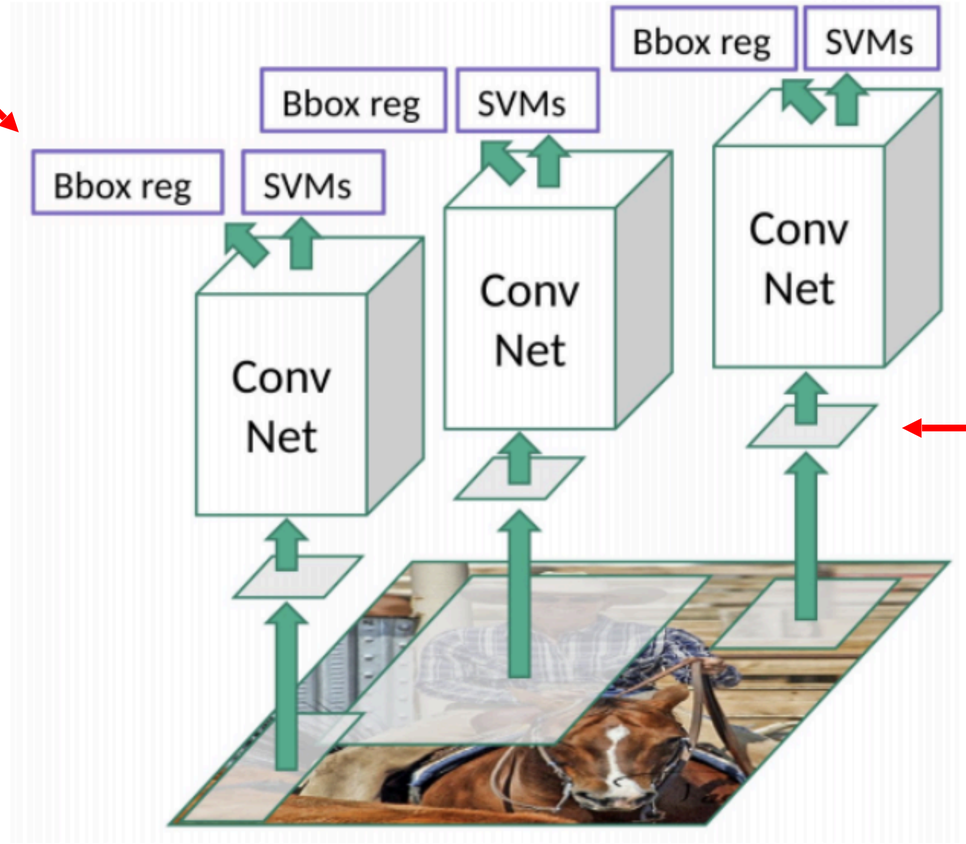
R-CNN



Girschick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014

R-CNN

Regression head for the bounding box location



Classification head

Extract features

Warping to a fix size 227 x 227

R-CNN

- Training scheme:
  1. Pre-train the CNN for image classification (ImageNet)
  2. Finetune the CNN on the number of classes the detector is aiming to classify
  3. Train a linear Support Vector Machine classifier to classify image regions – one linear SVM per class.
  4. Train the bounding box regressor

R-CNN

- Pros:
  - New: CNN features; the overall pipeline with proposals is heavily engineered → good accuracy.
  - CNN summarises each proposal into a 4096 vector (compare to HoG).
  - Leverage transfer learning: The CNN can be pre-trained for image classification with C classes. One needs only to change the FC layers to deal with Z classes.

R-CNN

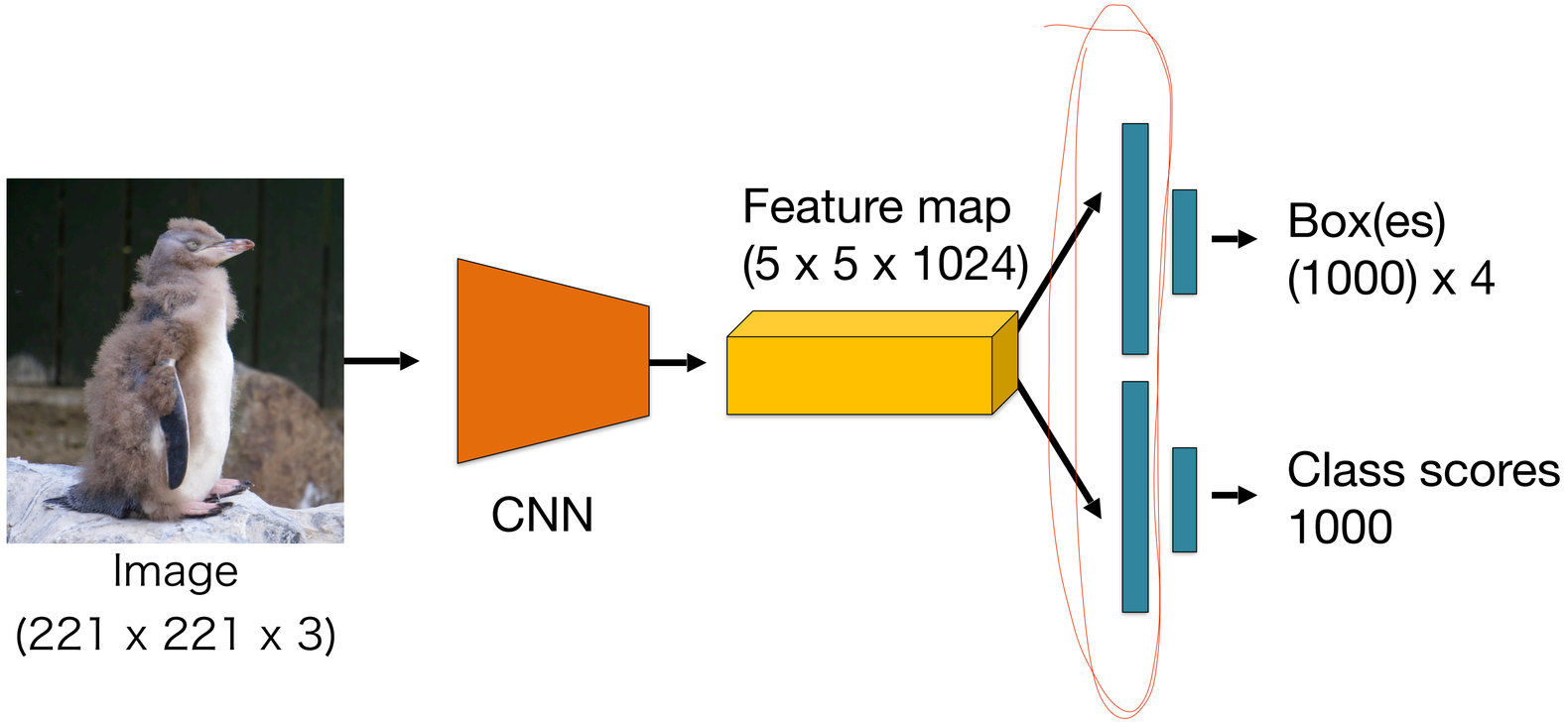
- Cons:
  - Slow: 47s/image with VGG16 backbone. One considers around 2000 proposals per image, they need to be warped and forwarded through the CNN.
  - Training is also slow and complex
  - The object proposal algorithm is fixed.
  - Feature extraction and SVM classifier are trained separately – features are not learned “end-to-end”

Our goals so far

- How to improve the runtime further?
- How to integrate the proposals into model training?

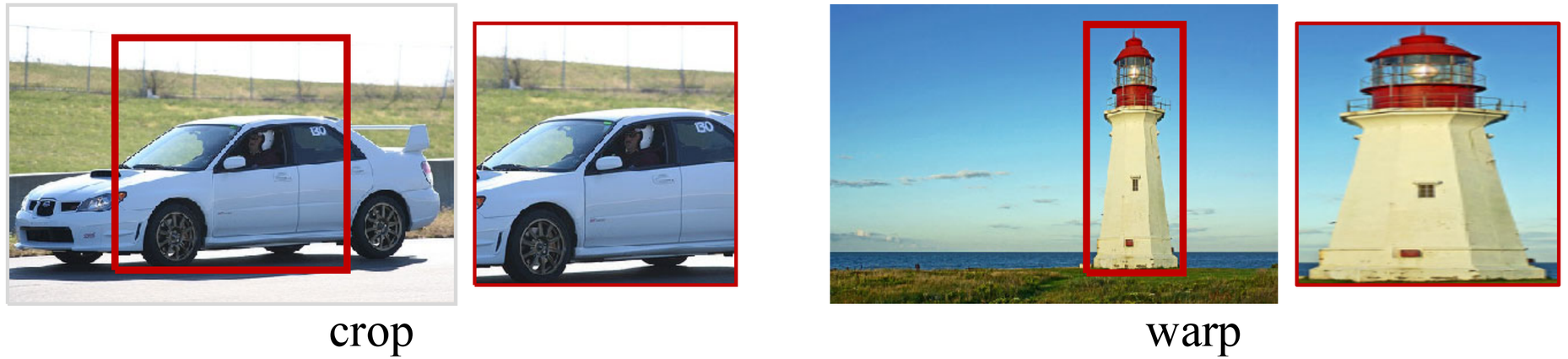
- A common problem of the two questions:
  - Our input image has fixed size.

Dealing with variable input size



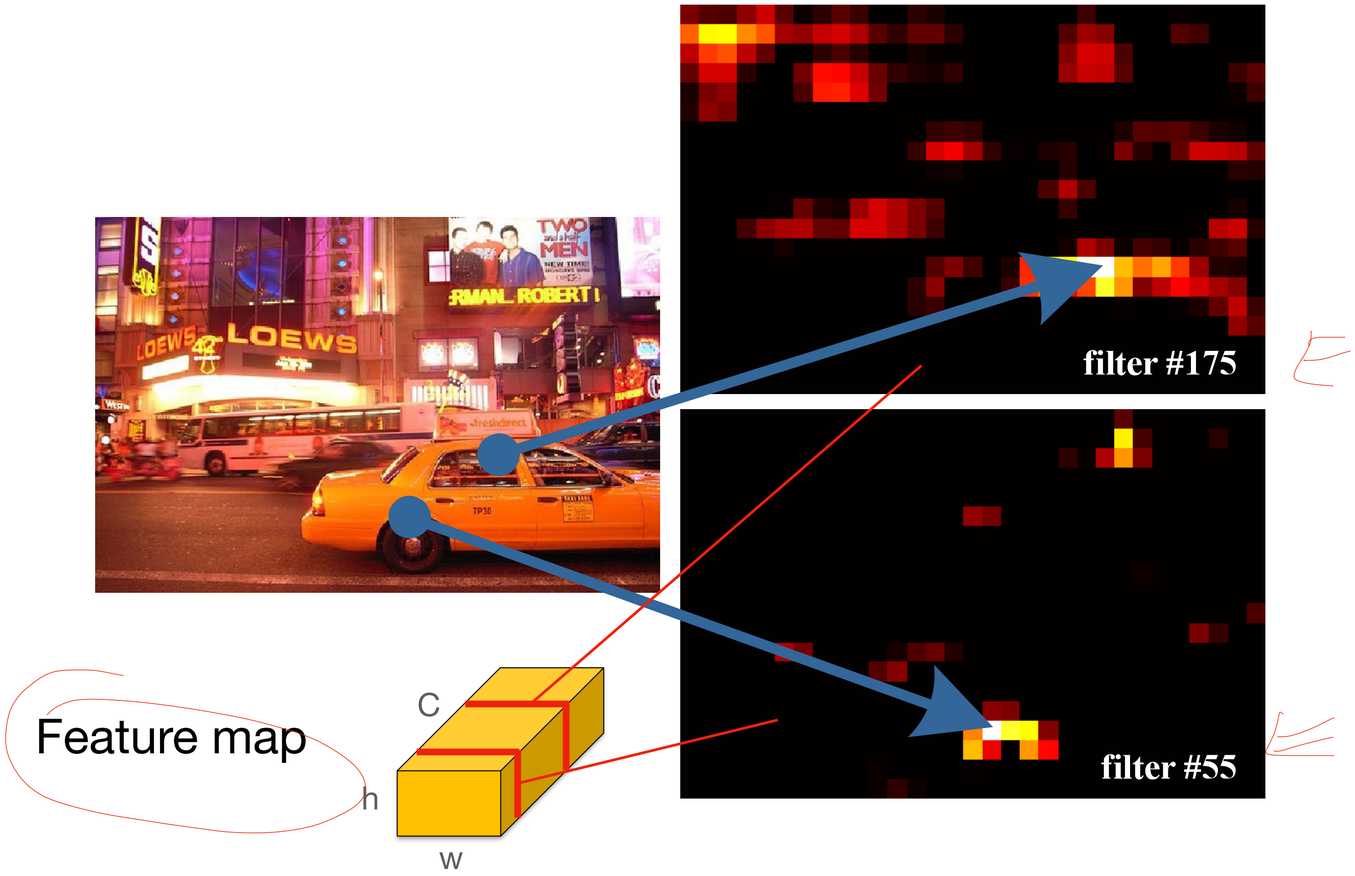
What prevents us from dealing with any image size?

SPP-Net: Spatial Pyramid Pooling



He et al. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. ECCV 2014.

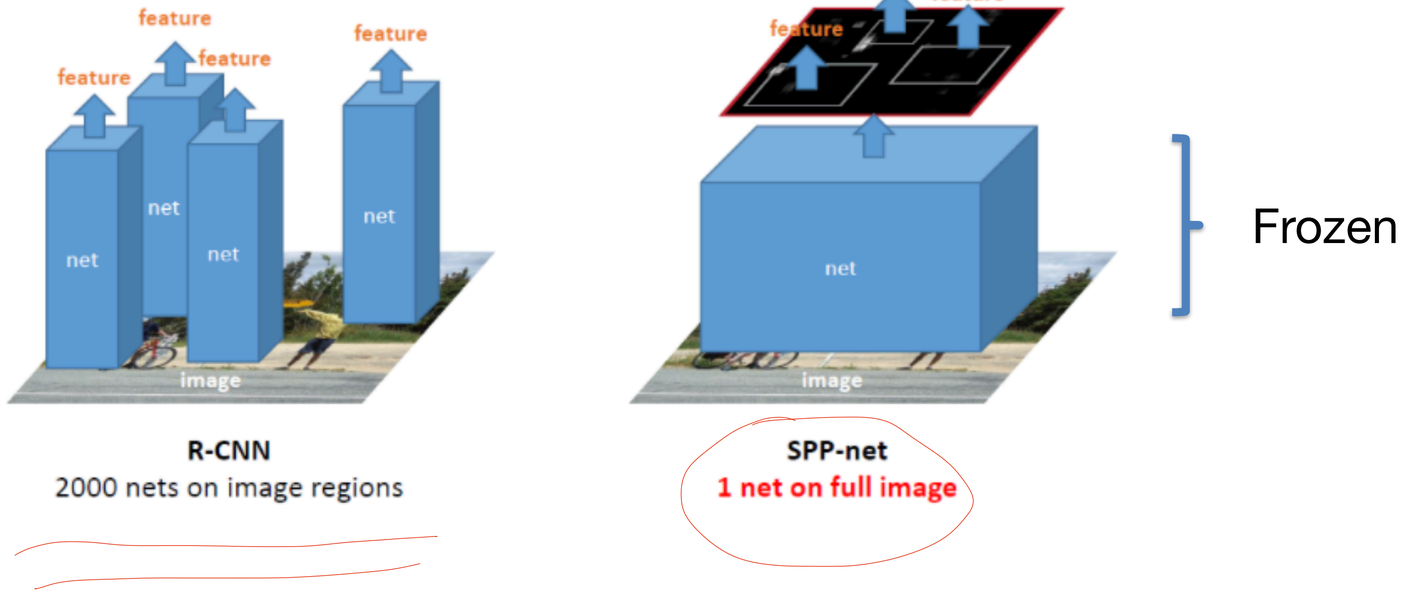
Object detection with deep nets



He et al. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. ECCV 2014.

SPP-Net: Overview

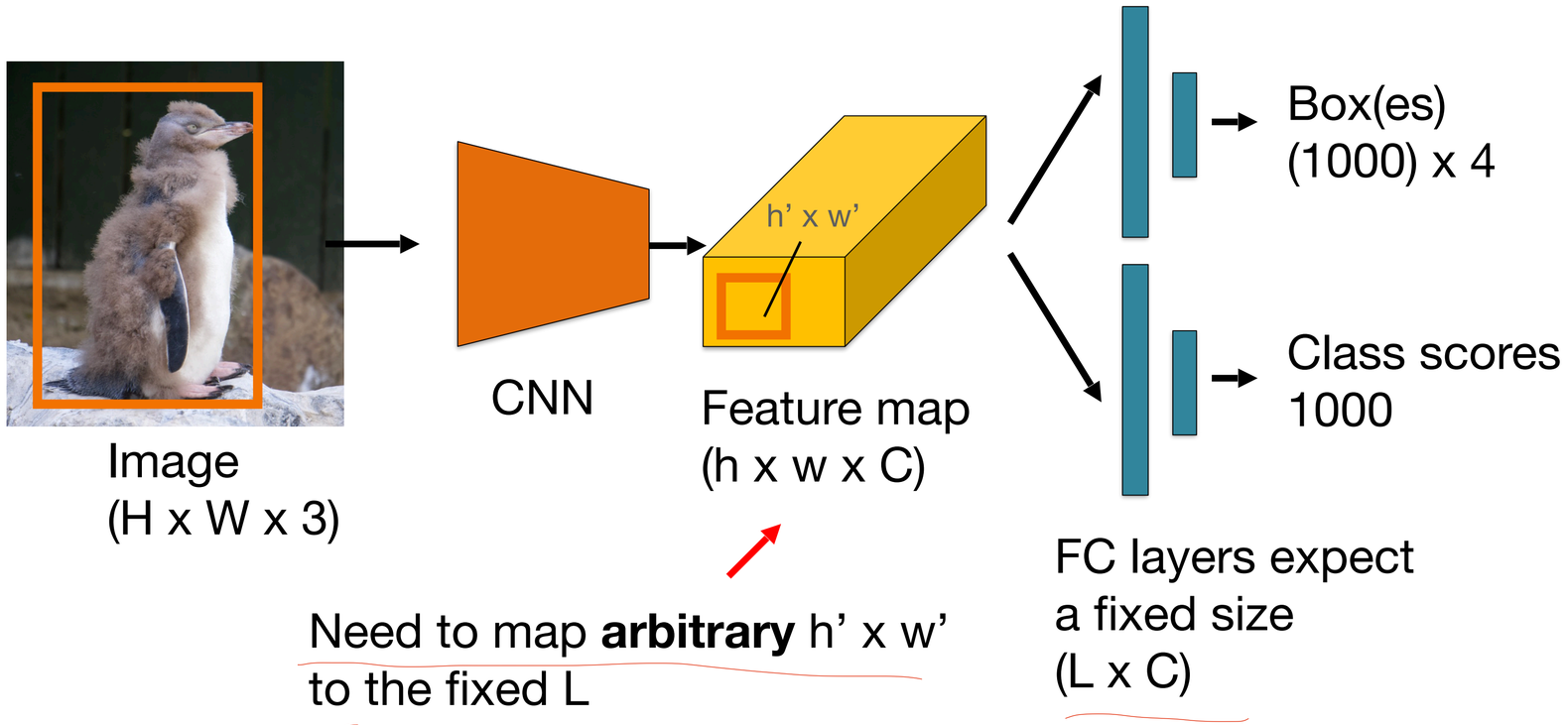
How do we “pool” these features into a common size



He et al. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. ECCV 2014.

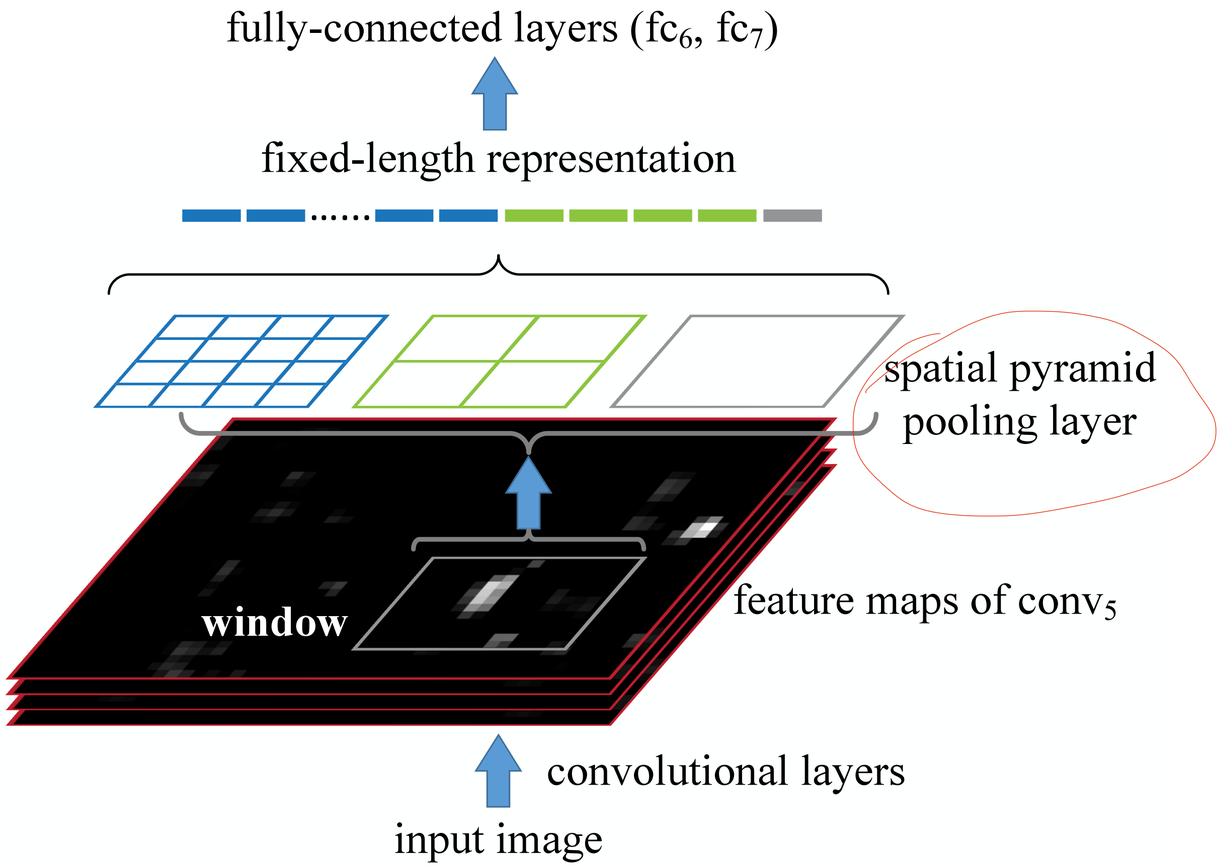
Fast R-CNN: RoI pooling

- Region-of-Interest pooling



Sermanet et al, "Integrated Recognition, Localization and Detection using Convolutional Networks", ICLR 2014

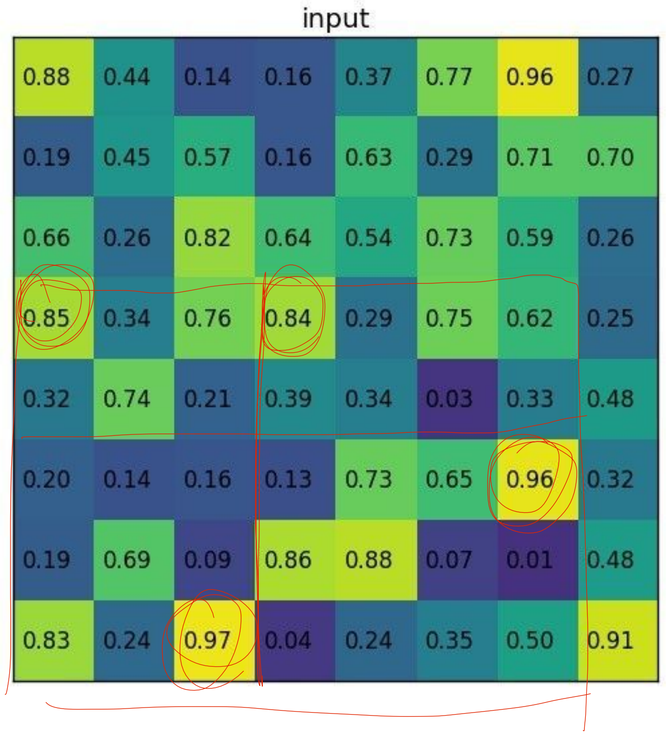
SPP-Net: Spatial Pyramid Pooling



He et al. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. ECCV 2014.

Spatial Pooling

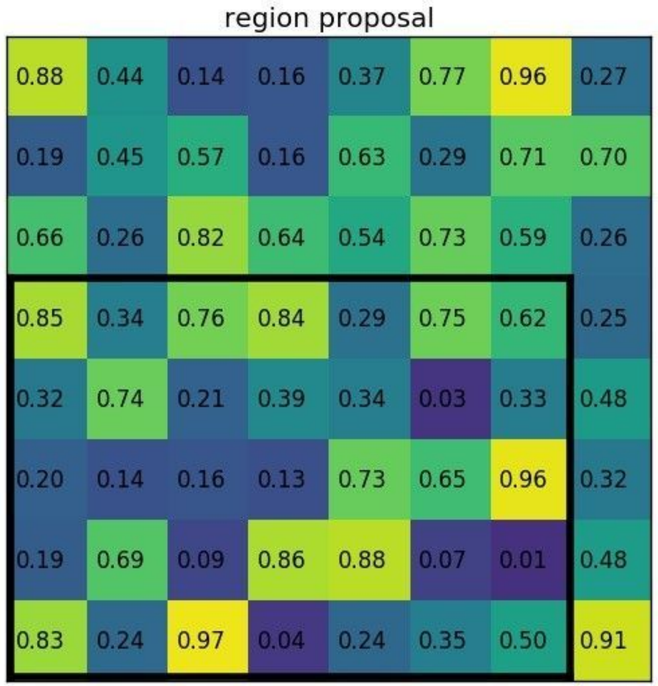
- Suppose we extract region proposal x=0, y=3, h = 5, w = 7



Credit: deepsense.ai

Spatial Pooling

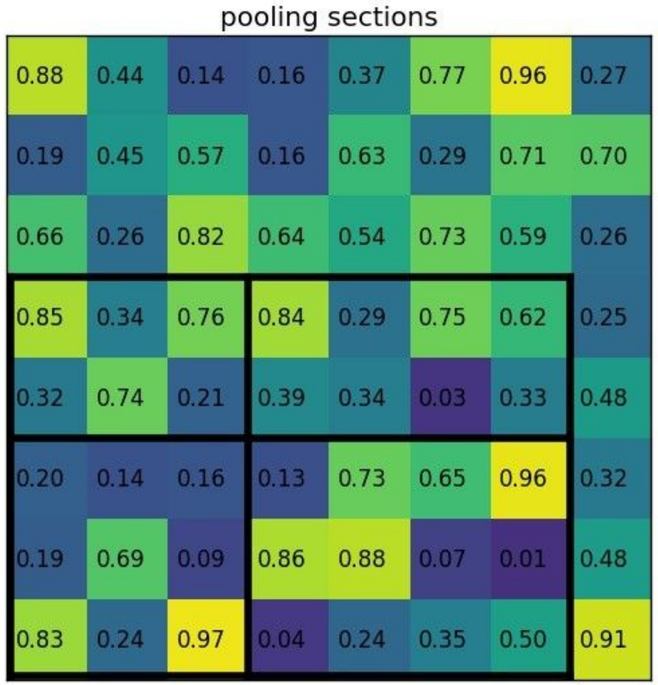
- Suppose we extract region proposal x=0, y=3, h = 5, w = 7



Credit: deepsense.ai

Spatial Pooling

- Suppose we extract region proposal x=0, y=3, h = 5, w = 7

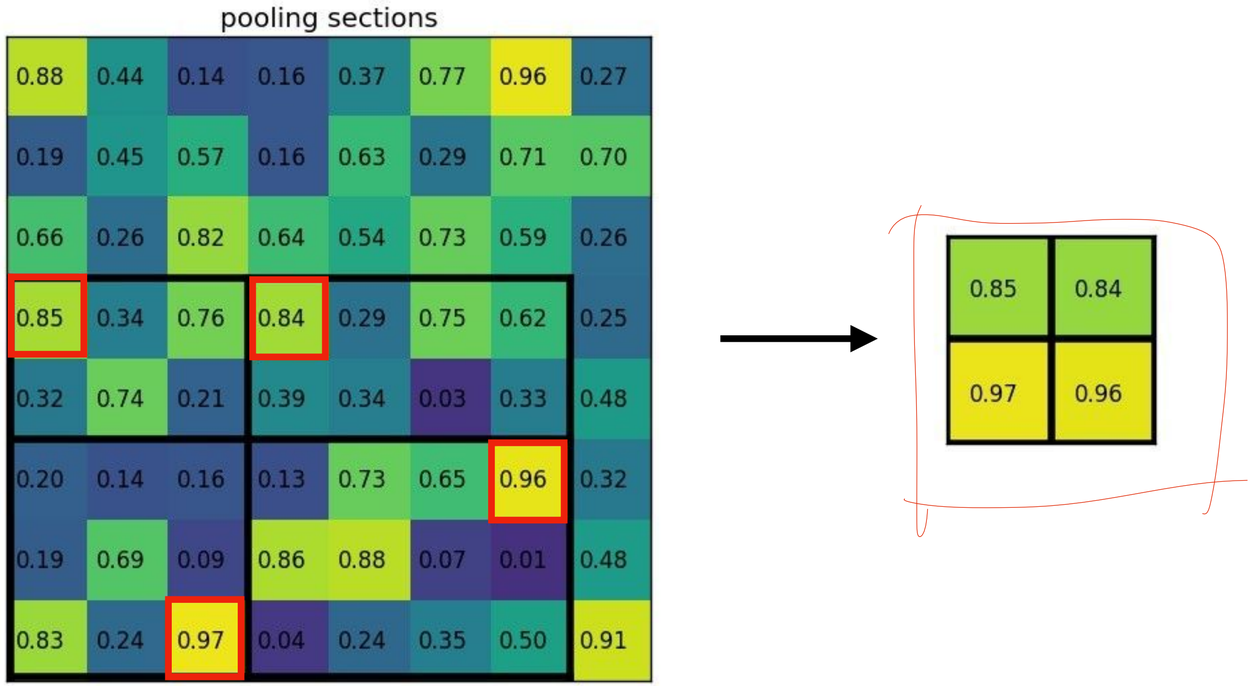


Discretise into 2x2 sub-windows

Credit: deepsense.ai

Spatial Pooling

- Suppose we extract region proposal x=0, y=3, h = 5, w = 7

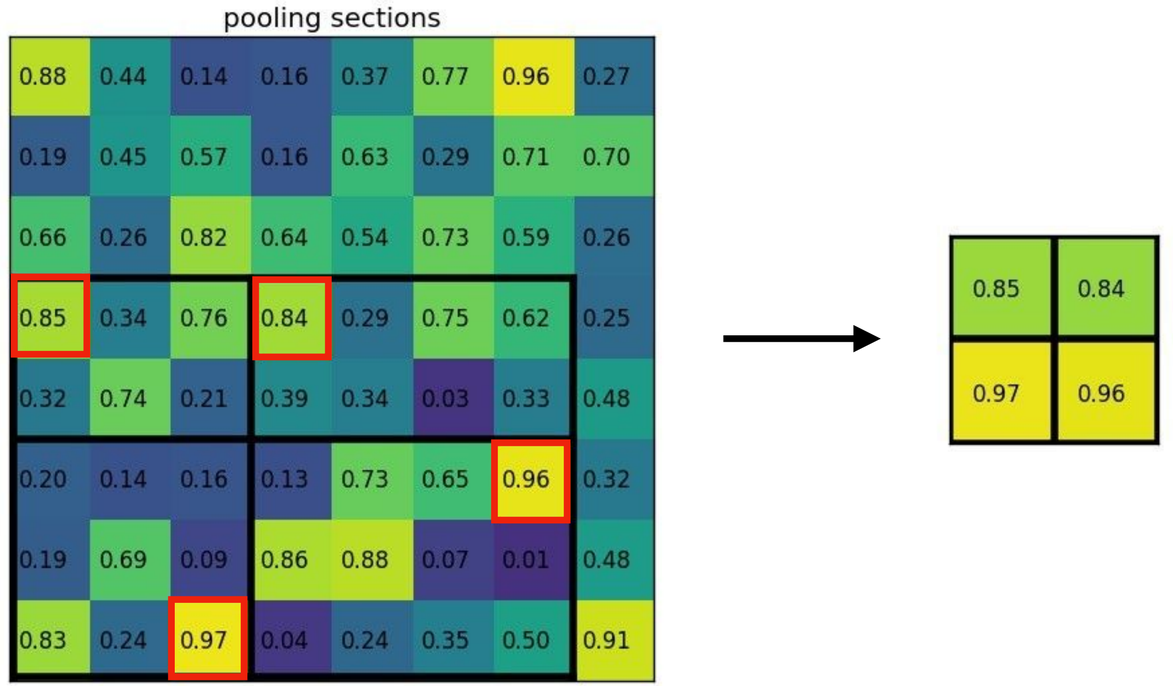


Max-pooling within each window

Credit: deepsense.ai

Spatial Pooling

- Suppose we extract region proposal x=0, y=3, h = 5, w = 7



Max-pooling within each window

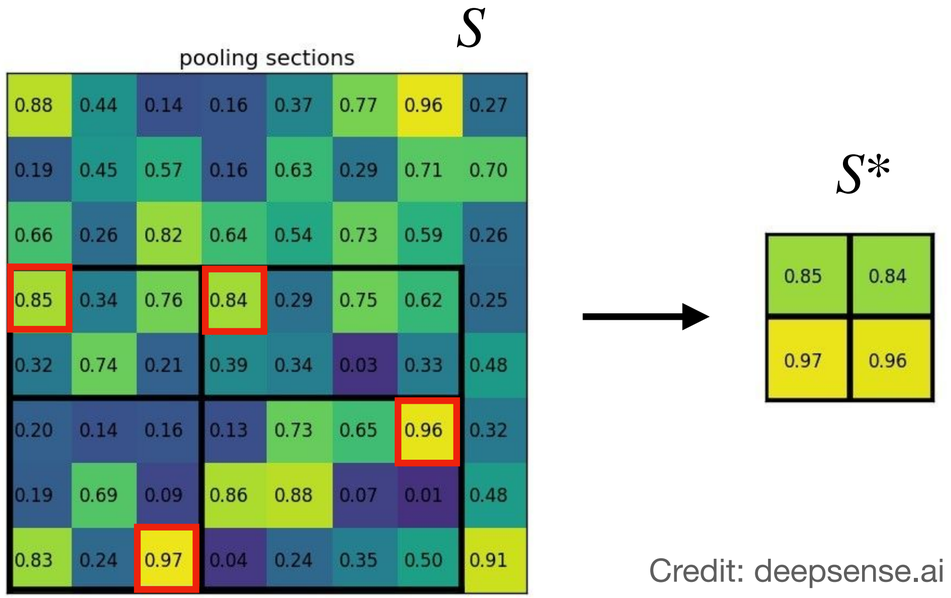
Quiz: Is it differentiable?

Credit: deepsense.ai

Spatial Pooling

- Is it differentiable?

- $S*$ is differentiable w.r.t. S ?
  - yes and no – depends on pooling
- $S*$ is differentiable w.r.t. (x, y, h, w) ?
  - no



Max-pooling within each window

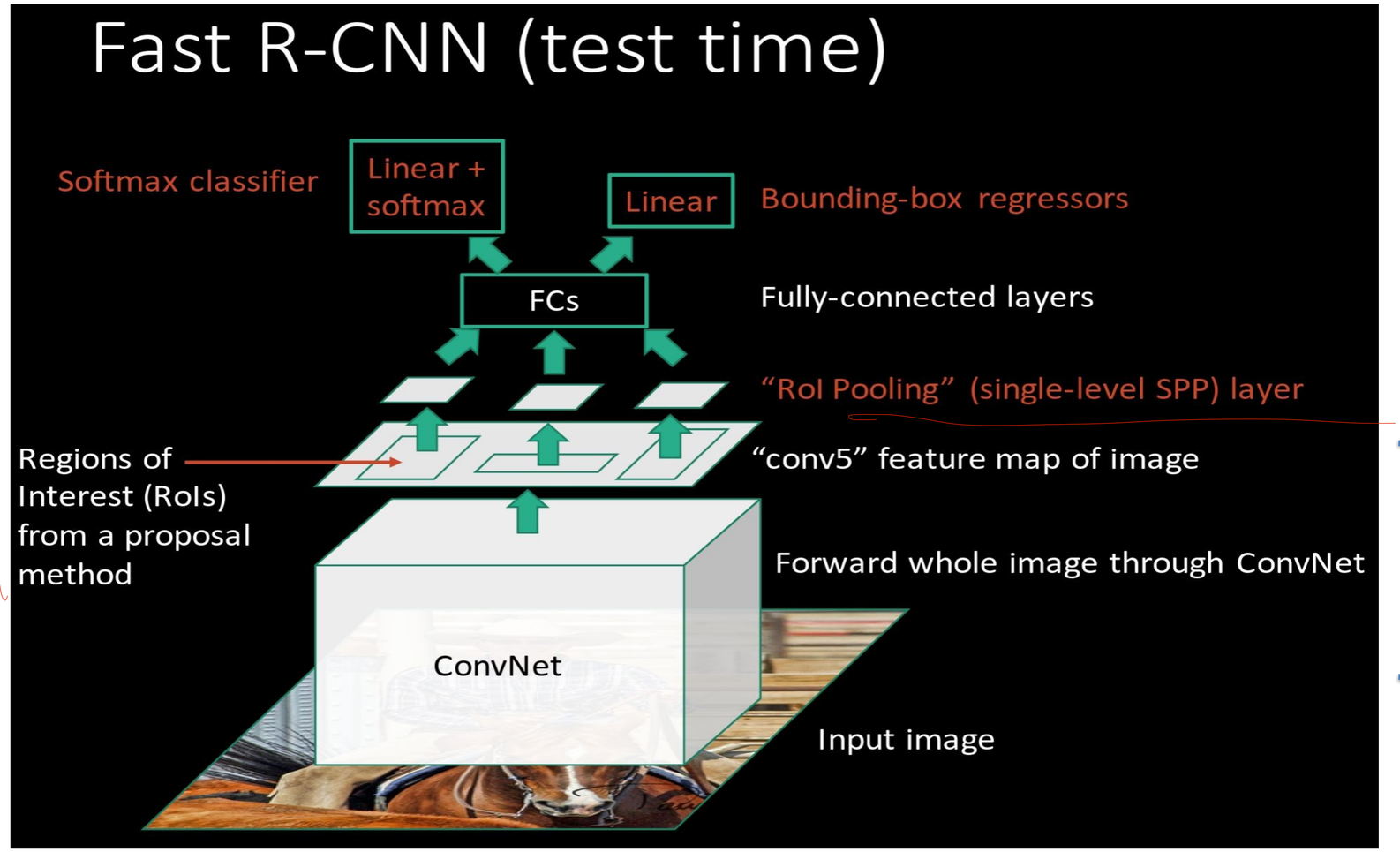
Credit: deepsense.ai

SPP-Net

- Faster training and testing than R-CNN
- Training scheme is still complex
- Still no end-to-end training (fixed convolutional layers)

- Integrate Spatial Pooling into R-CNN → Fast R-CNN

Fast R-CNN

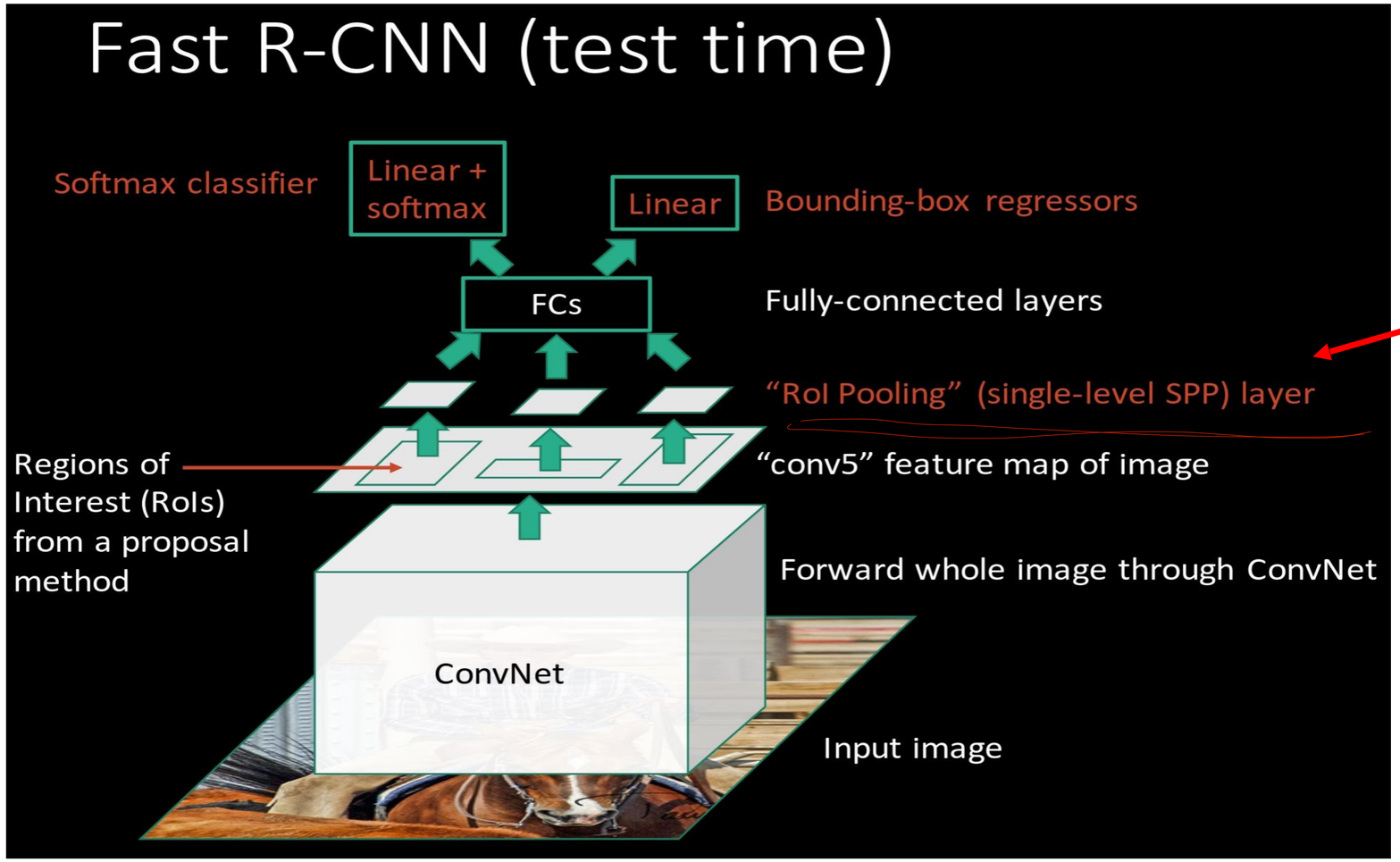


Shared computation at test time (like SPP)

Girschick, "Fast R-CNN", ICCV 2015

Slide credit: Ross Girschick

Fast R-CNN



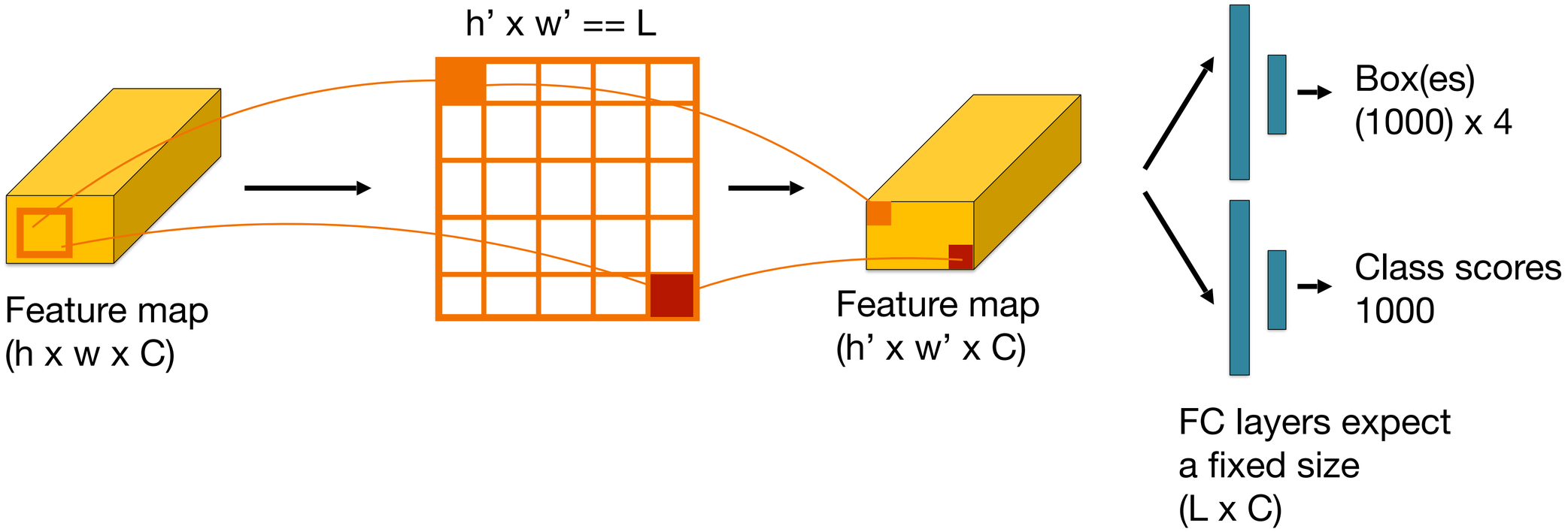
Region-of-Interest pooling

Girschick, "Fast R-CNN", ICCV 2015

Slide credit: Ross Girschick

Fast R-CNN: RoI pooling

- Region-of-Interest pooling



Sermanet et al, “Integrated Recognition, Localization and Detection using Convolutional Networks”, ICLR 2014

Fast R-CNN results

VGG-16 CNN on Pascal VOC 2007 dataset

	R-CNN	Fast R-CNN
Training Time:	84 hours	9.5 hours
Speed-up	1x	8.8x

Fast R-CNN results

VGG-16 CNN on Pascal VOC 2007 dataset

	R-CNN	Fast R-CNN
Training Time:	84 hours	9.5 hours
Speed-up	1x	8.8x
Test time per image	47 seconds	0.32 seconds
Speed-up	1x	146x

Making Fast R-CNN faster

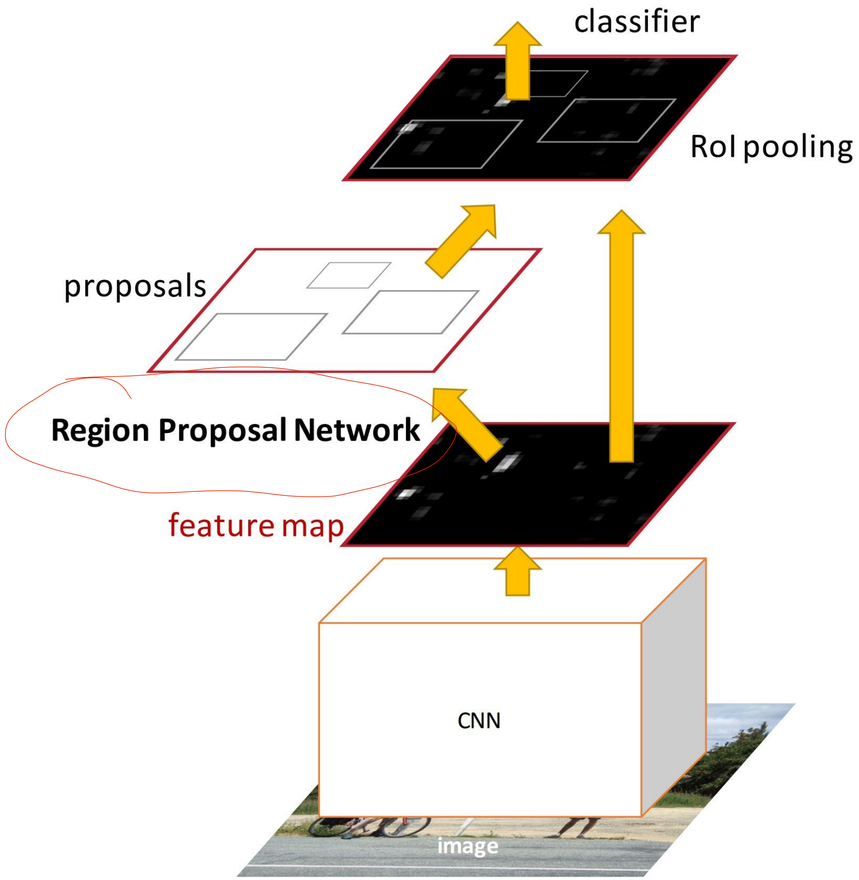
- We still rely on a standalone object proposals

独立提案

- Faster R-CNN: Integrated proposal generation
  - Region Proposal Network (RPN)

- Other than RPN, everything is like Fast R-CNN

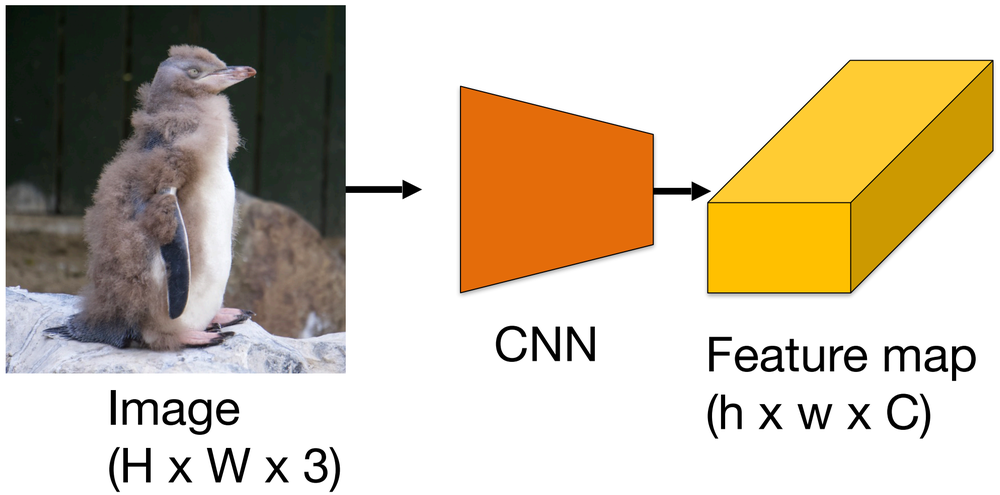
Faster R-CNN



Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

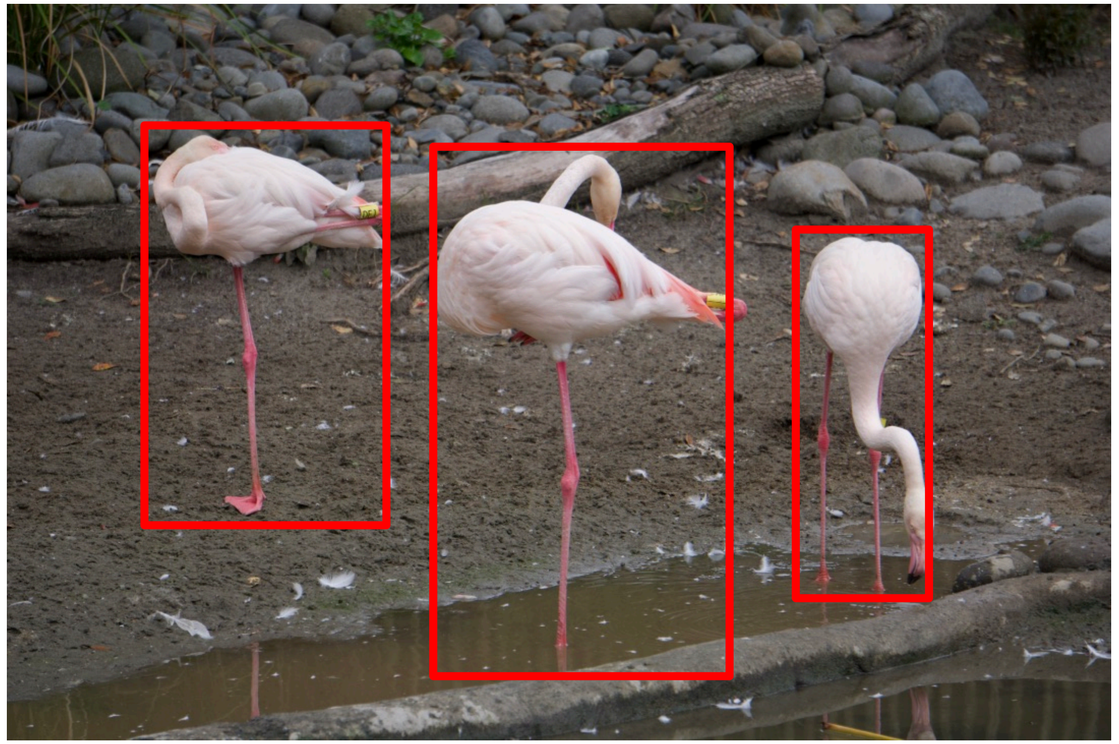
Slide credit: Ross Girschick

Region Proposal Network



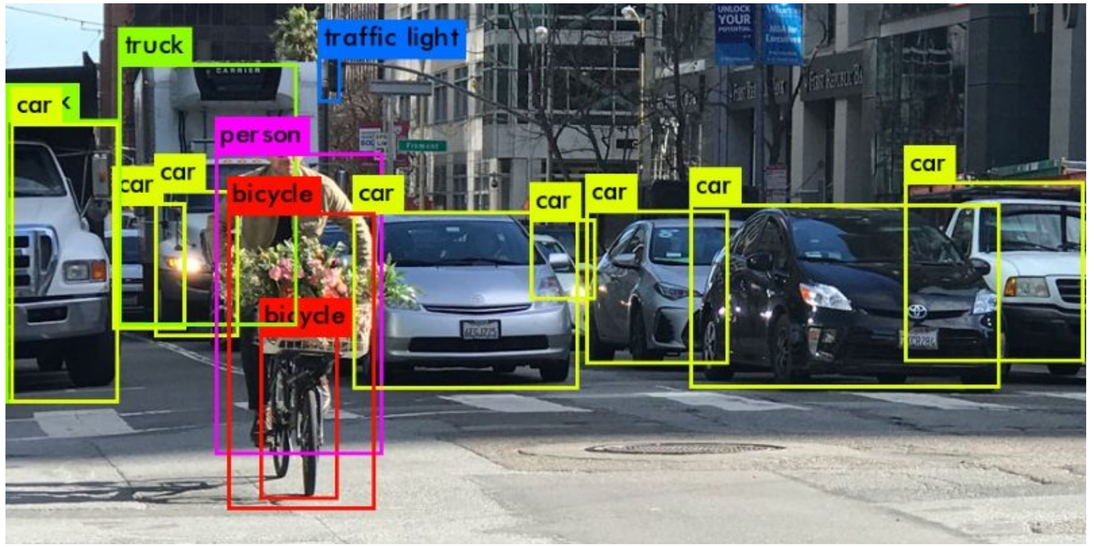
- How many proposals?
- How are they placed?

Multi-object detection



3 objects means having an output of 12 numbers (3 x 4)

Multi-object detection



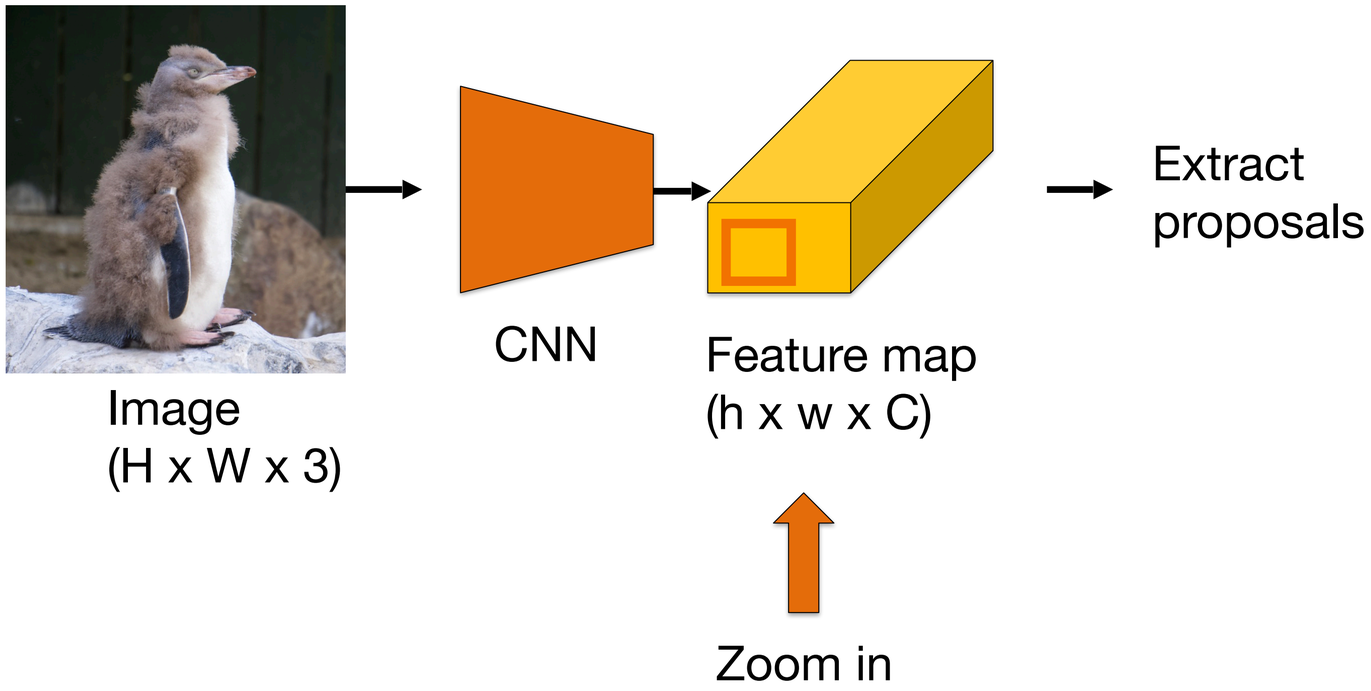
14 objects means having an output of 56 numbers (14 x 4)

Multi-object detection

- Dealing with variable-sized output is challenging
  - There are a couple of workarounds:
    - RNN: (Romera-Paredes and Torr, 2016; Ren et al., 2017; Araslanov et al., 2019).
    - Predict the # of objects: (e.g., Rezatofighi, 2018)

- RPN: Place multiple proposals uniformly densely
  - Learn to predict confidence for each proposal

Region Proposal Network

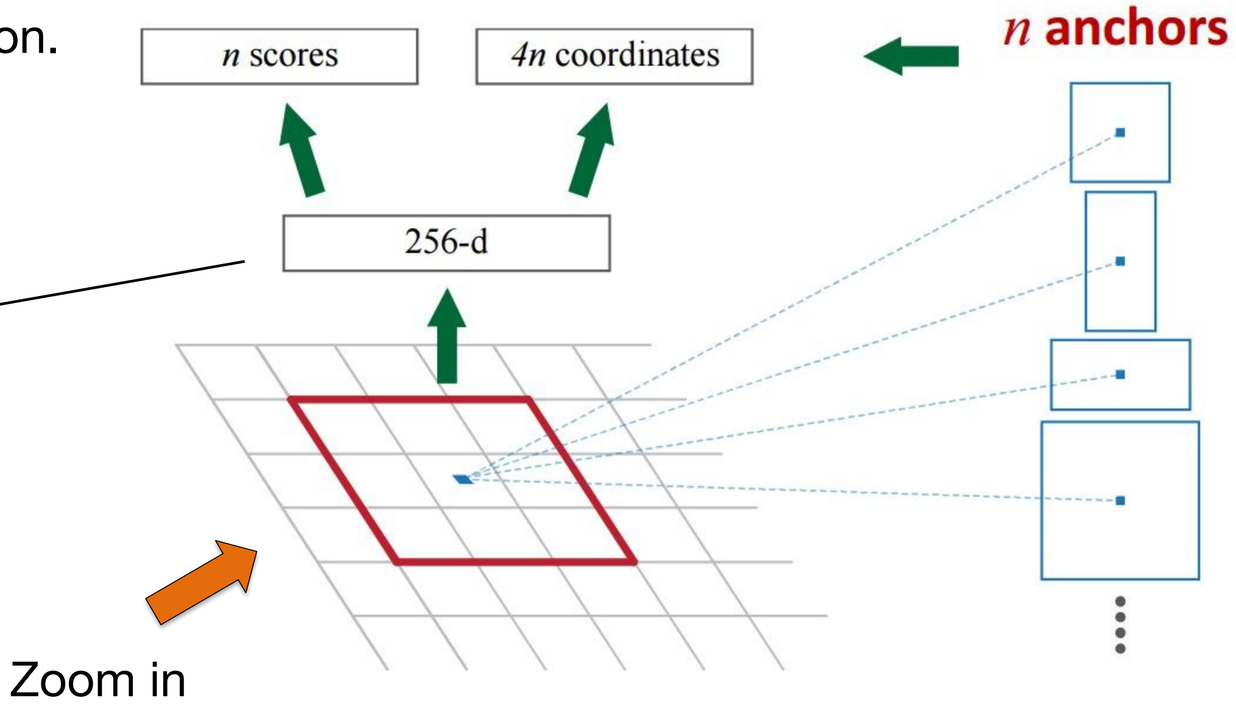


Region Proposal Network

We fix the number of proposals
by using a set of $n = 9$ anchors per location.

9 anchors = 3 scales x 3 aspect ratios

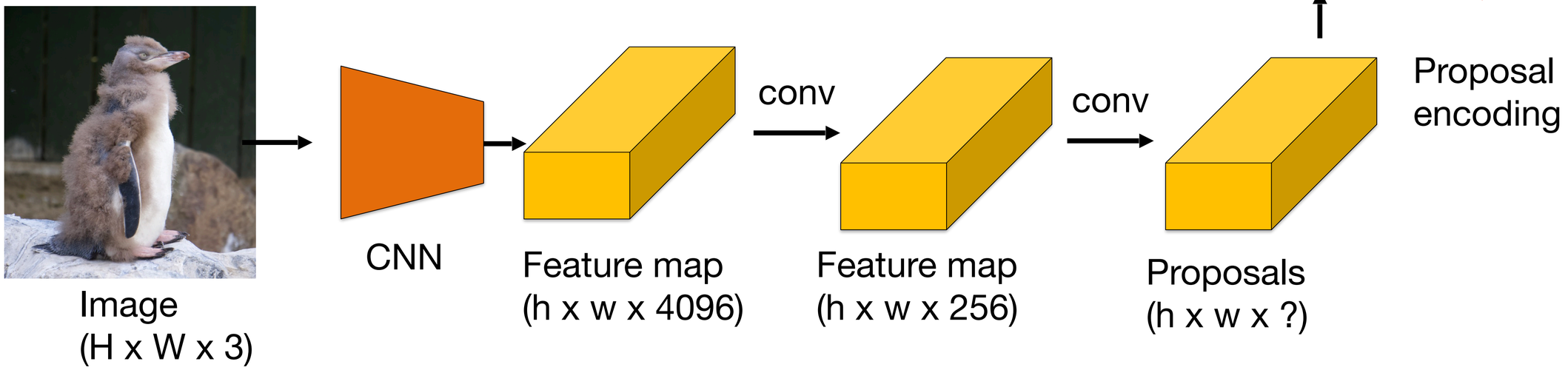
Every location is characterised by
a 256-d descriptor



Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

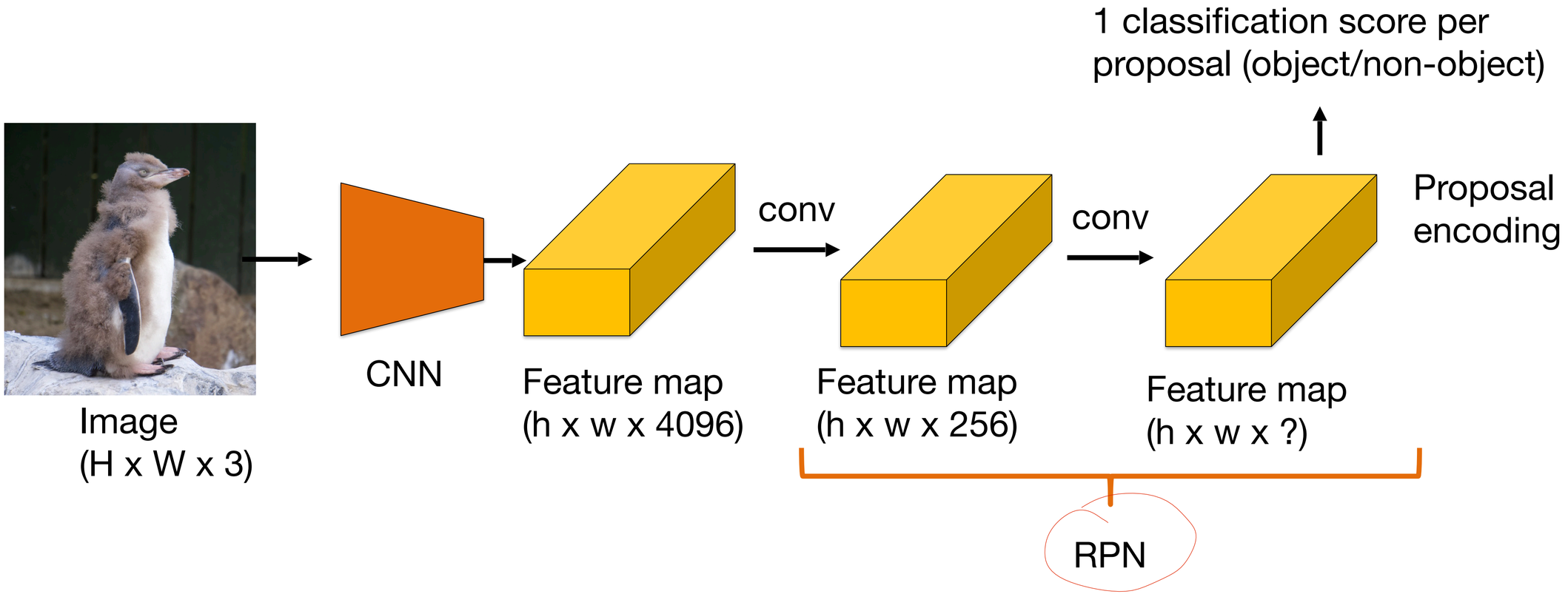
Slide credit: Ross Girschick

Region Proposal Network



QUIZ: What's the dimensionality of proposal encoding?
Suppose we have n anchors.

Region Proposal Network



Per feature map location, we get a set of anchor correction and classification into object/non-object

RPN: Training

- Classification ground truth: We compute p^* which indicates how much an anchor overlaps with the ground truth bounding boxes

$$p^* = 1 \quad if \quad \text{IoU} > 0.7$$

$$p^* = 0 \quad if \quad \text{IoU} < 0.3$$

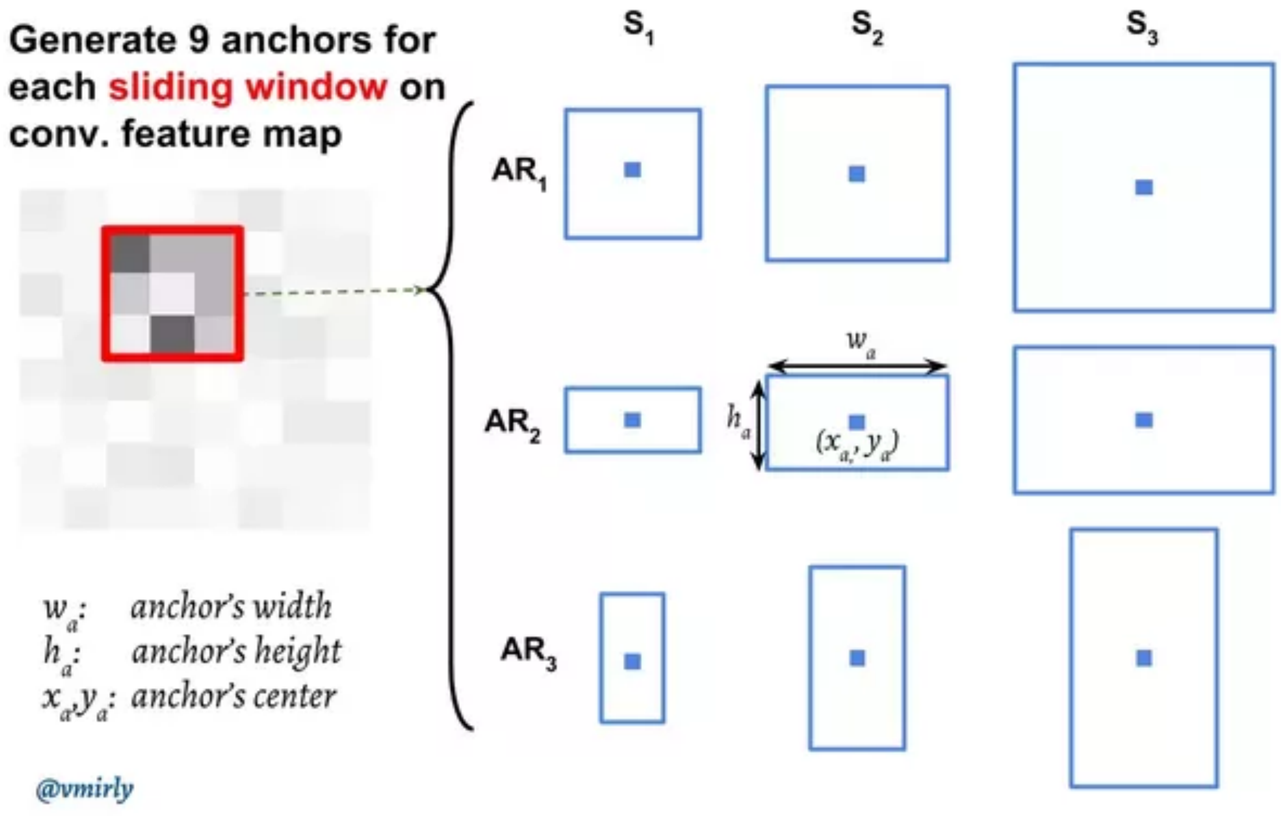
- 1 indicates the anchor represents an object (foreground) and 0 indicates a background object. The rest does not contribute to the training.

RPN: Training

- For an image, we randomly sample 256 anchors to form a mini-batch (balanced objects vs. non-objects)
- We learn anchor activate with the binary cross-entropy loss
- Those anchors that contain an object are used to compute the regression loss

RPN: Training

- Each anchor is described by the center position, width and height



RPN: Training

- Each anchor is described by the center position, width and height
- What the network actually predicts are

$$t_x = (x - x_a)/w_a, \qquad t_y = (y - y_a)/h_a,$$

Normalized horizontal shift — Normalized y

$$t_w = log(w/w_a), \qquad t_h = log(h/h_a),$$

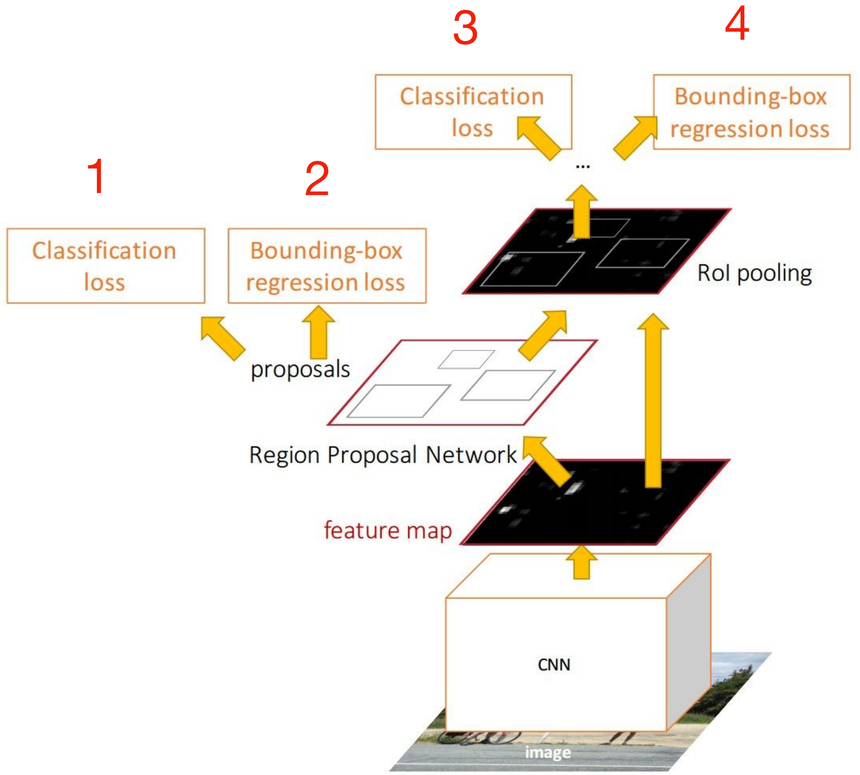
Normalized width — Normalized height

- Smooth L1 loss on regression targets.

Faster R-CNN: Training

- First implementation, training of RPN separate from the rest.
- Now we can train jointly!

- Four losses:
  1. RPN classification (object/non-object)
  2. RPN regression (anchor → proposal)
  3. Fast R-CNN classification (type of object)
  4. Fast R-CNN regression (proposal → box)



Faster R-CNN: Training

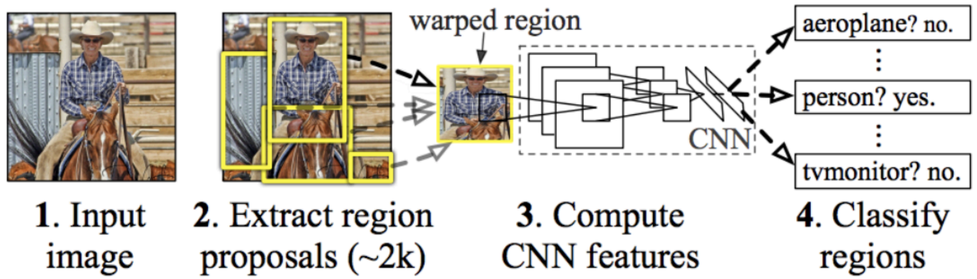
- 10x faster at test time w.r.t. Fast R-CNN
- Trained end-to-end including feature extraction, region proposals, classifier and regressor
- RPN is fully convolutional
- More accurate, since proposals are learned

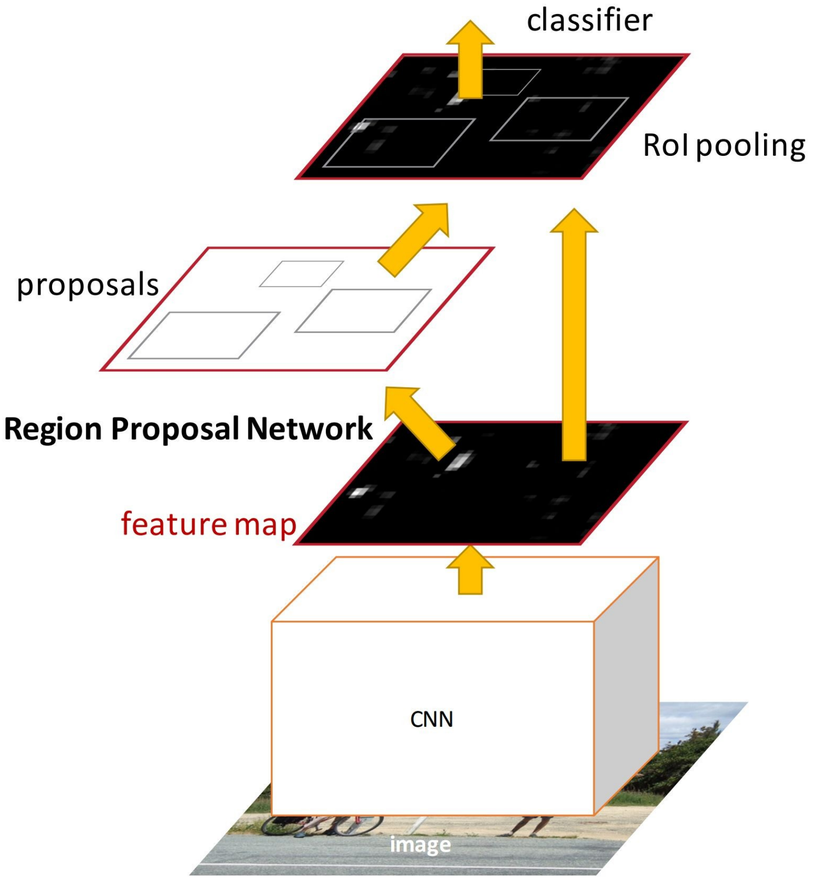
Faster R-CNN: Training

VGG-16 CNN on Pascal VOC 2007 dataset

	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image (w/ proposals, s)	50	2	0.2
Speed-up	1x	8.8x	250x
mAP	66.0	66.9	66.9

From R-CNN to Faster R-CNN





- More efficient and more accurate.
- Incremental but powerful improvements.
- Ent-to-end training: larger benefits from more data

What we've learned today

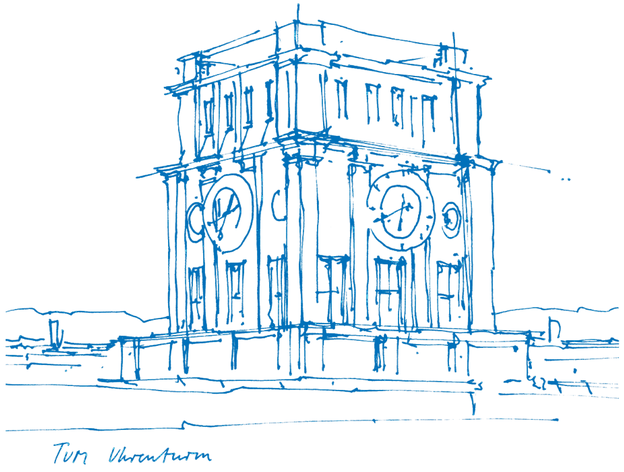
- Recap: Feature-based detection;
- Overfeat: An early deep net for object detection;
- Non-maximum Supression (NMS);
- Intersection-over-Union (IoU);
- Object proposals for two-stage detection;
- Spatial Pyramid Pooling (SPP);
- (Fast/Faster) R-CNN; Region Proposal Network (RPN).

Computer Vision III:

Two-stage object detectors

Dr. Nikita Araslanov
24.10.2023

Content credit:
Prof. Laura Leal-Taixé
https://dvl.in.tum.de



Related works

- Shrivastava, Gupta, Girshick. “Training region-based object detectors with online hard example mining”. CVPR 2016.
- Dai, Li, He and Sun. “R-FCN: Object detection via region-based fully convolutional networks”. 2016.
- Dai, Qi, Xiong, Li, Zhang, Hu and Wei. “Deformable convolutional networks”. ICCV 2017.
- Lin, Dollar, Girshick, He, Hariharan and Belongie. “Feature Pyramid Networks for object detection”. CVPR 2017.