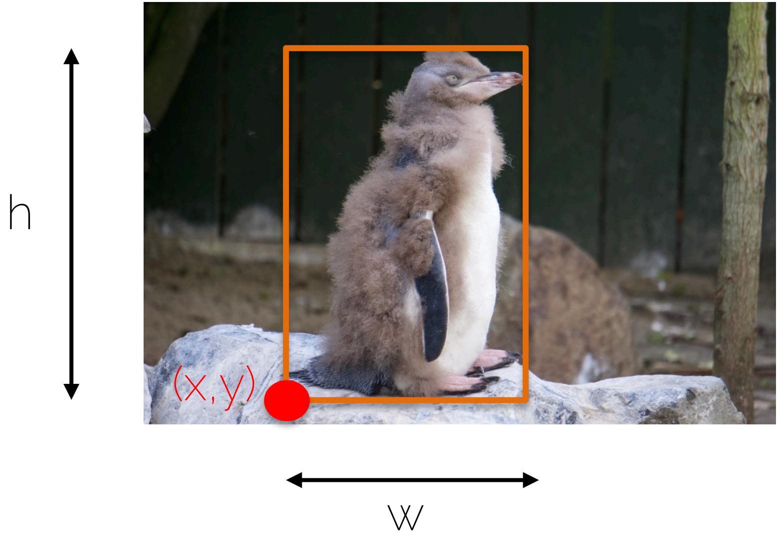
Lecture 1 recap

Task definition

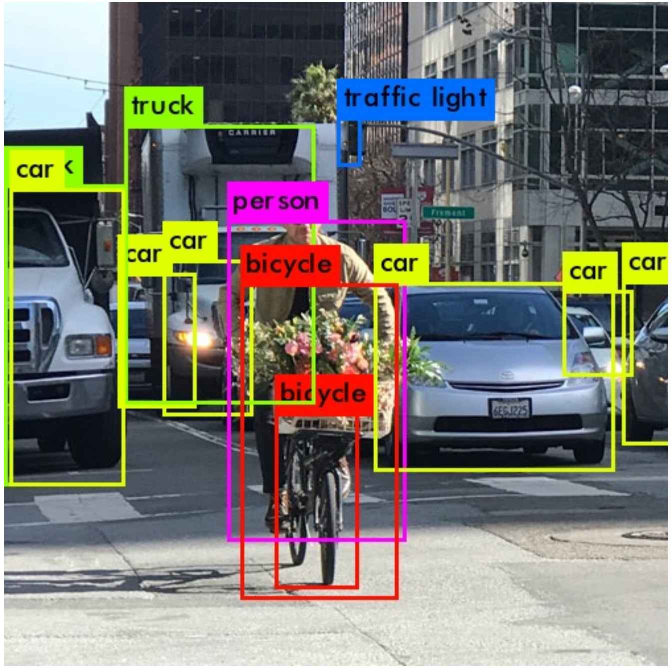
- Object detection problem



Bounding box. (x,y,w,h)

Task definition

- Object detection problem



Bounding box. (x,y,w,h)

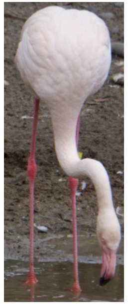
+

class

Traditional object detection methods

- 1. Template matching + sliding window

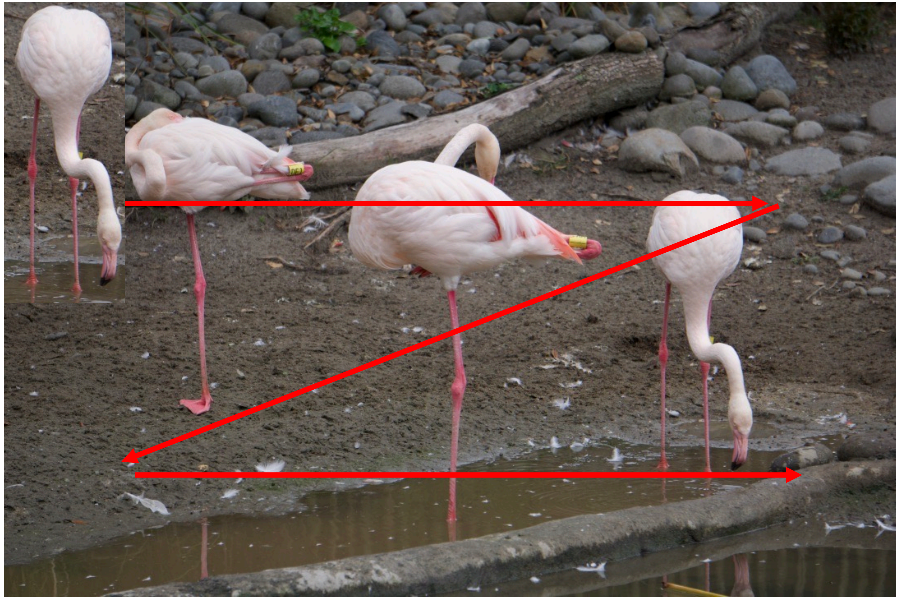
Image



Template

Traditional object detection methods

- 1. Template matching + sliding window



Image

Traditional object detection methods

- 1. Template matching + sliding window

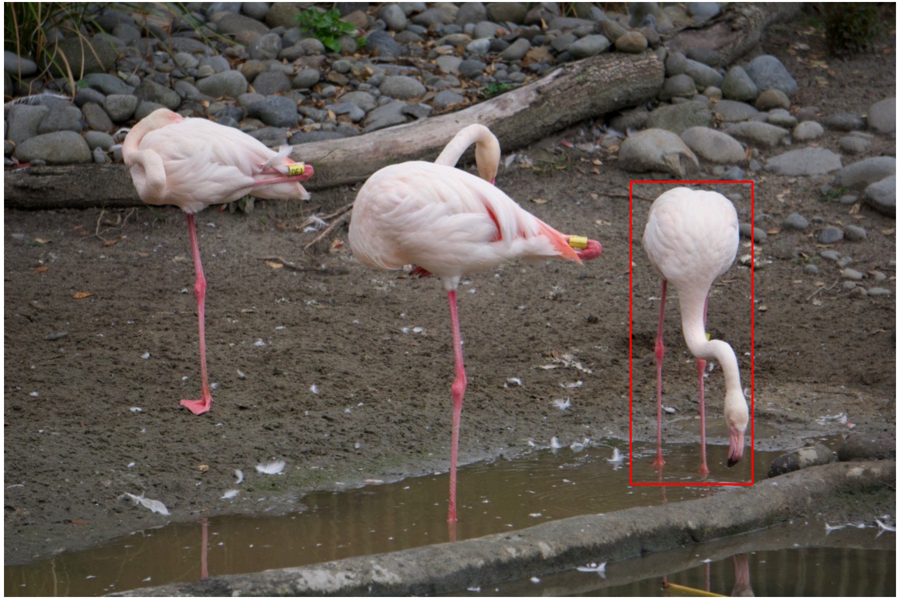
LOW correlation

Image

For every position you evaluate how much do the pixels in the image and template correlate

Traditional object detection methods

- 1. Template matching + sliding window



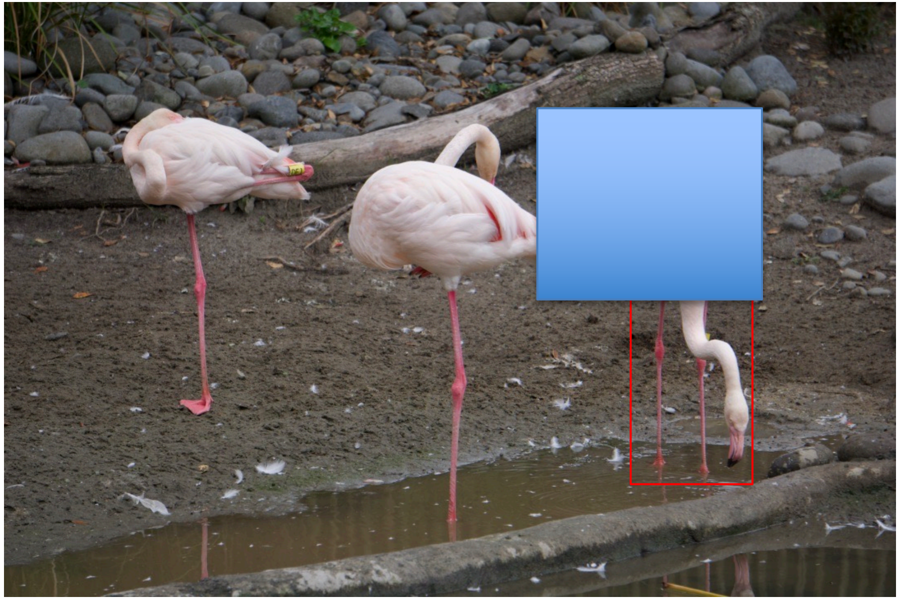
Image

HIGH correlation

For every position you evaluate how much do the pixels in the image and template correlate

Traditional object detection methods

- Problems of 1. Template matching + sliding window



For every position you evaluate how much do the pixels in the image and template correlate

Image

LOW correlation

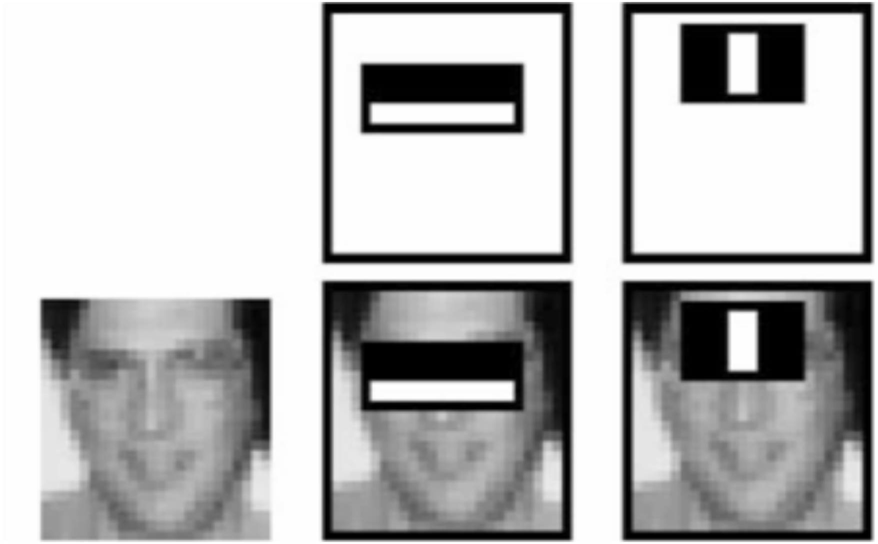
Viola-Jones detector

- 2. Feature extraction + classification
  - Learning multiple weak learners to build a strong classifier
  - That is, make many small decisions and combine them for a stronger final decision

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

Viola-Jones detector

- 2. Feature extraction + classification

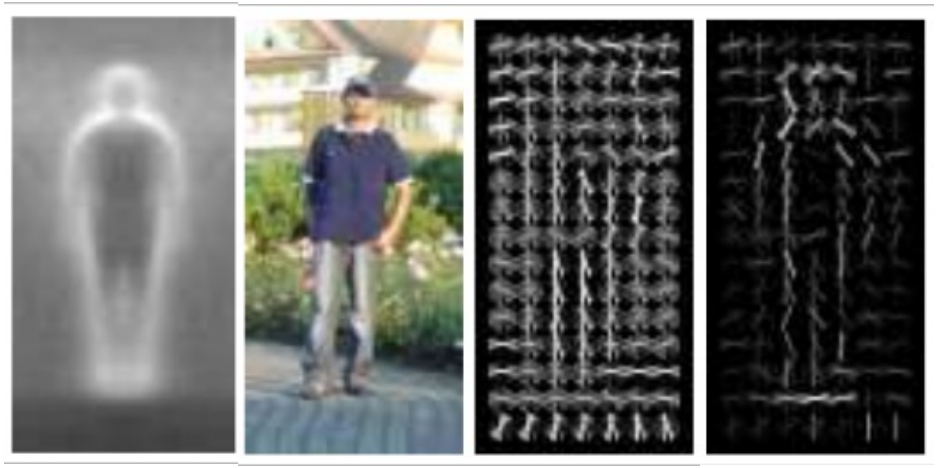


Haar features

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

Histogram of Oriented Gradients

- 2. Feature extraction + classification

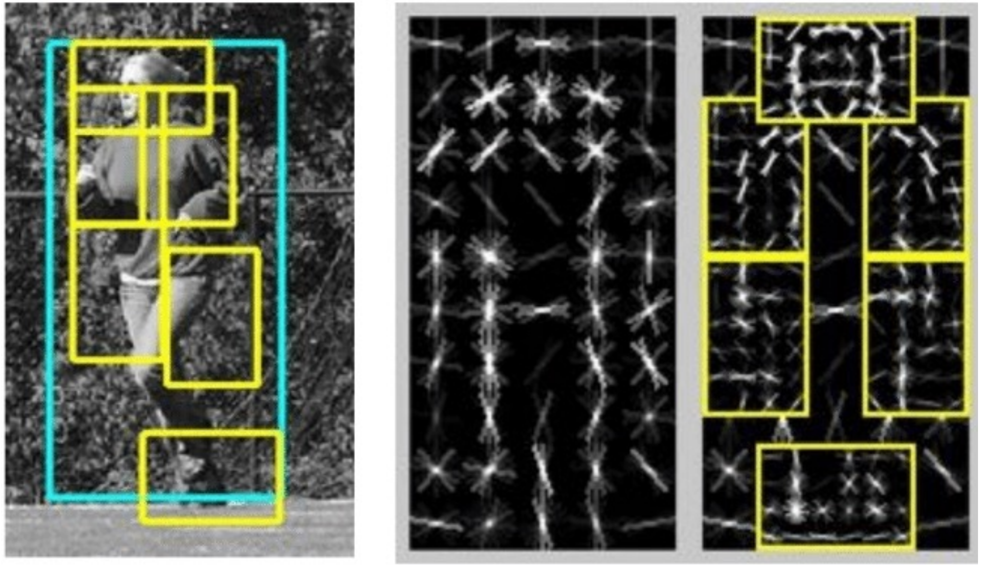


Features = histogram of oriented gradients
Classifier = Support Vector Machine (SVM)

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

Deformable Part Model

- Also based on HOG features, but based on body part detection → more robust to different body poses

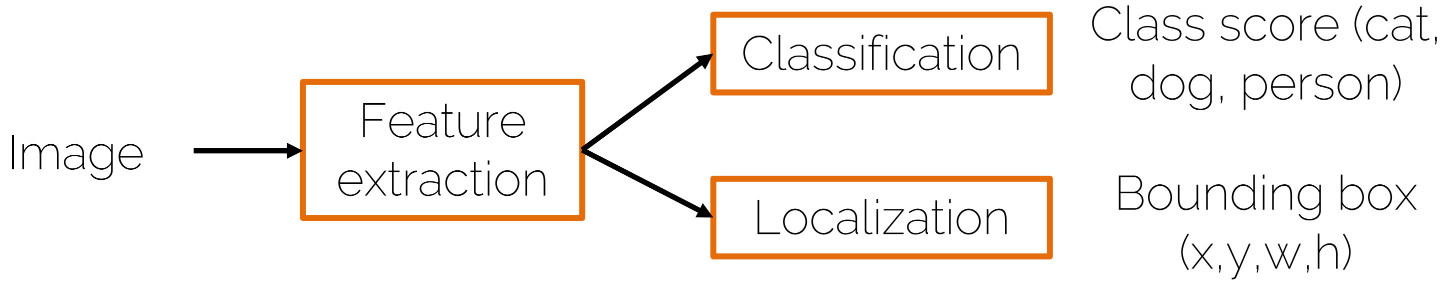


Felzenszwalb et al. A discriminatively trained, multiscale, deformable part model. CVPR 2008.

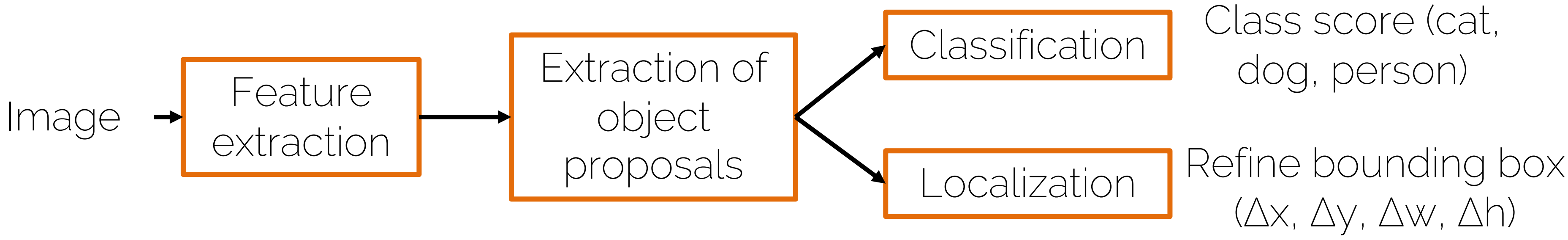
Two-stage object detectors

Types of object detectors

- One-stage detectors

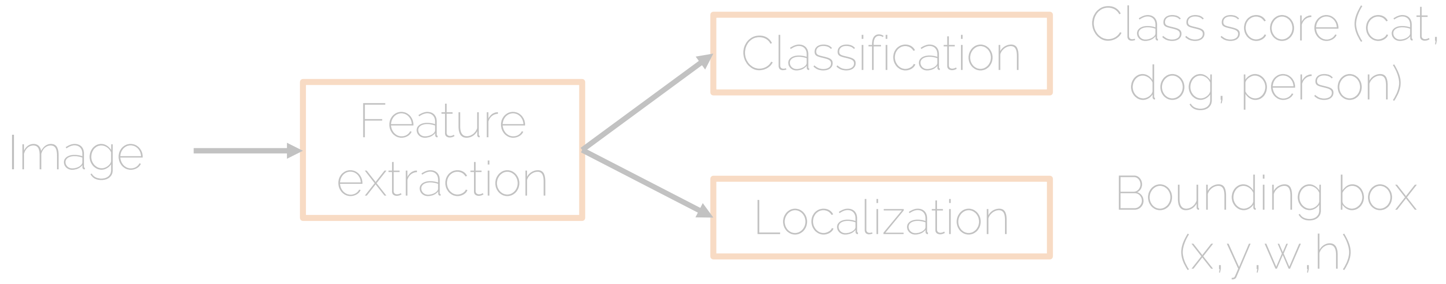


- Two-stage detectors

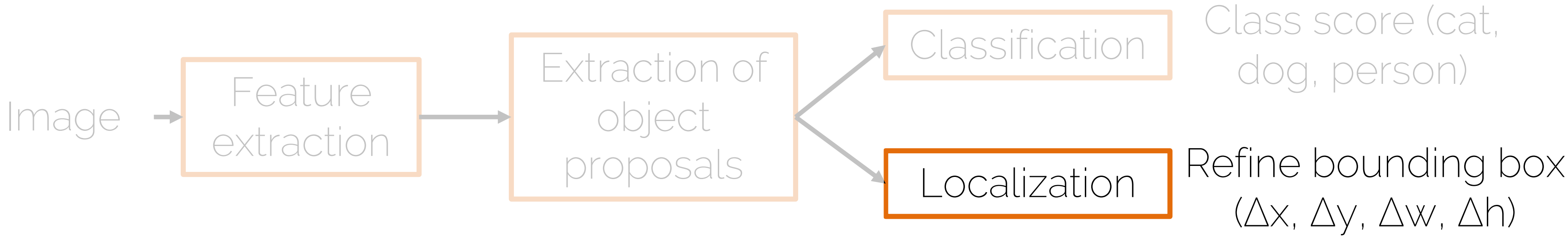


Types of object detectors

- One-stage detectors



- Two-stage detectors



Localization

- Bounding box regression

Image

Feature extraction (this time with a Neural Network)

Output:
Box coordinates (x,y,w,h)

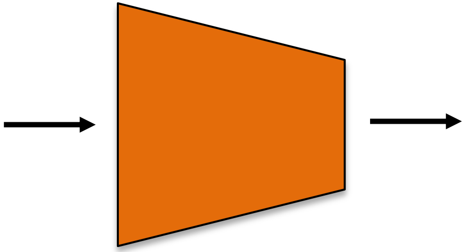
L2 loss function

Ground truth: Box coordinates

Localization

- Bounding box regression

Image



Convolutional
Neural Network

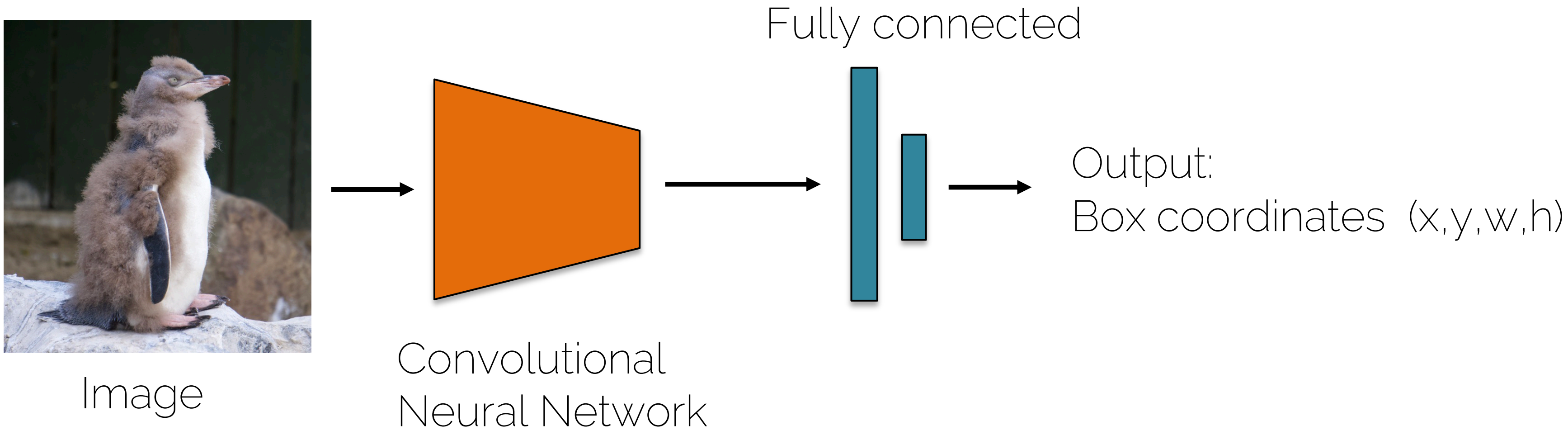
Output:
Box coordinates (x,y,w,h)

L2 loss function

Ground truth: Box
coordinates

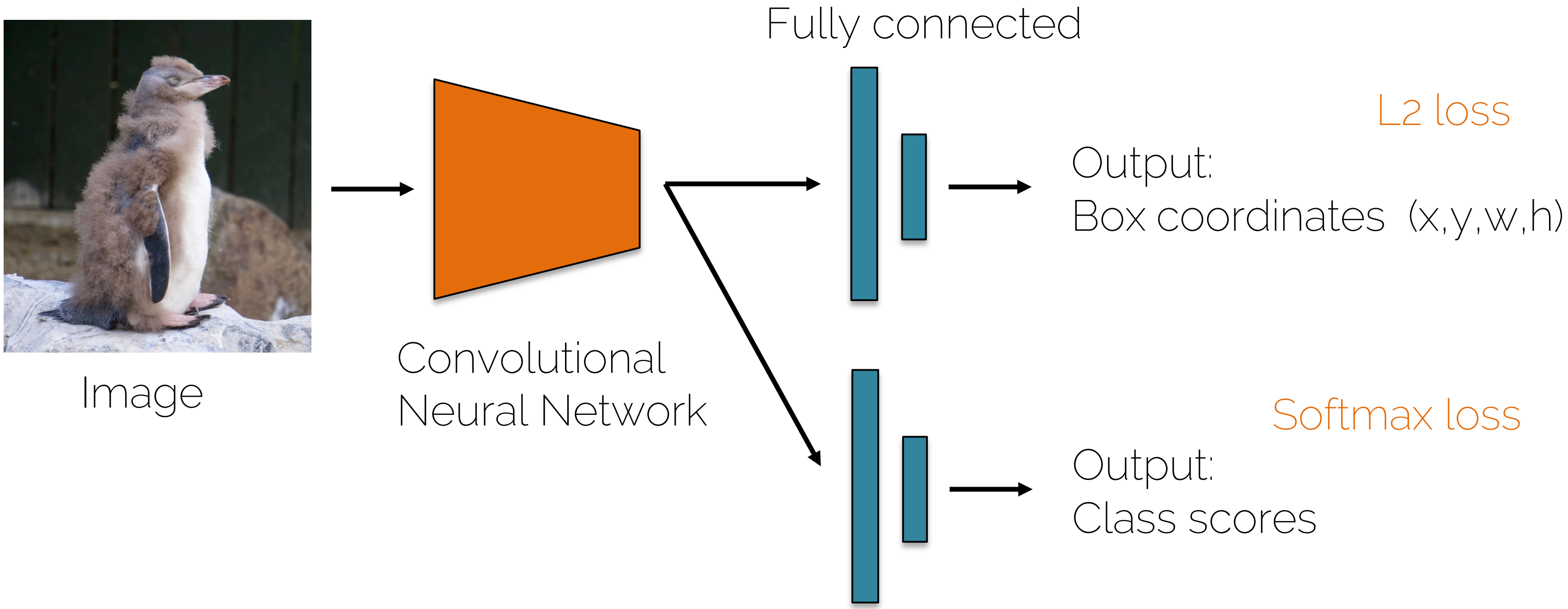
Localization and classification

- Bounding box regression



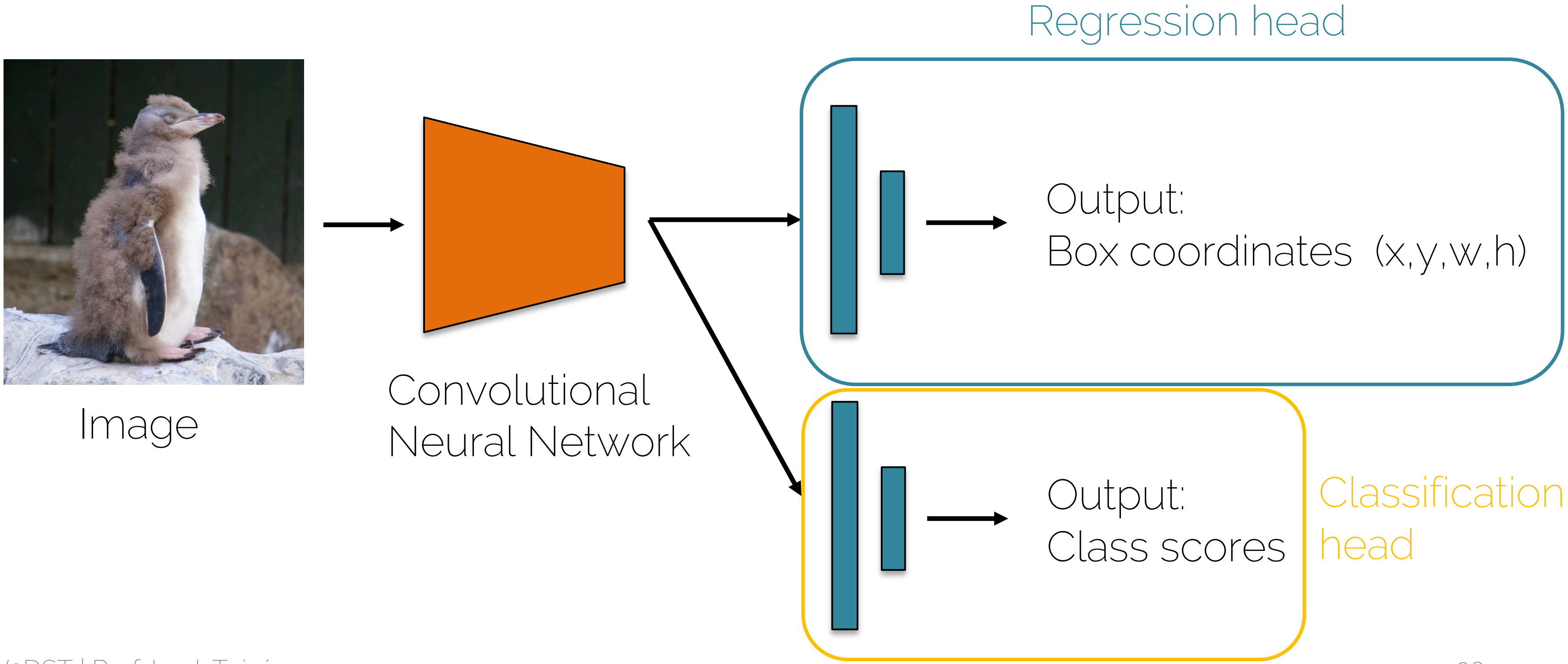
Localization and classification

- Bounding box regression



Localization and classification

- Bounding box regression



Localization and classification

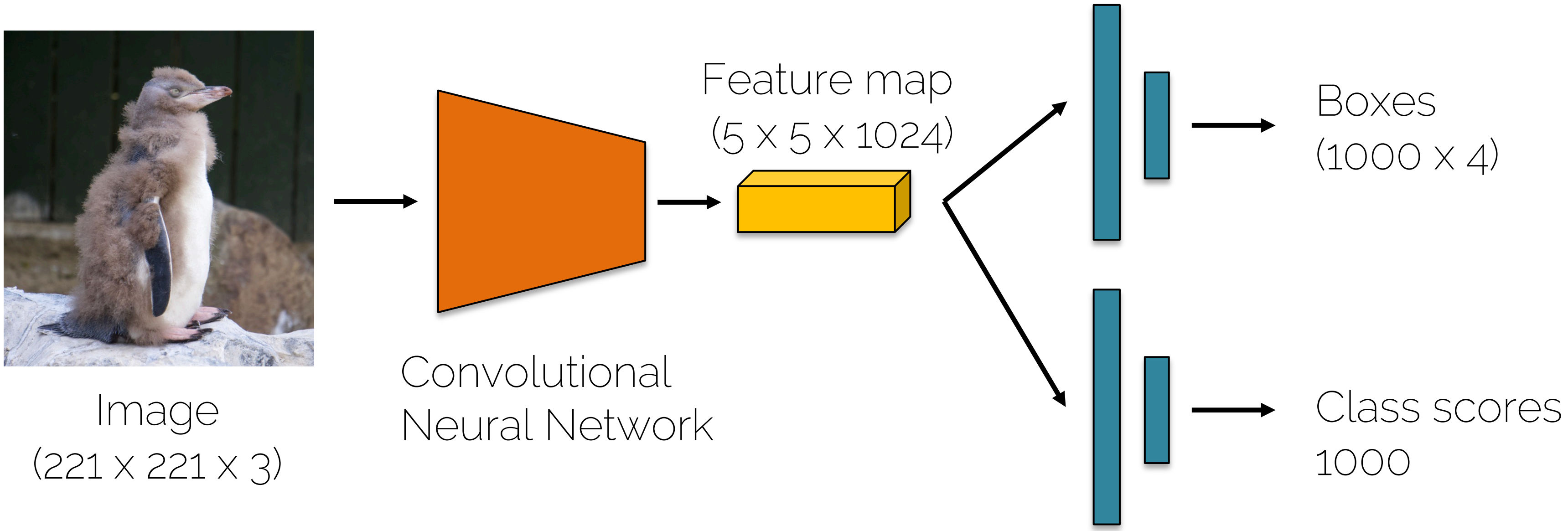
- It was typical to train the classification head first, freeze the layers
- Then train the regression head

- At test time, we use both!

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

Overfeat

- Sliding window + box regression + classification



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

Overfeat

- Sliding window + box regression + classification

Image (468 x 356 x 3)

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

Overfeat

- Sliding window + box regression + classification

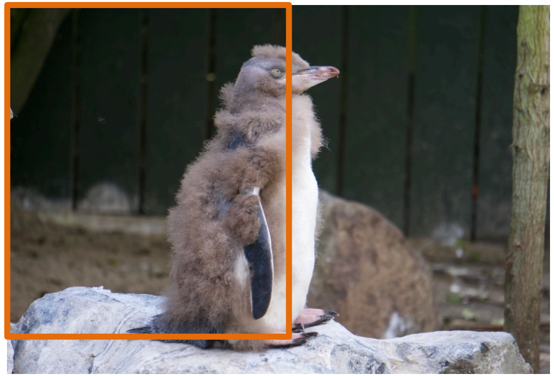


Image (468 x 356 x 3)

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

Overfeat

- Sliding window + box regression + classification

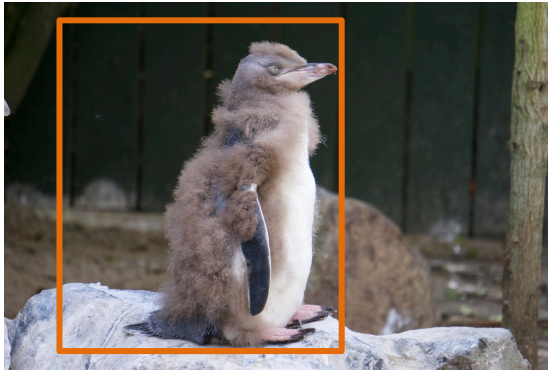


Image (468 x 356 x 3)

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

Overfeat

- Sliding window + box regression + classification

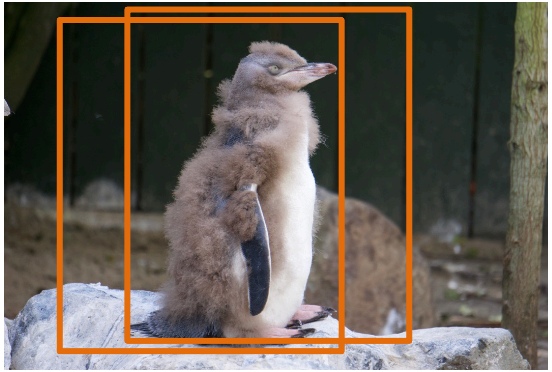


Image (468 x 356 x 3)

Sermanet et al, "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 (in Overfeat they have a greedy method)

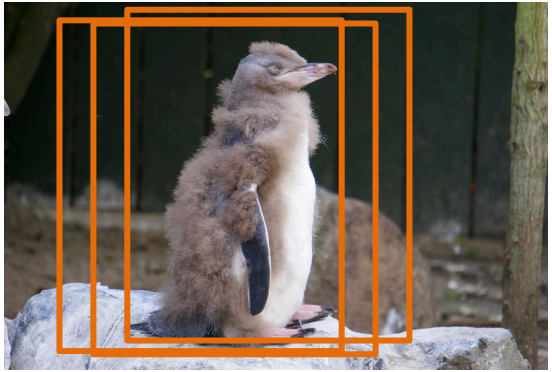


Image (468 x 356 x 3)

Sermanet et al, "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 (in Overfeat they have a greedy method)

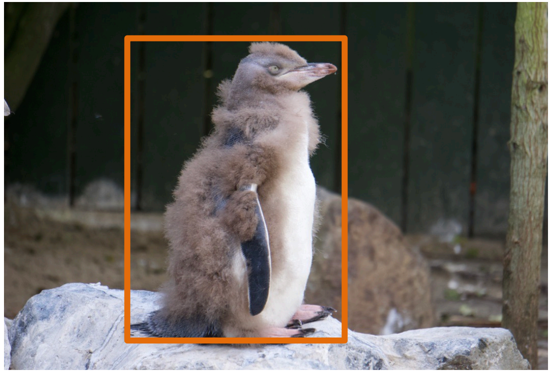


Image (468 x 356 x 3)

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

Non-Maximum Suppression (NMS)

Algorithm 1 Non-Max Suppression

procedure NMS(B,c)
 $B_{nms} \leftarrow \emptyset$
 for $b_i \in B$ **do** — Start with anchor box i
  $discard \leftarrow$ False
  for $b_j \in B$ **do** — For another box j
   if same$(b_i, b_j) > \boldsymbol{\lambda}_{\mathbf{nms}}$ **then** — If they overlap
    if score$(c, b_j) >$ score(c, b_i) **then**
     $discard \leftarrow$ True — Discard box i if the score is lower than the score of j
  if not $discard$ **then**
   $B_{nms} \leftarrow B_{nms} \cup b_i$
 return B_{nms}

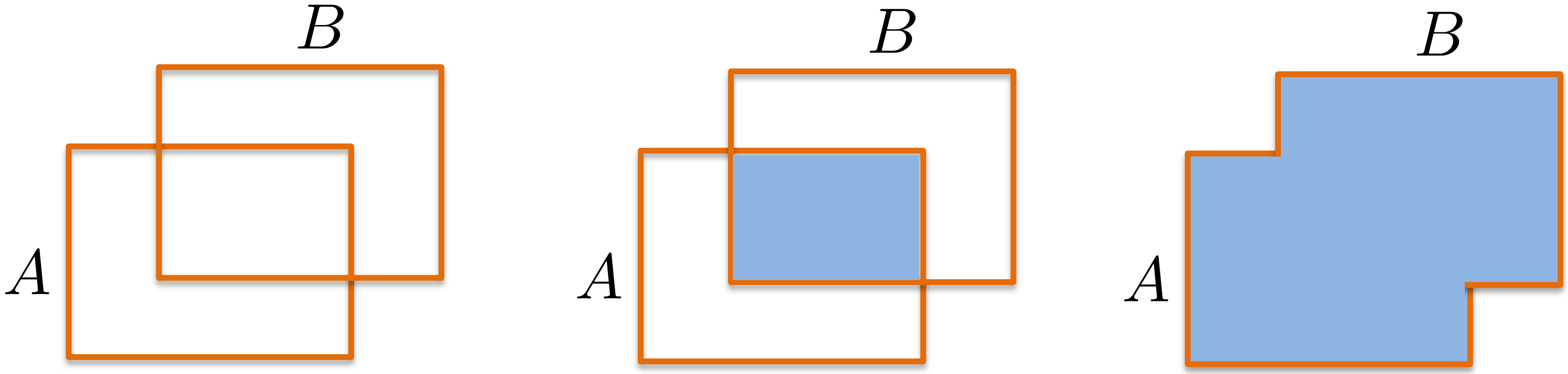
Overlap = to be defined

Score = depends on the task

Region overlap

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

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



Intersection Union

Non-Maximum Suppression (NMS)

Algorithm 1 Non-Max Suppression

procedure NMS(B,c)
  $B_{nms} \leftarrow \emptyset$
  for $b_i \in B$ **do** — Start with anchor box i
    $discard \leftarrow$ False
    for $b_j \in B$ **do** — For another box j
      if $\text{same}(b_i, b_j) > \boxed{\lambda_{\mathbf{nms}}}$ **then** — If they overlap
        if $\text{score}(c, b_j) > \text{score}(c, b_i)$ **then**
          $discard \leftarrow$ True — Discard box i if the score is lower than the score of j
    if not $discard$ **then**
      $B_{nms} \leftarrow B_{nms} \cup b_i$
  return B_{nms}

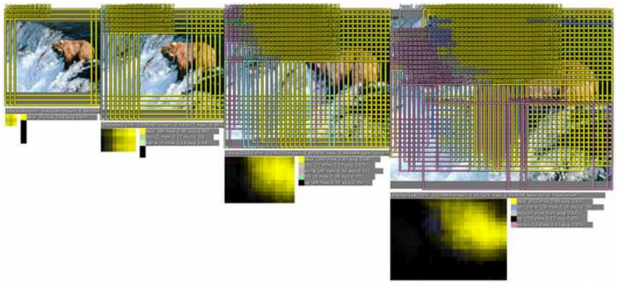
Overlap = to be defined

Score = depends on the task

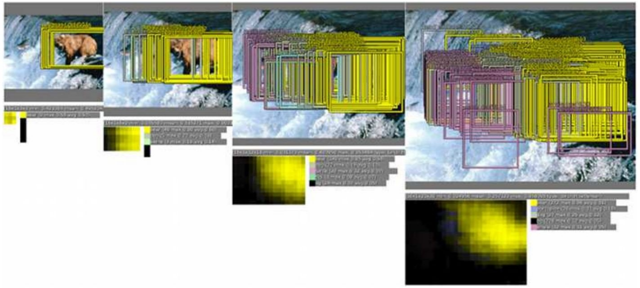
Overfeat

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

Window positions + score maps



Box regression outputs

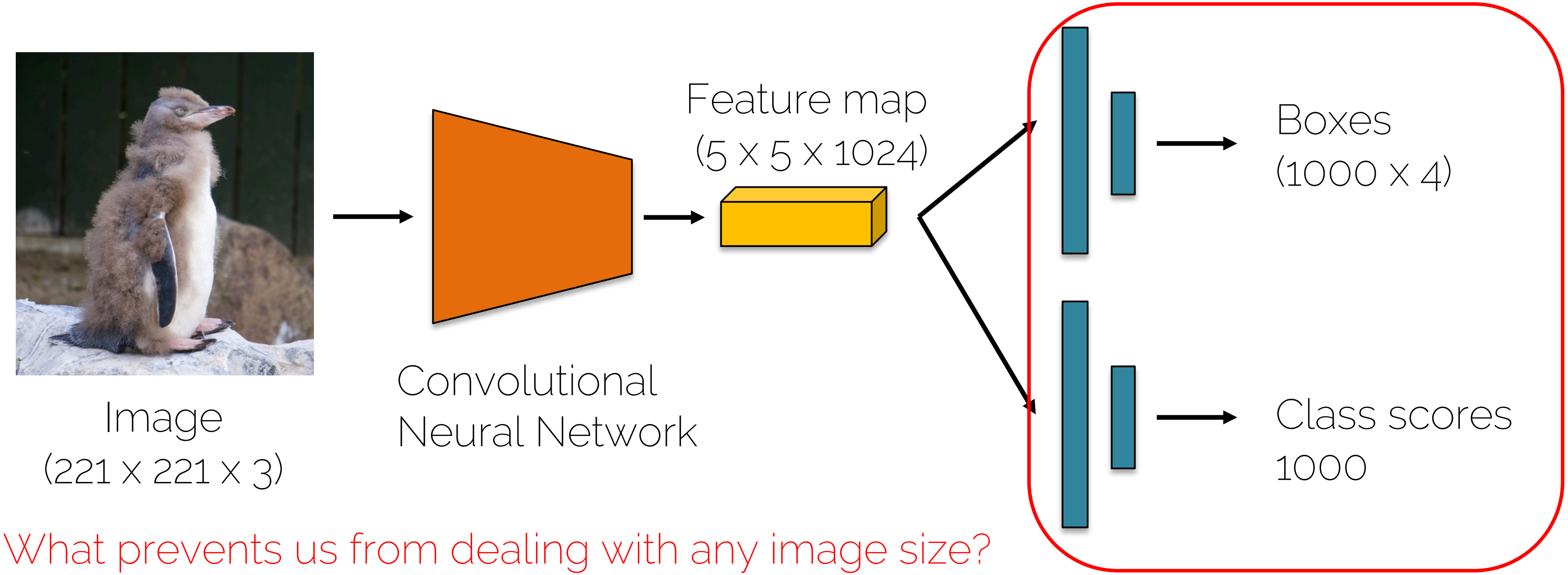


Final Predictions

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

Overfeat

- Sliding window + box regression + classification

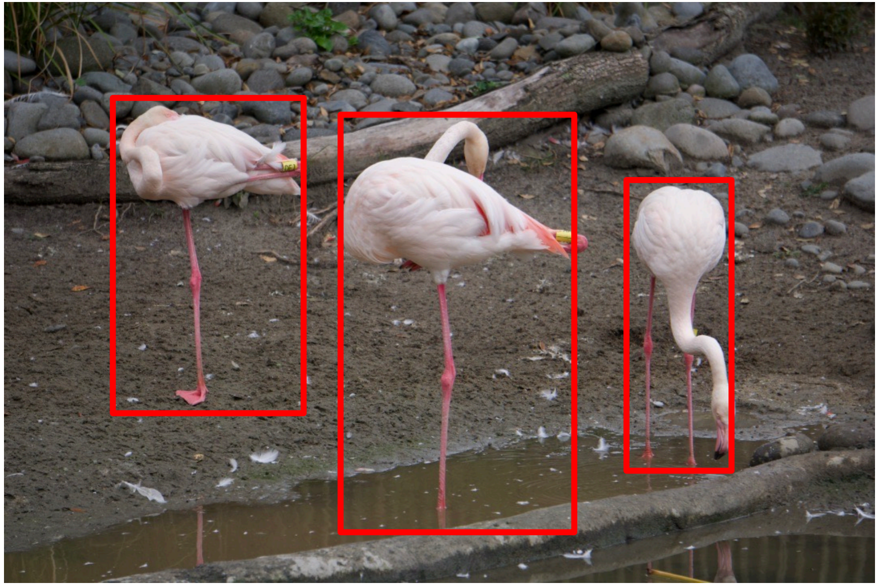


What prevents us from dealing with any image size?

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

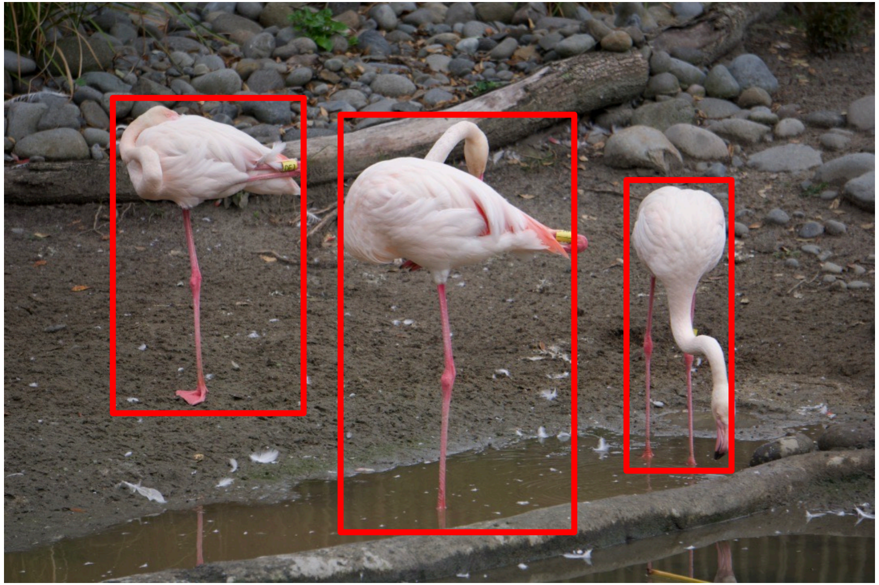
What about multiple objects?

- Localization: ✅ Regression
- How about detection?



What about multiple objects?

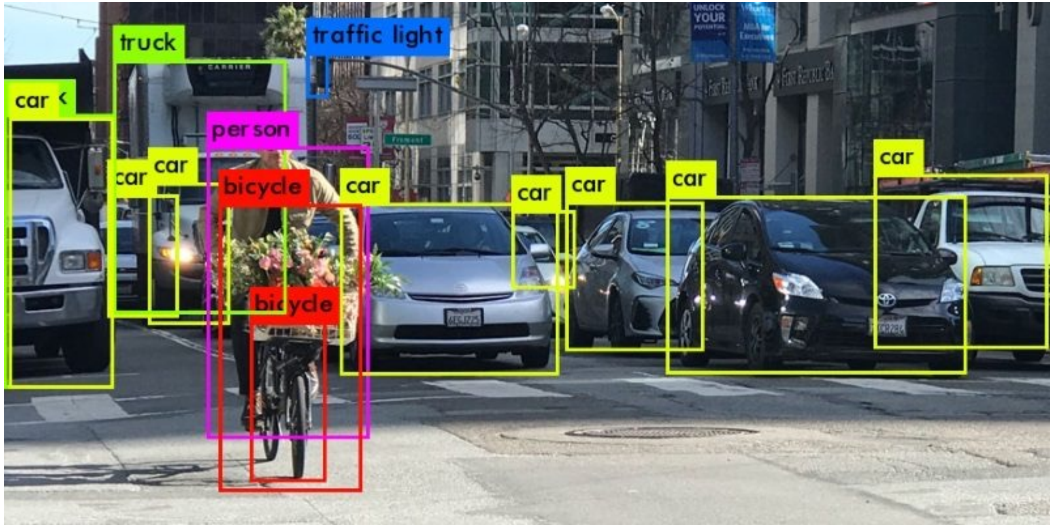
- Localization: ✅ Regression
- How about detection?



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

What about multiple objects?

- Localization: ✅ Regression
- How about detection?



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

What about multiple objects?

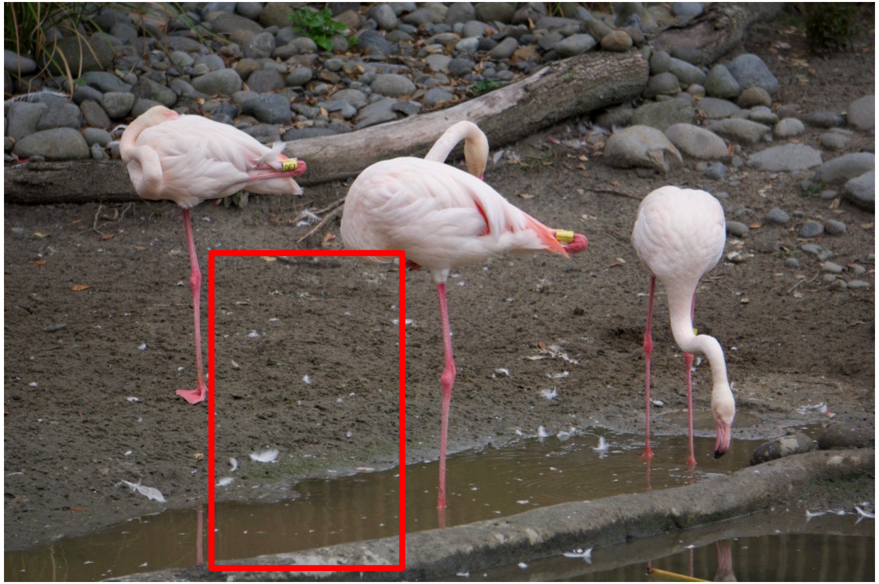
- Localization: ✅ Regression
- How about detection?

- Having a variable sized output is not optimal for Neural Networks

- There are a couple of workarounds:
  - RNN: Romera-Paredes and Torr. Recurrent Instance Segmentation. ECCV 2016.
  - Set prediction: Rezatofighi, Kaskman, Motlagh, Shi, Cremers, Leal-Taixé, Reid. Deep Perm-Set Net: Learn to predict sets with unknown permutation and cardinality using deep neural networks. Arxiv: 1805.00613

Detection as classification?

- Localization: ✅ Regression
- How about detection? ❌ Regression

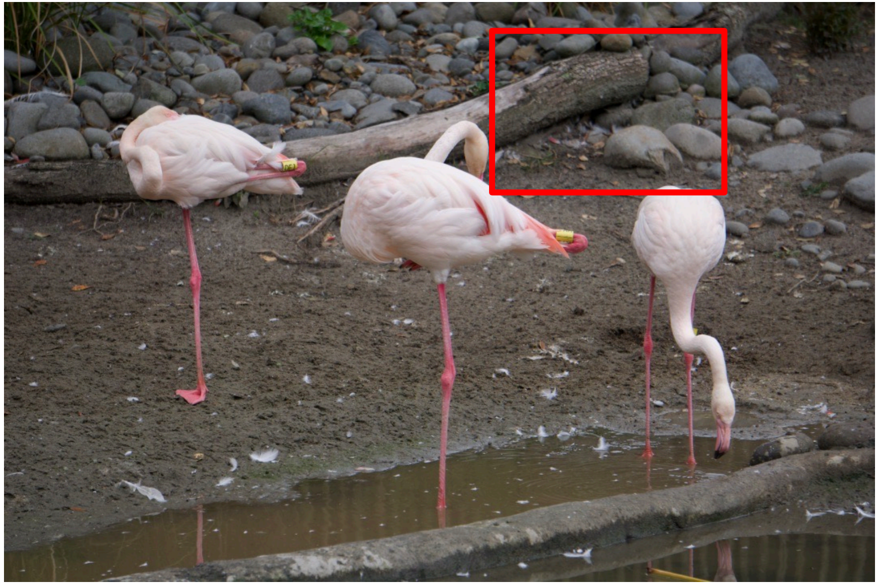


Is this a Flamingo?

NO

Detection as classification?

- Localization: ✅ Regression
- How about detection? ❌ Regression

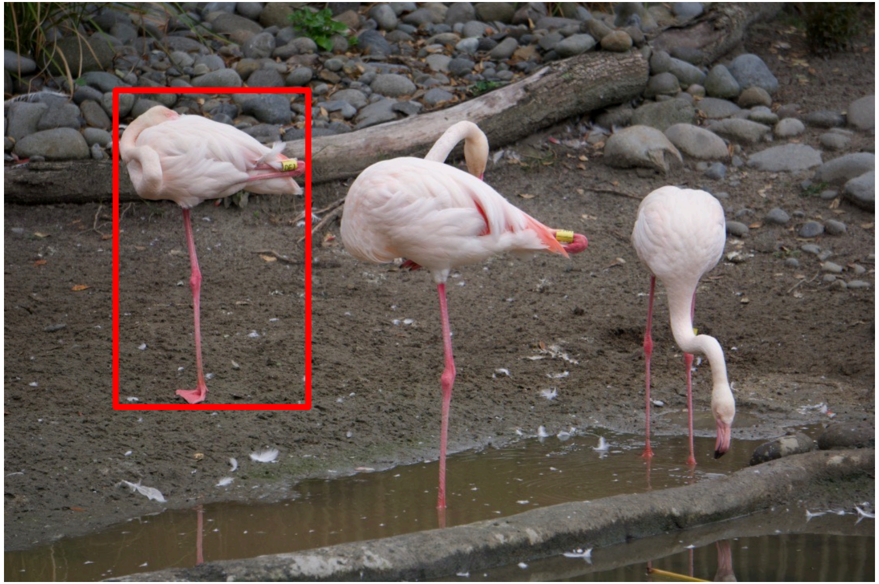


Is this a Flamingo?

NO

Detection as classification?

- Localization: ✅ Regression
- How about detection? ❌ Regression



Is this a Flamingo?

YES!

Detection as classification?

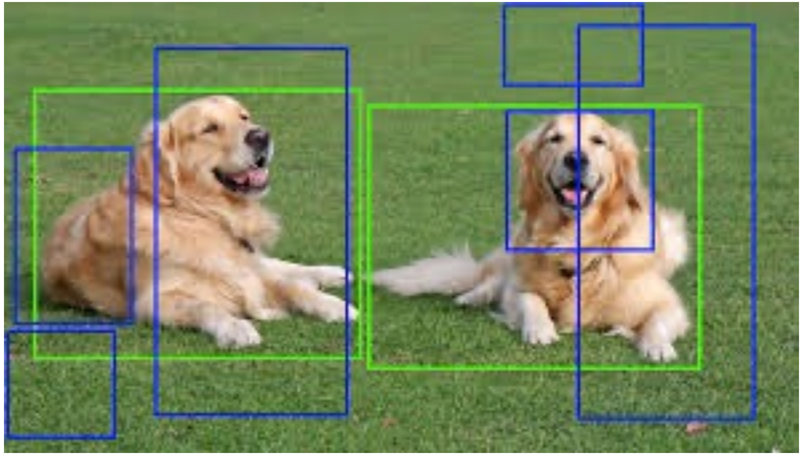
- Localization: ✅ Regression
- How about detection? ✅ Classification

- Problem:
  - Expensive to try all possible positions, scales and aspect ratios
  - How about trying only on a subset of boxes with most potential?

Region Proposals

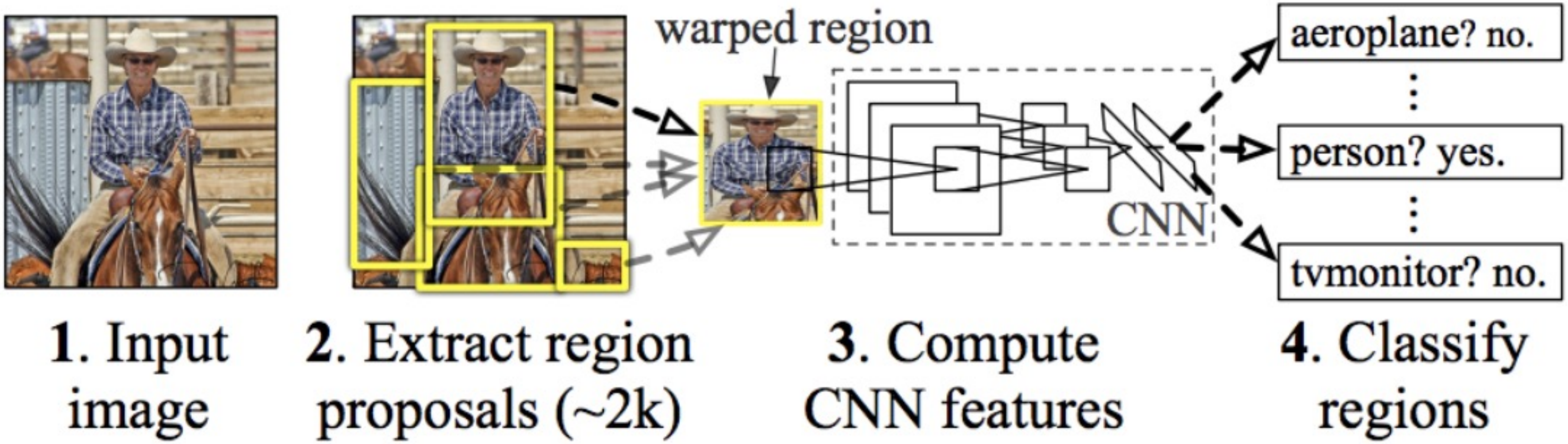
- We have already seen a method that gives us "interesting" regions in an image that potentially contain an object

- Step 1: Obtain region proposals
- Step 2: Classify them.



The R-CNN family

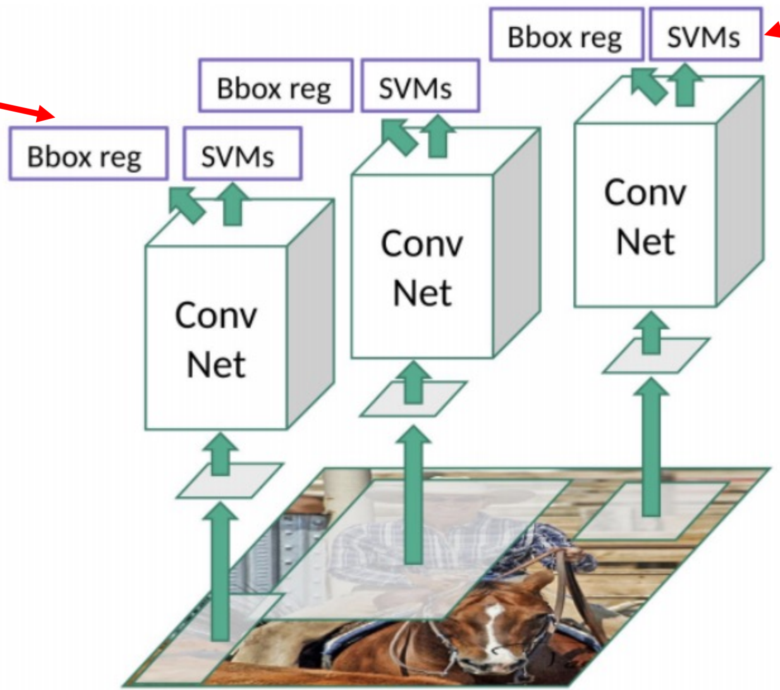
R-CNN



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

R-CNN

Regression head to refine the bounding box location



Classification head

Extract features

Warping to a fix size 227 x 227

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

R-CNN

- Training scheme:
  - 1. Pre-train the CNN on ImageNet
  - 2. Finetune the CNN on the number of classes the detector is aiming to classify (softmax loss)
  - 3. Train a linear Support Vector Machine classifier to classify image regions. One SVM per class! (hinge loss)
  - 4. Train the bounding box regressor (L2 loss)

R-CNN

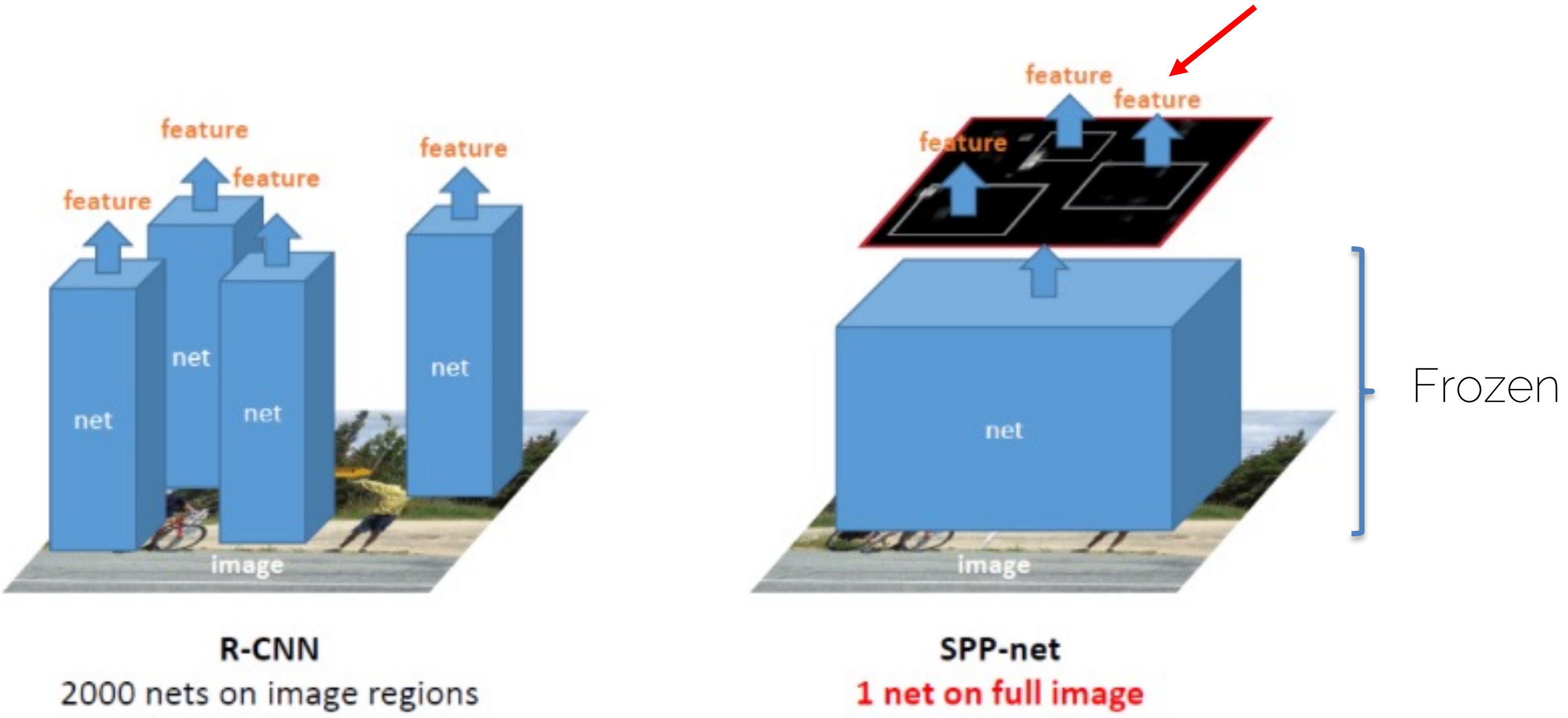
- PROS:
  - The pipeline of proposals, feature extraction and SVM classification is well-known and tested. Only features are changed (CNN instead of HOG).
  - CNN summarizes each proposal into a 4096 vector (much more compact representation compared to HOG)
  - Leverage transfer learning: the CNN can 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: Let us try to solve this first
  - 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 → not exploiting learning to its full potential.

SPP-Net

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.

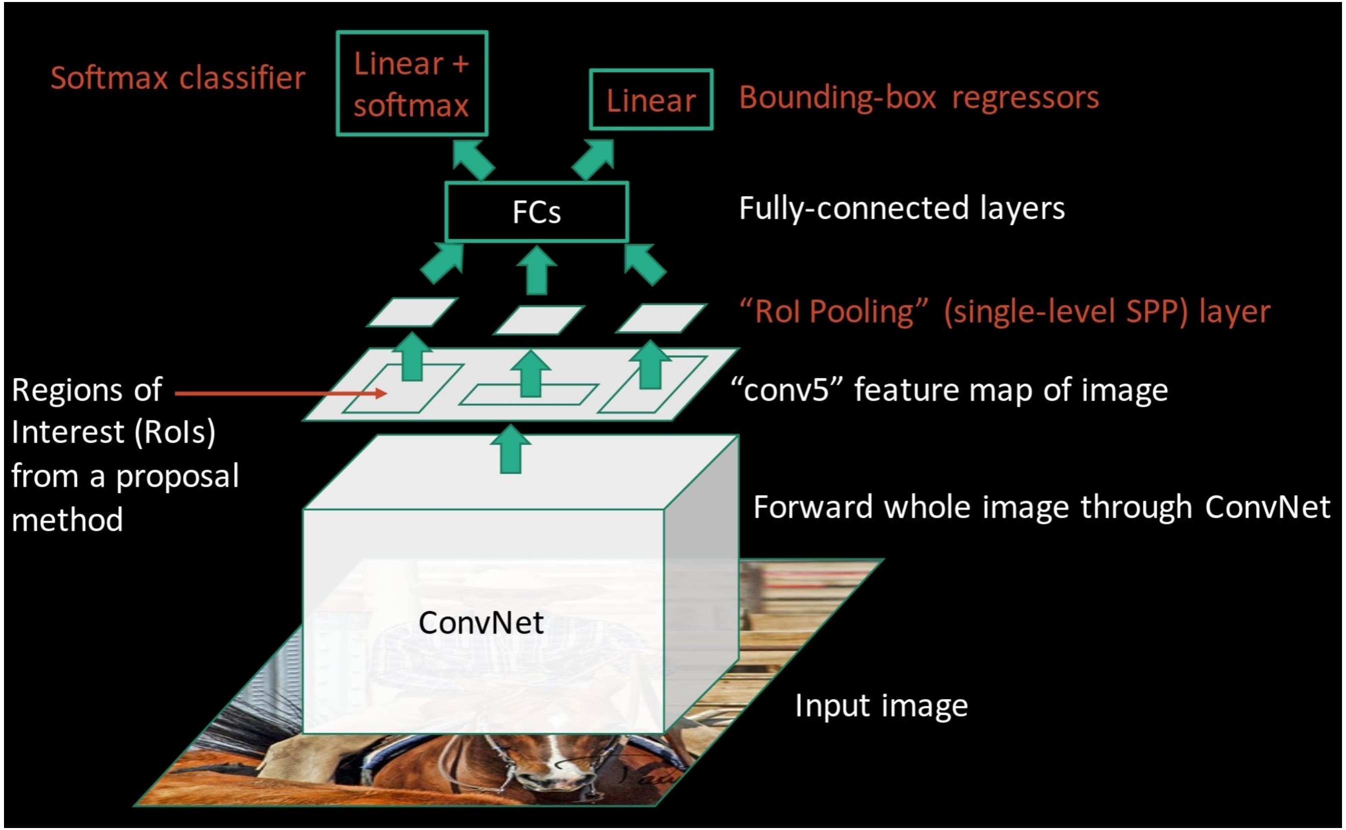
SPP-Net

- It solved the R-CNN problem of being slow at test time
- It still has some problems inherited from R-CNN:
  - Training is still slow (a bit faster than R-CNN)
  - Training scheme is still complex
  - Still no end-to-end training

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

Fast R-CNN

Fast R-CNN

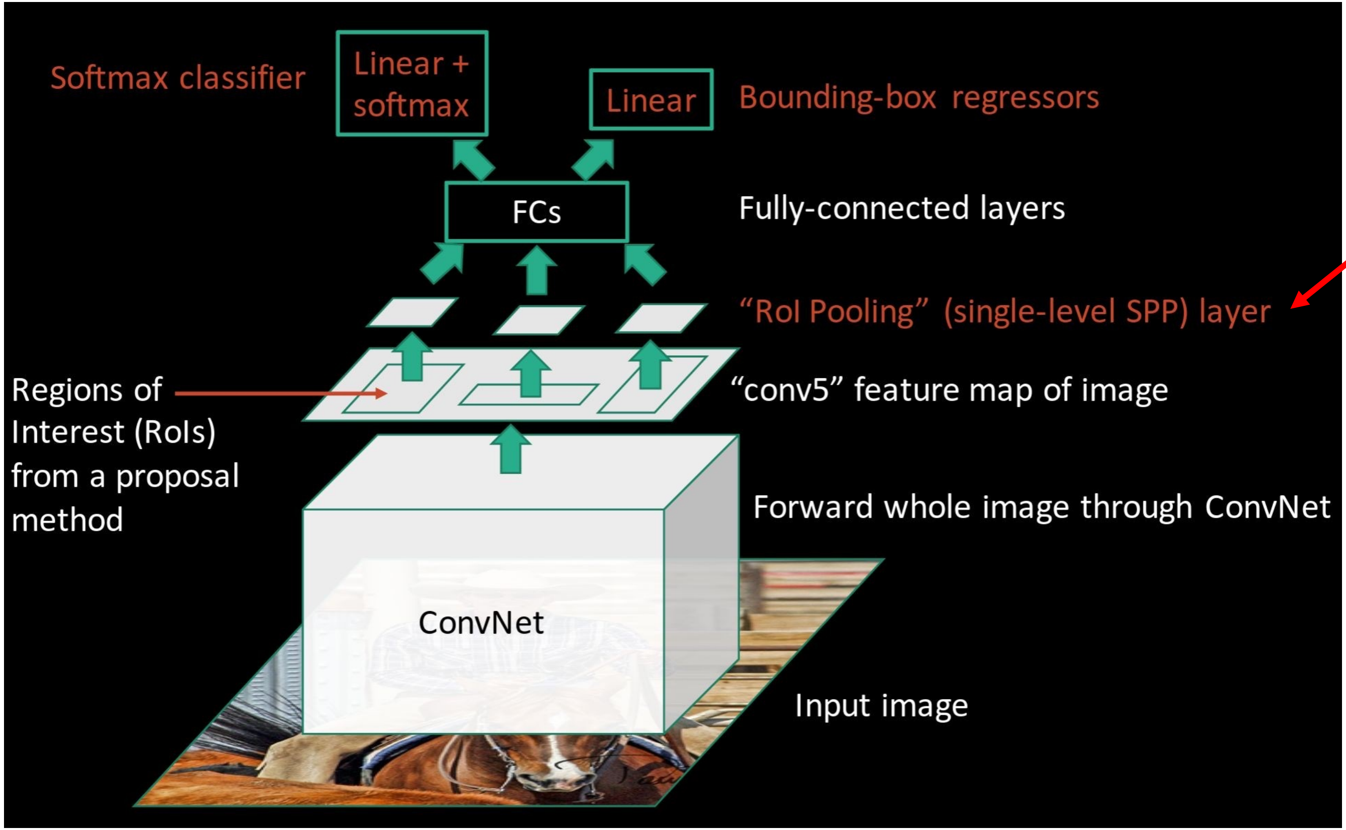


Shared computation at test time (like SPP)

Girschick, "Fast R-CNN", ICCV 2015

Slide credit: Ross Girschick

Fast R-CNN



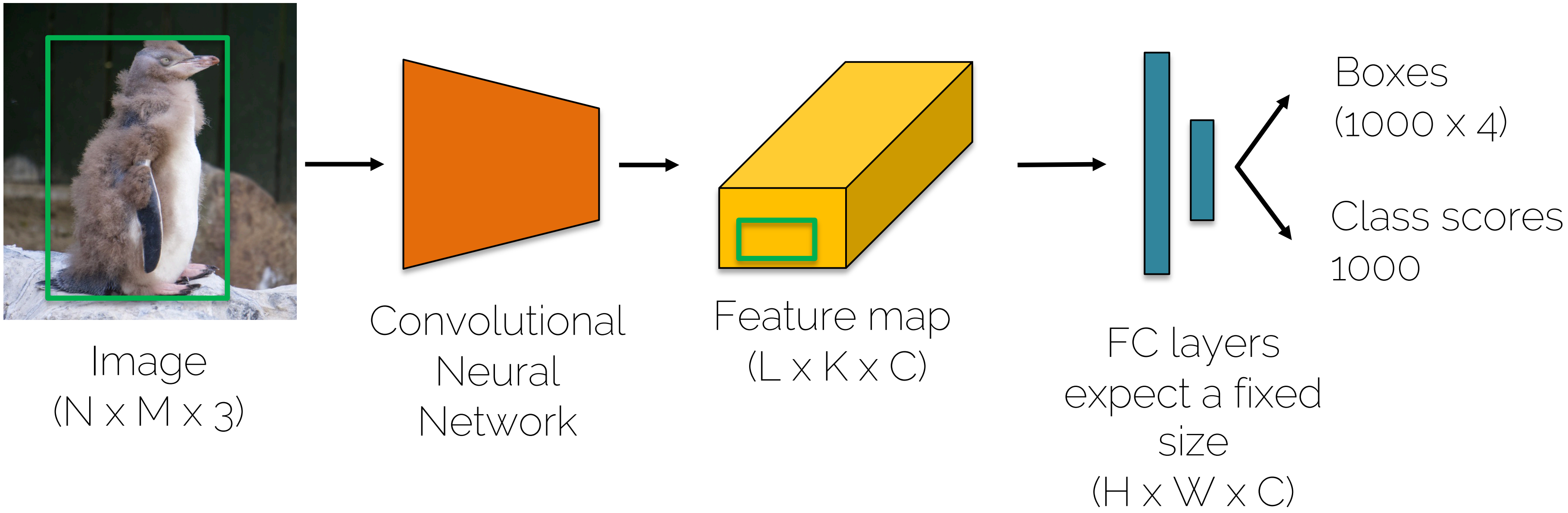
Region of Interest Pooling

Girschick, "Fast R-CNN", ICCV 2015

Slide credit: Ross Girschick

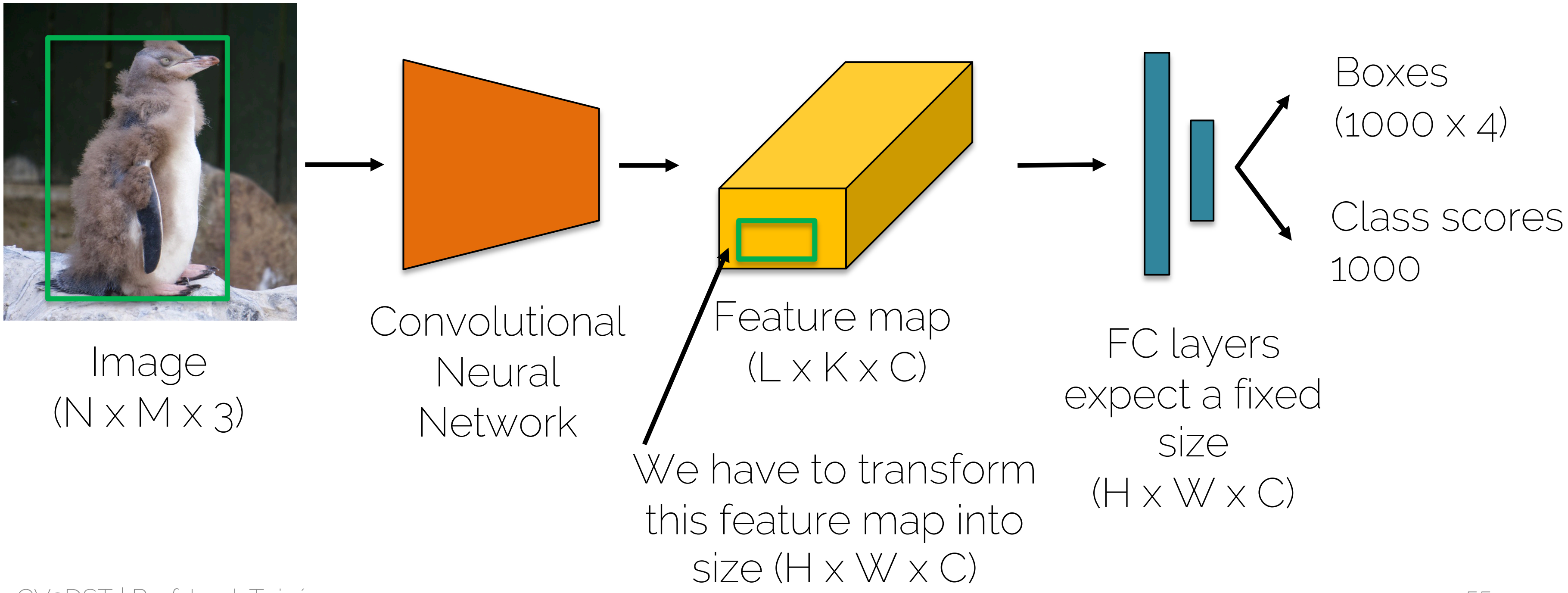
Fast R-CNN: RoI Pooling

- Region of Interest Pooling



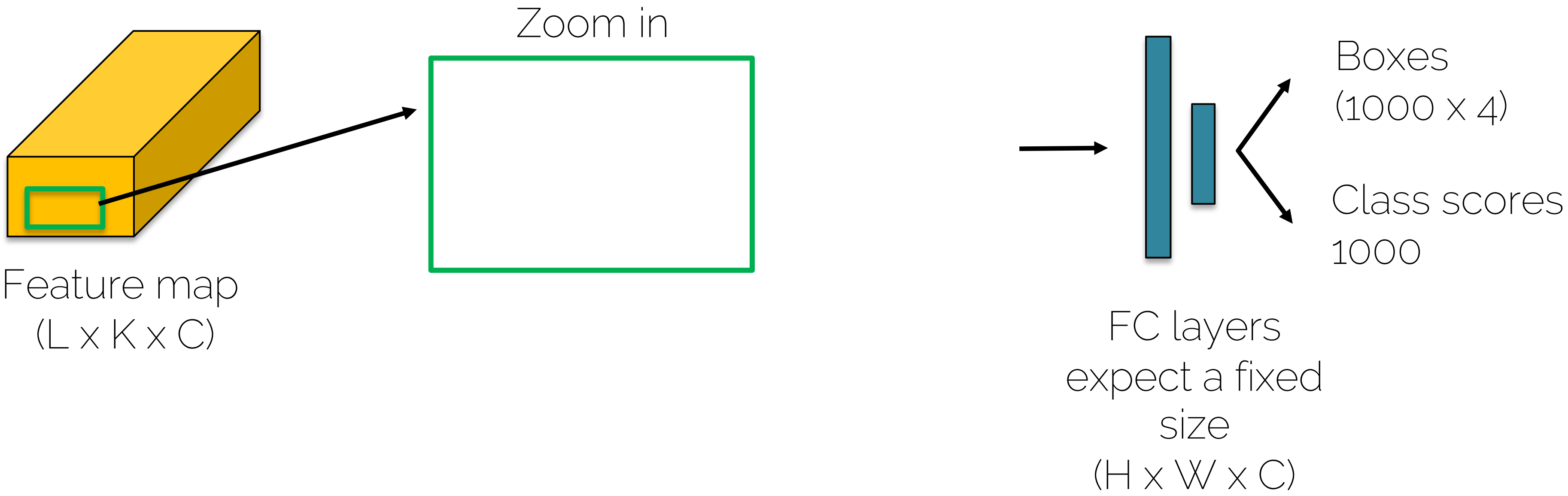
Fast R-CNN: RoI Pooling

- Region of Interest Pooling



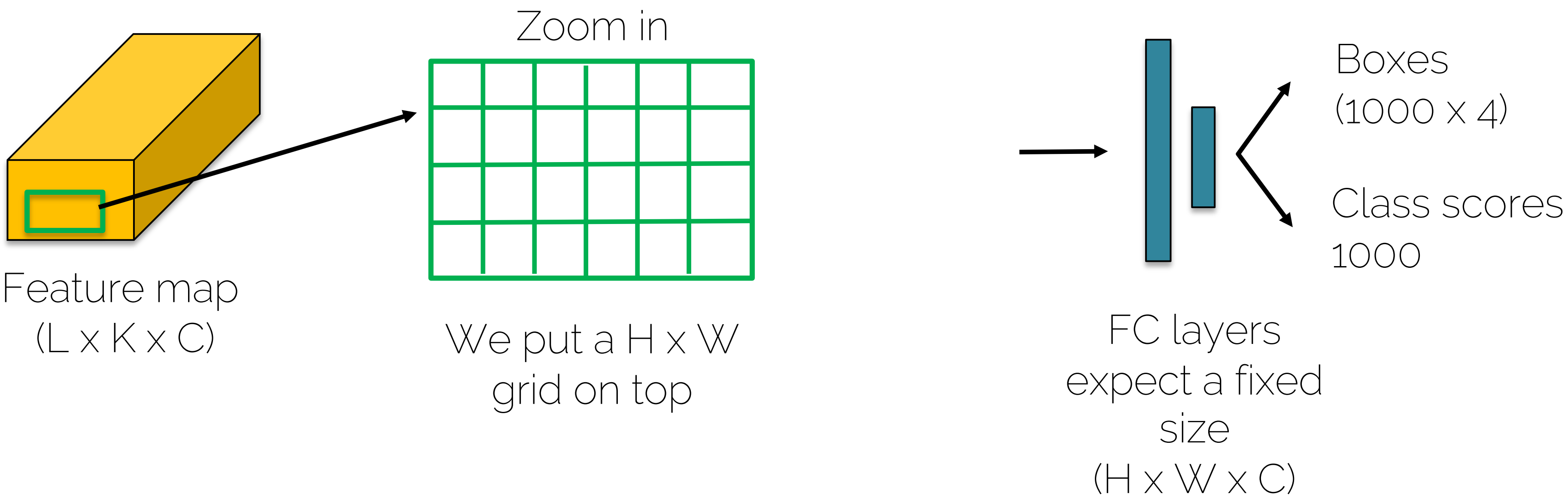
Fast R-CNN: RoI Pooling

- Region of Interest Pooling



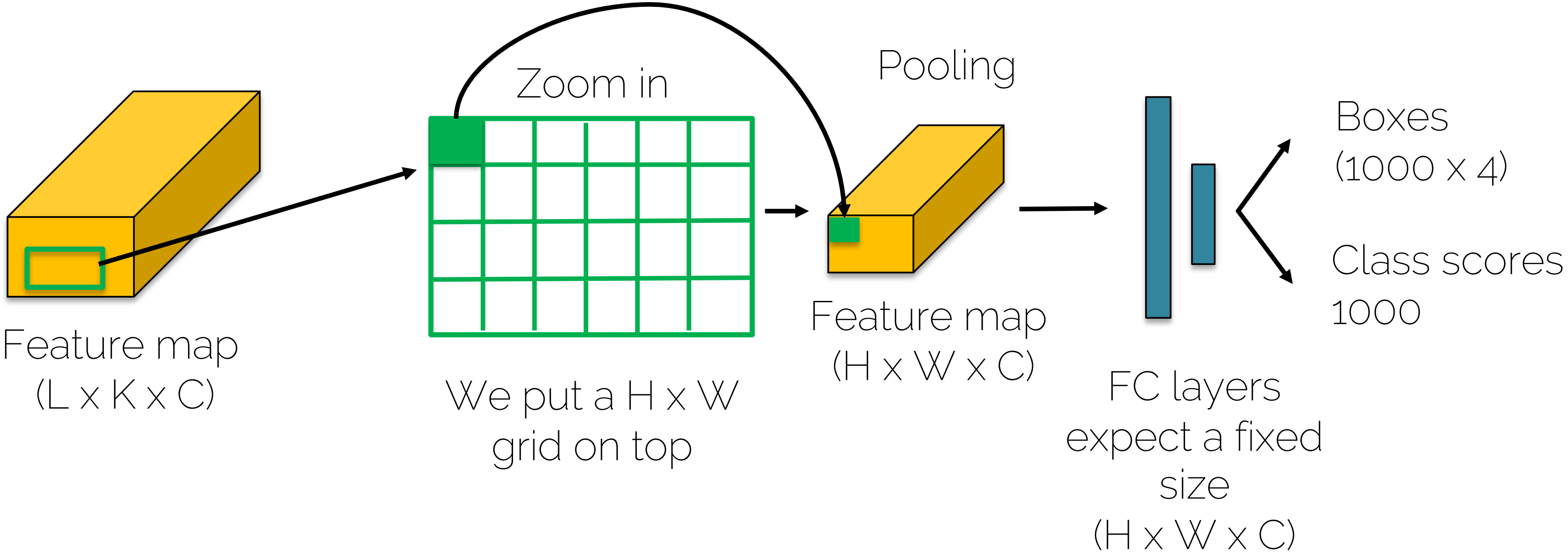
Fast R-CNN: RoI Pooling

- Region of Interest Pooling



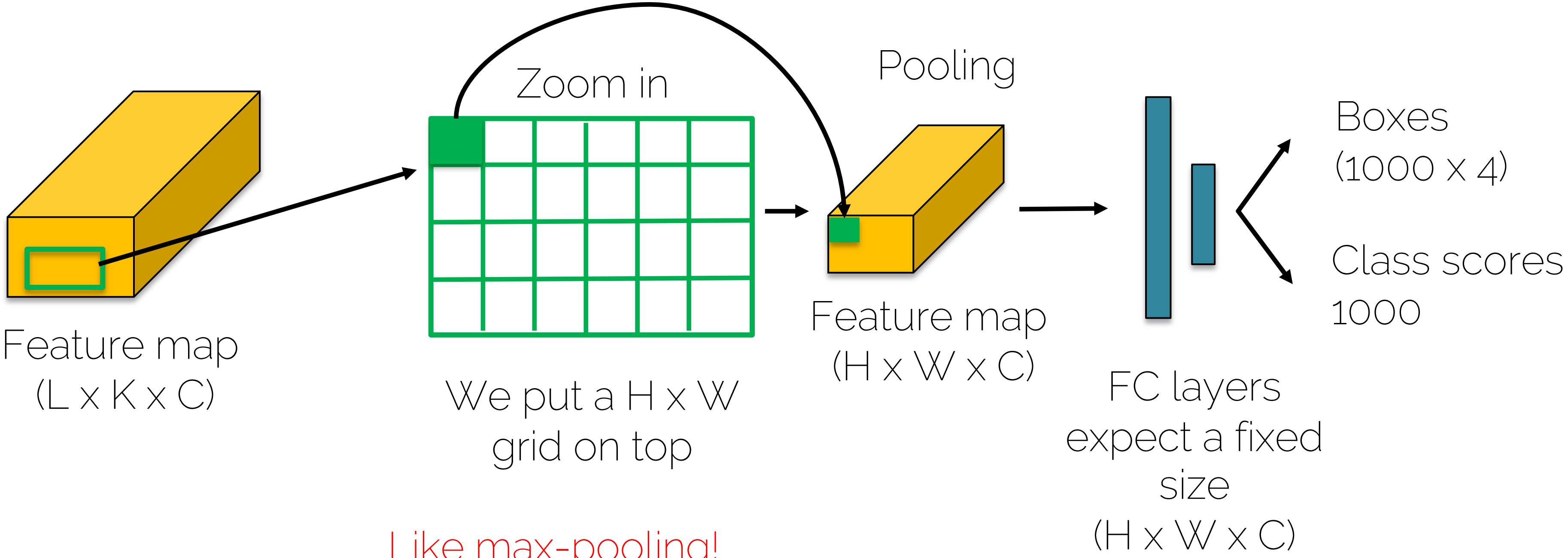
Fast R-CNN: RoI Pooling

- Region of Interest Pooling



Fast R-CNN: RoI Pooling

- RoI Pooling: how do you do backpropagation?



Like max-pooling!

Fast R-CNN Results

- VGG-16 CNN on Pascal VOC 2007 dataset

	R-CNN	Fast R-CNN
Training Time:	84 hours	9.5 hours
(Speedup)	1x	8.8x

Faster!

Fast R-CNN Results

- VGG-16 CNN on Pascal VOC 2007 dataset

		R-CNN	Fast R-CNN
Faster!	Training Time:	84 hours	9.5 hours
	(Speedup)	1x	8.8x
FASTER!	Test time per image	47 seconds	0.32 seconds
	(Speedup)	1x	146x

Fast R-CNN Results

- VGG-16 CNN on Pascal VOC 2007 dataset

		R-CNN	Fast R-CNN
Faster!	Training Time:	84 hours	9.5 hours
	(Speedup)	1x	8.8x
FASTER!	Test time per image	47 seconds	0.32 seconds
	(Speedup)	1x	146x
Better!	mAP (VOC 2007)	66.0	66.9

Fast R-CNN Results

The test times do not include proposal generation!

- VGG-16 CNN on Pascal VOC 2007 dataset

		R-CNN	Fast R-CNN
Faster!	Training Time:	84 hours	9.5 hours
	(Speedup)	1x	8.8x
FASTER!	Test time per image	47 seconds	0.32 seconds
	(Speedup)	1x	146x
Better!	mAP (VOC 2007)	66.0	66.9

Fast R-CNN Results

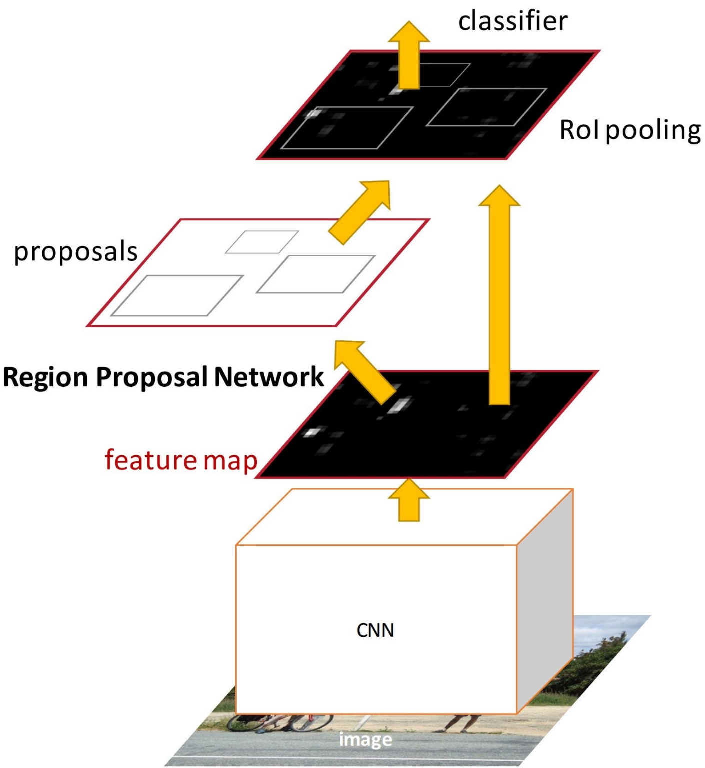
With proposals included

- VGG-16 CNN on Pascal VOC 2007 dataset

		R-CNN	Fast R-CNN
Faster!	Training Time:	84 hours	9.5 hours
	(Speedup)	1x	8.8x
FASTER!	Test time per image	50 seconds	2 seconds
	(Speedup)	1x	25x
Better!	mAP (VOC 2007)	66.0	66.9

Faster R-CNN

Faster R-CNN:



- Solution: Have the proposal generation integrated with the rest of the pipeline

- Region Proposal Network (RPN) trained to produce region proposals directly.

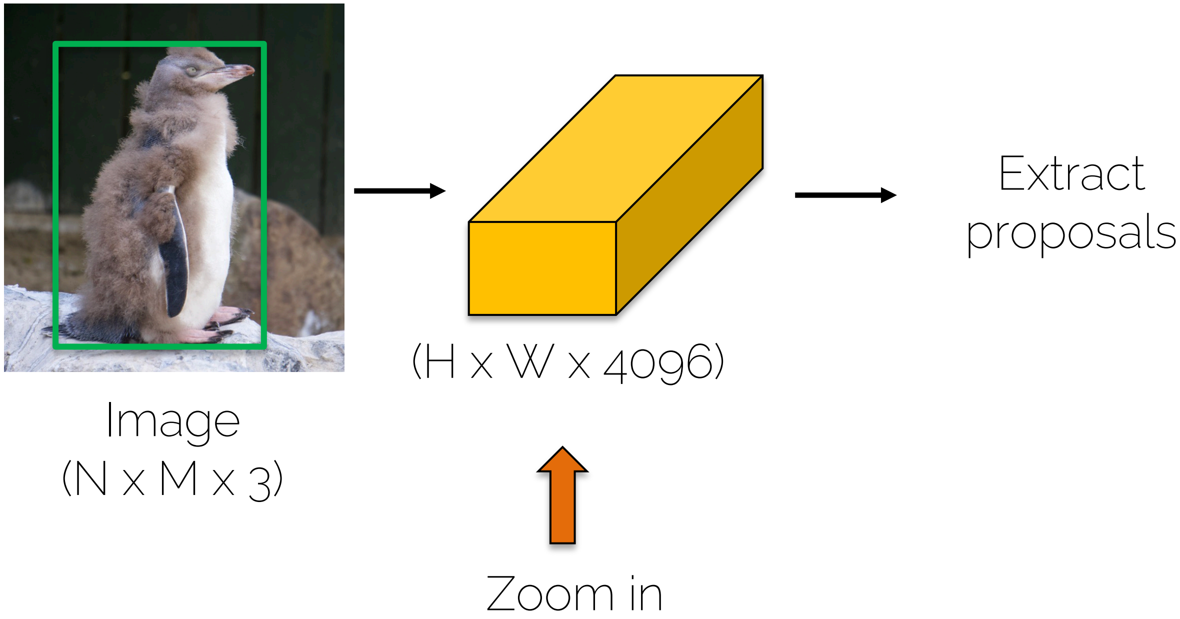
- After RPN, everything is like Fast R-CNN

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

Slide credit: Ross Girschick

Region proposal network

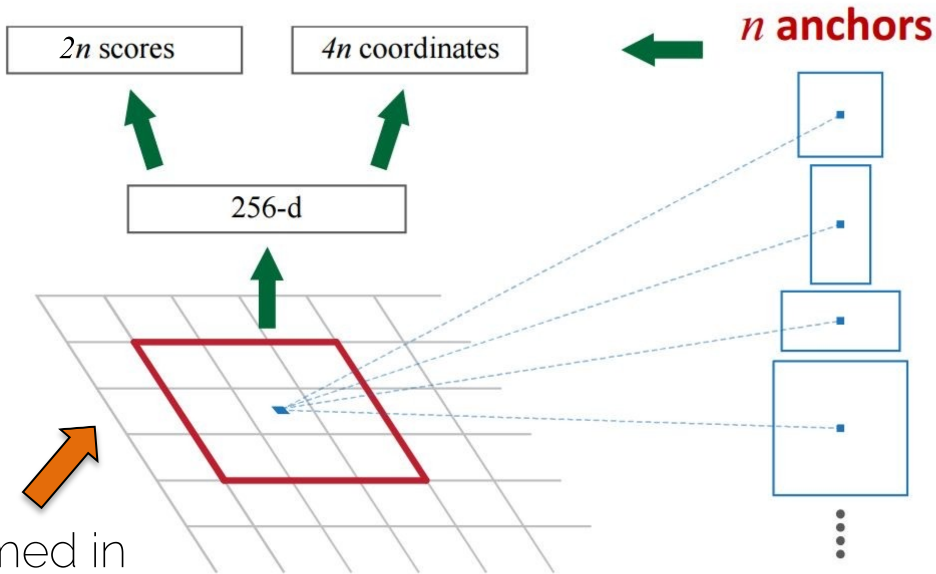
- How to extract proposals



- How many proposals?
  ✓ We need to decide a fixed number
- Where are they placed?
  ✓ Densely

Region proposal network

- We fix the number of proposals by using a set of n=9 anchors *per location*.
- 9 anchors = 3 scales and 3 aspect ratios

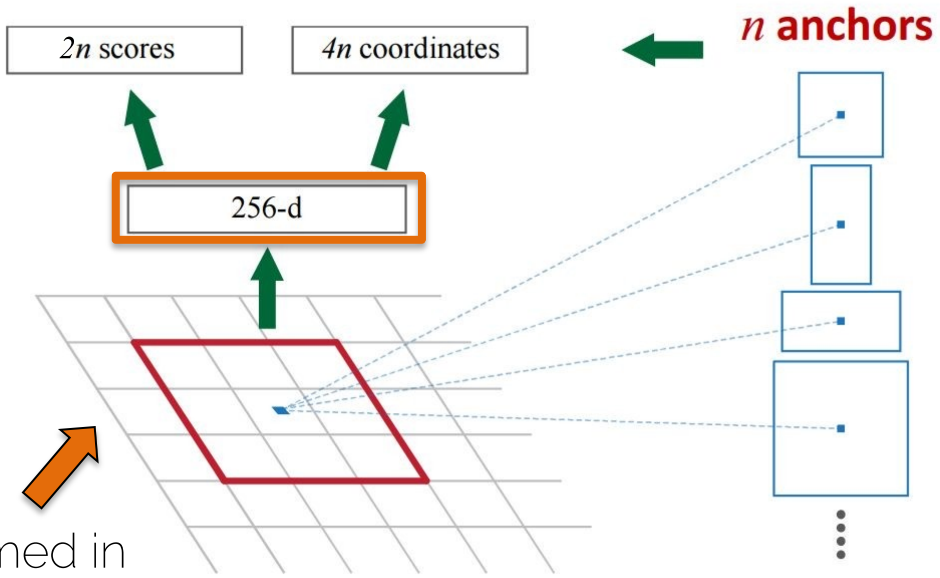


Zoomed in feature map

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

Region proposal network

- We fix the number of proposals by using a set of n=9 anchors *per location*.
- 9 anchors = 3 scales and 3 aspect ratios
- We extract a descriptor per *location*



Zoomed in feature map

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

Region proposal network

- How to extract proposals

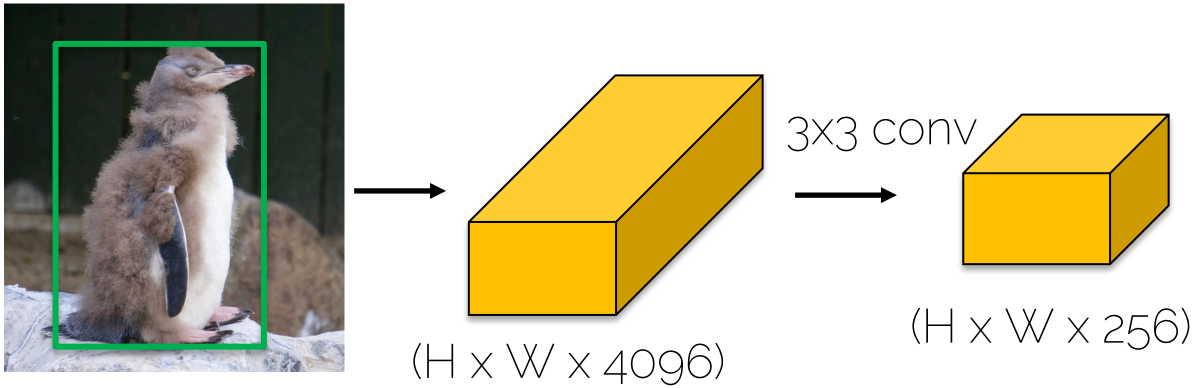
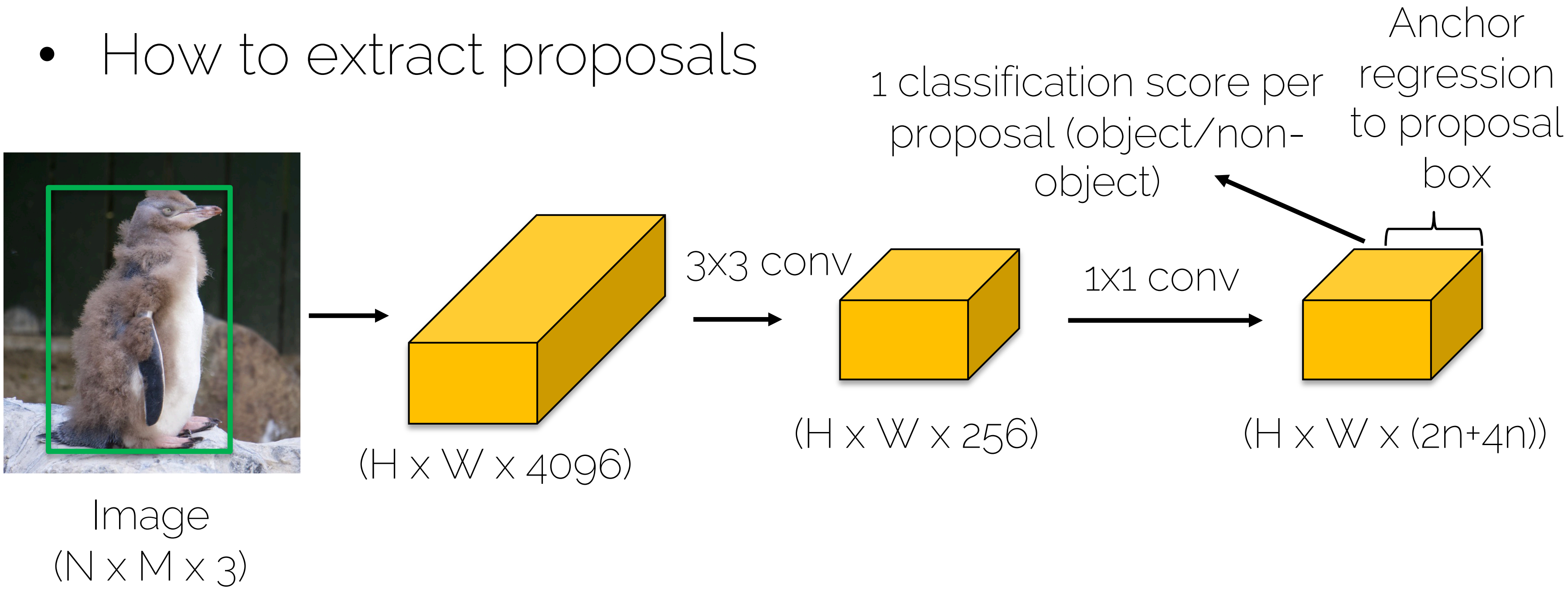


Image
(N x M x 3)

#anchors per image? (H x W x n)

Region proposal network

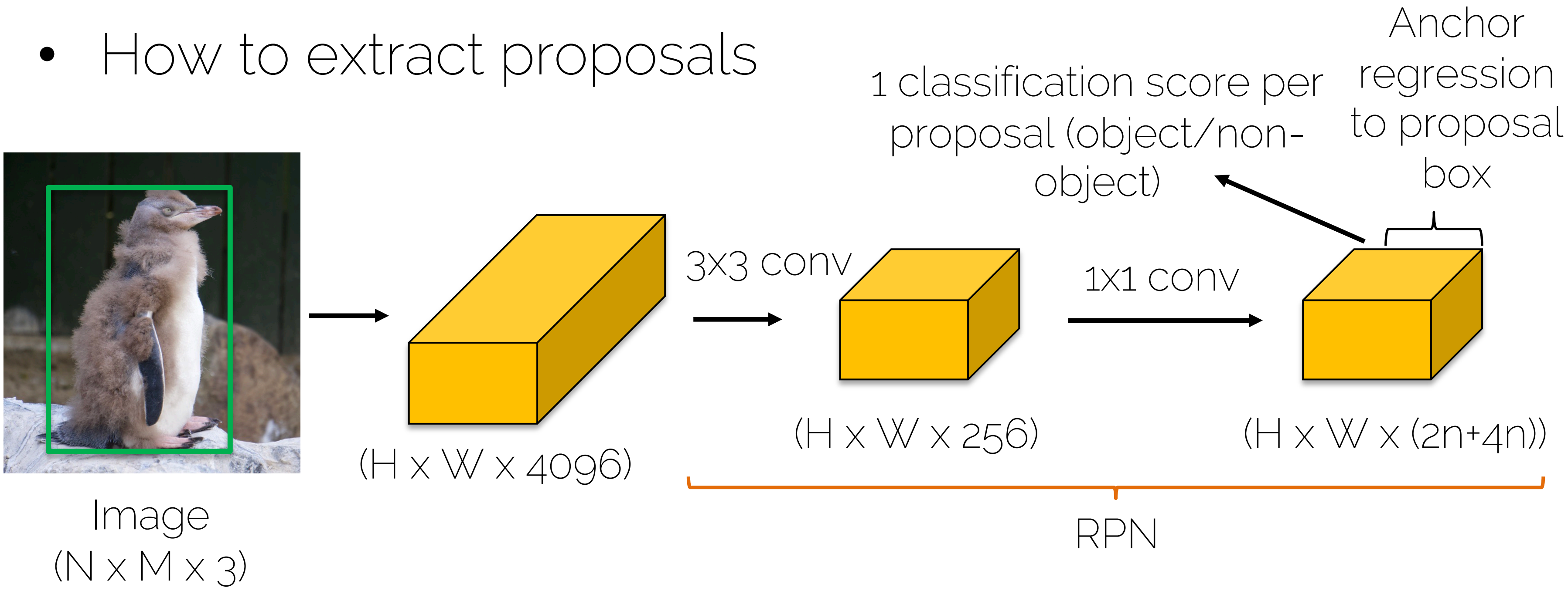
- How to extract proposals



#anchors per image? (H x W x n)

Region proposal network

- How to extract proposals



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

RPN: training and losses

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

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

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

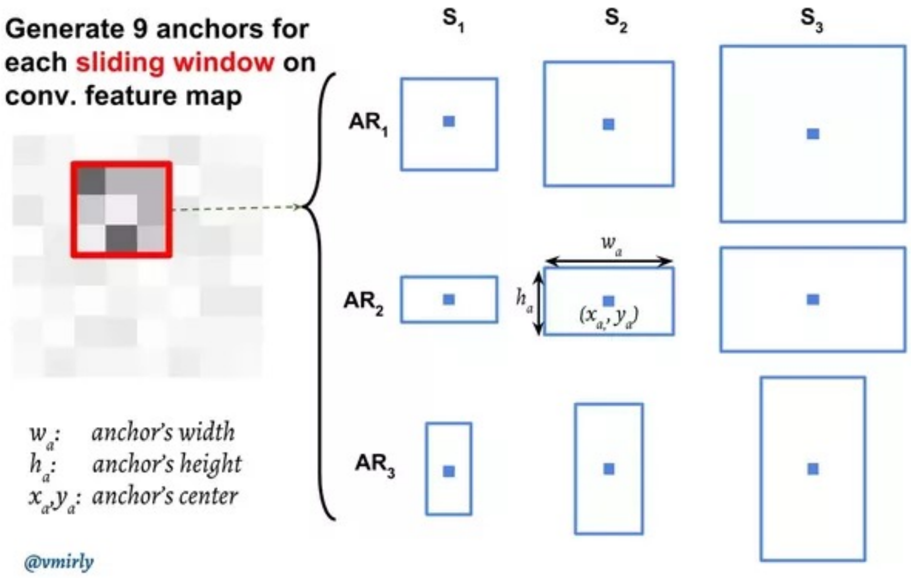
- 1 indicates the anchor represent an object (foreground) and 0 indicates background object. The rest do not contribute to the training.

RPN: training and losses

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

RPN: training and losses

- Each anchor is described by the center position, width and height x_a, y_a, w_a, h_a



RPN: training and losses

- Each anchor is described by the center position, width and height x_a, y_a, w_a, h_a
- What the network actually predicts are t_x, t_y, t_w, t_h

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

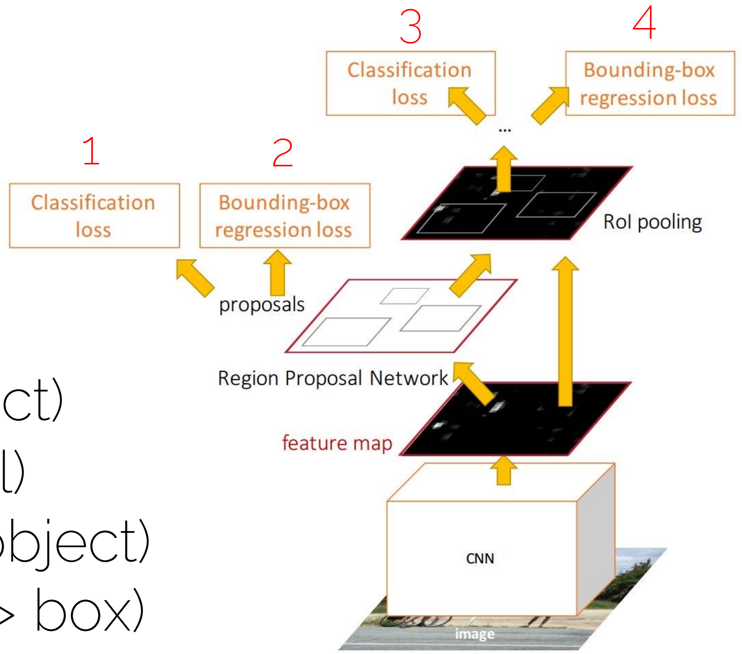
Normalized width $t_w = log(w/w_a),$ $t_h = log(h/h_a),$ Normalized height

- 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)



Slide credit: Ross Girschick

Faster R-CNN

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

Faster R-CNN: Results

	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image (with proposals)	50 seconds	2 seconds	0.2 seconds
(Speedup)	1x	25x	250x
mAP (VOC 2007)	66.0	66.9	66.9

Two-stage object detectors

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.