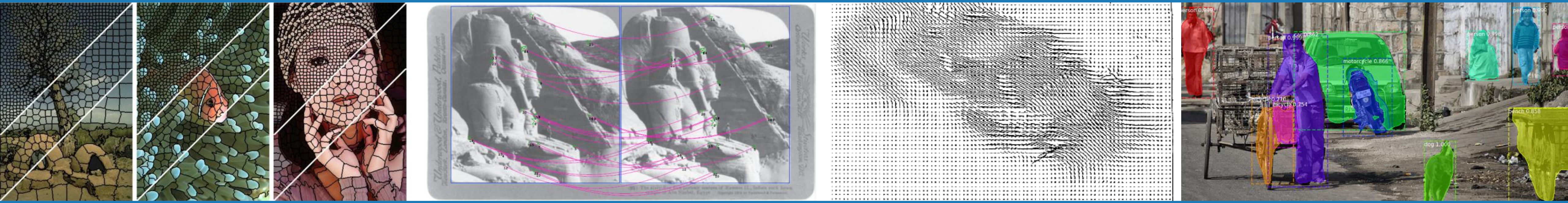
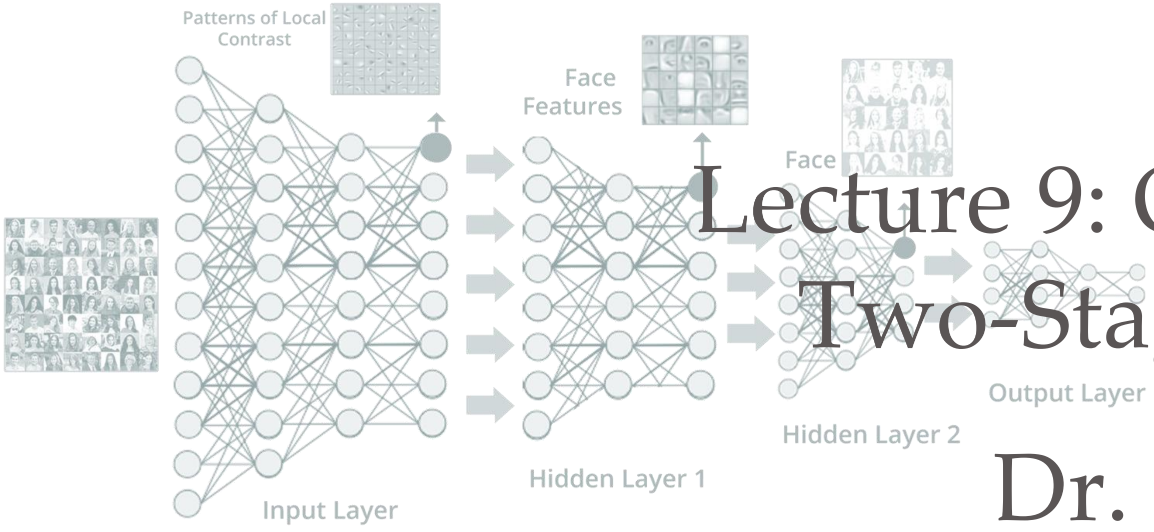
INSTITUTE FOR MACHINE
LEARNING AND ANALYTICS



Computer Vision



Lecture 9: Object Detection - Two-Stage Approaches

Dr. Xiao Zhao

Elektrotechnik, Medizintechnik und Informatik

23. Mai 2025

Content

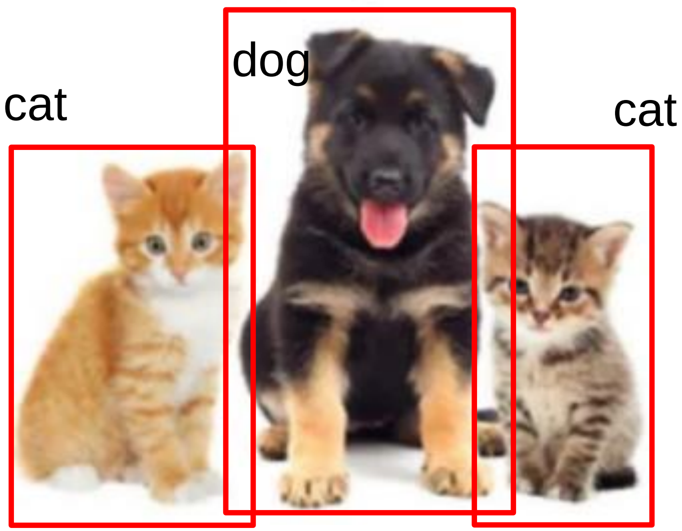
- Introduction
- Two-Stage Approaches (**lecture 9**)
  - Sliding window approach
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
- One-Stage Approaches (**lecture 10**)

Segmentation Tasks

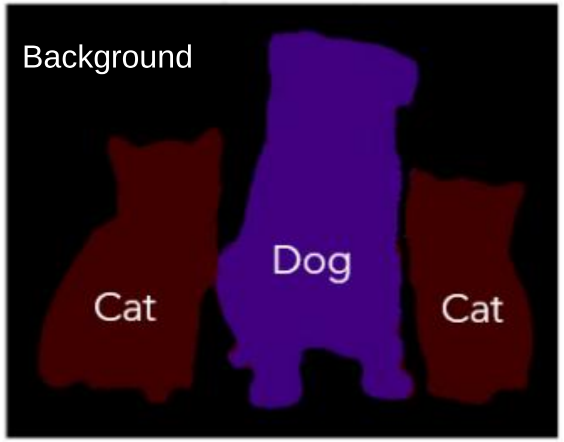
Input Image

network

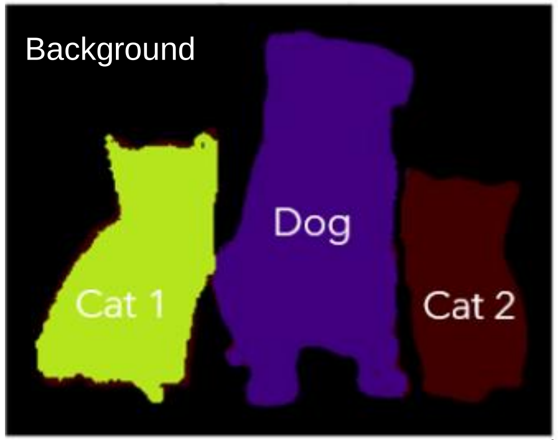
Object Detection



Semantic Segmentation



Instance Segmentation

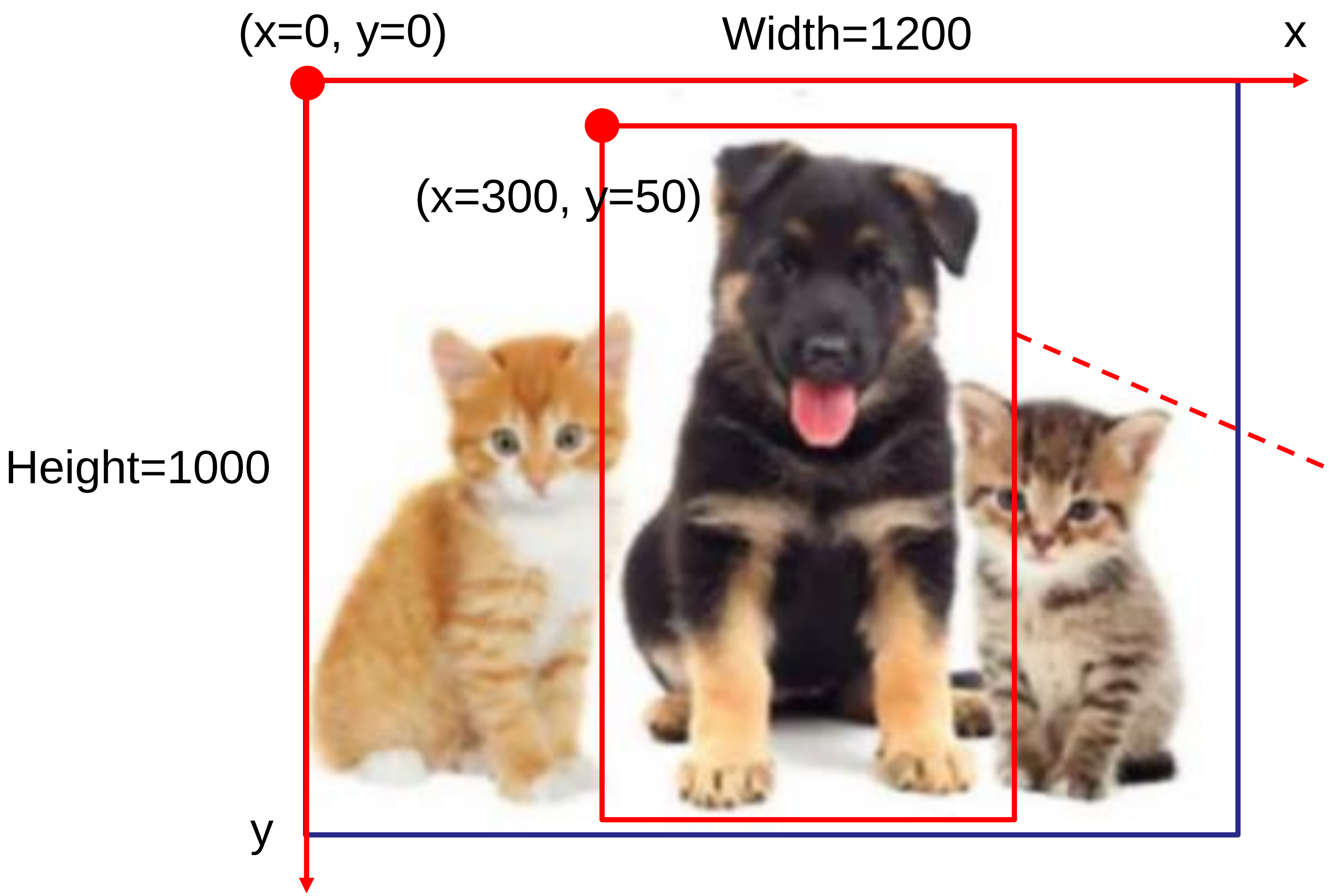


Source: https://www.v7labs.com/blog/instance-segmentation-guide

Object Detection

- Also called „Localization“
- Task A:
  - Predict the location of bounding boxes (4 real numbers)
- Task B:
  - Predict the classes of each detected bounding boxes

Image and Bounding Box coordinates

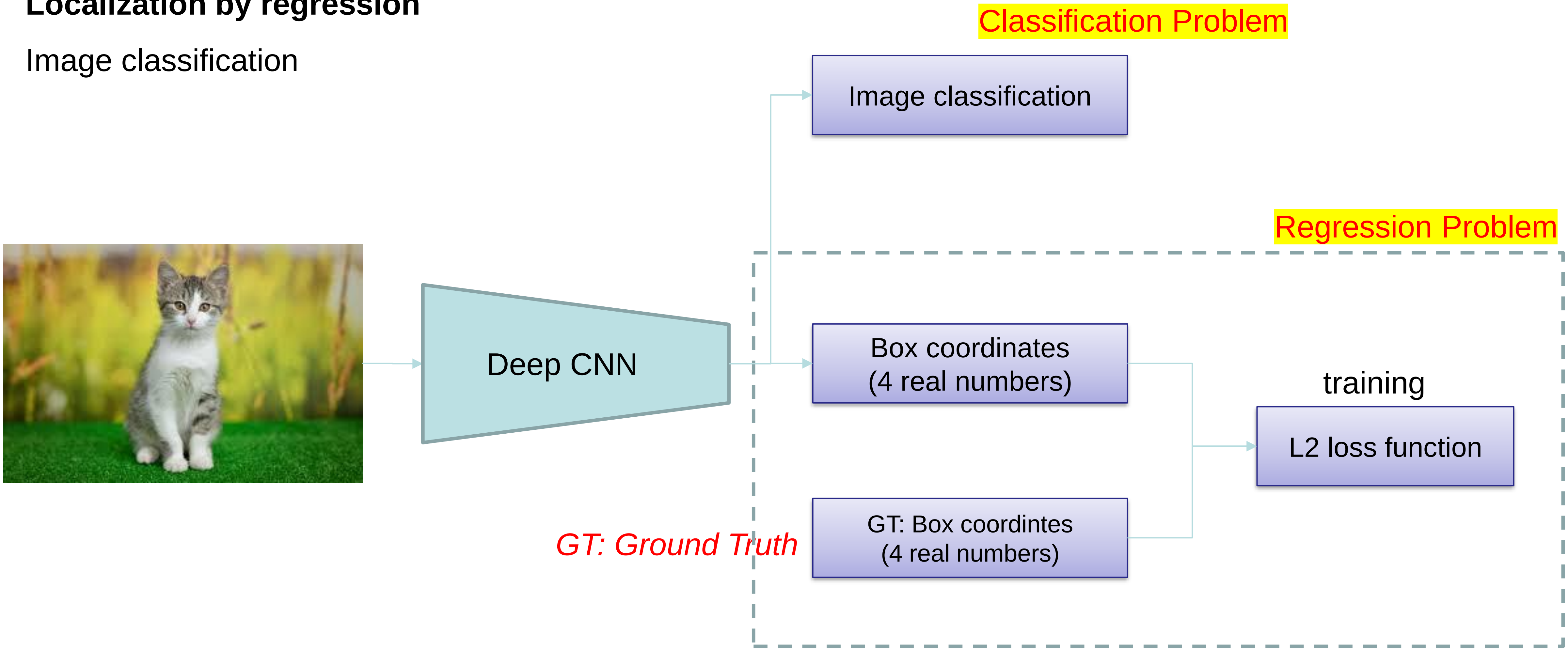


Bounding box:
(X=300, y=500, W=400, H=900)

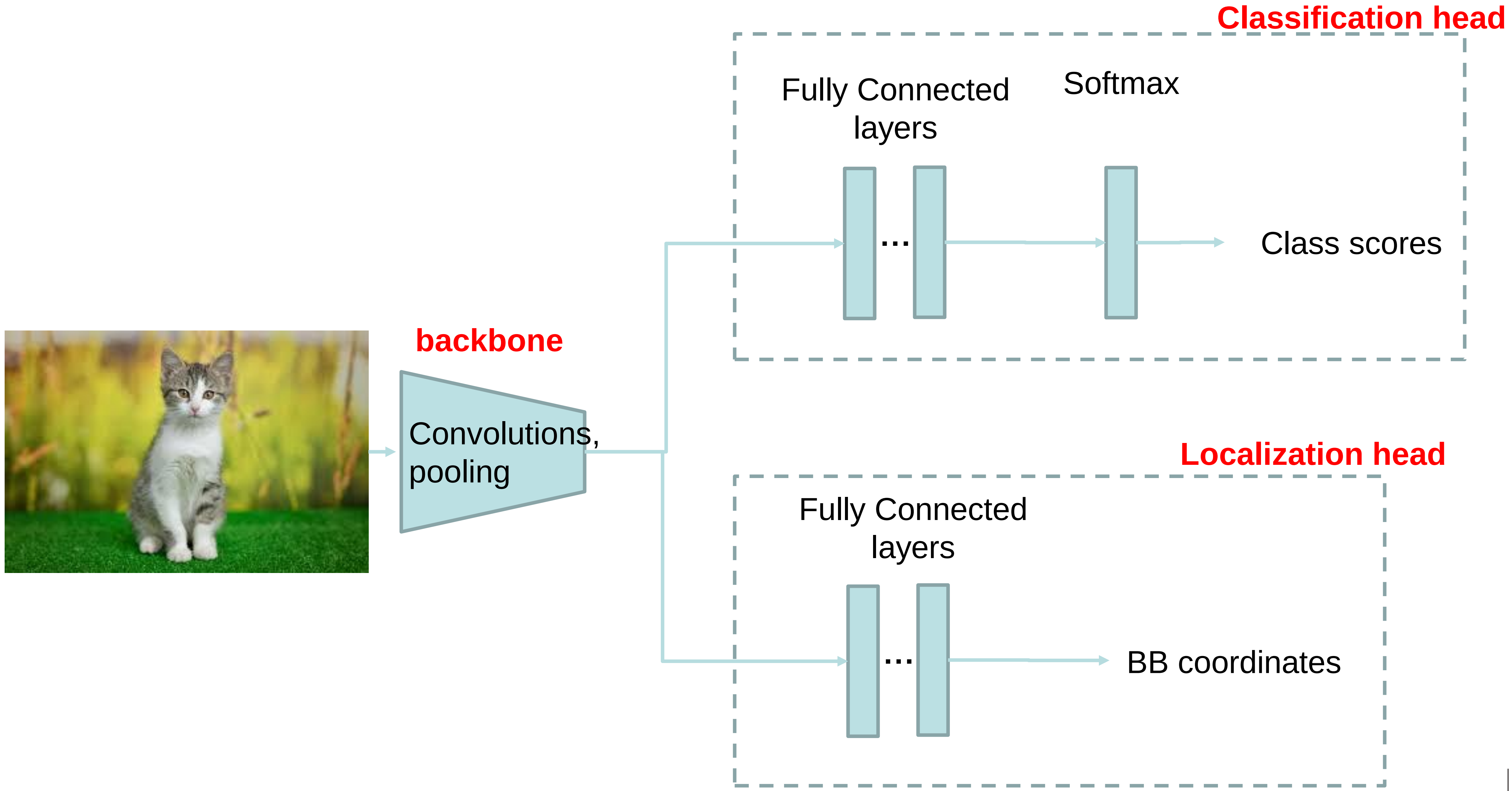
4 real numbers

A simple case: only one object

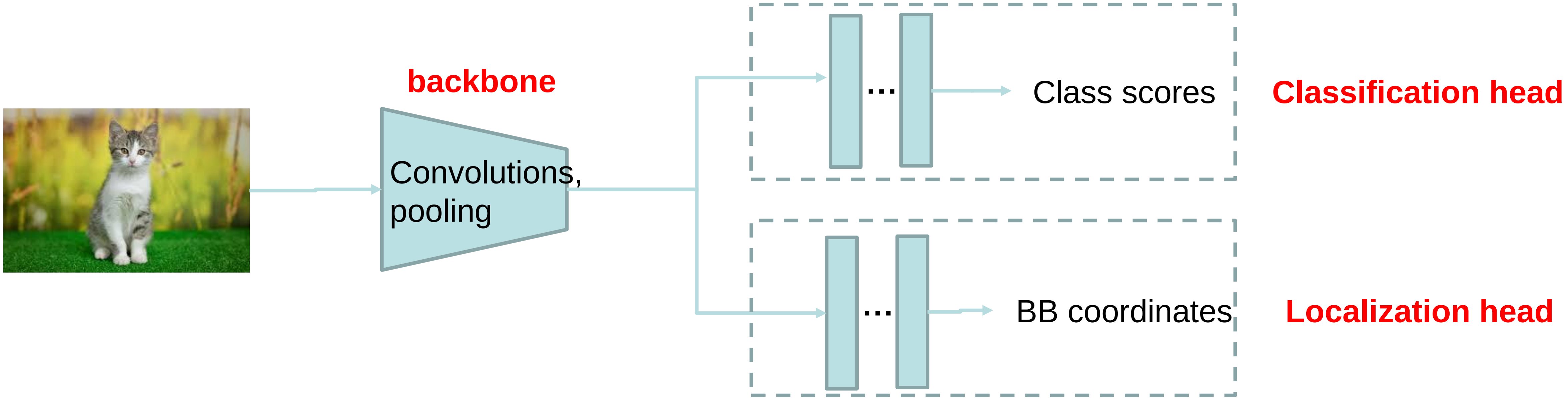
- **Localization by regression**
- Image classification



A simple case: only one object



A simple case: only one object



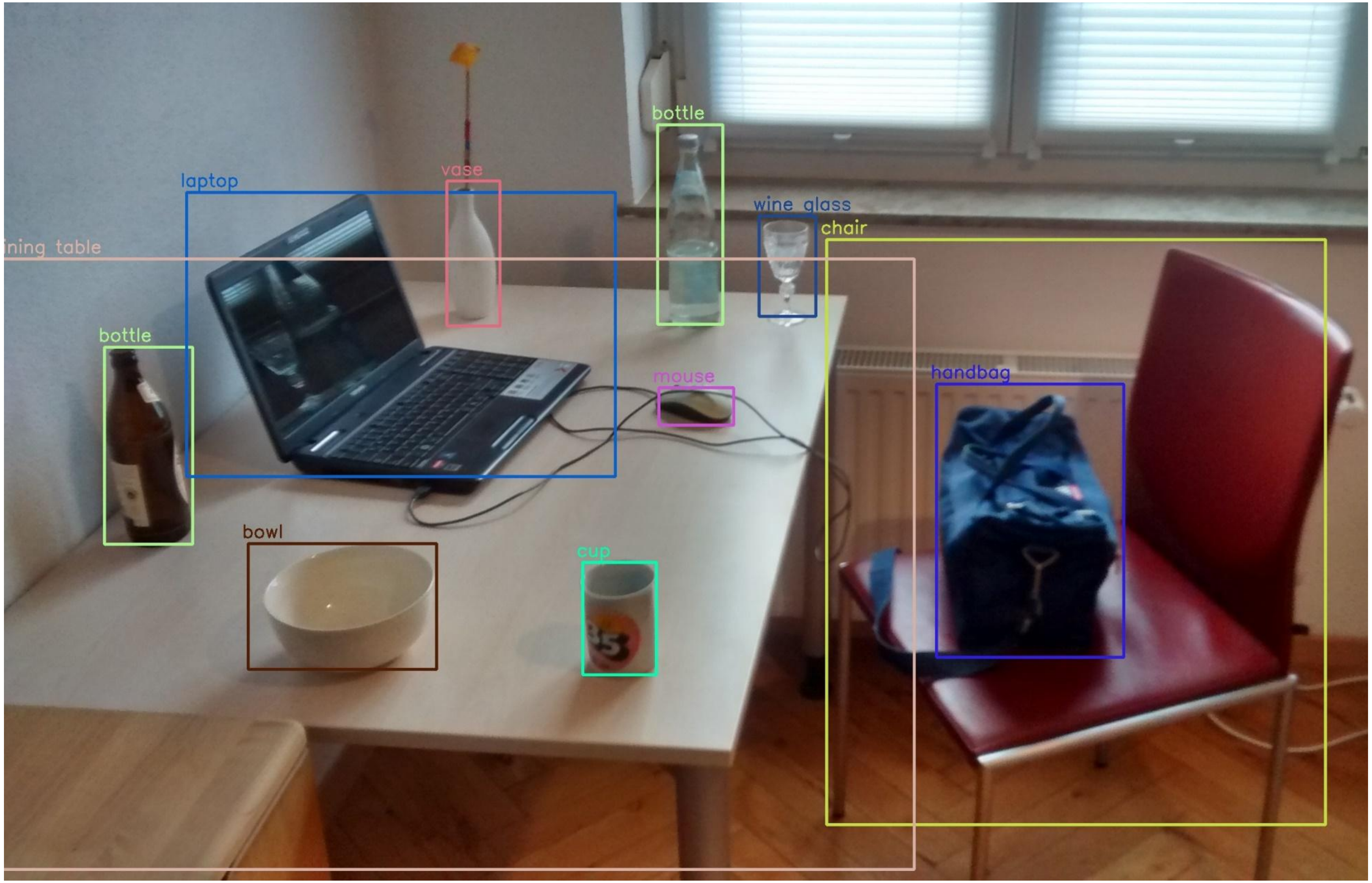
- Which objective function should be trained?

Approach A: Loss of head 1 + loss of head 2

Approach B: Fix backbone + classification head, only train localization head

8

Real case: multiple objects in an image

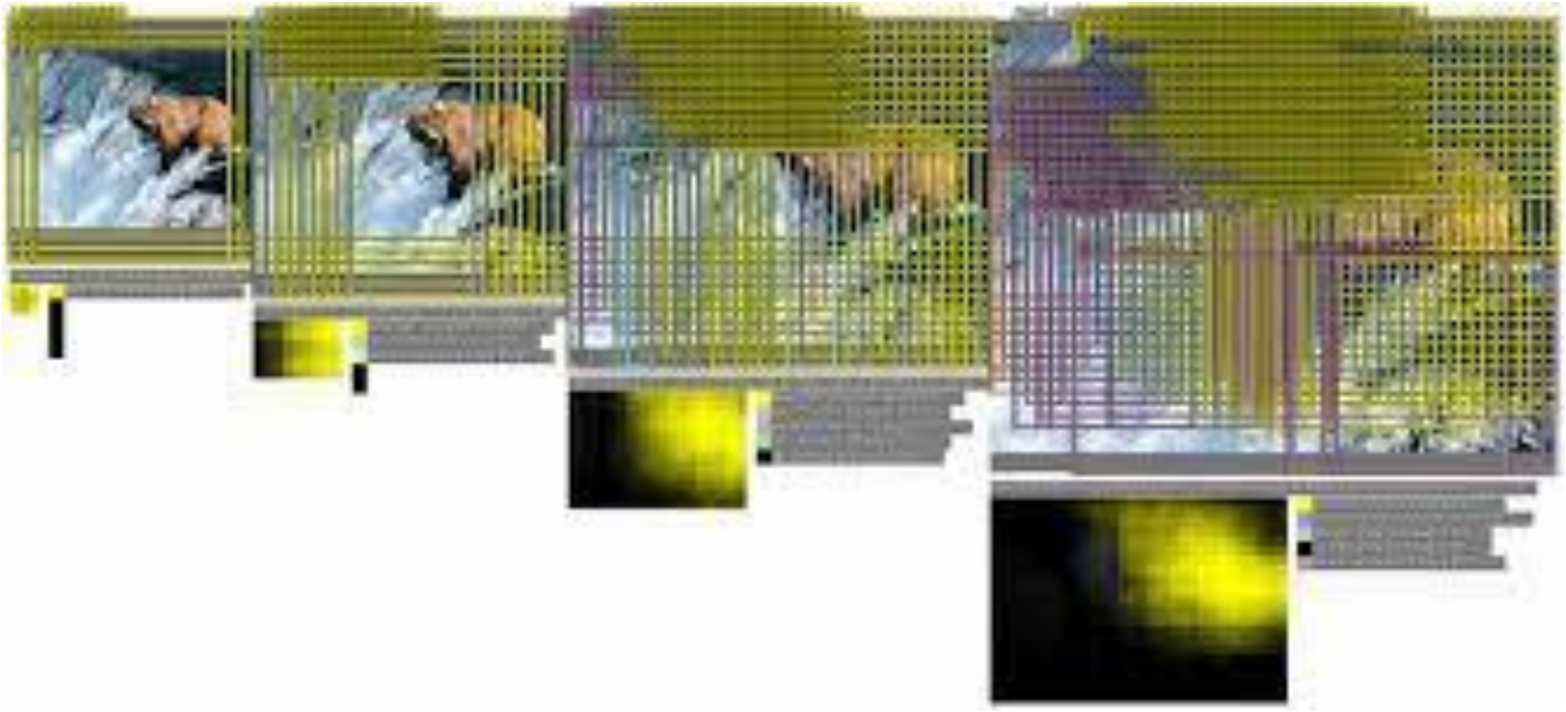


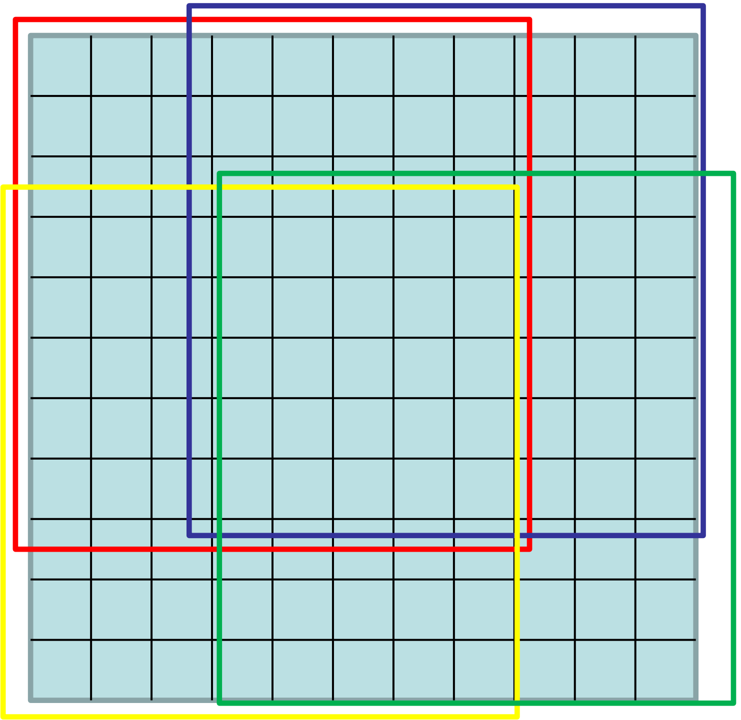
https://en.wikipedia.org/wiki/Object_detection

Sliding window approach

Idea 1: Sliding Window Approach [1]

- Step 1:
  - Generate candidate windows, which are evenly distributed in the original image
  - In [1], different scales of the original image are used

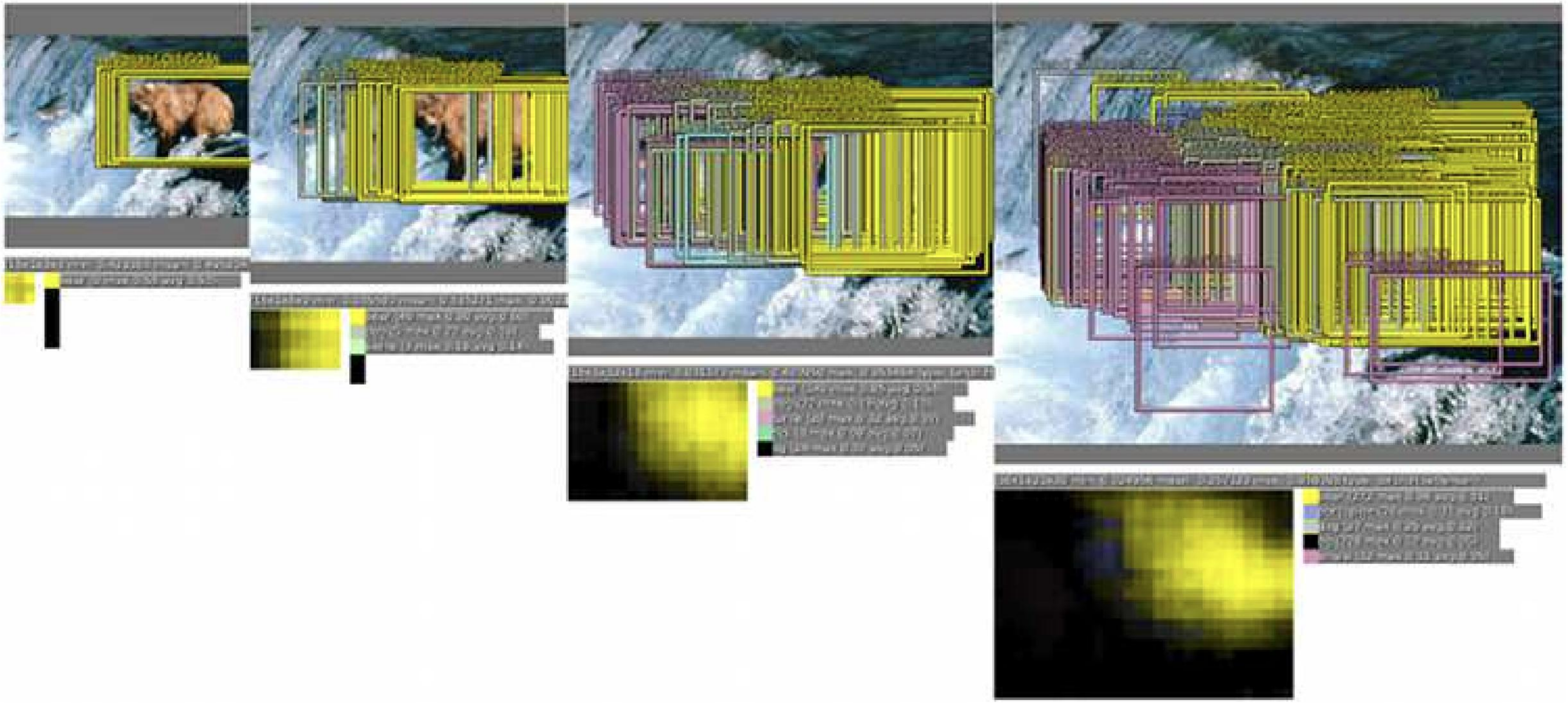




[1] Sermanet, et al., OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks, ICLR, 2014

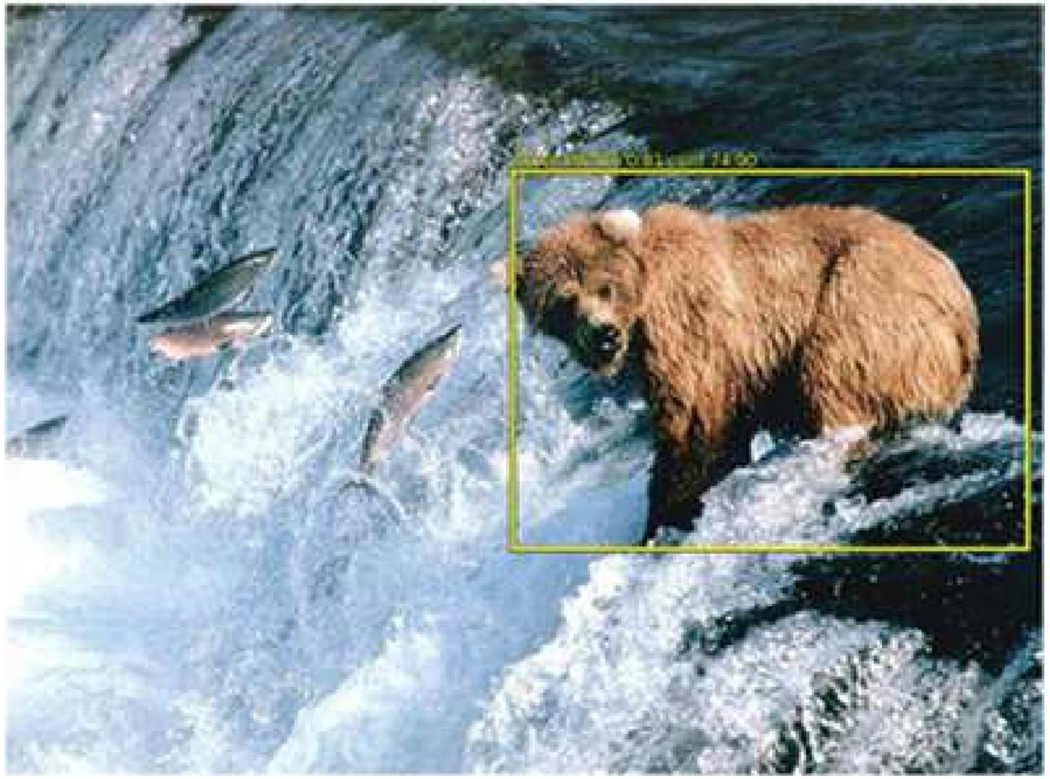
Idea 1: Sliding Window Approach [1]

- Step 2:
  - Classification prediction of each window (by e.g. ResNet)
  - Remove „background“ windows



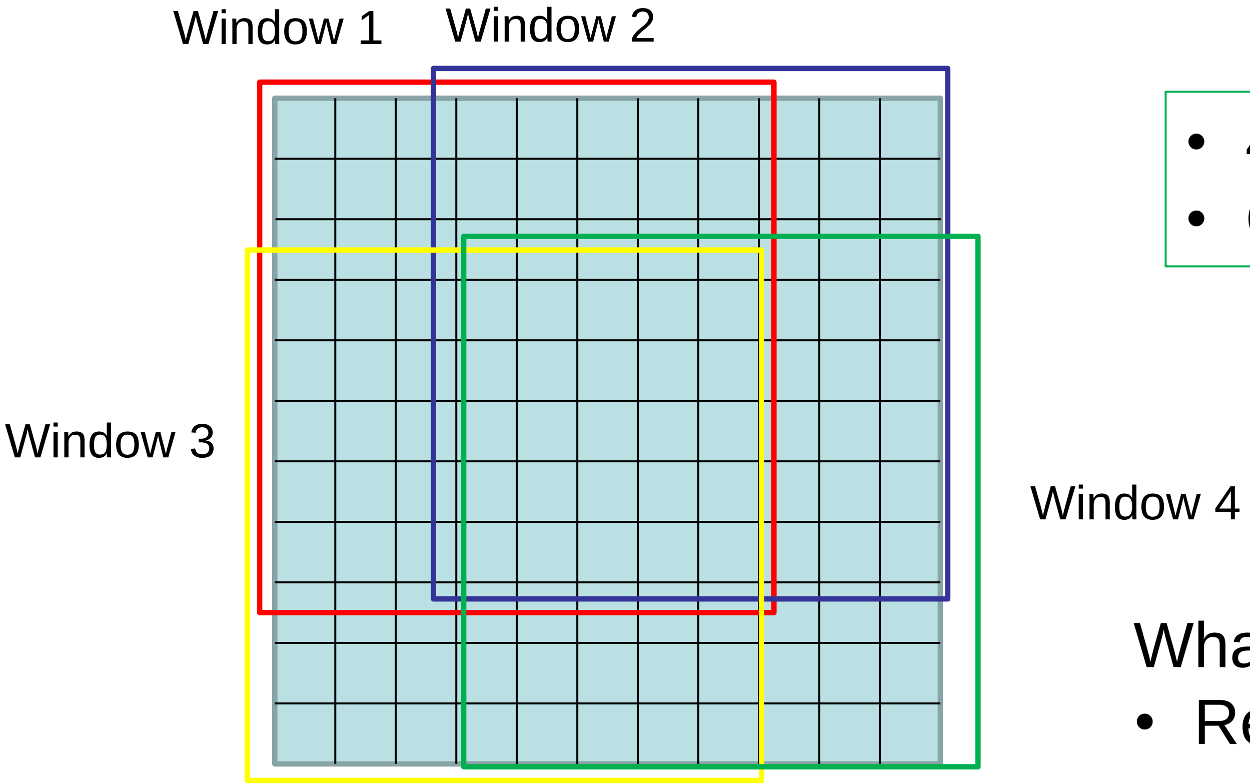
Idea 1: Sliding Window Approach [1]

- Step 3 (Post-precessing):
  - Combine and merge of resulted windows



Sliding Window Approach: Not effizient

- Step 1: Window Generation
- Step 2: Class prediction on each window (**Inefficient**)
- Step 3: Merge results



- 4 Windows ⇒ 4 Predictions
- Calling Deep NN 4 times (inefficient)

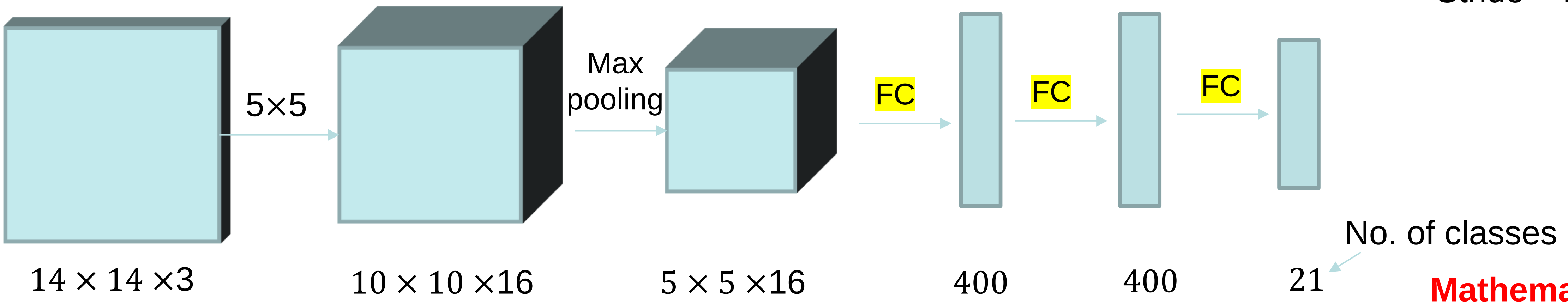
What to do?

- Reduce the number of calling Deep NN

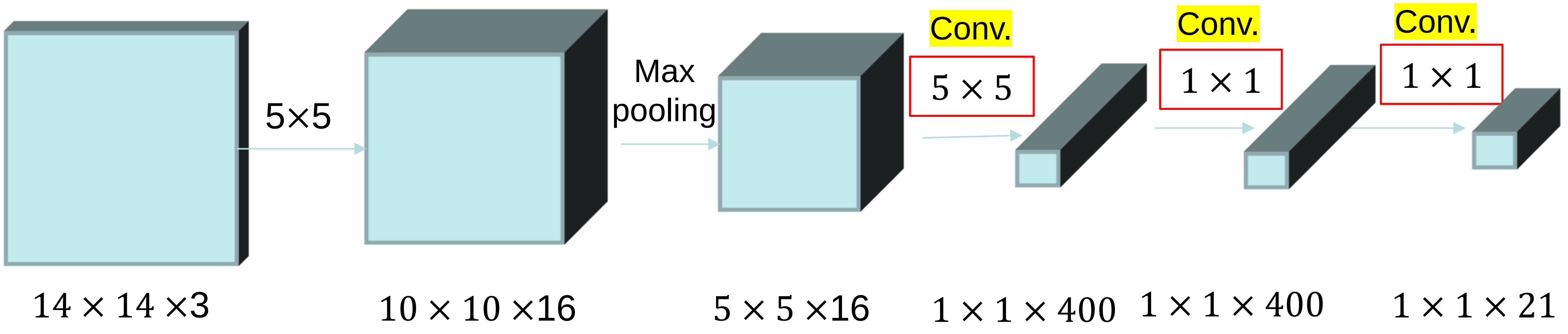
Recall: Represent Fully Connected Layers by Convolutions

- ❑ FC layers can be viewed equivalently as convolutions

- No padding
- Stride = 1



Mathematically equivalent!!



Recall: Represent Fully Connected Layers by Convolutions

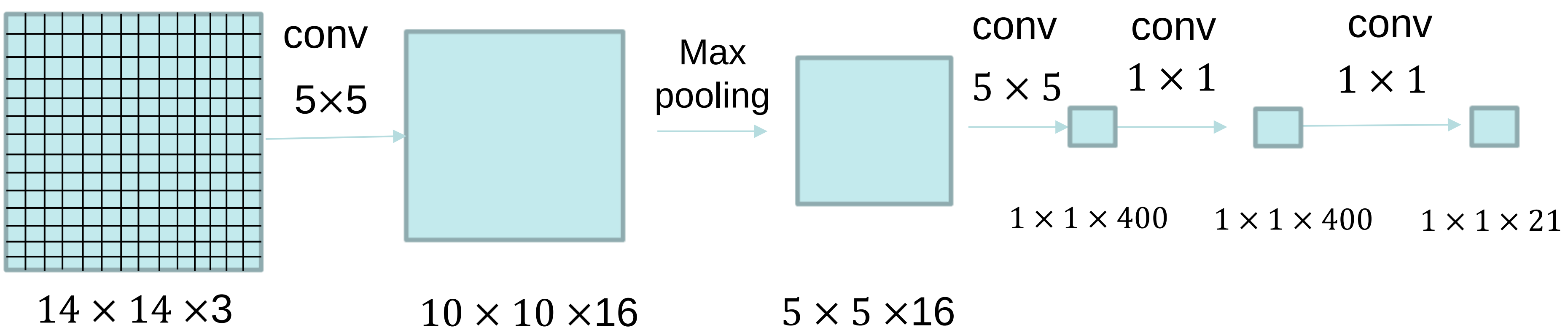
- By selecting proper kernel sizes, fully connected layers can be represented by convolutions **equivalently**
- Mathematically, there is no difference between fully connected layers and convolution layers
- Classification networks, e. g. VGG16, AlexNet, can be viewed as fully convolutional networks (only convolutions, no fully connected layers)

Sliding Window Approach: Not effizient

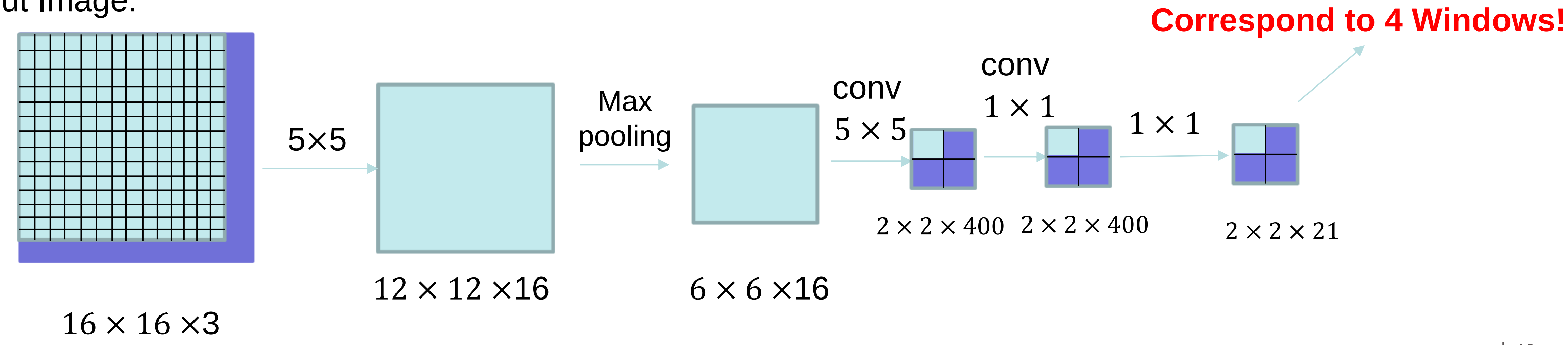
- Idea to reduce the total number of calling Deep NN
  - Replace classical classification network (with fully connected layers) by **fully convolutional network**
  - Do not call classification network for multiple times. But call the transformed fully convolutional network **only once**
  - It is called „**Convolution Implementation of Sliding Window**“

Convolution Implementation of Sliding Window

One Window:

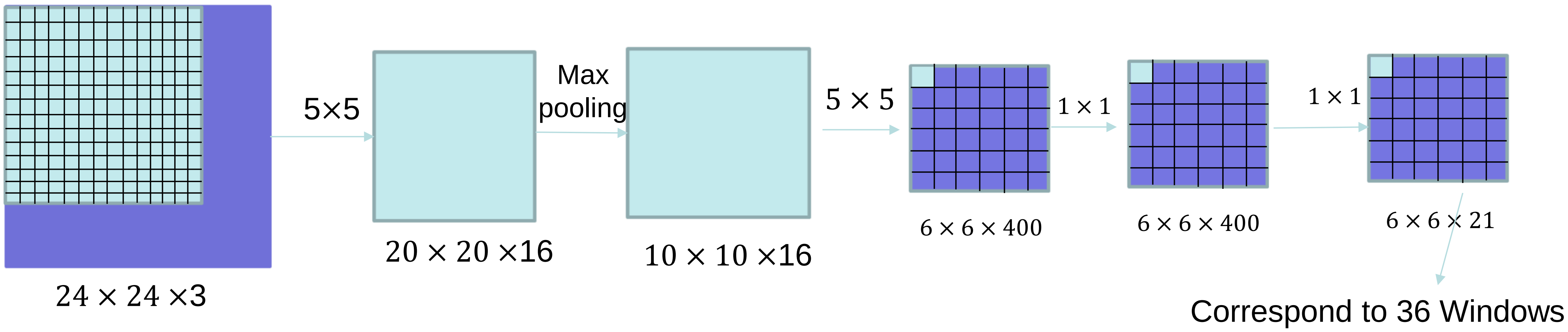


Input Image:



Convolution Implementation of Sliding Window

Input Image:



In Sum:

- ❑ Calculate the class predictions of **all windows** by calling a single convolution network for only once
- ❑ No need to call deep NN for multiple times

Sliding Window Approach: Summary

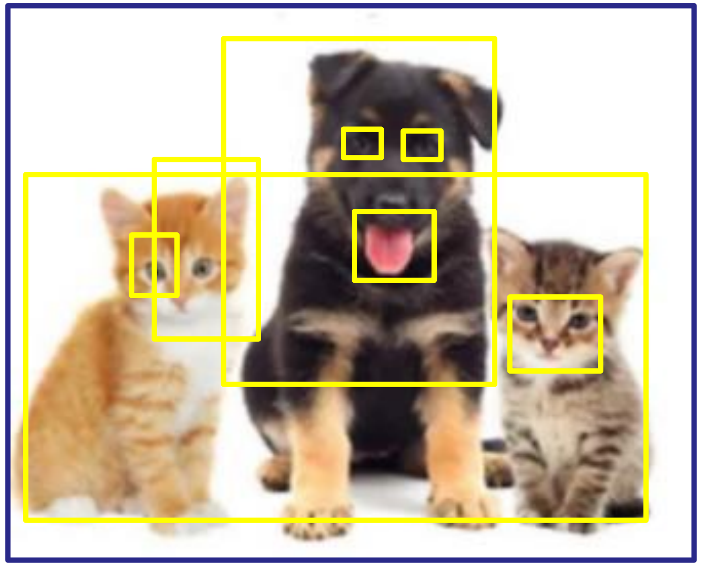
- Utilizing classification networks
- Efficient implementation exists (Convolution Implementation of Sliding Window)
- No explicit prediction of the coordinates of bounding boxes
- Bounding boxes are generated in post-processing step (no regression task is solved)

Content

- Introduction
- Sliding window approach
- Region-proposal network
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN

R-CNN [2]

Region proposal



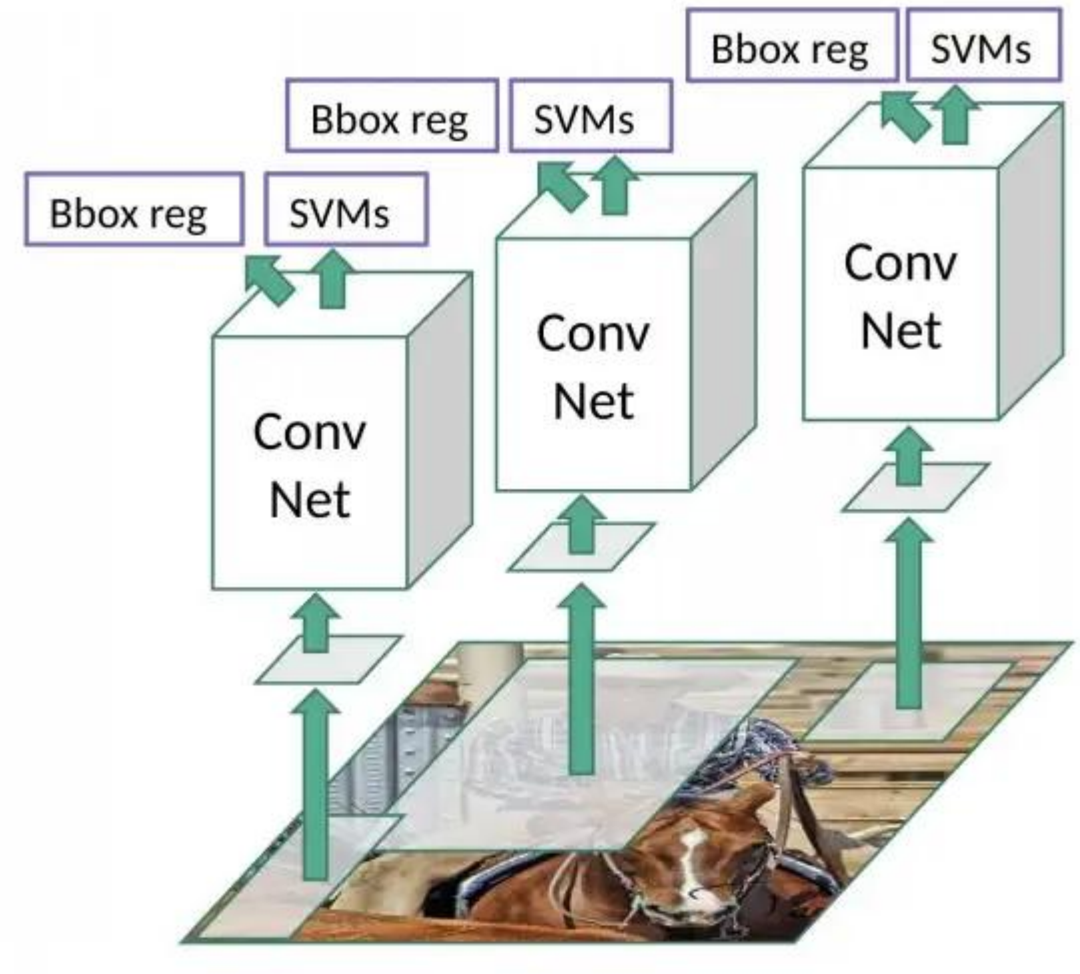
~2000 Region proposals

Algorithm:

- Find image regions which are likely to contain objects (no AI)
- Feed each region proposal: **classification** + **Bbox regression**
- Post-processing: merge results

[2] Girshick, et al., Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR, 2014

R-CNN [2]



Bounding box classification by SVM
Bounding box regression by Linear Regression

Computation of **feature maps**

Feed each region to network

Resize proposals

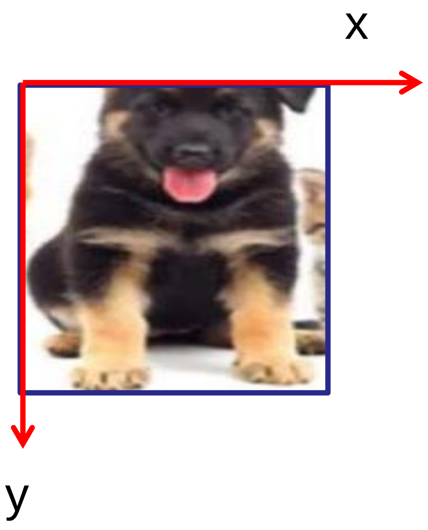
Region proposal

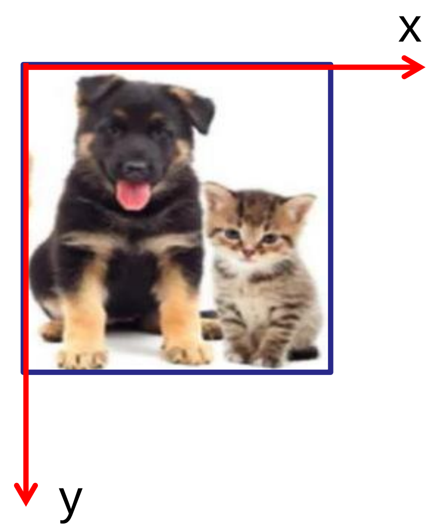
[2] Girshick, et al., Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR, 2014

R-CNN: Bounding Box Regression

- For each region, train a linear regression model to predict its **offsets** to the GT bounding box

Training Regions
(Inputs)





Regression Targets
(dx, dy, dw, dh)

GT=(0, 0, 0, 0)
Proposal is good

GT=(0, -10, 0, 0)
Proposal is too low

GT = (0, 0, -100, 0)
Proposal is too wide

R-CNN: Region Proposal Methods

Method	Approach	Outputs Segments	Outputs Score	Control #proposals	Time (sec.)	Repea-tability	Recall Results	Detection Results
Bing [18]	Window scoring		✓	✓	0.2	★★★	★	·
CPMC [19]	Grouping	✓	✓	✓	250	-	★★	★
EdgeBoxes [20]	Window scoring		✓	✓	0.3	★★	★★★	★★★
Endres [21]	Grouping	✓	✓	✓	100	-	★★★	★★
Geodesic [22]	Grouping	✓		✓	1	★	★★★	★★
MCG [23]	Grouping	✓	✓	✓	30	★	★★★	★★★
Objectness [24]	Window scoring		✓	✓	3	·	★	·
Rahtu [25]	Window scoring		✓	✓	3	·	·	★
RandomizedPrim's [26]	Grouping	✓		✓	1	★	★	★★
Rantalankila [27]	Grouping	✓		✓	10	★★	·	★★
Rigor [28]	Grouping	✓		✓	10	★	★★	★★
SelectiveSearch [29]	Grouping	✓	✓	✓	10	★★	★★★	★★★
Gaussian				✓	0	·	·	★
SlidingWindow				✓	0	★★★	·	·
Superpixels		✓			1	★	·	·
Uniform				✓	0	·	·	·

[3] Hosang, et al., What Makes for Effective Detection Proposals?, PAMI, 2015

R-CNN: Summary

Improvements (compared with Sliding Window Approach):

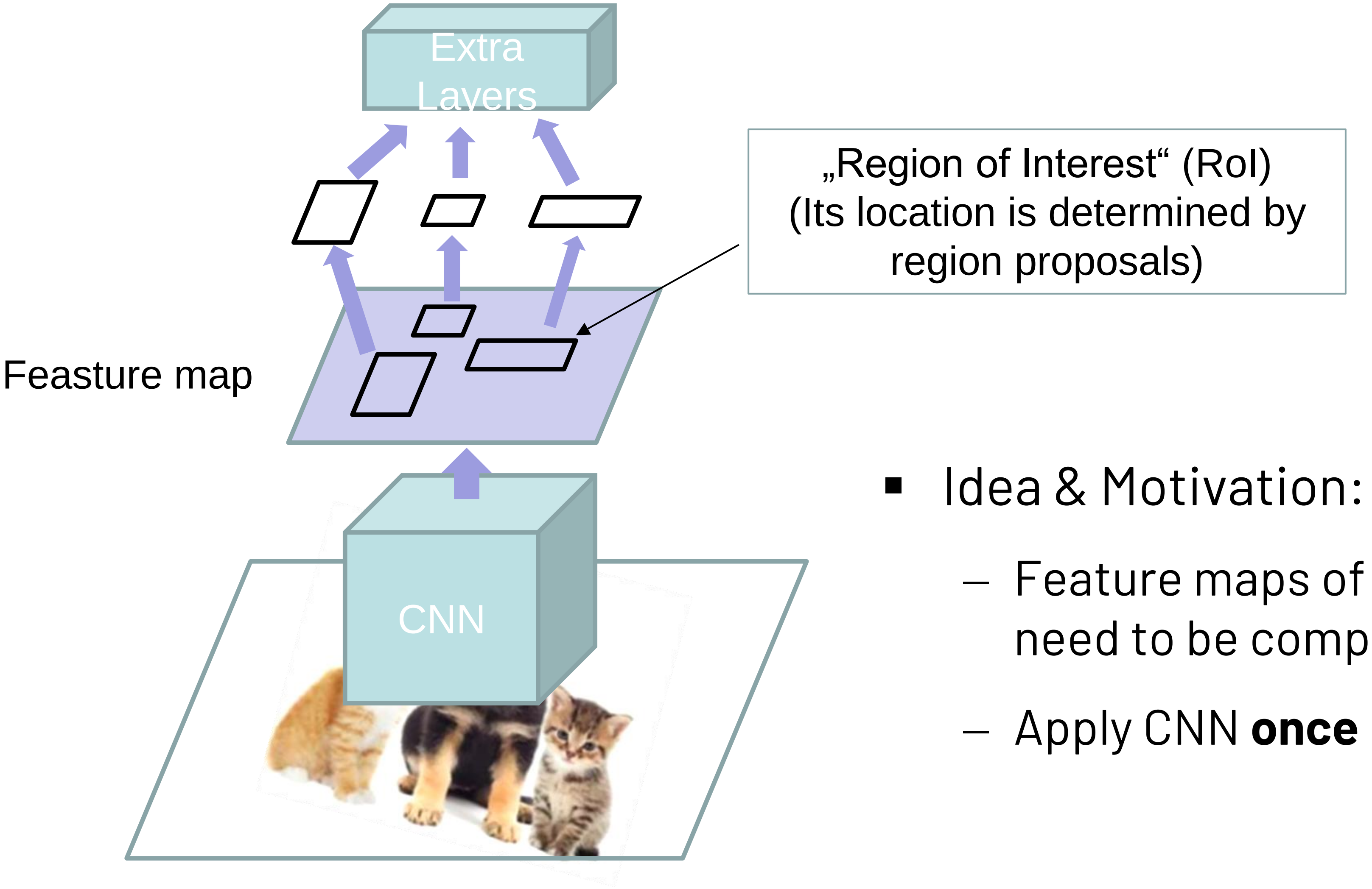
- Reduced number of „candidate windows“
- Classification + Bounding box regression

R-CNN: Problems

1. Performance inefficiency
   - Each image has 2000 proposals.
   - CNN must be runned for 2000 times !
2. Only classification & regression layers are trained
   - Feature maps are computed by CNN. They are inputs for SVM or regression layers
   - CNN is **not** trained together with classification & regression layers (suboptimal)
3. A complex system
   - Complicated to train them together („end“-to-“end“)

Fast R-CNN

Fast R-CNN [4]



- Idea & Motivation:
  - Feature maps of each region do not need to be computed repeatedly
  - Apply CNN **once**

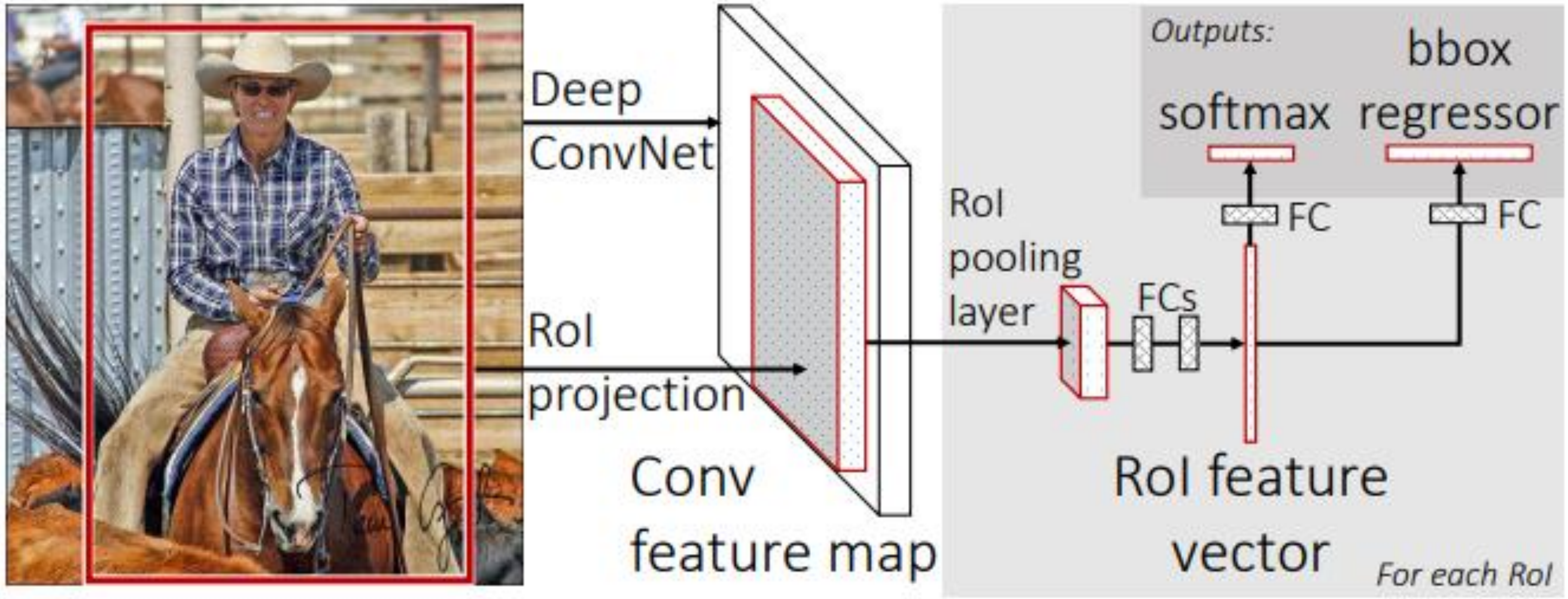
[4] Girshick, Fast R-CNN, ICCV, 2015

Fast R-CNN [4]

Method:

- Step 1: Apply CNN to get a feature map of the **whole** orignal image (saves time!)
- Step 2: Compute region proposals by non-AI algorithms
- Step 3: Crop features in the feature map („RoI“)
- Step 4: Feed each „RoI“ to classificatin & regression layers

Fast R-CNN [4]



- Feature map of the entire image is cropped using the projected "RoI".
- The network has two heads. One for classification, the other for bbox regression.

Fast R-CNN [4]: Summary

- Improvements (Compared to R-CNN):
  - No need to run CNN for 2000 times for a single image

- Disadvantage:
  - The method's performance depends on the proposed regions and region proposal methods
  - Generating region proposals slows down the prediction during test time
  - Not „End-to-End" trainable. **Why?**

„End-To-End" Trainable

- An end-to-end trainable neural network is one, where all parameters of the network can simultaneously updated, when optimizing one loss function.

- Classical deep CNN can be denoted as:

$$y = f(x, \theta)$$

x: input image
θ: parameters of all layers
f: mapping of all layers

- Training process (= "updating θ", "process of optimization"):

$$\theta_{k+1} = \theta_k + \lambda \frac{\partial f}{\partial \theta}$$

k: training step
λ: learning rate
$\frac{\partial f}{\partial \theta}$: gradients

Network is „end-to-end" trainable, if gradients $\frac{\partial f}{\partial \theta}$ are available

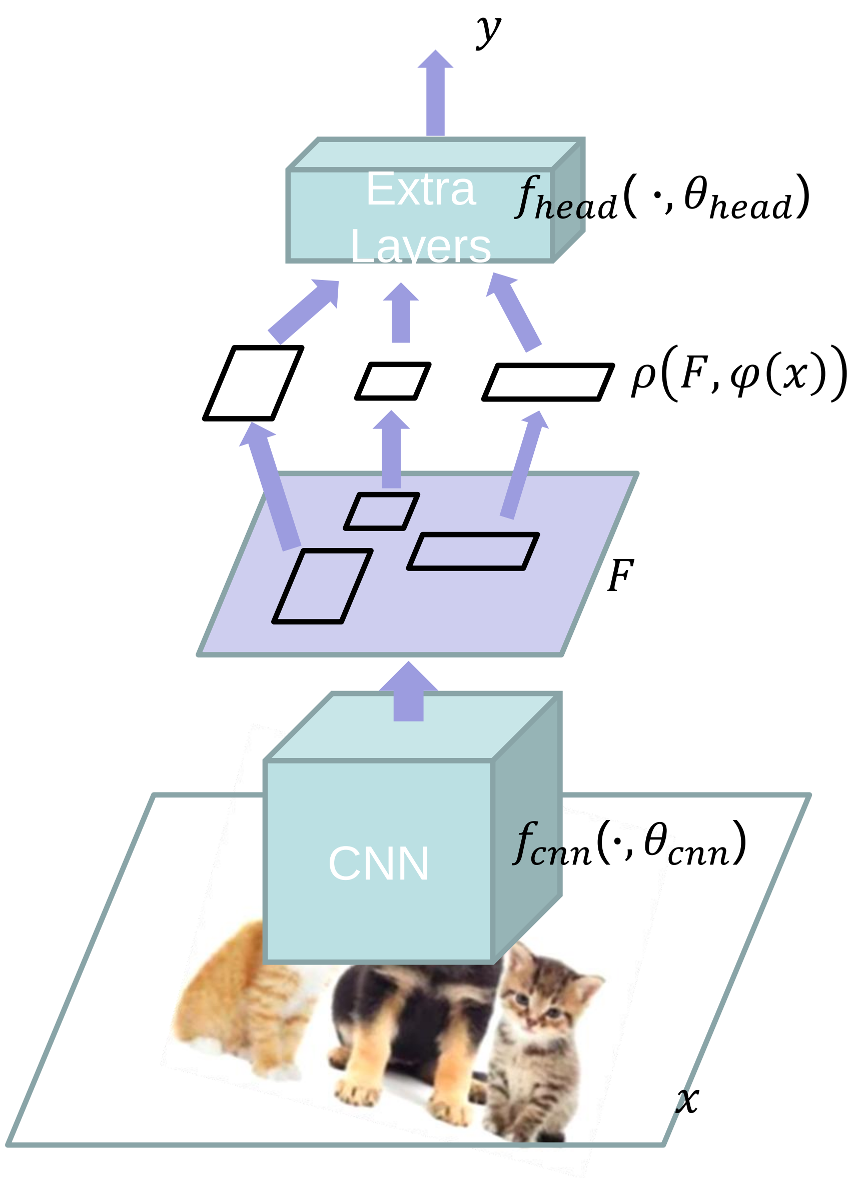
Fast R-CNN: „End-To-End“ Trainable?

- Input image: $x \epsilon \mathbb{R}^{H \times W}$
- Feature map of the entire image: $F = f_{cnn}(x, \theta_{cnn}) \epsilon \mathbb{R}^{A \times B}$
- Coordinates of RoI: $(o_x, o_y, \mathrm{w}, \mathrm{h}) = \varphi(x) \epsilon \mathbb{R}^4$
- **Cropped features of RoI:** $\rho(F, \varphi(x)) \epsilon \mathbb{R}^{w \times h}$
- Output of Network:

$$y = f_{head}(\rho(F, \varphi(x)), \theta_{head})$$

$$= f_{head}\{\rho[f_{cnn}(x, \theta_{cnn}), \varphi(x)], \theta_{head}\}$$

- $\frac{\partial y}{\partial \theta_{cnn}}$ is unavailable, because $\frac{\partial \rho}{\partial F}$ is unavailable!



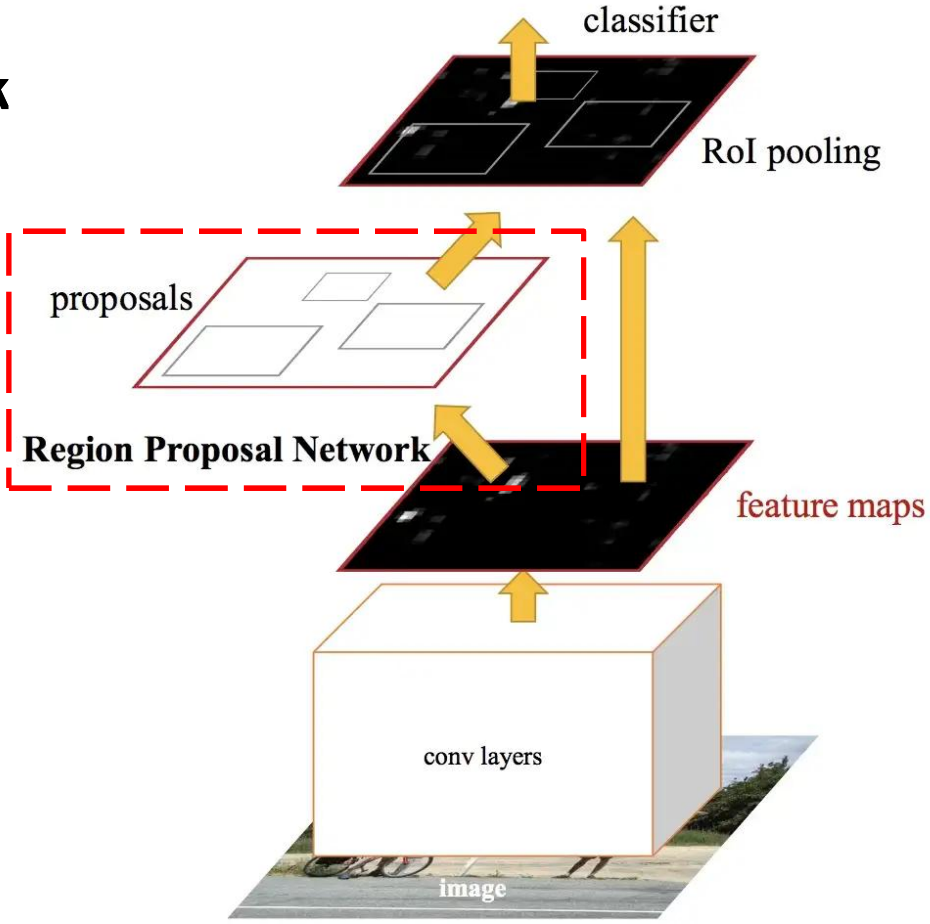
Fast R-CNN: „End-To-End" Trainable?

- $\frac{\partial y}{\partial \theta_{cnn}}$ is unavailable, therefore the backbone CNN is not "end- to-end" trainable (from image x to heads' output y)
- $\frac{\partial y}{\partial \theta_{head}}$ is available, therefore heads can be "end-to-end" trained
- The network is not "end-to-end" trainable, because the partial gradients of the cropping function $\rho\big(F, \varphi(x)\big)$ with respect to the feature map F can not be obtained
- Fast R-CNN is a "two-stage" approach

Faster R-CNN

Faster R-CNN [5]

- Idea
  - **Module 1: Use Region Proposal Network (RPN) to propose region´s coordinates**
    - Much faster than region proposal algorithms. Why?
  - Module 2: Use classifier (the same as R-CNN) to classify each region
  - Propose „**Anchor box**“
    - an influential idea!



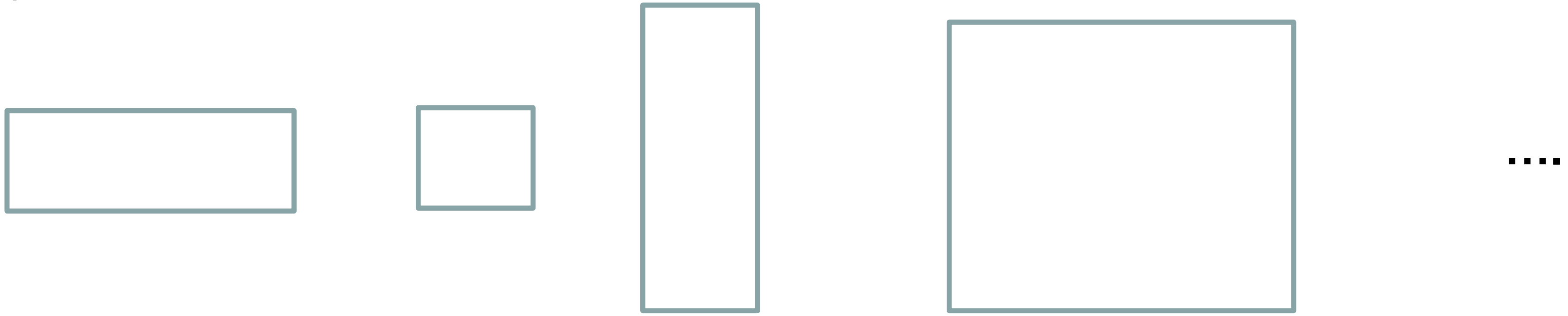
[5] Ren, et al., Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS, 2015

Anchor Box

- Definition:

  „Anchor boxes are bounding boxes, whose scale (height, width) and position is specified and fixed.“

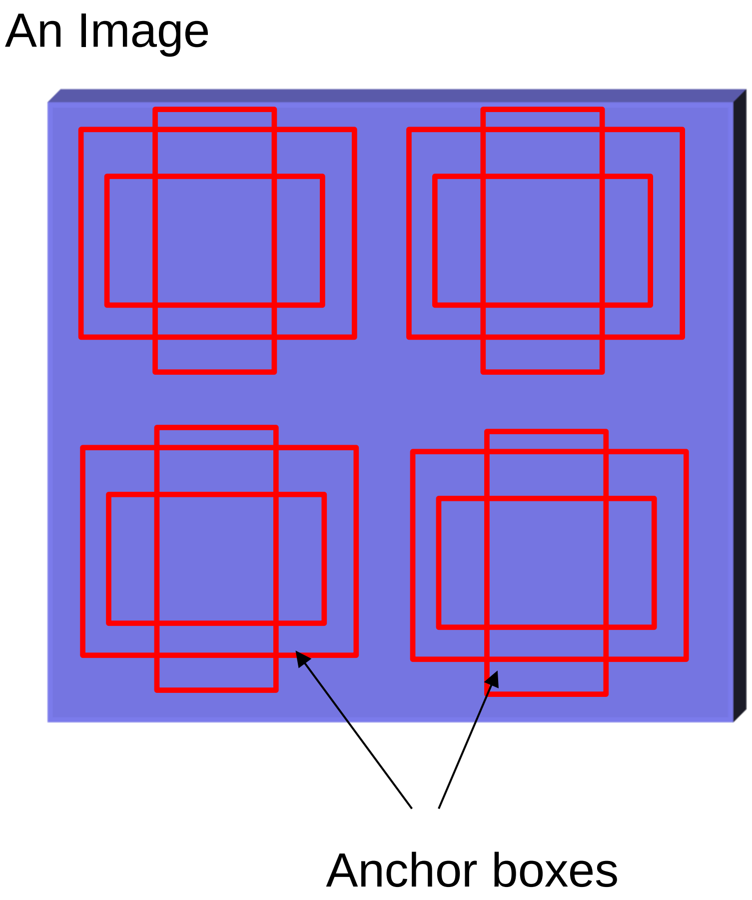
- Examples:



- Anchor boxes are usually determined and fixed before training (Model’s hyper-parameter)
- Firstly, introduced in [5]
- Simple, but a very import concept

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

Anchor Boxes of an Image: example



- The position, shape and dimension of anchor boxes are fixed in an image, which are design's hyperparameter
- An image may have multiple anchor boxes
- Anchor boxes are usually placed in **subgroups**. Each subgroup has the same center (left figure).
- They usually cover the whole image, so that objects at different places of an image can be detected.

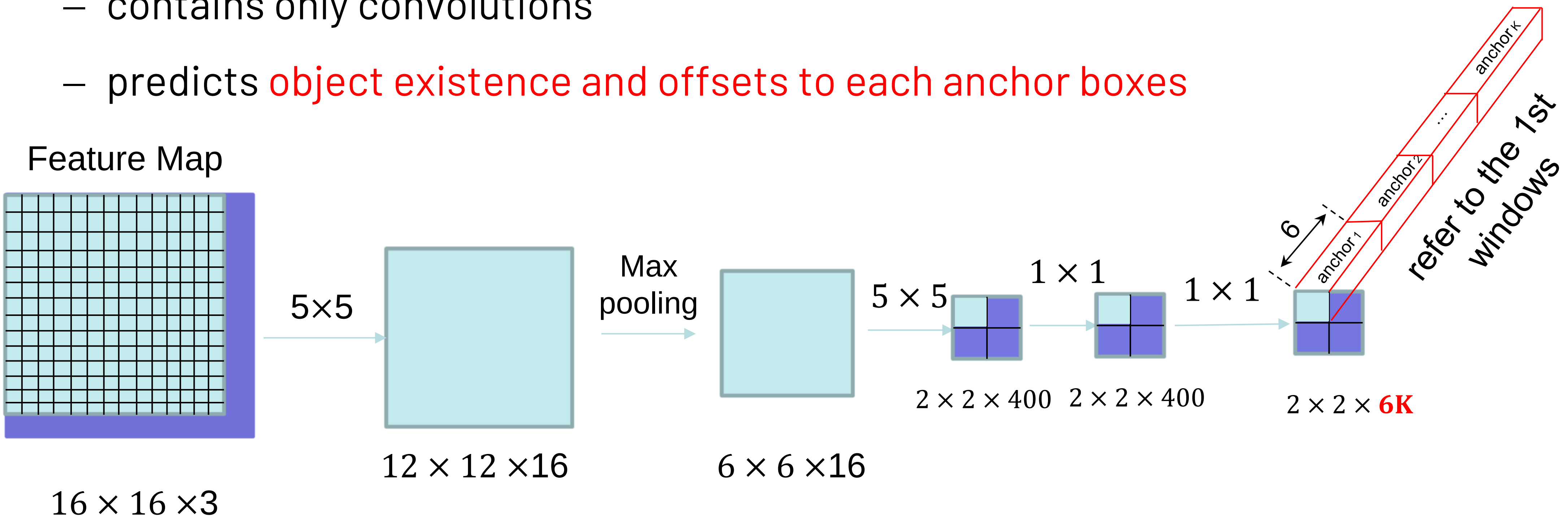
Anchor Boxes: Why need?

Why need?

- We do not want to predict the absolute coordinates of bbox, but their offsets to some reference boxes (="anchor box")
- Anchor boxes are locational references for the ground-truth and predicted bounding boxes

Region Proposal Network (RPN)

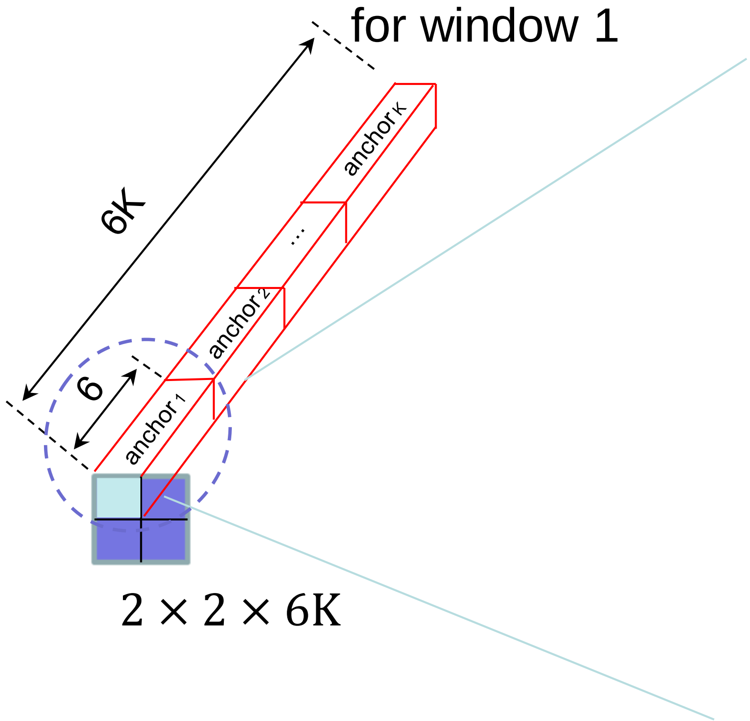
- **Idea**: Multi-scale anchors as regression's reference
- Consider that **in each subgroup/window** K anchor boxes are selected
- Region Proposal Network:
  - contains only convolutions
  - predicts object existence and offsets to each anchor boxes



K: the number of anchors for each window

Region Proposal Network

- Output of Region Proposal Network



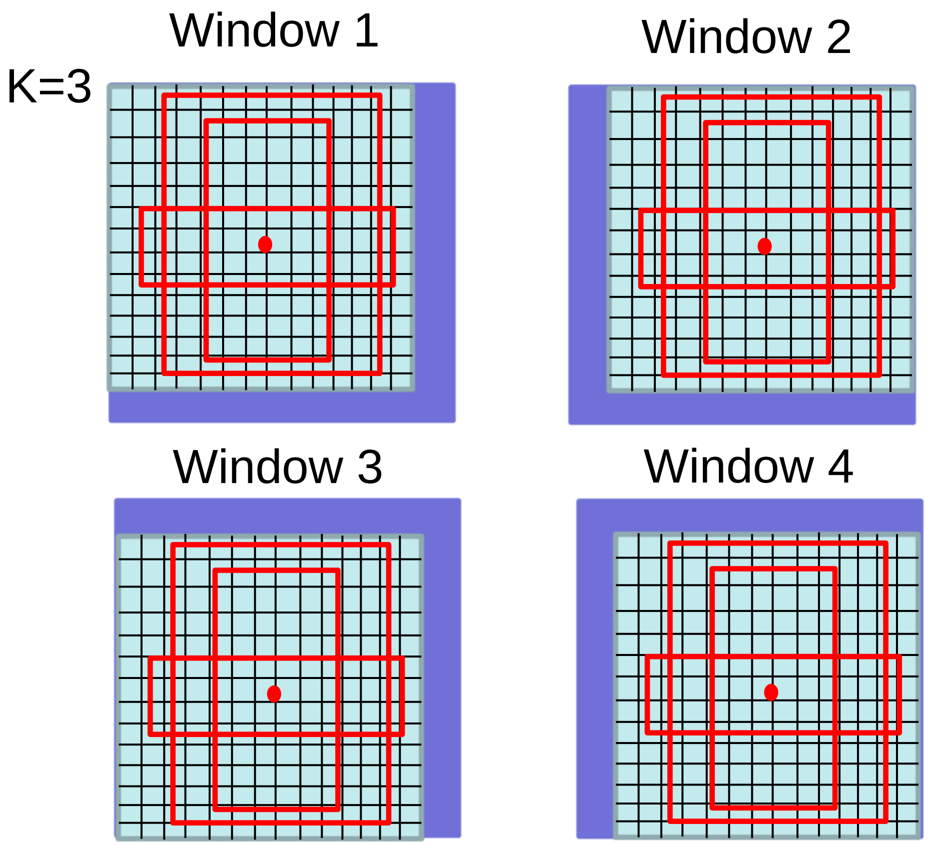
Exist= 0.94	NotE. =0.06	Dx=10	Dy= -5	Dw =8	Dh=-6

existente of an object in anchor 1 (classification)

Object´s offsets to anchor 1 (regression)

Region Proposal Network

- The position of 4 windows and the corresponded anchor boxes of each window

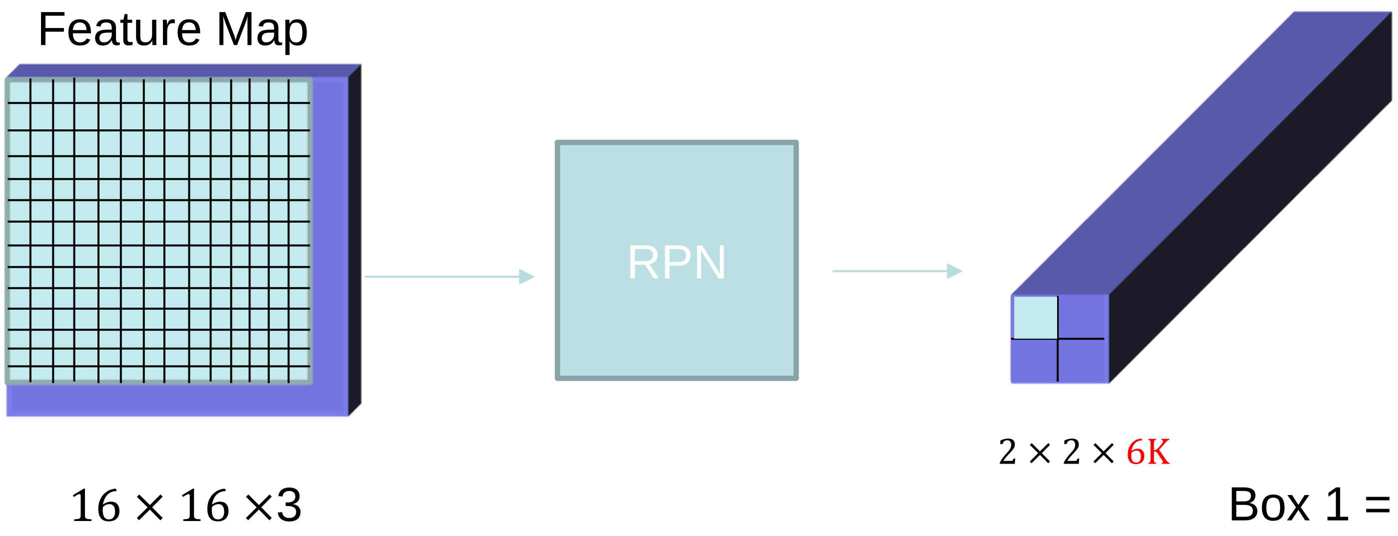


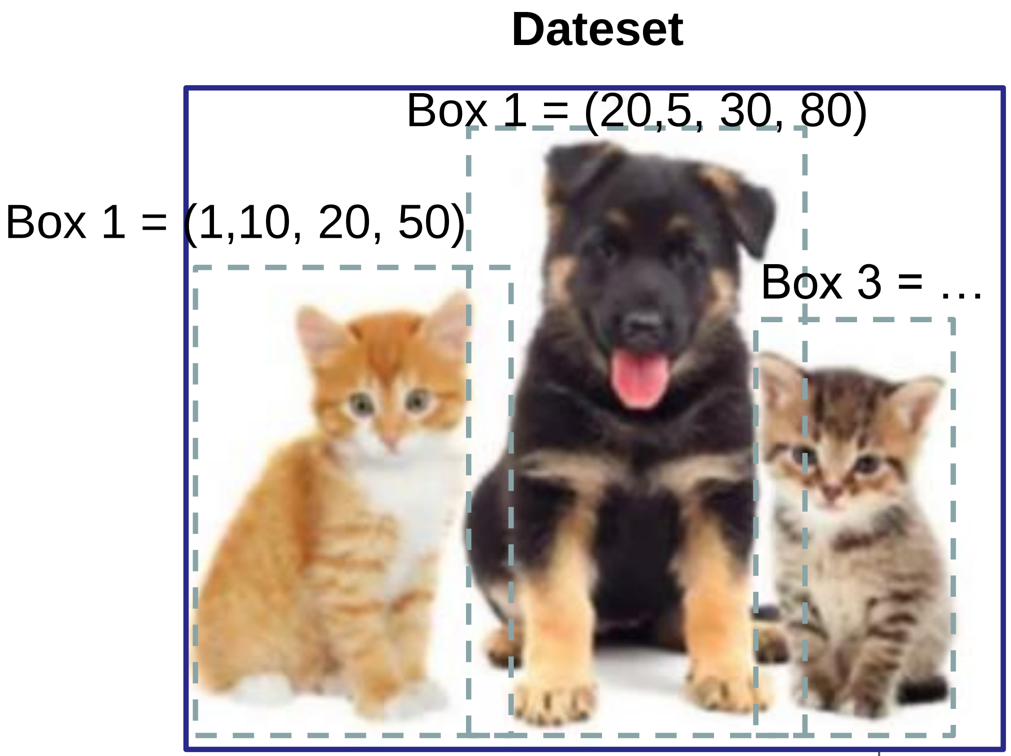
- Anchor boxes are centered in each window
- Their positions and dimension are fixed
- In this figure, #Window=4, K=3
- In [5], #Window=2400 (?), K=9

ratio (Figure 3, left). By default we use 3 scales and 3 aspect ratios, yielding $k = 9$ anchors at each sliding position. For a convolutional feature map of a size $W \times H$ (typically ~2,400), there are WHk anchors in total.

Region Proposal Network (RPN): Ground Truth

- ❑ RPN takes an input image and outputs a tensor:





- ❑ What do we need to train RPN?

⇒ GT value of tensor $2 \times 2 \times 6K$

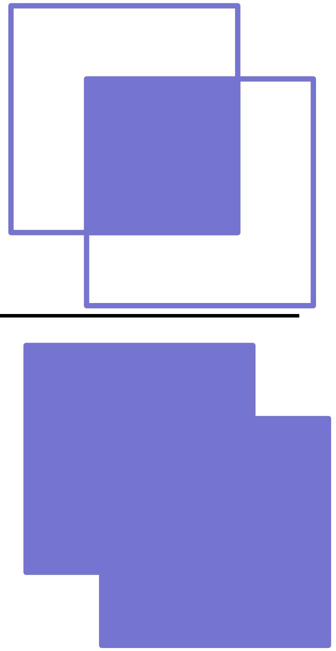
- ❑ Classical dataset contains only the GT location of bounding boxes (right figure)

⇒ How to get the GT value of this tensor from the dataset?

Recall: Intersection-over-Union (IoU)

For any two bounding boxes:

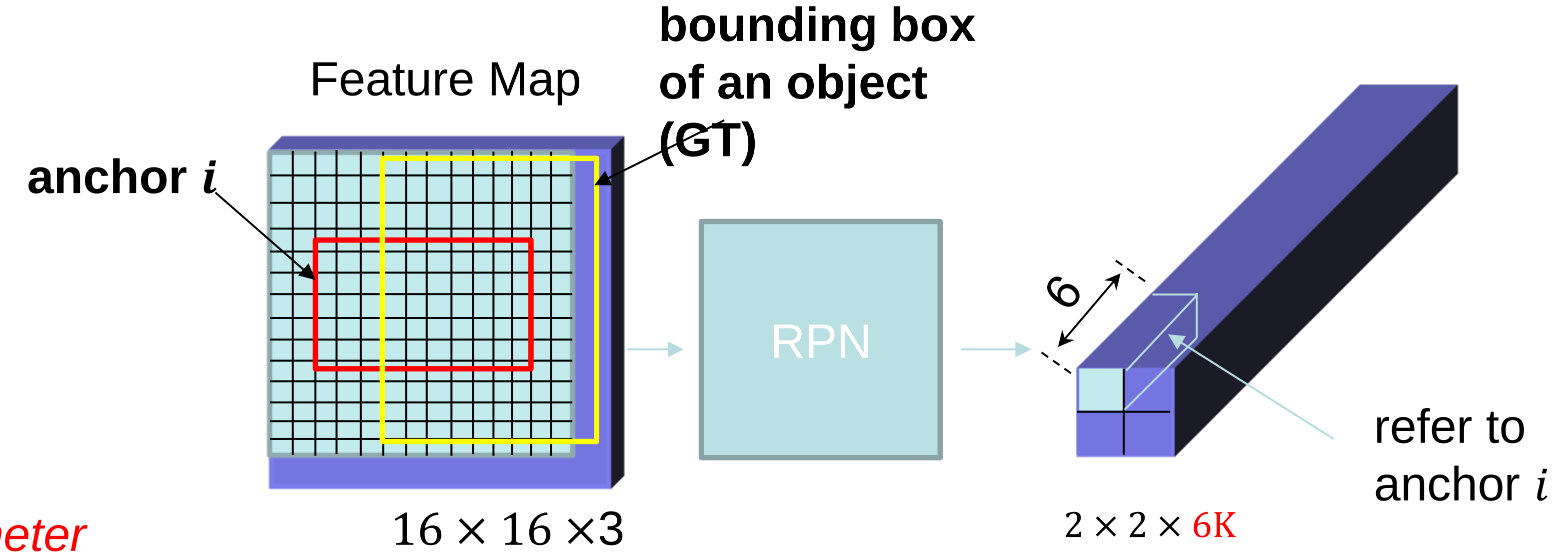
$$IoU = \frac{Area\ of\ Intersection}{Area\ of\ Union}$$



- IoU is a number that quantifies the degree of overlap between two boxes
- IoU can be used to evaluate e.g. the overlap of the GT box and prediction region, the overlap of GT box and anchor box
- $IoU \in [0, 1]$

Region Proposal Network (RPN): GT Creation

- To train RPN, each anchor is assigned with 6 positions in the output tensor
- The values of these 6 positions are calculated based on **IoU**



hyperparameter

- If IoU≥ 0.7, an object exists.

Exist= =1	NotE. =0	dx =10	dy= -5	dw =-8	dh =12

GT for anchor i

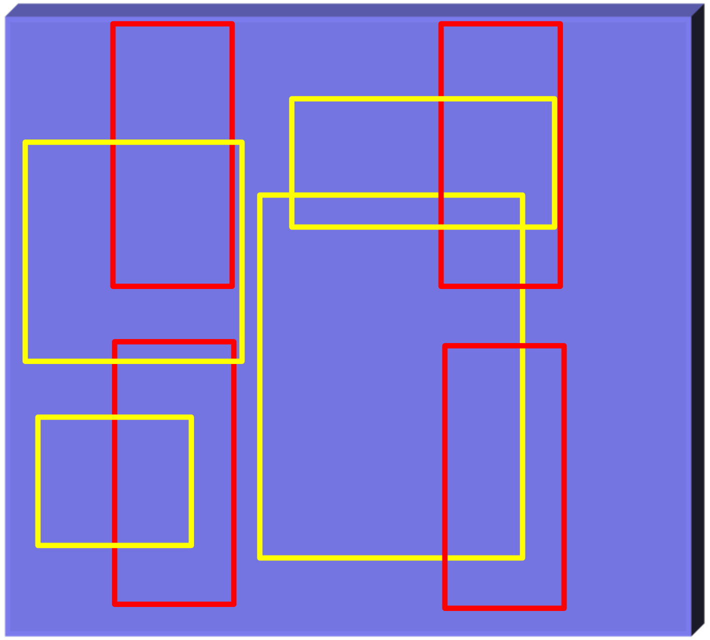
- If IoU< 0.7, no object exist.

Exist= =0	NotE. =1	dx= any	dy= any	dw = any	dh= any

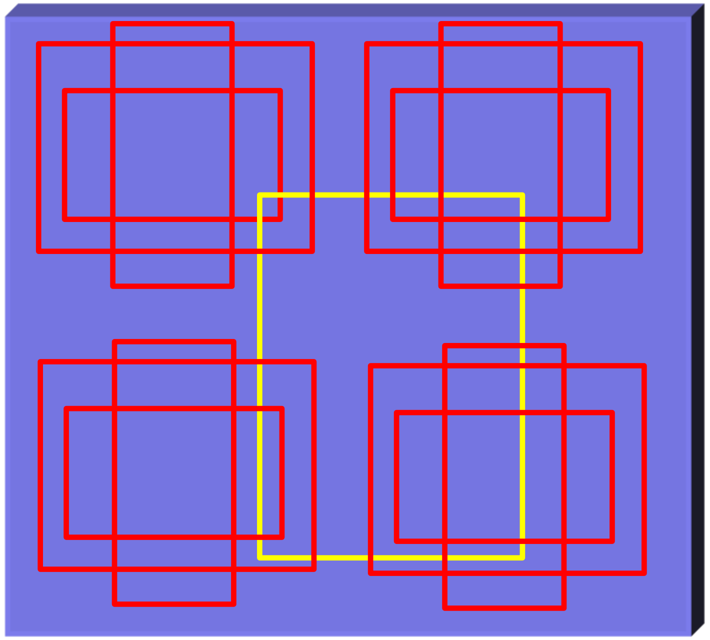
GT for anchor i

Region Proposal Network (RPN): Different Situations

- Single anchor (for each window), multiple objects

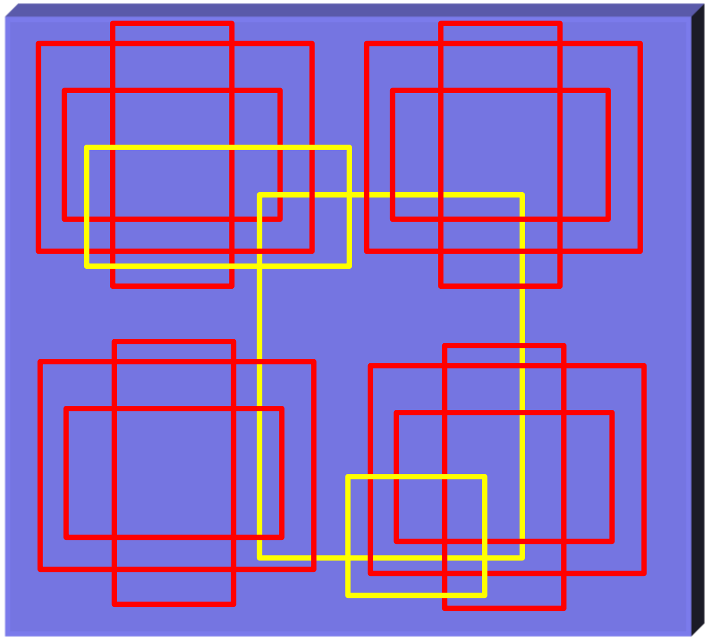


- Multiple anchors, single Objects



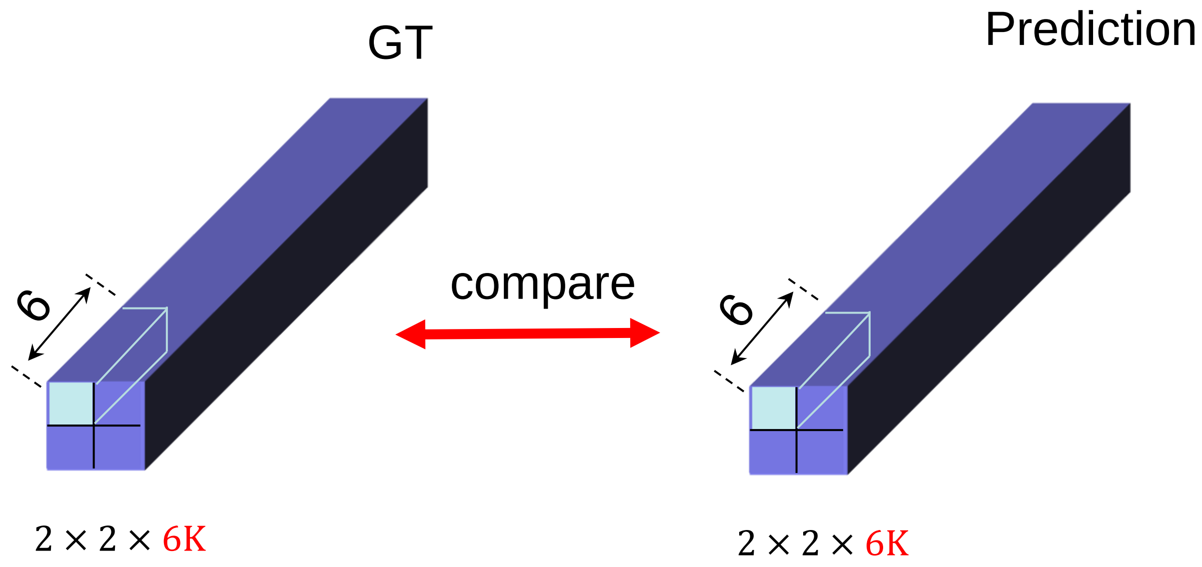
Region Proposal Network (RPN): Different Situations

- Multiple anchors, multiple Objects
  - Real case



- ❑ There is no unique and 100% correct solution to create GT tensor.
- ❑ Creating GT tensor for anchor boxes is an „**engineering**“ process
- ❑ Different solutions / steps exist

Region Proposal Network (RPN): Training



Element-by-element:

GT

Ex	NEx	dx	dy	dw	dh

compare

Prediction

Ex	NEx	dx	dy	dw	dh

Region Proposal Network (RPN): Training

	GT							Prediction					
Notation:	e_{gt}	n_{gt}	x_{gt}	y_{gt}	w_{gt}	h_{gt}	$\longleftrightarrow$	e_p	n_p	x_p	y_p	w_p	h_p
For example:	1	0	10	20	−5	−4		0.9	0.1	12	15	3	−3

- ❑ $\boldsymbol{Objective\ function = \ loss_{classification} + \ loss_{regression}}$

- $loss_{class.} = \sum_{all\ anchor} e_{gt} \log e_p + (1 - n_{gt}) \log n_p$
  - Binary classification
  - Cross-entropy

- $loss_{reg.} = \sum_{\boxed{all\ anchor\ with\ \boldsymbol{e_{gt}=1}}} (\boldsymbol{x_{gt} - x_p})^2 + (\boldsymbol{y_{gt} - y_p})^2 + (\boldsymbol{w_{gt} - w_p})^2 + (\boldsymbol{h_{gt} - h_p})^2$
  - Regression
  - Mean-Squared-Error

Only when an anchor box contains an object, GT value of bounding box is considered!

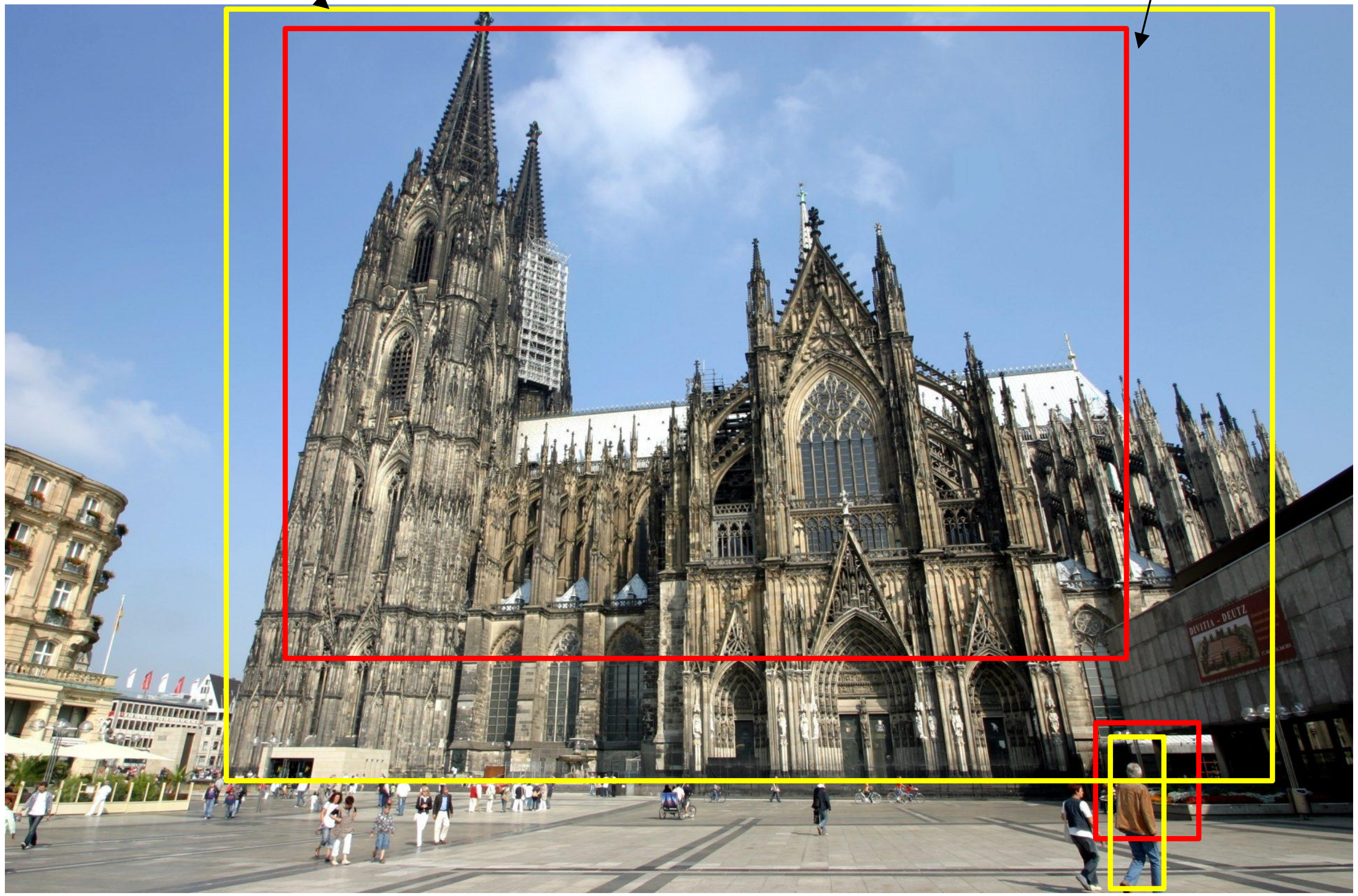
Why use anchor boxes?

- Anchors exist in different scale, i.e. size, height-width ratio
- Objects in different scales can be detected.
  - Big anchors responsible for big objects
  - Small anchors responsible for small objects
- Avoid pyramid of scaled images [6], avoid sliding windows with different sizes

  → more efficient

GT box of an object

Responsible anchor



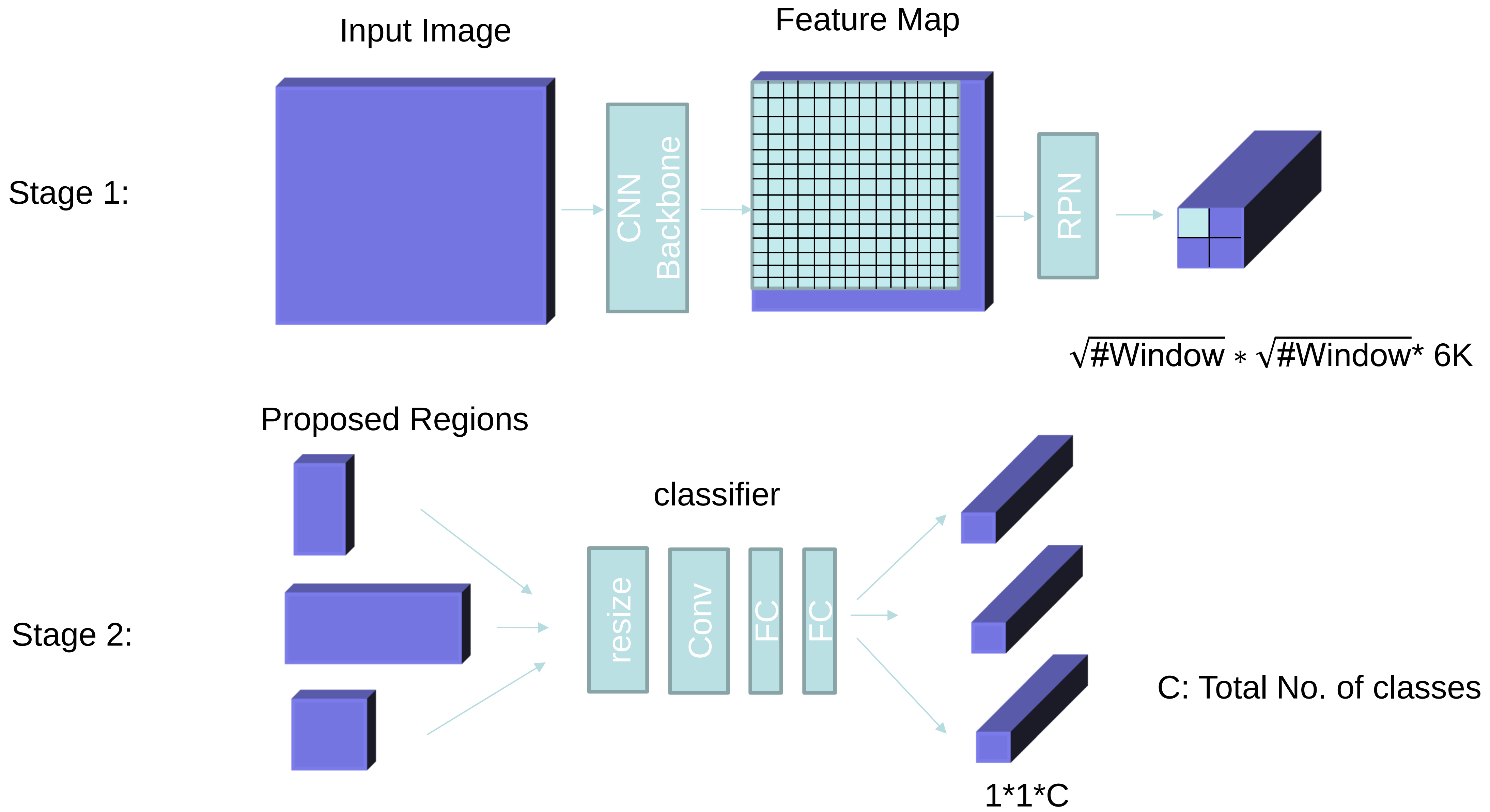
A Breakthrough Idea!

Anchor box: Limitations

- Total number of detected objects is limited to (#anchors × #windows)
  – In previous case, total objects ≤ K × 4
- Parameters of anchor box (K, size, ratio, …) and #windows are design hypermeters
  – Too many
  – Hard to tune

Hundreds of anchor boxes are needed!

Faster R-CNN Overview: Two-stage Approach



Faster R-CNN Overview: Training steps

- Training steps:
  - Step 1: Train RPN
  - Step 2: Train Classifier
    - RPN is called to generate region proposals
  - Repeat step 1 and 2 („alternating training“ [5])

Faster R-CNN Overview: Two-stage Approach

- **Question**: Can we train RPN + Classifier together („end-to-end“)?
  - Not trivial
  - Why?

- Therefore, faster R-CNN is a „two-stage“ approach

Faster R-CNN [5]: Summary

- Two modules: RPN, classifier/regression
- Firstly introduced „Anchor“ (promising!)
- RPN
  - Predicts the existence of objects and their coordinates
  - A fully convolutional network (FCN)
  - Can run on GPU. Faster than region proposal method (run on CPU)
  - Can be trained „end-to-end“
- A two-stage method
  - Two modules (RPN & Classifier) can not be trained „end-to-end“

Reference

[1] Sermanet, et al., OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks, ICLR, 2014

[2] Girshick, et al., Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR, 2014

[3] Hosang, et al., What Makes for Effective Detection Proposals?, PAMI, 2015

[4] Girshick, Fast R-CNN, ICCV, 2015

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

[6] Sermant, et al., Overfeat: Integrated recognition, localization and detection using convolutional networks, ICLR, 2014

Resource

- Faster R-CNN
  - https://blog.paperspace.com/faster-r-cnn-explained-object-detection/