

Deep Learning 1, lecture 5

Computer vision tasks

Sadrdinov Ildus, 17.11.23

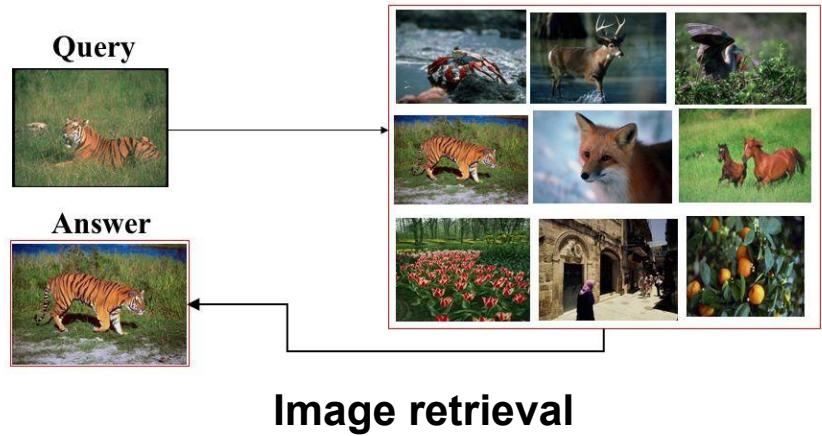
CV tasks



Object detection

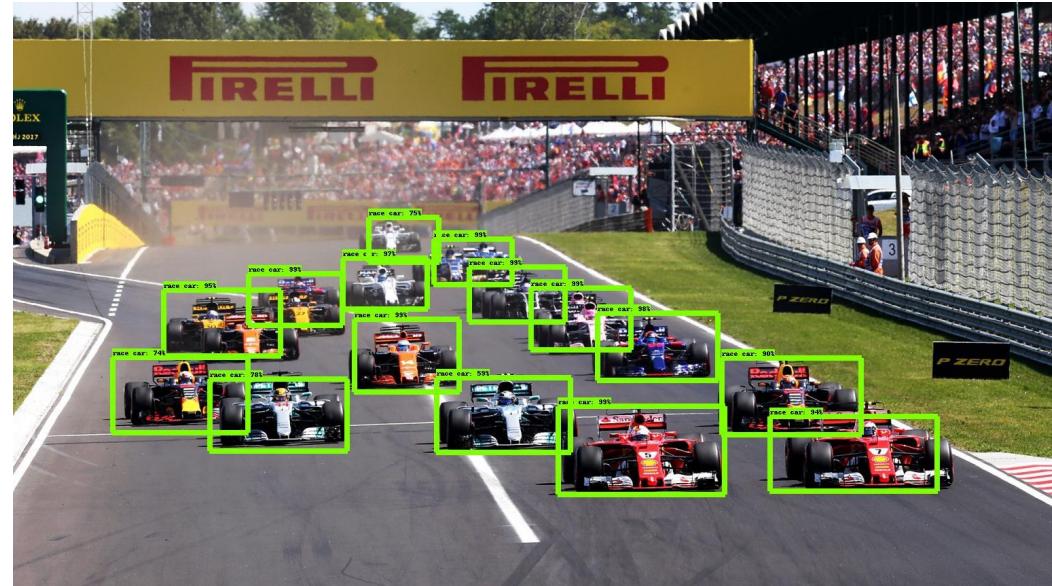


Semantic segmentation



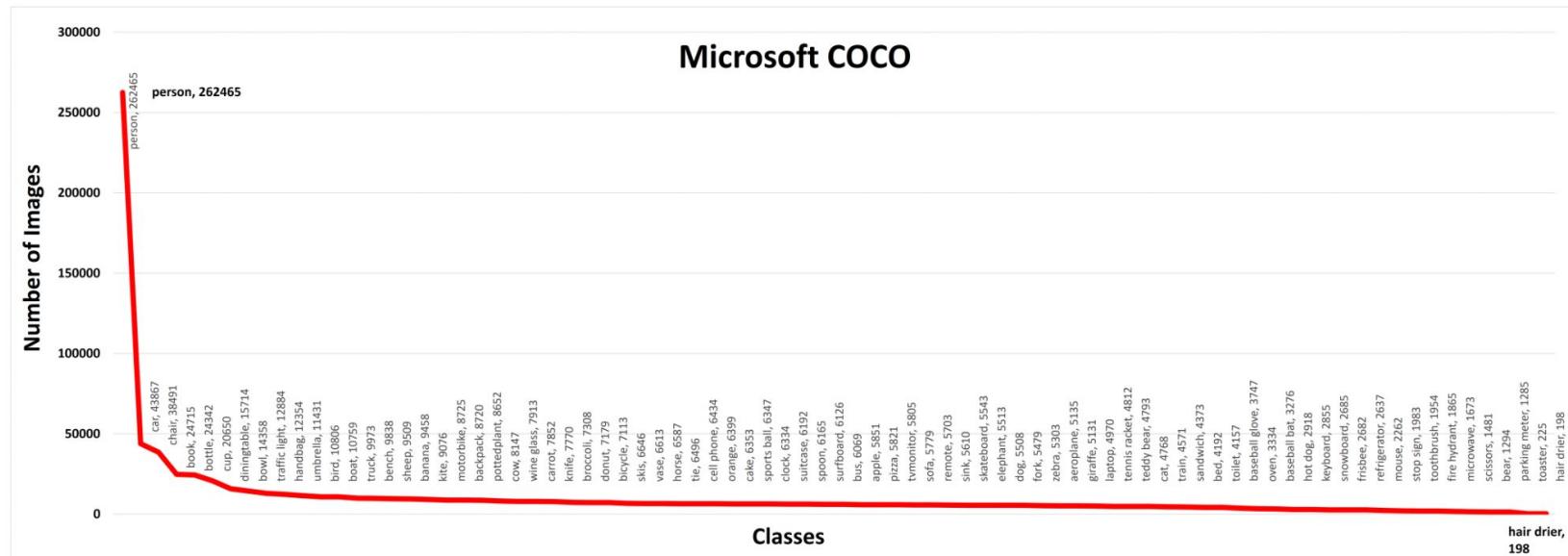
Object detection

- **Goal:** find objects in the image and localize them with **bounding boxes (BB)**
 - Variable number of objects in each image
 - Intra-class variation of objects
 - Imbalanced / rare classes
 - Efficacy (FPS)



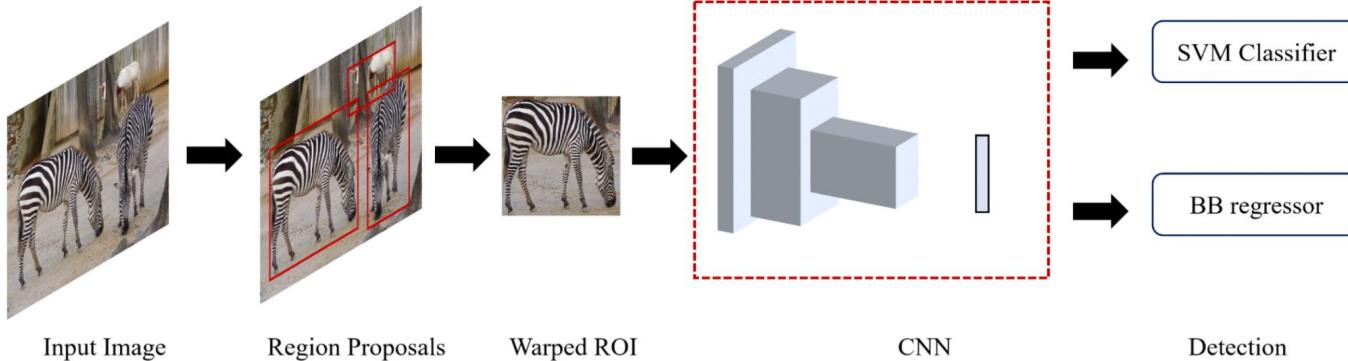
Datasets

- Pascal VOC (**V**isual **O**bject **C**lasses), 20 classes
- MS COCO (**M**icrosoft **C**ommon **O**bjects in **C**ontext), 91 classes
- ImageNet, 200 classes



R-CNN (2013)

- Region-based Convolutional Neural Network
- Two-staged detection
 - Propose regions of interest (by deterministic heuristic, e.g. **Selective Search**)
 - Classify proposed regions
- Pre-trained convolutional backbone
- Model head outputs class prediction and BB position (center, width, height)



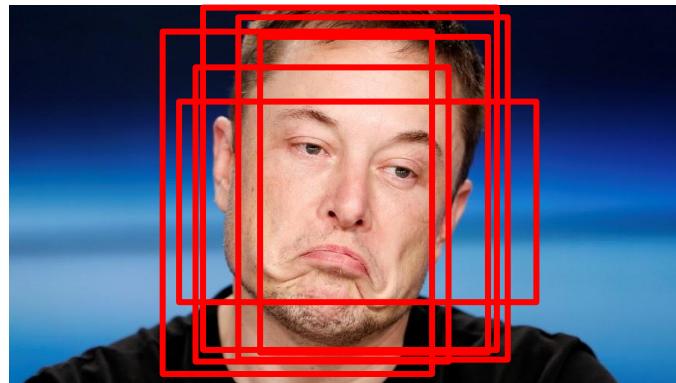
Selective Search



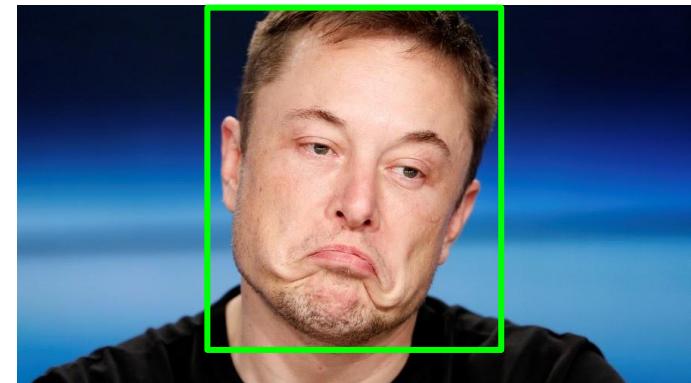
Figure 2: Two examples of our selective search showing the necessity of different scales. On the left we find many objects at different scales. On the right we necessarily find the objects at different scales as the girl is contained by the tv.

Non-maximum suppression (NMS)

- Many overlapping regions of the same object
- Take bounding box with the highest confidence
- Remove all other bounding boxes (with IoU higher than some threshold)



NMS
→



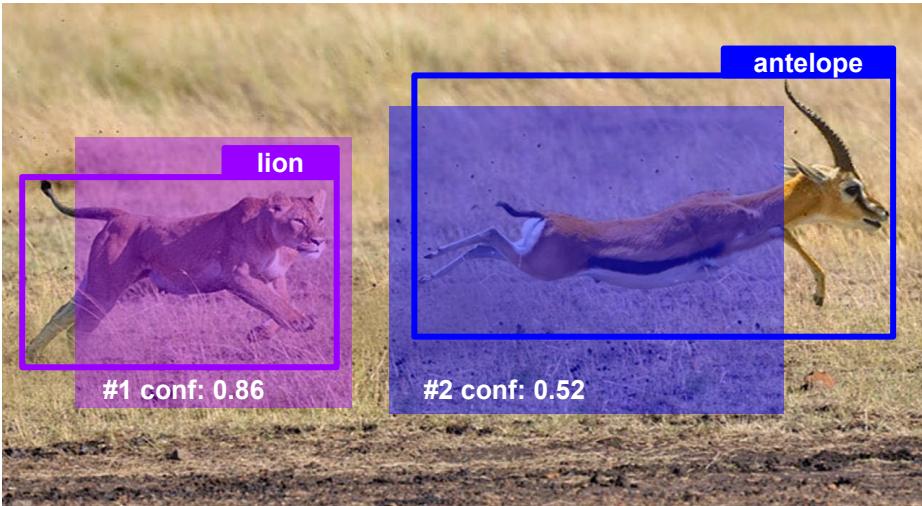
R-CNN problems

- **Extremely slow:** ~47 seconds/image
- Very long training even for small datasets
- Too many proposed regions to check (2000 per image)
- Many regions are overlapping or nonsense
- Deterministic algorithm (i.e. no training) for proposing regions



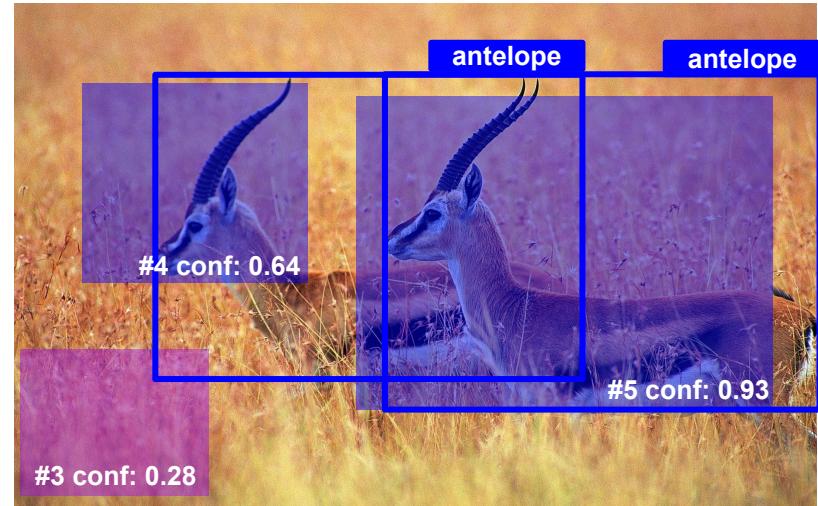
$$\text{IoU} = \frac{\text{area of overlap}}{\text{area of union}}$$

Object detection metrics: mAP



Class: antelope

BB	conf	IoU (thr = 0.5)	TP/FP	precision	recall
#5	0.93	0.74	TP	1	0.33
#4	0.64	0.15	FP	0.5	0.33
#2	0.52	0.54	TP	0.66	0.66

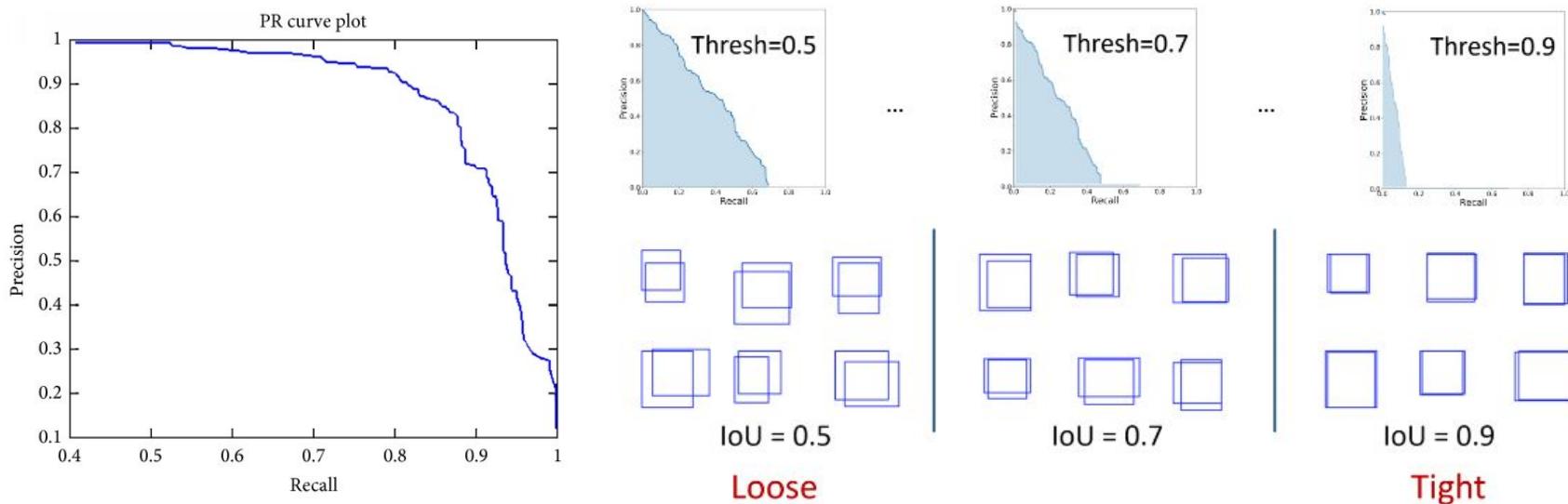


Class: lion

BB	conf	IoU (thr = 0.5)	TP/FP	precision	recall
#1	0.86	0.68	TP	1	1
#3	0.28	0.0	FP	0.5	1

Object detection metrics: mAP

- **AP (average precision)** = area under PR-curve
- **mAP (mean AP)** = AP averaged over all classes
- Different variants depending on IoU threshold: mAP_{50} , mAP_{75} , $\text{mAP}_{[50:95]}$



mAP calculation algorithm

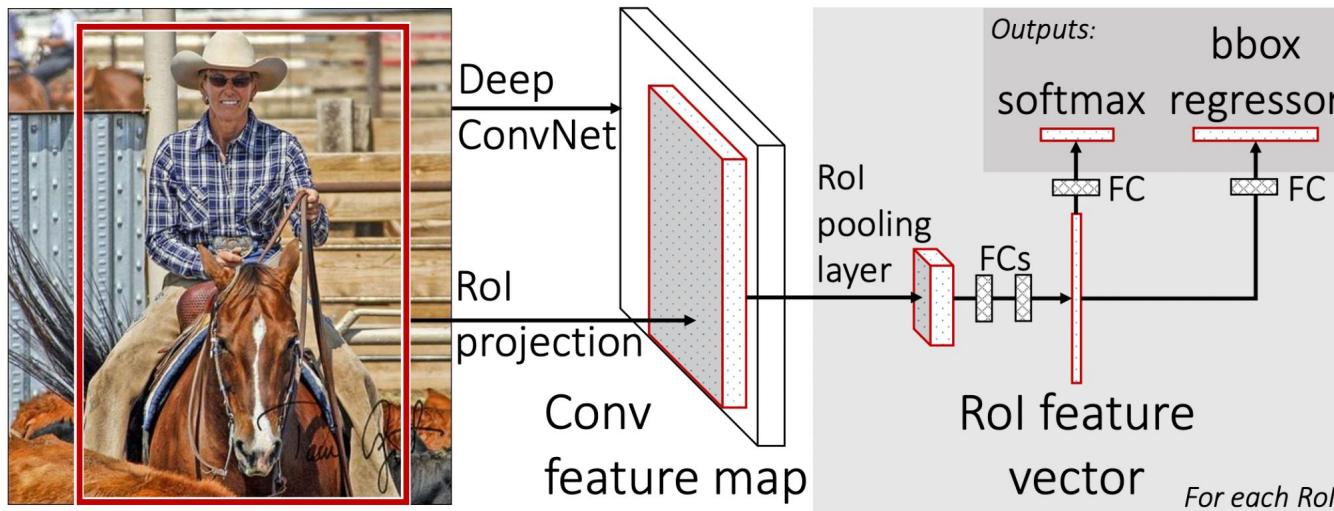
For each class:

1. Determine whether each bounding box is TP or FP based on **IoU thresholding**
2. Sort bounding boxes by **model confidence**
3. Calculate binary PR-AUC (i.e., AP – average precision)

Then do macro averaging over all classes

Fast R-CNN (2015)

- Pass **whole image** through the convolutional feature extractor
- Select sub-feature maps corresponding to each **Region of Interest (RoI)**
- Pass sub-feature maps to **RoI pooling**, get **fixed size vectors**
- **146x times faster** than R-CNN



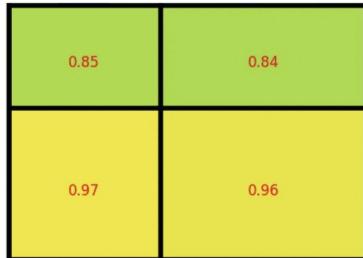
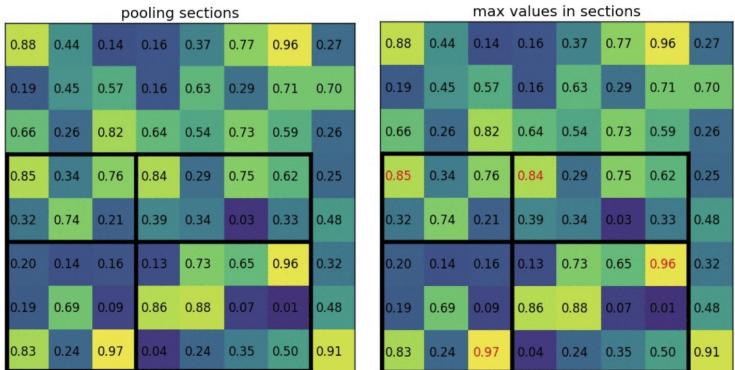
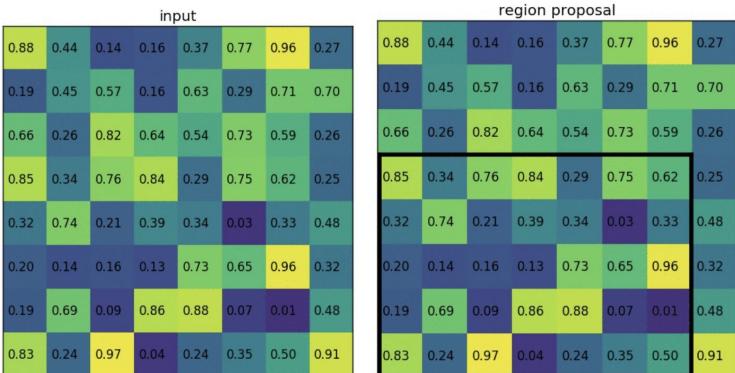
Girshick, 2015

Region of Interest (RoI) pooling

- **Input:** feature map, N region proposals
- **Output:** N fixed size maps

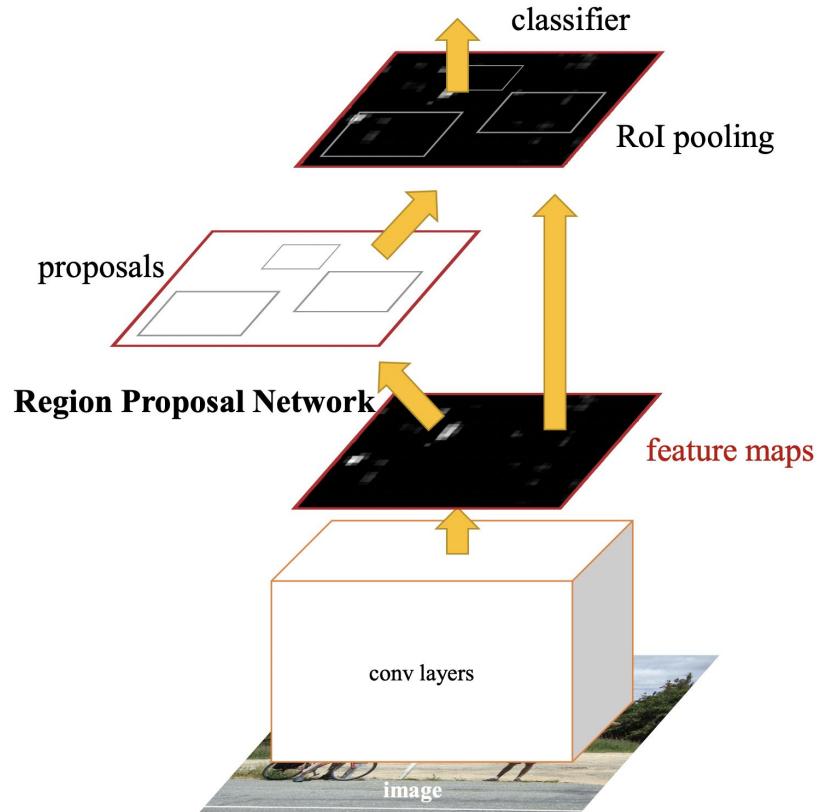
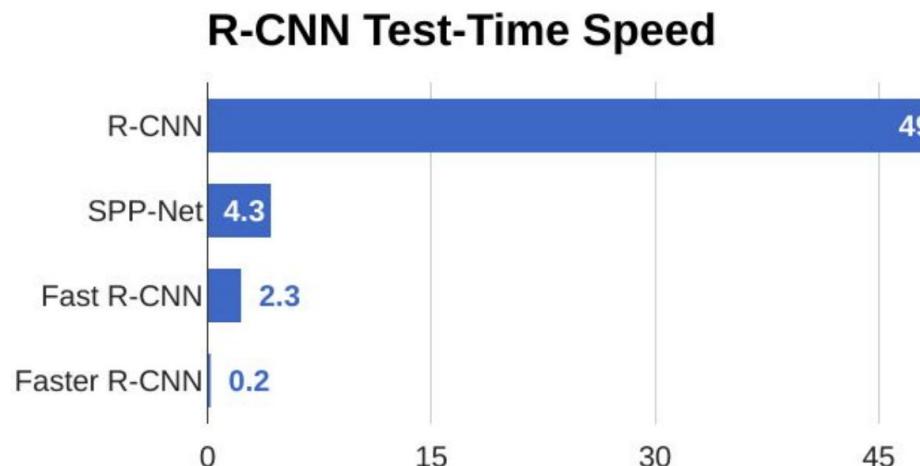
- Select region proposal
- Divide region by sections
- Number of sections equals output size
(example: 2x2, real model: 7x7)
- Apply max pooling to each section

- Significant speed up of training and inference



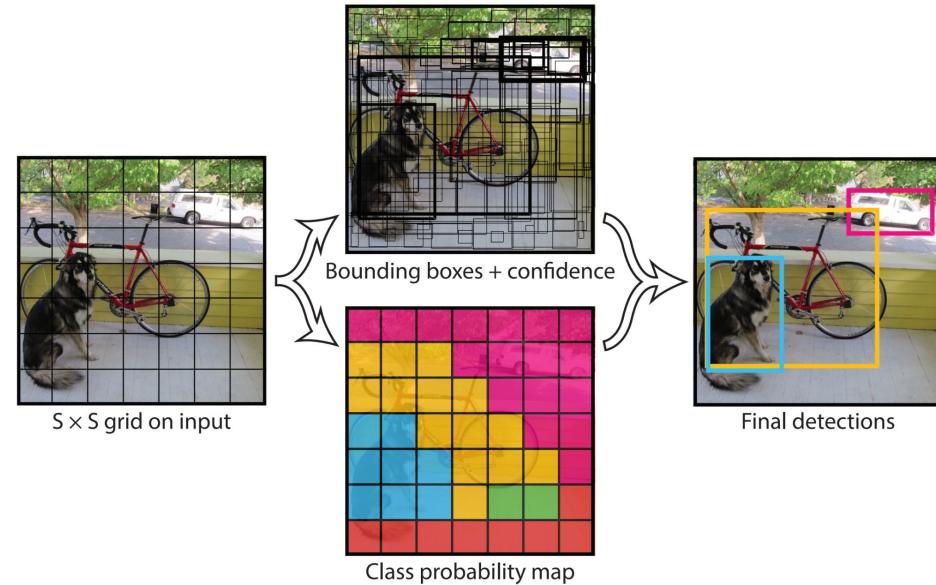
Faster R-CNN (2015)

- Replace **Selective Search** with **Region Proposal Network**



YOLO (2015)

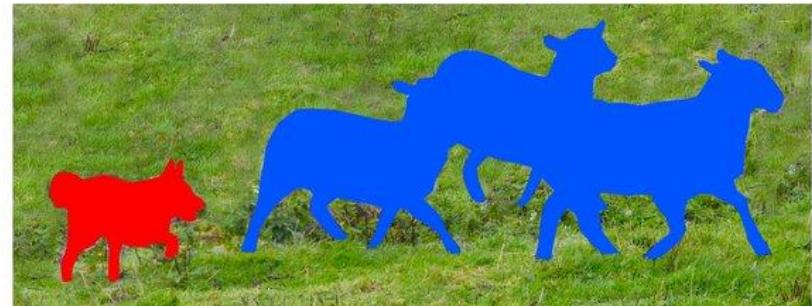
- You Only Look Once
- Give up two-staged detection
- Divide image into **SxS** grid cells
- Predict **B** bounding boxes for each cell: $(x_c, y_c, h, w, \text{conf})$
- Also predict class probabilities for each cell
- **45 FPS** (Faster R-CNN: **5 FPS**)
- 2023: YOLOv8



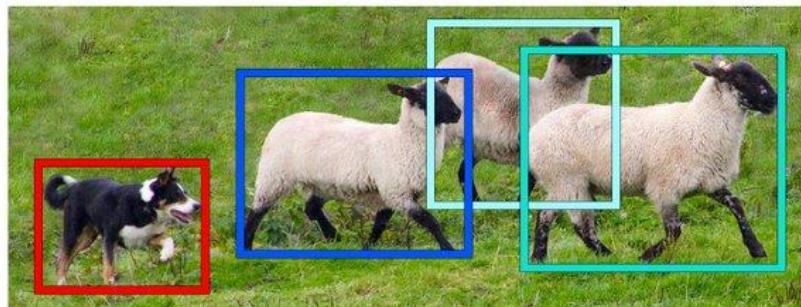
Semantic vs. Instance segmentation



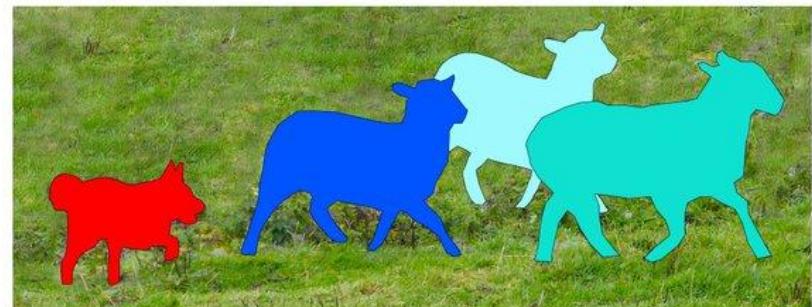
Image Recognition



Semantic Segmentation



Object Detection



Instance Segmentation

Datasets

- Pascal VOC (**V**isual **O**bject **C**lasses), 20 classes
- MS COCO (**M**icrosoft **C**ommon **O**bjects in **C**ontext), 91 classes
- Cityscapes, 30 classes

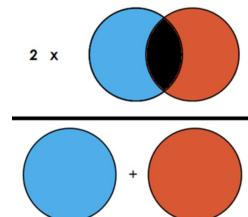
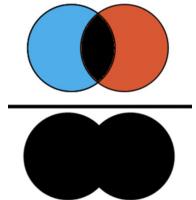


Segmentation metrics

- Per-pixel accuracy
- Mean per-class accuracy

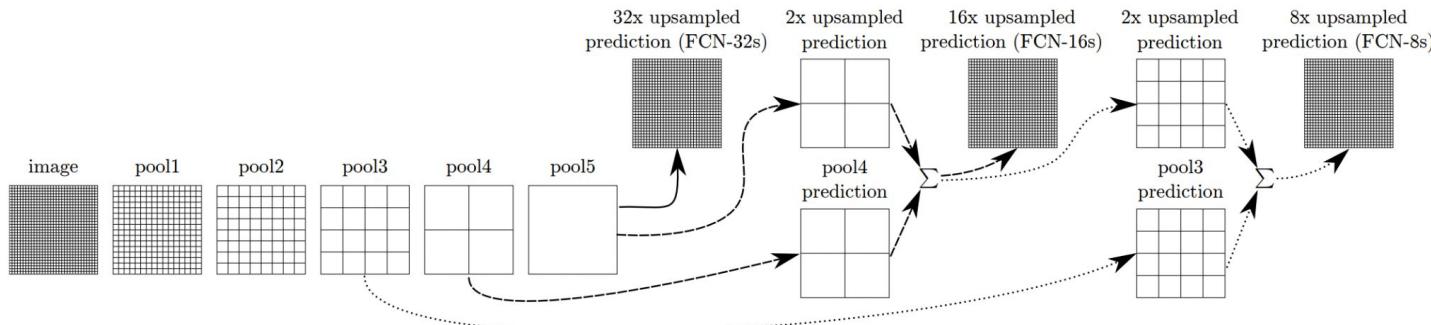
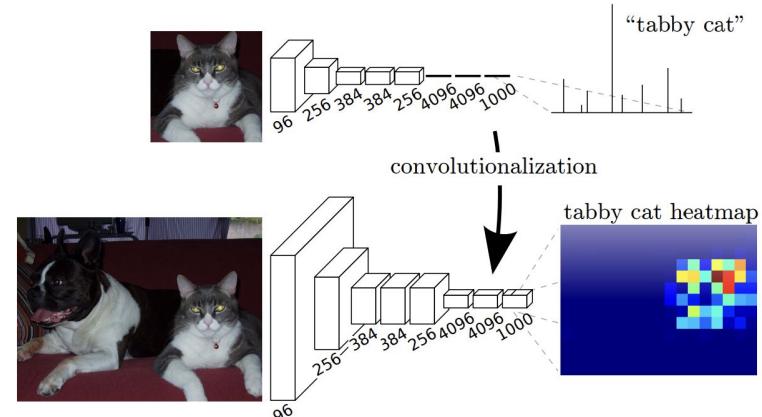


- Mean per-class IoU: $\frac{\text{Intersection}}{\text{Union}}$
- Mean per-class Dice coefficient: $\frac{2 \times \text{Intersection}}{\text{Sum of Squares}}$



Fully-convolutional Networks (FCN, 2014)

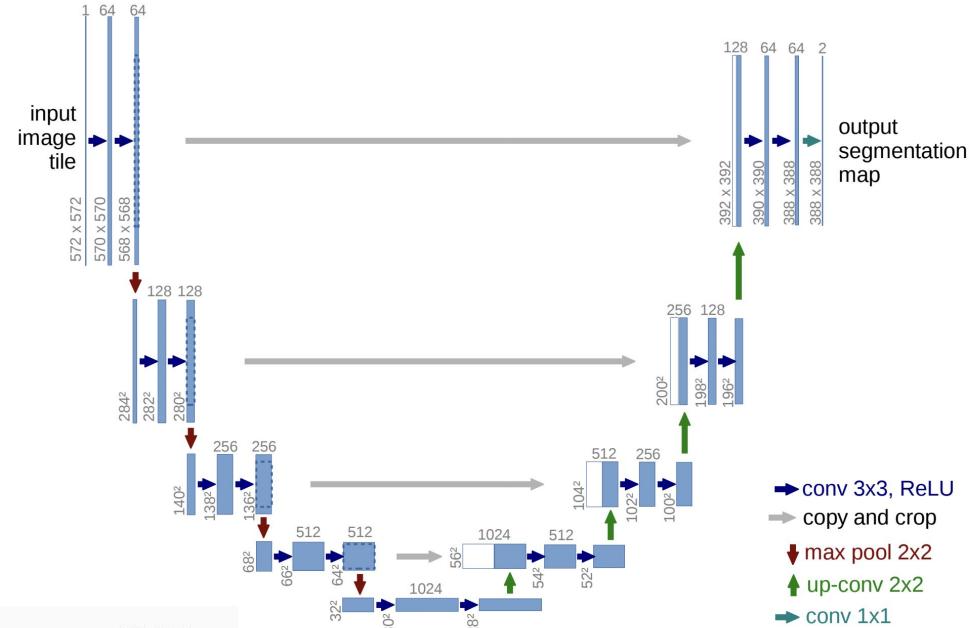
- Use CNNs to create dense feature map
- Combine feature maps of different resolutions



Long et al., 2014

U-Net architecture

- **Encoder-decoder approach**
- Often used for **image2image** tasks (not necessarily semantic segmentation)
- Uses **Up-Convolution**s (a.k.a Transposed Convolutions)



Transposed convolution

Input	Kernel	Output
$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 4 & 6 \\ 4 & 12 & 9 \end{bmatrix}$
$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 4 & 6 \\ 4 & 12 & 9 \end{bmatrix}$
$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 4 & 6 \\ 4 & 12 & 9 \end{bmatrix}$

Diagram illustrating the computation of a transposed convolution output. The input is a 2x2 matrix with values [0, 1; 2, 3]. The kernel is a 2x2 matrix with values [0, 1; 2, 3]. The output is a 3x3 matrix with values [0, 0, 1; 0, 4, 6; 4, 12, 9]. The calculation shows how the kernel is applied to overlapping receptive fields in the input to produce the output values.

Ronneberger et al., 2015

Instance segmentation: Mask R-CNN (2017)

- **Main idea:** add dense prediction head

$$L = L_{cls} + L_{box} + L_{mask}$$

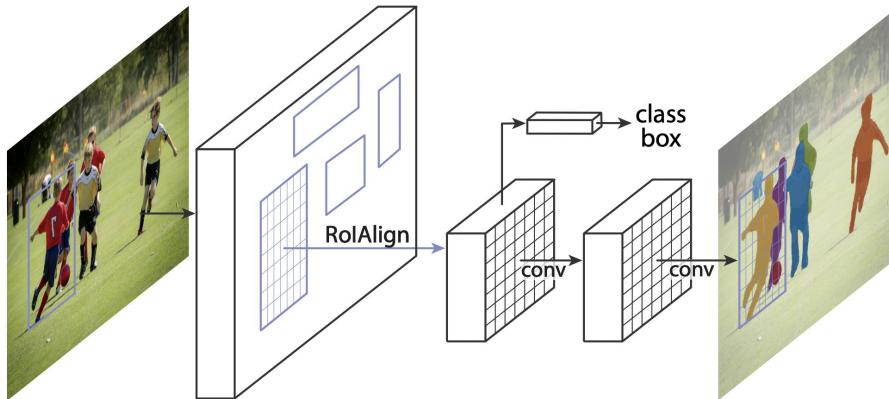


Image retrieval

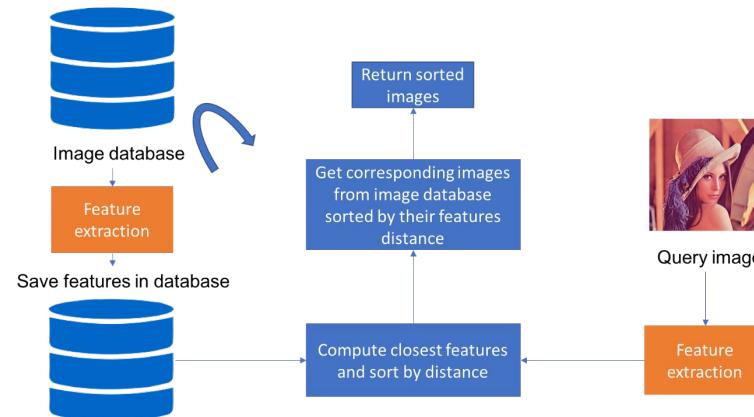


?



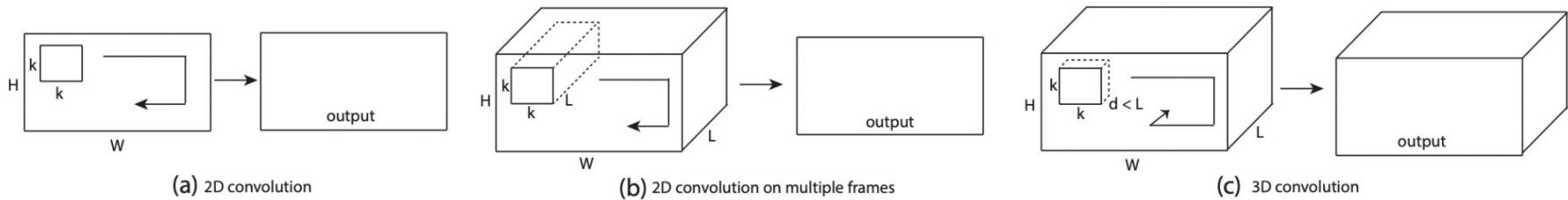
Image retrieval

- Generate low-dimensional **embeddings** for images
- Search for images with the closest embeddings
(by **L2 distance** or **cosine similarity**)
- Possible to use approximate search algorithms
- Typical source of embeddings – a pre-trained convolutional network



Approaches to video processing

- Applying CNNs to separate frames
- 3D convolutions
- Two-stream architectures



[Tran et al. 2014](#)

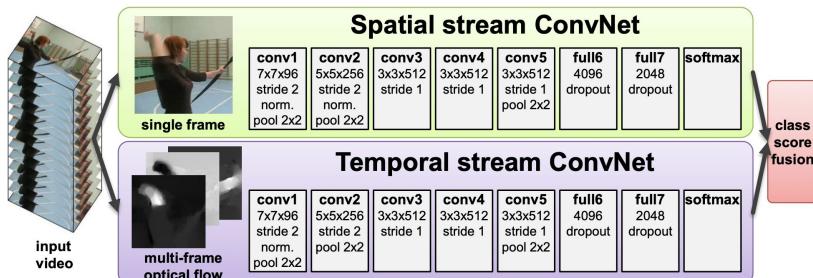


Figure 1: Two-stream architecture for video classification.

[Simonyan and Zisserman, 2014](#)

Conclusion

- Re-use pre-trained weights as much as possible
- Use task-specific prediction heads
- Multi-component losses are frequently used
- Careful metrics selection is important
- For dense tasks (i.e. segmentation) we often need features of different resolution