Object detection

Deep learning - BMDC 2025-2026

Gheorghe Cosmin Silaghi

Universitatea Babeș-Bolyai

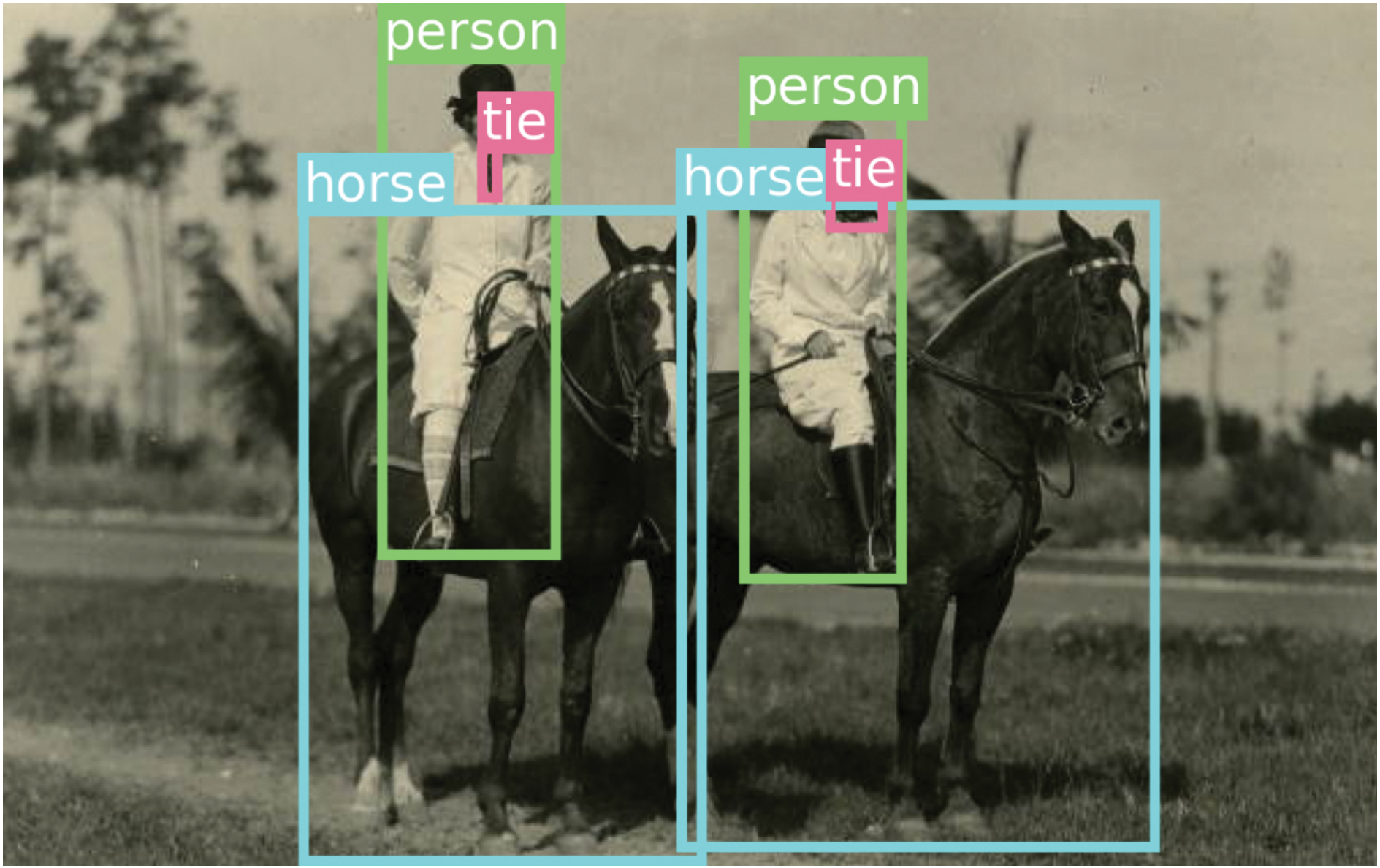
October 17, 2025

Definition

- draw bounding boxes around objects of interest in a picture. Let us to determine both which objects are in the picture and where they are located

Common applications

- counting: find out how many instances of an object are in the image
- tracking: track how objects move in scene over time, by performing object detection on every frame of a moovie
- cropping: identifying the area of an image that contains an object of interest to crop it and send a higher-resolution version of the image patch to an OCR or a classifier


person
tie
horse
person
horse
tie

Segmentation for object detection

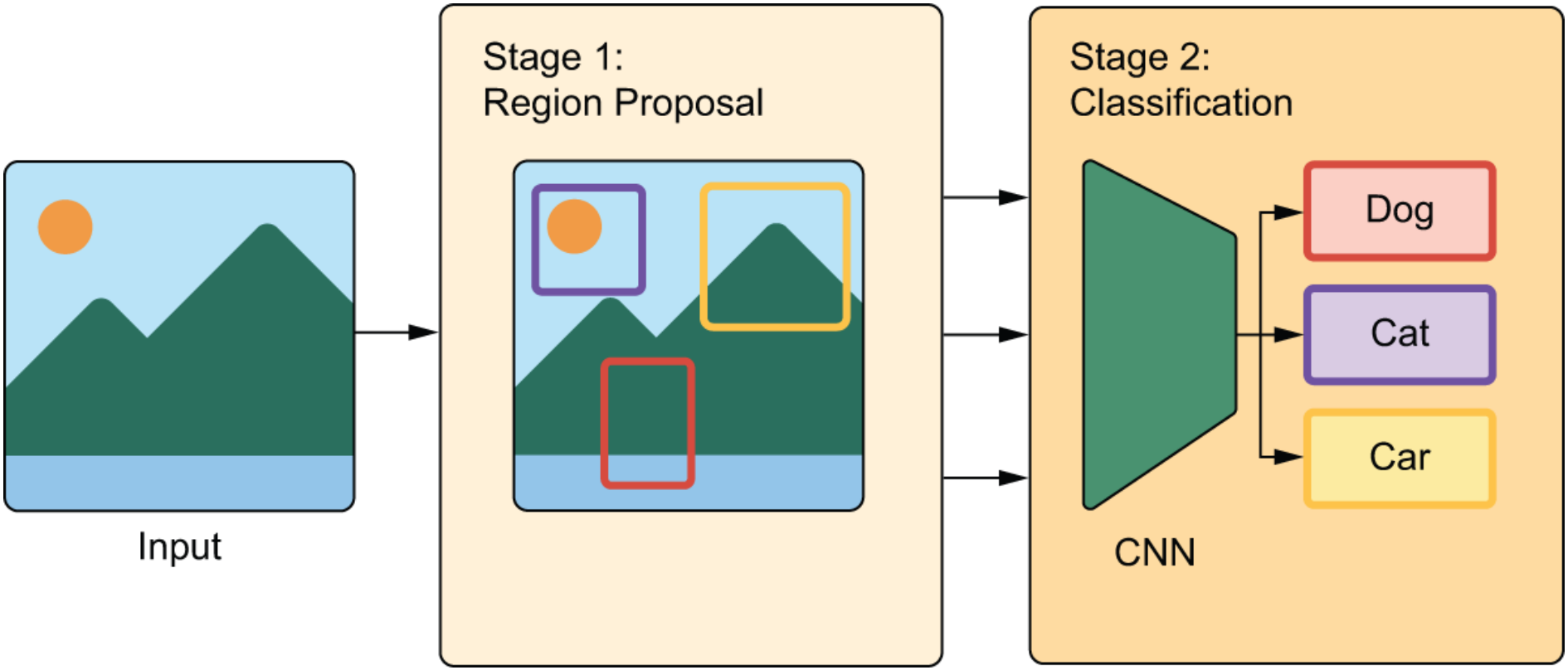
- segmentation could be used for object detection. In returns all information needed, plus more
- This increase wealth of information comes with a significant computational cost: a good object detection model should typically run faster than an image segmentation
- image segmentation has also a significant data labeling cost

Types of object detectors

- two-stage detectors: first extract region proposals. they are called Region-based CNNs
- single-stage detectors: RetinaNet, or YOLO (You Only Look Once) models

Two stage detectors

- the first stage takes and image a produce a few thousands partially overlapping bounding boxes (*region proposals*) around areas that look object-like
- second step: a ConvNet looks at each region proposal and classifies it into a number of predetermined classes
- region proposals with low score across all classes are discarded. we get a much smaller number of boxes, each with a high class presence score
- at the end, bounding boxes around each object are further refined to eliminate duplicates and make each bounding box as precise as possible


Stage 1:
Region Proposal
Stage 2:
Classification
Dog
Cat
Car
Input
CNN

Box generation in a R-CNN

- Selective Search heuristics was used in the first versions of R-CNN. it used some definition of spatial consistency to identify object-like areas
- in later models (like the Faster-R-CNN) box generation became a deep learning model

Performance considerations for R-CNNs

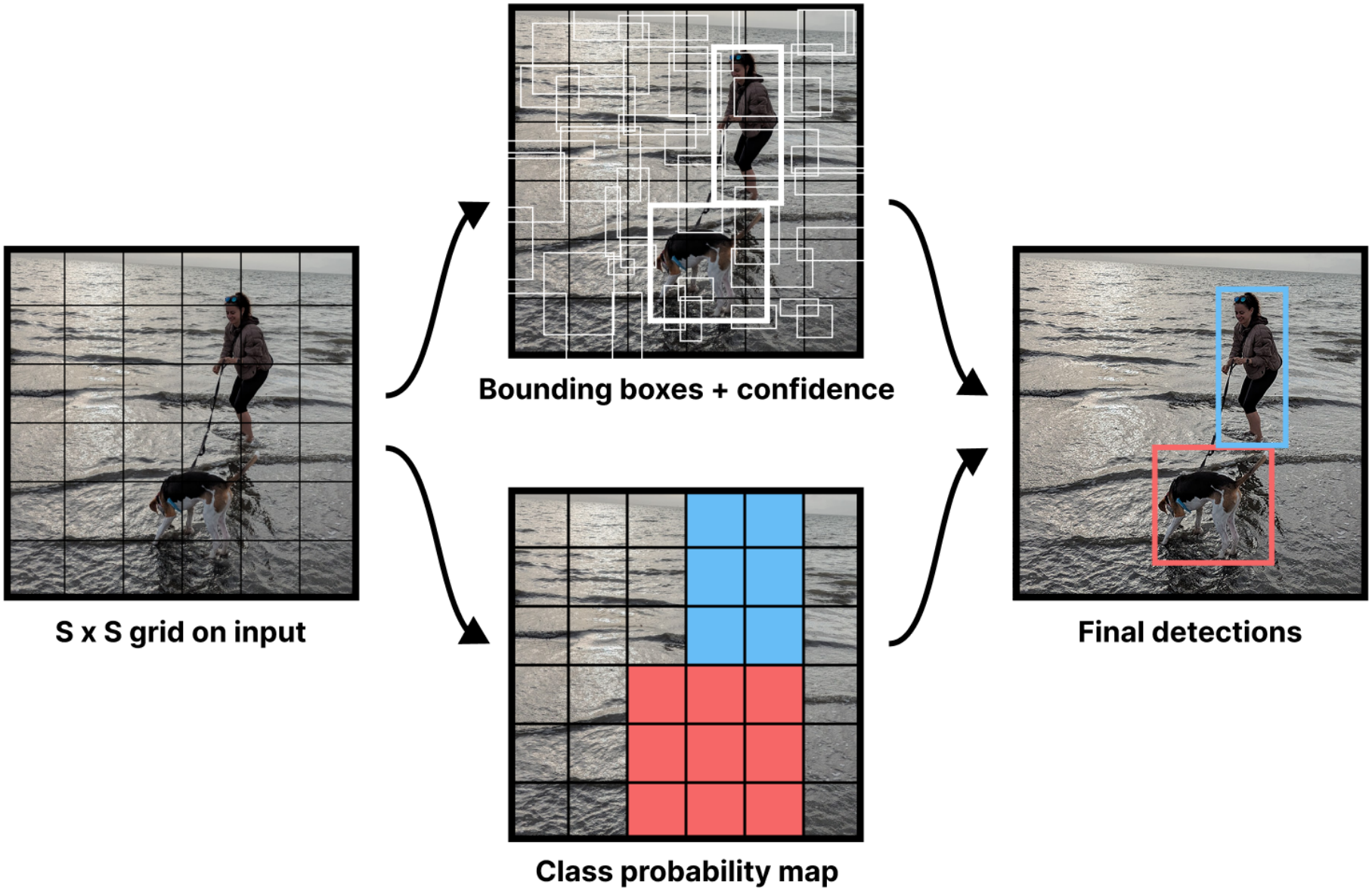
- it works very well, but it is quite computationally expensive - you need to classify tousands of patches
- in a practical application if you are doing a server-side inference having a good GPU available, you would be better of using a segmentation model instead (like SAM).
- if you are resource-constrained, you are going to use a more computationally efficient object detection architecture

Single stage detectors

- main families are: RetinaNet, Single Shot MultiBox Detectors, and YOLO
- they boast significantly faster speed and greater efficiency
- YOLO si the most popular when it comes to real-time applications
- YOLO (originated in 2015) is just a lot of code for manipulating bounded boxes and predicted output

YOLO

- YOLO will propose bounded boxes and object labels in one step
- the model will divide the image into a grid and predict two separate outputs at each grid location - a bounding box and a class label.
- most images will not have objects evenly distributed across the grid, therefore, the model will compute a confidence score along each box
- the confidence score should be high when an object is detected in the location and zero when no object inside


Bounding boxes + confidence
S x S grid on input
Class probability map
Final detections

YOLO architecture

- it uses a ConvNet backbone to obtain interesting high-level features of an input image. We will use KerasHub to load a pre-trained backbone (ResNet)
- in comparison with Xception, ResNet uses strided convolutions to downsample - i.e. it works better when we care about the spatial location of the input
- next, we turn the backbone in a detection model by addind layers for outputing box and class predictions: two densely connected layers with an activation in the middle
- split the output: first 5 numbers used for bounded box prediction (four for the box and one for confidence). rest output will be the class probability map - a classification prediction for each grid location over all possible 91 labels

Using a pretrained RetinaNet detector

- operates on the same basic principles as YOLO
- in the YOLO model we took the final outputs of the ConvNet and used them to build our object detector. These output features map to large areas on the input image: they are not very effective in fining small objects in the scene
- to solve this scale issue would be to directly use the output of earlier layers in the ConvNet
- but the output of these earlier layers is not very semantically interesting: they get edges and curves later used by the model to aggregate the larger concepts
- RetinaNet uses a feature pyramid network: final features of the ConvNet base are upsampled with progressive **Conv2DTranspose** layers
- RetinaNet includes lateral connections: which sum these upsampled feature maps with the feature maps of the same size from the original Conv2D: combines the semantically interesting low-resolution features at the end of the ConvNet with high-resolution small-scale featrues from the beginning of the ConvNet