Recognition: Past, present, future?

Benozzo Gozzoli, Journey of the Magi, c. 1459

Classification vs detection

Classification:

- there is an X in this image
  - what

Detection:

- there is an X HERE in this image
  - what AND where

Key issues

- how to specify where
- relationship between what and where
  - efficiency, etc
- evaluation
  - surprisingly fiddly

Two threads

Localize then classify

- find boxes that likely contain objects
- decide what is in the box

YOLO: Localize while classifying

- in parallel, score
  - boxes for “goodness of box”
  - boxes for “what is in it”
- combine

Start simple

Where = axis aligned box

- **Decide on a window shape:** this is easy. There are two possibilities: a box, or something else. Boxes are easy to represent, and are used for almost all practical detectors. The alternative — some form of mask that cuts the object out of the image — is hardly ever used, because it is hard to represent.
- **Build a classifier for windows:** this is easy – we've seen multiple constructions for image classifiers.
- **Decide which windows to look at:** this turns out to be an interesting problem. Searching all windows isn't efficient.
- **Choose which windows with high classifier scores to report:** this is interesting, too, because windows will overlap, and we don't want to report the same object multiple times in slightly different windows.
- **Report the precise locations of all faces using these windows:** this is also interesting. It turns out our window is likely not the best available, and we can improve it after deciding it contains a face.

Which window

Astonishing fact

- Easy to tell whether a region is likely to be an object
  – even if you don't know what object (Endres+Hoiem, 10; Uijlings et al 12)
  – if it's an object
- there's contrast with surroundings in texture, etc
  – if not
- often neighbor region is similar

Selective Search

Construct hierarchy of image regions

- using a hierarchical segmenter

Rank regions using a learned score

Make boxes out of high-ranking regions

Selective search pipeline

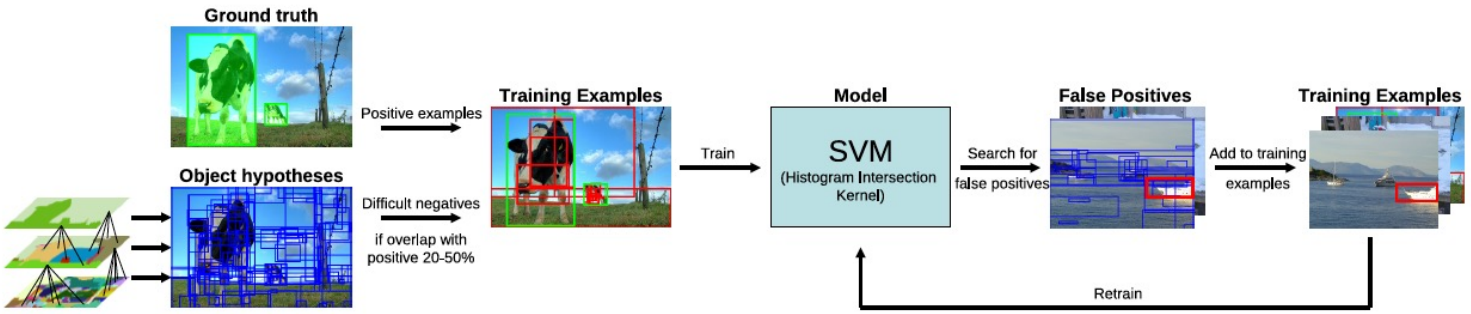


Figure 3: The training procedure of our object recognition pipeline. As positive learning examples we use the ground truth. As negatives we use examples that have a 20-50% overlap with the positive examples. We iteratively add hard negatives using a retraining phase.

Uijlings et al, 12

You need to search at multiple scales

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.

Simplest detector

Use selective search to propose boxes

Check boxes with classifier

BUT

- boxes likely overlap - non-maximum suppression
- boxes likely in poor location - bounding box regression

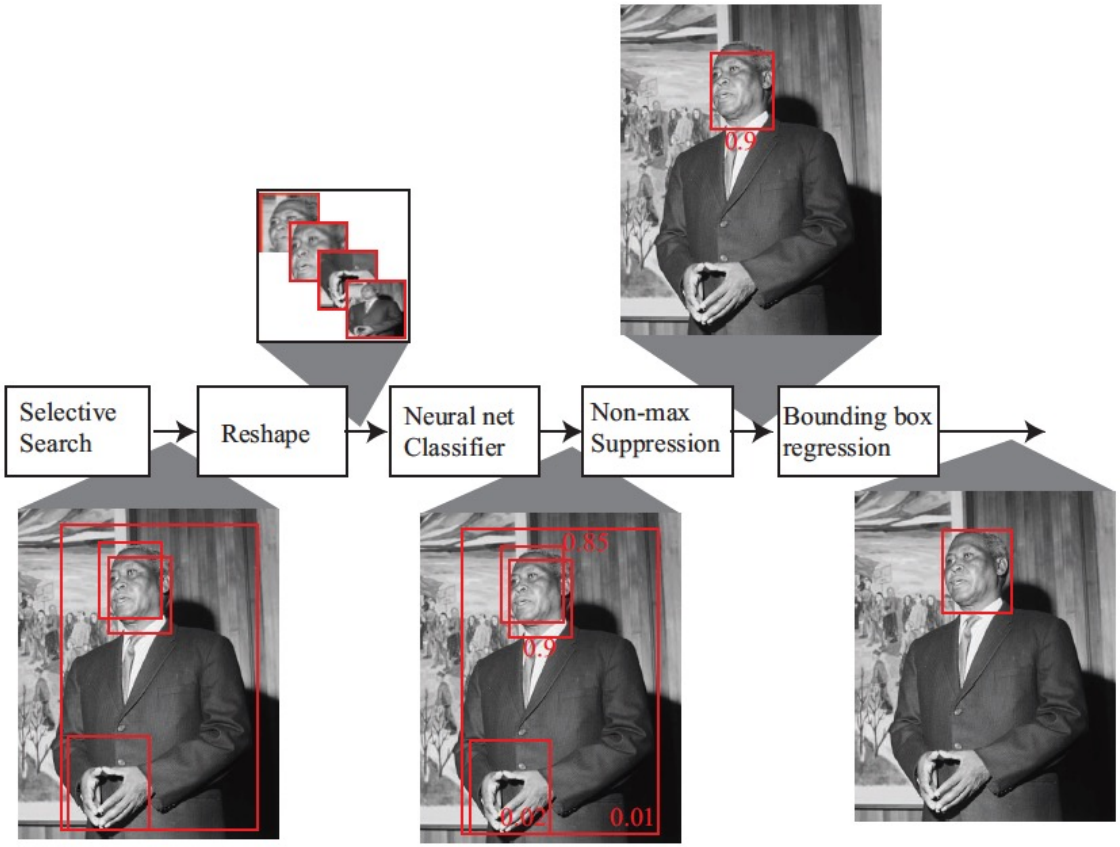


FIGURE 18.6: *A schematic picture of how R-CNN works. A picture of Inkosi Albert Luthuli is fed in to selective search, which proposes possible boxes; these are cut out of the image, and reshaped to fixed size; the boxes are classified (scores next to each box); non-maximum suppression finds high scoring boxes and suppresses nearby high scoring boxes (so his face isn't found twice); and finally bounding box regression adjusts the corners of the box to get the best fit using the features inside the box.*

Non maximum suppression

Deciding which windows to report presents minor but important problems. Assume you look at 32×32 windows with a stride of 1. Then there will be many windows that overlap the object fairly tightly, and these should have quite similar scores. Just thresholding the value of the score will mean that we report many instances of the same object in about the same place, which is unhelpful. If the stride is large, no window may properly overlap the object and it might be missed. Instead, most methods adopt variants of a greedy algorithm usually called **non-maximum suppression**. First, build a sorted list of all windows whose score is over threshold. Now repeat until the list is empty: choose the window with highest score, and accept it as containing an object; now remove all windows with large enough overlap on the object window.

Bounding box regression

Deciding precisely where the object is also presents minor but important problems. Assume we have a window that has a high score, and has passed through non-maximum suppression. The procedure that generated the window does not do a detailed assessment of all pixels in the window (otherwise we wouldn't have needed the classifier), so this window likely does not represent the best localization of the object. A better estimate can be obtained by predicting a new bounding box using a feature representation for the pixels in the current box. It's natural to use the feature representation computed by the classifier for this **bounding box regression** step.

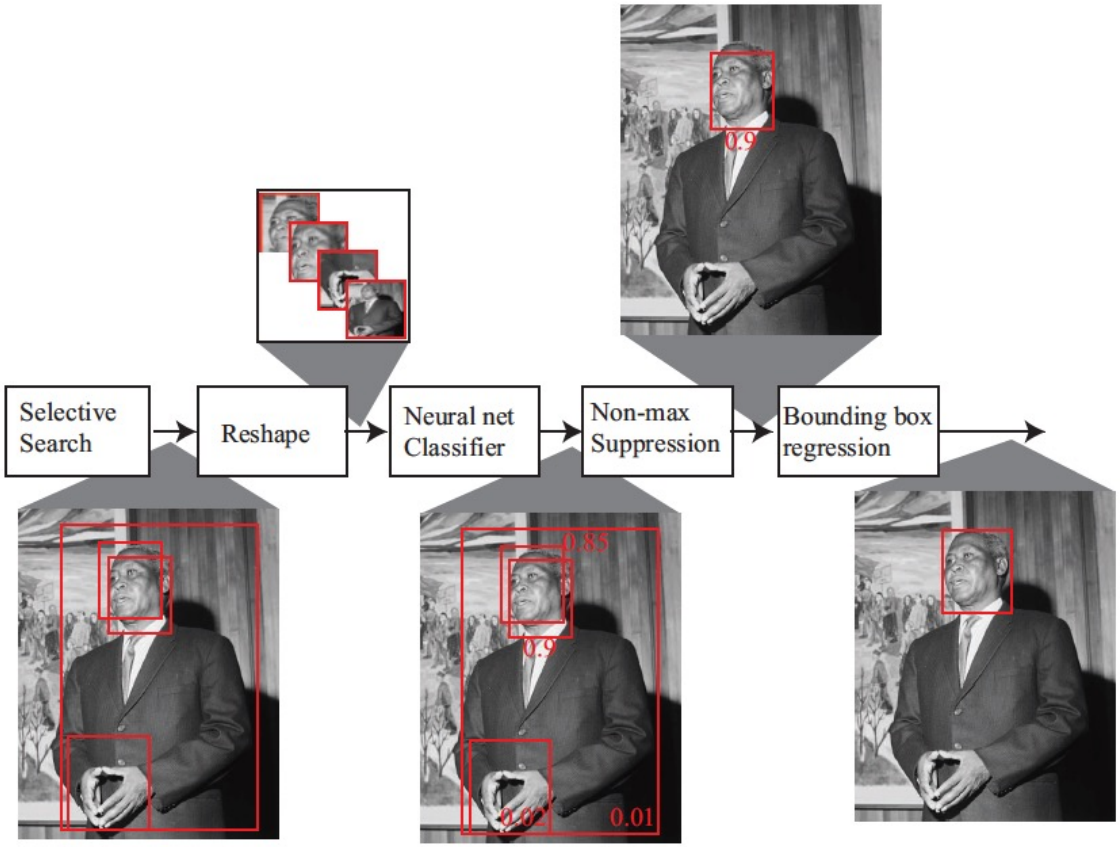


FIGURE 18.6: *A schematic picture of how R-CNN works. A picture of Inkosi Albert Luthuli is fed in to selective search, which proposes possible boxes; these are cut out of the image, and reshaped to fixed size; the boxes are classified (scores next to each box); non-maximum suppression finds high scoring boxes and suppresses nearby high scoring boxes (so his face isn't found twice); and finally bounding box regression adjusts the corners of the box to get the best fit using the features inside the box.*

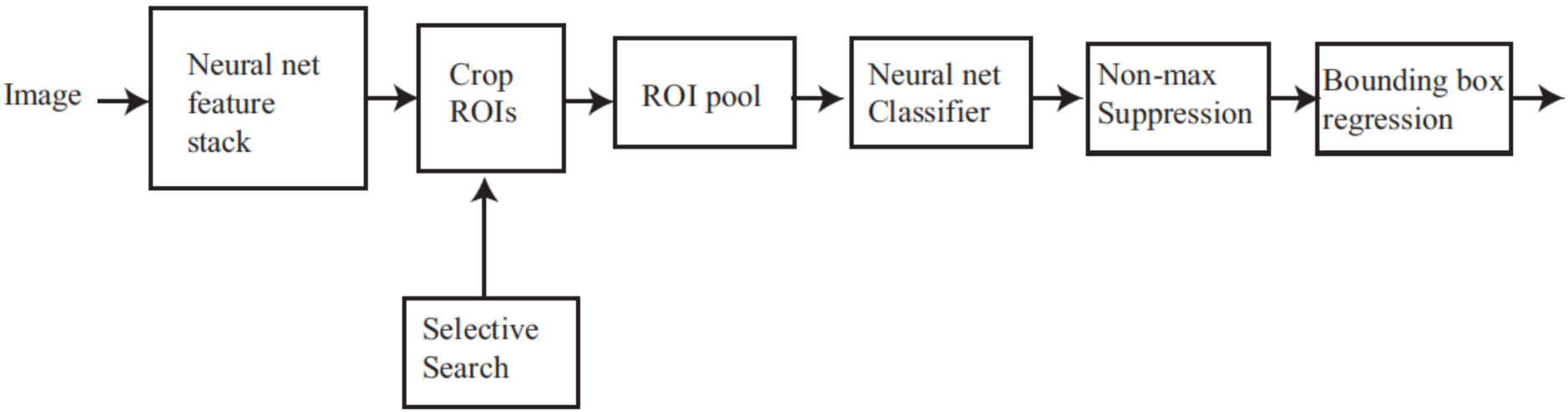


FIGURE 18.7: *Fast R-CNN is much more efficient than R-CNN, because it computes a single feature map from the image, then uses the boxes proposed by selective search to cut regions of interest (ROI's) from it. These are mapped to a standard size by a ROI pooling layer, then presented to a classifier. The rest should be familiar.*

Configuration spaces

You should think of a box as a point in a 4D space

- configuration space of the boxes

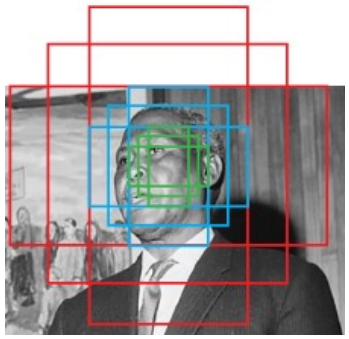
Selective search is weird

- networks don't do lists much

Alternative

- sample the configuration space on some form of grid
  - eg three aspect ratios, three scales, grid of locations
  - important: many possible sampling schemes
- check each sample with rank score

Anchor boxes



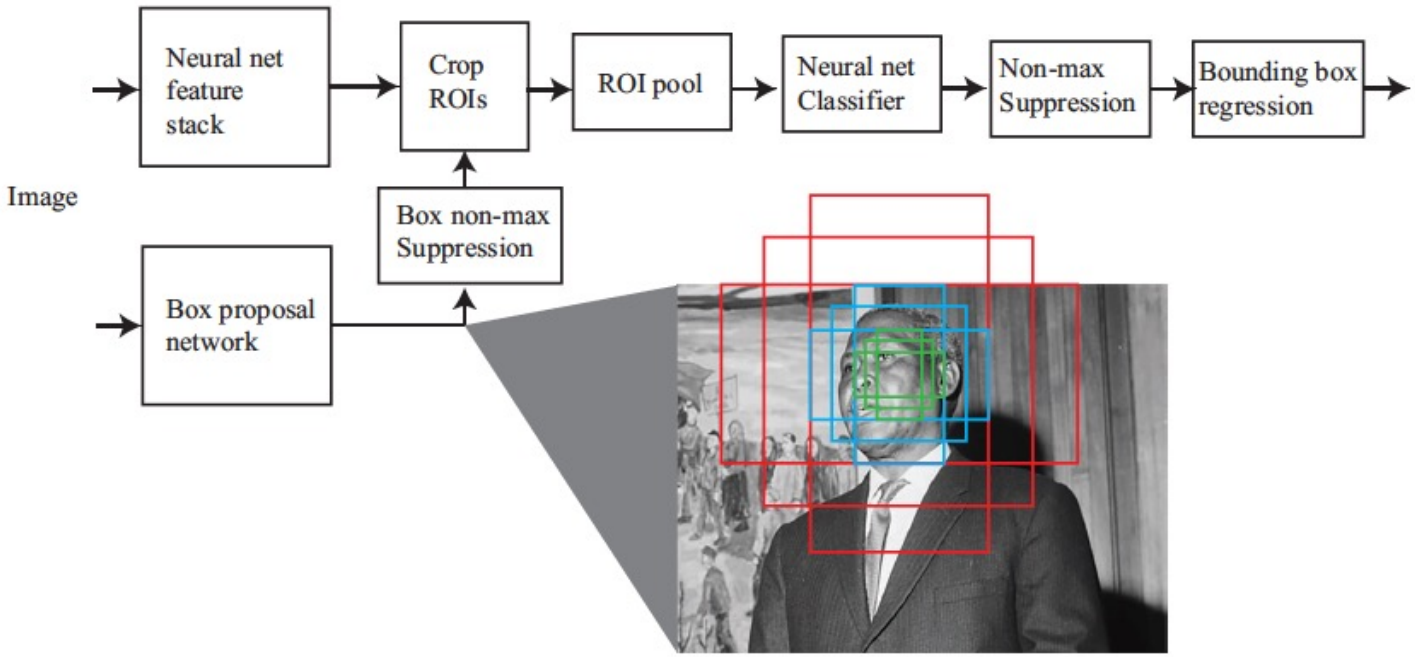


FIGURE 18.8: *Faster RCNN uses two networks. One uses the image to compute "objectness" scores for a sampling of possible image boxes. The samples (called "anchor boxes") are each centered at a grid point. At each grid point, there are nine boxes (three scales, three aspect ratios). The second is a feature stack that computes a representation of the image suitable for classification. The boxes with highest objectness score are then cut from the feature map, standardized with ROI pooling, then passed to a classifier. Bounding box regression means that the relatively coarse sampling of locations, scales and aspect ratios does not weaken accuracy.*

Evaluating detectors

Compare detected boxes w ground truth boxes

Favor

- right number of boxes with right label in right place

Penalize

- awful lot of boxes
- multiple detections of the same thing

Strategy

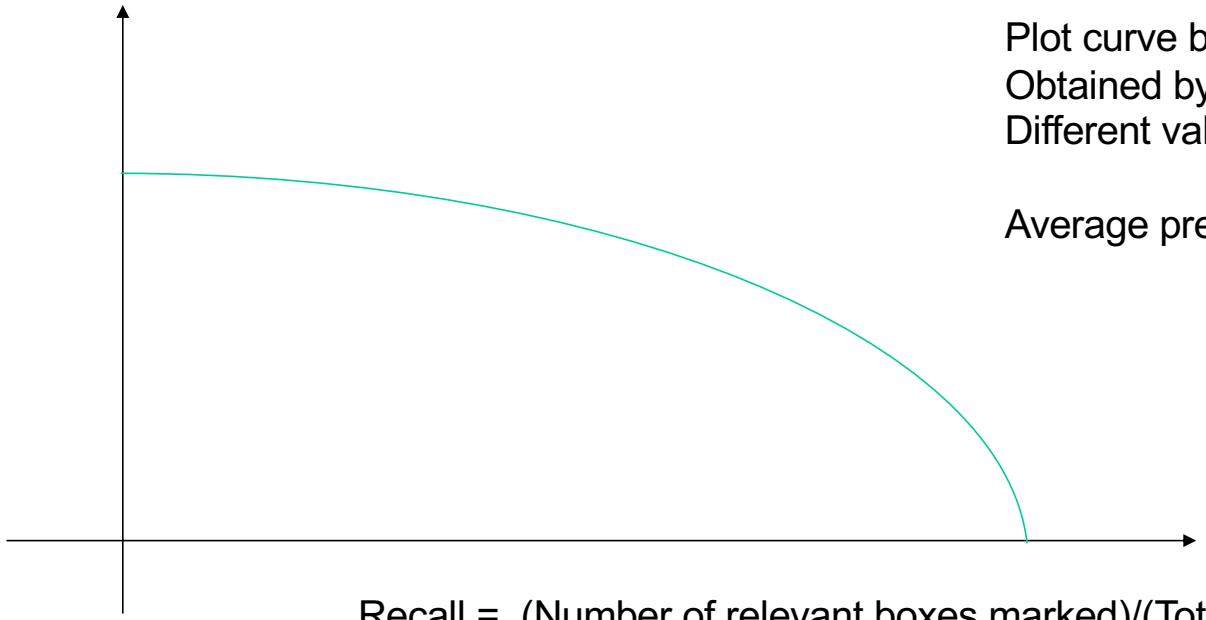
Strategy:

- Detector makes a ranked list of boxes
- GT is a list of boxes
- Mark detector boxes with relevant/irrelevant
- summarize lists

Marking boxes:

- All are irrelevant, then
- For each GT box:
  - Overlap measured as IOU (intersection over union)
  - Find highest ranking box with largest overlap
    - mark relevant if IOU> threshold

Average precision



Plot curve by computing recall, precision
Obtained by taking top k boxes in list for
Different values of k

Average precision is area under curve

Recall = (Number of relevant boxes marked)/(Total number of relevant boxes)

Precision =
(number of relevant boxes marked)/(total number of boxes marked

Strategy

Strategy:

- Detector makes a ranked list of boxes
- GT is a list of boxes
- Mark detector boxes with relevant/irrelevant
- summarize lists

Summarize lists:

- Sort by box ranking
- Compute AP per class
- Compute average of AP

- MAP at IOU 0.5 has been standard for a while
- Higher IOU's are harder.

YOLO

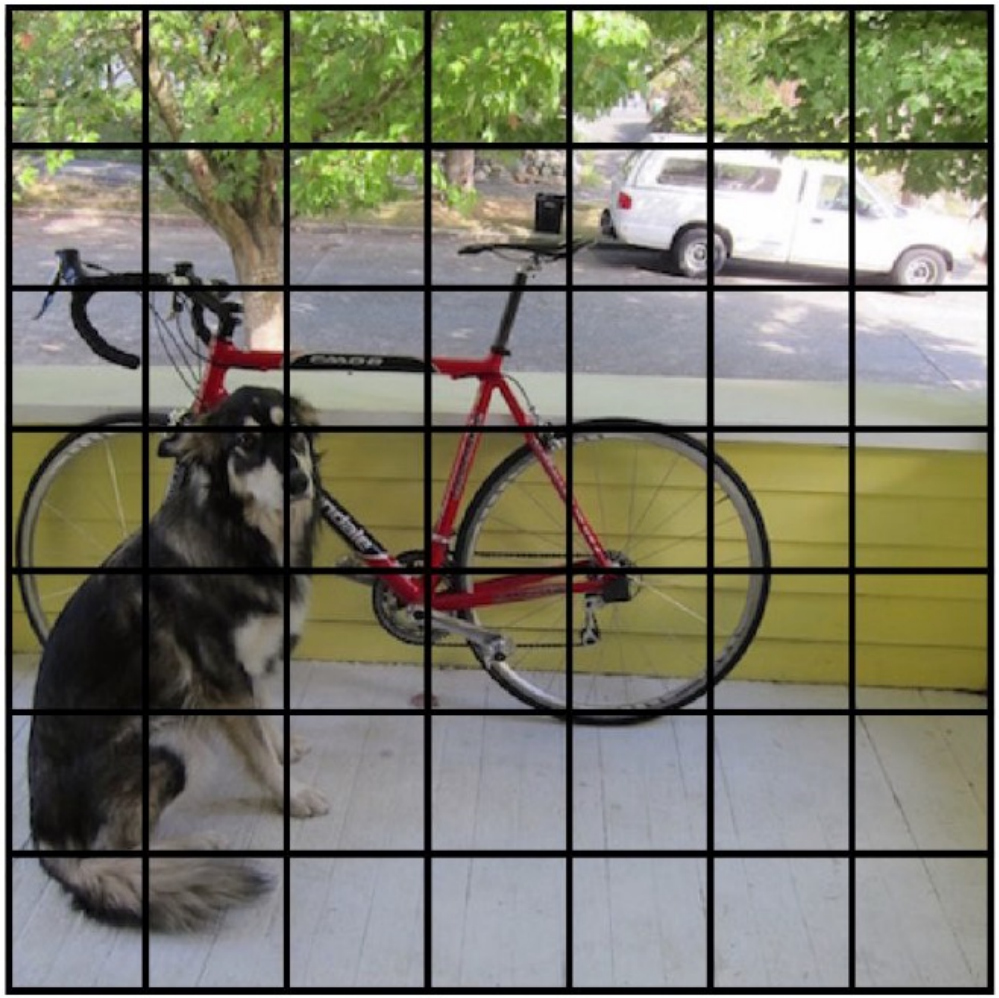
YOLO v8 is about as fast and accurate as you can get

link on webpage

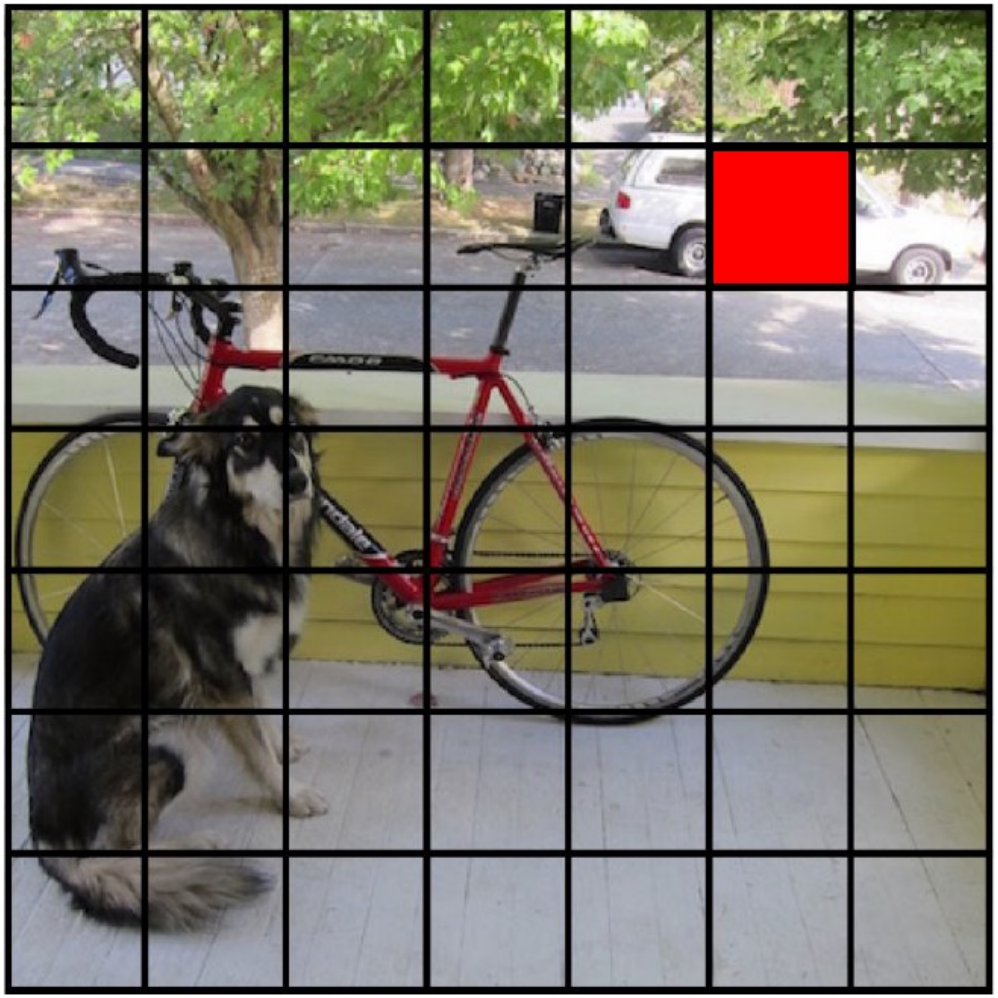
key idea

- look at box scores, label values independently

We split the image into a grid

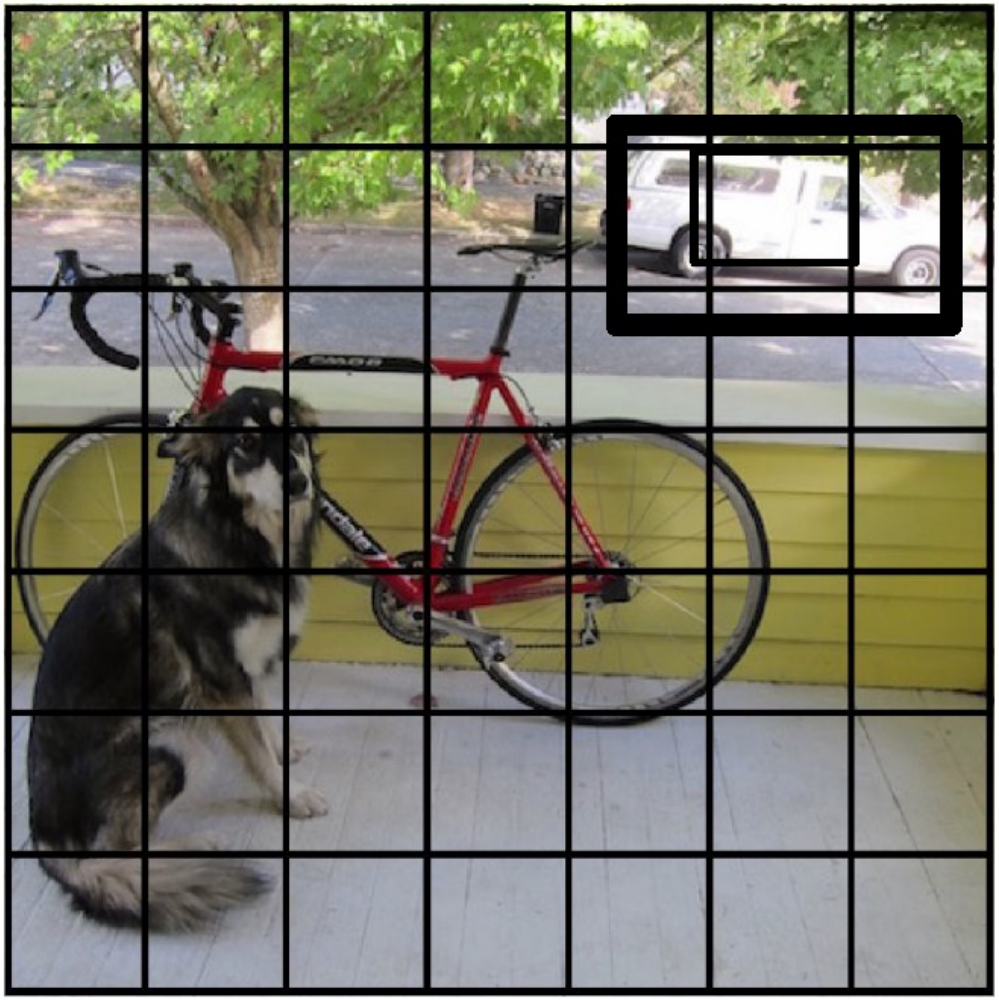


Each cell predicts boxes and confidences: P(Object)

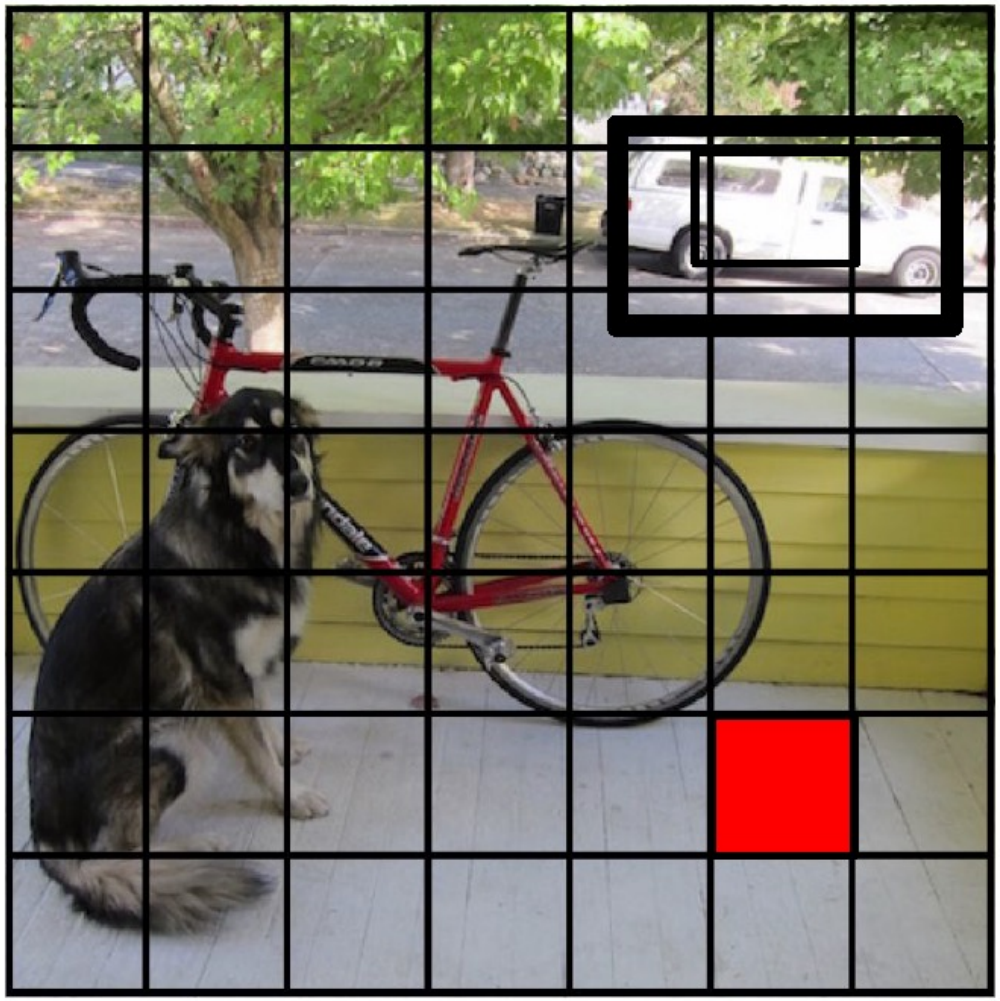


Each cell predicts boxes and confidences: P(Object)

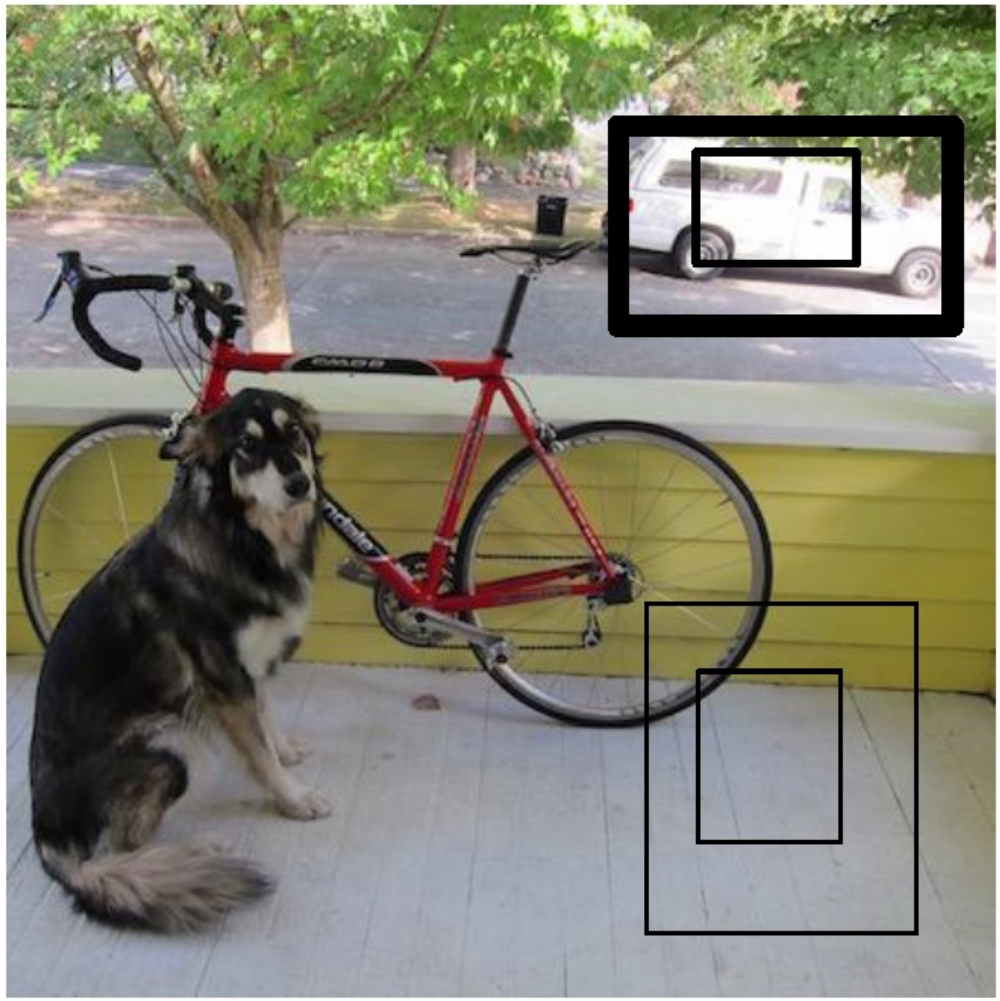
Each cell predicts boxes and confidences: P(Object)



Each cell predicts boxes and confidences: P(Object)

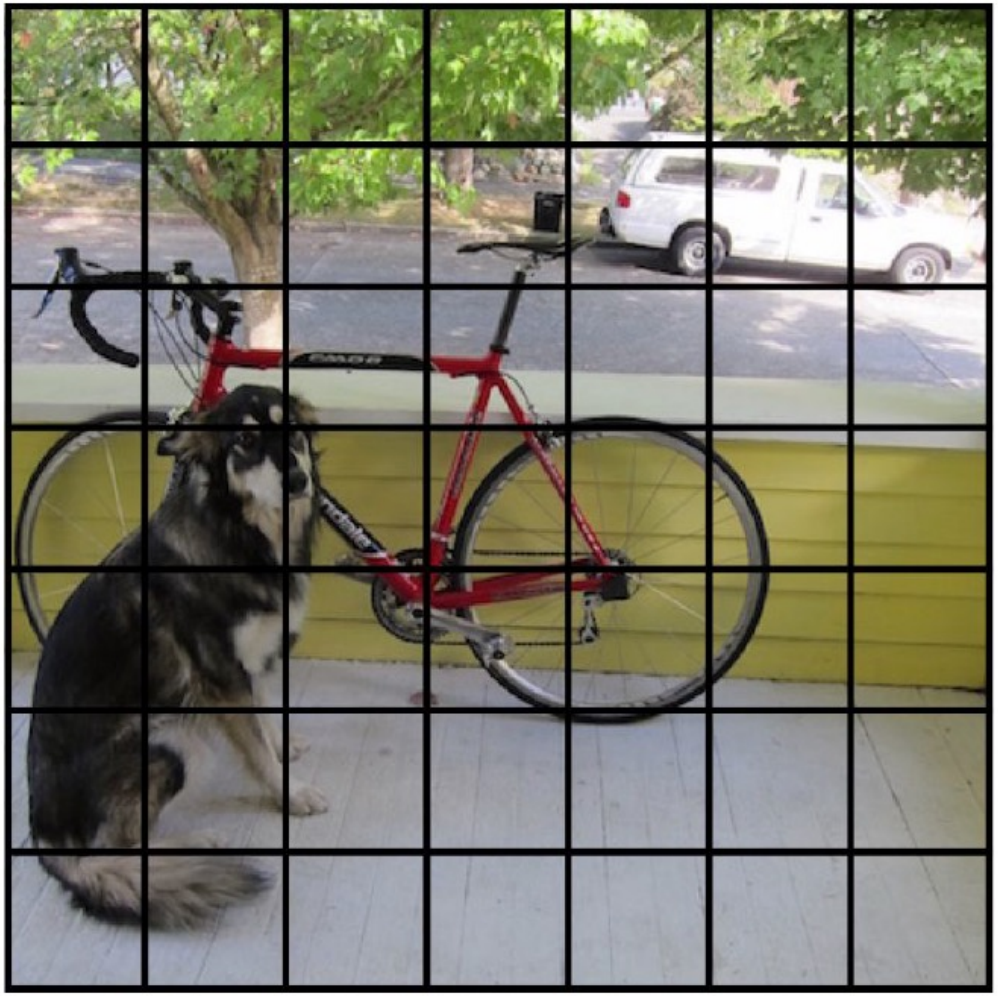


Each cell predicts boxes and confidences: P(Object)

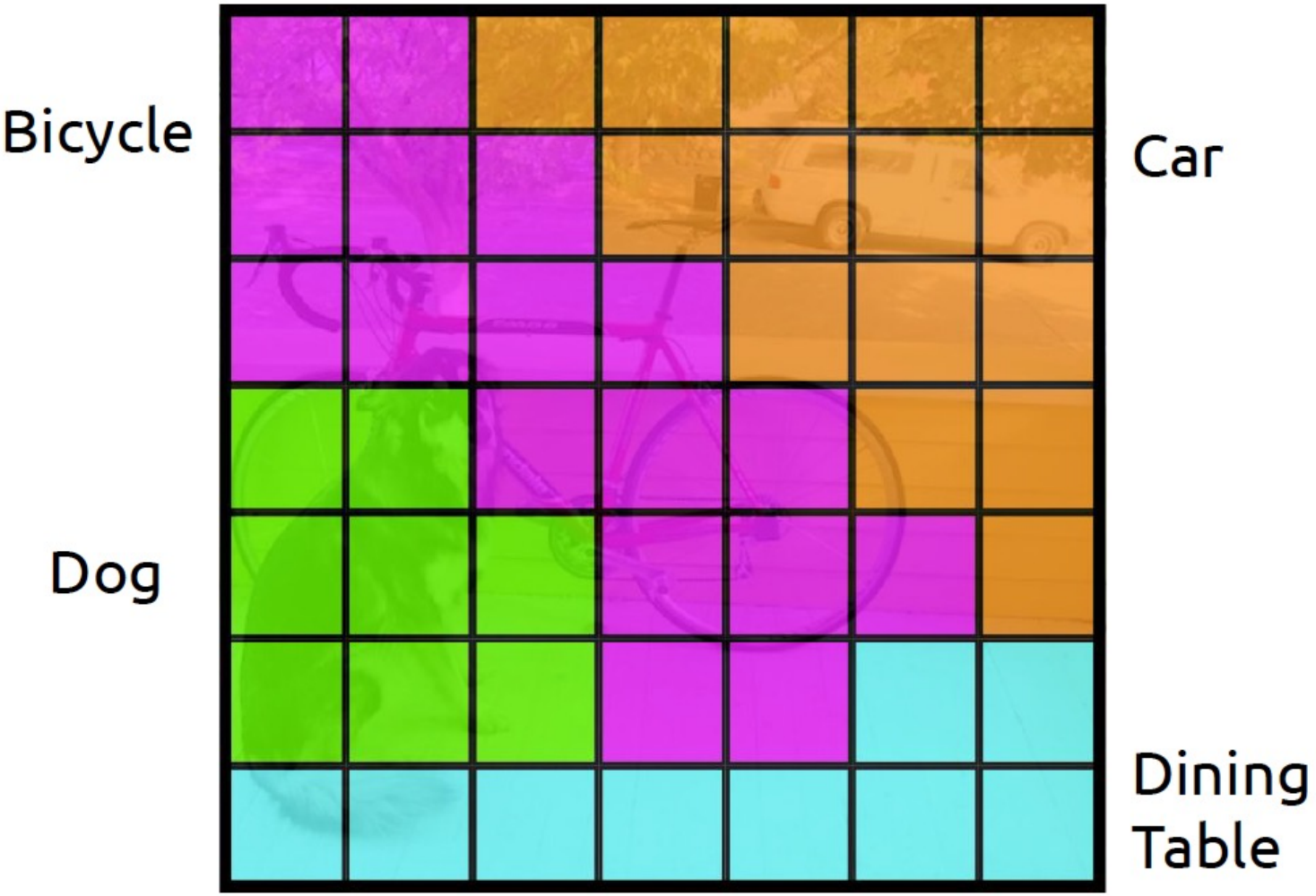


Each cell predicts boxes and confidences: P(Object)

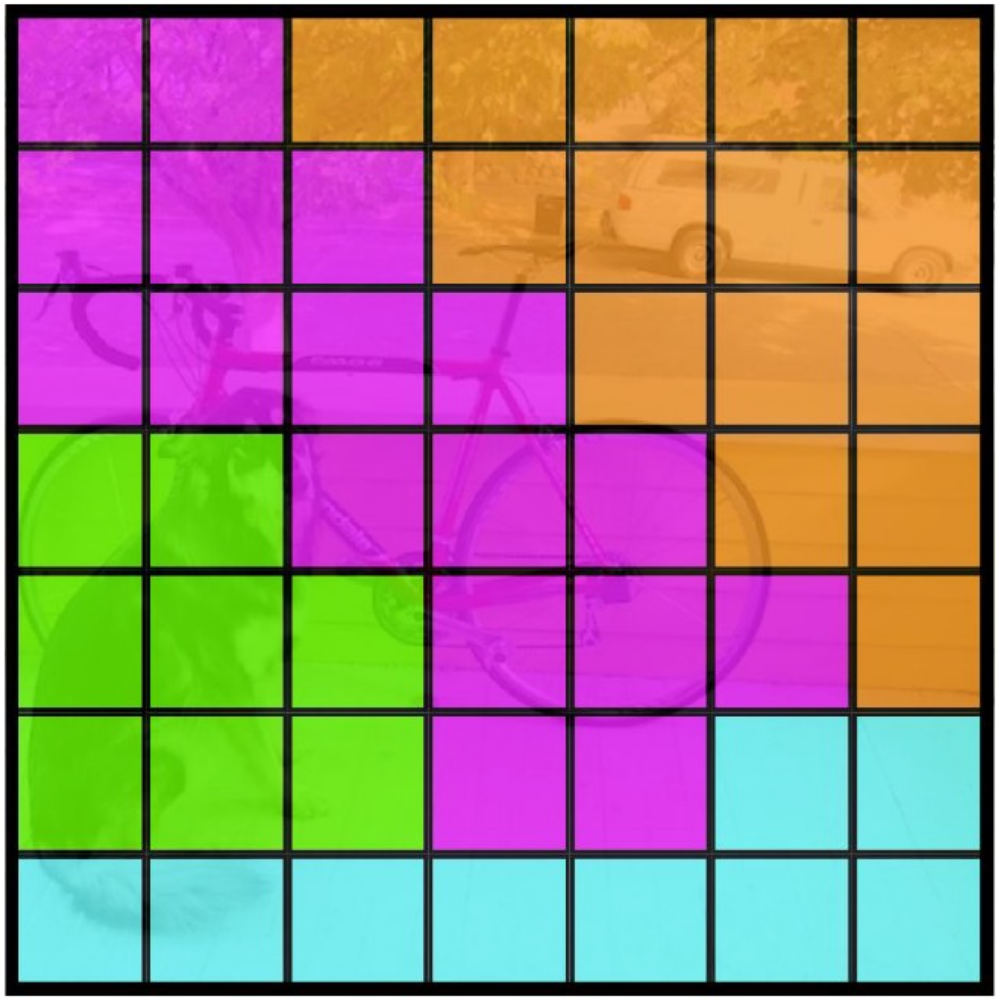
Each cell also predicts a class probability.



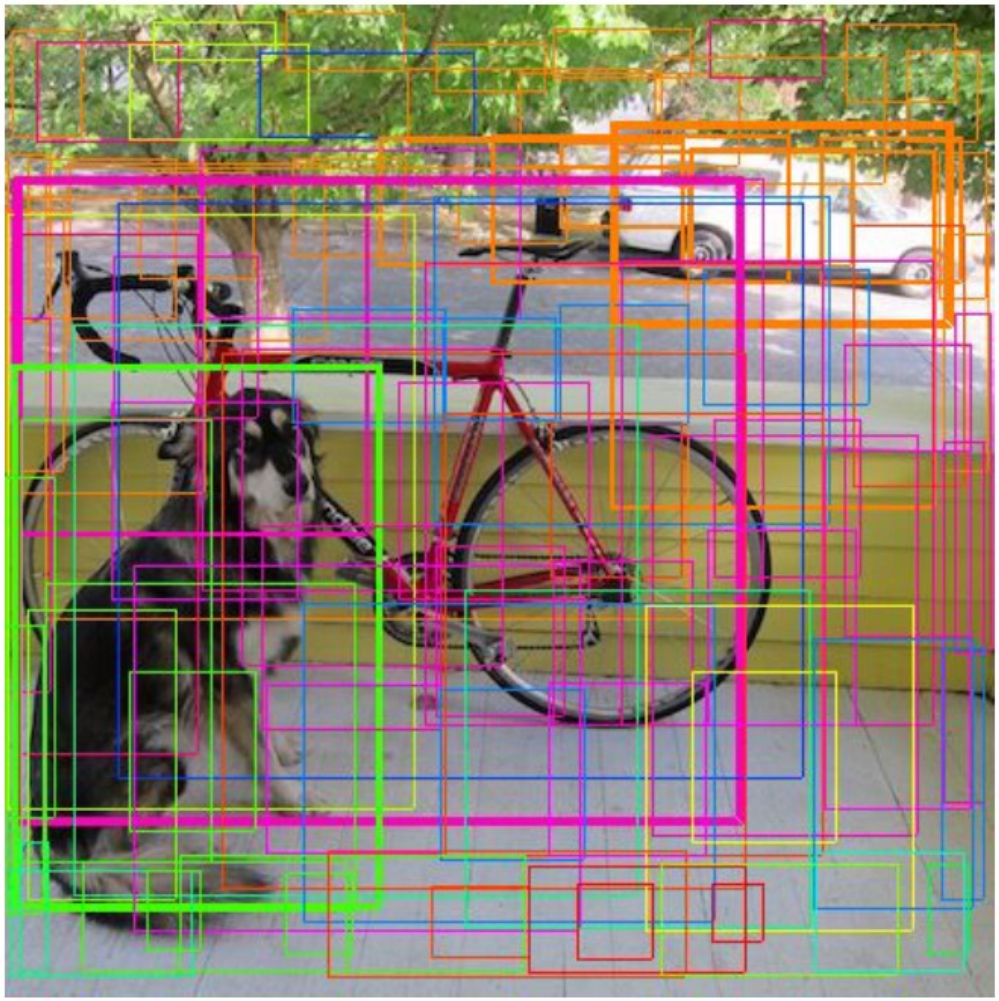
Each cell also predicts a class probability.



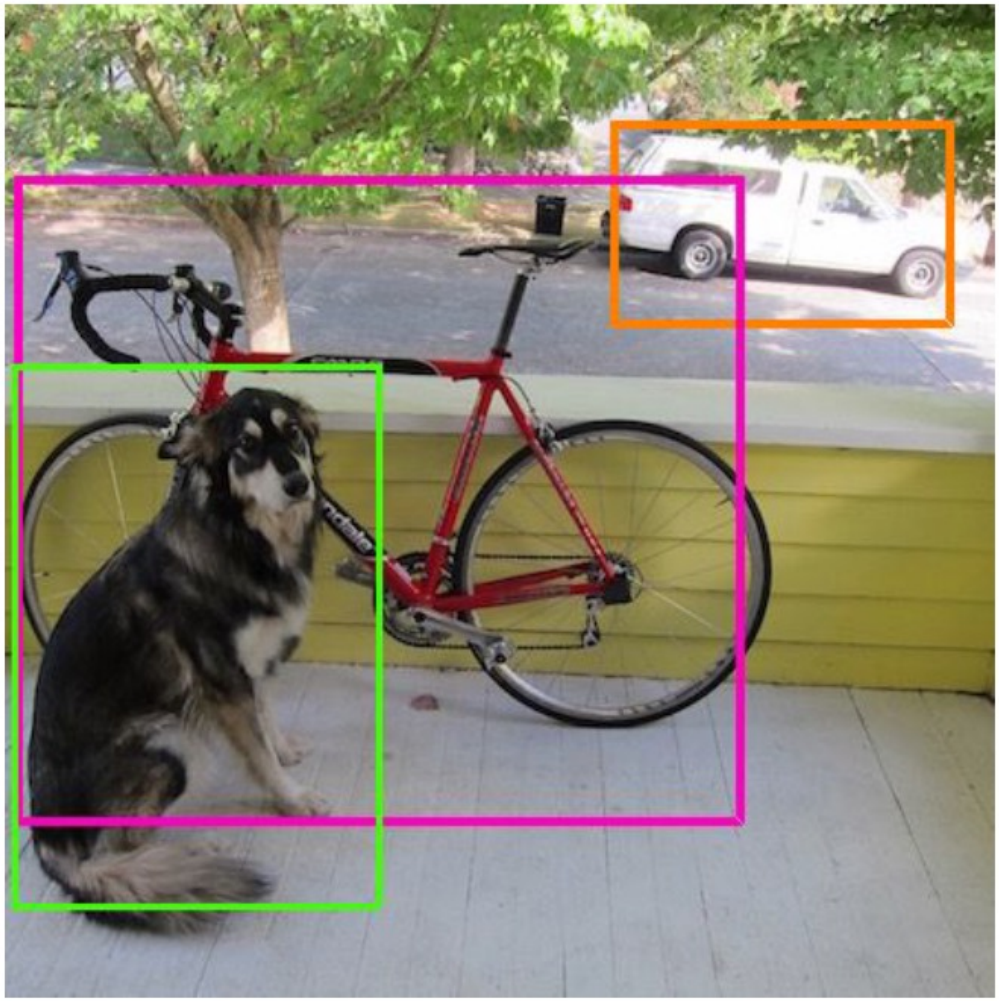
Conditioned on object: P(Car | Object)



Then we combine the box and class predictions.



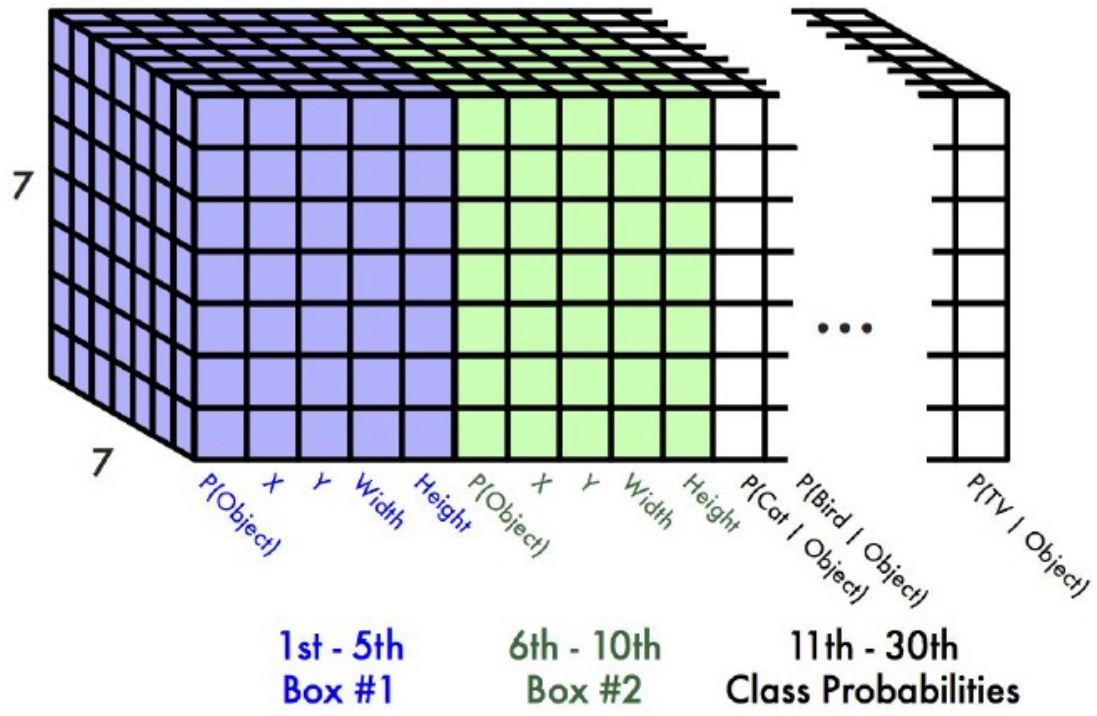
Finally we do NMS and threshold detections



This parameterization fixes the output size

Each cell predicts:

- For each bounding box:
  - 4 coordinates (x, y, w, h)
  - 1 confidence value
- Some number of class probabilities

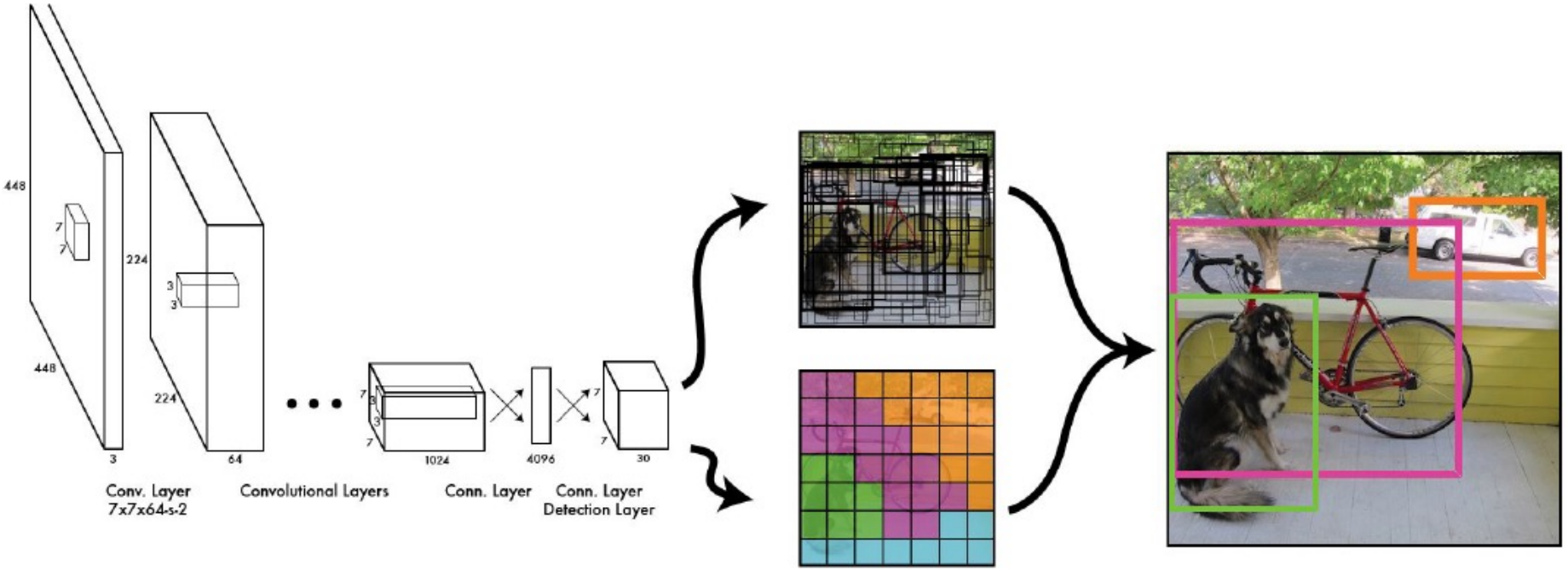


For Pascal VOC:

- 7x7 grid
- 2 bounding boxes / cell
- 20 classes

7 x 7 x (2 x 5 + 20) = 7 x 7 x 30 tensor = **1470 outputs**

Thus we can train one neural network to be a whole detection pipeline

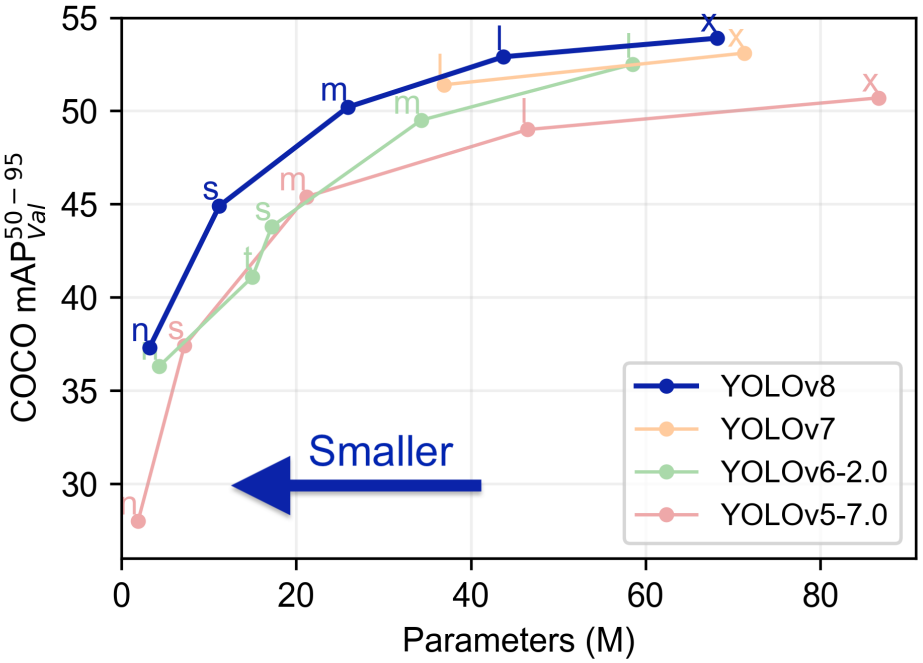


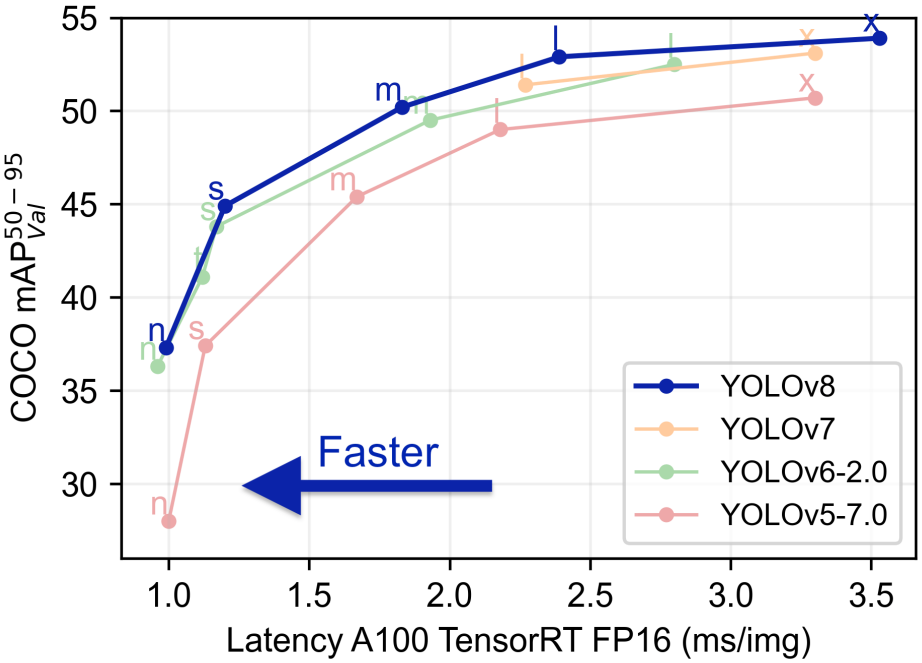
Evaluation, YOLOv8

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

https://github.com/ultralytics/ultralytics

YOLOv8 Tuning





SOA and variants: rough summary

Very accurate detection for hundreds of categories

- with enough training data
- important variations in training data available
  - – you don’t have to put a box on everything

YOLO allows a tradeoff between speed and accuracy

- and can be very fast

Variants

- Localization more accurate than boxes
- Incorporate LIDAR, etc.
- Boxes in 3D rather than 2D
- Variant feature constructions are very important

Ghost(s) at the party

Tremblay et al 20