Presented by:

Jie Li
Takuto Sato

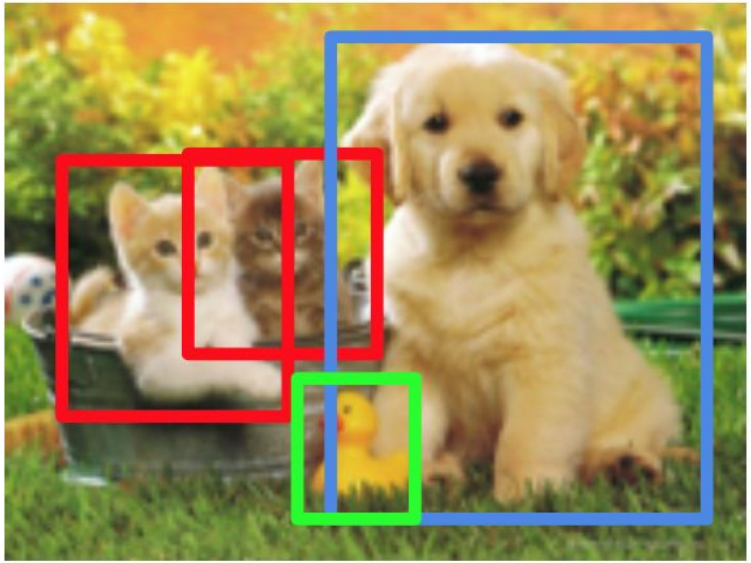
Original Slides from:
https://pjreddie.com/darknet/yolo/

Classification + Localization

CAT

v.s.

Object Detection



CAT, DOG, DUCK

Localization as Regression

Input: image

Neural Net

Output:
Box coordinates
(4 numbers)

Correct output:
box coordinates
(4 numbers)

Loss:
L2 distance

Only one object,
simpler than detection

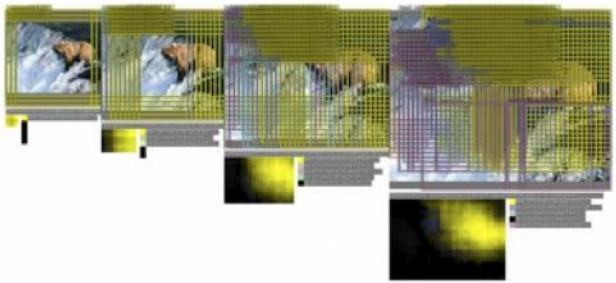
* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

Sliding Window: Overfeat

(DPM as well!)

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

* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

Detection as Regression?

DOG, (x, y, w, h)
CAT, (x, y, w, h)
CAT, (x, y, w, h)
DUCK (x, y, w, h)

= 16 numbers

* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

Detection as Regression?

DOG, (x, y, w, h)
CAT, (x, y, w, h)

= 8 numbers

* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

Detection as Regression?

CAT, (x, y, w, h)
CAT, (x, y, w, h)
....
CAT (x, y, w, h)

= many numbers

Need variable sized outputs

* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

Detection as Classification

CAT? NO

DOG? NO

* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

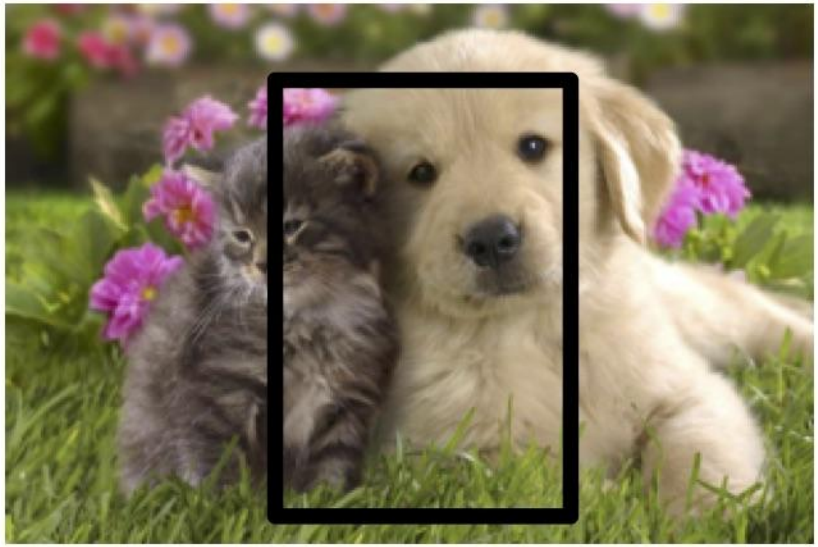
Detection as Classification

CAT? YES!

DOG? NO

* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

Detection as Classification



CAT? NO

DOG? NO

* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

Detection as Classification

Problem: Need to test many positions and scales

Solution: If your classifier is fast enough, just do it

* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

Detection as Classification

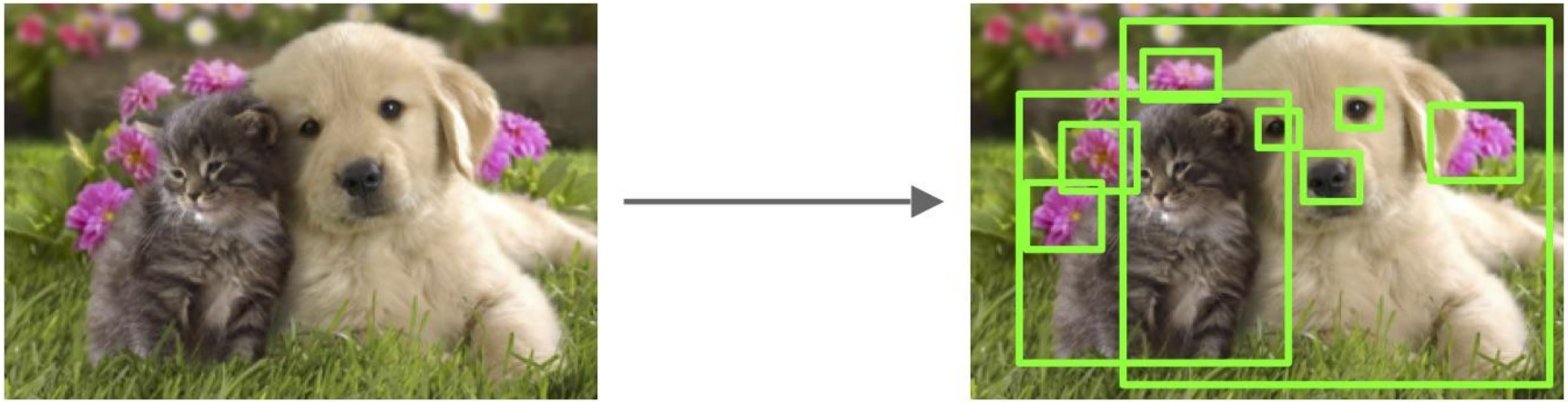
Problem: Need to test many positions and scales, and use a computationally demanding classifier (CNN)

Solution: Only look at a tiny subset of possible positions

* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

Region Proposals

- Find “blobby” image regions that are likely to contain objects
- “Class-agnostic” object detector
- Look for “blob-like” regions

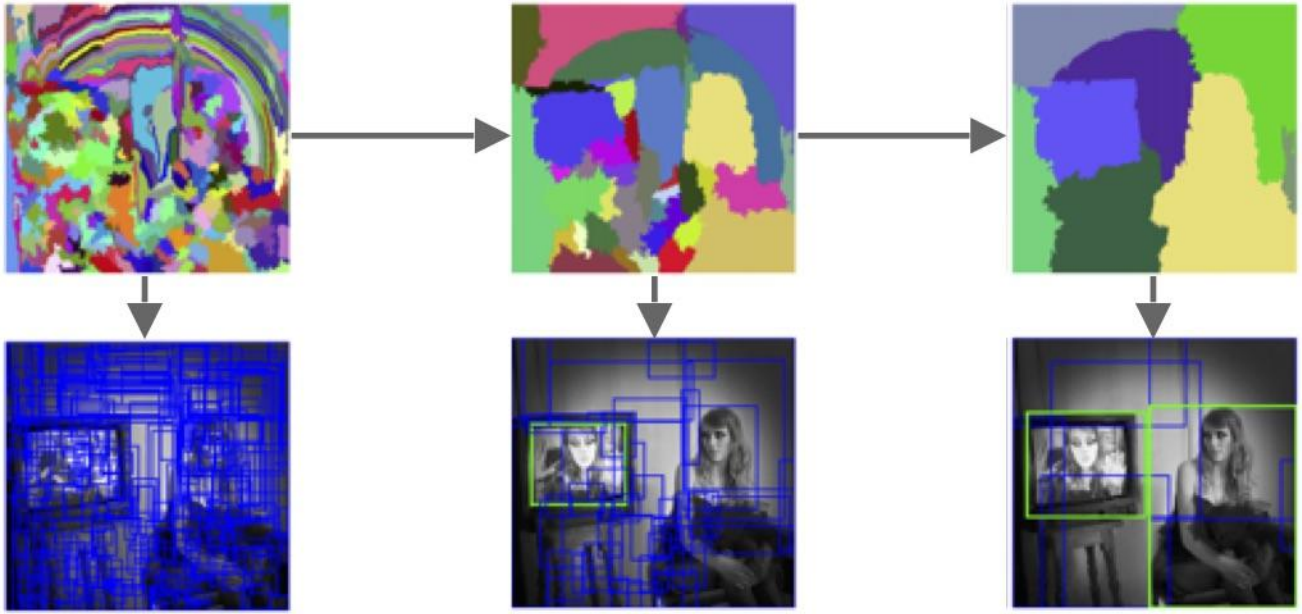


* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

Region Proposals: Selective Search

Bottom-up segmentation, merging regions at multiple scales

Convert regions to boxes



Uijlings et al, "Selective Search for Object Recognition", IJCV 2013

* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

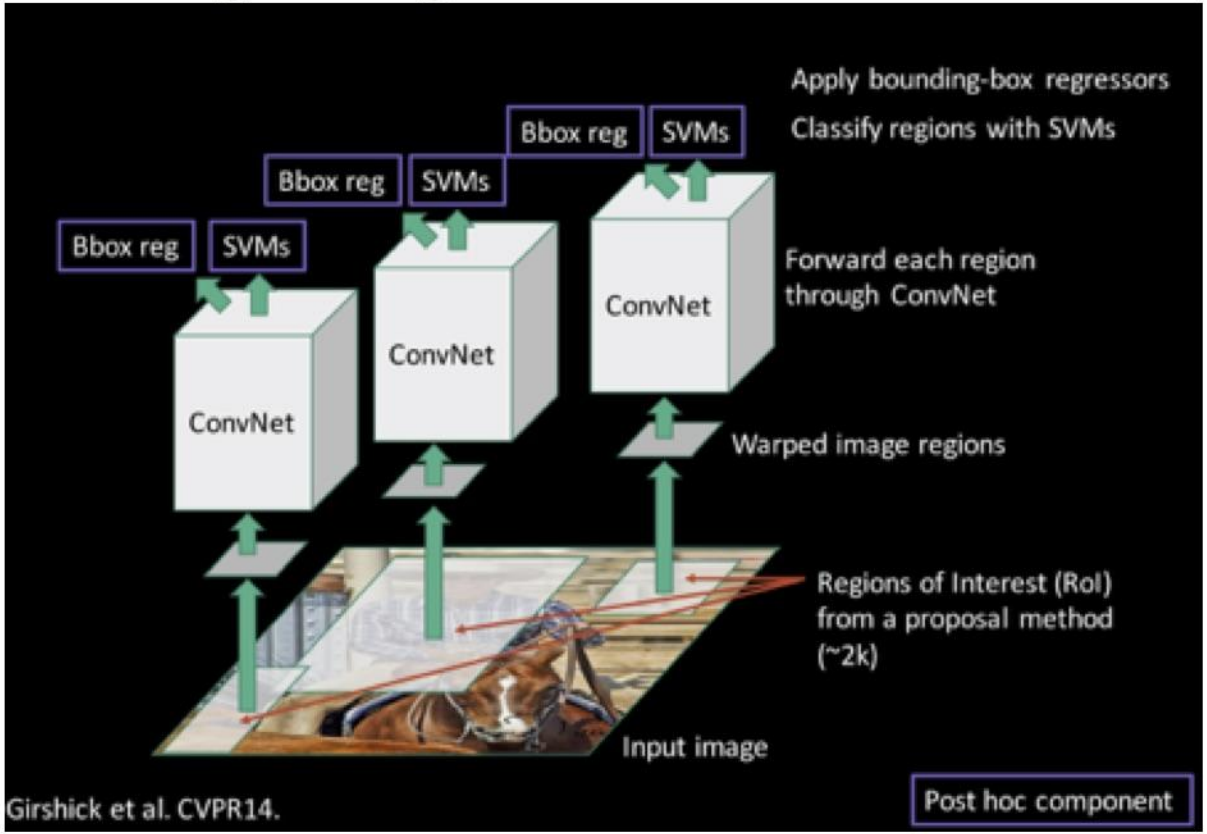
Region Proposals: Many other choices

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	·	·	·

Hosang et al, "What makes for effective detection proposals?", PAMI 2015

* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

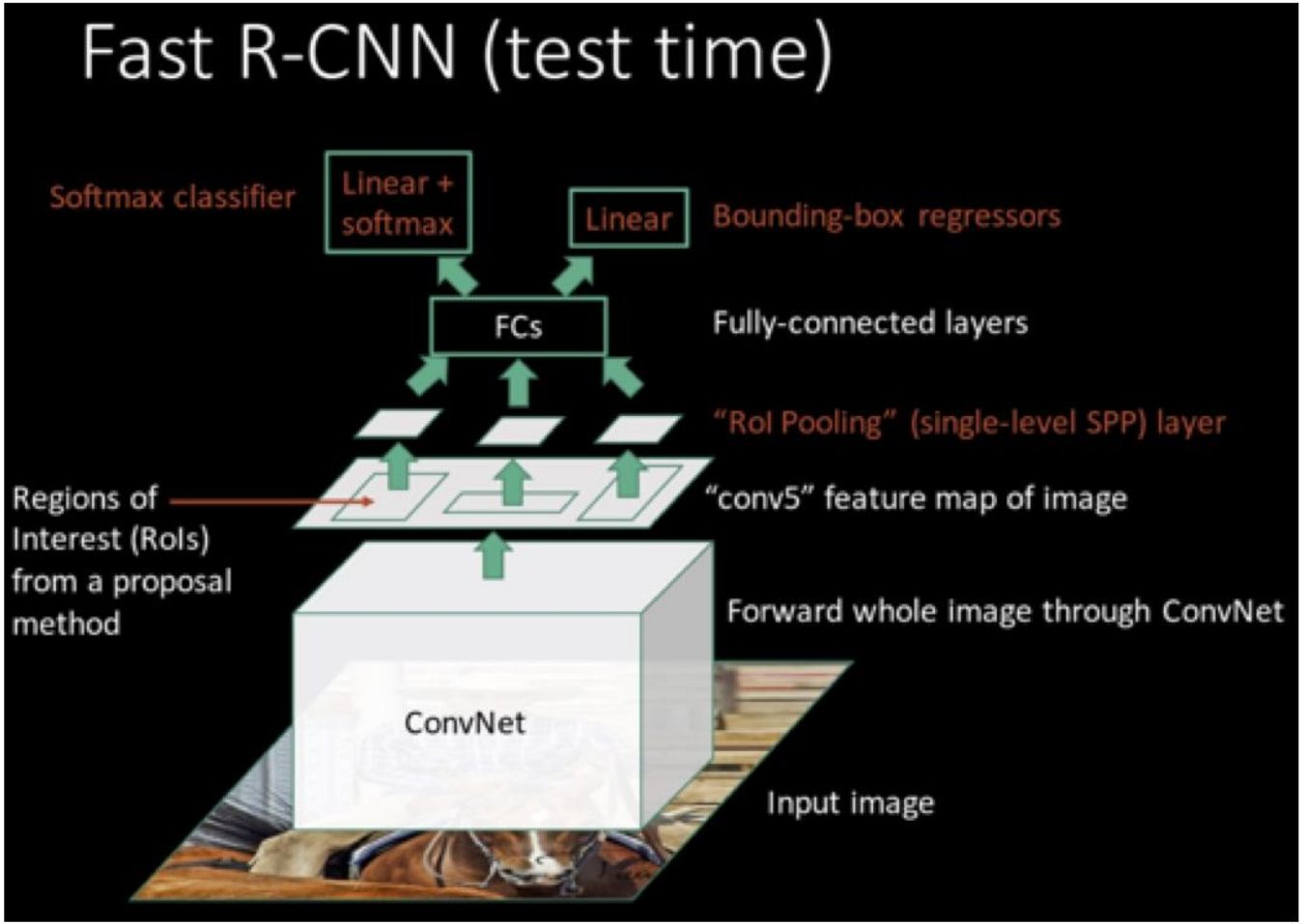
Putting it together: R-CNN



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

Slide credit: Ross Girschick

* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

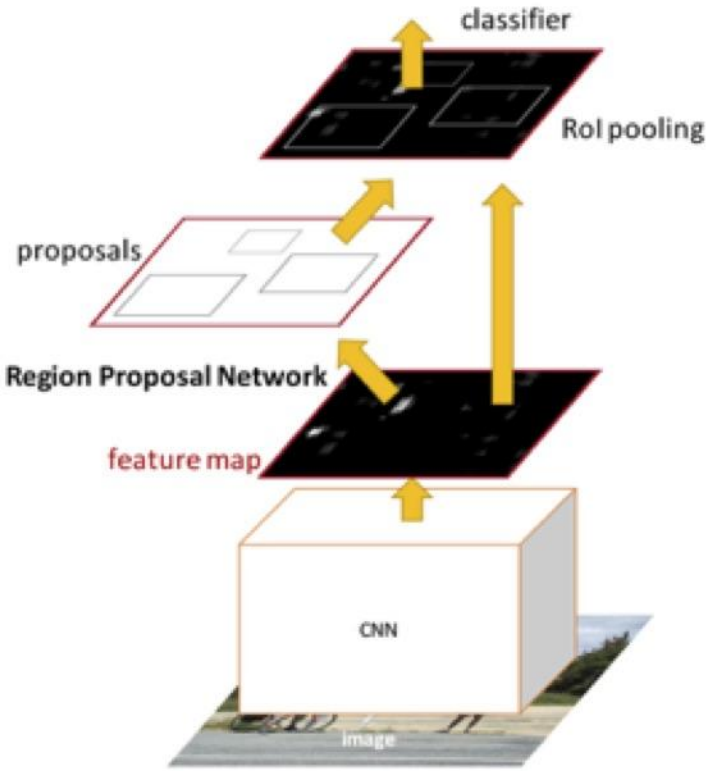


R-CNN Problem #1:
Slow at test-time due to independent forward passes of the CNN

Solution:
Share computation of convolutional layers between proposals for an image

* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

Faster R-CNN:



Insert a **Region Proposal Network (RPN)** after the last convolutional layer

RPN trained to produce region proposals directly; no need for external region proposals!

After RPN, use RoI Pooling and an upstream classifier and bbox regressor just 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

* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

Summary:

How to frame detection problem?

Regression?

- OK for one or a fixed number of objects (localization with sliding windows)
- Sliding windows is slow (especially when the number of objects too large)
- Number of objects unknown?

Classification?

- Need region-based method to propose bounding boxes (~2000 boxes, R-CNN)
- Still kind of slow (especially considering real-time detection)

You Only Look Once: Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/

Abstract

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other de-

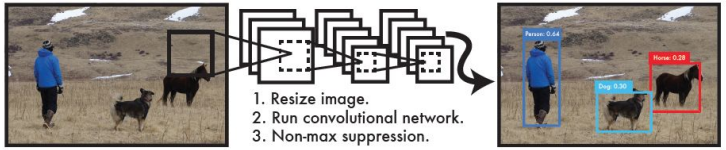
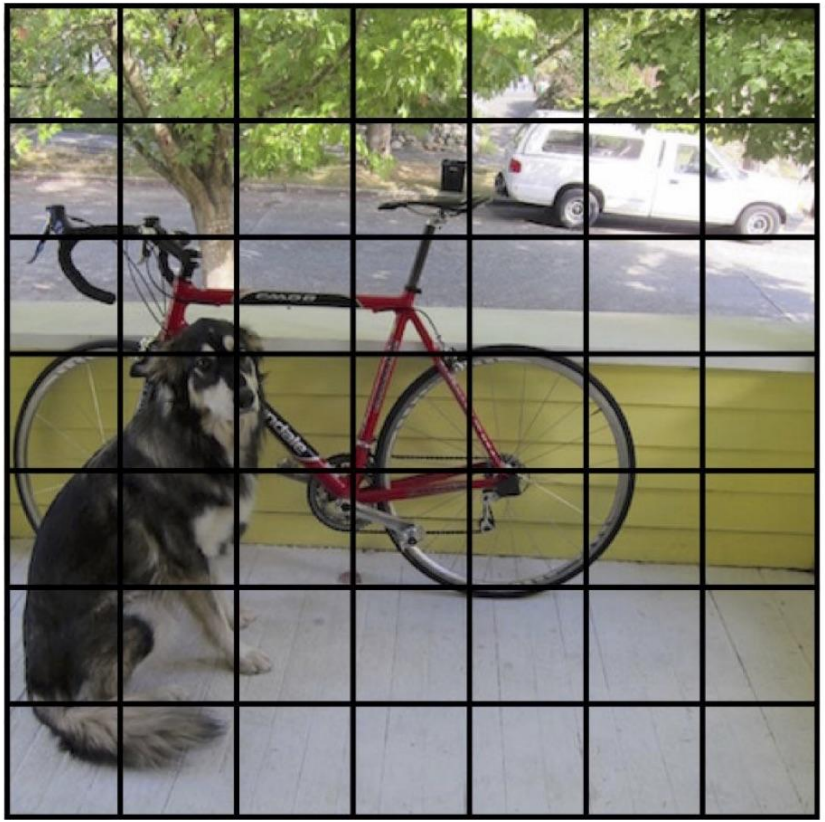


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene [13]. These complex pipelines are slow and hard to optimize because each individual component must be trained separately.

Original Slides from: https://pjreddie.com/darknet/yolo/

We split the image into a grid

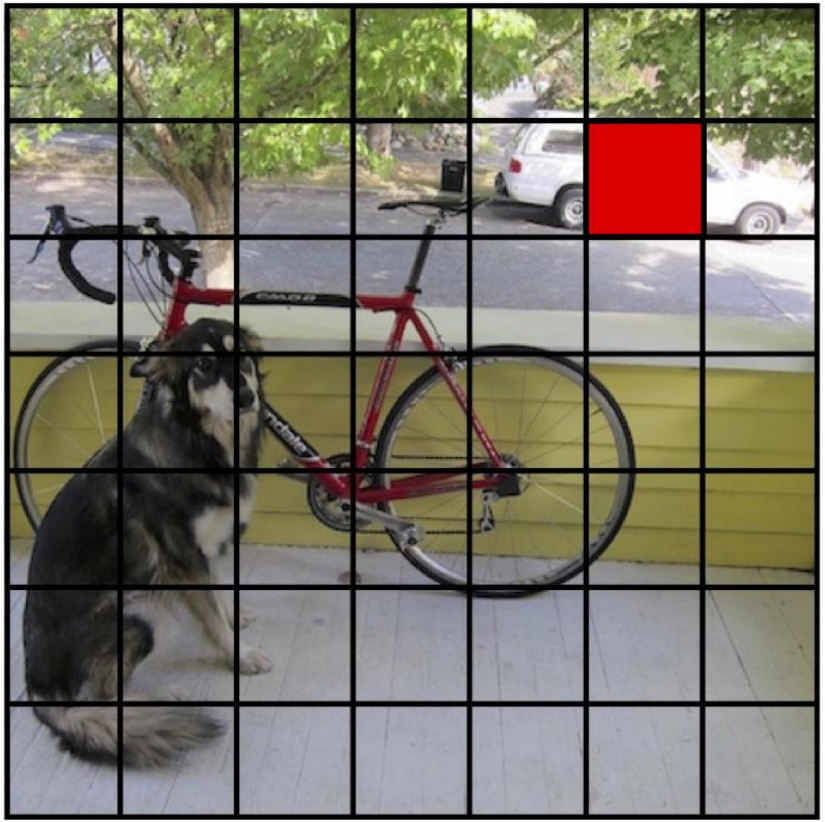


S = 7

Total # of Cells:
S x S =49

Original Slides from:
https://pjreddie.com/darknet/yolo/

Each cell predicts boxes and confidences: P(Object)

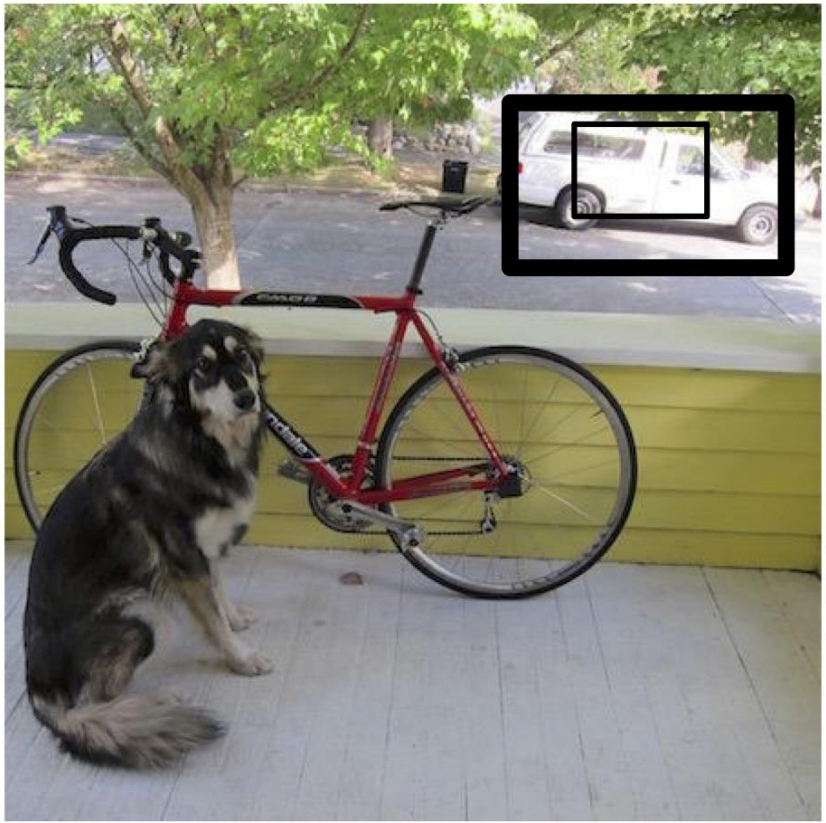


S = 7

Total # of Cells:
S x S =49

Original Slides from:
https://pjreddie.com/darknet/yolo/

Each cell predicts boxes and confidences: P(Object)



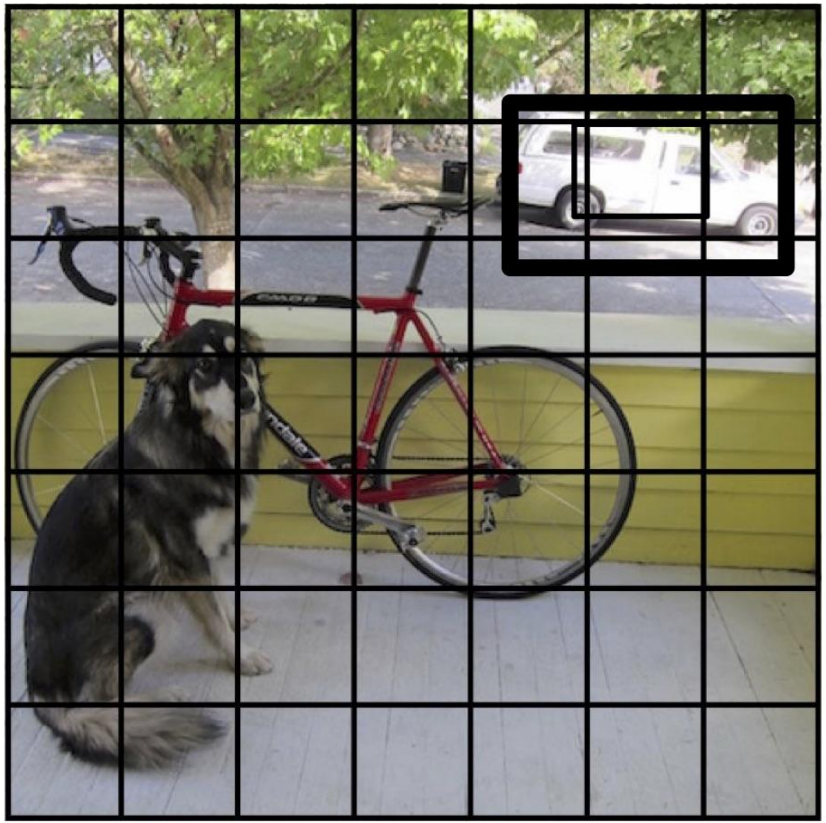
S = 7

Total # of Cells:
S x S =49

B = 2

Original Slides from:
https://pjreddie.com/darknet/yolo/

Each cell predicts boxes and confidences: P(Object)



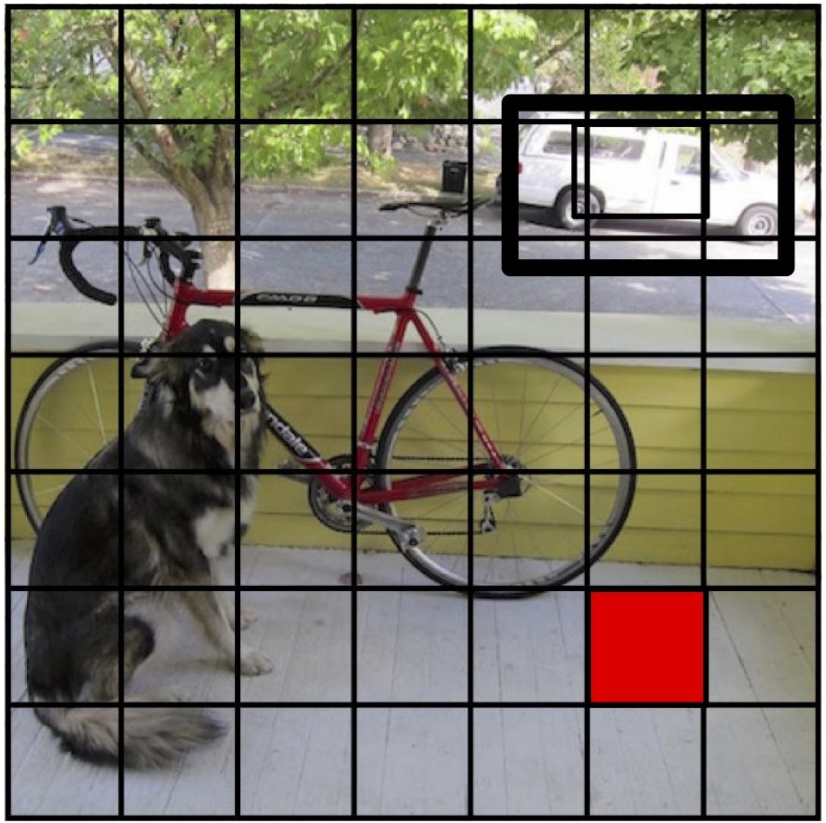
S = 7

Total # of Cells:
S x S =49

B = 2

Original Slides from:
https://pjreddie.com/darknet/yolo/

Each cell predicts boxes and confidences: P(Object)



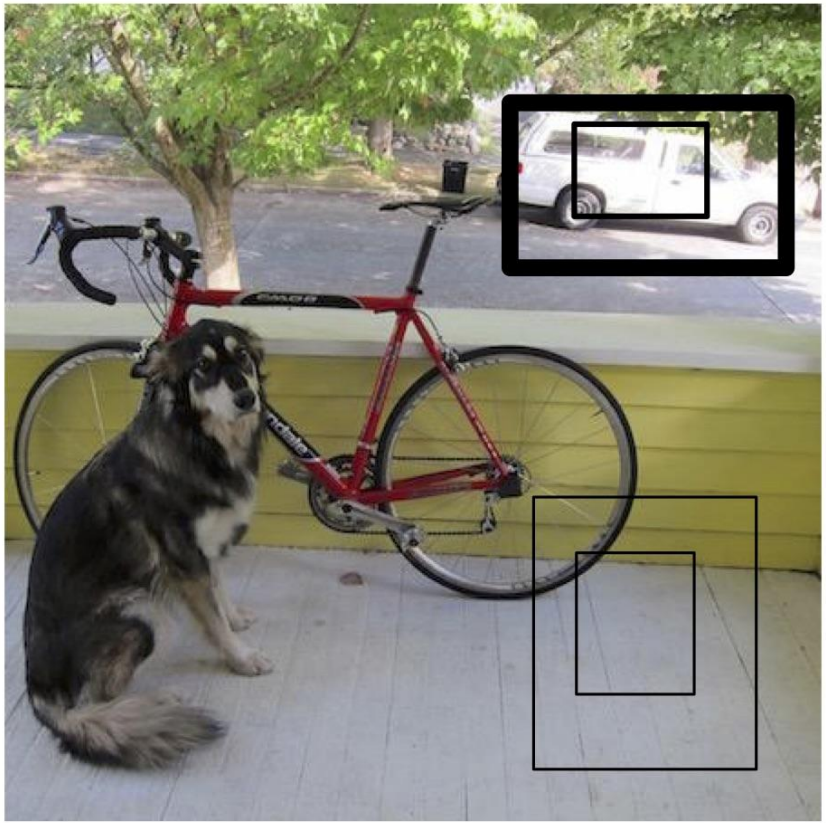
S = 7

Total # of Cells:
S x S =49

B = 2

Original Slides from:
https://pjreddie.com/darknet/yolo/

Each cell predicts boxes and confidences: P(Object)



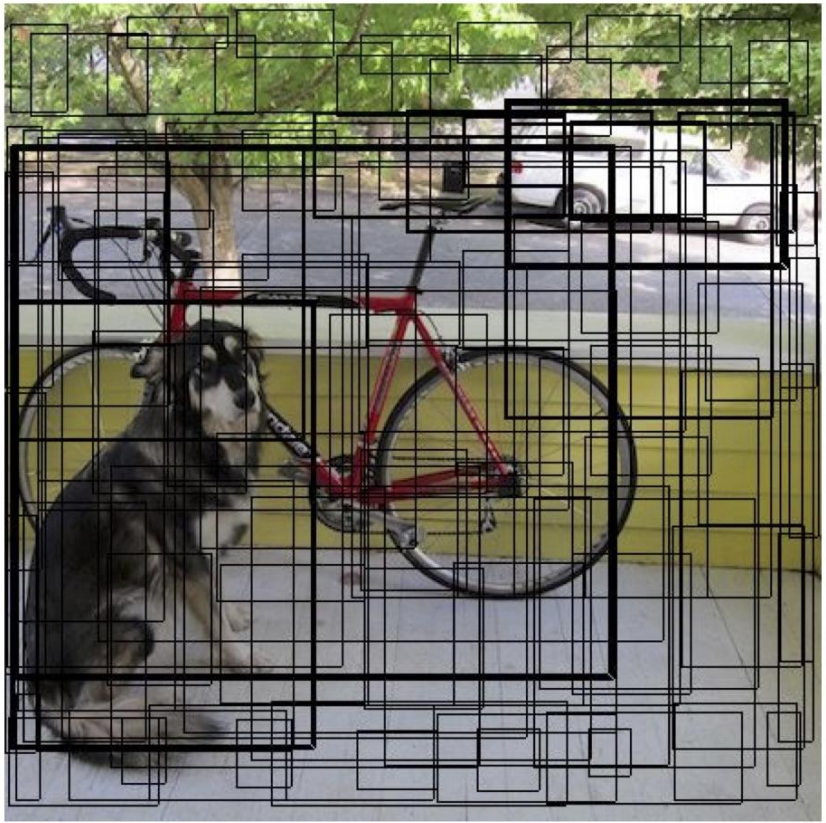
S = 7

Total # of Cells:
S x S =49

B = 2

Original Slides from:
https://pjreddie.com/darknet/yolo/

Each cell predicts boxes and confidences: P(Object)



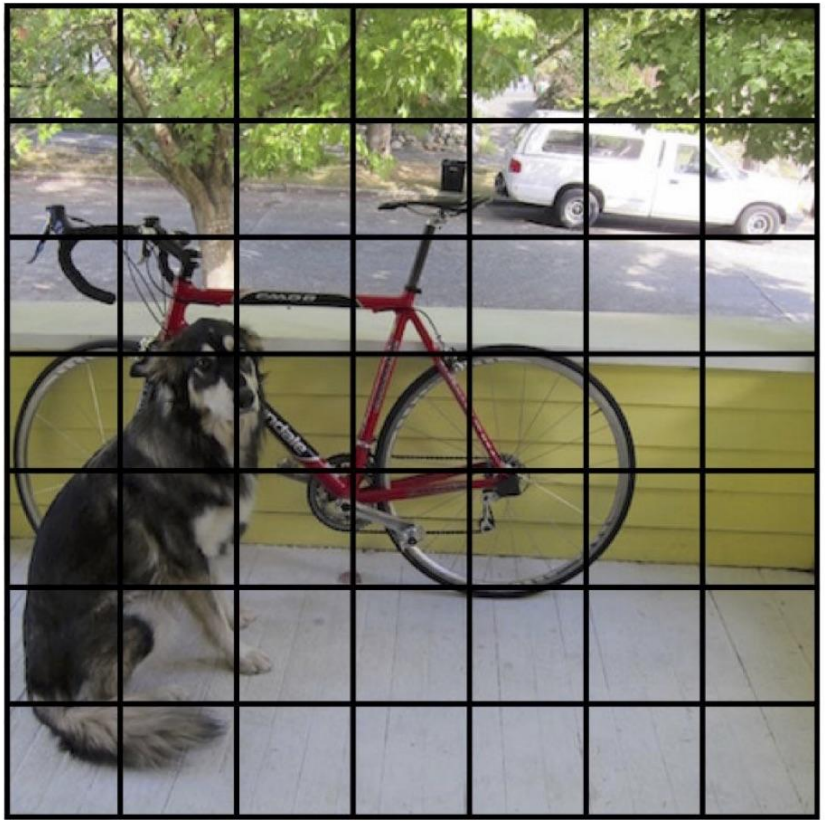
S = 7

Total # of Cells:
S x S =49

B = 2
Total # of boxes:
S x S x B = 98

Original Slides from:
https://pjreddie.com/darknet/yolo/

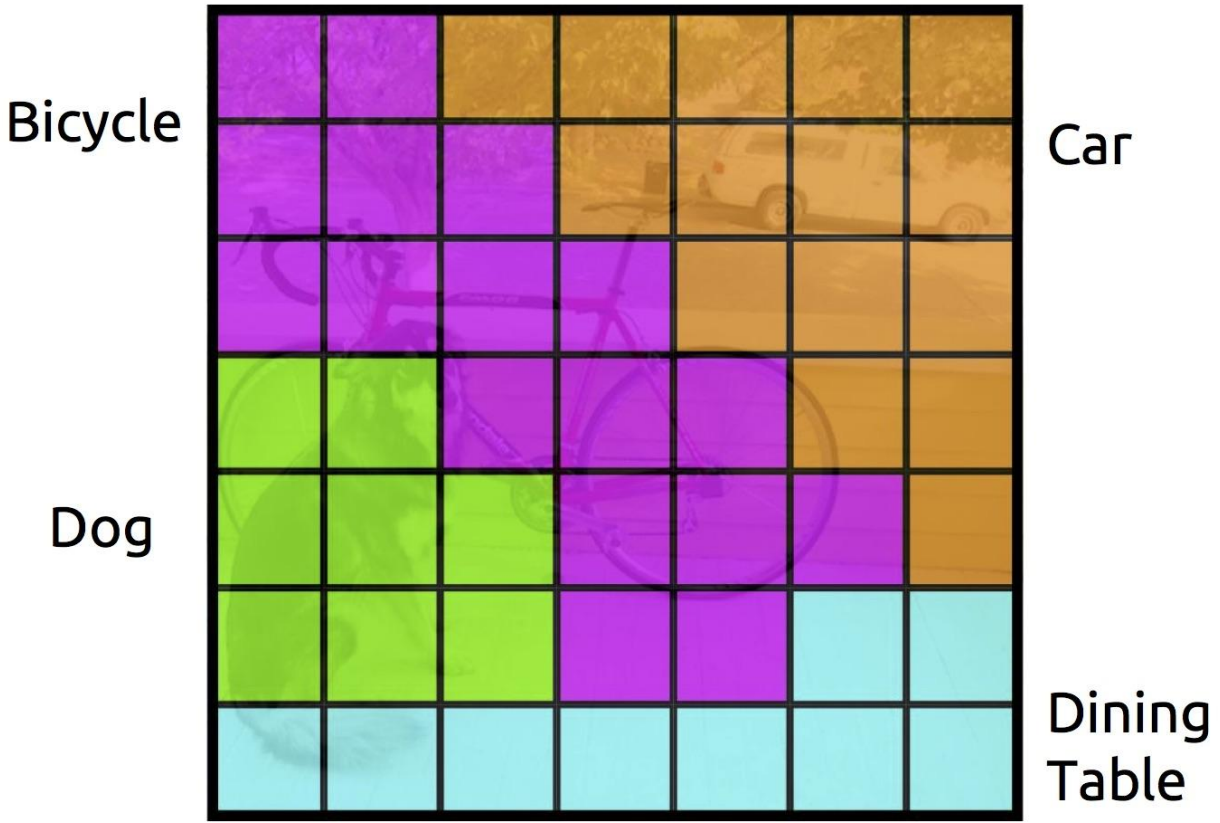
Each cell also predicts a class probability.



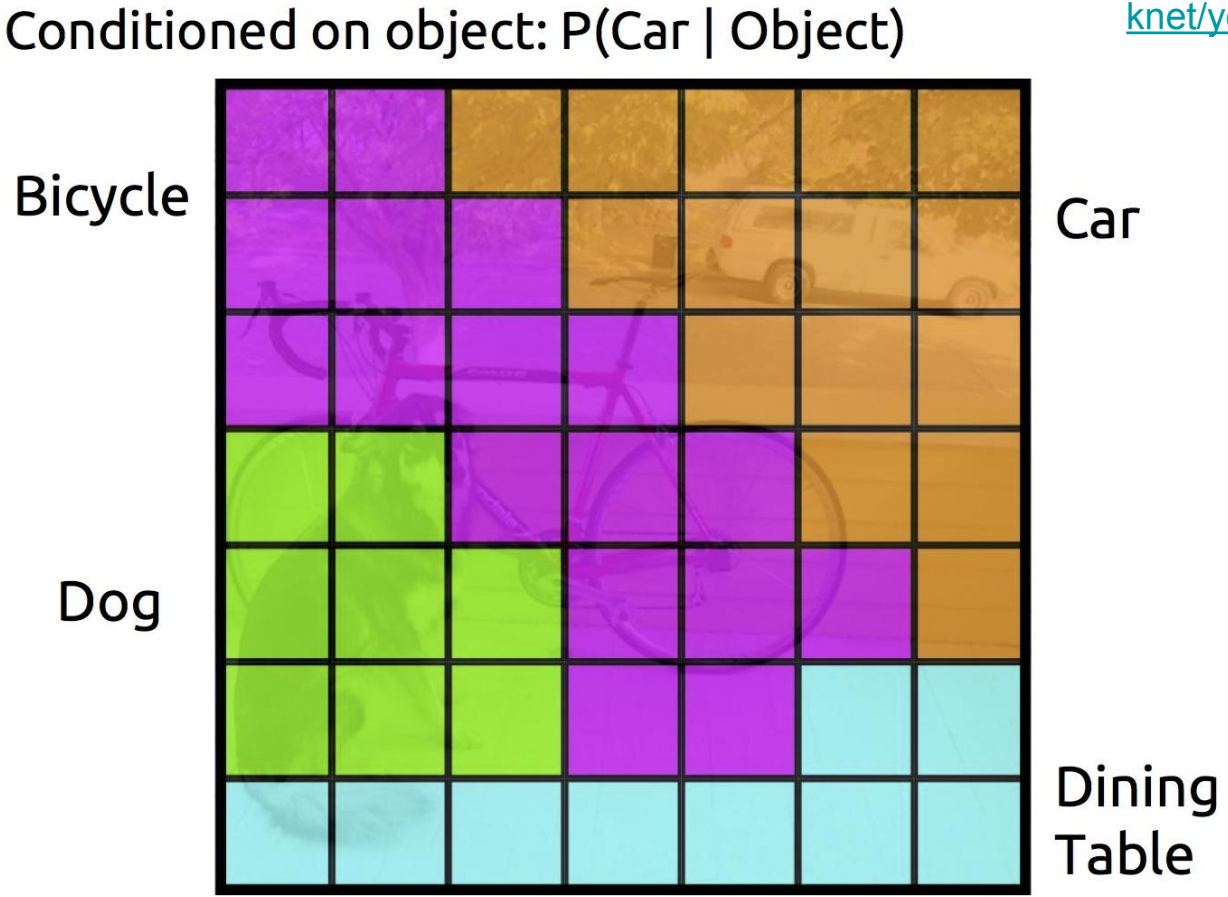
Original Slides from:
https://pjreddie.com/darknet/yolo/

Original Slides from:
https://pjreddie.com/darknet/yolo/

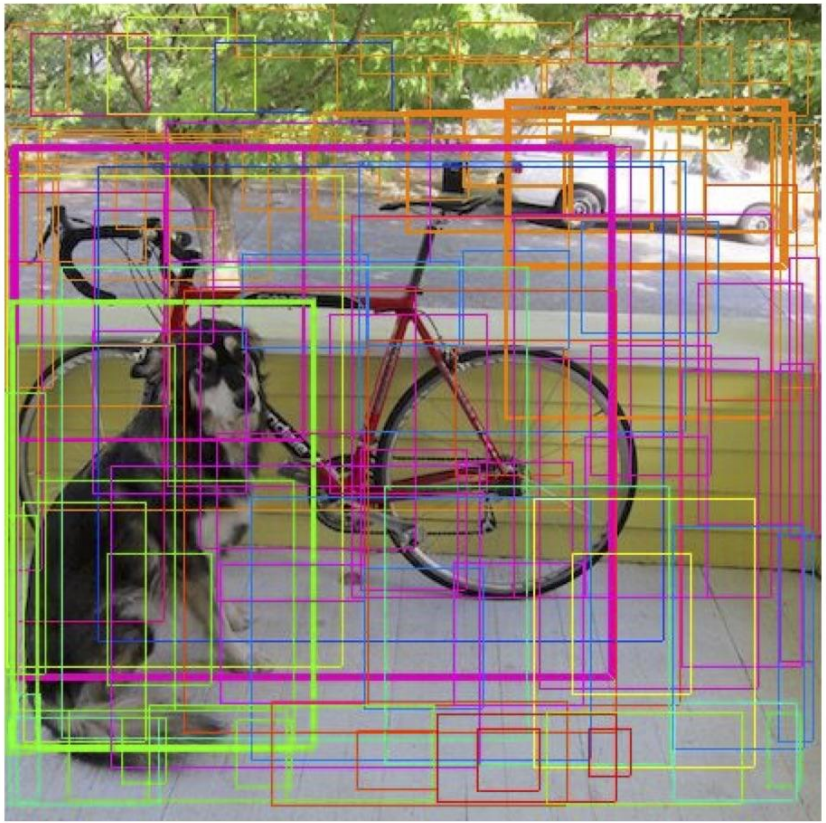
Each cell also predicts a class probability.



Original Slides from:
https://pjreddie.com/darknet/yolo/

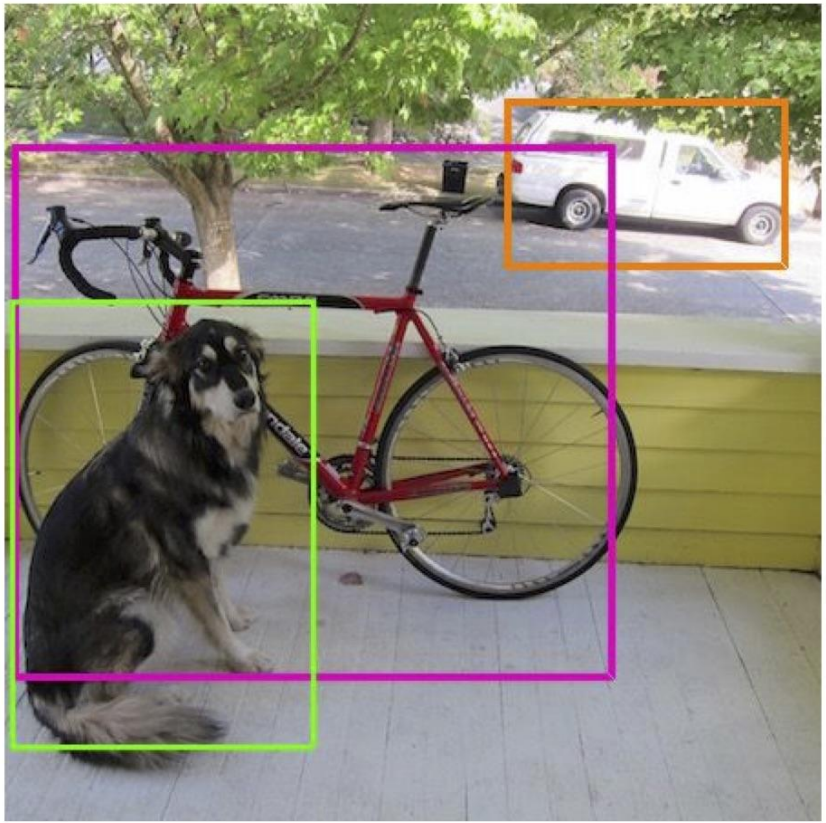


Then we combine the box and class predictions.



Original Slides from:
https://pjreddie.com/darknet/yolo/

Finally we do NMS and threshold detections

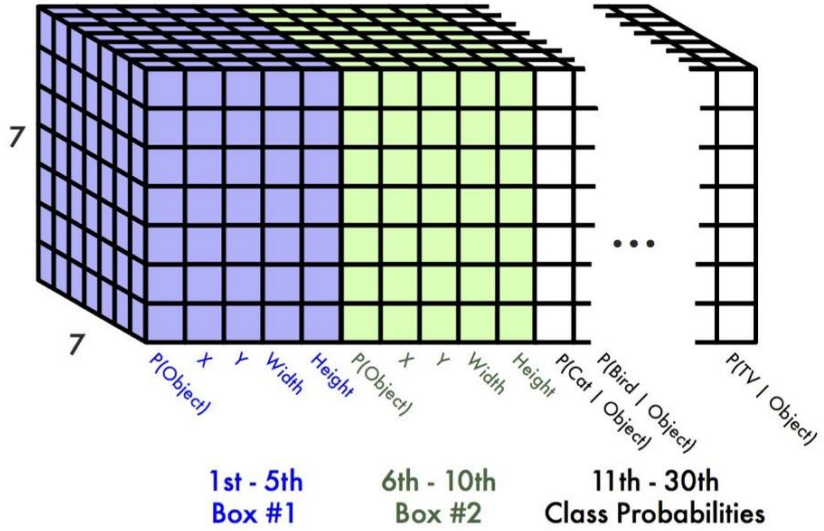


Original Slides from:
https://pjreddie.com/darknet/yolo/

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



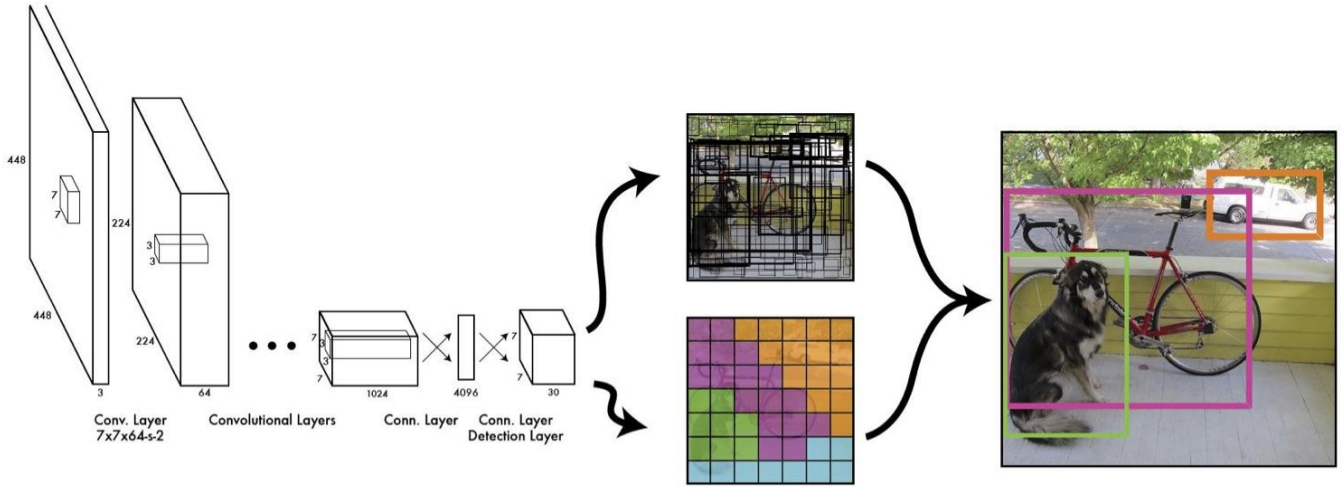
For Pascal VOC:

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

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

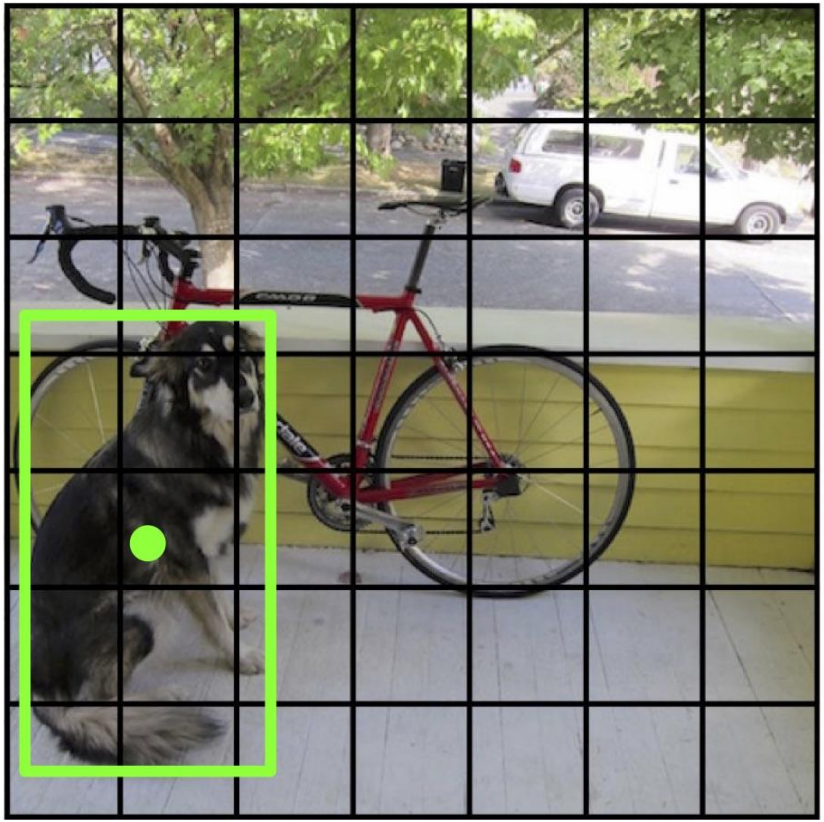
Original Slides from: https://pjreddie.com/darknet/yolo/

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



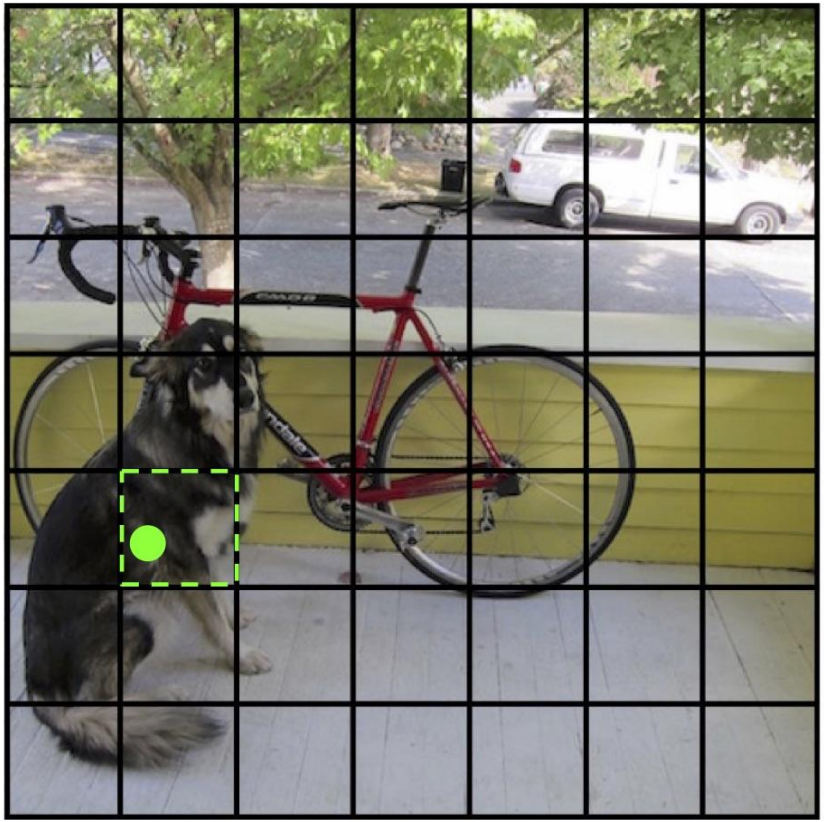
Original Slides from: https://pjreddie.com/darknet/yolo/

During training, match example to the right cell



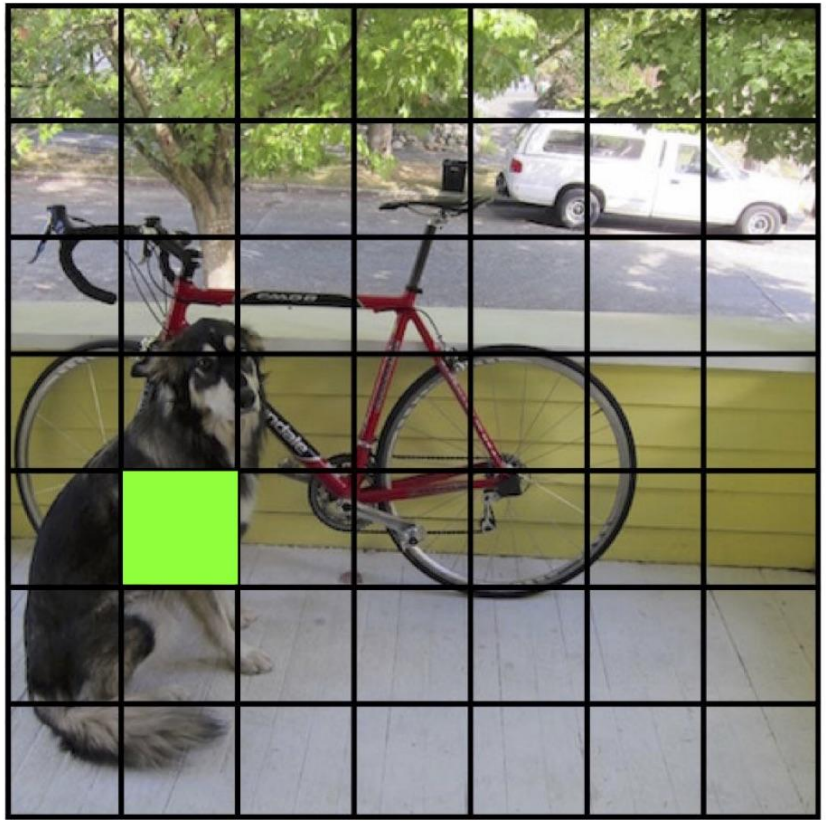
Original Slides from:
https://pjreddie.com/darknet/yolo/

During training, match example to the right cell



Original Slides from:
https://pjreddie.com/darknet/yolo/

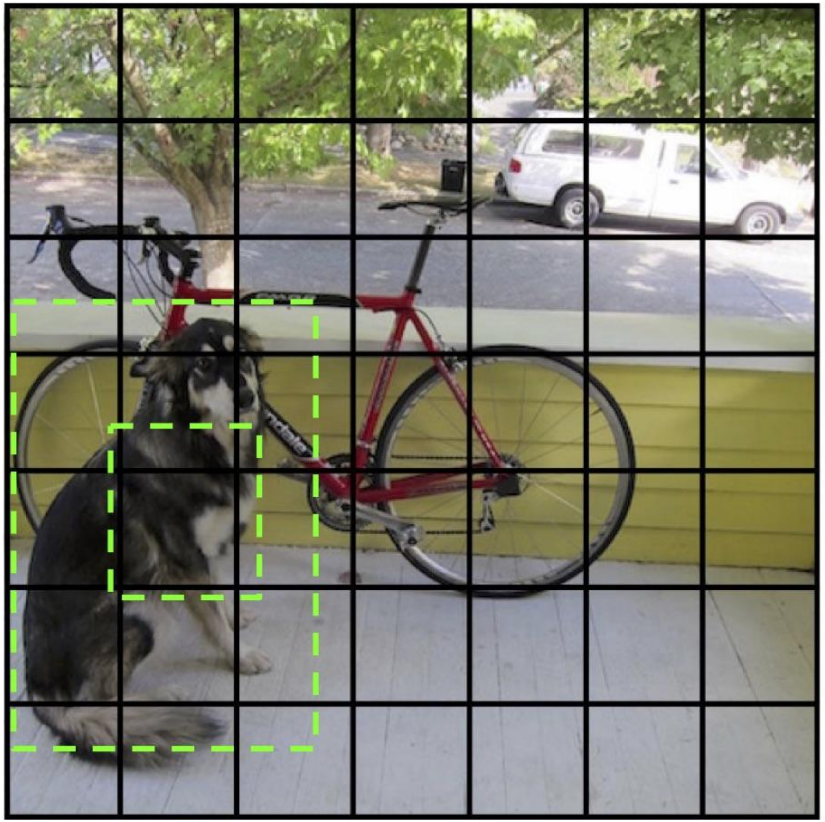
Adjust that cell's class prediction



Dog = 1
Cat = 0
Bike = 0
...

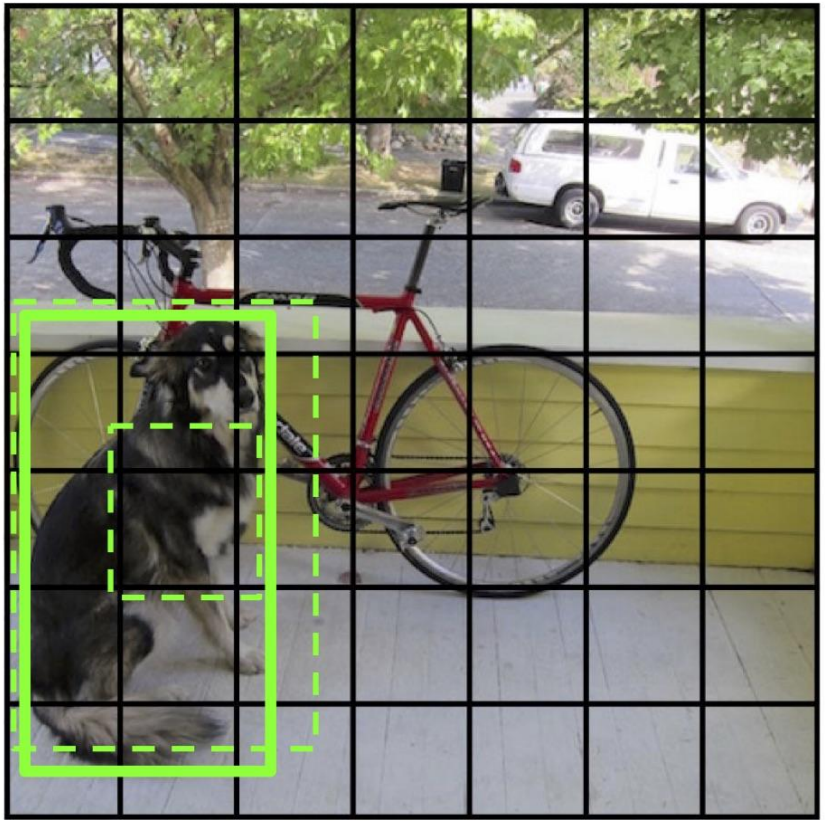
Original Slides from:
https://pjreddie.com/darknet/yolo/

Look at that cell's predicted boxes



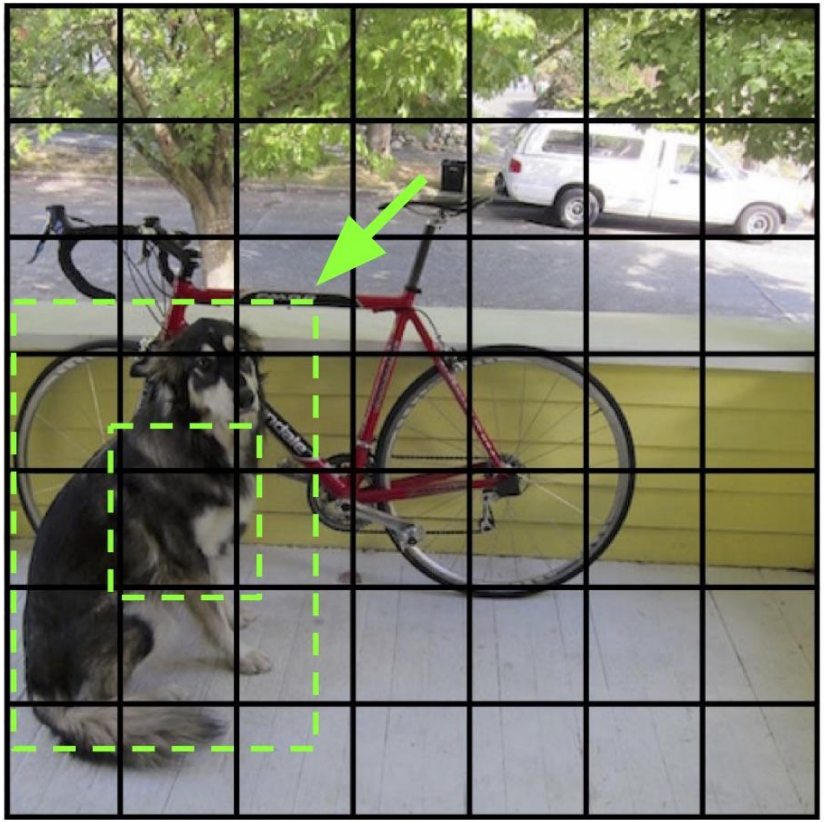
Original Slides from:
https://pjreddie.com/darknet/yolo/

Find the best one, make it “responsible for that prediction



Original Slides from:
https://pjreddie.com/darknet/yolo/

Find the best one, make it “responsible for that prediction



Original Slides from:
https://pjreddie.com/darknet/yolo/

loss function:

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \qquad (3)$$

Strengths and Weaknesses (compared to the R-CNN family)

Strengths

- Fast
- Generalizes better (i.e. good at detecting objects in paintings)
- Less background false positives

Weaknesses

- Localization not as accurate
- Limited to B objects per grid (B=2 in paper/this presentation)

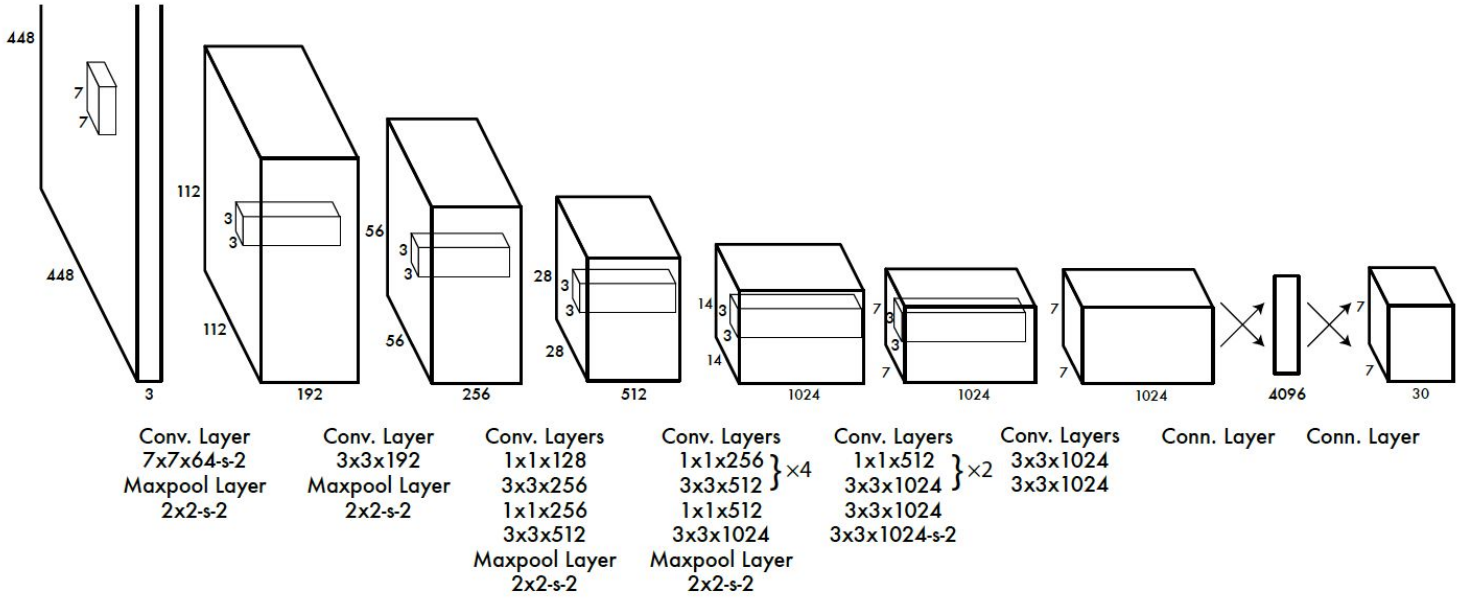


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.