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 detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

1. Introduction

Humans glance at an image and instantly know what objects are in the image, where they are, and how they interact. The human visual system is fast and accurate, allowing us to perform complex tasks like driving with little conscious thought. Fast, accurate algorithms for object detection would allow computers to drive cars without specialized sensors, enable assistive devices to convey real-time scene information to human users, and unlock the potential for general purpose, responsive robotic systems.

Current detection systems repurpose classifiers to perform detection. To detect an object, these systems take a

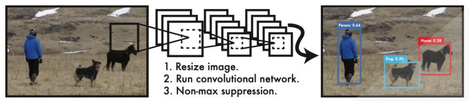


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

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.

We reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Using our system, you only look once (YOLO) at an image to predict what objects are present and where they are.

YOLO is refreshingly simple: see Figure 1. A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection.

First, YOLO is extremely fast. Since we frame detection as a regression problem we don't need a complex pipeline. We simply run our neural network on a new image at test time to predict detections. Our base network runs at 45 frames per second with no batch processing on a Titan X GPU and a fast version runs at more than 150 fps. This means we can process streaming video in real-time with

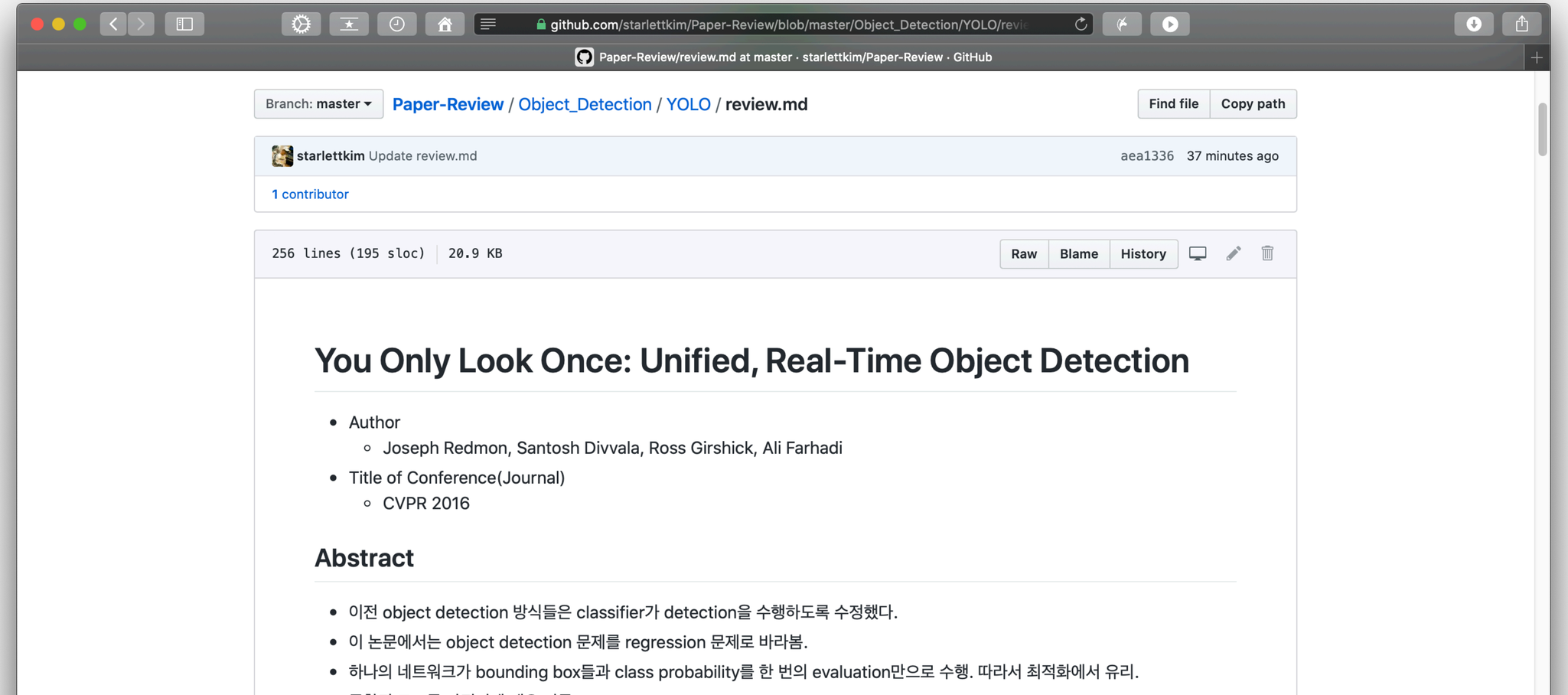
You Only Look Once

Unified, Real-Time Object Detection

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016).
You only look once: Unified, real-time object detection.
In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

CVLab @ Hanyang Univ.
Paper Review 16 Jan 2020.
Jihun Kim

Detailed review available on my Github 🥳

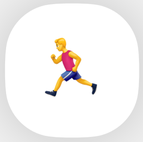


You Only Look Once

Unified, Real-Time Object Detection

Introduction

Unified Detection



Training

Experiments

Conclusion

You Only Look Once

Unified, Real-Time Object Detection

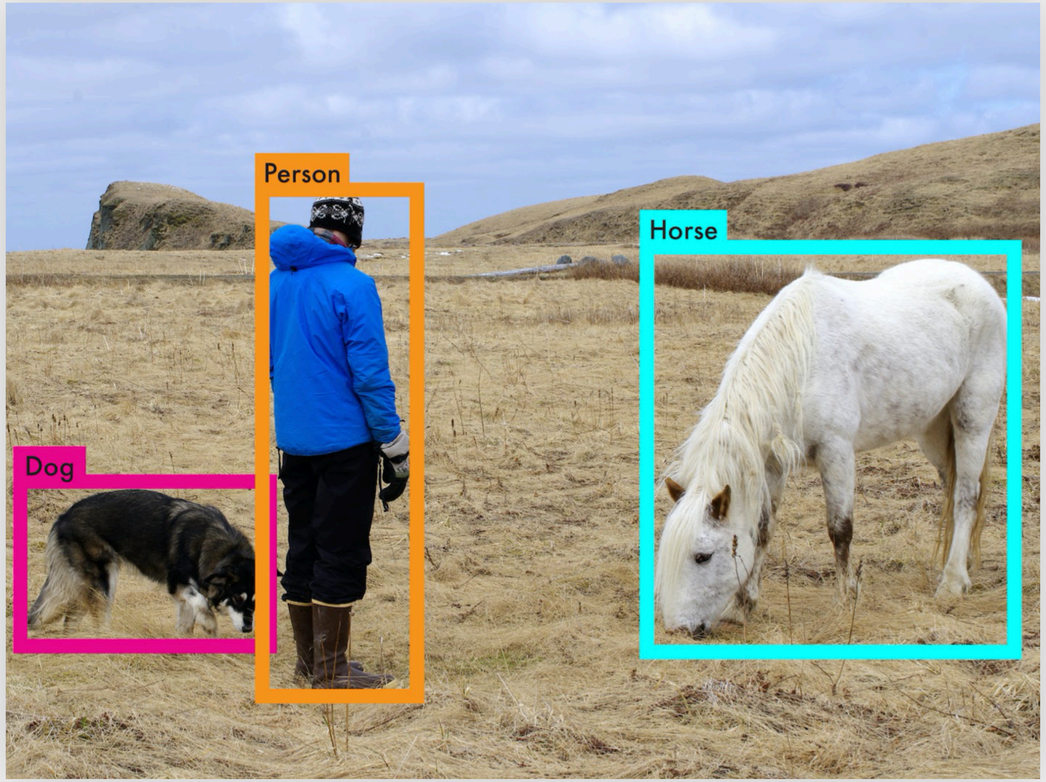
Introduction 👀

You only look once (YOLO) is a state-of-the-art, real-time object detection system.

- CVPR 2016, OpenCV People's Choice Award

Three versions have been announced so far.

- YOLO, YOLO9000, YOLOv3.



Introduction 👀

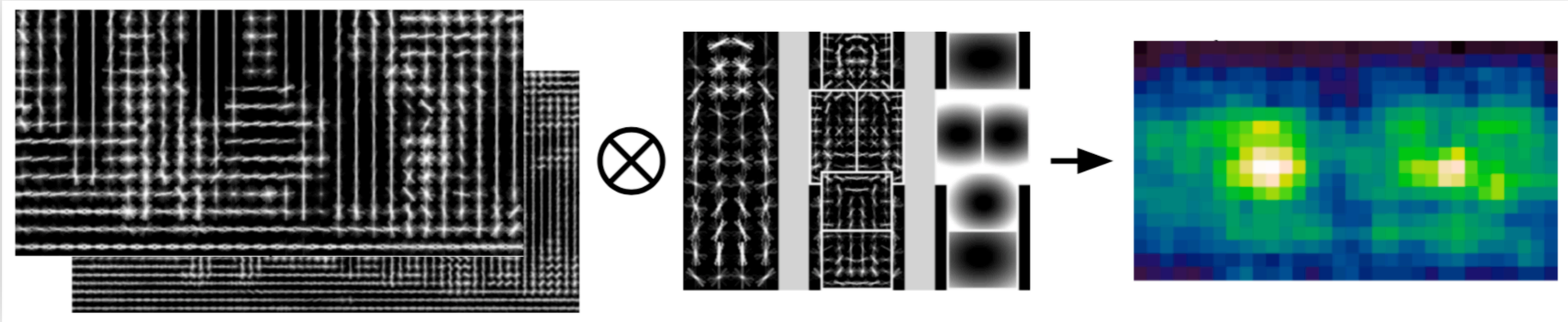
Accurate object detection is slow!

	Pascal 2007 mAP	Speed	
DPM v5	33.7	0.07 fps	14s/img
R-CNN	66.0	0.05 fps	20s/img
Fast R-CNN	70.0	0.5 fps	2s/img
Faster R-CNN	**73.2**	7 fps	140ms/img
YOLO	63.4 (→69.0)	**45 fps**	**22ms/img**

Introduction 👀

Current detection systems repurpose classifiers to perform detection.

- These systems take a classifier for that object and evaluate it at various locations and scales.

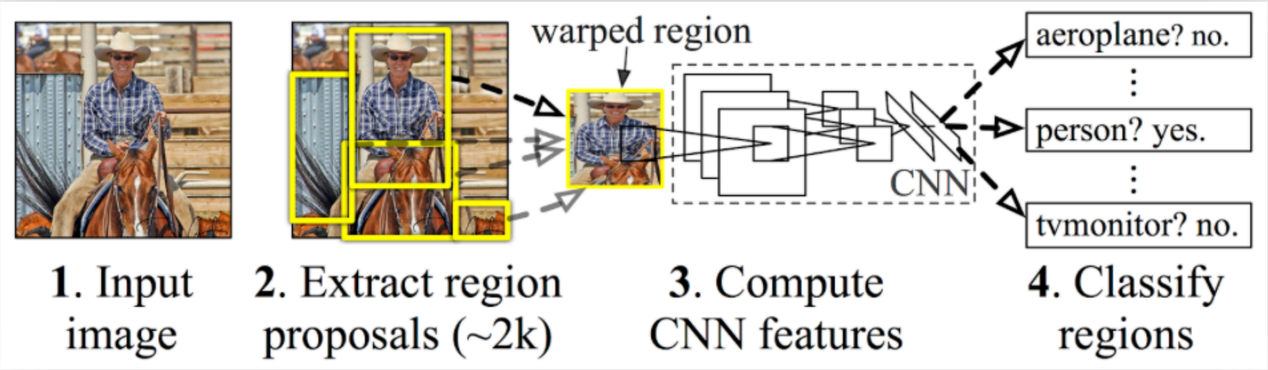


DPM: Deformable Part Models.

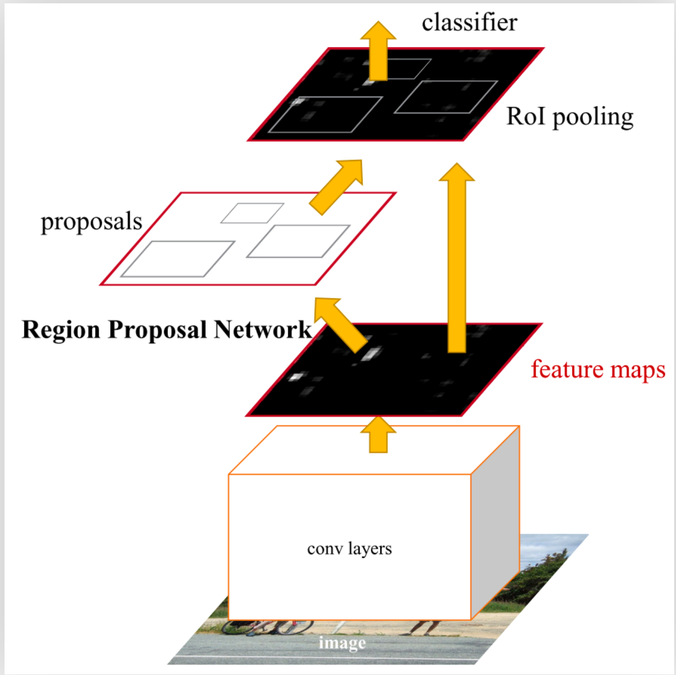
Introduction 👀

More recent approaches like R-CNN use complex piplines.

- first generate potential bounding boxes in an image and then run a classifier on these proposed boxes.
- They are slow and hard to optimize.
- Because each individual component must be trained separately.



R-CNN.

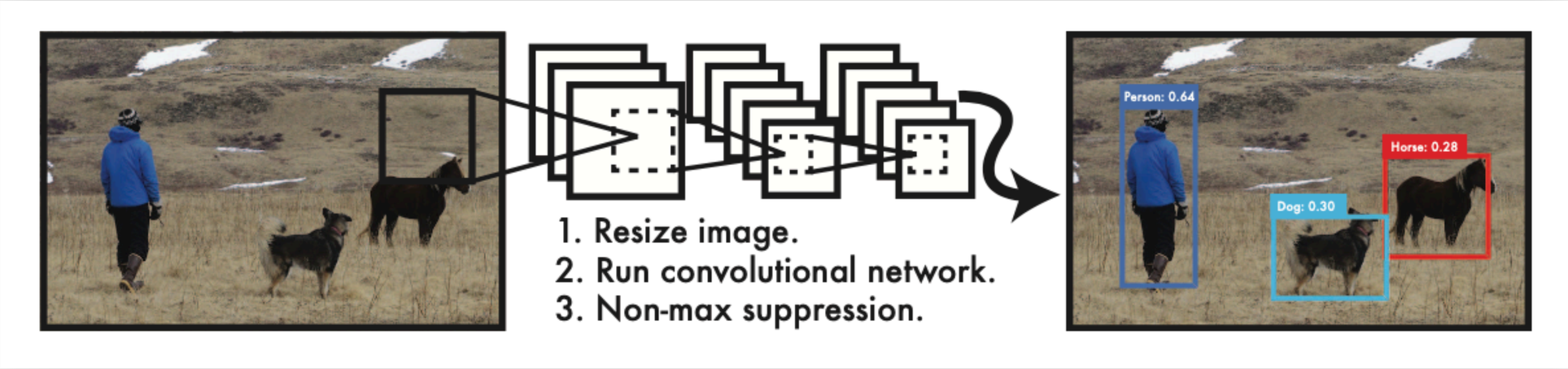


Faster R-CNN.

Introduction 👀

We reframe object detection as a single regression problem,
straight from image pixels to bounding box coordinates and class probabilities.

- Using our system, you only look once (YOLO) at an image.
- This unified model has several benefits over traditional methods of object detection.



YOLO (You Only Look Once).

Introduction 👀

First, YOLO is extremely fast.

- Since we frame detection as a regression problem we don't need a complex pipeline.
- Our base network runs at 45 fps and a fast version runs at more than 150 fps.
- Furthermore, YOLO achieves more than twice the mAP of other real-time systems.

Second, YOLO reasons globally about the image when making predictions.

- Unlike sliding window and region proposal-based techniques,
  YOLO sees the entire image so it implicitly encodes contextual information.

Third, YOLO learns generalizable representations of objects.

- Since YOLO is highly generalizable it is less likely to break down
  when applied to new domains or unexpected inputs.

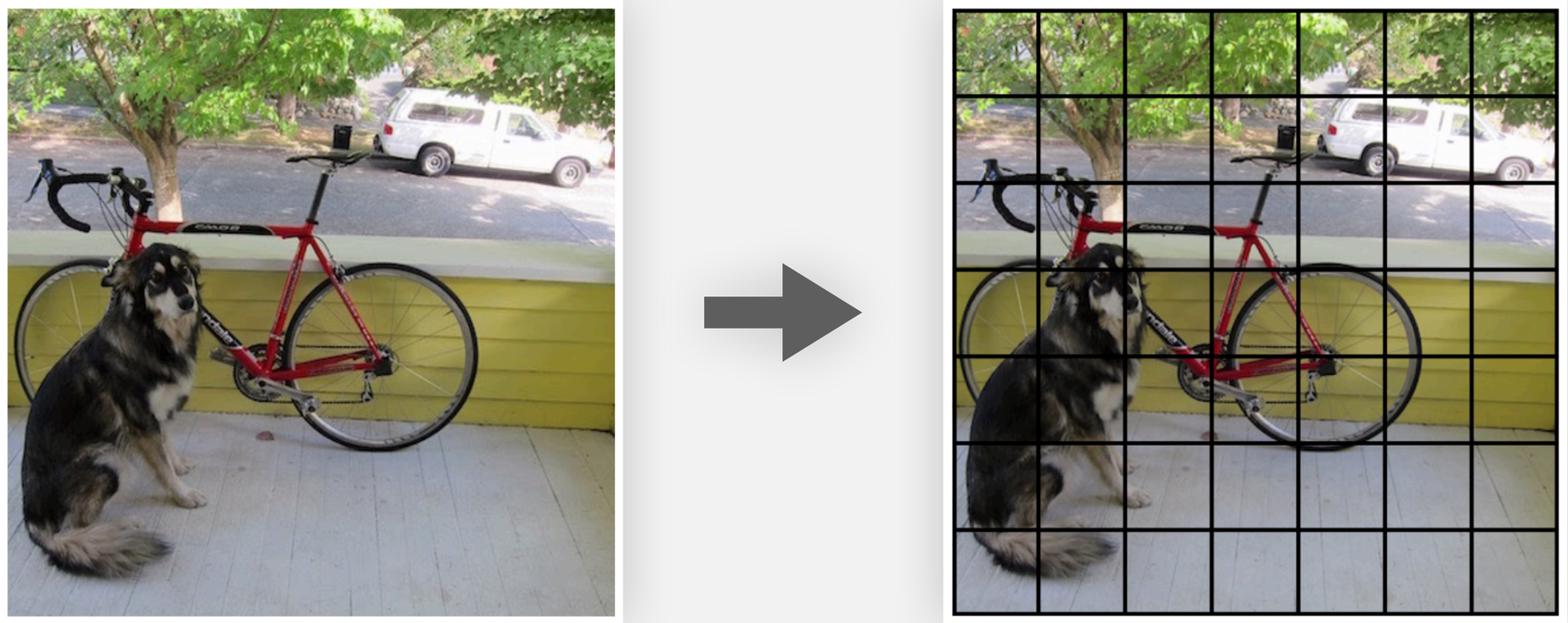
You Only Look Once

Unified, Real-Time Object Detection

Unified Detection 🔗

Our system divides the input image into an $S \times S$ grid.

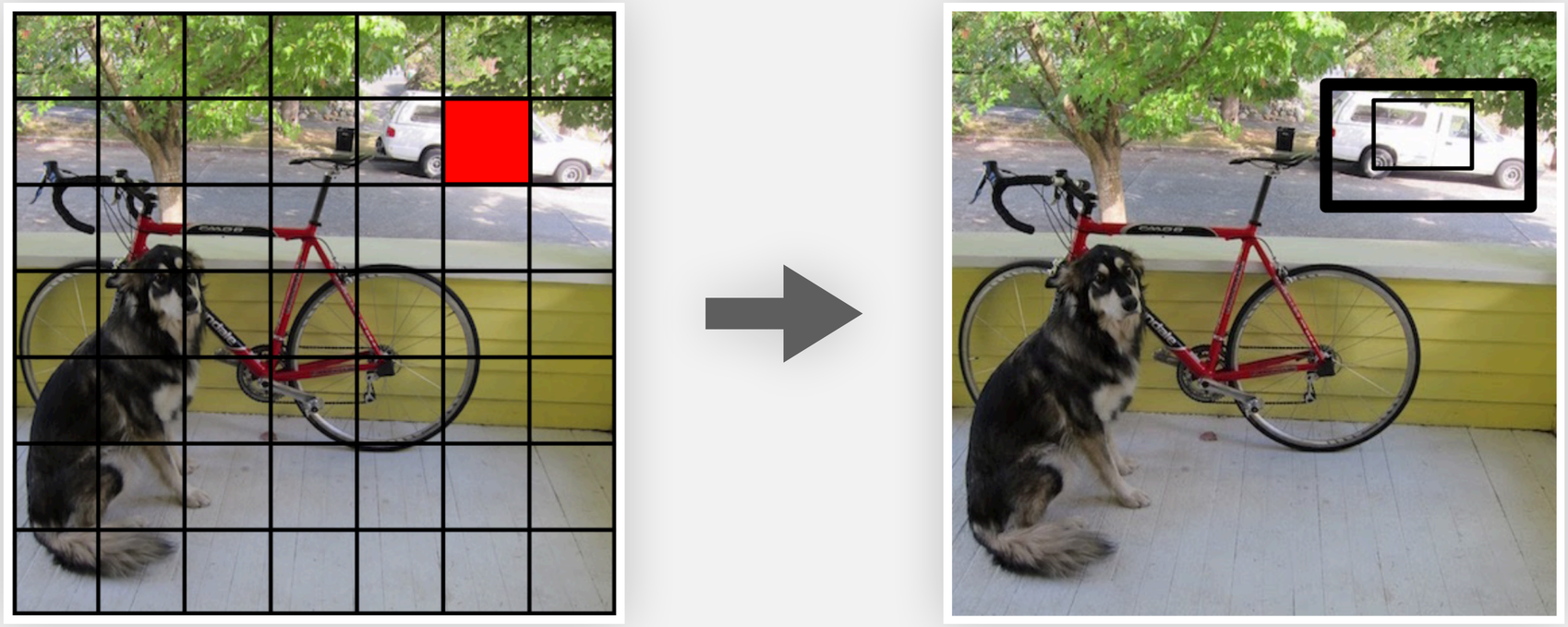
- If the center of an object falls into a grid cell,
  that grid cell is responsible for detecting that object.



Unified Detection 🔗

Each grid cell predicts B bounding boxes and confidence scores for those boxes.

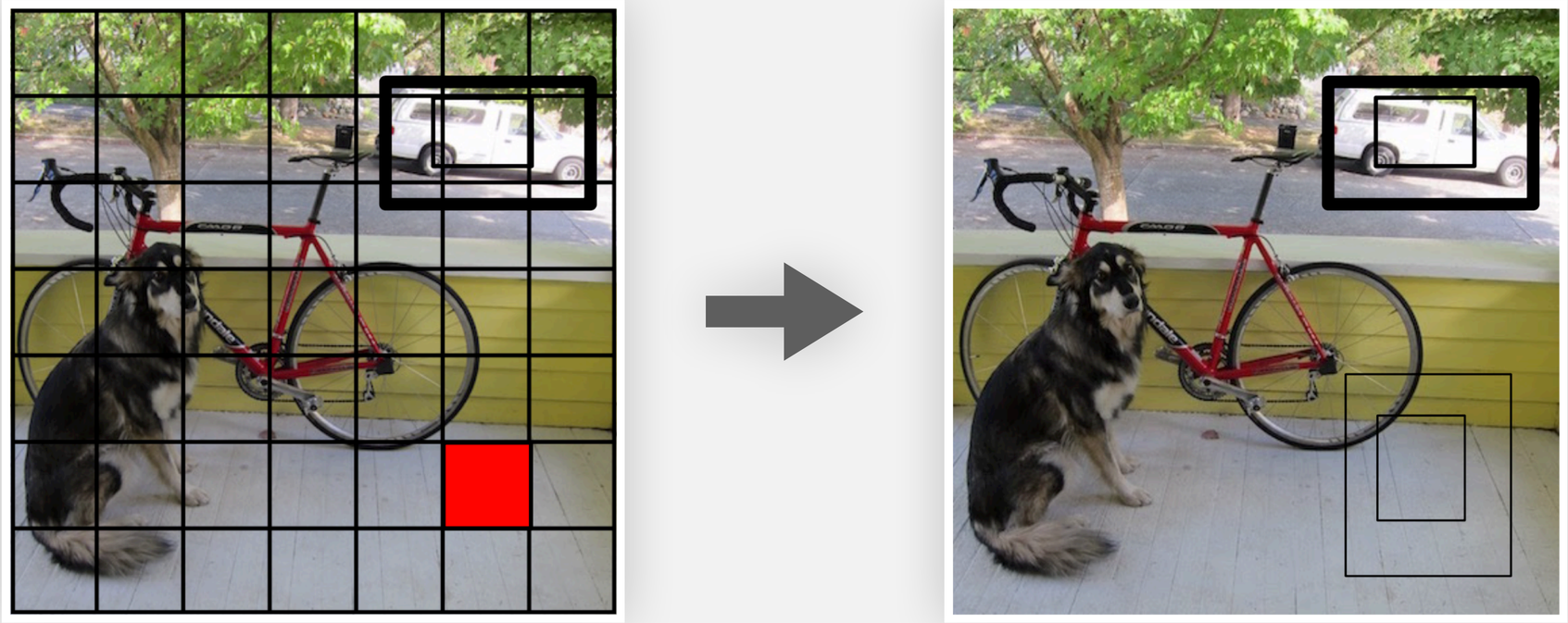
- These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts.
- Formally we define confidence as $Pr(Object) * IOU_{pred}^{truth}$.



Unified Detection 🔗

Each grid cell predicts B bounding boxes and confidence scores for those boxes.

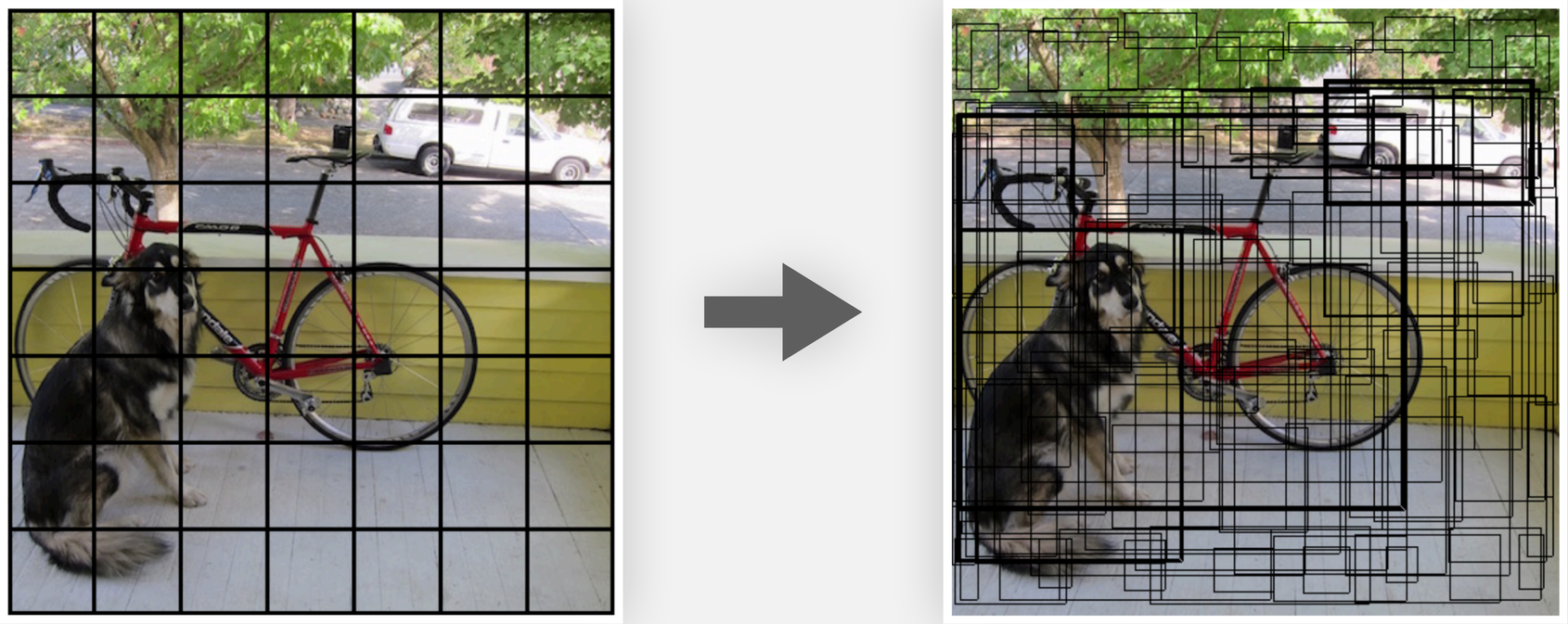
- These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts.
- Formally we define confidence as $Pr(Object) * IOU_{pred}^{truth}$.



Unified Detection 🔗

Each grid cell predicts B bounding boxes and confidence scores for those boxes.

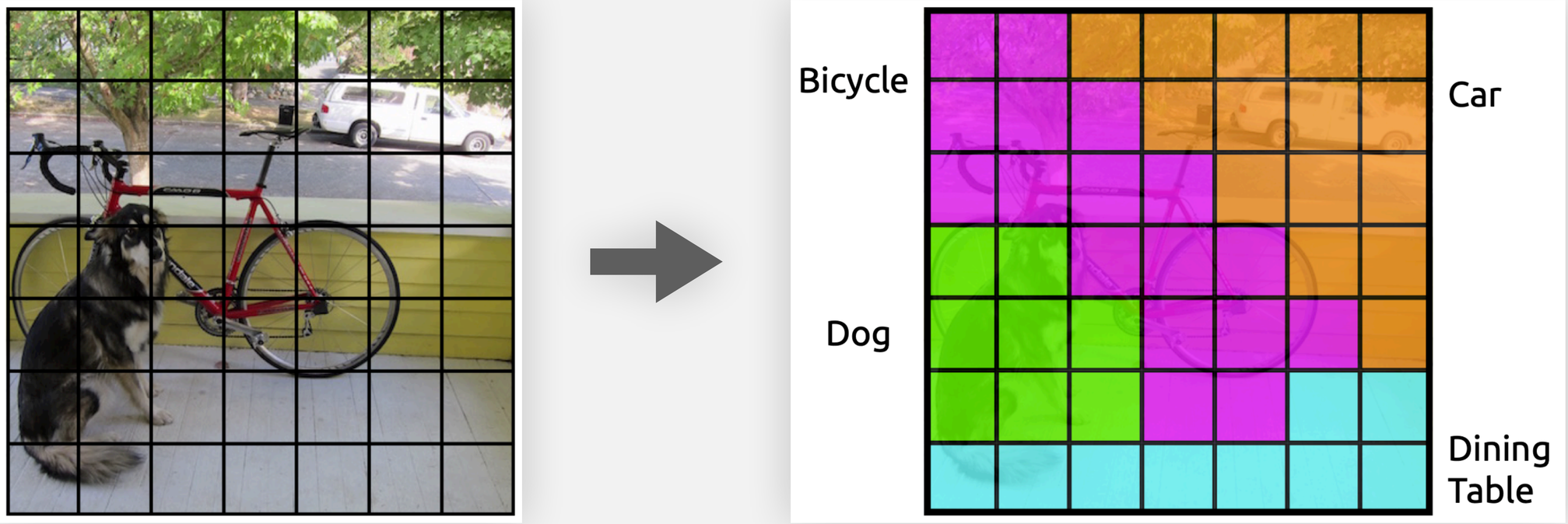
- These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts.
- Formally we define confidence as $Pr(Object) * IOU_{pred}^{truth}$.



Unified Detection 🔗

Each grid cell also predicts C conditional class probabilities, $Pr(Class_i | Object)$.

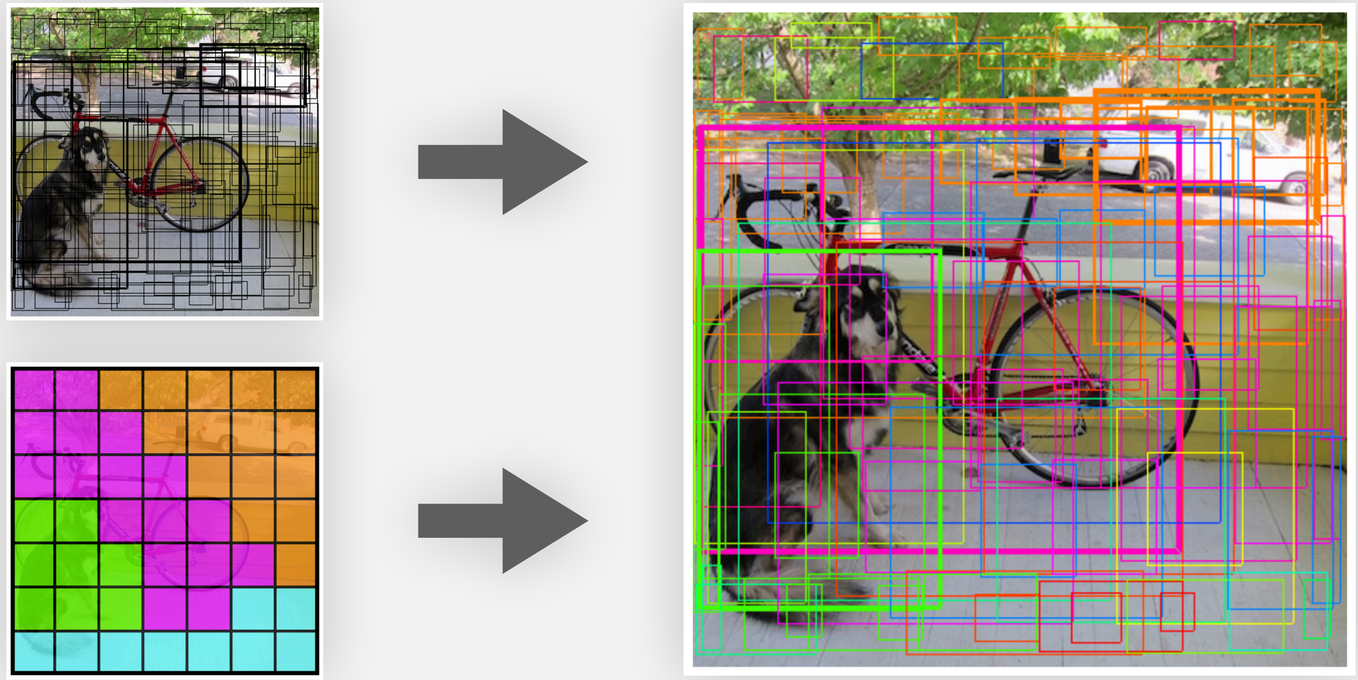
- These probabilities are conditioned on the grid cell containing an object.
- We only predict one set of class probabilities per grid cell, regardless of the number of boxes B.



Unified Detection 🔗

We multiply the conditional class probabilities and the individual box confidence, which gives us class-specific confidence scores for each box.

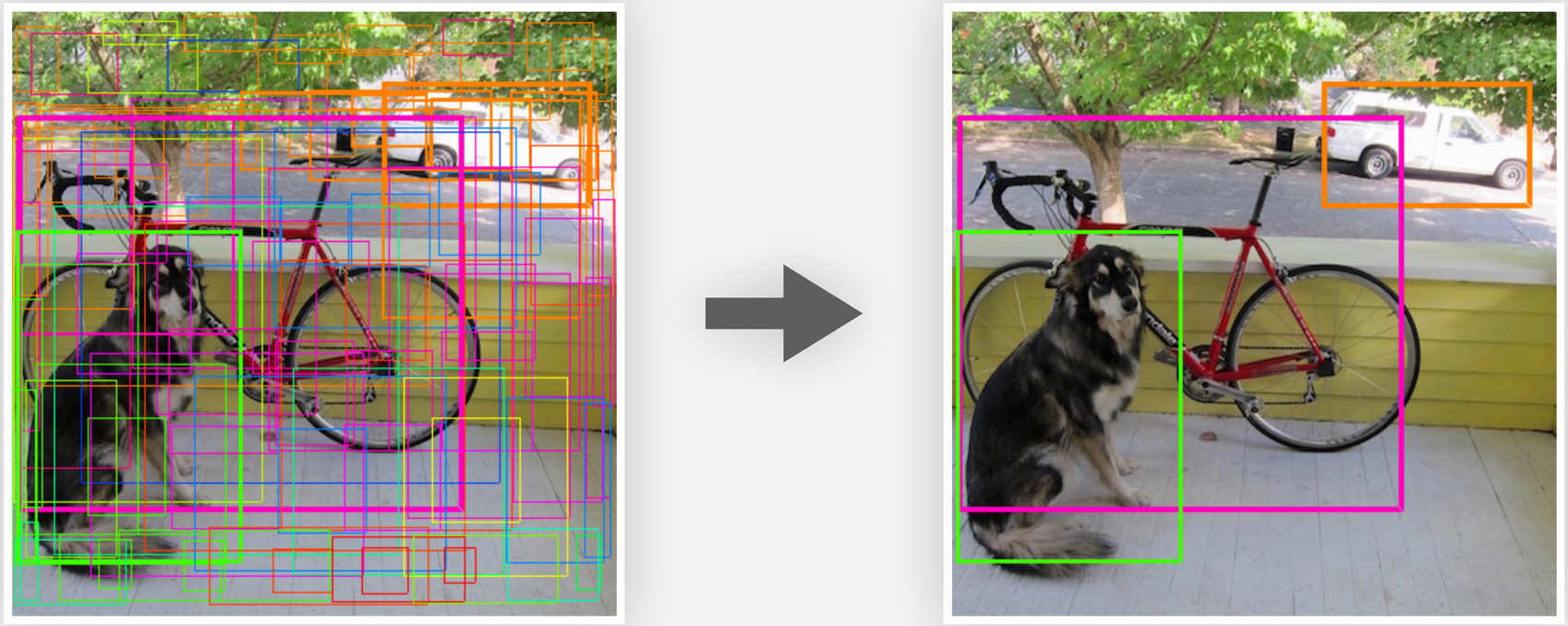
- Formally, $Pr(Class_i|Object) * Pr(Object) * IOU_{pred}^{truth} = Pr(Class_i) * IOU_{pred}^{truth}$.



Unified Detection 🔗

Finally we do Non-Maximal Suppression (NMS) and threshold detection.

- Non-maximal suppression adds 2-3% in mAP.



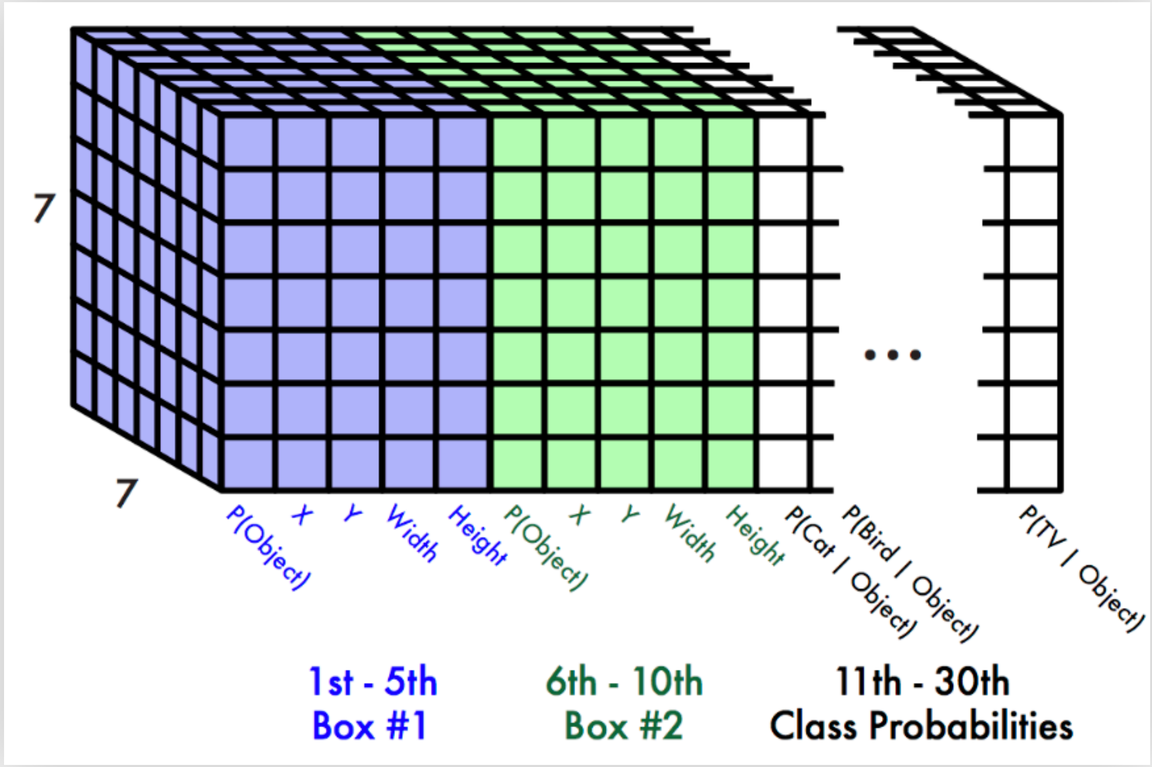
Network Design 🖌️

This parameterization fixes the output size.

- For each bounding box, 4 coordinates and 1 confidence value.

For Pascal VOC:

- 7×7 grid
- 2 bounding boxes per cell
- 20 classes
- $7 \times 7 \times (2 \times 5 + 20) = 7 \times 7 \times 30$ tensor

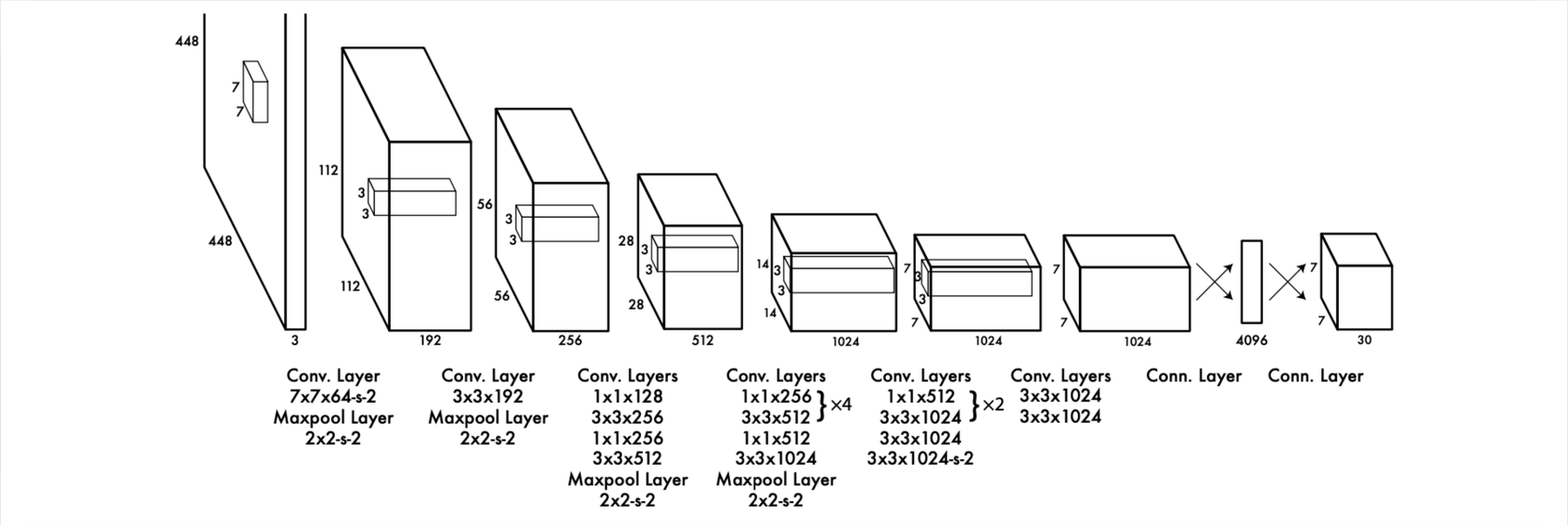


Network Design 🖌️

The initial convolutional layers extract features from the image
while the fully connected layers predict the output probabilities and coordinates.

We also train a fast version of YOLO.

- Fast YOLO uses a neural network with fewer convolutional layers and fewer filters in those layers.



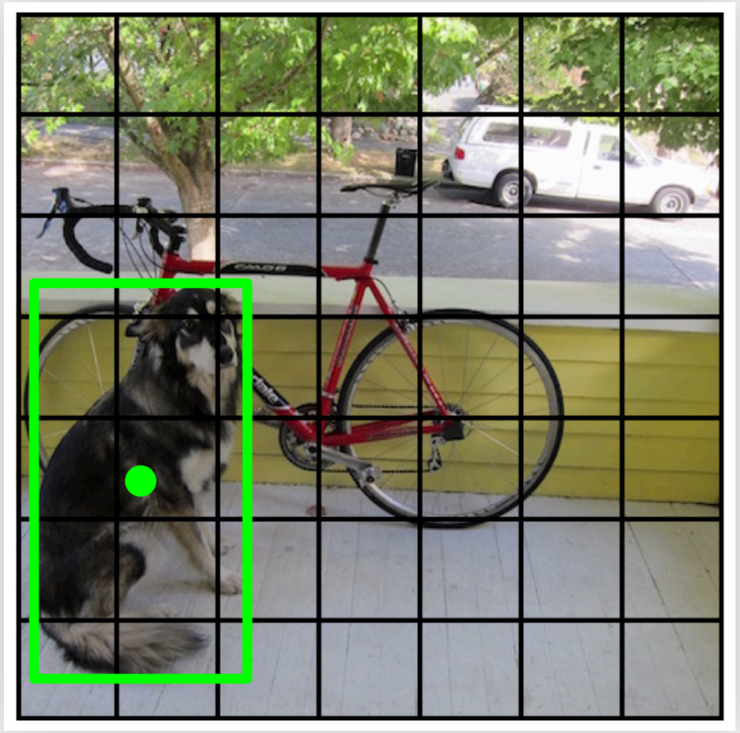
You Only Look Once

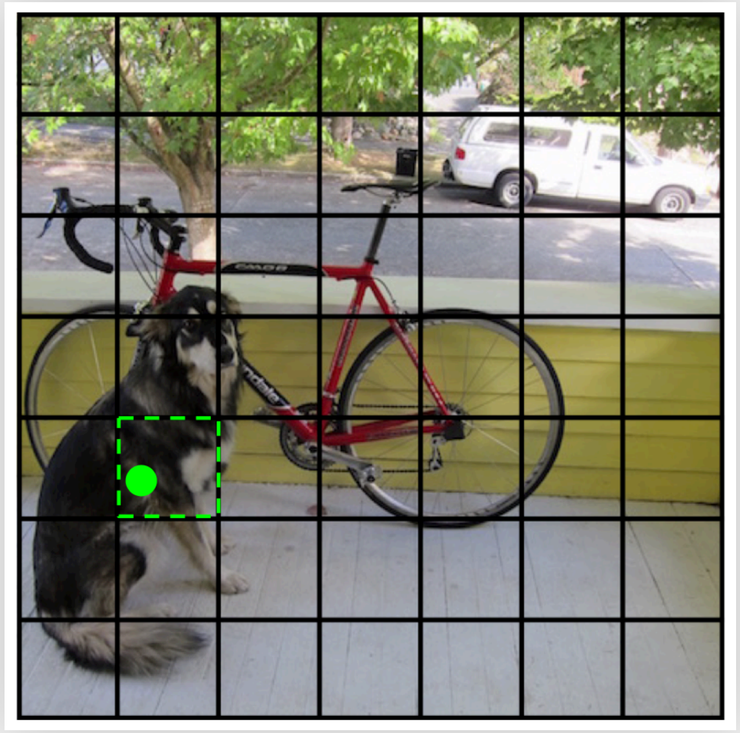
Unified, Real-Time Object Detection

Training 🏃

During training, match example to the right cell.

- Recall:
  If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object.

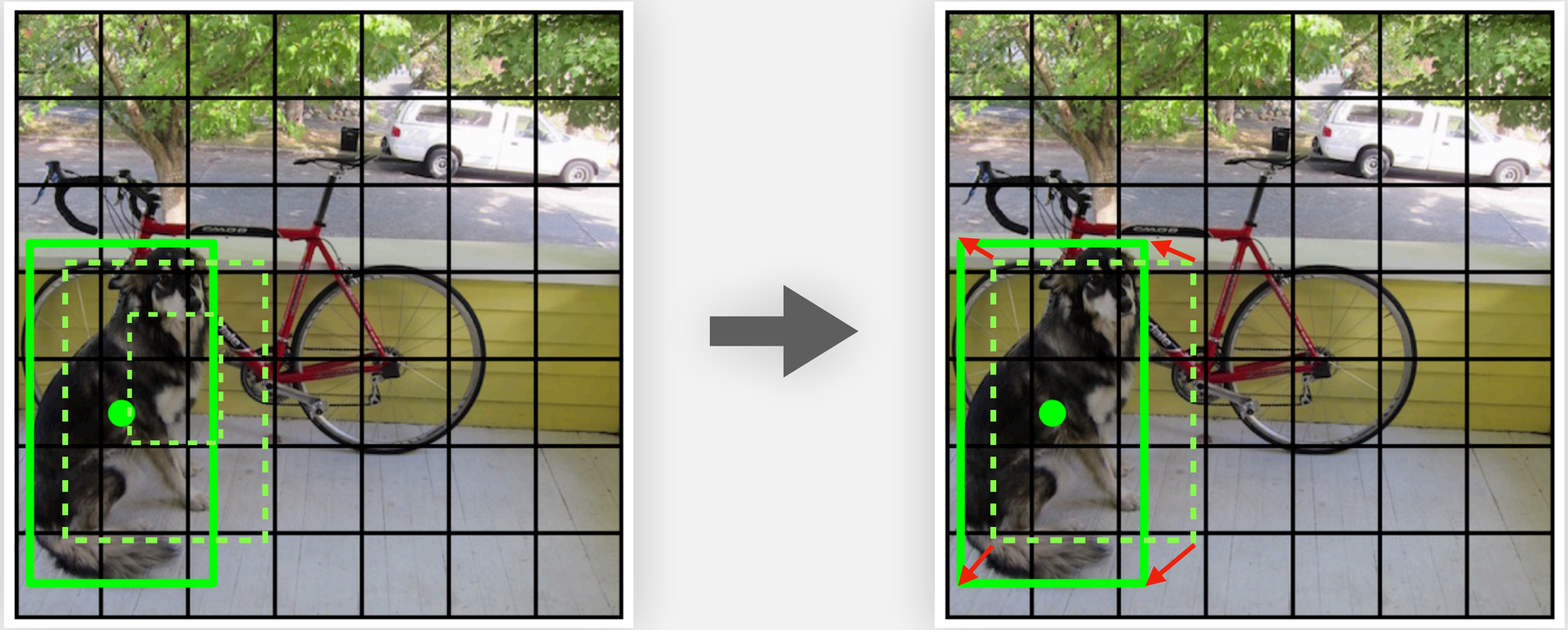




Coordniate Error

Assign one predictor to be “responsible” for predicting an object.

- Based on which prediction has the highest current IOU with the ground truth.
- This leads to specialization between the bounding box predictors.



Coordniate Error

Sum-squared error equally weights errors in large boxes and small boxes.

- Error metric should reflect that small deviations in large boxes matter less than in small boxes.
- To partially address this we predict the square root of the bounding box width and height instead of the width and height directly.

Coordinate part of the loss function.

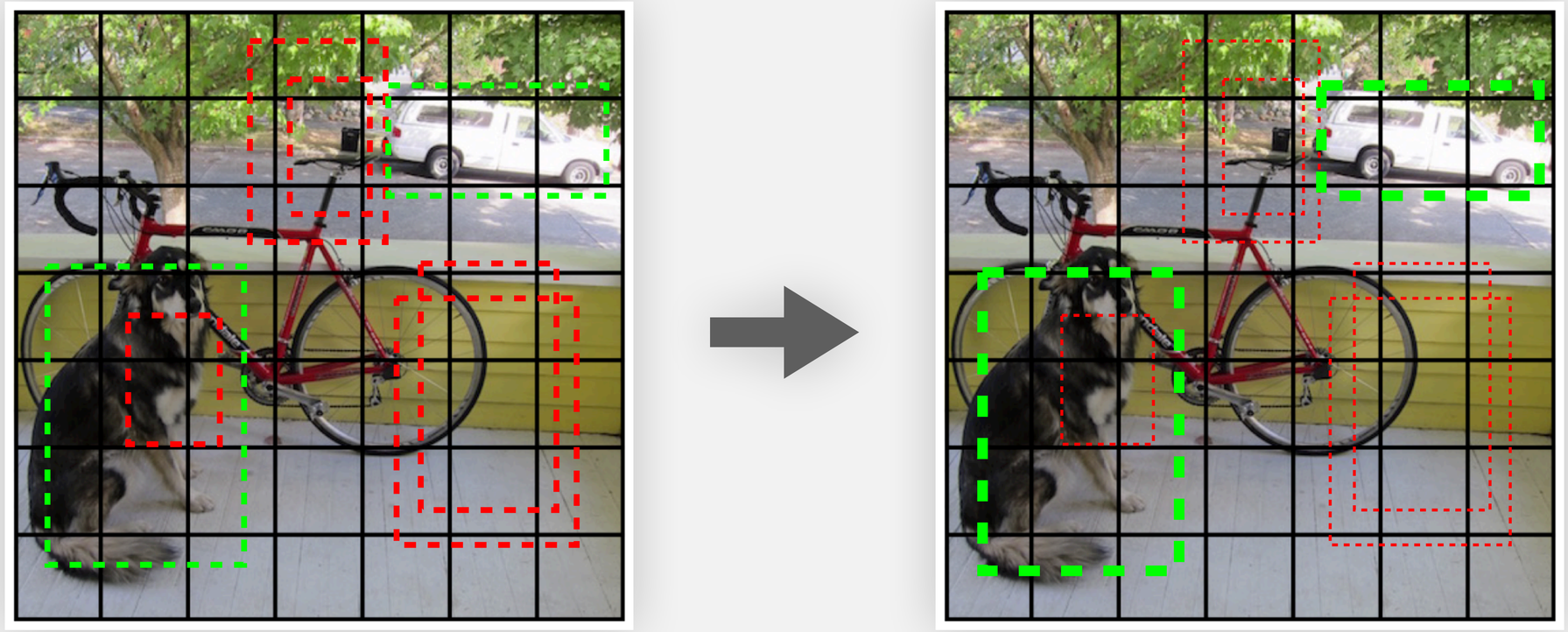
$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

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

Denotes the jth bounding box predictor in cell i is “responsible” for that prediction.

Confidence Error

Increase the confidence of predictors which are “responsible” for predicting objects, and decrease it of predictors which are not responsible.



Confidence Error

Increase the confidence of predictors which are “responsible” for predicting objects, and decrease it of predictors which are not responsible.

Confidence part of the loss function.

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

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

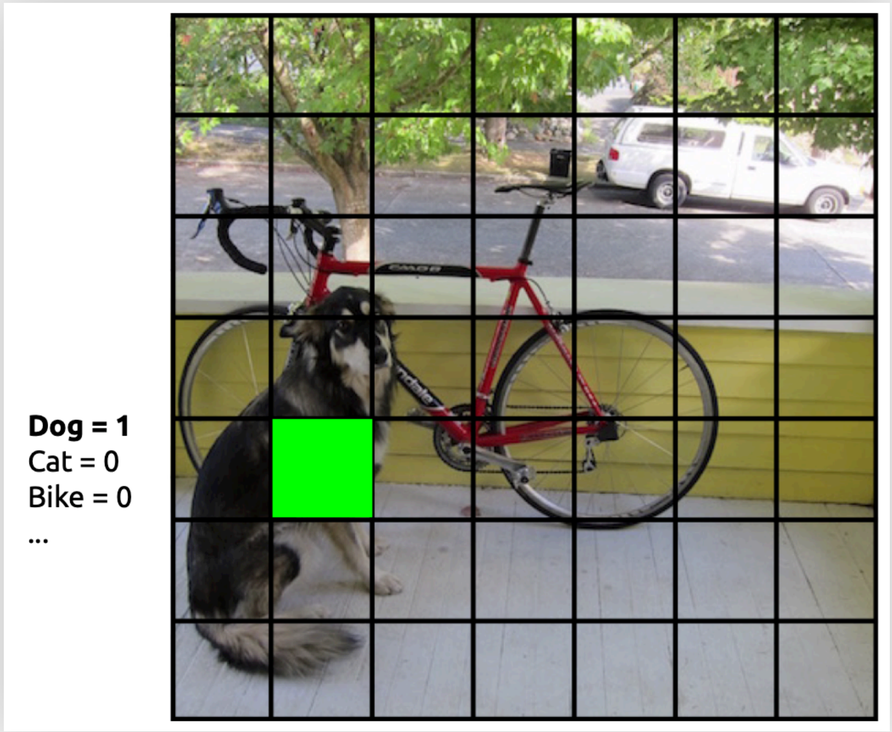
Classification Error

Loss function only penalizes classification error if an object is present in that grid cell.

Classification part of the loss function.

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

Denotes if object appears in cell i.



Loss Function

During training we optimize the following, multi-part loss function.

- We use sum-squared error, because it is easy to optimize.

$$
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2
\end{aligned}
$$

Loss Function

However it does not perfectly align with our goal of maximizing average precision.

- It weights localization error equally with classification error which may not be ideal.

Many grid cells do not contain any object.

- This pushes the “confidence” scores of those cells towards zero, often overpowering the gradient from cells that do contain objects.
- This can lead to model instability, causing training to diverge early on.

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

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

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

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

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

Loss Function

To remedy this, we increase the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don't contain objects.

- We use two parameters, λ_{coord} and λ_{noobj} to accomplish this.
- We set $\lambda_{coord} = 5$ and $\lambda_{noobj} = 0.5$.

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2
\end{aligned}
$$

Details

We use a batch size = 64, a momentum = 0.9 and a decay = 0.0005.

To avoid overfitting we use dropout and extensive data augmentation.

- A dropout layer with rate = 0.5 after the first fc layer prevents co-adaptation between layers.
- For data augmentation we introduce random scaling and translations of up to 20% of the original image size.
- We also randomly adjust the exposure and saturation of the image by up to a factor of 1.5 in the HSV color space.

You Only Look Once

Unified, Real-Time Object Detection

Comparison to Other Real-Time Systems

Many research efforts in object detection focus on making standard detection pipelines fast.

- However, only 30Hz/100Hz DPM actually produce a detection system that runs in real-time.

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Real-time systems on Pascal VOC 2007.

Experiments 🧪 >

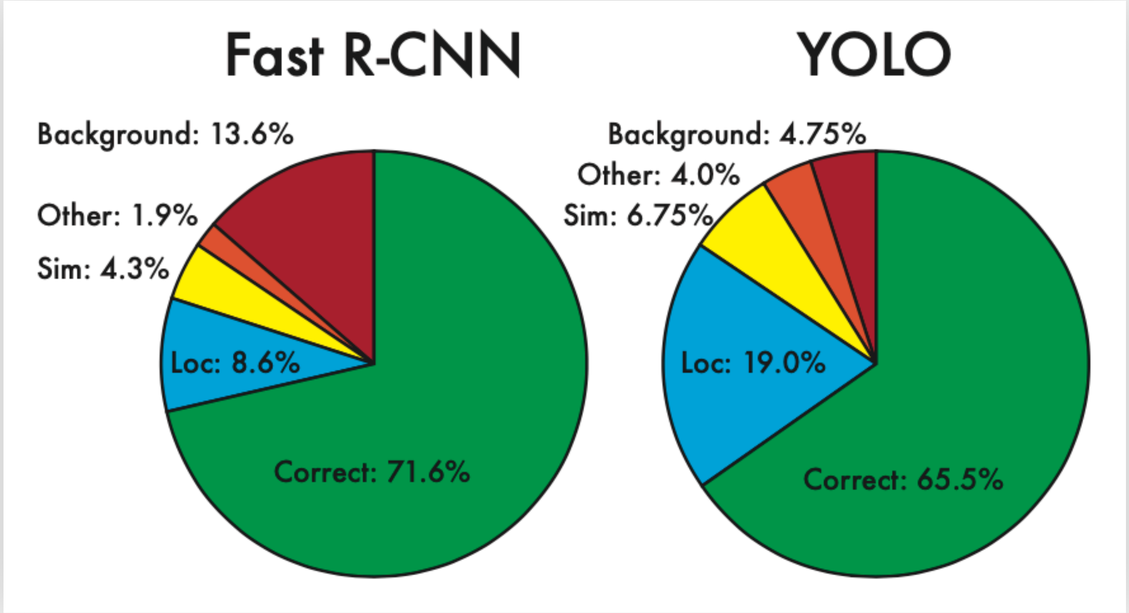
VOC 2007 Error Analysis

We compare YOLO to Fast R-CNN.

- Since it's one of the highest performing detectors on Pascal and it’s detections are publicly available.

YOLO struggles to localize objects correctly.

- Fast R-CNN makes much fewer localization errors but far more background errors.
- It is almost 3x more likely to predict background detections than YOLO.



Combining Fast R-CNN and YOLO

By using YOLO to eliminate background detections from Fast R-CNN
we get a significant boost in performance.

- For every bounding box that R-CNN predicts
  we check to see if YOLO predicts a similar box.

When combined with YOLO,
its mAP increases by 3.2% to 75.0%.

- YOLO makes different kinds of mistakes at test time
  that it is so effective at boosting Fast R-CNN's performance.
- Since YOLO is so fast it doesn't add any
  significant computational time.

VOC 2012 test	mAP	aero	bik
MR_CNN_MORE_DATA [11]	**73.9**	**85.5**	**82.**
HyperNet_VGG	71.4	84.2	78.
HyperNet_SP	71.3	84.1	78.
Fast R-CNN + YOLO	70.7	83.4	78.
MR_CNN_S_CNN [11]	70.7	85.0	79.
Faster R-CNN [27]	70.4	84.9	79.
DEEP_ENS_COCO	70.1	84.0	79.
NoC [28]	68.8	82.8	79.
Fast R-CNN [14]	68.4	82.3	78.
UMICH_FGS_STRUCT	66.4	82.9	76.
NUS_NIN_C2000 [7]	63.8	80.2	73.
BabyLearning [7]	63.2	78.0	74.
NUS_NIN	62.4	77.9	73.
R-CNN VGG BB [13]	62.4	79.6	72.
R-CNN VGG [13]	59.2	76.8	70.
YOLO	57.9	77.0	67.

Pascal VOC 2012 leaderboard.

Experiments 🧪 >

VOC 2012 Results

Our system struggles with small objects compared to its closest competitors.

VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR_CNN_MORE_DATA [11]	**73.9**	**85.5**	**82.9**	**76.6**	**57.8**	**62.7**	**79.4**	77.2	86.6	**55.0**	**79.1**	**62.2**	87.0	**83.4**	**84.7**	78.9	45.3	73.4	65.8	80.3	74.0
HyperNet_VGG	71.4	84.2	78.5	73.6	55.6	53.7	78.7	**79.8**	87.7	49.6	74.9	52.1	86.0	81.7	83.3	**81.8**	**48.6**	**73.5**	59.4	79.9	65.7
HyperNet_SP	71.3	84.1	78.3	73.3	55.5	53.6	78.6	79.6	87.5	49.5	74.9	52.1	85.6	81.6	83.2	81.6	48.4	73.2	59.3	79.7	65.6
Fast R-CNN + YOLO	70.7	83.4	78.5	73.5	55.8	43.4	79.1	73.1	**89.4**	49.4	75.5	57.0	**87.5**	80.9	81.0	74.7	41.8	71.5	68.5	**82.1**	67.2
MR_CNN_S_CNN [11]	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Faster R-CNN [27]	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
DEEP_ENS_COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	**68.8**	75.9	71.4
NoC [28]	68.8	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1
Fast R-CNN [14]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	**87.5**	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
UMICH_FGS_STRUCT	66.4	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2
NUS_NIN_C2000 [7]	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
BabyLearning [7]	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6
NUS_NIN	62.4	77.9	73.1	62.6	39.5	43.3	69.1	66.4	78.9	39.1	68.1	50.0	77.2	71.3	76.1	64.7	38.4	66.9	56.2	66.9	62.7
R-CNN VGG BB [13]	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3
R-CNN VGG [13]	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	80.4	71.6	74.0	60.0	30.8	63.4	52.0	63.5	58.7
YOLO	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
Feature Edit [32]	56.3	74.6	69.1	54.4	39.1	33.1	65.2	62.7	69.7	30.8	56.0	44.6	70.0	64.4	71.1	60.2	33.3	61.3	46.4	61.7	57.8
R-CNN BB [13]	53.3	71.8	65.8	52.0	34.1	32.6	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.9	59.3	54.1
SDS [16]	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7
R-CNN [13]	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6

Pascal VOC 2012 leaderboard.

Generalizability: Person Detection in Artwork

In real-world applications it is hard to predict all possible use cases

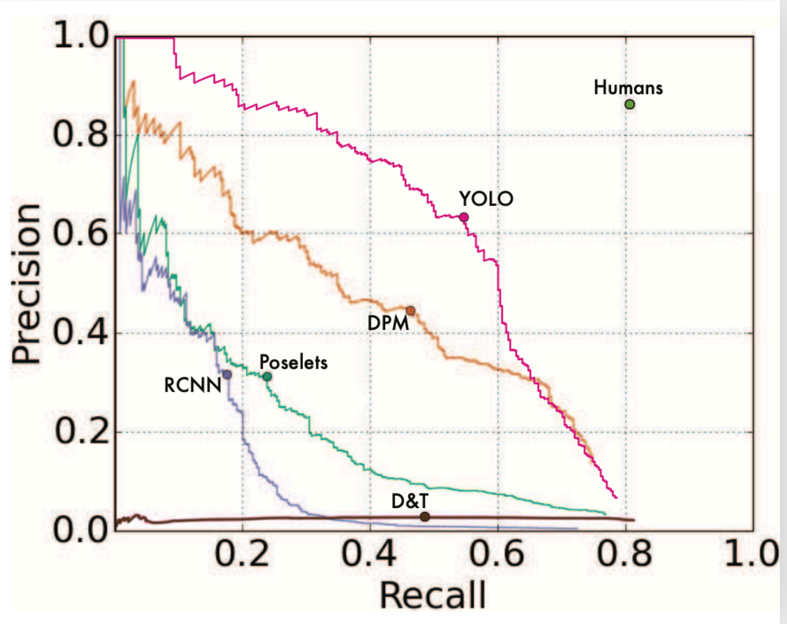
- Test data can diverge from what the system has seen before.

YOLO's AP degrades less than other methods when applied to artwork.

- Like DPM, YOLO models the size and shape of objects, as well as relationships between objects and where objects commonly appear.

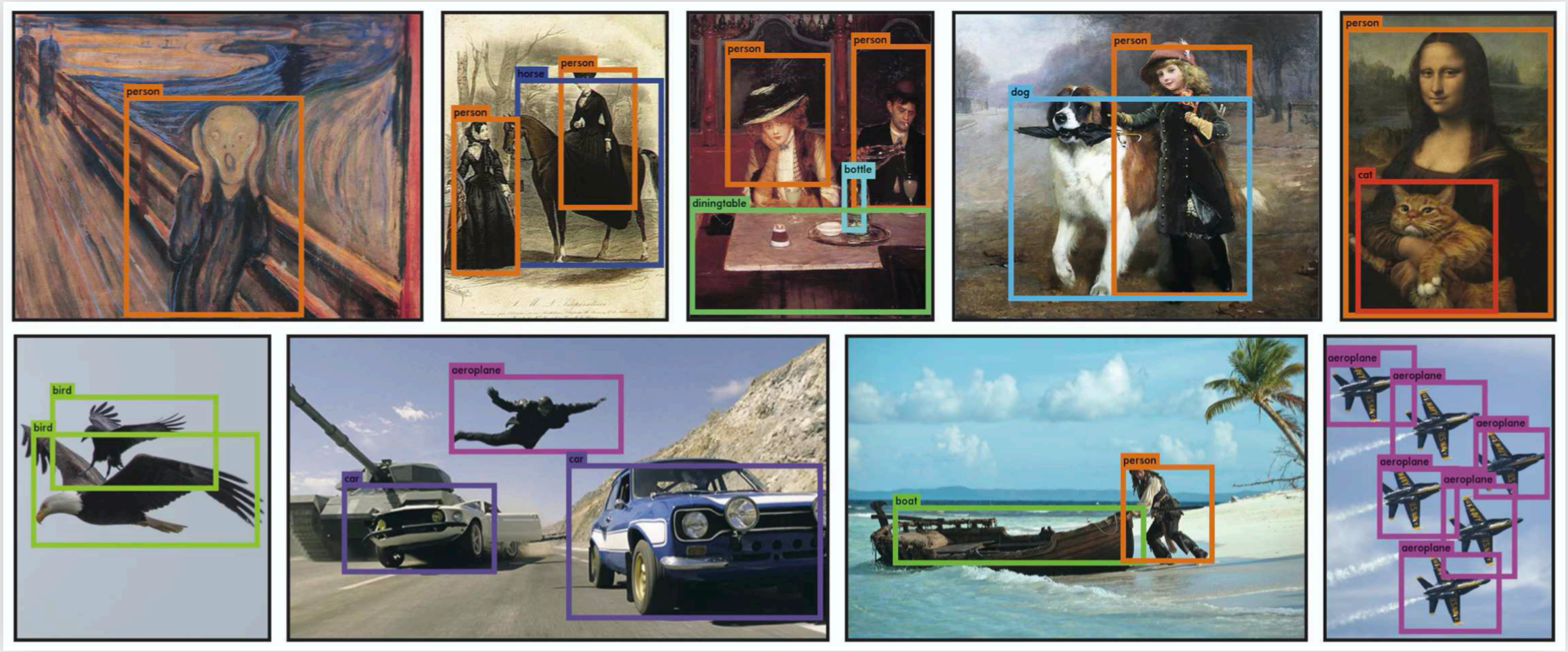
	VOC 2007	Picasso		People-Art
	AP	AP	Best F_1	AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32
Poselets [2]	36.5	17.8	0.271	
D&T [4]	-	1.9	0.051	

Quantitative results on the VOC 2007, Picasso, and People-Art Datasets.



Picasso Dataset precision-recall curves.

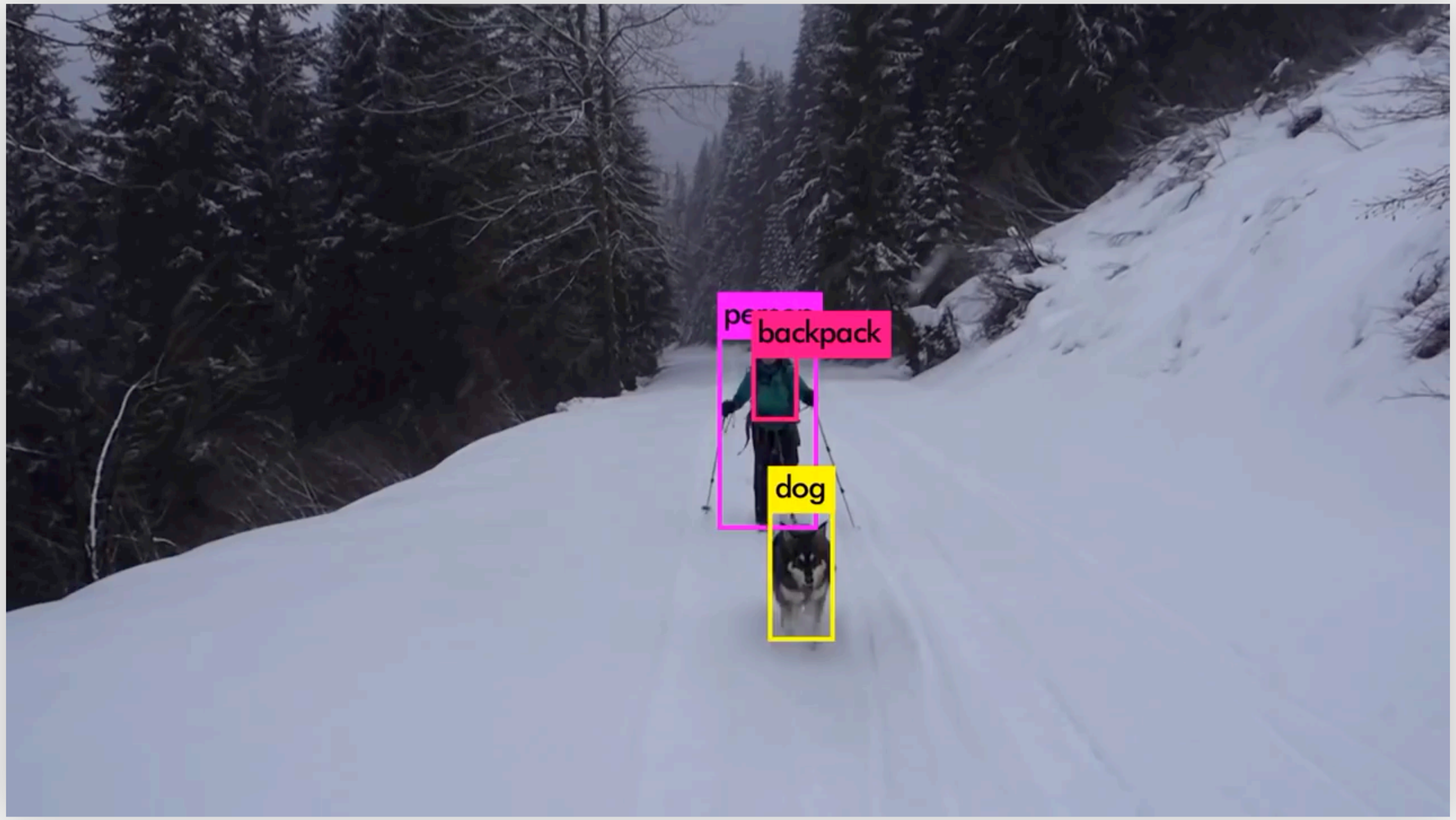
Qualitative Results



Qualitative results.

Experiments 🧪 >

Qualitative Results



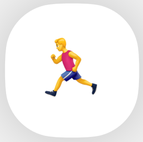
Qualitative results on their youtube. (YOLOv3)

You Only Look Once

Unified, Real-Time Object Detection

Introduction

Unified Detection



Training

Experiments

Conclusion

Conclusion ⭐

We introduce YOLO, a unified model for object detection.
Our model is simple to construct and can be trained directly on full images.

Fast YOLO is the fastest general-purpose object detector in the literature
and YOLO pushes the state-of-the-art in real-time object detection.

YOLO also generalizes well to new domains making it ideal for applications
that rely on fast, robust object detection.

Limitations of YOLO 🙅

YOLO imposes strong spatial constraints on bounding box predictions.

- Since each grid cell only predicts two boxes and can only have one class.
- This spatial constraint limits the number of nearby objects that our model can predict.

It struggles to generalize to objects in new or unusual aspect ratios or configurations.

- Since our model learns to predict bounding boxes from data.

Our loss function treats errors the same in
small bounding boxes versus large bounding boxes.

- A small error in a large box is generally benign but a small error in a small box has a much greater effect on IOU.
- Our main source of error is incorrect localizations.

References 

CVPR Paper
https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf

Talk
https://www.youtube.com/watch?v=NM6lrxy0bxs

Slides
https://docs.google.com/presentation/d/1kAa7NOamBt4calBU9iHgT8a86RRHz9Yz2oh4-GTdX6M/edit?usp=sharing

arXiv
http://arxiv.org/abs/1506.02640

Thank You for Listening 😆

CVLab @ Hanyang Univ.
Paper Review 16 Jan 2020.
Jihun Kim