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

Abstract

we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities
bounding boxes and class probabilities directly from full images in one evaluation
45 frames per second

Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors.

YOLO makes more localization errors but is less likely to predict false positives on background.

learns very general representations of objects

Introduction

Humans glance at an image and instantly know what objects are in the image, where they are, and how they interact.
fast and accurate

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

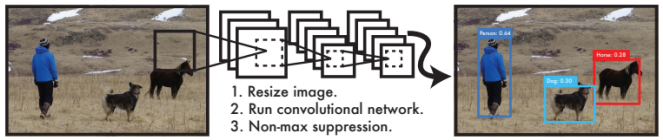


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448×448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

This unified model has several benefits over traditional methods of object detection.
1 YOLO is extremely fast.
we don't need a complex pipeline.
45 frames per second with no batch processing on a Titan X GPU and a fast version runs at more than 150 fps.

2 YOLO reasons globally about the image when making predictions
YOLO sees the entire image
Fast R-CNN, a top detection method [14], mistakes background patches in an image for objects because it can't see the larger context.

3 learns generalizable representations of objects.
less likely to break down when applied to new domains or unexpected inputs.

Unified Detection

uses features from the entire image to predict each bounding box

predicts all bounding boxes across all classes for an image simultaneously

divides the input image into an S × S grid
center of an object falls into a grid cell, that grid cell is responsible for detecting that object.

$$\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth

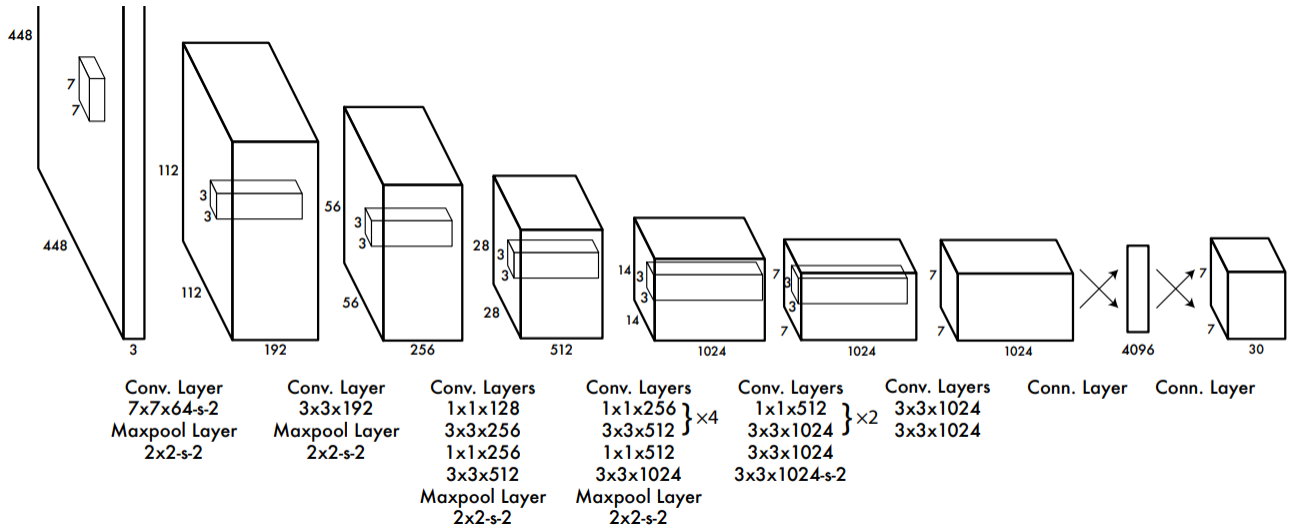
Each bounding box consists of 5 predictions: x, y, w, h, and confidence.

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

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

class-specific confidence scores for each box.

Network Design



The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates
Our network architecture is inspired by the GoogLeNet model for image classification

train a fast version of YOLO designed to push the boundaries of fast object detection.
Fast YOLO uses a neural network with fewer convolutional layers (9 instead of 24) and fewer filters in those layers.
all training and testing parameters are the same between YOLO and Fast YOLO.

Training

We pretrain our convolutional layers on the ImageNet 1000-class competition dataset
We then convert the model to perform detection.
adding both convolutional and connected layers to pretrained networks can improve performance

Detection often requires fine-grained visual information so we increase the input resolution of the network from 224 × 224 to 448 × 448

Our final layer predicts both class probabilities and bounding box coordinates
We normalize the bounding box width and height by the image width and height so that they fall between 0 and 1.
We parametrize the bounding box x and y coordinates to be offsets of a particular grid cell location so they are also bounded between 0 and 1

We use a linear activation function for the final layer and all other layers use the following leaky rectified linear activation:

sum-squared error
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.
Also, in every image 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

we increase the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don’t contain objects

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

YOLO predicts multiple bounding boxes per grid cell.
We 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

Note that the loss function only penalizes classification error if an object is present in that grid cell
also only penalizes bounding box coordinate error if that predictor is “responsible” for the ground truth box

Inference

since it only requires a single network evaluation, unlike classifier-based methods
The grid design enforces spatial diversity in the bounding box predictions

Non-maximal suppression can be used to fix these multiple detections
non-maximal suppression adds 2- 3% in mAP.

Limitations of YOLO

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.

Our model struggles with small objects that appear in groups, such as flocks of birds.

it struggles to generalize to objects in new or unusual aspect ratios or configurations
also uses relatively coarse features for predicting bounding boxes since our architecture has multiple downsampling layers from the input image

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.

Comparison to Other Detection Systems

generally start by extracting a set of robust features from input images
Then, classifiers [36, 21, 13, 10] or localizers [1, 32] are used to identify objects in the feature space.

Deformable parts models.
sliding window approach to object detection
DPM uses a disjoint pipeline to extract static features, classify regions, predict bounding boxes for high scoring regions, etc

Our system replaces all of these disparate parts with a single convolutional neural network

R-CNN
use region proposals instead of sliding windows to find objects in images
Selective Search [35] generates potential bounding boxes, a convolutional network extracts features, an SVM scores the boxes, a linear model adjusts the bounding boxes, and non-max suppression eliminates dupl
Each stage of this complex pipeline must be precisely tuned independently and the resulting system is very slow, taking more than 40 seconds per image at test time

Each grid cell proposes potential bounding boxes and scores those boxes using convolutional features.
our system puts spatial constraints on the grid cell proposals which helps mitigate multiple detections of the same object
Our system also proposes far fewer bounding boxes, only 98 per image compared to about 2000 from Selective Search.
Finally, our system combines these individual components into a single, jointly optimized model.

Other Fast Detectors
Fast and Faster R-CNN focus on speeding up the R-CNN framework by sharing computation and using neural networks to propose regions instead of Selective Search

both still fall short of real-time performance

Many research efforts focus on speeding up the DPM pipeline
They speed up HOG computation, use cascades, and push computation to GPUs

Instead of trying to optimize individual components of a large detection pipeline, YOLO throws out the pipeline entirely and is fast by design

YOLO is a general-purpose detector that learns to detect a variety of objects simultaneously

Deep MultiBox

Both YOLO and MultiBox employ convolutional networks to forecast bounding boxes within an image;
however, YOLO stands out as a self-contained detection system rather than just a component within a larger detection pipeline.

OverFeat

OverFeat cannot reason about global context and thus requires significant post-processing to produce coherent detections.

MultiGrasp

Our work is similar in design to work on grasp detection by Redmon et al [27].
Our grid approach to bounding box prediction is based on the MultiGrasp system for regression to grasps.

However, grasp detection is a much simpler task than object detection.
MultiGrasp only needs to predict a single graspable region for an image containing one object.
YOLO predicts both bounding boxes and class probabilities for multiple objects of multiple classes in an image.

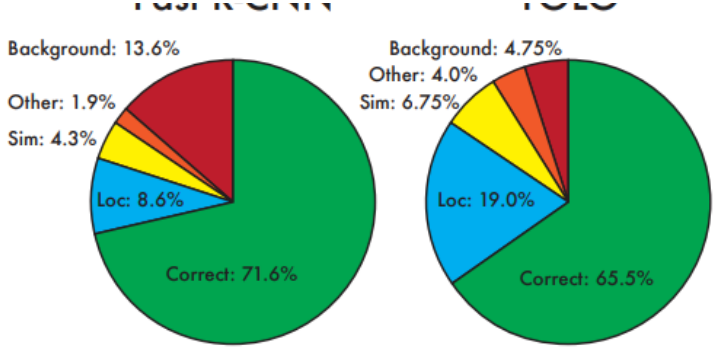
Experiments

Comparison to Other Real-Time Systems

Many research efforts in object detection focus on making standard detection pipelines fast.
However, only Sadeghi et al. actually produce a detection system that runs in real-time (30 frames per second or better).

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	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[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

VOC 2007 Error Analysis



Localization errors account for more of YOLO's errors than all other sources combined.
Fast R-CNN makes much fewer localization errors but far more background errors.

Combining Fast R-CNN and YOLO

	mAP	Combined	Gain
Fast R-CNN	71.8	-	-
Fast R-CNN (2007 data)	**66.9**	72.4	.6
Fast R-CNN (VGG-M)	59.2	72.4	.6
Fast R-CNN (CaffeNet)	57.1	72.1	.3
YOLO	63.4	**75.0**	**3.2**

Rather, it is precisely because YOLO makes different kinds of mistakes at test time that it is so effective at boosting Fast R-CNN's performance.
Unfortunately, this combination doesn't benefit from the speed of YOLO since we run each model separately and then combine the results.
However, since YOLO is so fast it doesn't add any significant computational time compared to Fast R-CNN.

VOC 2012 Results

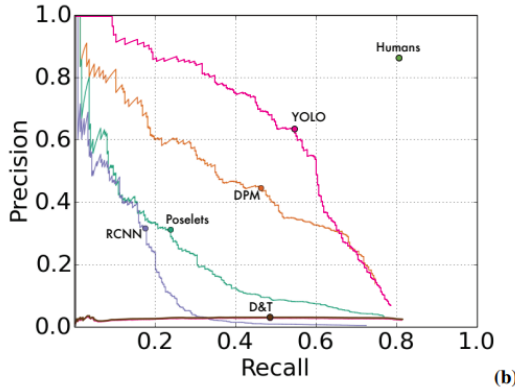
VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR_CNN_MORE_DATA [11]	**73.9**	**85.5**	**82.9**	**76.6**	**57.8**	**62.7**	**79.4**	77.2	86.6	**55.0**	**79.1**	**62.2**	87.0	**83.4**	**84.7**	78.9	45.3	73.4	65.8	80.3	74.0
HyperNet_VGG	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 [28]	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 [29]	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 [33]	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

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

Our combined Fast R-CNN + YOLO model is one of the highest performing detection methods.

Generalizability: Person Detection in Artwork



(a) Picasso Dataset precision-recall curves.

	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	

(b) Quantitative results on the VOC 2007, Picasso, and People-Art Datasets. The Picasso Dataset evaluates on both AP and best F_1 score.

In real-world applications it is hard to predict all possible use cases and the test data can diverge from what the system has seen before [3].

We compare YOLO to other detection systems on the Picasso Dataset [12] and the People-Art Dataset [3], two datasets for testing person detection on artwork.

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

Artwork and natural images are very different on a pixel level but they are similar in terms of the size and shape of objects, thus YOLO can still predict good bounding boxes and detections.

Real-Time Detection In The Wild

We connect YOLO to a webcam and verify that it maintains real-time performance, including the time to fetch images from the camera and display the detections.

While YOLO processes images individually, when attached to a webcam it functions like a tracking system, detecting objects as they move around and change in appearance.

Conclusion

Unlike classifier-based approaches, YOLO is trained on a loss function that directly corresponds to detection performance and the entire model is trained jointly.

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.