Оглавление

[1 Introduction 2](#_Toc166604990)

[1.1 Unified Detection 5](#_Toc166604991)

# 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 classifier for that object and evaluate it at various locations and scales in a test image. Systems like deformable parts models (DPM) use a sliding window approach where the classifier is run at evenly spaced locations over the entire image [. More recent approaches like R-CNN use region proposal

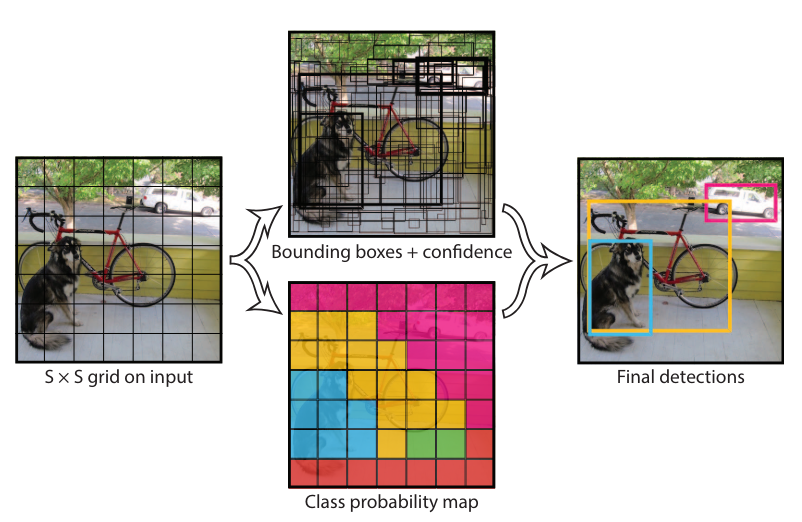
1. Resize image.
2. Run convolutional network.
3. Non-max suppression.



Figure 1 - The YOLO Detection System. Processing images with YOLO is simple and straightforward

* Dog: 0.30
* Person: 0.64
* Horse: 0.28

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.

Figure 2 - S\*S grid on input, etc.

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.

Table 1 - Real-Time Detectors Train MAP FPS

|  |  |  |  |
| --- | --- | --- | --- |
| Real-Time Detectors | Train | MAP | FPS |
| 100Hz DPM | 30 | 2007 | 16.0 |
| 30Hz DPM 30 | 30 | 2007 | 26.1 |
| Fast YOLO | 2007+2012 | 155 | 52.7 |
| YOLO | 2007+2012 | 63.4 | 45 |

Table 2 - Less Than Real-Time

|  |  |  |  |
| --- | --- | --- | --- |
| Less Than Real-Time | Train | MAP | FPS |
| 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-1627 | 2007+2012 | 73.2 | 7 |
| Faster R-CNN ZF | 2007+2012 | 62.1 | 18 |

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 less than 25 milliseconds of latency. Furthermore, YOLO achieves more than twice the mean average precision of other real-time systems. Second, YOLO reasons globally about the image when 1779 making predictions (1). Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance.

Fast R-CNN, a top detection method [14], mistakes background patches in an image for objects because it can’t see the larger context. YOLO makes less than half the number of background errors compared to Fast R-CNN. Third, YOLO learns generalizable representations of objects. When trained on natural images and tested on artwork, YOLO outperforms top detection methods like DPM and R-CNN by a wide margin. Since YOLO is highly generalizable it is less likely to break down when applied to new domains or unexpected inputs. YOLO still lags behind state-of-the-art detection systems in accuracy. While it can quickly identify objects in images it struggles to precisely localize some objects, especially small ones. We examine these tradeoffs further in our experiments. All of our training and testing code is open source. A variety of trained models are also available to download.

## Unified Detection

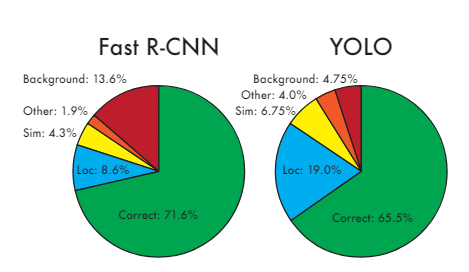


Figure 3 - Fast R-CNN

We unify the separate components of object detection into a single neural network. Our network uses features from the entire image to predict each bounding box. It also predicts all bounding boxes across all classes for an image simultaneously. This means our network reasons globally about the full image and all the objects in the image. The YOLO design enables end-to-end training and realtime speeds while maintaining high average precision. Our system divides the input image into an S × S grid (2). If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. 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) ∗ IOUtruth pred (2) . If no object exists in that cell, the confidence scores should be zero. Otherwise 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.

Where 1 coordinates represent the center of the box relative to the bounds of the grid cell