

---

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

---

000  
001  
002  
003  
004  
005  
006  
007  
008  
009  
010  
011  
012  
013  
014  
015  
016  
017  
018  
019  
020  
021  
022  
023  
024  
025  
026  
027  
028  
029  
030  
031  
032  
033  
034  
035  
036  
037  
038  
039  
040  
041  
042  
043  
044  
045  
046  
047  
048  
049  
050  
051  
052  
053  
**Anonymous Author(s)**

Affiliation  
Address  
email

## Abstract

We present YOLO, a unified pipeline for object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separate 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 also extremely fast; YOLO processes images in real-time at 45 frames per second, hundreds to thousands of times faster than existing detection systems. Our system uses global image context to detect and localize objects, making it less prone to background errors than top detection systems like R-CNN. By itself, YOLO detects objects at unprecedented speeds with moderate accuracy. When used in combination with state-of-the-art detectors, YOLO boosts performance by 2-3% points mAP.

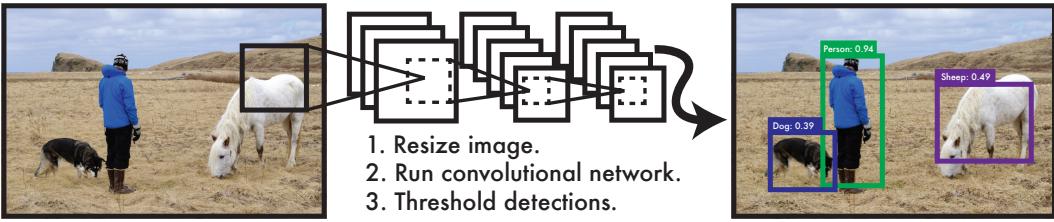
## 1 Introduction

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 or grocery shopping with little conscious thought. Fast, accurate, algorithms for object detection would allow computers to drive cars in any weather 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.

Convolutional neural networks (CNNs) achieve impressive performance on classification tasks at real-time speeds [13]. Yet top object detection systems like R-CNN take seconds to process individual images and hallucinate objects in background noise. We believe these shortcomings result from how these systems approach object detection.

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 [7]. More recent approaches like R-CNN use region proposal 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 box, eliminate duplicate detections, and rescore the box based on other objects in the scene [9].

These region proposal techniques typically generate a few thousand potential boxes per image. Selective Search, the most common region proposal method, takes 1-2 seconds per image to generate these boxes [26]. The classifier then takes additional time to evaluate the proposals. The best performing systems require 2-40 seconds per image and even those optimized for speed do not achieve real-time performance. Additionally, even a highly accurate classifier will produce false positives



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

067 when faced with so many proposals. When viewed out of context, small sections of background can  
068 resemble actual objects, causing detection errors.

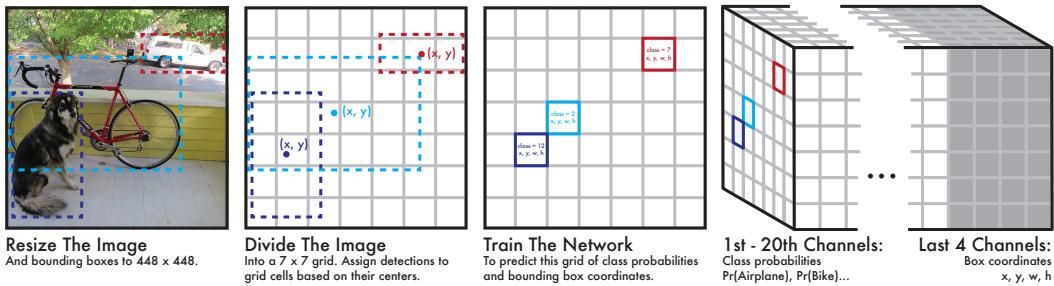
069 Finally, these detection pipelines rely on independent techniques at every stage that cannot be optimized  
070 jointly. A typical pipeline uses Selective Search for region proposals, a convolutional network  
071 for feature extraction, a collection of one-versus-all SVMs for classification, non-maximal suppression  
072 to reduce duplicates, and a linear model to adjust the final bounding box coordinates. Selective  
073 Search tries to maximize recall while the SVMs optimize for single class accuracy and the linear  
074 model learns from localization error.

075 Our system is refreshingly simple, see Figure 1. A single convolutional network simultaneously  
076 predicts multiple bounding boxes and class probabilities for those boxes. We train our network on  
077 full images and directly optimize detection performance. Context matters in object detection. Our  
078 network uses global image features to predict detections which drastically reduces its errors from  
079 background detections. At test time, a single network evalution of the full image produces detections  
080 of multiple objects in multiple categories without any pre or post-processing.

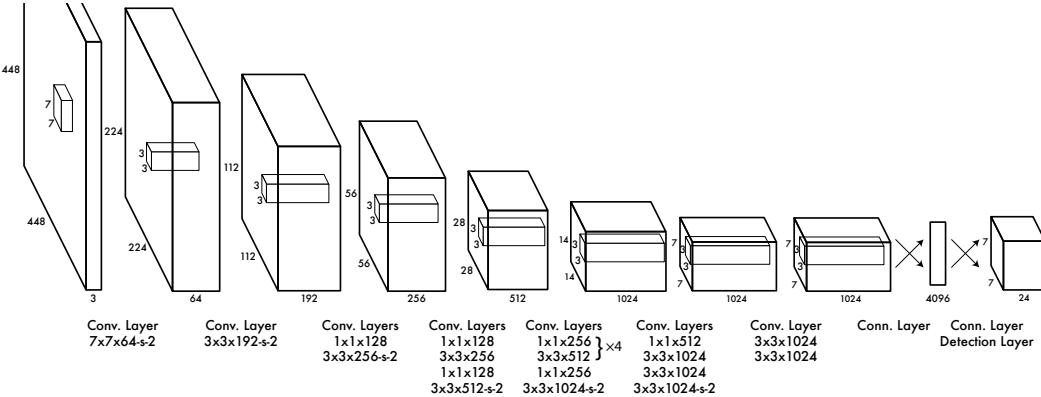
081 Our training and testing code are open source and available online at [redacted]. A variety of pre-  
082 trained models are also available to download.

## 085 2 Unified Detection

086 We unify the separate components of object detection into a single neural network. Using our  
087 system, you only look once (YOLO) at an image to predict what objects are present and where they  
088 are. Our network uses features from the entire image to predict each bounding box. It also predicts  
089 all bounding boxes for an image simultaneously. This means our network reasons globally about  
090 the full image and all the objects in the image. The YOLO design enables end-to-end training and  
091 real-time speeds while maintaining high average precision.



105 **Figure 2: The Model.** Our system models detection as a regression problem to a  $7 \times 7 \times 24$  tensor.  
106 This tensor encodes bounding boxes and class probabilities for all objects in the image.



**Figure 3: The Architecture.** Our detection network has 24 convolutional layers followed by 2 fully connected layers. The network uses strided convolutional layers to downsample the feature space instead of maxpooling layers. Alternating  $1 \times 1$  convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution ( $224 \times 224$  input image) and then double the resolution for detection.

## 2.1 Design

Our system divides the input image into a  $7 \times 7$  grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts a bounding box and class probabilities associated with that bounding box, see Figure 2.

We implement this model as a convolutional neural network and evaluate it on the PASCAL VOC detection dataset [6]. 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 [25]. Our network has 24 convolutional layers followed by 2 fully connected layers. However, instead of the inception modules used by GoogLeNet we simply use  $1 \times 1$  reduction layers followed by  $3 \times 3$  convolutional layers, similar to Lin et al [17]. We also replace maxpooling layers with strided convolutions. The full network is shown in Figure 3.

The final output of our network is a  $7 \times 7$  grid of predictions. Each grid cell predicts 20 conditional class probabilities, and 4 bounding box coordinates.

## 2.2 Training

We pretrain our convolutional layers on the ImageNet 1000-class competition dataset [22]. For pretraining we use the first 20 convolutional layers from Figure 3 followed by a maxpooling layer and two fully connected layers. We train this network for approximately a week and achieve a top-5 accuracy of 86% on the ImageNet 2012 validation set.

We then adapt the model to perform detection. Ren et al. show that adding both convolutional and connected layers to pretrained networks can benefit performance [21]. Following their example, we add four convolutional layers and two fully connected layers with randomly initialized weights. Detection often requires fine-grained visual information so we increase the input resolution of the network from  $224 \times 224$  to  $448 \times 448$ .

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 parameterize 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 logistic activation function to reflect these constraints on the final layer. All other layers use the following leaky rectified linear activation:

$$\phi(x) = \begin{cases} 1.1x, & \text{if } x > 0 \\ .1x, & \text{otherwise} \end{cases} \quad (1)$$

We optimize for sum-squared error in the output of our model. We use sum-squared error because it is easy to optimize, 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. To remedy this, we use a scaling factor  $\lambda$  to adjust the weight given to error from coordinate predictions versus error from class probabilities. In our final model we use the scaling factor  $\lambda = 4$ .

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.

If cell  $i$  predicts class probabilities  $\hat{p}_i(\text{aeroplane}), \hat{p}_i(\text{bicycle})\dots$  and the bounding box  $\hat{x}_i, \hat{y}_i, \hat{w}_i, \hat{h}_i$  then our full loss function for an example is:

$$\sum_{i=0}^{48} \left( \lambda \mathbb{1}_i^{\text{obj}} ((x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2) + \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \right) \quad (2)$$

Where  $\mathbb{1}_i^{\text{obj}}$  encodes whether any object appears in cell  $i$ . Note that if there is no object in a cell we do not consider any loss from the bounding box coordinates predicted by that cell. In this case, there is no ground truth bounding box so we only penalize the associated probabilities with that region.

We train the network for about 120 epochs on the training and validation data sets from PASCAL VOC 2007 and 2012 as well as the test set from 2007, a total of 21k images. Throughout training we use a batch size of 64, a momentum of 0.9 and a decay of 0.0005. We use two learning rates during training:  $10^{-2}$  and  $10^{-3}$ . Training diverges if we use the higher learning rate,  $10^{-2}$ , from the start. We use the lower rate,  $10^{-3}$ , for one epoch so that the randomly initialized weights in the final layers can settle to reasonable values. Then we train with the following learning rate schedule:  $10^{-2}$  for 80 epochs, and  $10^{-3}$  for 40 epochs.

To avoid overfitting we use dropout and extensive data augmentation. A dropout layer with rate = .5 after the first connected layer prevents co-adaptation between layers [14]. For data augmentation we introduce random scaling and translations of up to 10% of the original image size. We also randomly adjust the exposure and saturation of the image by up to a factor of 2.

### 2.2.1 Parameterizing Class Probabilities

Each grid cell predicts class probabilities for that area of the image. There are 49 cells with a possible 20 classes each yielding 980 predicted probabilities per image. Most of these probabilities will be zero since only a few objects appear in any given image. Left unchecked, this imbalance pushes all of the probabilities to zero, leading to divergence during training.

To overcome this, we add an extra variable to each grid location, the probability that any object exists in that location regardless of class. Thus instead of 20 class probabilities we have 1 “objectness” probability,  $\text{Pr}(\text{Object})$ , and 20 conditional probabilities:  $\text{Pr}(\text{Airplane}|\text{Object})$ ,  $\text{Pr}(\text{Bicycle}|\text{Object})$ , etc.

To get the unconditional probability for an object class at a given location we simply multiply the “objectness” probability by the conditional class probability:

$$\text{Pr}(\text{Dog}) = \text{Pr}(\text{Object}) * \text{Pr}(\text{Dog}|\text{Object}) \quad (3)$$

We can optimize these probabilities independently or jointly using a novel “detection layer” in our convolutional network. During the initial stages of training we optimize them independently to improve model stability. We update the “objectness” probabilities at every location however we only update the conditional probabilities at locations that actually contain an object. This means there are far fewer probabilities getting pushed towards zero.

During later stages of training we optimize the unconditioned probabilities by performing the required multiplications in the network and calculating error based on the result.

216    **2.2.2 Predicting IOU**  
217

218    Like most detection systems, our network has trouble precisely localizing small objects. While it  
219    may correctly predict that an object is present in an area of the image, if it does not predict a precise  
220    enough bounding box the detection is counted as a false positive.

221    We want YOLO to have some notion of uncertainty in its probability predictions. Instead of predicting  
222    1-0 probabilities we can scale the target class probabilities by the IOU of the predicted bounding  
223    box with the ground truth box for a region. When YOLO predicts good bounding boxes it is also  
224    encouraged to predict high class probabilities. For poor bounding box predictions it learns to predict  
225    lower confidence probabilities.

226    We do not train to predict IOU from the beginning, only during the second stage of training. It is not  
227    necessary for good performance but it does boost our mean average precision by 3-4%.  
228

229    **2.3 Inference**  
230

231    Just like in training, predicting detections for a test image only requires one network evaluation.  
232    The network predicts 49 bounding boxes per image and class probabilities for each box. YOLO is  
233    extremely fast at test time since it only requires a single network evaluation unlike classifier-based  
234    methods.

235    The grid design enforces spatial diversity in the bounding box predictions. Often it is clear which  
236    grid cell an object falls in to and the network only predicts one box for each object. However, some  
237    large objects or objects near the border of multiple cells can be well localized by multiple cells. Non-  
238    maximal suppression can be used to fix these multiple detections. While not critical to performance  
239    as it is for R-CNN or DPM, non-maximal suppression adds 2-3% in mAP.  
240

241    **2.4 Limitations of YOLO**  
242

243    YOLO imposes strong spatial constraints on bounding box predictions since each grid cell only  
244    predicts one box. This spatial constraint limits the number of nearby objects that our model can  
245    predict. If two objects fall into the same cell our model can only predict one of them. Our model  
246    struggles with small objects that appear in groups, such as flocks of birds.

247    Since our model learns to predict bounding boxes from data, it struggles to generalize to objects in  
248    new or unusual aspect ratios or configurations. Our model also uses relatively coarse features for  
249    predicting bounding boxes since our architecture has multiple downsampling layers from the input  
250    image.

251    Finally, while we train on a loss function that approximates detection performance, our loss function  
252    treats errors the same in small bounding boxes versus large bounding boxes. A small error in a large  
253    box is generally benign but a small error in a small box has a much greater effect on IOU. Our main  
254    source of error is incorrect localizations.  
255

256    **3 Comparison to Other Detection Systems**  
257

258    Object detection is a core problem in computer vision. Detection pipelines generally start by ex-  
259    tracting a set of robust features from input images (Haar [19], SIFT [18], HOG [2], convolutional  
260    features [3]). Then, classifiers [27, 16, 9, 7] or localizers [1, 23] are used to identify objects in the  
261    feature space. These classifiers or localizers are run either in sliding window fashion over the whole  
262    image or on some subset of regions in the image [26, 11, 28]. We compare the YOLO detection  
263    system to several top detection frameworks, highlighting key similarities and differences.

264    **Deformable parts models.** Deformable parts models (DPM) use a sliding window approach to  
265    object detection [7]. DPM uses a disjoint pipeline to extract static features, classify regions, predict  
266    bounding boxes for high scoring regions, etc. Our system replaces all of these disparate parts with  
267    a single convolutional neural network. The network performs feature extraction, bounding box  
268    prediction, non-maximal suppression, and contextual reasoning all concurrently. Instead of static  
269    features, the network trains the features in-line and optimizes them for the detection task. Our  
unified architecture leads to a faster, more accurate model than DPM.

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.S.CNN [8]	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
DEEP.LENS.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
<b>Fast R-CNN + YOLO</b>	<b>70.0</b>	<b>83.0</b>	<b>78.4</b>	<b>73.4</b>	<b>55.7</b>	<b>42.5</b>	<b>78.2</b>	<b>72.7</b>	<b>89.5</b>	<b>48.2</b>	<b>74.0</b>	<b>56.4</b>	<b>87.2</b>	<b>80.8</b>	<b>80.7</b>	<b>74.4</b>	<b>41.1</b>	<b>70.0</b>	<b>67.1</b>	<b>81.2</b>	<b>66.0</b>
NoC [21]	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 [10]	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
NUS.NIN.C2000 [4]	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 [4]	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
R-CNN VGG BB [9]	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
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 [9]	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
Feature Edit [24]	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
<b>YOLO</b>	<b>53.6</b>	<b>71.6</b>	<b>62.2</b>	<b>55.2</b>	<b>35.9</b>	<b>23.2</b>	<b>62.4</b>	<b>53.7</b>	<b>78.1</b>	<b>34.0</b>	<b>52.9</b>	<b>38.7</b>	<b>72.2</b>	<b>67.3</b>	<b>66.3</b>	<b>62.1</b>	<b>25.5</b>	<b>50.0</b>	<b>46.9</b>	<b>67.4</b>	<b>46.3</b>
R-CNN BB [9]	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 [12]	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 [9]	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

**Table 1: PASCAL VOC 2012 Leaderboard.** YOLO compared with the full comp4 (outside data allowed) public leaderboard as of June 5th, 2015. Mean average precision and per-class average precision are shown for a variety of detection methods. YOLO is the top detection method that is not based on the R-CNN detection framework. Fast R-CNN + YOLO is one of the top methods overall, with a 1.6% boost over Fast R-CNN and the highest average precision in 6 out of 20 classes.

**R-CNN.** R-CNN and its variants use region proposals instead of sliding windows to find objects in images. These systems use region proposal methods like Selective Search [26] to generate potential bounding boxes in an image. Instead of scanning through every region in the window, now the classifier only has to score a small subset of potential regions in an image. Good region proposal methods maintain high recall despite greatly limiting the search space. This performance comes at a cost. Selective Search, even in “fast mode” takes about 2 seconds to propose regions for an image.

R-CNN shares many design aspects with DPM. After region proposal, R-CNN uses the same multi-stage pipeline of feature extraction (using CNNs instead of HOG), SVM scoring, non-maximal suppression, and bounding box prediction using a linear model [9].

YOLO shares some similarities with R-CNN. Each grid cell proposes a potential bounding box and then scores that bounding box using convolutional features. However, 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 49 per image compared to about 2000 from Selective Search. Finally, our system combines these individual components into a single, jointly optimized model.

**Deep MultiBox.** Unlike R-CNN, Szegedy et al. train a convolutional neural network to predict regions of interest [5] instead of using Selective Search. MultiBox can also perform single object detection by replacing the confidence prediction with a single class prediction. However, MultiBox cannot perform general object detection and is still just a piece in a larger detection pipeline, requiring further image patch classification. Both YOLO and MultiBox use a convolutional network to predict bounding boxes in an image but YOLO is a complete detection pipeline.

**OverFeat.** Sermanet et al. train a convolutional neural network to perform localization and adapt that localizer to perform detection [23]. OverFeat efficiently performs sliding window detection but it is still a disjoint system. OverFeat optimizes for localization, not detection performance. Like DPM, the localizer only sees local information when making a prediction. OverFeat cannot reason about global context and thus requires significant post-processing to produce coherent detections.

Our work is most similar in design to work on grasp detection by Redmon et al [20].

## 4 Experiments

We present detection results for the PASCAL VOC 2012 dataset and compare our mean average precision (mAP) and runtime to other top detection methods. We also perform error analysis on the VOC 2007 dataset. We compare our results to Fast R-CNN, one of the highest performing versions of R-CNN [10]. We use publicly available runs of Fast R-CNN available on GitHub. Finally we show that a combination of our method with Fast R-CNN gives a significant performance boost.

324      **4.1 VOC 2012 Results**

325

326      On the VOC 2012 test set we achieve 53.6 mAP. This is lower than the current state of the art,  
 327      closer to R-CNN based methods that use AlexNet, see Table 1. Our system struggles with small ob-  
 328      jects compared to its closest competitors. On categories like `bottle`, `sheep`, and `tv/monitor`  
 329      YOLO scores 8-10 percentage points lower than R-CNN or Feature Edit. However, on other cate-  
 330      gories like `cat` and `horse` YOLO achieves significantly higher performance. We further investi-  
 331      gate the source of these performance disparities in Section 4.3.

332      **4.2 Speed**

333

334      At test time YOLO processes images at 45 frames per second on an Nvidia Titan X GPU. It is  
 335      considerably faster than classifier-based methods with similar mAP. Normal R-CNN using AlexNet  
 336      or the small VGG network take 400-500x longer to process images. The recently proposed Fast  
 337      R-CNN shares convolutional features between the bounding boxes but still relies on selective search  
 338      for bounding box proposals which accounts for the bulk of their processing time. YOLO is still  
 339      around 100x faster than Fast R-CNN. Table 2 shows a full comparison between multiple R-CNN  
 340      and Fast R-CNN variants and YOLO.

	mAP	Prediction Time	FPS	Compared to YOLO
R-CNN (VGG-16)	66.0	48.2 hr	0.02 fps	1500x
FR-CNN (VGG-16)	66.9	3.1 hr	0.45 fps	100x
R-CNN (Small VGG)	60.2	14.4 hr	0.09 fps	500x
FR-CNN (Small VGG)	59.2	2.9 hr	0.48 fps	93x
R-CNN (Caffe)	58.5	12.2 hr	0.11 fps	409x
FR-CNN (Caffe)	57.1	2.8 hr	0.48 fps	93x
YOLO	58.8	110 sec	45 fps	-

350      **Table 2: Prediction Timing.** mAP and timing information for R-CNN, Fast R-CNN, and YOLO on the VOC  
 351      2007 test set. Timing information is given both as frames per second and the time each method takes to process  
 352      the full 4952 image set. The final column shows the relative speed of YOLO compared to that method.

354      **4.3 VOC 2007 Error Analysis**

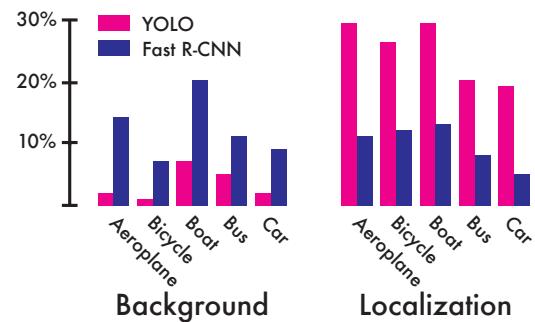
355

356      An object detector must have high recall for  
 357      objects in the test set to obtain high perfor-  
 358      mance. Our model imposes spatial constraints  
 359      on bounding box predictions which limits re-  
 360      call on small objects that are close together. We  
 361      examine how detrimental this is in practice by  
 362      calculating our highest possible recall assum-  
 363      ing perfect coordinate prediction. Under this  
 364      assumption, our model can achieve a 93.1% re-  
 365      call for objects in the VOC 2007 test set. This  
 366      is lower than Selective Search (98.0% [26]) but  
 367      still relatively high.

368      Using the methodology and tools of Hoiem et  
 369      al. [15] we analyze our performance on the  
 370      VOC 2007 test set. We compare YOLO to Fast  
 371      R-CNN using VGG-16, one of the highest per-  
 372      forming object detectors.

373      Figure 4 compares frequency of localization  
 374      and background errors between Fast R-CNN and YOLO. A detection is considered a localiza-  
 375      tion error if it overlaps a true positive in the image but by less than the required 50% IOU. A detection is  
 376      a background error if the box does not overlap any objects of any class in the image.

377      Fast R-CNN makes around the same number of localization errors and background errors. Over all  
 378      20 classes in the top N detections 13.6% are localization errors and 8.6% are background errors.



379      **Figure 4: Error Analysis: Fast R-CNN vs. YOLO**  
 380      These charts show the percentage of localization and  
 381      background errors in the top N detections for various  
 382      categories ( $N = \#$  objects in that category).

378 YOLO makes far more localization errors but relatively few background errors. Averaged across all  
379 classes, of its top N detections 24.7% are localization errors and a mere 4.3% are background errors.  
380 This is about twice the number of localization errors but half the number of background detections.  
381

382 YOLO uses global context to evaluate detections while R-CNN only sees small portions of the  
383 image. Many of the background detections made by R-CNN are obviously not objects when shown  
384 in context. YOLO and R-CNN are good at different parts of object detection. Since their main  
385 sources of error are orthogonal, combining them should produce a model that is better across the  
386 board.

#### 387 4.4 Combining Fast R-CNN and YOLO

389 YOLO makes far fewer background mistakes  
390 than Fast R-CNN. By using YOLO to elimi-  
391 nate background detections from Fast R-CNN  
392 we get a significant boost in performance. For  
393 every bounding box that R-CNN predicts we  
394 check to see if YOLO predicts a similar box.  
395 If it does, we give that prediction a boost based  
396 on the probability predicted by YOLO and the  
397 overlap between the two boxes. This reorders  
398 the detections to favor those predicted by both  
399 systems. Since we still use Fast R-CNN’s  
400 bounding box predictions we do not introduce  
401 any localization error. Thus we take advantage  
402 of the best aspects of both systems.

403 The best Fast R-CNN model achieves a mAP of  
404 71.8% on the VOC 2007 test set. When com-  
405 bined with YOLO, its mAP increases by 2.9% to 74.7%. We also tried combining the top Fast  
406 R-CNN model with several other versions of Fast R-CNN. Those ensembles produced small in-  
407 creases in mAP between .3 and .6%, see Table 3 for details. Thus, the benefit from combining Fast  
408 R-CNN with YOLO is unique, not a general property of combining models in this way.

409 Using this combination strategy we achieve a significant boost on the VOC 2012 and 2010 test sets  
410 as well, around 2%. The combined Fast R-CNN + YOLO model performs on par with the best  
411 models on the VOC 2012 leaderboard.

## 412 5 Conclusion

415 We introduce YOLO, a unified pipeline for object detection. Our model is simple to construct and  
416 can be trained directly on full images. Unlike classifier-based approaches, YOLO is trained on a  
417 loss function that directly corresponds to detection performance and every piece of the pipeline can  
418 be trained jointly.

419 The network reasons about the entire image during inference which makes it less likely to predict  
420 background false positives than sliding window or proposal region detectors. Moreover, it predicts  
421 detections with only a single network evaluation, making it extremely fast.

422 YOLO achieves impressive performance on standard benchmarks considering it is 2-3 orders of  
423 magnitude faster than existing detection methods. It can also be used to rescore the bounding boxes  
424 produced by state-of-the-art methods like R-CNN for a significant boost in performance.

## 426 References

- 428 [1] M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In  
429 *Computer Vision–ECCV 2008*, pages 2–15. Springer, 2008.
- 430 [2] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision*  
431 *and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages  
886–893. IEEE, 2005.

- 432 [3] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.
- 433 [4] J. Dong, Q. Chen, S. Yan, and A. Yuille. Towards unified object detection and semantic segmentation. In *Computer Vision–ECCV 2014*, pages 299–314. Springer, 2014.
- 434 [5] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2155–2162. IEEE, 2014.
- 435 [6] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, Jan. 2015.
- 436 [7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.
- 437 [8] S. Gidaris and N. Komodakis. Object detection via a multi-region & semantic segmentation-aware CNN model. *CoRR*, abs/1505.01749, 2015.
- 438 [9] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014.
- 439 [10] R. B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.
- 440 [11] S. Gould, T. Gao, and D. Koller. Region-based segmentation and object detection. In *Advances in neural information processing systems*, pages 655–663, 2009.
- 441 [12] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *Computer Vision–ECCV 2014*, pages 297–312. Springer, 2014.
- 442 [13] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.
- 443 [14] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.
- 444 [15] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error in object detectors. In *Computer Vision–ECCV 2012*, pages 340–353. Springer, 2012.
- 445 [16] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–900. IEEE, 2002.
- 446 [17] M. Lin, Q. Chen, and S. Yan. Network in network. *CoRR*, abs/1312.4400, 2013.
- 447 [18] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.
- 448 [19] C. P. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *Computer vision, 1998. sixth international conference on*, pages 555–562. IEEE, 1998.
- 449 [20] J. Redmon and A. Angelova. Real-time grasp detection using convolutional neural networks. *CoRR*, abs/1412.3128, 2014.
- 450 [21] S. Ren, K. He, R. B. Girshick, X. Zhang, and J. Sun. Object detection networks on convolutional feature maps. *CoRR*, abs/1504.06066, 2015.
- 451 [22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015.
- 452 [23] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013.
- 453 [24] Z. Shen and X. Xue. Do more dropouts in pool5 feature maps for better object detection. *arXiv preprint arXiv:1409.6911*, 2014.
- 454 [25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.
- 455 [26] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.
- 456 [27] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 4:34–47, 2001.
- 457 [28] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *Computer Vision–ECCV 2014*, pages 391–405. Springer, 2014.

481  
482  
483  
484  
485