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

Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi

Abstract

Unlike existing work that reuse classifiers for detection, in this paper, the authors propose the YOLO system for object detection. The YOLO system formulates the detection problem as a regression problem. The system is composed of a single neural network that predicts spatially separated bounding boxes in an image and the associated class probabilities in one evaluation. The system is also easily optimized end-to-end. The YOLO system is fast as it can process real-time data at a rate of 45 frames per second. Fast YOLO is a faster version of the YOLO system, which is also proposed in this paper and can process frames at a rate of 155 frames per second while achieving double the mAP when compared to other real-time detectors. Although the YOLO system has higher localization error than existing detectors, yet it makes less false positive predictions on objects in the background. Moreover, YOLO generalizes to images from other domains like artwork better than DPM and R-CNN detectors.

1 Introduction

While humans can easily detect objects in an image by simply looking at the image, computers need fast and accurate algorithms for object detection. These fast, accurate algorithms would allow computers to perform complex tasks such as driving cars and conveying real-time information to humans.

Existing detection systems reuse classifiers for object detection. To detect an object in an image, a classifier is used for that object and run at several locations and scales in the image. For example, the Deformable Parts Model (DPM) system uses a sliding window technique

for object detection, where a classifier is run at evenly spaced locations in the image. A more recent algorithm is Region-Based Convolutional Neural Networks (R-CNNs) which uses a region-proposal based method to generate bounding boxes for an image, and then run a classifier on these bounding boxes. However, these existing algorithms (DPM and R-CNN) are complex, slow, and difficult to optimize since each component needs to be trained individually.

On the other hand, the YOLO system provides a unified approach for real-time object detection. YOLO formulates the detection problem as a regression problem, where a single neural network only looks once at the entire image and makes predictions about spatially separated bounding boxes and associated class probabilities, i.e. the network predicts what objects are present in the image and where these objects are located in one evaluation. The network is easily trained end-to-end on full images and its parameters are optimized for improved detection performance.

When compared to existing real-time detectors, YOLO offers three advantages, but has one drawback. As for the advantages, first, YOLO is fast. It can process real-time video with latency of less than 25 ms. The reason for this is that only a single network is used to make predictions for an image in one evaluation, i.e. there is no complex pipeline. The network runs at 45 frames per second with no batch processing on a Titan X GPU. Fast YOLO is a faster version of the YOLO system and runs images at rate of more than 155 frames per second while achieving more than twice the mean average precision (mAP) of other detectors. Second, since YOLO looks at the entire image during training time and test time, it can capture global contextual information about an image unlike sliding window (DPM) and region-proposal based (R-CNN) techniques. Compared to YOLO, fast R-CNN falsely predicts patches in the background as objects because it does not see the global context of image as YOLO. Hence, YOLO makes less than half of the false positive predictions made by fast R-CNN. Third, YOLO generalizes well on images from new domains. When trained on natural images and tested on artwork images, YOLO outperforms DPM and R-CNN. However, when compared to state-of-the-art systems, YOLO cannot precisely determine the location of an object in an image.

2 Unified Detection

The YOLO system uses a single neural network to detect objects in a image. The system divides the input image into an $S \times S$ grid, where each grid cell predicts B bounding boxes and one set of class probabilities C. Each bounding box consists of five predictions:

- The (x, y) coordinates of the center of the box with respect to the cell grid bounds.
- The width w and height h of the box with respect to the whole image.
- A confidence score which reflects both whether an object exists in the cell or not and how accurately the box can predict that object (if present). The confidence score is defined as:

$$Pr[Object] \times IOU_{pred}^{truth}, \tag{1}$$

where $Pr[Object] = 0$ if an object does not exist while $Pr[Object] = 1$ if the center of an object falls into the grid cell, and therefore, the cell detects that object. The IOU_{pred}^{truth} is the intersection over union between the ground truth and prediction and represents how accurate the box predicts the object. In other words, if an object is not present in the cell, then the confidence score equals to zero. Otherwise, the confidence score equals to IOU_{pred}^{truth}.

In addition to predicting B bounding boxes, each grid cell predicts C class probabilities that are conditioned over the presence of an object within the grid cell $Pr(Class_i|Object)$.

At inference, the class-specific confidence scores for each bounding box is obtained as:

$$Pr(Class_i|Object) * Pr[Object] \times IOU_{pred}^{truth} = Pr(Class_i) \times IOU_{pred}^{truth}, \tag{2}$$

which reflect both the probability that the box predicts that class and how accurately the box predicts it.

2.1 Network Design

The model is a convolutional neural network inspired by the GoogLeNet model and consists of 24 convolutional layers followed by 2 fully connected layers. The convolutional layers extract features form the image whereas the fully connected layers make predictions about coordinates of the bounding boxes and class probabilities. A 3×3

convolutional layer is followed by a 1×1 convolutional layer in an alternating manner. Reduction 1×1 convolutional layers reduce the feature space from the preceding 3×3 convolutional layer. The final output of the network is a tensor of predictions of dimensions $S \times S \times (B \times 5 + C) = 7 \times 7 \times 30$. To evaluate the YOLO system, the authors used the PASCAL VOC dataset. The authors assumed $S = 7$ and $B = 2$. The PASCAL VOC dataset contains 20 labelled classes ($C = 20$).

The authors also trained a Fast YOLO model which differs from YOLO only in the size of the network. In Fast YOLO, the network has 9 convolutional layers with fewer filters in each layer.

2.2 Training

The authors pretrained the first 20 convolutional layers followed by an average-pooling layer on the ImageNET competition dataset (with 1000 class probabilities). After training that network for one week, the network achieved 55% validation accuracy on the ImageNET 2012 validation set. To perform detection (which is the original task of the network), the authors then added four convolutional layers followed by 2 fully connected layers and randomly initialized their weights. They also increased the input image resolution from 224×224 to 448×448 to obtain fine-grained visual information necessary for detection. The final fully connected layer of the network predicts coordinates of the bounding boxes and class probabilities. The center of the bounding box (x, y) coordinates is calculated as an offset from the location of the grid cell so that x and y are in the range between 0 and 1. Also, the width w and height h of the bounding box are normalized to the width and height of the whole image, respectively, to be in the range between 0 and 1. The authors used a linear activation function for the final layer whereas they used the leaky rectified linear activation for all other layers:

$$\phi(x) = \begin{cases} x, & x > 0 \\ 0.1x, & \text{otherwise} \end{cases} \tag{3}$$

To train the network, the authors minimized the sum-squared error (loss function) because it is easy to optimize. However, the sum squared error has three limitations. First, it weights the localization error (i.e. the error in x, y, w and h predictions) equally with the classification error (i.e. the error in class probabilities predictions),

which is not ideal. To address this problem, the authors weighted the localization error by a weighting parameter $\lambda_{\text{coord}} = 5$ (i.e. increase loss due to coordinates predictions). Second, for grid cells that do not contain an object, their confidence score will be zero which will dominate the gradient and lead to model instability. To solve this problem, the authors weighted the confidence score prediction error for boxes that do not contain an object by a weighting parameter $\lambda_{\text{noobj}} = 0.5$ (i.e. decrease the loss due to confidence predictions). Third, the sum-squared error also equally weights errors in large boxes and errors in small boxes. However, small errors in large boxes should have less effect on loss than small errors in small boxes. To achieve this, the authors minimized the sum-squared error between the actual and predicted square root of the width and height of the bounding box. The loss function is expressed as:

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \Big[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \Big] \\
&\quad + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \Big[\Big(\sqrt{w_i} - \sqrt{\hat{w}_i} \Big)^2 + \Big(\sqrt{h_i} - \sqrt{\hat{h}_i} \Big)^2 \Big] \\
&\quad + \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \Big(C_i - \hat{C}_i \Big)^2 \\
&\quad + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \Big(C_i - \hat{C}_i \Big)^2 \\
&\quad + \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2, \qquad (4)
\end{aligned}
$$

where $\mathbb{1}_i^{\text{obj}}$ determines whether an object exists in cell i or not. Therefore, the loss function penalizes the classification error (fifth term) only if an object is present in cell i. $\mathbb{1}_{ij}^{\text{obj}}$ determines whether the j^{th} bounding box in cell i is responsible for the prediction or not. Hence, the loss function penalizes the coordinates error (first and second terms) and confidence score error (third term) only if the j^{th} box is responsible for predicting an object. To illustrate this, in YOLO each grid cell predicts B bounding boxes. During training, only the box with the highest IOU_{truth}^{pred} is responsible for predicting an object. This improves the prediction performance.

The model was trained using the training and validation data from PASCAL VOC 2007 and 2012. The testing data from VOC 2007 was also used for training. The authors set the hyper-parameters as follows:

- Number of training epochs = 135
- Batch size = 64
- Momentum = 0.9
- Learning rate was slowly raised from 10^{-3} to 10^{-2} for the first epochs, then 10^{-2} for 75 epochs, then 10^{-3} for 30 epochs, and lastly 10^{-4} for 30 epochs.

Dropout is used with a rate of 0.5 after the first fully connected layer to avoid overfitting. Data augmentation is also performed to avoid overfitting. This includes random scaling and translations of up to 20% of the original image size and random adjusting the exposure and saturation of the image by up to a factor of 1.5 in the HSV color space.

2.3 Inference

The model was tested on the PASCAL VOC 2012 testing data. At test time, YOLO is fast since it requires only one network evaluation for detecting an object in contrast to classifier-based methods. A disadvantage in the YOLO system is that an object can be detected by multiple grid cells which happens when an object is large or is near the boundaries of multiple cells. Since each grid cell predicts one bounding box for that object, this leads to spatial diversity in bounding boxes predictions. Although non-maximal suppression can be used to solve the problem of multiple detections, yet this problem does not affect the performance of the YOLO system as it does for R-CNN or DPM.

2.4 Limitations of YOLO

The limitations of the YOLO system include the following:

- A cell can detect limited number of nearby objects
- Not good at predicting objects that are grouped such as flock of birds

- Fails to generalize to objects with unusual aspect ratios
- Extracts coarse features from the input image due to the use of downsampling layers
- Loss function penalizes small errors in large bounding boxes same as it penalizes small errors in small bounding boxes.
- Incorrect localization dominates the error.

3 Comparison to Other Detection Systems

Generally speaking, computer vision and object detection require 2 main steps for successful implementation. Extracting features from the incoming images and then classifying them based on the data extracted are the main steps followed by different algorithms, and we will compare YOLO to current algorithms.

1. Deformable Parts Model (DPM): This object detection system uses a sliding window with separate steps to get the bounding boxes of high scores. Instead of static feature extraction, region classification, and bounding box prediction done separately as in DPM, the YOLO procedure replaces all that with a single convolutional neural network (CNN) that is optimized to extract features based on the application unlike the static feature extraction of DPM. This allows YOLO to be faster and more accurate in object detection.
2. Region Based Convolutional Neural Networks (R-CNN): This method uses proposed regions for the bounding box instead of sliding window. Selective search is used for generating the bounding boxes, followed by a CNN to extract features from the bounding boxes, support vector machine to give scores to the boxes, a linear model to adjust the bounding boxes, and non-max suppression to eliminate duplicate detections. This long-separated steps require independent fine tuning and takes more time (40 seconds at test time) to detect objects. R-CNN is similar to YOLO in the bounding box generation, but YOLO provides much fewer bounding boxes with minimal duplications and the whole process is joined in a single CNN.

3. Other Fast Detectors (Fast, Faster R-CNN): 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. These make the R-CNN much faster but still not enough for real time performance. Similarly, some proposed faster DPM by faster computation and GPU utilization, but only 30Hz DPM is suitable for real time operation. Compared to YOLO, which is fast by design and is considered a general-purpose detector that can be trained for a variety of applications like face detection.
4. Deep MultiBox: Instead of selective search to get bounding boxes as in R-CNN, this approach trains a convolutional neural network to predict regions of interest and can perform single class prediction by using the confidence as a threshold. This approach is considered a single step in image object detection as further classification is needed after getting the bounding boxes.
5. OverFeat: This approach uses a CNN to perform localization from which object detection can be made. 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.
6. MultiGrasp: The YOLO approach is similar to MultiGrasp approach in the bounding box prediction based on the grid approach. Grasp detection done by MultiGrasp is a much simpler task compared to the Object detection done by YOLO in which object size, location, and boundaries are needed to predict the class. MultiGrasp works with images containing only one object and only finds region suitable for grasping, while YOLO predicts both bounding boxes and class probabilities for multiple objects of multiple classes.

4 Experiments

The dataset PASCAL VOC 2007 is used to compare the performance of YOLO and other detection methods. Studying the errors made by YOLO and Fast RCNN was also made to merge the two algorithms, thus boosting the overall performance. Also, VOC 2012 is used to test the performance of YOLO and its generalizability is tested on two artwork datasets.

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

Table 1: Real-Time Systems on PASCAL VOC 2007. Comparing the performance and speed of fast detectors. Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real-time detector. YOLO is 10 mAP more accurate than the fast version while still well above real-time in speed.

Figure 1: Real-Time Systems on PASCAL VOC

4.1 Comparison to Other Real-Time Systems

Object detection algorithms need to be fast for real-time operation. For instance, the minimal accepted performance is to successfully process 30 images per second to be able to analyze a real-time video stream of 30FPS. The only systems with such performance are the DPM (30Hz, 100Hz) running on GPU and YOLO (Fast, Normal). As shown in Fig. 1, Fast YOLO is the fastest known algorithm to detect objects in the Pascal VOC 2007, 2012 Dataset with a speed of 155FPS and performance measured in "mean average precision" mAP of 52.7 %. This performance metric is used with object detection algorithms to quantize the precision recall of each class averaged. Compared to YOLO, DPM detectors are of poor mAP and are not as fast as YOLO. However, YOLO is not of the highest mAP when we check detectors that cannot work in real-time even with the increase of 10% in mAP when using YOLO compared to Fast YOLO. Different Fast RCNN are tested with different bounding box generation methods. RCNN Minus R uses static bounding box locations, while the conventional

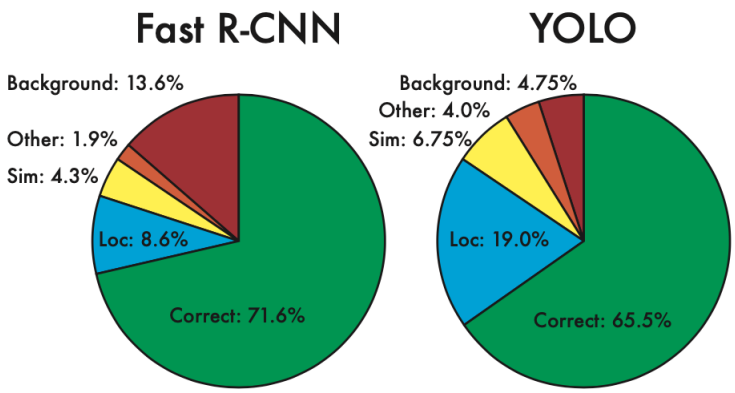


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

Figure 2: Error Analysis: Fast R-CNN vs. YOLO

Fast RCNN uses selective search which is the slowest in the performed tests (taking 2 seconds per image). Using neural networks instead of selective search boosts speed up to 7FPS and using a smaller network of less accuracy bumps the speed of Fast RCNN into 18 FPS which is still less than the Real-time needed performance. The table shows performance and speed of different object detection algorithms and on which datasets are they trained. The VGG-16 version of Faster R-CNN is 10 mAP higher but is also 6 times slower than YOLO. The Zeiler- Fergus Faster R-CNN is only 2.5 times slower than YOLO but is also less accurate.

4.2 VOC 2007 Error Analysis

The study of errors in the detection algorithms is used to identify the weak points of each detection system and how to overcome them by combining different models together. To study the error, the following classification of model results is used:

- Correct: class is correct, and Intersection over union (IOU) is

greater than 0.5.

- Localization: class is correct, and IOU is greater than 0.1 but less than 0.5.
- Similar: class is similar, and IOU is greater than 0.1.
- Other: class is wrong, and IOU is greater than 0.1.
- Background: for all IOU less than 0.1.

It can be seen in Fig. 2 that Fast RCNN had a higher correct percentage than YOLO, but as we have seen earlier this higher mAP is paid for in performance speed. The YOLO system had most of its not correct classifications as localization errors (which means the system got the class correct but couldn't accurately determine its position in the image). Also, Fast RCNN had a large background error which are false positives with no object in the image. Notice that from this error analysis we can see that if we use Fast RCNN first, then used YOLO to correct the results of the Fast RCNN, we will boost the accuracy of this model greatly as seen in the next section.

	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**

Table 2: Model combination experiments on VOC 2007. We examine the effect of combining various models with the best version of Fast R-CNN. Other versions of Fast R-CNN provide only a small benefit while YOLO provides a significant performance boost.

Figure 3: Combining Fast R-CNN and YOLO

4.3 Combining Fast R-CNN and YOLO

For every bounding box that R-CNN predicts, YOLO is used to check if it predicts a similar box. If it does, that prediction takes a boost based on the probability predicted by YOLO and the overlap between the two boxes. The best Fast R-CNN model achieves a mAP of 71.8% on the VOC 2007 test set. When combined with YOLO, its mAP increases by 3.2% to 75.0% as shown in Fig. 3. This boost in performance is not just a conventional ensemble because YOLO has errors in different categories from those made by Fast RCNN, so YOLO rectifies the errors made by Fast RCNN. This approach however does not benefit from the speed of YOLO as Fast RCNN is run first and then YOLO is applied separately.

4.4 VOC 2012 Results

As shown in Fog. 4, on this dataset, the YOLO algorithm has a mAP of 57.9% which is considered low compared to other current object detection algorithms. This reduction in mAP is justified when each Average Precision is calculated for each class alone. It is noted that YOLO struggles with smaller objects compared to other algorithms like Faster RCNN. On categories like bottle, sheep, and tv/monitor YOLO scores 8-10% lower than R-CNN or Feature Edit. However, on other categories like cat and train YOLO achieves higher performance. When we combine YOLO and Fast RCNN, this model is boosting the performance of Fast RCNN with 2.3% and is considered one of the top 4 performing detection methods.

4.5 Generalizability: Person Detection in Artwork

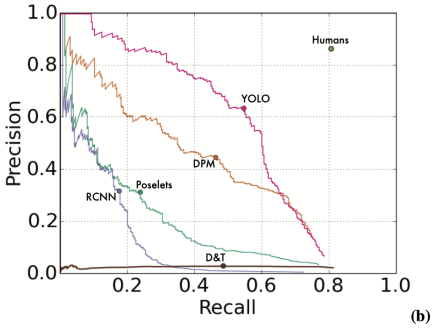
Because real life scenarios can lead the detection algorithm to deal with images from a different source than its training dataset, this section deals with testing detection algorithms on new artwork data different from the VOC 2007 dataset previously trained. Picasso Dataset and People - Art Dataset are used in this testing. Fig. 5 shows that although RCNN had a high AP on VOC 2007-person classification, its performance reduced dramatically on the Picasso dataset. This happened due to the RCNN usage of selective search which is more suited to natural images and not person detection as the bounding

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

Table 3: PASCAL VOC 2012 Leaderboard. YOLO compared with the full `comp4` (outside data allowed) public leaderboard as of November 6th, 2015. Mean average precision and per-class average precision are shown for a variety of detection methods. YOLO is the only real-time detector. Fast R-CNN + YOLO is the forth highest scoring method, with a 2.3% boost over Fast R-CNN.

Figure 4: VOC 2012 Results

boxes proposed do not capture the whole person and only sees a small region at a time. DPM has a minimal degradation in AP because of its spatial feature preservation but already starts with a low AP in the VOC 2007 dataset. YOLO on the other hand had good performance on VOC 2007 and with minimal degradation when applied to artwork datasets. YOLO models the size and shape of objects, as well as relationships between objects and where objects commonly appear.



(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.

Figure 5: Generalization results on Picasso and People-Art datasets.

Figure 5: Generalization Results

5 Real-Time Detection in the Wild

Although YOLO is an image-based object detection system, because of its fast and accurate detection, it can be used in computer vision as the camera captures a video stream from which images are extracted and processed by YOLO. This provides an interactive tracking system because of the fast response of the YOLO algorithm.

6 Conclusion

YOLO, the object detection system designed to be fast and easy to train directly on full images, is trained on a loss function that directly corresponds to detection performance and the entire model is trained jointly, unlike classifier-based approaches of separate modules. A faster version of YOLO (Fast YOLO) is proven to be the fastest object detector although with a reduction in its accuracy compared to YOLO. "You only look once" is the intuitive approach to object detection as it also generalizes well to new domains of data from other sources than that of training.