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

1. Introduction:

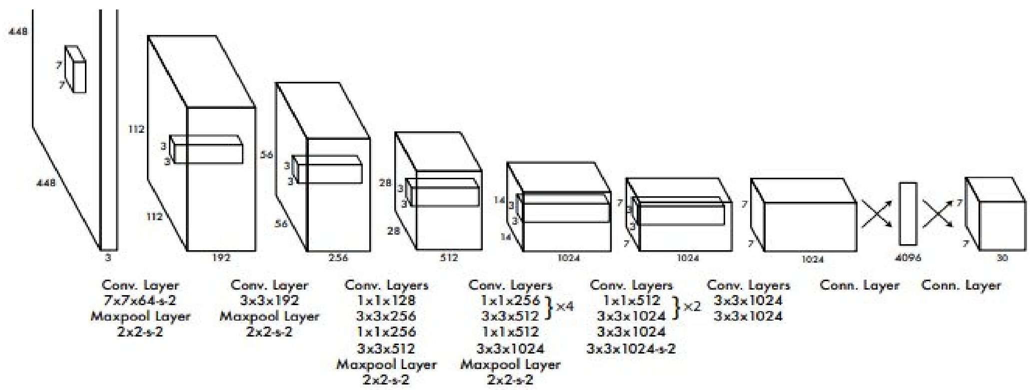
- Previous models- take a classifier, use it at different positions
  - DPM- sliding window with classifier
  - RCNN- potential bounding box, classify and refine boxes
- YOLO- single regression problem with convolutional layers
  - Fast
  - Reasons globally about the images, fast RCNN fails to do it
  - Learns generalizable representations
- Still lags behind state-of-the-art methods in terms of accuracy

2. Unified detection:

- Divide the image in (S*S) number of grids
- Each grid cell predicts B bounding boxes and confidence scores for those boxes
- confidence= pred * IOU
- Each bounding box consists of 5 predictions: x, y, w, h, and confidence
- Each grid cell also predicts C conditional class probabilities,
- (x,y)- center of the bounding box
- one set of class probabilities per grid cell, regardless of the number of boxes B
- in test, conditional class probabilities * individual box confidence predictions

2.1. Network design

- Network architecture:



- Inspired by GoogLeNet, 24 convolutional layers, 2 fully connected layers
- Instead of the inception modules used by GoogLeNet, we simply use 1 × 1 reduction layers followed by 3 × 3 convolutional layers,
- Final output shape: 7*7*30

2.2. Training

- Pre-train on imagenet 1000 class dataset
- First 20 pre-trained convolutional layers kept intact, custom fully connected layers added
- Train for 1 week, acc: 88% top 5 on imagenet 2012
- Converted for detection
- added four convolutional layers and two fully connected layers with randomly initialized weights
- increase the input resolution from 224*224 to 448*448
- final layer predicts both class probabilities and bounding box coordinates
- normalize the bounding box width and height by the image width and height so that they fall between 0 and 1
- 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
- Final layer- linear activation, other layers- leaky relu
- Optimize- sum-squared error
- Problem: sum-squared error weights can't differentiate object localization error from classification error.
- Solution: 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 λ coord = 5 and λ noobj = .5

- Problem: 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
- Solution: we predict the square root of the bounding box width and height instead of the width and height directly
- While training, one bounding box (highest IOU) for one class
- Loss function:

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$
$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right]$$
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)$$

where $\mathbb{1}_i^{\text{obj}}$ denotes if object appears in cell i and $\mathbb{1}_{ij}^{\text{obj}}$ denotes that the jth bounding box predictor in cell i is "responsible" for that prediction.

- This error penalizes:
  - Classification error
  - Bounding box error
- train the network for about 135 epochs on the training and validation data sets from PASCAL VOC 2007 and 2012.

- When testing on 2012 we also include the VOC 2007 test data for training.
- Throughout training we use a batch size of 64, a momentum of 0.9 and a decay of 0.0005
- learning rate schedule is as follows: For the first epochs we slowly raise the learning rate from 10**(−3) to 10**(−2) . If we start at a high learning rate our model often diverges due to unstable gradients. We continue training with 10−2 for 75 epochs, then 10**(−3) for 30 epochs, and finally 10**(−4) for 30 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
- 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

2.3. Inference

- Testing: requires one network evaluation, so it runs faster
- On PASCAL VOC the network predicts 98 bounding boxes per image and class probabilities for each box
- some large objects or objects near the border of multiple cells can be well localized by multiple cells. Non-maximal suppression can be used to fix these multiple detections
- non-maximal suppression adds 2- 3% in mAP

2.4. Limitations of yolo

- Each grid can predict 2 bounding boxes at max

- Struggles in images with unusual aspect ratio or shapes
- 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.

3. Comparison of other detection system

Other systems: extract feature >> classifier or localizer in sliding fashion

3.1. DPM:

- Sliding window
- disjoint pipeline to extract static features, classify regions, predict bounding boxes for high scoring regions

3.2. R-CNN:

- region proposals instead of sliding windows
- Selective Search generates potential bounding boxes
- a convolutional network extracts features
- an SVM scores the boxes
- a linear model adjusts the bounding boxes
- non-max suppression eliminates duplicate detections
- Very slow, ~40 sec per image
- YOLO shares some similarities with R-CNN

3.3. 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

- speed up HOG computation, use cascades, and push computation to GPUs
- only 30Hz DPM actually runs in real-time
- Instead of trying to optimize individual components of a large detection pipeline, YOLO throws out the pipeline entirely and is fast by design

3.4. Deep multibox:

- train a convolutional neural network to predict regions of interest instead of selective search
- MultiBox cannot perform general object detection and is still just a piece in a larger detection pipeline

3.5. Overfeat:

- train a convolutional neural network to perform localization and adapt that localizer to perform detection
- OverFeat efficiently performs sliding window detection but it is still a disjoint system
- 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

3.6. Multigrasp:

- 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

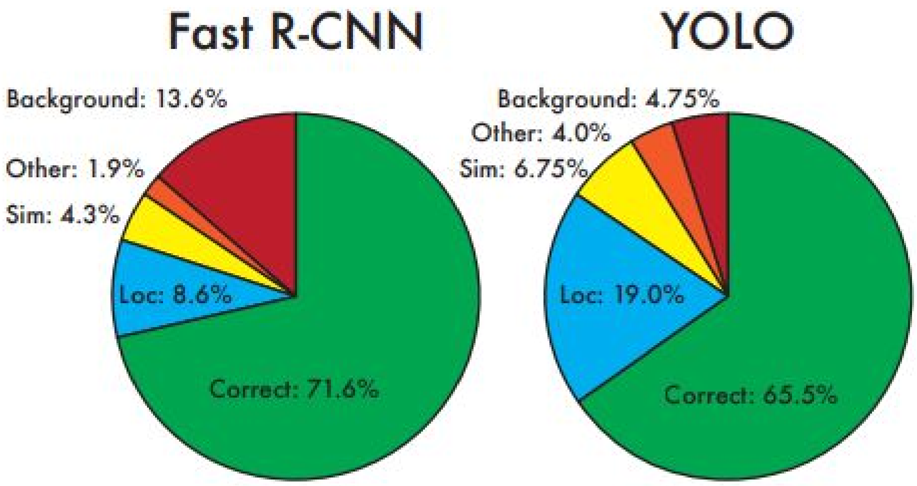
- It doesn’t have to estimate the size, location, or boundaries of the object or predict it’s class, only find a region suitable for grasping
- YOLO predicts both bounding boxes and class probabilities for multiple objects of multiple classes in an image

4. Experiments

4.1. Comparison to Other Real-Time Systems

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

4.2. VOC 2007 Error Analysis



- Correct: correct class and IOU > .5
- Localization: correct class, .1 < IOU < .5
- Similar: class is similar, IOU > .1
- Other: class is wrong, IOU > .1
- Background: IOU < .1 for any object

4.3. 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**

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.

4.4. 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

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.

4.5. Generalizability: Person Detection in Artwork

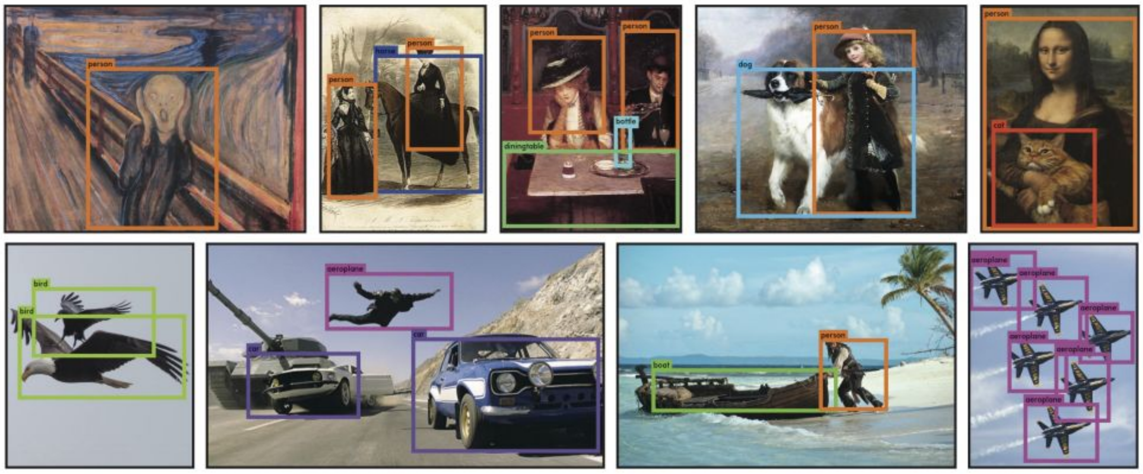


Figure 6: Qualitative Results. YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.

5. Real-Time Detection In The Wild

- The resulting system is interactive and engaging.
- 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

6. 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
- YOLO pushes the state-of-the-art in real-time object detection