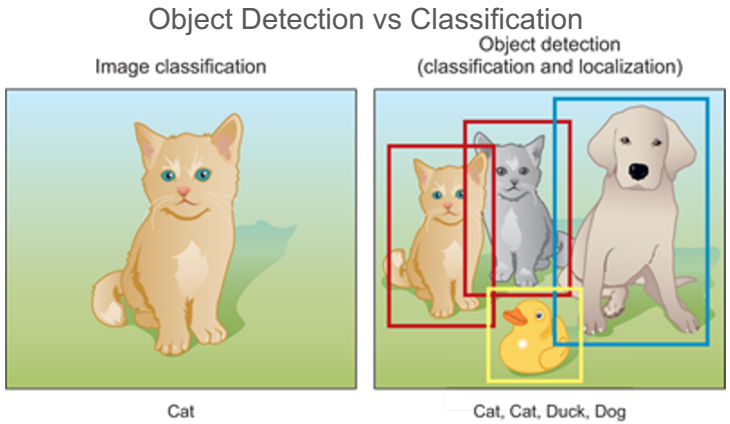
You Only Look Once (YOLO): Unified, Real-Time Object Detection

Presenter: Shivang Singh

Sept 2nd, 2021

Problem Addressed: Object Detection

- Object detection is the problem of both locating **AND** classifying objects
- Goal of YOLO algorithm is to do object detection both fast **AND** with high accuracy

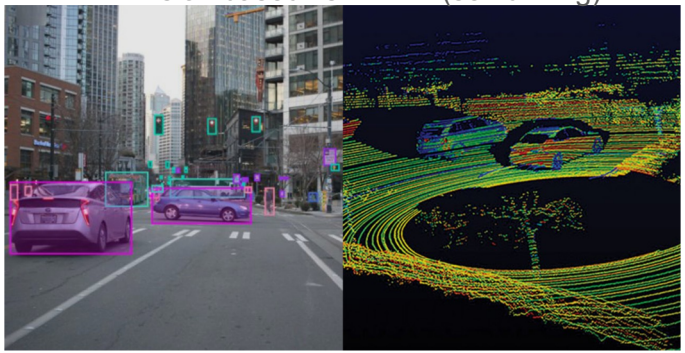


"Deep Learning for Vision Systems" (Elgendy)

Importance of Object Detection for Robotics

- Visual modality is very powerful
- Humans are able to detect objects and do perception using just this modality in real time (not needing radar)
- If we want responsive robot systems that work in real time (without specialized sensors) almost real time vision based object detection can help greatly

Vision based vs LIDAR (self driving)

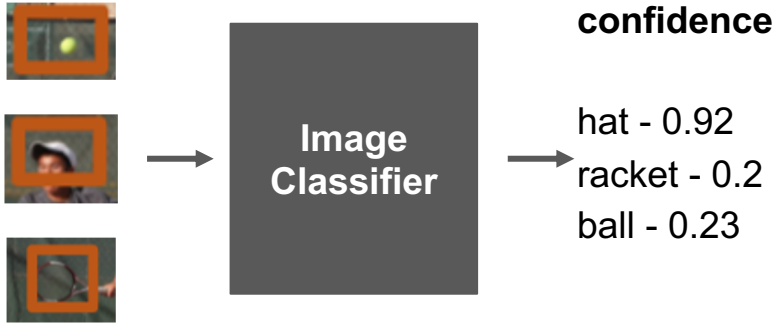


Tesla Investor Day Presentation

Previous Object Detection Paradigm

This pipeline was used in nearly all SOTA Object Detection prior:

Step 1: Scan the image to generate candidate bounding boxes



Step 2: Run the bounding box through a classifier

Step 3: Conduct post-processing (filtering out redundant bounding boxes)

Diagram developed by presenter

Key Insights

Previous Approaches

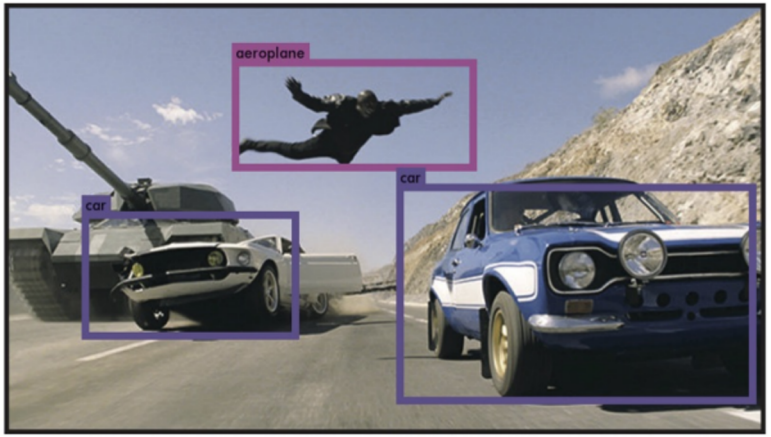
- A separate model for generating bounding boxes and for classification (more complicated model pipeline)
- Need to run classification many times (expensive computation)
- Looks at limited part of the image (lacks contextual information for detection)

YOLO algorithm

- A single neural network for localization and for classification (less complicated pipeline)
- Need to inference only once (efficient computation)
- Looks at the entire image each time leading to less false positives (has contextual information for detection)

Formal Problem Setting

- Given an image generate bounding boxes, one for each detectable object in image
- For each bounding box, output 5 predictions: x, y, w, h, confidence. Also output class



- x, y (coordinates for center of bounding box)
- w,h (width and height)
- confidence (probability bounding box has object)
- class (classification of object in bounding box)

Related Work

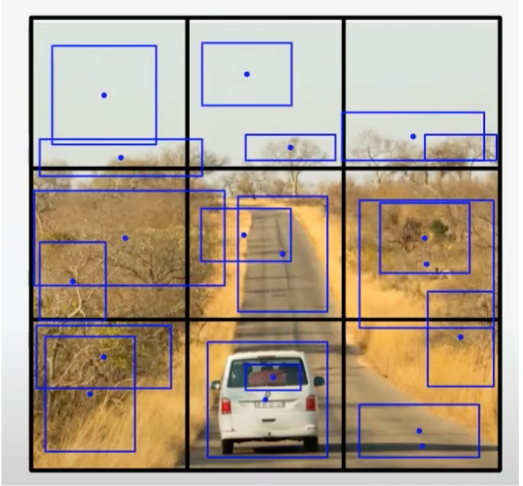
- R-CNN or Region Based Convolutional Network (Girshick et al. 2014):
  - Used the sliding window approach from earlier, with Selective Search, a smarter way to select candidates (which means there is less computation)
  - Still feeds a limited part of the image to the classifier
  - Drawbacks: Large pipeline, slow, too many false positives
- Fast and Faster R-CNN:
  - Optimize parts of the pipeline described earlier
  - Drawbacks: loses accuracy
- Deep Multibox (Szegedy et. al 2014):
  - Train a CNN to find areas of interest
  - Drawbacks: Doesn’t address classification only localization

Related Work

- MultiGrasp (Redmon et. al 2014)
  - Similar to YOLO
  - A much simpler task (only needs to predict object not multiple objects)

YOLO overview

- First, image is split into a SxS grid
- For each grid square, generate B bounding boxes
- For each bounding box, there are 5 predictions: x, y, w, h, confidence



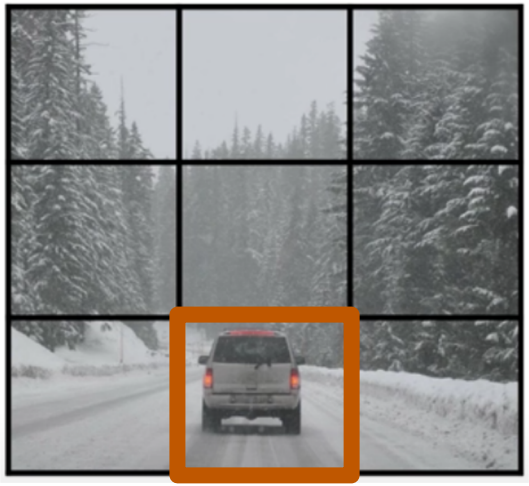
S = 3, B = 2

YOLO Training

- YOLO is a regression algorithm. What is X? What is Y?
- X is simple, just an image width (in pixels) * height (in pixels) * RGB values
- Y is a tensor of size S * S * (B * 5 + C)
- B*5 + C term represents the predictions + class predicted distribution for a grid block

For each grid block, we have a vector like this. For this example B is 2 and C is 2

ρ_1
b_x_1
b_y_1
b_h_1
b_w_1
ρ_2
b_x_2
b_y_2
b_h_2
b_w_2
c_1
c_2

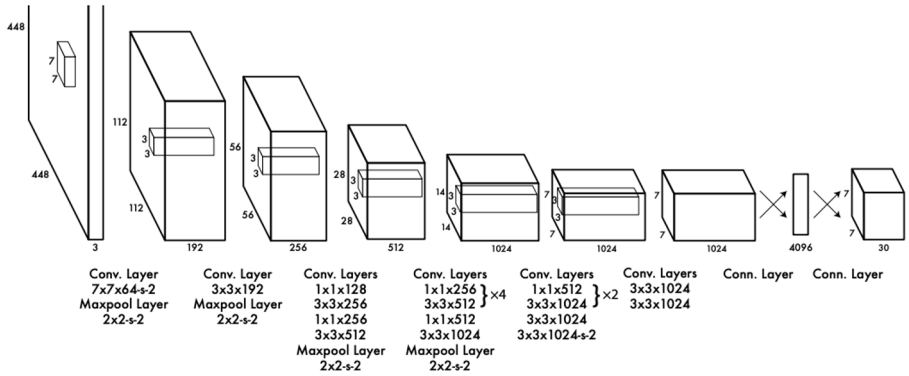


GT label example:

1
b_x_1
b_y_1
b_h_1
b_w_1
0
?
?
?
?
c_1 = 1
c_2 = 0

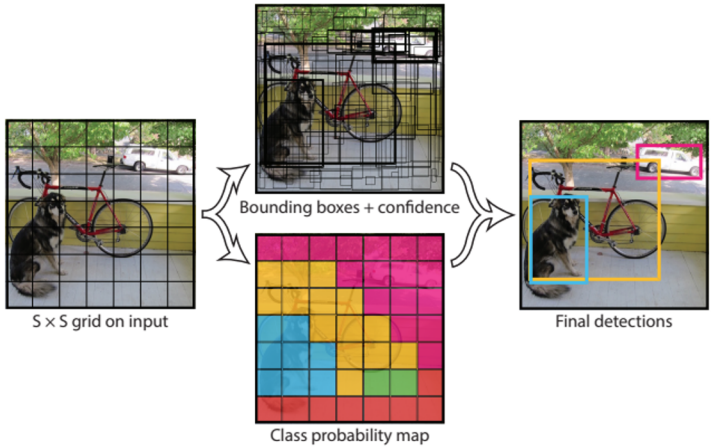
YOLO Architecture

- Now that we know the input and output, we can discuss the model
- We are given 448 by 448 by 3 as our input.
- Implementation uses 7 convolution layers
- Paper parameters: S = 7, B = 2, C = 20
- Output is S*S*(5B+C) = 7*7*(5*2+20) = 7*7*30



YOLO Prediction

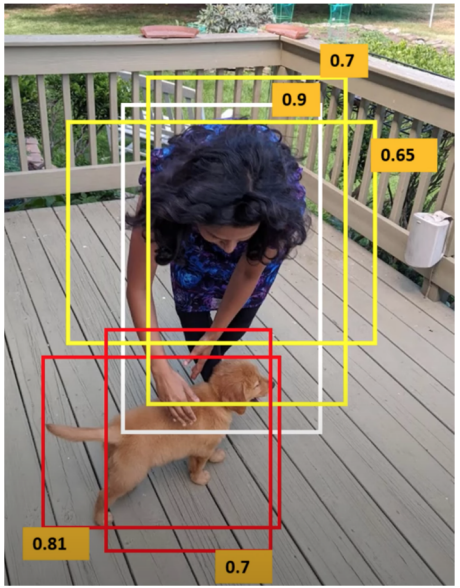
- We then use the output to make final detections
- Use a threshold to filter out bounding boxes with low P(Object)
- In order to know the class for the bounding box compute score take argmax over the distribution Pr(Class|Object) for the grid the bounding box's center is in



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

Non-maximal suppression

- Most of the time objects fall in one grid, however it is still possible to get redundant boxes (rare case as object must be close to multiple grid cells for this to happen)
- Discard bounding box with high overlap (keeping the bounding box with highest confidence)
- Adds 2-3% on final mAP score



YOLO Objective Function

- For YOLO, we need to minimize the following loss
- Sum squared error is used

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

Coordinate Loss: Minimize the difference between x,y,w,h pred and x,y,w,h ground truth. ONLY IF object exists in grid box and if bounding box is resp for pred

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

Confidence Loss: Loss based on confidence ONLY IF there is object

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

No Object Loss based on confidence if there is no object

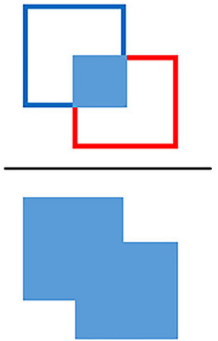
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

Class loss, minimize loss between true class of object in grid box

Experimental Setup

- Authors compare YOLO against the previous work described above on PASCAL VOC 2007, and VOC 2012 as well as out of domain art dataset
- Correct if IOU metric above .5 and class is correct
- Use two performance metrics:
  - mAP score: mean average precision
  - FPS: frames per second
- Add FAST YOLO: which has less parameters

$$IoU = \frac{area\ of\ intersection}{are\ of\ union} =$$



Experimental Results

- Baseline YOLO outperform real time detectors by large amount
- Do better than most less than real time as well

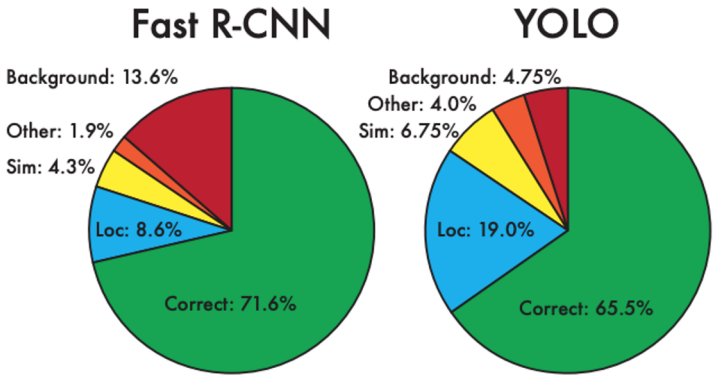
Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Experimental 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

Experimental Results - Error Analysis

- Makes far less background errors (less likely to predict false positives on background)
  - IOU is VERY small with any ground truth label
- But far more localization errors
  - Correct class, IOU is somewhat small



Localization error

Background error

Experimental Results - Out of Domain

- Ran YOLO + competitors (trained on natural images) on art
- Does well on artistic datasets where more having global context greatly helps

	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	

Discussion of Results

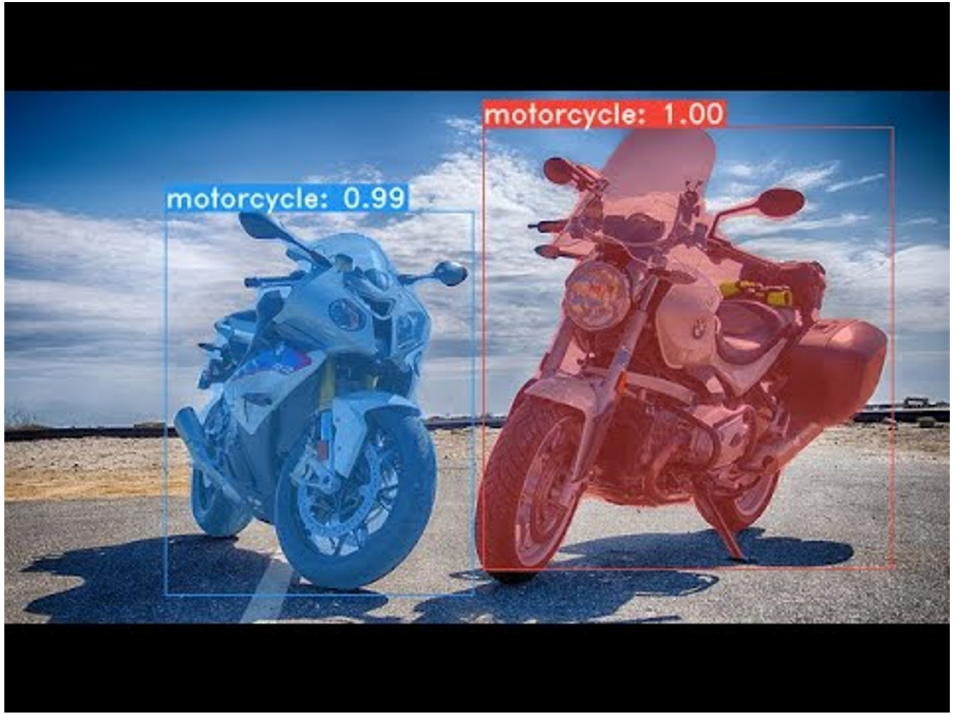
- Pro: YOLO is a lot faster than the other algorithms for image detection
- Pro: YOLO's use of global information rather than only local information allows it to understand contextual information when doing object detection
  - Does better in domains such as artwork due to this
- Con: YOLO lagged behind the SOTA models in object detection
  - This is attributed to making many localization errors and unable to detect small object

Critique / Limitations / Open Issues

- Performance lags behind SOTA
- Requires data to be labeled with bounding boxes, hard to collect for many classes
  - Previous work could generalize better since it used image classifier
  - 2014 COCO dataset (very large dataset) addressed this somewhat
- Regarding experiments: number of classes predicted is very limited
  - Not convinced that YOLO v1 is generalizable
- Confidence output of YOLO not confidence of class but P(Object), lowers interpretability
- Another limitation of YOLO is that it imposed spatial constraints on the objects in the image since only B boxes can be predicted on an SxS grid
- Since the architecture only predicts boxes, this might make it less useful for irregular shapes

Future Work for Paper / Reading

- One extension of this work would be to look at image segmentation and see if the insights carry over
  - YOLOACT (Boyla et al 2019): Real time image segmentation
- YOLO has been upgraded 2 times
  - Solves a lot of issues relating to detecting small objects, generalizability, and localization



YOLOACT example

Extended Readings

- YOLO v2 (https://arxiv.org/abs/1506.02640) (extends on the work greatly) (Redmond et al 2016)
  - Deals with the generalizability problem, has 9000 classes
  - Class probability distribution per bounding box, not per grid
  - High resolution classifier (finetune on high resolution)
  - Batch norm
  - Trained on MSCOCO (released after YOLO v1 paper)
- YOLO v3 (https://arxiv.org/abs/1804.02767)
  - "Incremental Improvement"
  - Uses independent logistic classifiers for class
    - Allows for more specificity in classes

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓
anchor boxes?				✓	✓			
new network?					✓	✓	✓	✓
dimension priors?						✓	✓	✓
location prediction?						✓	✓	✓
passthrough?							✓	✓
multi-scale?							✓	✓
hi-res detector?								✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Table 2: The path from YOLO to YOLOv2. Most of the listed design decisions lead to significant increases in mAP. Two exceptions are switching to a fully convolutional network with anchor boxes and using the new network. Switching to the anchor box style approach increased recall without changing mAP while using the new network cut computation by 33%.

Summary

- Object detection is the problem of detecting multiple objects in an image
- Almost real time object detection can make highly responsive robot systems without complex sensors
- Prior work relies on a large architecture with numerous parts to optimize
- YOLO proposes a unified architecture, which does all the tasks in one model and by one inference over the entire image
- They show enormous speed improvement and show that they can beat most other prior work in terms of mAPs