YOLO v1/v2/v3

Real-time object detector

Outline

- Detection systems 2016
- YOLO v1
- YOLO v2
- YOLO v3

Detection systems in 2016

- Deformable Parts Models (DPM)
    - Uses a sliding window approach, classifies at each window location
    - “Brute force method”
    - Slow
- R-CNN
    - Generates potential bounding boxes, then CNN to compute feature vectors, then feature vectors are fed into a SVM classifier(s), then post-processing…
    - Complex pipeline, each image takes multiple seconds to classify

Sliding window approach

Why YOLO?

- Detection systems in 2016:
    - Deformable Parts Models (DPM)
        - Uses a sliding window approach and classifies at each step
        - “Brute force method” => inefficient
        - Doesn’t retain contextual information
    - R-CNN
        - Generates potential bounding boxes, then CNN to compute feature vectors, then feature vectors are fed into a SVM classifier(s), then post-processing...
        - Complex pipeline, each image takes multiple seconds to classify each image

YOLO v1

- You Only Look Once
- Adresses DPM and R-CNN’s shortcomings
  - Much faster
    - 25 ms latency
  - Encodes contextual information about classes
    - Fewer misclassifications where background are treated as objects
- Also, it generalizes better
  - Trained on natural images, yet it detects well on artwork

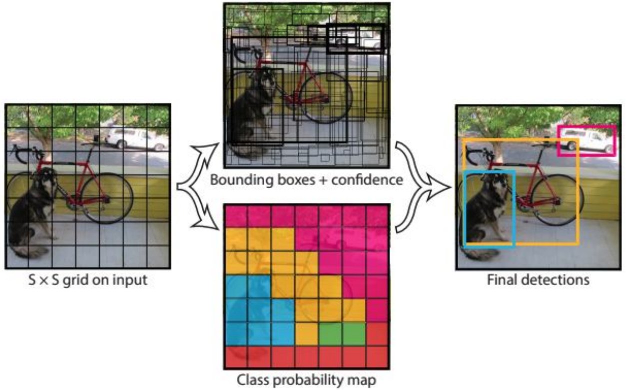
YOLO v1 - How does it work?

“A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes”

…

“Trained on a loss function that directly corresponds to detection performance and the entire model is trained jointly.“

Reframing object detection into a loss function

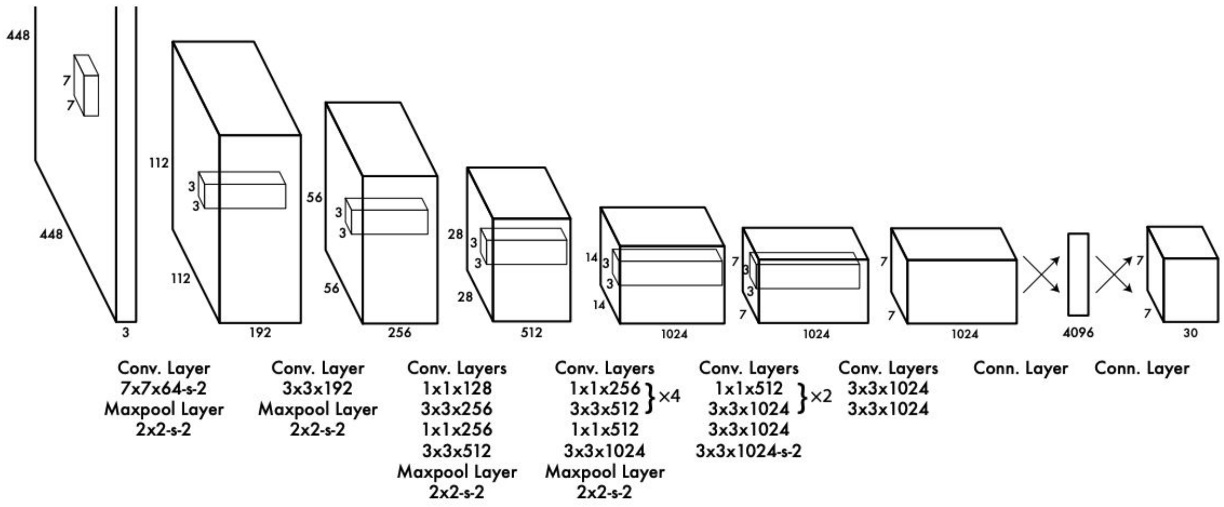


=>

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

YOLO v1 - Architecture

- Inspired by GoogLeNet
- 23 convolutional layers followed by 2 fully connected layers



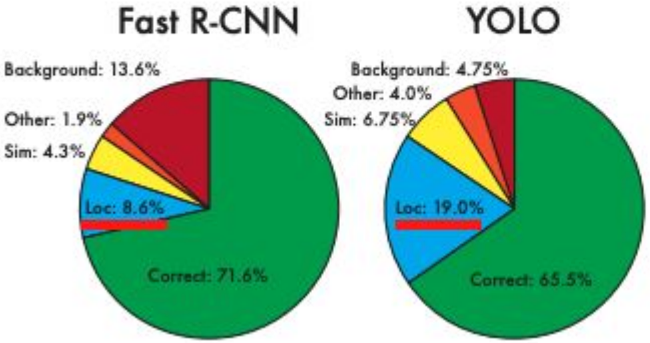
Results - Pascal VOC 2007

- Fast, yet fairly good accuracy

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

Results - Pascal VOC 2007

- Struggles with localization



- … and with small objects that appears in groups

YOLO v2

- Addresses YOLO v1 issues
- Better
- Faster
- Stronger

YOLO v2 - Better

- Implements a lot of techniques to increase the mean average precision

	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**

- E.g. adding passthrough layer so the model can use fine grained features
  - The model detects smaller objects better

YOLO v2 - Faster

- New architecture
- Darknet-19
- Reduces the amount of operations needed for a forward pass
  - YOLO v1 - 8.52 billion operations - 88% accuracy
  - YOLO v2 - 5.82 billion operations - 91.2% accuracy

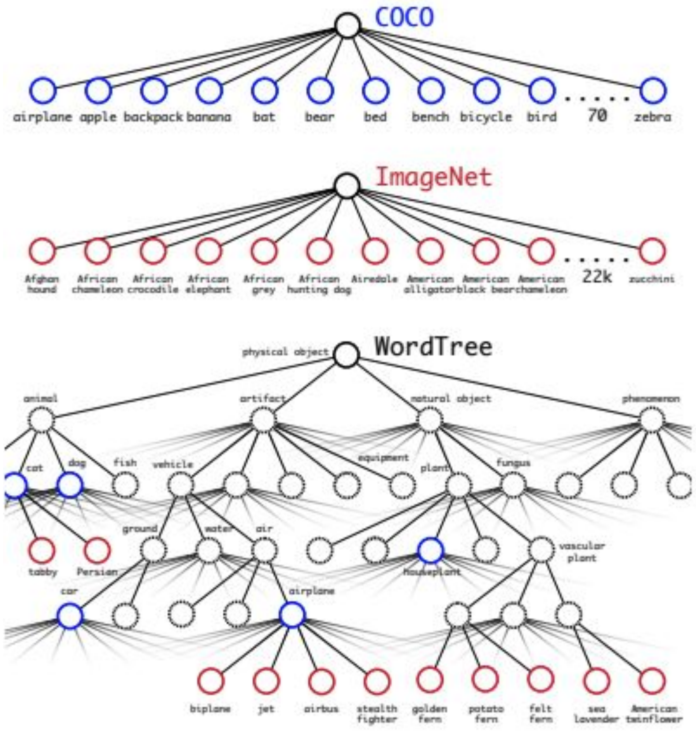
Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Table 6: Darknet-19.

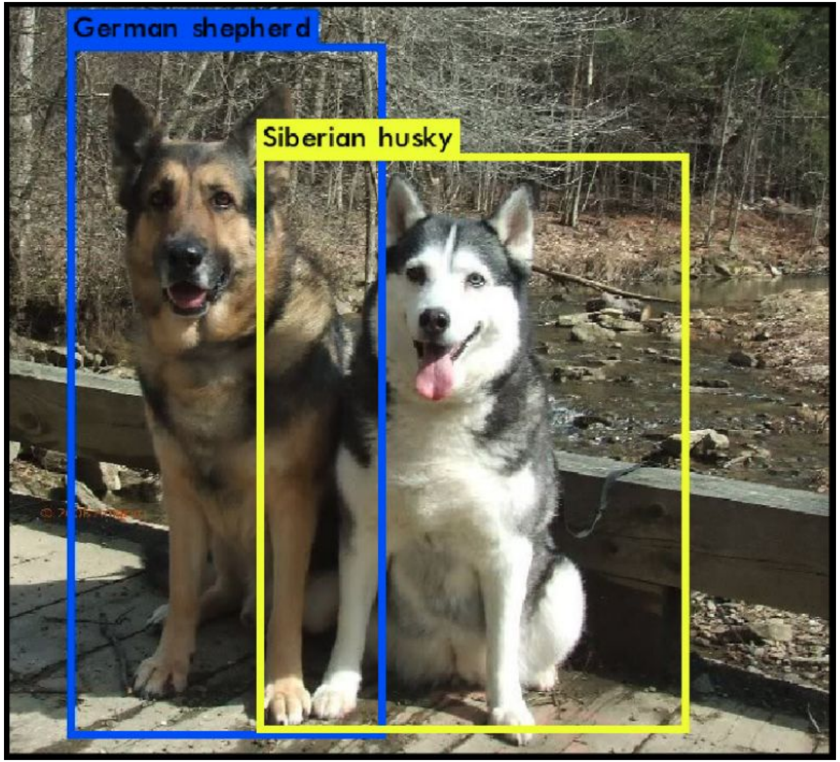
YOLO v2 - Stronger

- YOLO v1 could not classify many different objects
  - Trained on Pascal VOC, only 20 classes
  - Object detection datasets are limited
    - Object detection: Thousands to hundreds of thousands of images, few classes
    - Classification: Millions of images, tens or hundred of thousands classes
- YOLO v2
  - Combines COCO (detection) and ImageNet (classification)
    - Can classify more than 9000 classes

YOLO v2 - Combined dataset

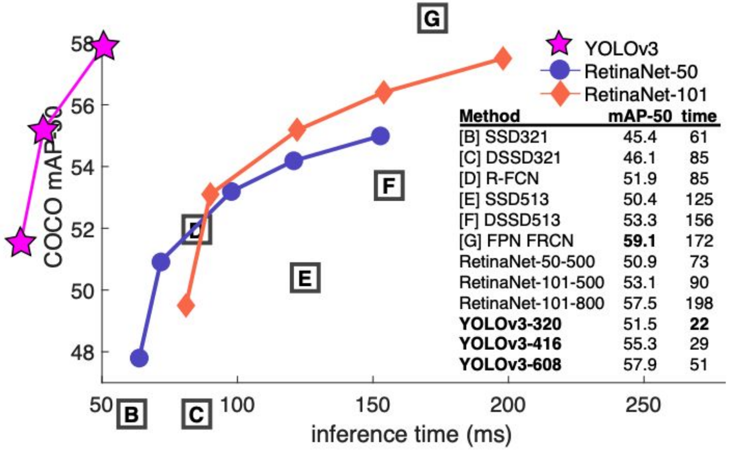


YOLO v2 - more specific classifications



YOLO v3

- "... nothing super interesting, just a bunch of small changes that make it better"
    - E.g better bounding box prediction, tweak an hyperparameter for class prediction etc.
    - New model, Darknet-53



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

YOLO v3 in action