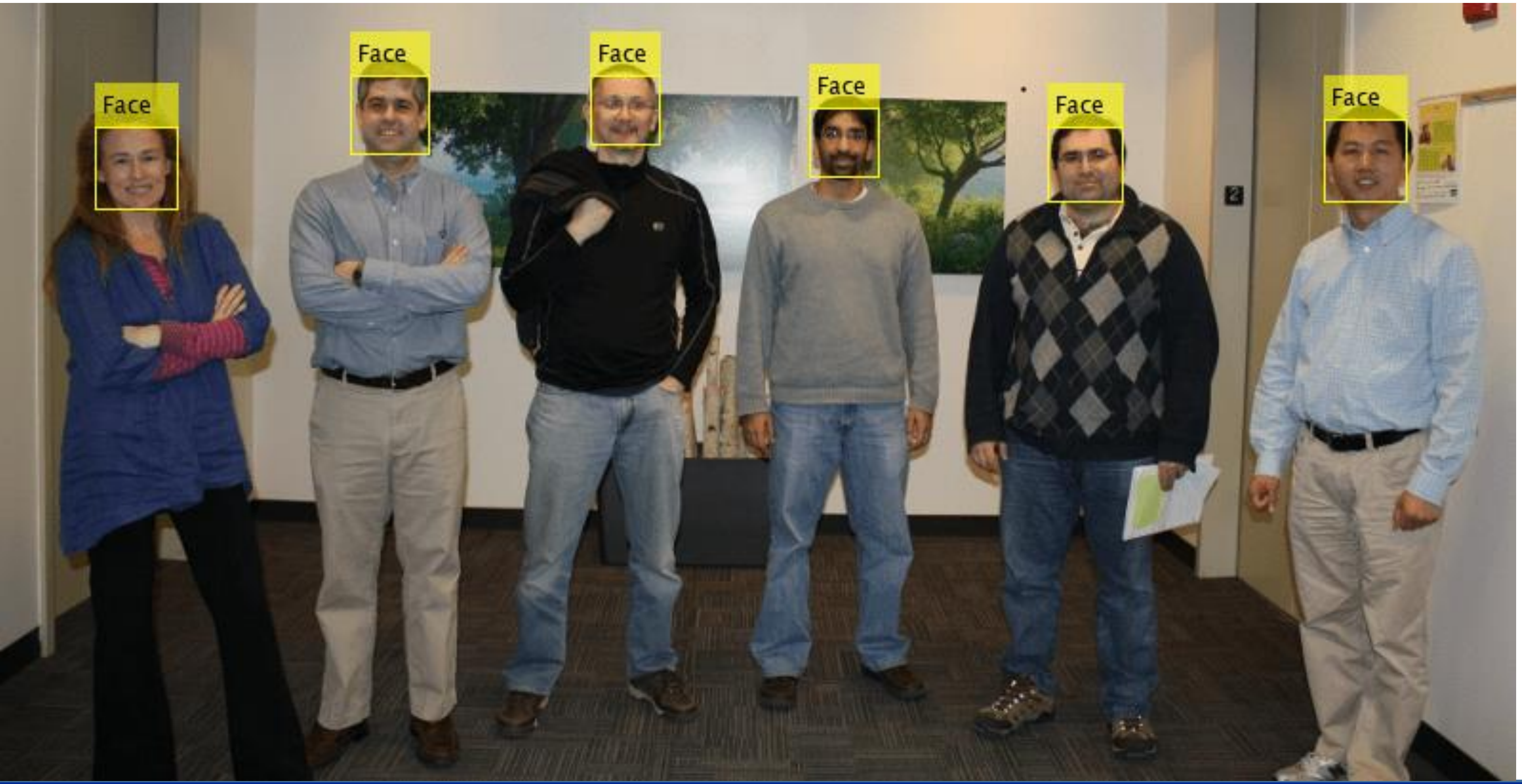
Computer Vision

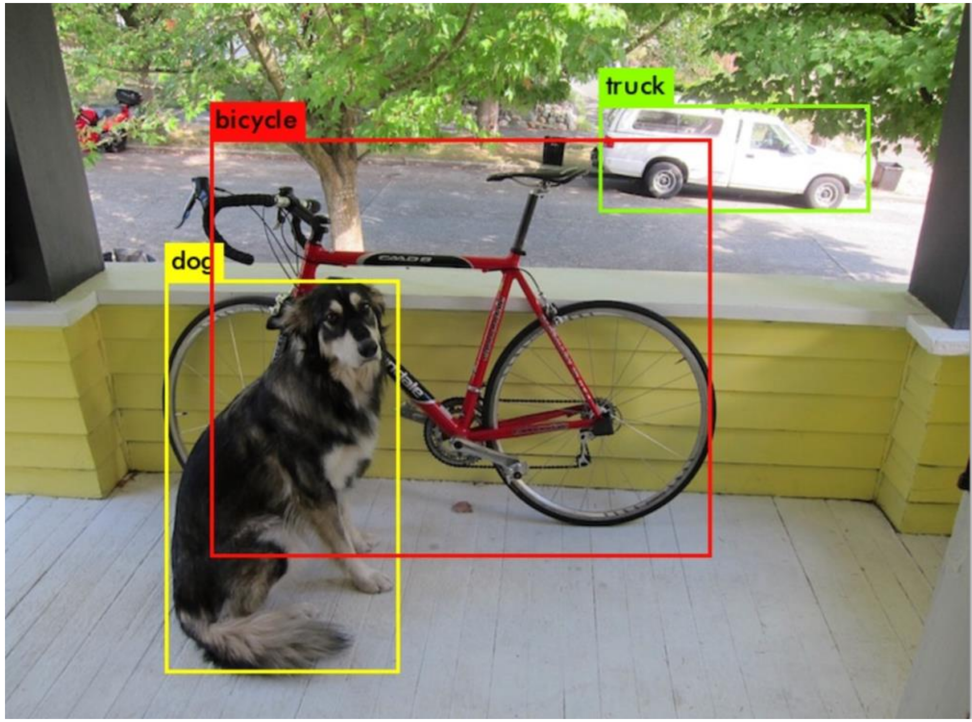
YOLO v1

Pham Viet Cuong
Dept. Control Engineering & Automation, FEEE
Ho Chi Minh City University of Technology

Face Detection: Viola - Jones Algorithm

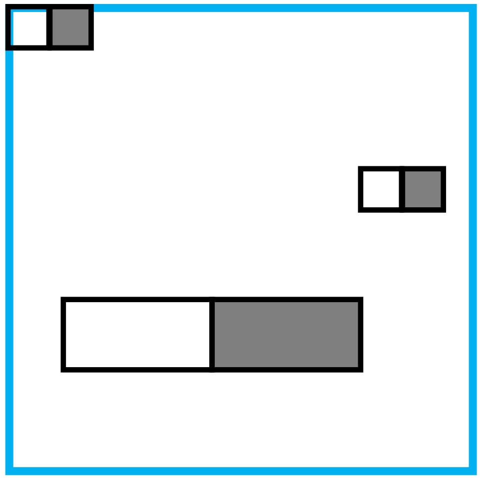
- Object detection problem
  - Single object detection
  - Multiple object detection
  - Bounding box(es)
  - Class(es) of object(s)

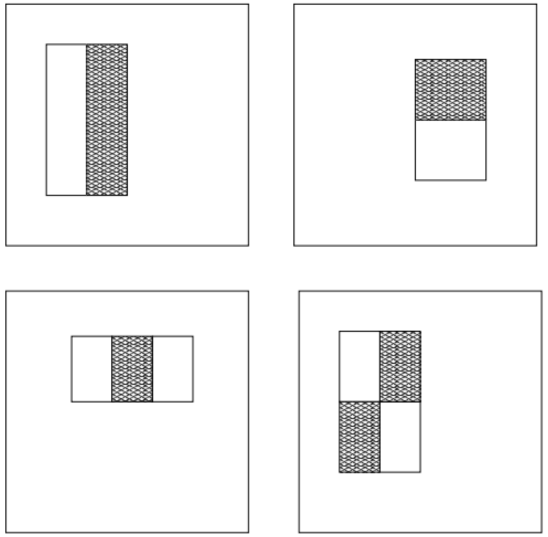




Face Detection: Viola - Jones Algorithm

- Haar-like features
  - Window size: 24x24
  - Type
  - Position
  - Size
  - ~ 160K features
- Features usefulness?





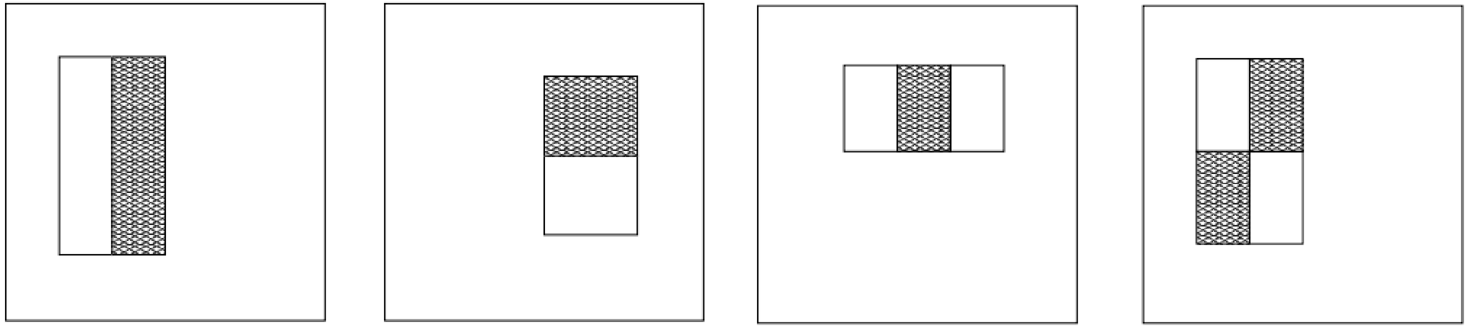
Face Detection: Viola - Jones Algorithm

- Feature selection
  - Positive & negative sets
    10K examples/set
  - Weak classifiers

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

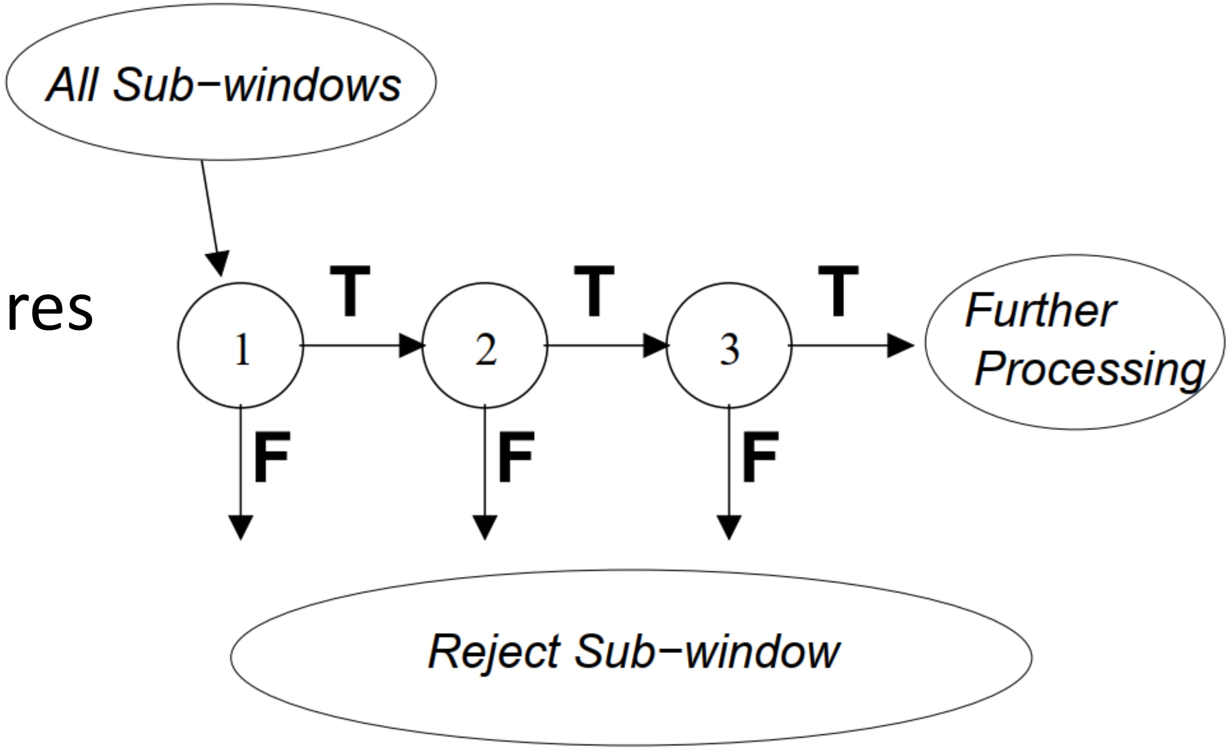
24x24 window feature

  - Objective: min # examples misclassified
  - ~ 6K features selected



Face Detection: Viola - Jones Algorithm

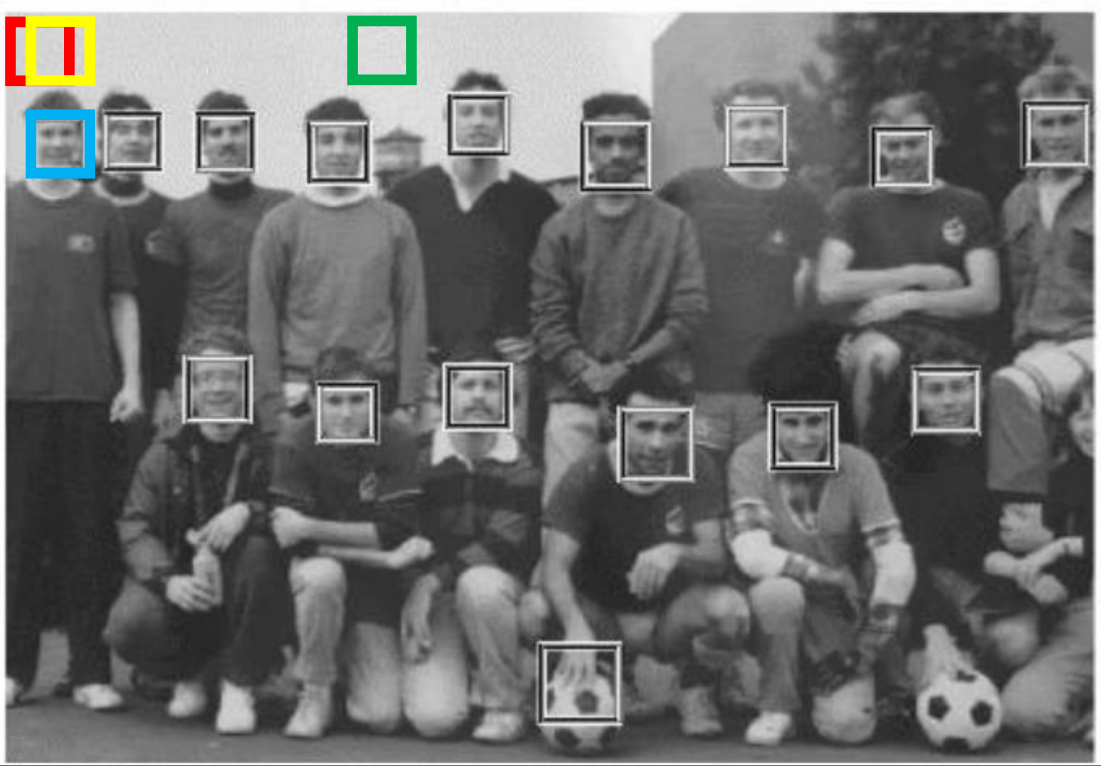
- Trade-off
  - More features (higher detection rate, lower false positive rate)
  - More computational complexity
- Cascade structure
  - 6061 features
  - 38 stages
  - First 5 layers: 1, 10, 25, 25, 50 features
  - Average: 10 feature evaluations per window
  - 15 – 600 times faster than others
  - Negative examples? false positive examples

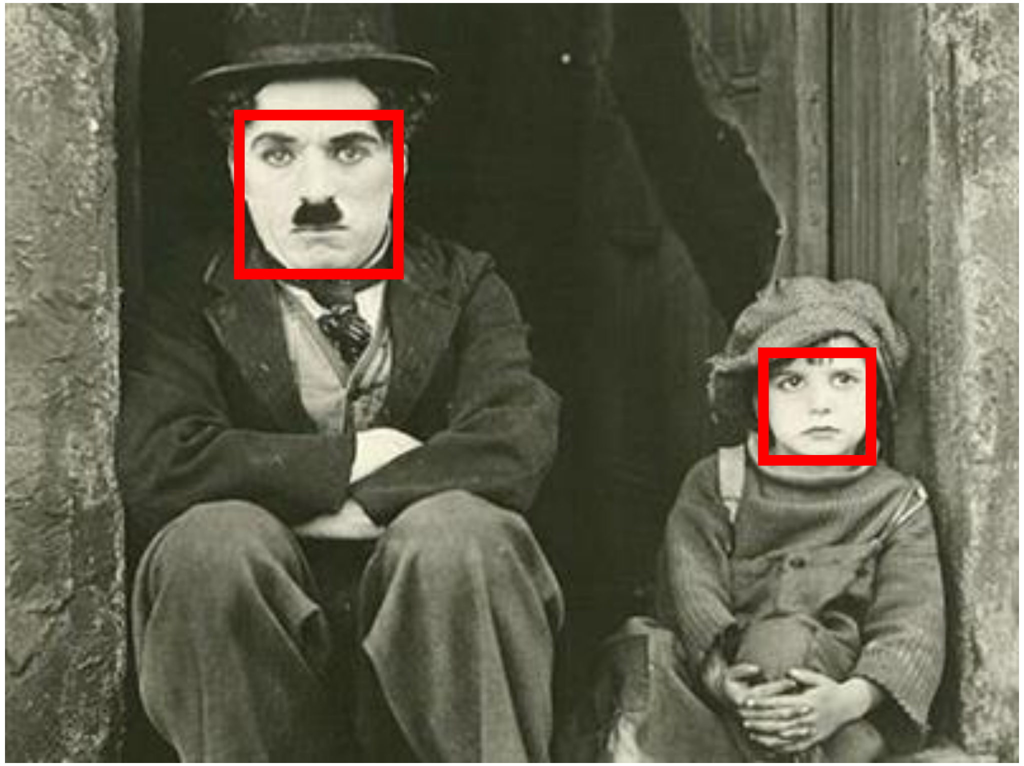


Face Detection: Viola - Jones Algorithm

✓ How to detect face(s) in an image?

- ❖ Sliding window: 24x24 (384x288 image)
- ❖ Binay classifier: Face / Non face
- ❖ Window scaling

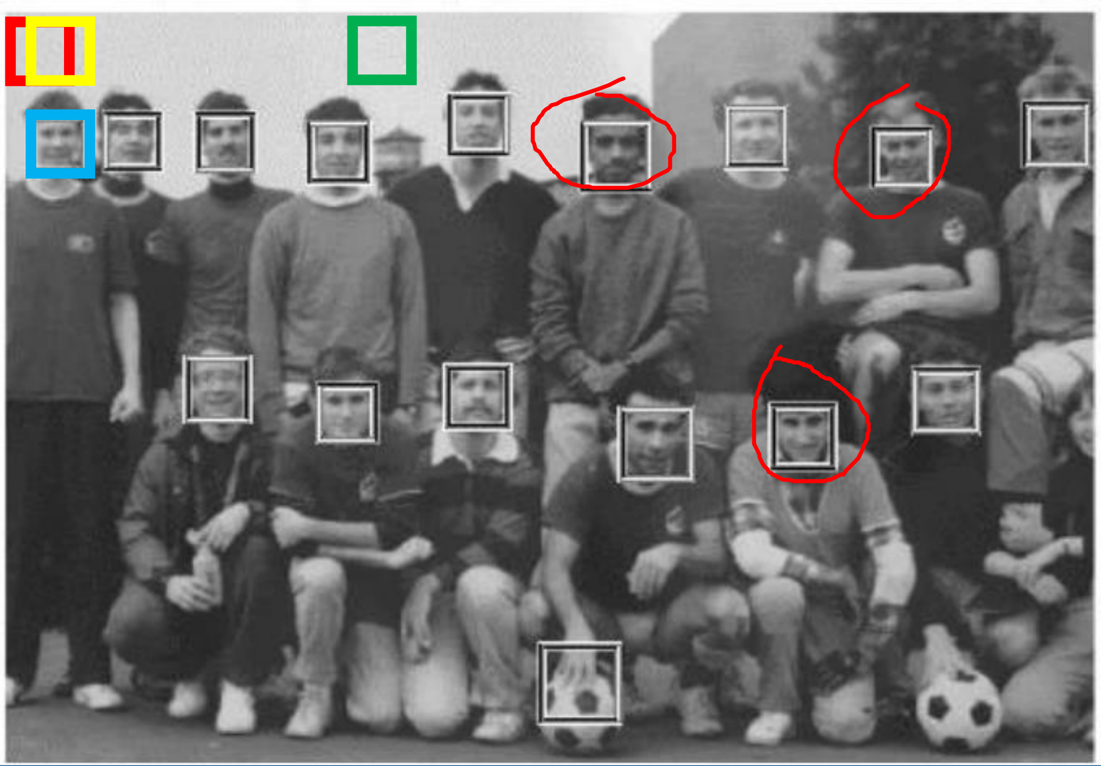




Face Detection: Viola - Jones Algorithm

✓ How to detect face(s) in an image?

- ❖ Sliding window: 24x24 (384x288 image)
- ❖ Binay classifier: Face / Non face



- ❖ AlexNet?
  - ▪ Binary classifier → AlexNet
  - ▪ Multiple object detection
- ❖ More efficient?
  - ▪ Region proposal

YOLO v1

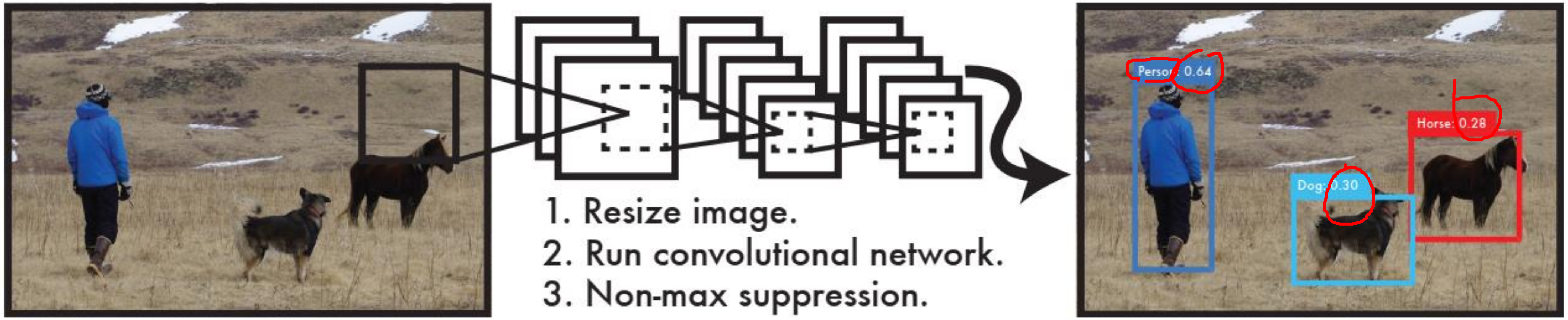
✓ Two-stage object detection (R-CNN, fast R-CNN, faster R-CNN)

Image → **Region Proposal** → **Object Classification** →

✓ One-stage object detection

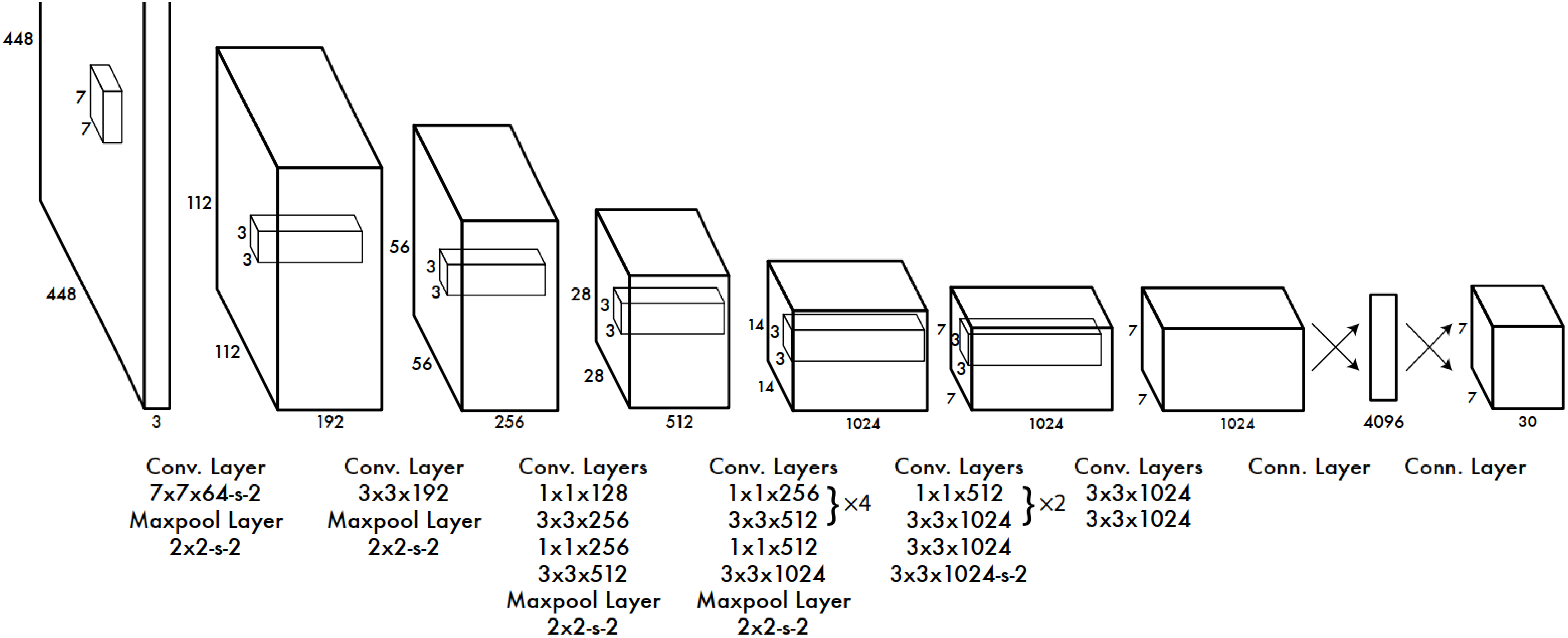
Image → **CNN** →

❖ YOLO – You Only Look Once



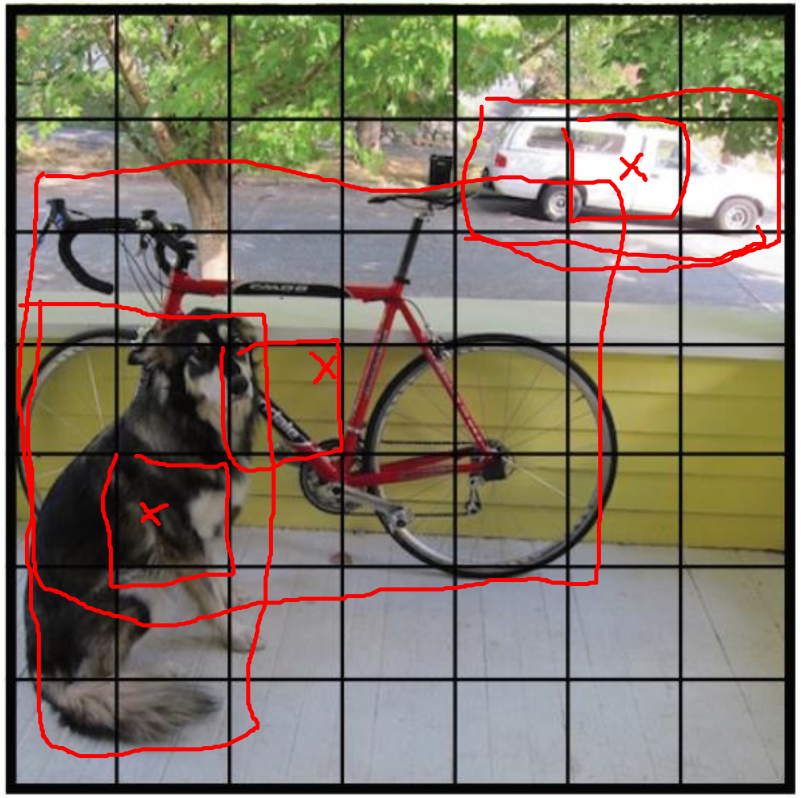
YOLO v1

✓ Structure

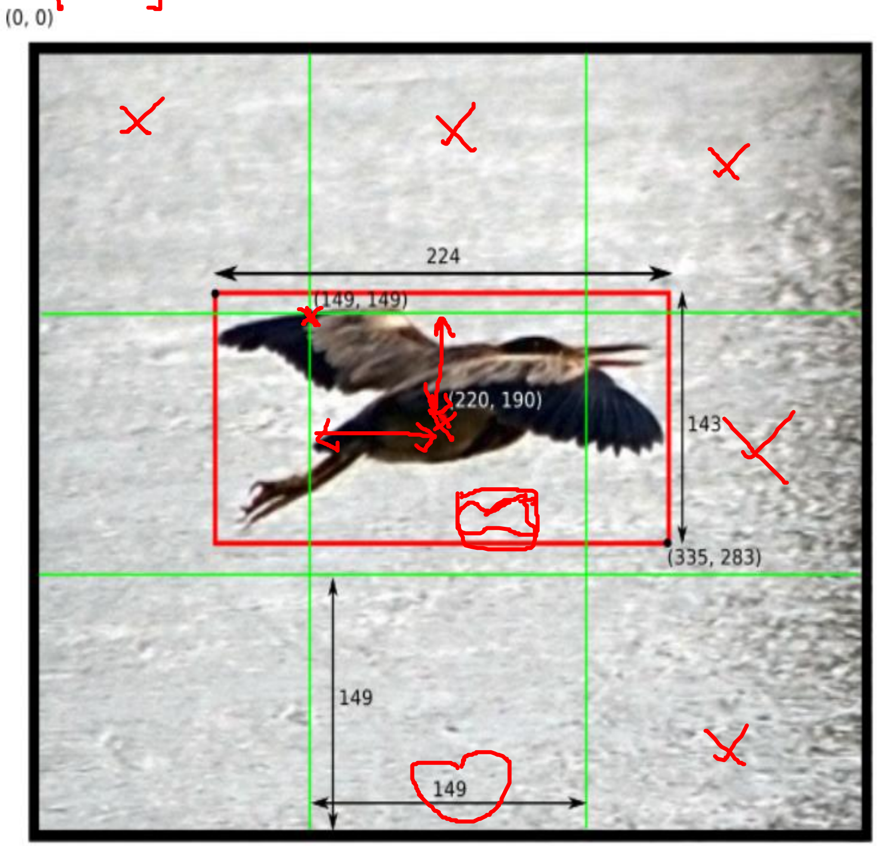


YOLO v1

- ✓ S = 7
- ✓ Bounding box: x, y, w, h $\in [0, 1]$
- ✓ Confidence score



S × S grid on input



x = (220-149) / 149 = 0.48

y = (190-149) / 149 = 0.28

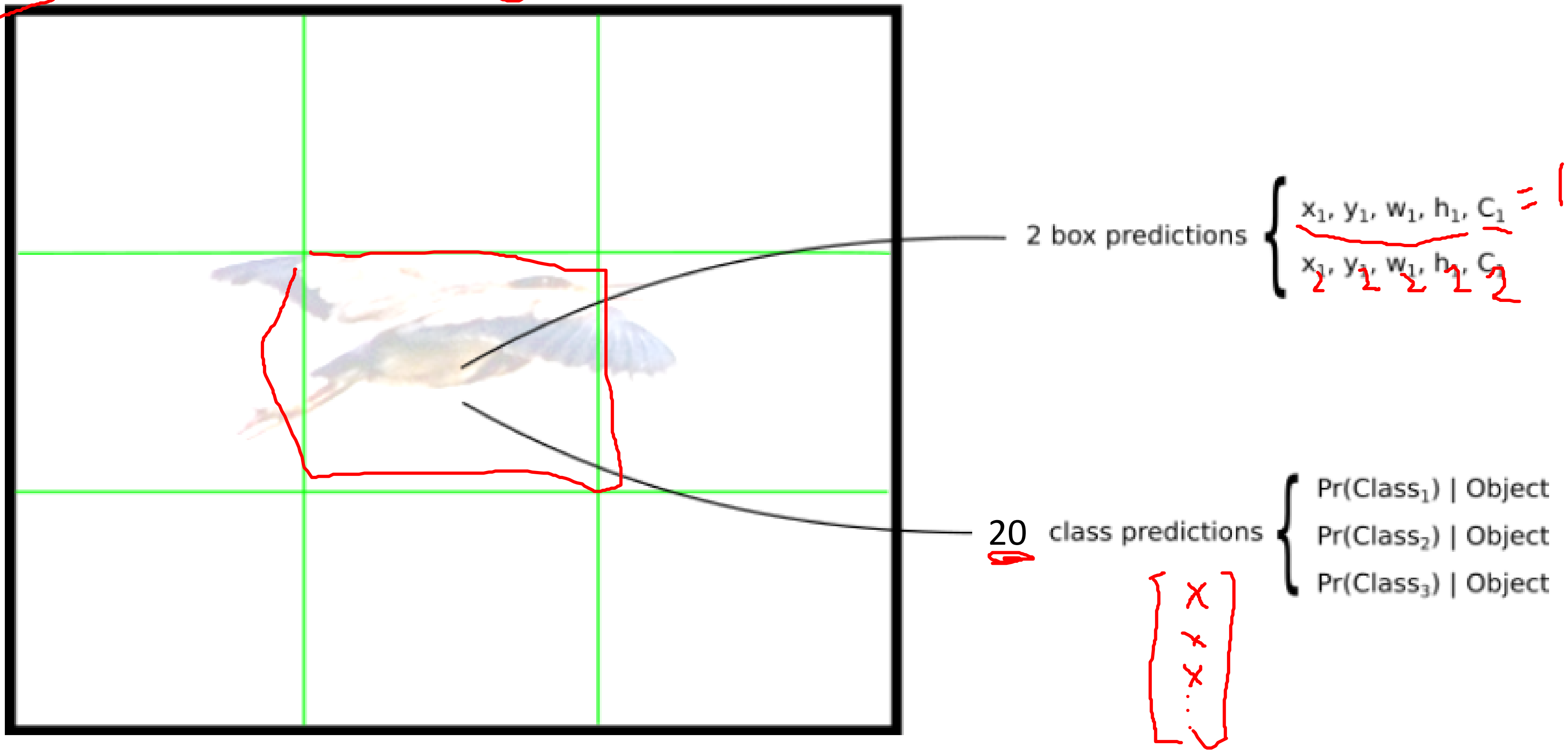
w = 224 / 448 = 0.50

h = 143 / 448 = 0.32

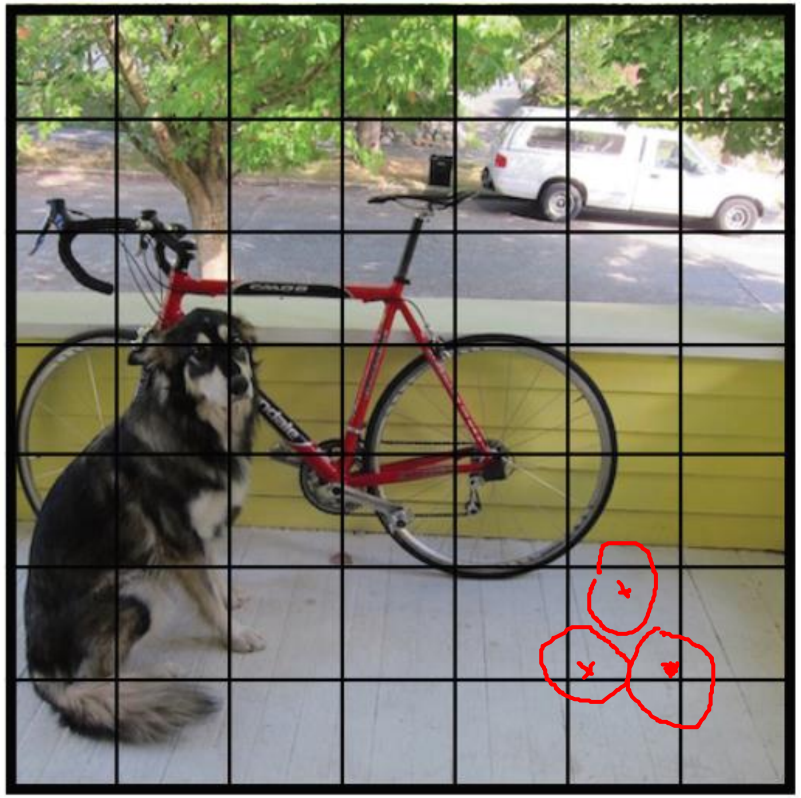
YOLO v1

- # outputs:
  - $(5B + C)S^2$, $B = 2$, $C = 20$, $S = 7$

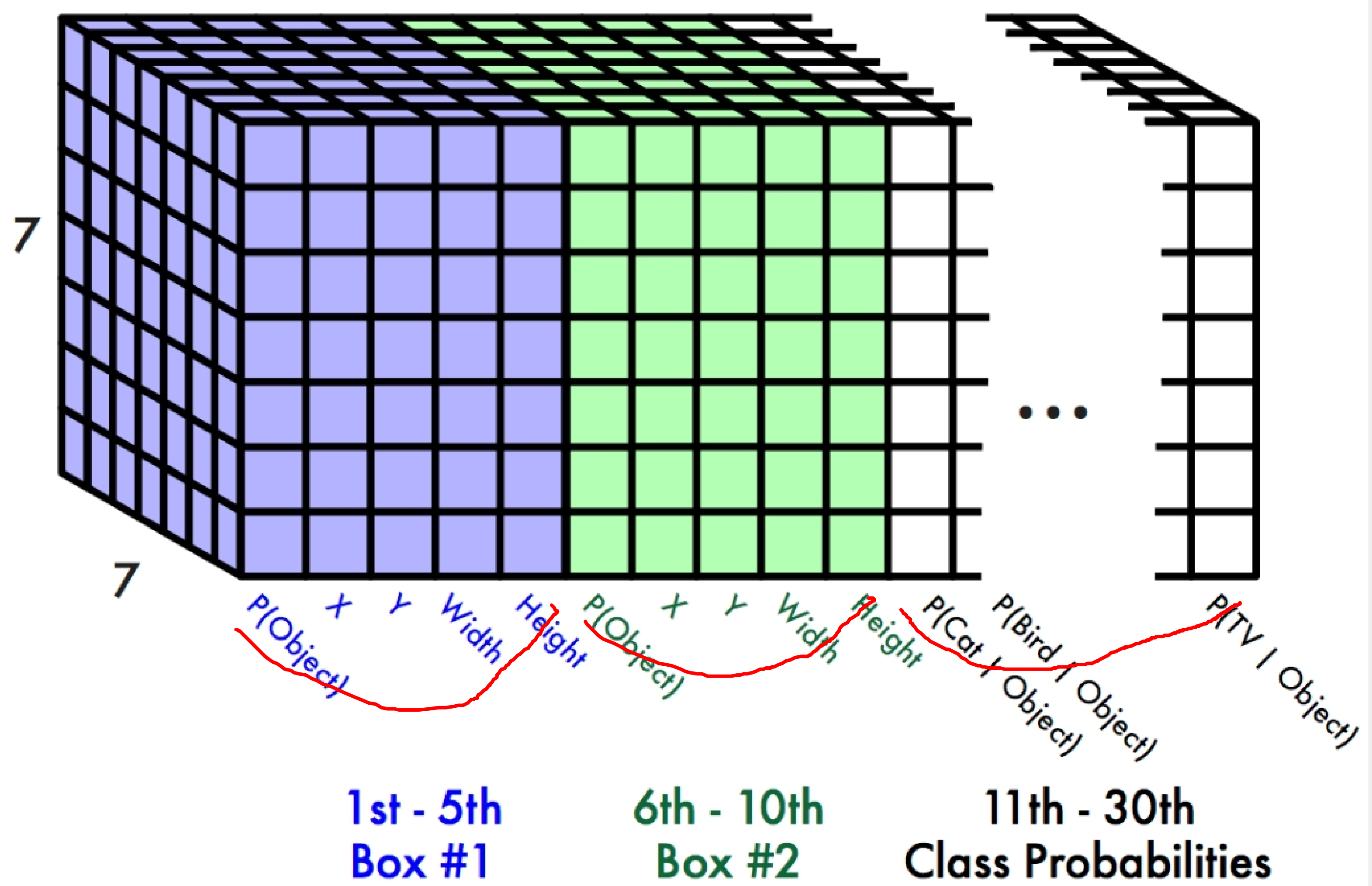
1470



YOLO v1



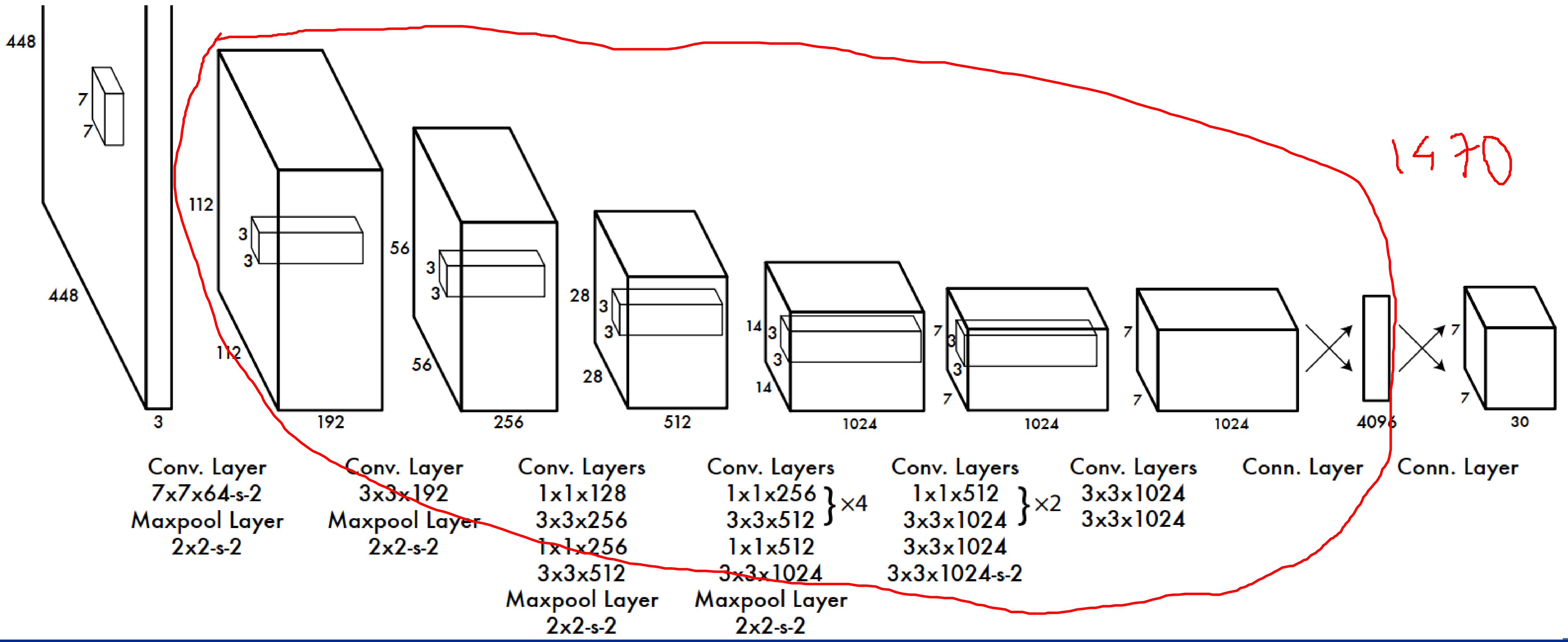
S × S grid on input



7
7
P(Object)
X
Y
Width
Height
P(Object)
X
Y
Width
Height
P(Cat / Object)
P(Bird / Object)
P(TV / Object)
...
1st - 5th
Box #1
6th - 10th
Box #2
11th - 30th
Class Probabilities

YOLO v1

✓ ~~Linear~~ regression problem



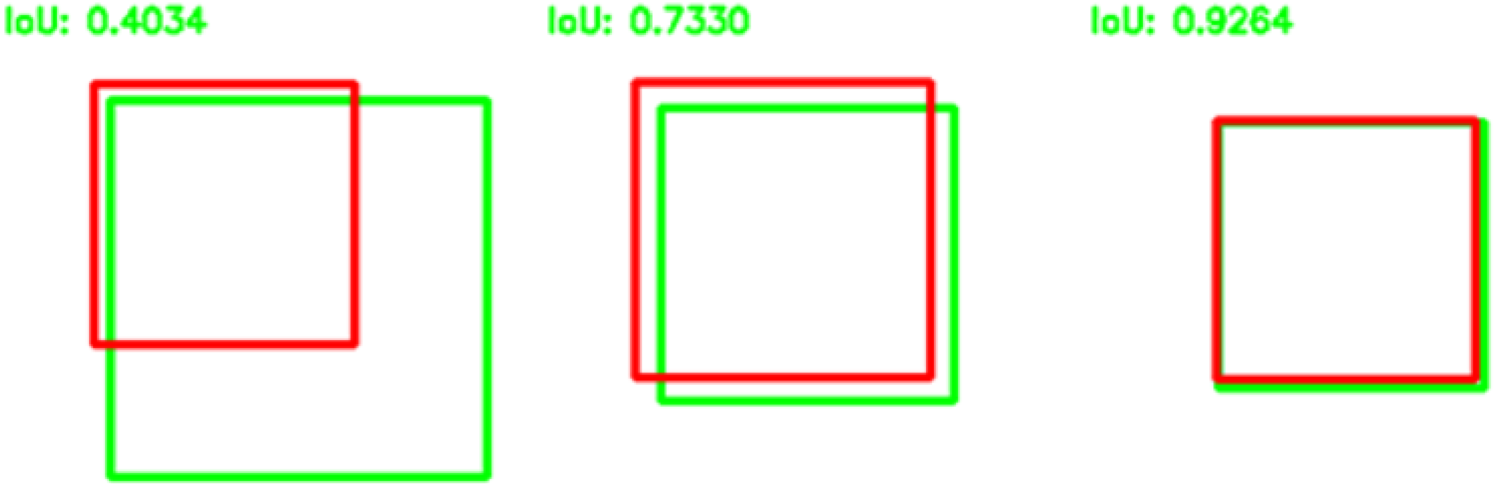
YOLO v1

✓ Confidence score

- How likely the bounding box contains an object?
- How accurate is the bounding box (location and size)?

Confidence score = Pr(Object)*IoU

IoU: Intersection over Union

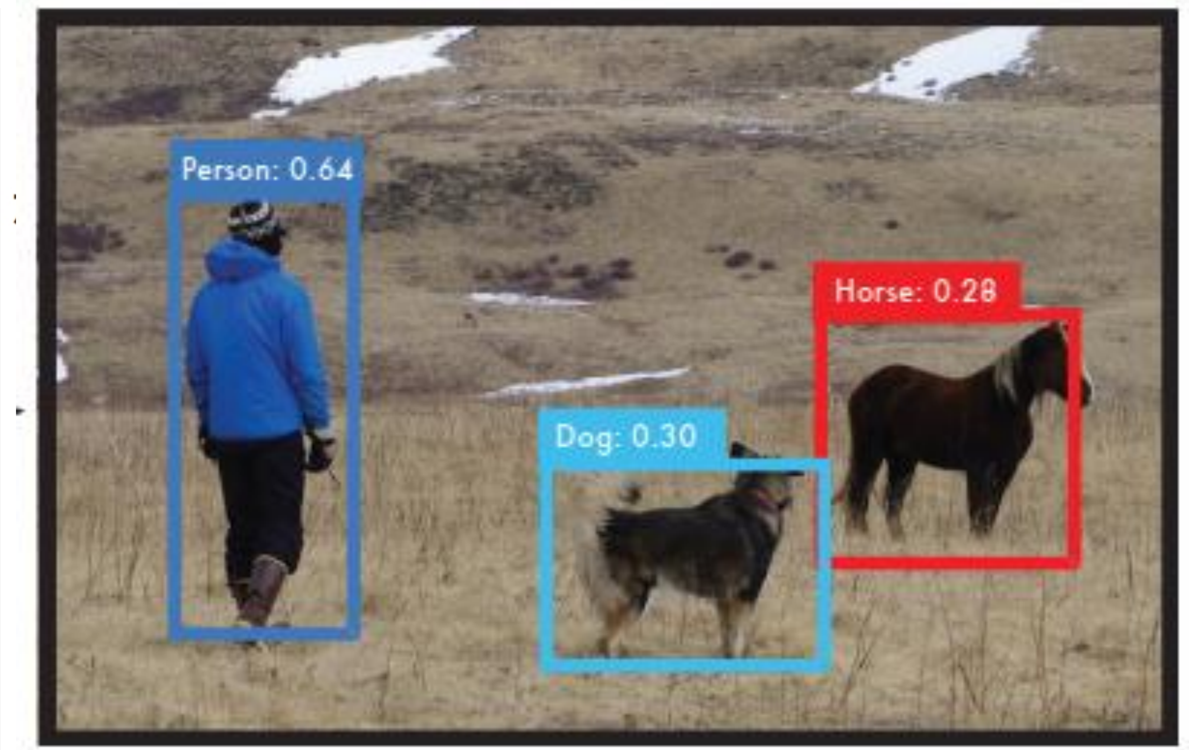


$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

YOLO v1

- ✓ Confidence score
  - ❖ How likely the bounding box contains
  - ❖ How accurate is the bounding box (lo

  Confidence score C = Pr(Object)*IoU



- ✓ Conditional class probability

  $p_i(c) = Pr(Class_i|Object)$
- ✓ Test:

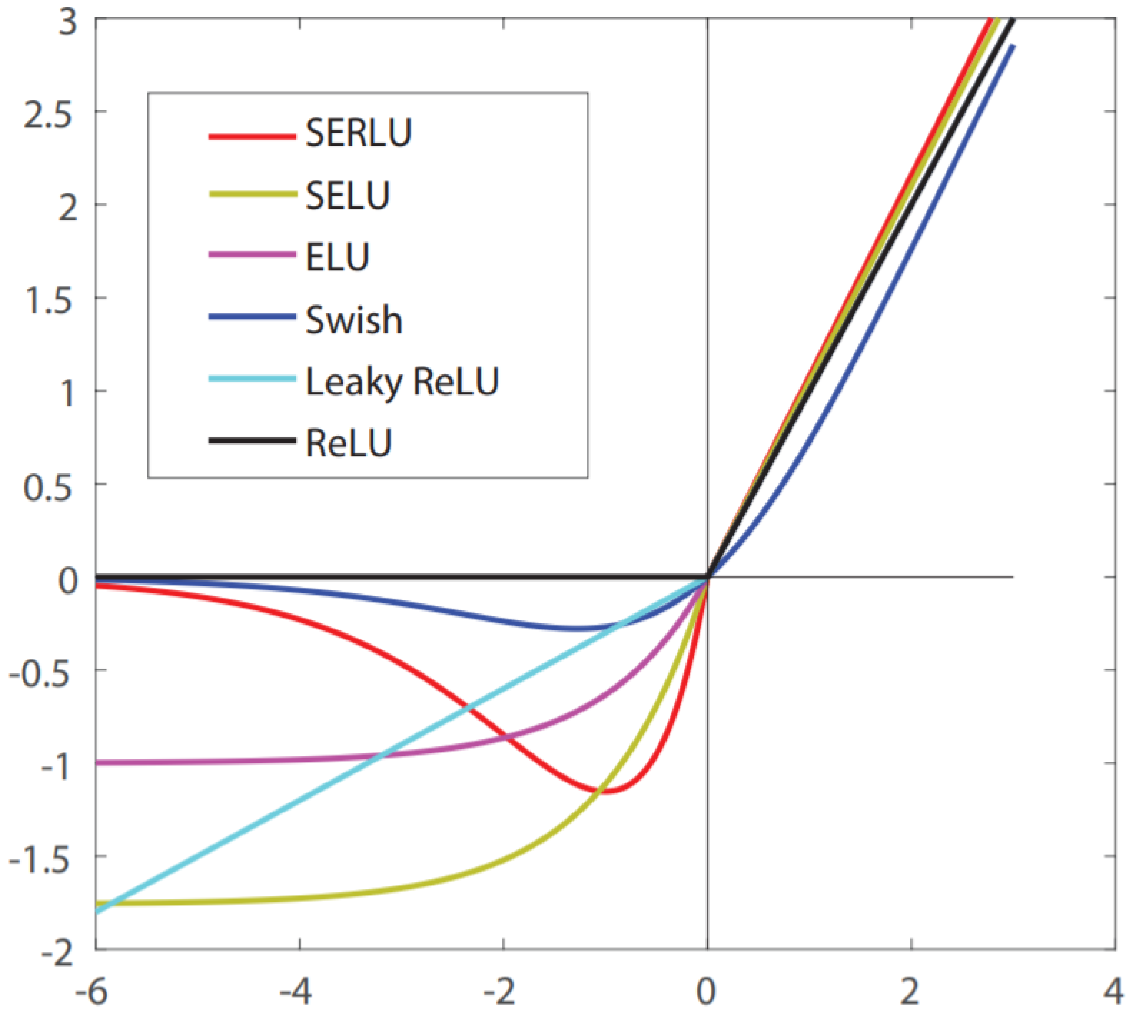
$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

Class-specific confidence scores for each box: probability of that class appearing in the box and how well the predicted box fits the object.

YOLO v1

✓ Activation function

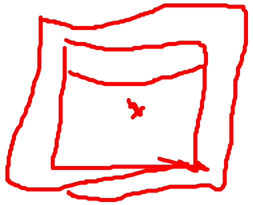
$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases}$$



YOLO v1

$(w - \hat{w})$

✓ Loss function

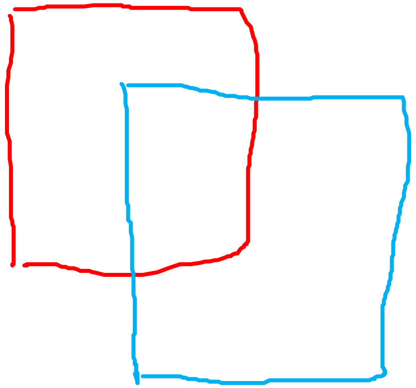


$$
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2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

hat

$w = 5$
$\hat{w} = 15$

$w = 100$
$\hat{w} = 110$



YOLO v1

✓ Training

We pretrain our convolutional layers on the ImageNet 1000-class competition dataset [30]. For pretraining we use the first 20 convolutional layers from Figure 3 followed by a average-pooling layer and a fully connected layer. We train this network for approximately a week and achieve a single crop top-5 accuracy of 88% on the ImageNet 2012 validation set, comparable to the GoogLeNet models in Caffe's Model Zoo [24]. We use the Darknet framework for all training and inference [26].

We then convert the model to perform detection. Ren et al. show that adding both convolutional and connected layers to pretrained networks can improve performance [29]. 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 224×224 to 448×448.

YOLO v1

We 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.

Our 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.

YOLO v1

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 [18]. 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.

YOLO v1

- Limitations
  - Spatial constrain
    - Two bounding boxes, one class per grid cell
    - Struggle with small objects in groups, e.g. flocks of birds
  - Relatively coarse features due to multiple downsampling layers
  - Main error: incorrect localization

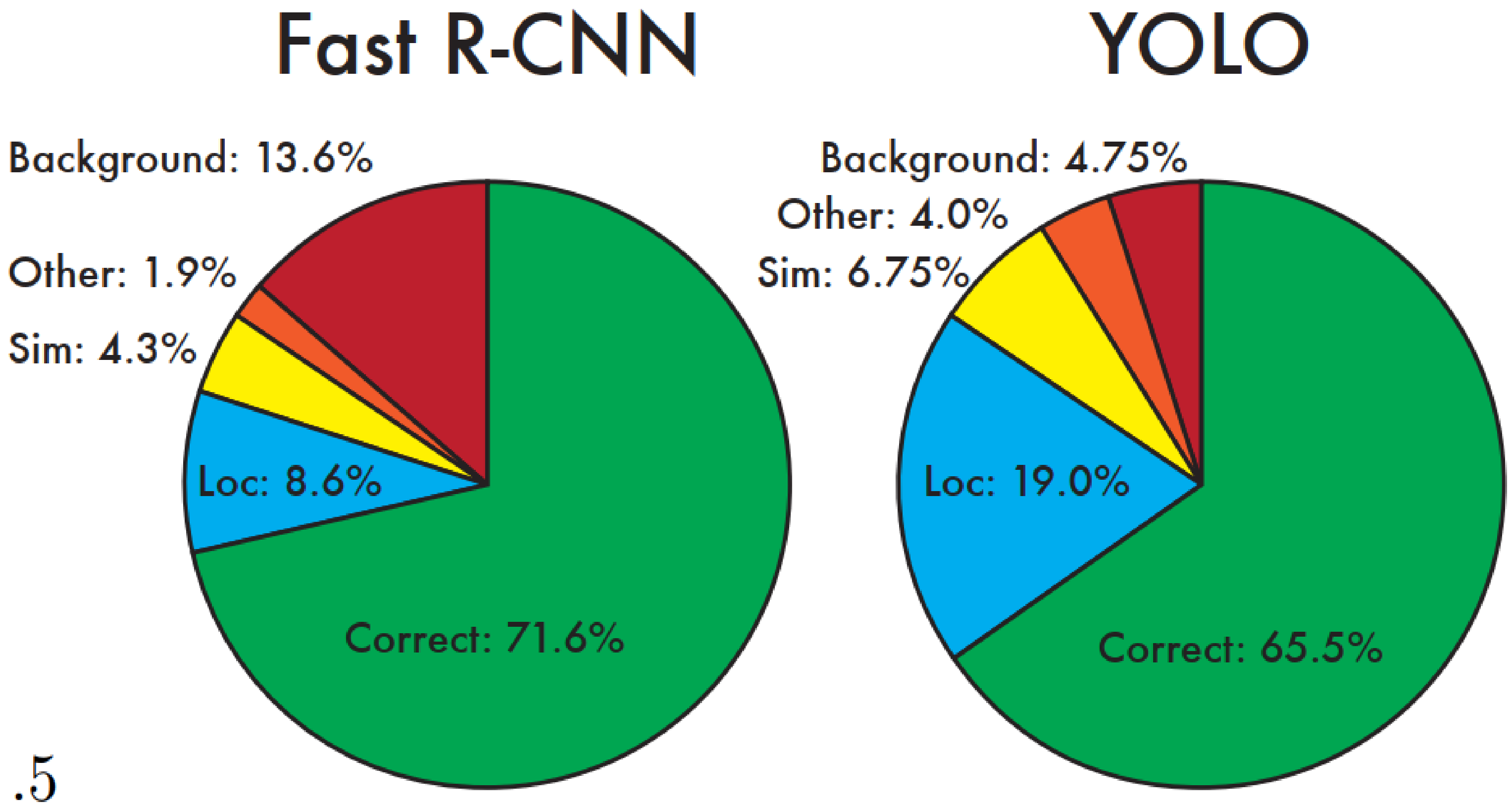
YOLO v1

✓ Results – PASCAL VOC 2007

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

YOLO v1

✓ Results – PASCAL VOC 2007



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

YOLO v1

✓ Results – PASCAL VOC 2007

	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**

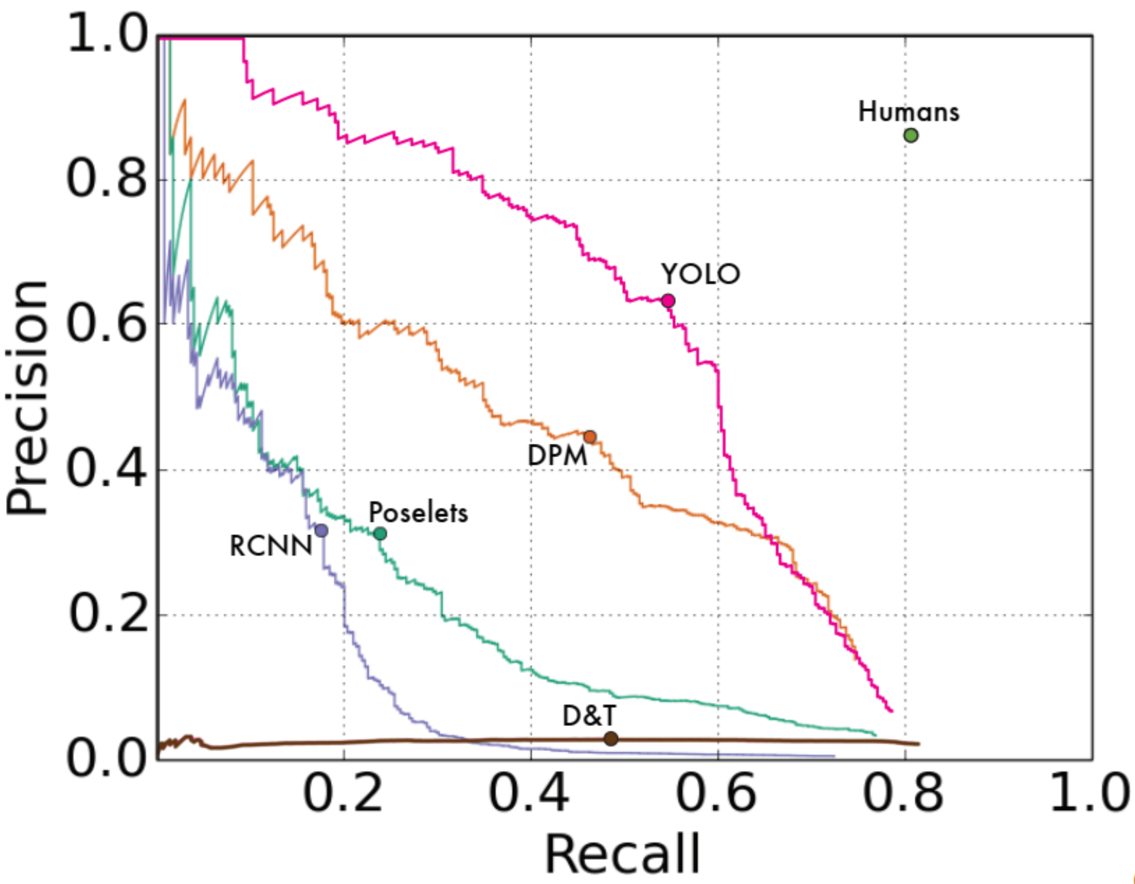
YOLO v1

✓ Results – PASCAL VOC 2012

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

YOLO v1

✓ Results

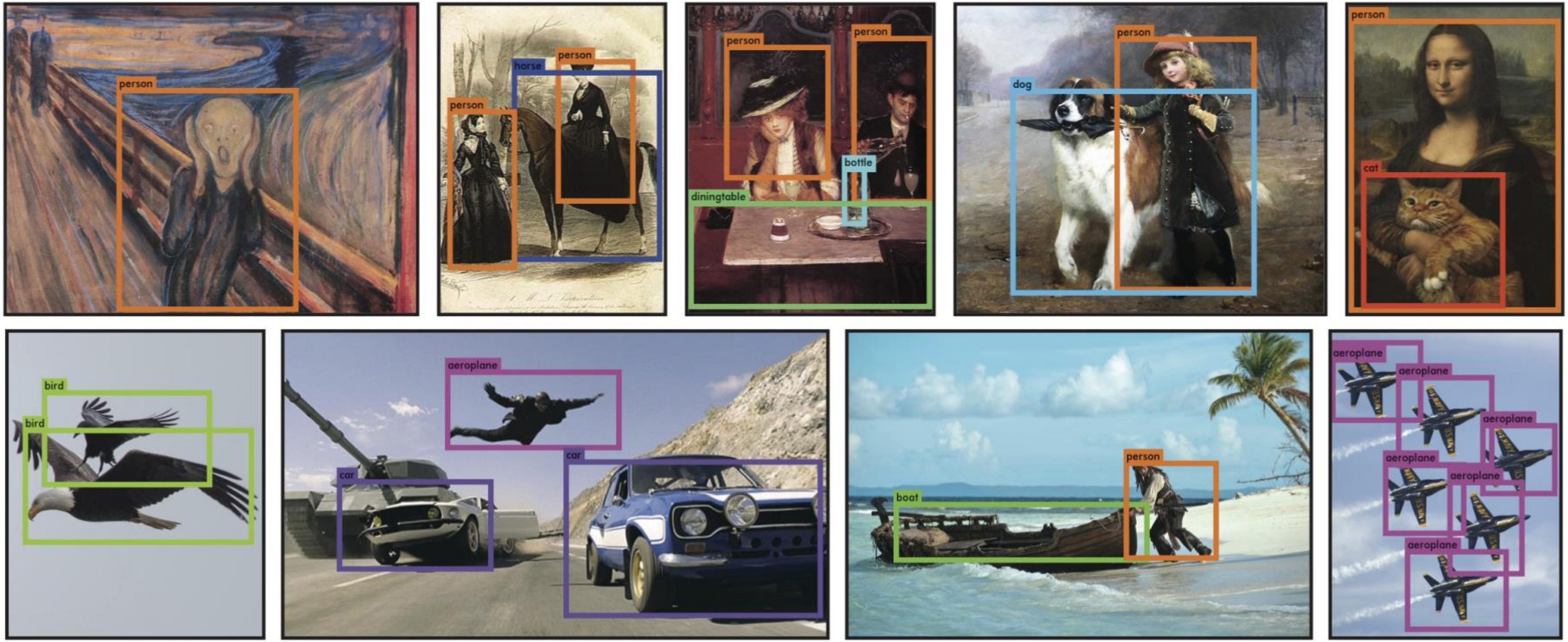


(a) Picasso Dataset precision-recall curves.

	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	

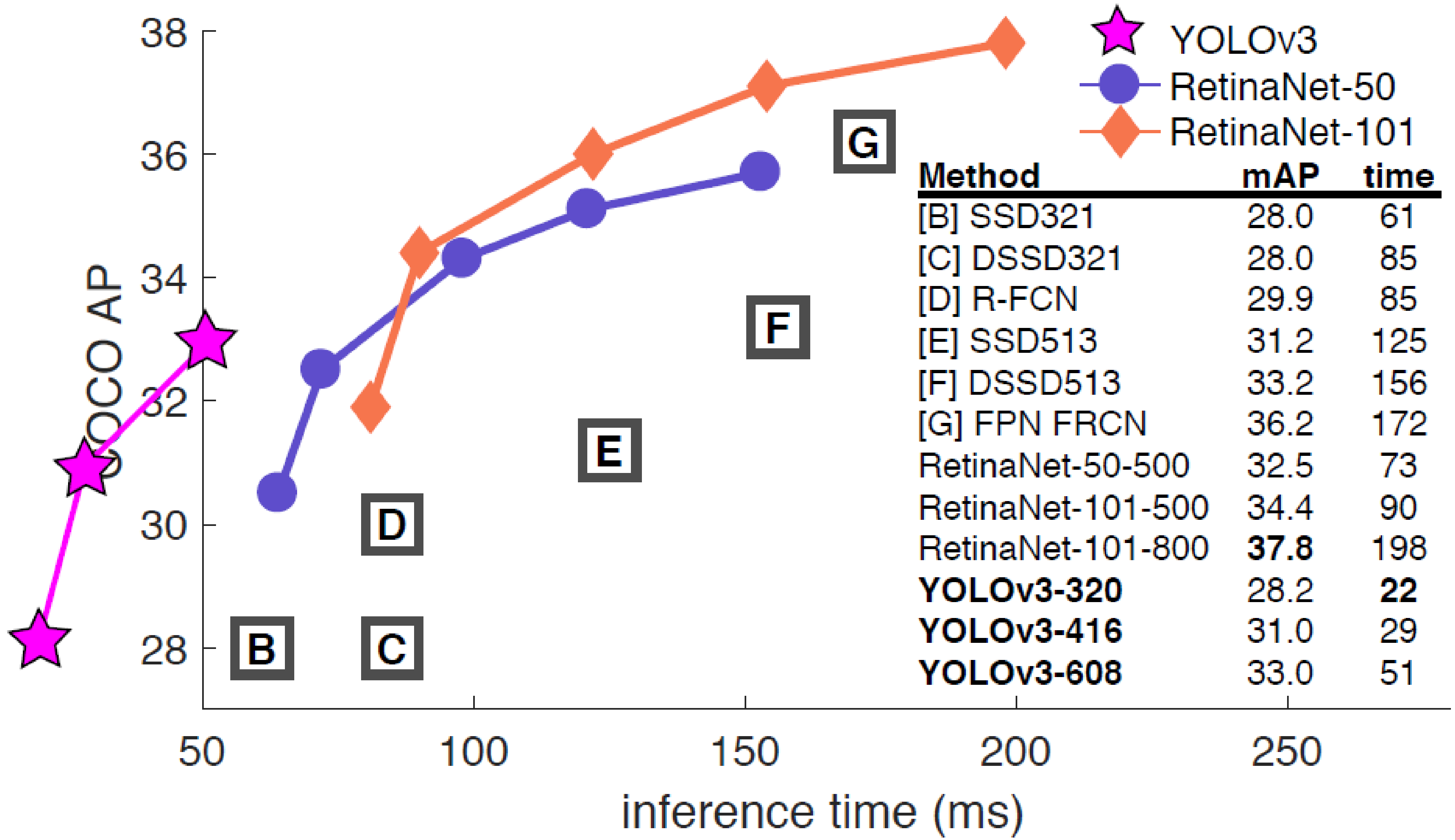
(b) Quantitative results on the VOC 2007, Picasso, and People-Art Datasets. The Picasso Dataset evaluates on both AP and best F_1 score.

YOLO v1


person
horse
person
person
person
person
bottle
diningtable
person
dog
person
cat
bird
bird
aeroplane
car
car
person
boat
aeroplane
aeroplane
aeroplane
aeroplane
aeroplane
aeroplane

YOLO v1

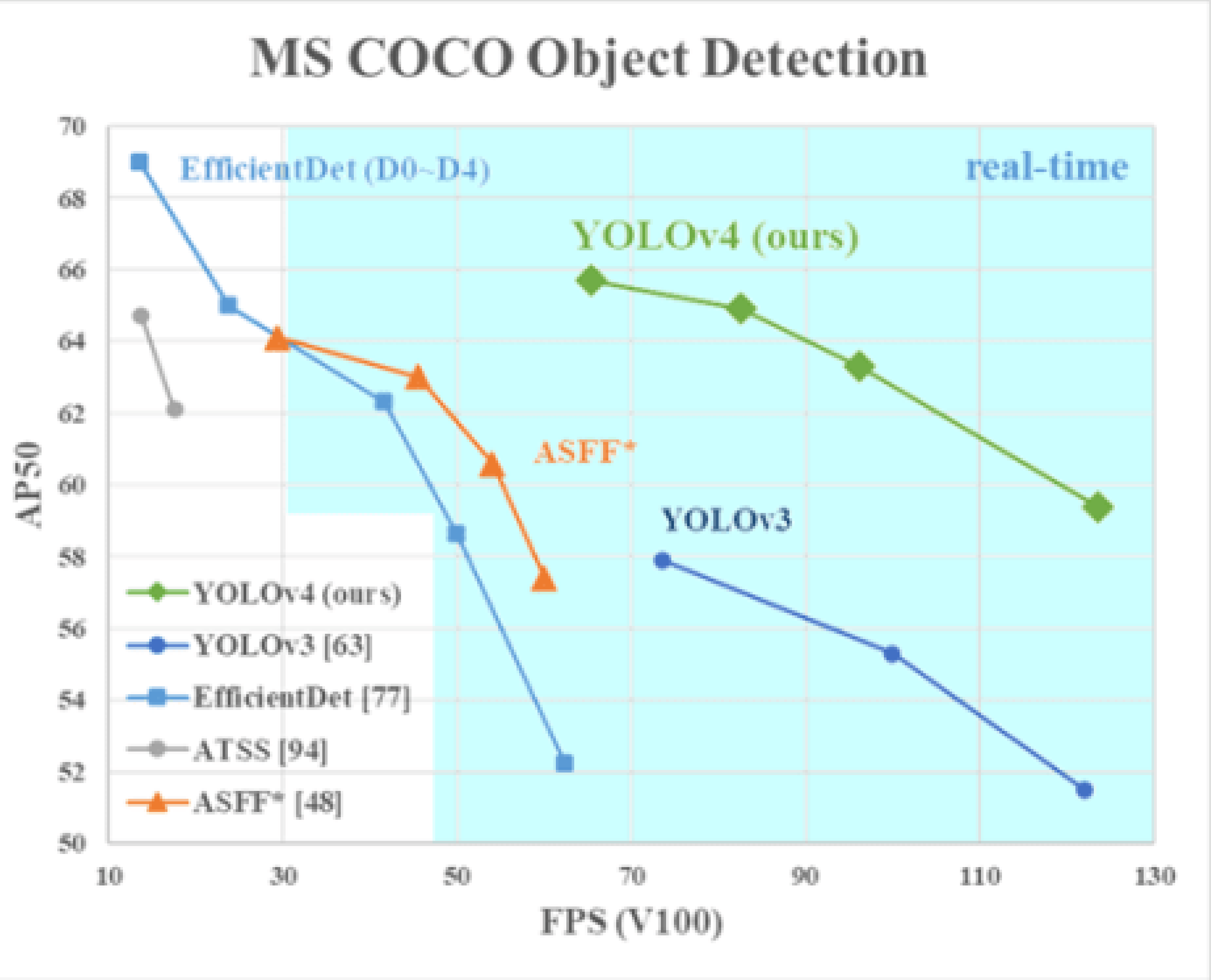
✓ Results



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

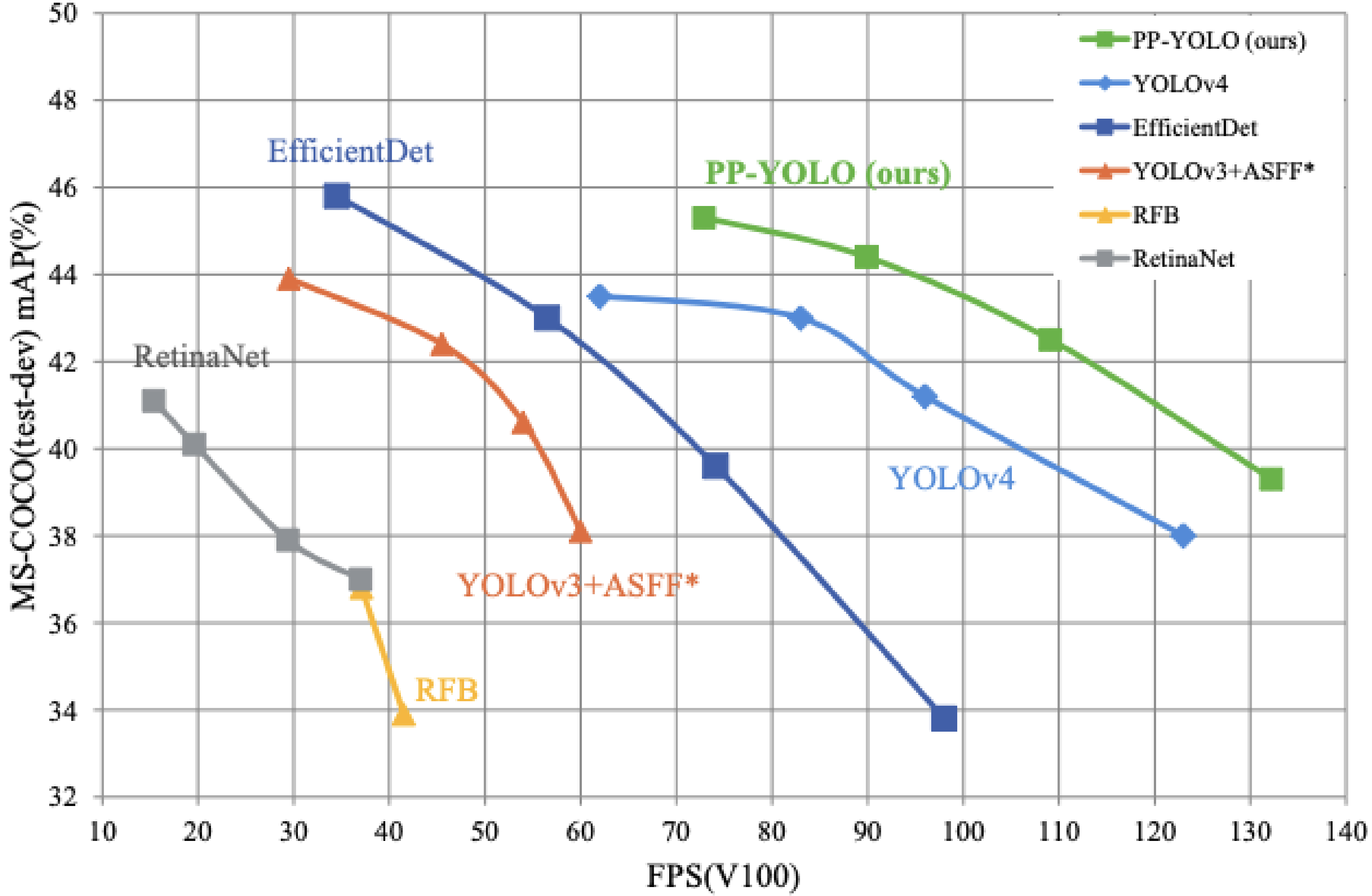
YOLO v1

✓ Comparison



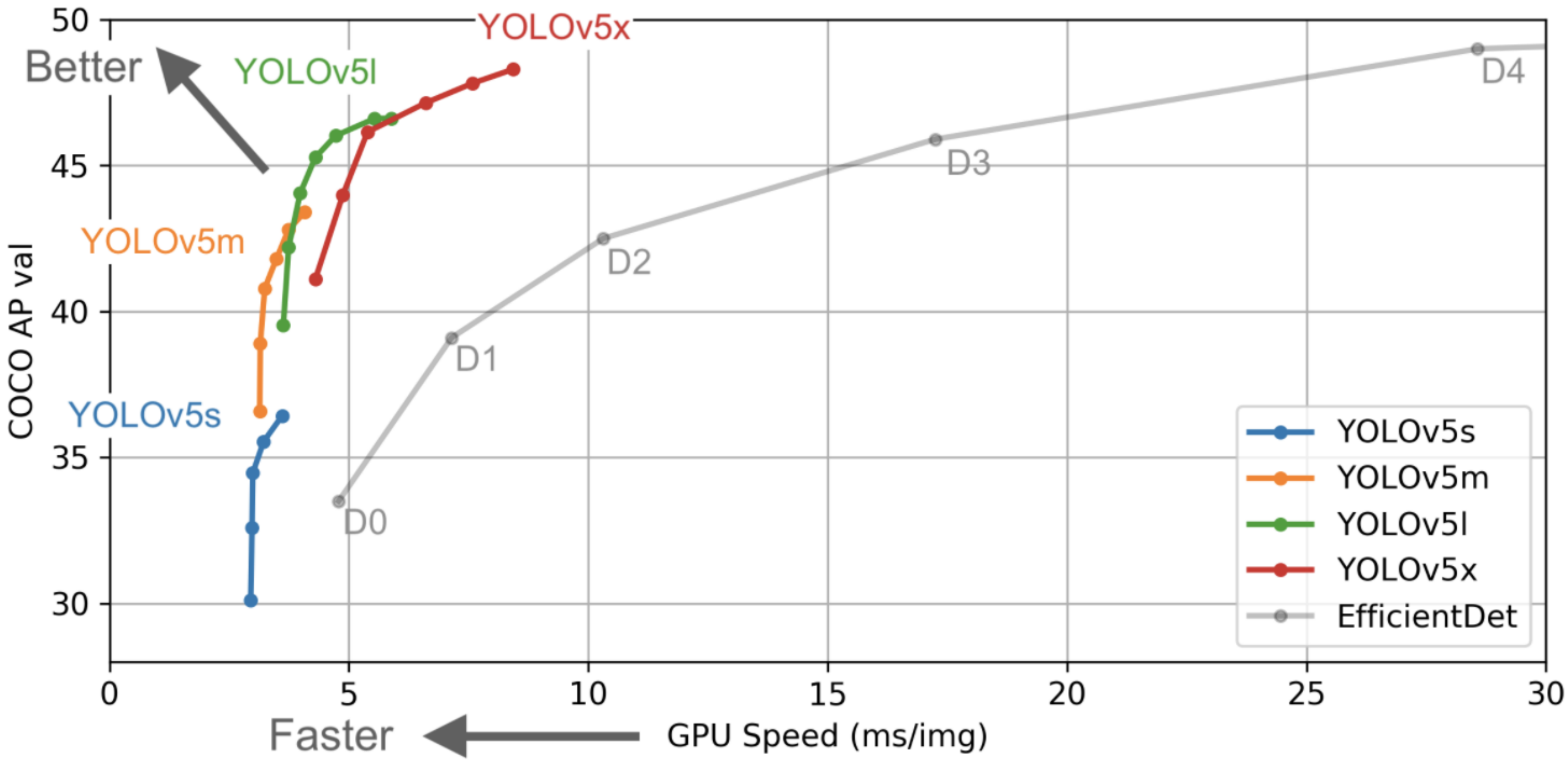
YOLO v1

✓ Comparison



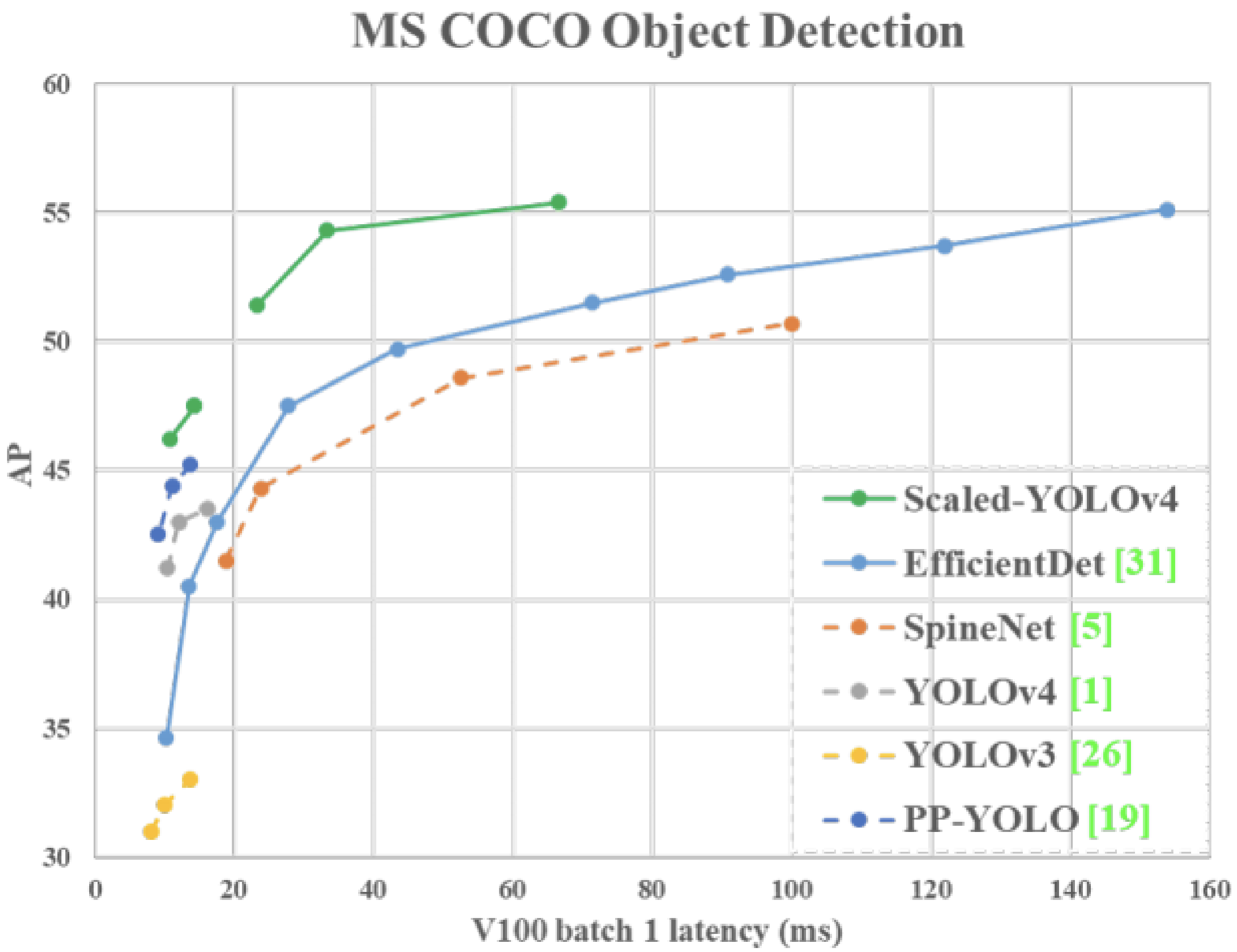
YOLO v1

✓ Comparison

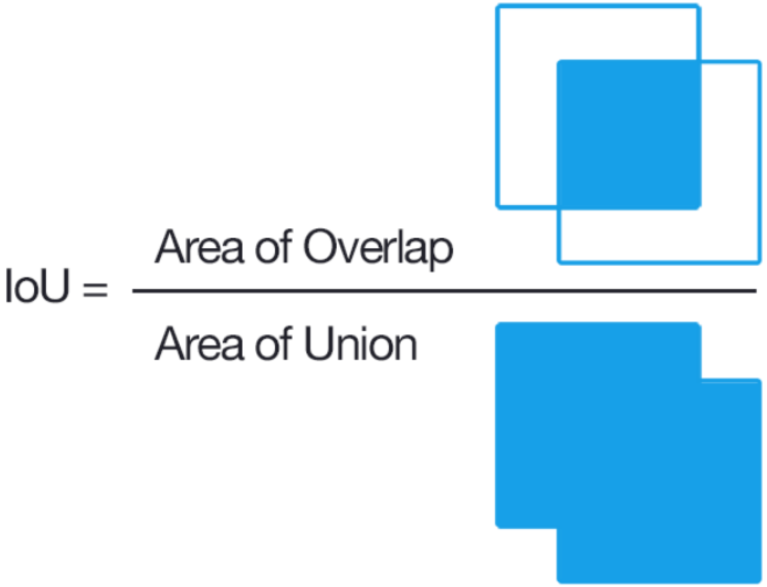


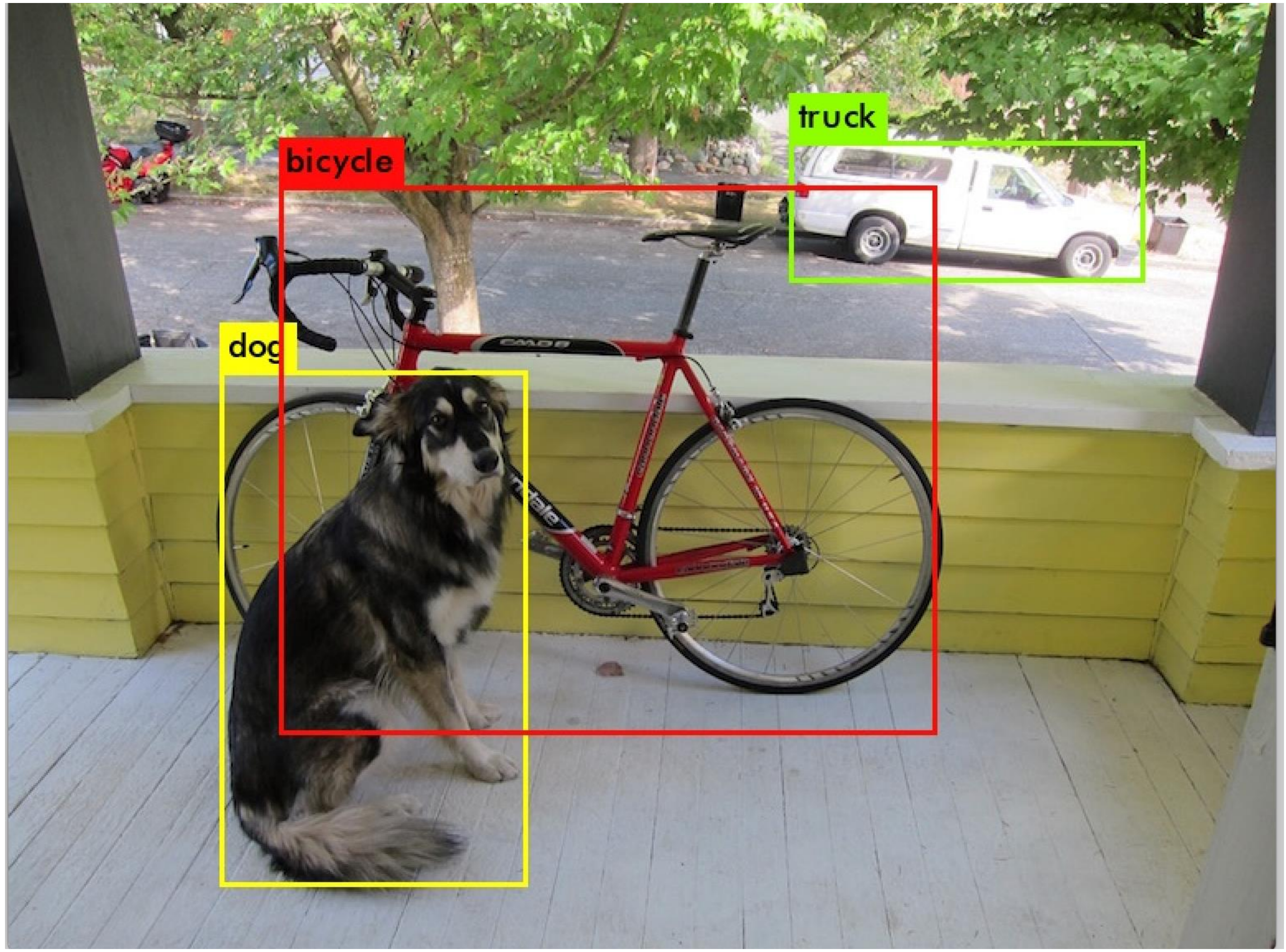
YOLO v1

- Comparison



- ✓ Evaluation
  - ❖ Classification problem?
    - Top-1 error rate
    - Top-5 error rate
  - ❖ Object detection problem?





YOLO v1

- Evaluation
  - Confusion matrix
  - Recall (detection rate, true positive rate, sensitivity)

$$Recall = DR = \frac{TP}{TP + FN}$$

  - Precision

$$Precision = \frac{TP}{TP + FP}$$

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

YOLO v1

✓ Evaluation

- Precision - Recall curve
- Interpolated Precision - Recall curve
- AP
- AP50, AP75
- mAP

✓ Dataset

- PASCAL VOC
- COCO