Object Detection in AI2Thor Environment

Lecturer: Dr. Nguyễn Đỗ Văn

Team members:

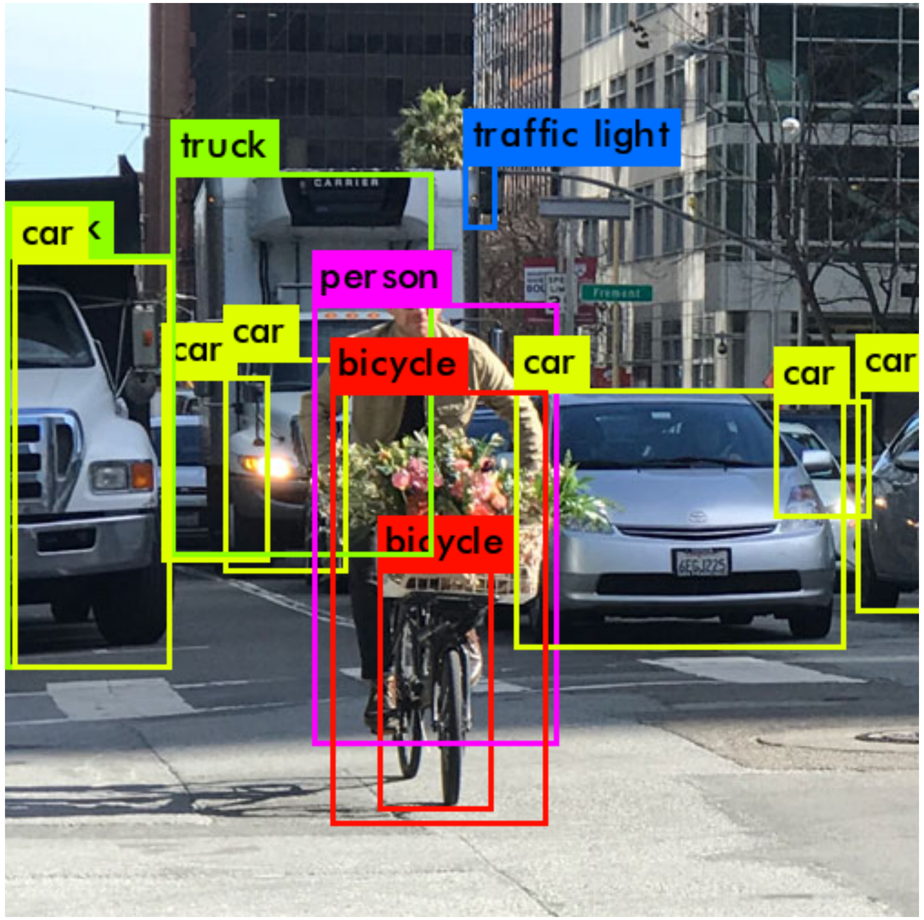
Hoàng Giang
Trần Minh Đức
Đỗ Đình Hiếu

Table of content

I. Introduction

Why object detection ?

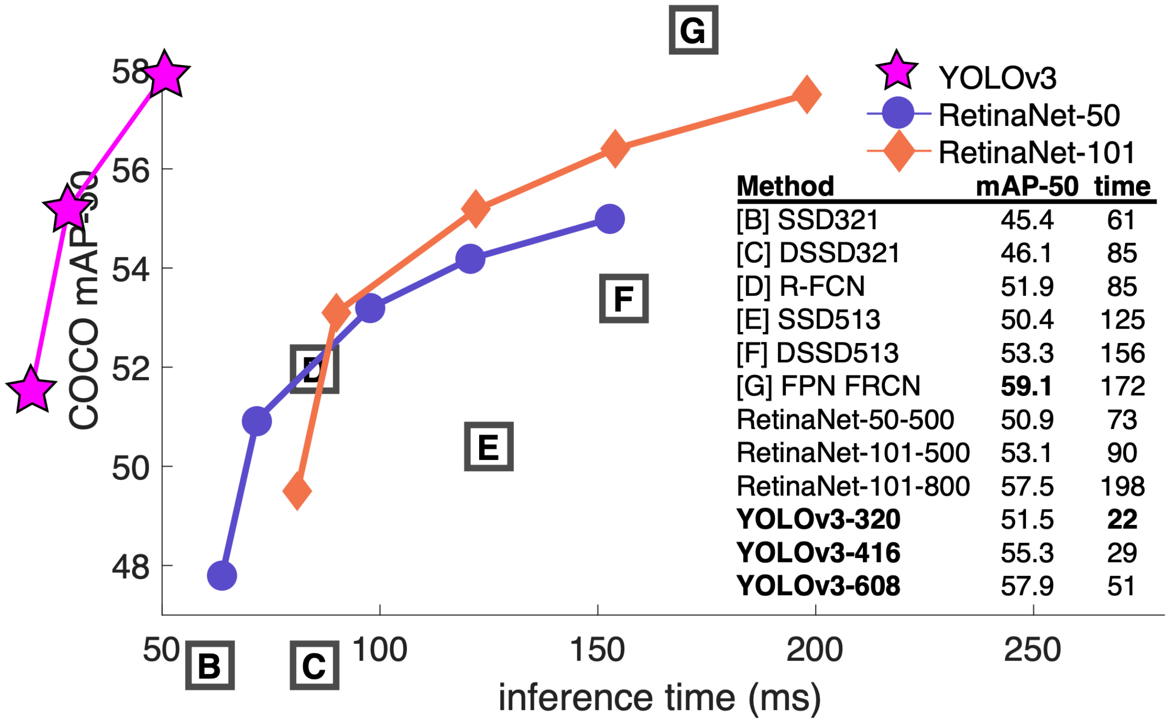
- A crucial task in AI
- Pre-trained models availability
- Large amount of dataset for training and evaluation
- Large amount of techniques for improving accuracy



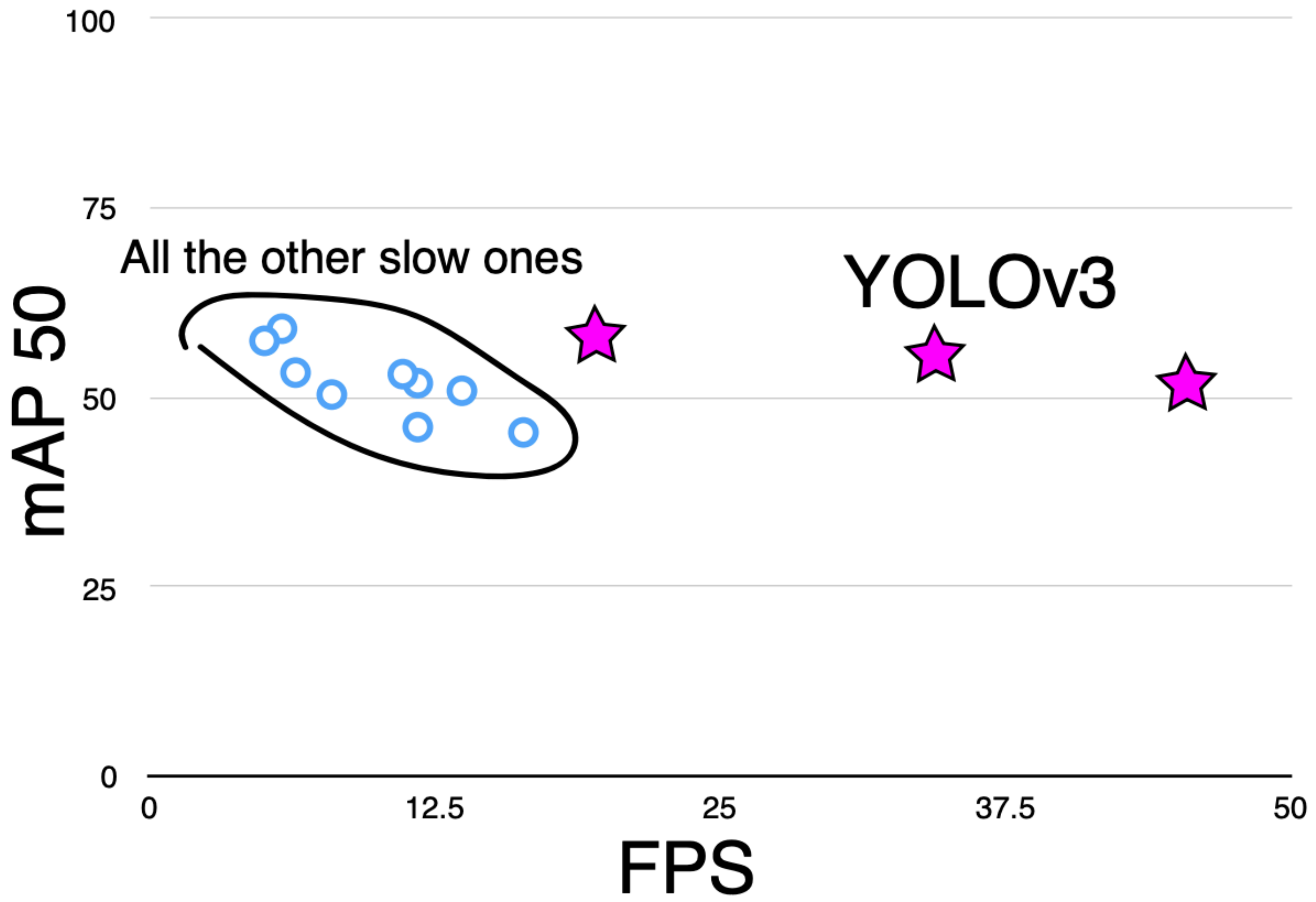
II. YOLO

1. Why YOLOv3?

- Fast, real-time
- Acceptable accuracy



1. Why YOLOv3?



100
75
50
25
0
mAP 50
All the other slow ones
YOLOv3
0
12.5
25
37.5
50
FPS

2. YOLOv3

"You Only Look Once"

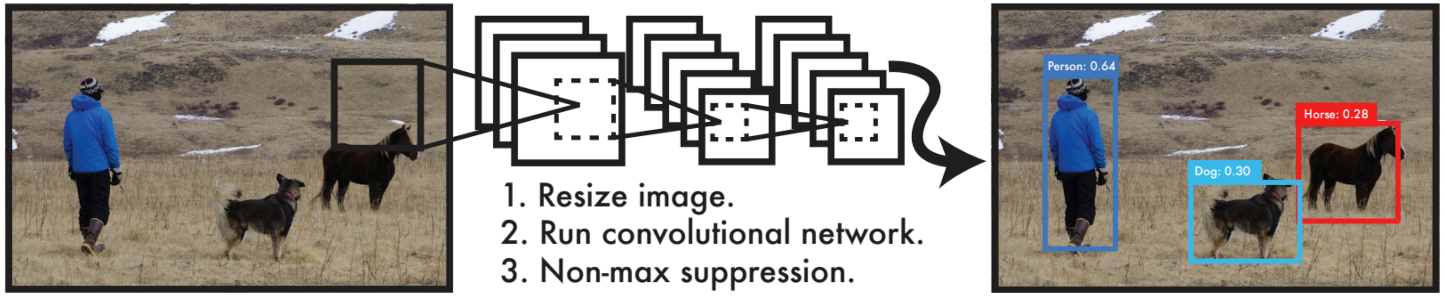


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448×448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

2. YOLOv3

- Darknet-53 (vs. 19 in YOLOv2)

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Table 1. **Darknet-53.**

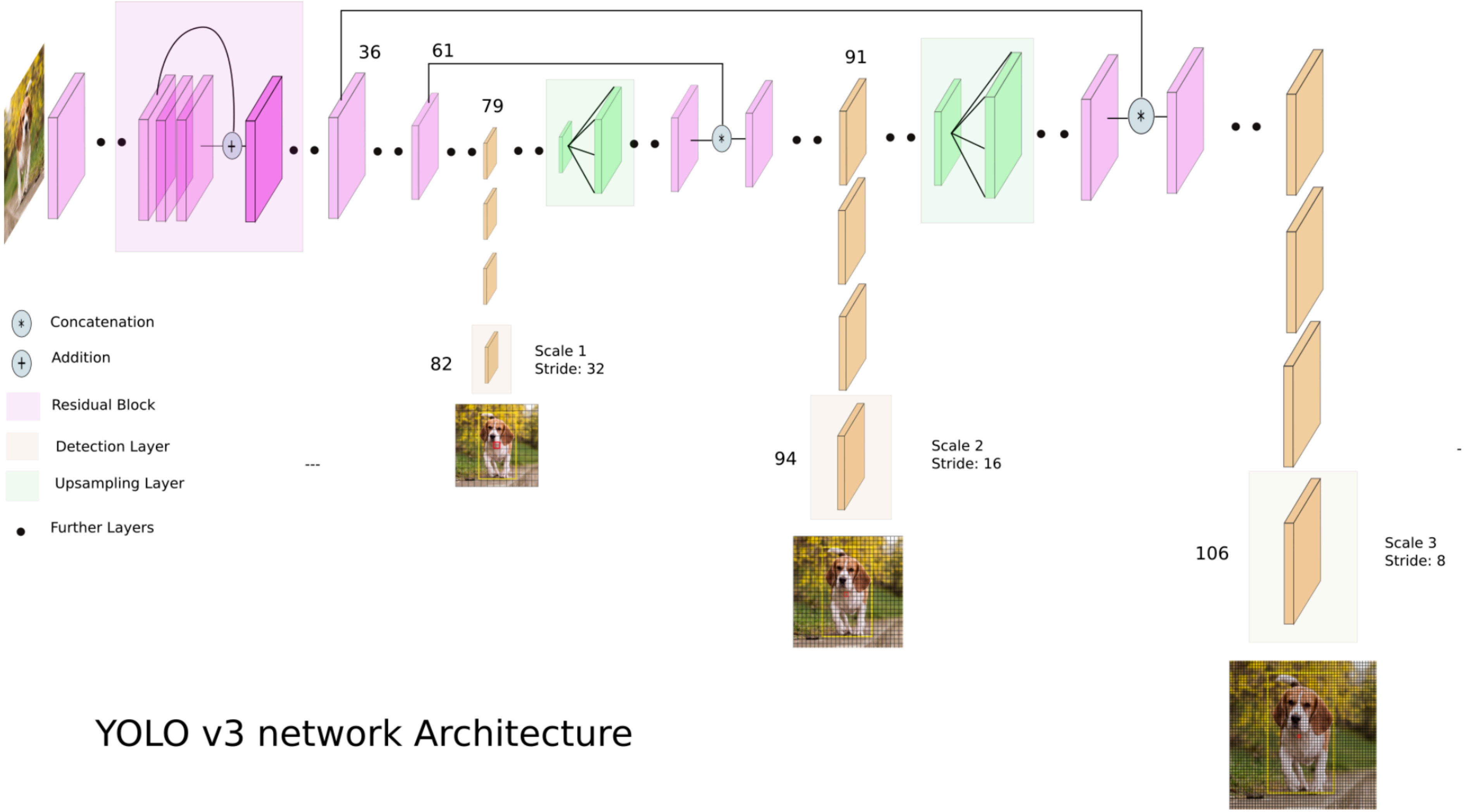
2. YOLOv3

- Make detections at three different scales at three different places.

-> Better at detecting smaller objects (smaller stride smaller objects)

- 3 anchor boxes at each scale.
- Predicts 10x number of boxes by YOLOv2.

2. YOLOv3



YOLO v3 network Architecture

2. YOLOv3

- Result

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [5]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [8]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [6]	Inception-ResNet-v2 [21]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [20]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [15]	DarkNet-19 [15]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [11, 3]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [3]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [9]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [9]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608×608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

3. Terminologies

- Stride of network (or layer) is the ratio by which it downsamples the input (YOLOv3: 3 stride: 32, 16, 8)
- mAP: mean Average Precision
- (Intersection over union) IoU = overlap/union : measures the overlap between 2 boundaries (prediction vs ground truth)

III. RetinaNet

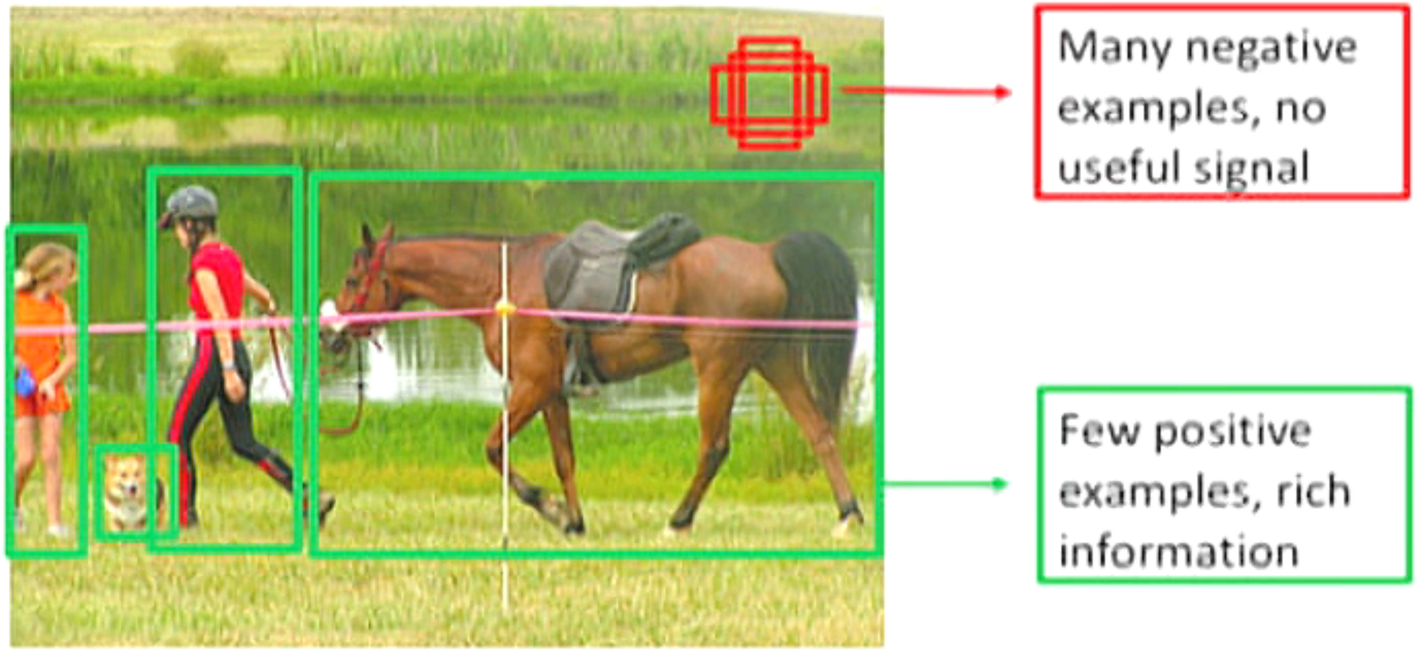
1. Class imbalance problem

- Two-stage detector (Faster R-CNN):
  - Stage 1: Region proposal network narrows down the number of candidate object locations to a small number (e.g. 1-2k).
  - Stage 2: Use OHEM (online-hard example mining) for training classifier network.

  -> There is manageable class balance between foreground and background.
- One-stage detector (Yolo, SSD):
  - Suffers from class imbalance problem.

1. Class imbalance problem

- The training procedure is dominated by easily classified background examples.
- It is typically addressed via bootstrapping or hard example mining. But they are not efficient enough.



2. Focal loss

- The loss function is reshaped to down-weight easy examples and thus focus training on hard negatives.
- A modulating factor *(1-p_t)^γ* is added to the Cross Entropy (CE) Loss where γ is tested from [0,5] in the experiment.

$$\mathrm{FL}(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log p_t$$

- $p_t = \begin{cases} p & y = 1 \\ 1 - p & y = -1 \end{cases}$
- $\alpha_t = \begin{cases} \alpha & y = 1 \\ 1 - \alpha & y = -1 \end{cases}$

2. Focal loss

- Properties:
  - When an example is misclassified and p_t is small, the modulating factor is near 1 and the loss is unaffected. As $p_t \rightarrow 1$, the factor goes to 0 and the loss for well-classified examples is down-weighted.
  - The focusing parameter γ smoothly adjusts the rate at which easy examples are down-weighted. When $\gamma = 0$, FL is equivalent to CE. When γ is increased, the effect of the modulating factor is likewise increased.

3. RetinaNet

An efficient in-network feature pyramid combined with anchor boxes

- Backbone:
  - ResNet is used for deep feature extraction.
  - FPN is used on top of ResNet for constructing a rich multi-scale feature pyramid from one single resolution input image.
- Classification and Box regression subnet:
  - Fully Convolutional Network attached to each FPN level.

3. RetinaNet

- Focal loss hyper-parameters:
  - Choose γ: $\gamma = 2$

    NOTE: RetinaNet is relatively robust to $\gamma \in [0.5, 5]$
  - Choose α: $\alpha = 0.25$

    NOTE: α should be decreased slightly as γ is increased

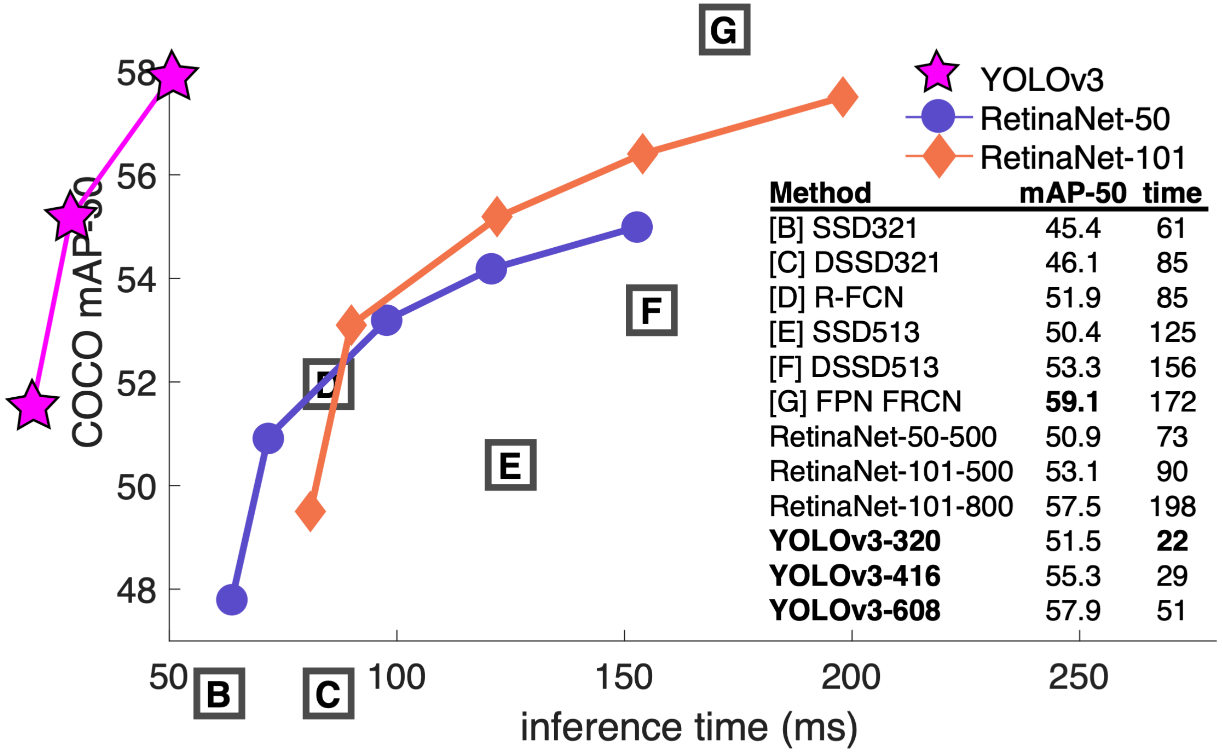
- Weights initialization
  - Weights: white Gaussian with standard deviation σ for some σ
  - Bias of all new conv layers (except the final one) in RetinaNet: $b = 0$
  - Bias of the final layer of the classification subnet: $b = -\log \frac{1-\pi}{\pi}$
    - Meaning: at the start of training, every anchor should be labeled as foreground with confidence π

3. RetinaNet

- Optimization:
    - Optimizer: synchronized SGD
    - Data augmentation: horizontal image flipping
    - Loss function:
        - Classification: focal loss
        - Regression: smooth L_1 loss

4. Comparision

Speed versus Accuracy Tradeoff



IV. Training and inference tricks

1. Training

- Observations
  - The amount available data is too small.
  - The models should be robust to transformations like translation, scaling, etc.
- Data augmentation
  - Translation
  - Scaling
  - Gaussian noise
  - Random flip
  - Color blend

2. Inference

- Ensembling: use an ensemble of some models (YOLO and RetinaNet)
- Tricks:
  - Multi-scale detection
- Results combination:
  - Assign weights to each of the models based on their validating accuracy.
  - Use Non-Max Weighted (NMW) instead of Non-Max Suppression (NMS).

Model ensembling

- Model weight:

$$w_i = \frac{\exp(s_i)}{\exp(s_1)+\exp(s_2)} \; \forall i \in \{1, 2\}$$

- Bounding box score:

$$b_i = w_1 p_1 + w_2 p_2$$

Non-max weighted

- Non-max suppression: $$B_{\text{pre}} = B_{\arg\max_i b_i}$$

- Non-max weighted:

$$B_{\text{pre}} = \frac{\sum_i c_i B_i}{\sum_i c_i} \text{ where } c_i = b_i \cdot \mathrm{IoU}(B_i, B_{\arg\max_i b_i})$$

Thanks for listening

Have a good day!