YOLOv3:

An Incremental Improvement

(Joseph Redmon, Ali Farhadi, 2018, Tech report)

IVPG Lab Seminar 2022.04.13
세종대학교 지능기전공학부
18학번 장윤정

Introduction

Abstract

We present some updates to YOLO! We made a bunch of little design changes to make it better. We also trained this new network that's pretty swell. It's a little bigger than last time but more accurate. It's still fast though, don't worry. At 320×320 *YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves* $57.9\ AP_{50}$ *in 51 ms on a Titan X, compared to* $57.5\ AP_{50}$ *in 198 ms by RetinaNet, similar performance but 3.8× faster. As always, all the code is online at* `https://pjreddie.com/yolo/`.

1. Introduction

Sometimes you just kinda phone it in for a year, you know? I didn't do a whole lot of research this year. Spent a lot of time on Twitter. Played around with GANs a little. I had a little momentum left over from last year [12] [1]; I managed to make some improvements to YOLO. But, honestly, nothing like super interesting, just a bunch of small changes that make it better. I also helped out with other people's research a little.

Actually, that's what brings us here today. We have a camera-ready deadline [4] and we need to cite some of the random updates I made to YOLO but we don't have a source. So get ready for a TECH REPORT!

The great thing about tech reports is that they don't need intros, y'all know why we're here. So the end of this introduction will signpost for the rest of the paper. First we'll tell you what the deal is with YOLOv3. Then we'll tell you how we do. We'll also tell you about some things we tried that didn't work. Finally we'll contemplate what this all means.

Bounding Box Prediction & Class Prediction

Bounding Box Prediction

- **In the YOLOv2**

- tx, ty, tw, th를 예측해서 구한 bx, by, bw, bh 값으로 L2 loss(sum of squared error loss) 계산해 학습

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$

YOLOv2's Bounding box prediction

$$t_x = \sigma^{-1}(b_x - c_x)$$
$$t_y = \sigma^{-1}(b_y - c_y)$$
$$t_w = \ln {b_w}/{p_w}$$
$$t_h = \ln {b_h}/{p_h}$$

YOLOv3's Bounding box prediction

- **In the YOLOv3**

- **YOLOv2와 동일한 방식**(sum of squared error loss)**으로 구한다?**

- **기존 식을 inverse해서 직접 구한 tx, ty, tw, th로 L1 loss를 계산해 학습한다?**

  - logistic을 그대로 사용하려면 제곱꼴이 없는 것이 학습에 좋은 영향을 준다..

L1 Loss: $\hat{t_x} - t_x = \hat{t_x} - \sigma(b_x - c_x)^{-1}$

L2 Loss: $(\hat{t_x} - t_x)^2 = (\hat{t_x} - \sigma(b_x - c_x)^{-1})^2$

- **각 bounding box에 대한 objectness score를 logistic regression으로 예측**

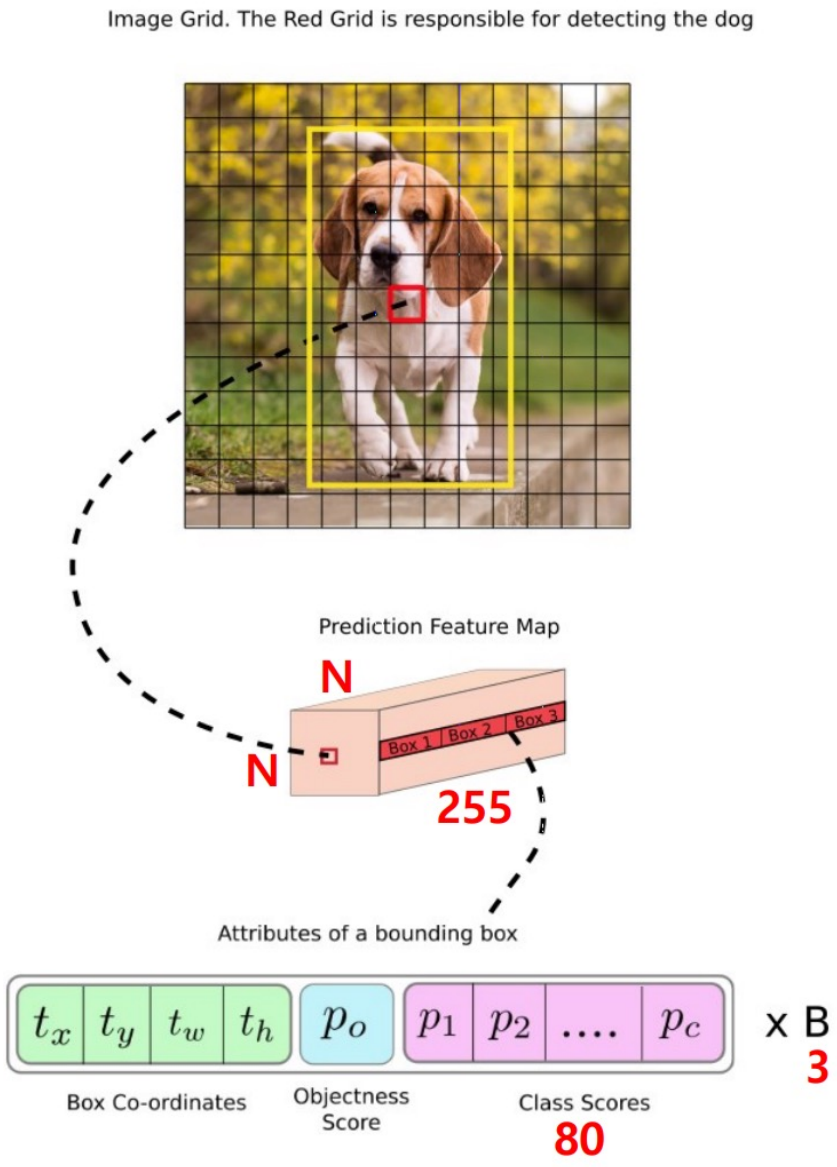
  - GT와 가장 많이 겹치는 bbox는 objectness score=1이 됨 (각 GT에 대해 이렇게 정해진 bbox 1개씩만 할당)

  - 가장 많이 겹쳐지지는 않았지만 일정 threshold(=0.5) 이상 겹쳐진 박스들은 유추에서 무시됨 (무시된 박스들은 위치 정보와 클래스에 관한 loss는 0이 되고, objectness loss만 계산)

Class Prediction

- **Multilabel classification** 사용해서 물체의 class 유추
- Softmax 사용하지 않고, class별로 각각 시그모이드를 취해 binary classification을 수행
- 사람&여자 처럼 중복되는 label로 이루어진 데이터셋을 고려하기 위해..

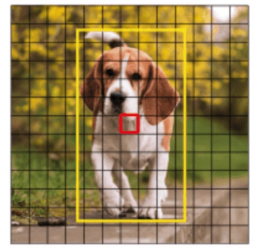
Predictions Across Scales



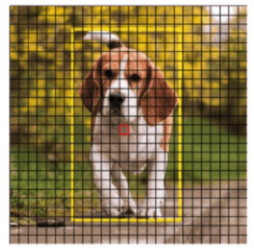
- **3가지 Scale**에 대해 각각 **3개의 Anchor Box**를 사용함
- Tensor size = N x N x {3 x (4 + 1 + 80)} = N x N x 255
  - N x N : Feature Map 그리드 셀의 크기
  - 3 : 3개의 bbox
  - 4 : bbox의 x, y, w, h 좌표들
  - 1 : objectness score
  - 80 : COCO dataset class scores

- Anchor Box는 YOLOv2 처럼 **K-means clustering** 사용해서 9개의 clusters 지정
  - small size object : (10x13),(16x30),(33x23)
  - middle size object : (30x61),(62x45),(59x119)
  - large size object : (116x90),(156x198),(373x326)

- 사용되는 BBOX의 개수 비교
  - YOLO : 98 boxes(7x7 grid cells, 2 boxes per cell)
  - YOLOv2 : 845 boxes(13x13 grid cells, 5 anchor boxes per cell)
  - **YOLOv3 : 10,647 boxes ((13x13x3)+(26x26x3)+(52x52x3)=10,647)** →



13 x 13



26 x 26

52 x 52

Feature Extractor (Darknet-53)

Softmax 결과에 대한 accuracy
(Dartnet-53은 어떻게 측정했지?)

billion operations (네트워크 전체 돌리는데 들어가는 연산량)

1초당 가능한 연산량 (Darknet-53이 가장 큼 → GPU를 연산에 잘 활용한다, 효율적)

1초당 몇 장의 frame 수행

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

YOLOv2에 비해 빠르다고 하지 못하는 이유

Table 2. **Comparison of backbones.** Accuracy, billions of operations, billion floating point operations per second, and FPS for various networks.

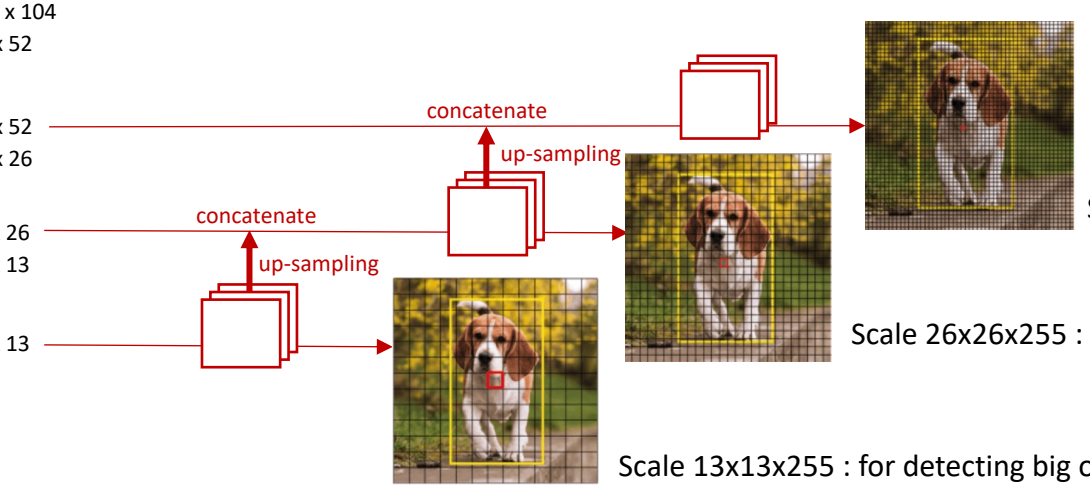
〈Darknet-53〉

	Type	Filters	Size	Output	
	Convolutional	32	3 × 3	~~256 × 256~~	416 x 416
	Convolutional	64	3 × 3 / 2	~~128 × 128~~	208 x 208
1×	Convolutional	32	1 × 1		
	Convolutional	64	3 × 3		
	Residual			~~128 × 128~~	208 x 208
	Convolutional	128	3 × 3 / 2	~~64 × 64~~	104 x 104
2×	Convolutional	64	1 × 1		
	Convolutional	128	3 × 3		
	Residual			~~64 × 64~~	104 x 104
	Convolutional	256	3 × 3 / 2	~~32 × 32~~	52 x 52
8×	Convolutional	128	1 × 1		
	Convolutional	256	3 × 3		
	Residual			~~32 × 32~~	52 x 52
	Convolutional	512	3 × 3 / 2	~~16 × 16~~	26 x 26
8×	Convolutional	256	1 × 1		
	Convolutional	512	3 × 3		
	Residual			~~16 × 16~~	26 x 26
	Convolutional	1024	3 × 3 / 2	~~8 × 8~~	13 x 13
4×	Convolutional	512	1 × 1		
	Convolutional	1024	3 × 3		
	Residual			~~8 × 8~~	13 x 13
	Avgpool		Global		
	Connected		1000		
	Softmax				

Table 1. **Darknet-53.**

(d) Feature Pyramid Network

- 상위 레벨의 계산된 semantic 정보를 재사용 하면서 여러 scale의 특징들을 효율적으로 사용



Scale 52x52x255 : for detecting small object

Scale 26x26x255 : for detecting medium object

Scale 13x13x255 : for detecting big object

Training

2.5. Training

We still train on full images with no hard negative mining or any of that stuff. We use multi-scale training, lots of data augmentation, batch normalization, all the standard stuff. We use the Darknet neural network framework for training and testing [14].

- **Hard negative mining 없이 full image 학습**
- **Hard negative mining** 이란?
  - hard negative : 실제로는 negative(배경)인데 positive(물체)로 잘못 예측하기 쉬운 데이터
  - 뽑힌 bounding box중 실제로 탐지된 object의 수는 매우 적고, background class에만 데이터가 몰려 네트워크가 제대로 학습되지 않는 현상을 방지하기 위해 적용되는 방법
  - **YOLOv3는 따로 background class 없이 objectness score를 사용해서 object 없는 것은 날려버리기 때문에 필요 없음**
- 그 외에도, multi-scale training, data augmentation, batch normalization 등 다양한 일반적인 방법 사용

How We Do & What this all means

Object size에 따른 AP

IOU threshold에 따른 AP

	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

Table 3. I'm seriously just stealing all these tables from [9] they take soooo long to make from scratch. Ok, YOLOv3 is doing alright. Keep in mind that RetinaNet has like 3.8× longer to process an image. YOLOv3 is much better than SSD variants and comparable to state-of-the-art models on the AP_{50} metric.

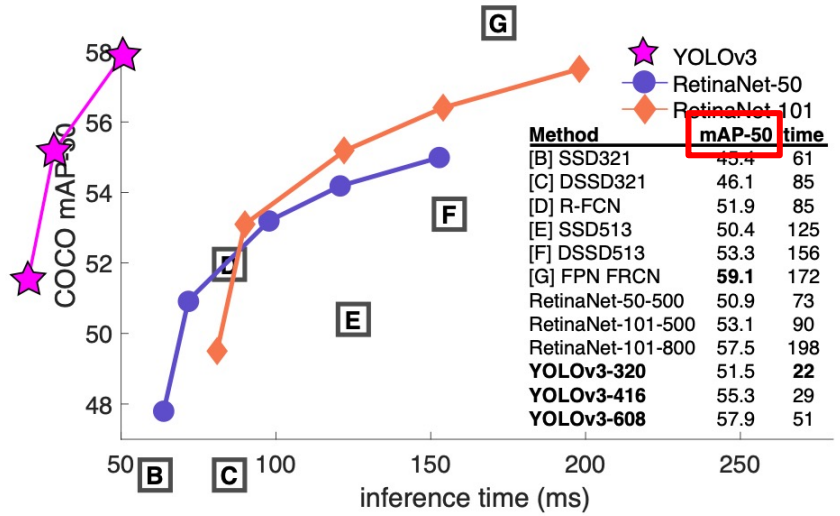


Figure 3. Again adapted from the [9], this time displaying speed/accuracy tradeoff on the mAP at .5 IOU metric. You can tell YOLOv3 is good because it's very high and far to the left. Can you cite your own paper? Guess who's going to try, this guy → [16]. Oh, I forgot, we also fix a data loading bug in YOLOv2, that helped by like 2 mAP. Just sneaking this in here to not throw off layout.

- COCO dataset 에서 사용하는 AP : IOU를 0.5부터 0.95까지 0.05간격으로 높여가면서 측정한 AP의 평균을 구하는 방식
- **YOLOv3 저자들은 이 metric을 마음에 들어 하지 않음**

  - YOLOv3가 AP에서는 성능이 낮고 **AP_50(IOU 0.5 이상이면 정답이라고 보는 matric, PASCAL VOC에서 전통적으로 쓰임)**에서는 성능이 좋음

  - 사람의 눈으로도 IOU의 미세한 차이를 구분하지 못하는데 0.5~0.95 사이에서 타이트하게 측정하는게 의미가 있는가?

- YOLO의 약점이었던 **small object**에 대한 성능이 많이 개선됨

Things We Tried That Didn’t Work

[시도해봤지만 도움 되지 않았던 방법들 소개]

1. Anchor box x, y offsets predictions

- $\hat{G}_x = P_w d_x(P) + P_x$
- $\hat{G}_y = P_h d_y(P) + P_y$
- $\hat{G}_w = P_w exp(d_w(P))$
- $\hat{G}_h = P_h exp(d_h(P))$

: dx, dy를 너비, 높이의 비율로 계산하게 했더니 성능 저하

2. Linear x, y, predictions instead of logistic regression

: x, y를 logistic 대신 linear regression으로 직접 뽑게 했더니 성능 저하

3. Focal Loss

: Retinanet에서 쓰인 class imbalance 문제를 완화시켜주는 loss를 사용해봤더니 2% 성능 저하

4. Dual IoU threshold and truth assignment

: Faster R-CNN에서 나온 방법, IOU threshold 0.7이상은 positive, 0.3미만은 negative, 나머지는 버림(네트워크를 더 헷갈리게 한다고 해서) --> 성능 저하

Rebuttal

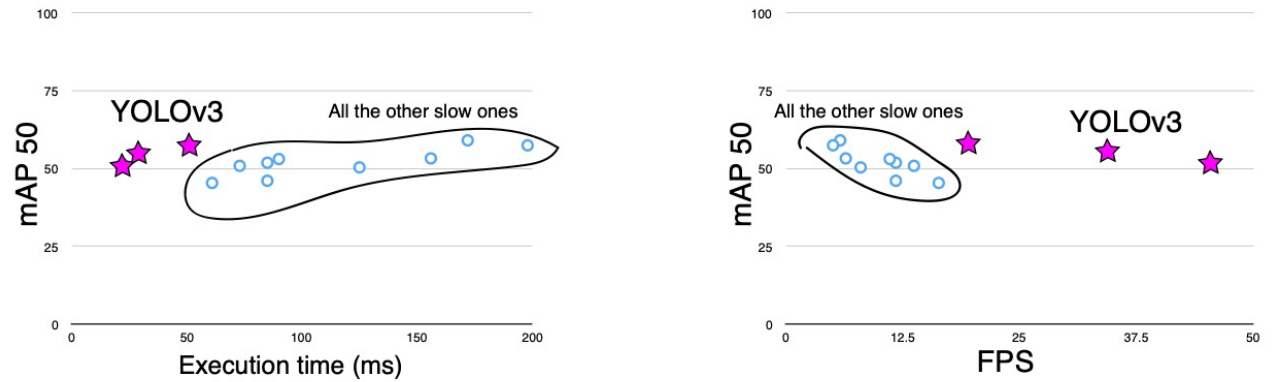


Figure 4. Zero-axis charts are probably more intellectually honest... and we can still screw with the variables to make ourselves look good!

[mAP_50에 대해]

- 0부터 그래프가 시작되도록 그려 봄 → 역시 연산시간도 빠르고 FPS도 매우 좋음
- PASCAL VOC가 IOU threshold를 0.5를 쓴 것은 덜 정확한 GT 데이터 때문
- COCO는 그것보다 labelling이 잘 되어있겠지만,

  그렇다고 해서 IOU를 0.5 이상으로 정확하게 그리는 것이 더 좋은 방식이라는 정당성을 갖지는 못함
- Classification보다 box regression을 더 중요하게 보는 것은 옳지 못함

Rebuttal

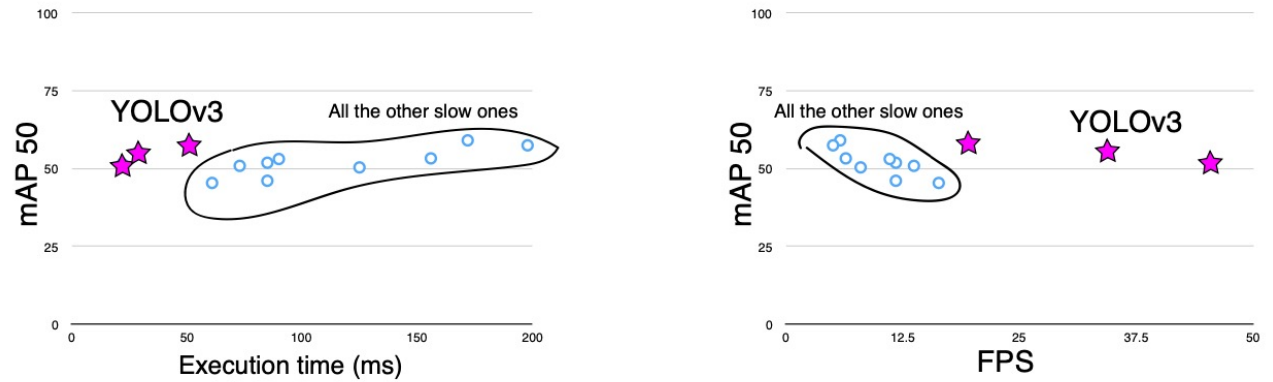
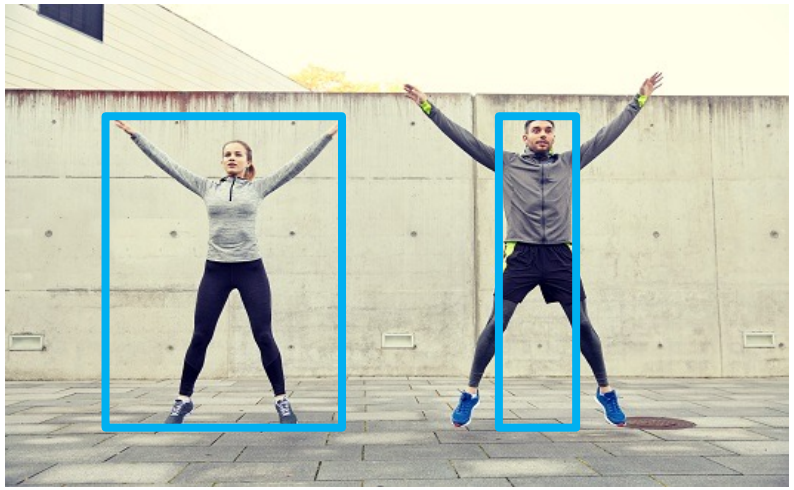


Figure 4. Zero-axis charts are probably more intellectually honest... and we can still screw with the variables to make ourselves look good!

[mAP_50에 대해]

- 0부터 그래프가 시작되도록 그려 봄 → 역시 연산시간도 빠르고 FPS도 매우 좋음
- PASCAL VOC가 IOU threshold를 0.5를 쓴 것은 덜 정확한 GT 데이터 때문
- COCO는 그것보다 labelling이 잘 되어있겠지만,

  그렇다고 해서 IOU를 0.5 이상으로 정확하게 그리는 것이 더 좋은 방식이라는 정당성을 갖지는 못함
- Classification보다 box regression을 더 중요하게 보는 것은 옳지 못함



이상적인 bounding box는 무엇인지 정답이 있을까?
저자는 Box 자체가 멍청하다고 생각함

Reference

- https://www.youtube.com/watch?v=HMgcvgRrDcA (PR-207: YOLOv3: An Incremental Improvement)
- https://www.youtube.com/watch?v=jqykPH3jbic ([Paper Review] YOLOv3: An Incremental Improvement)
- https://www.youtube.com/watch?v=cNFpo7kDf-s (박경찬 - YOLO)
- https://89douner.tistory.com/109 (13. YOLO V3)
- https://seongkyun.github.io/papers/2019/11/20/yolov3/ (YOLOv3 : An Incremental Improvement)
- https://darkpgmr.tistory.com/179 (YOLO와 성능지표(mAP, AP50))

감사합니다

IVPG Lab Seminar 2022.04.13

세종대학교 지능기전공학부

18학번 장윤정