YOLO v5

https://arxiv.org/pdf/1506.02640

Unified, Real-Time Object Detection

0. Abstract

- try to frame object detection as a regression prob to **spatially separated bounding boxes and associated class probs**
- whole detection pipeline이 single nw → can be optimized end to end directly on detect perfo
- localization error가 다른 모델에 비해 크나 background에 대한 FP 예측은 better
- learn **general representations of objects**

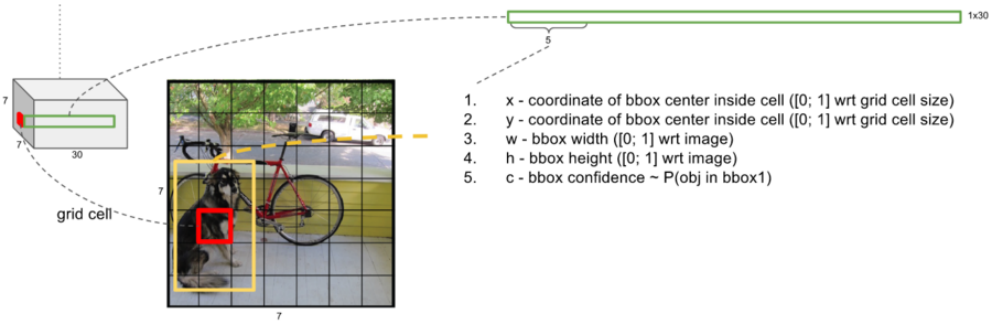
1. Introduction

- 실시간 정보를 전달하여 다양한 테스크를 수행하기 위해선 인간처럼 빠르고 정확히 detect하는 알고리즘 필요
- DPM : sliding window를 통해 clf가 spaced location을 균일하게 돌아가도록 함
- R-CNN : region proposal을 통해 potential bounding box를 생성하고, 그 박스들에 clf 돌림
  - 이후 post-processing을 통해 바운딩박스 정돈, 중복 제거, rescore 등 수행
  - 속도 느리고 최적화 힘듦 → 각 컴포넌트가 개별적으로 훈련되어야 하기에
- YOLO
  - 따라서 객체 인식을 하나의 회귀 문제로 접근 → **straight from img pixels to bounding box coordinates and class probabilities**
  - 전체 이미지를 학습하고, 직접 감지 성능 최적화하는 방식
    - 속도 빠름
      - 복잡한 파이프라인 불필요, 새로운 이미지를 NN에 돌리면 끝
      - 실시간 스트리밍 비디오를 처리할 수 있을 정도로 적은 latency 보임 → 다른 시스템에 비해 2배 이상의 mean avg precision
    - Reason globally
      - 전체 이미지를 보기에 **내재적으로 class와 appearance에 대한 contextual info를 encode**
      - Fast R-CNN은 큰 context를 보지 못하기 때문에 background error 발생 → 그에 비해 yolo는 에러 절반 이하
    - Learn generalizable representaions of object
      - 새로운 도메인이나 예상치 못한 인풋에 덜 break down
    - 빠르게 인식하다보면 물체를 세심하게 localize하기 힘듦 → trade off 존재

2. Unified Detection

- 전체 이미지에서 피처를 사용해서 각 바운딩 박스를 예측하고 동시에 모든 클래스에 걸쳐 모든 바운딩박스를 예측 → 모든 이미지와 물체에 대해 globally 사고
- high avg precision 유지하면서 end to end training & real time speeds 가능하게~!

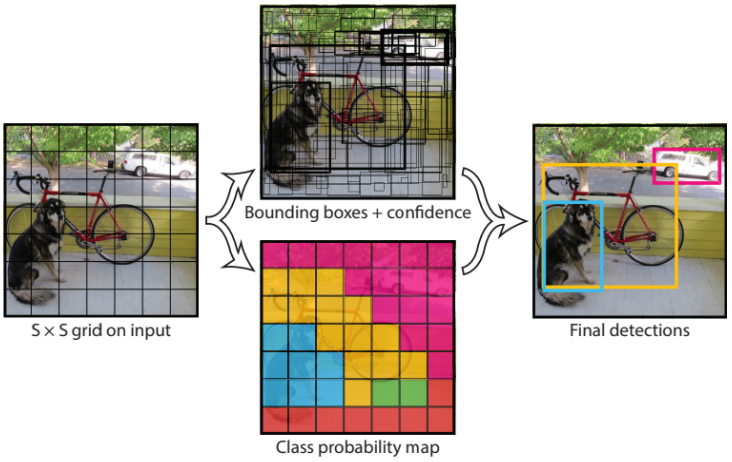
- 이미지는 S*S grid로 나누어 만약 물체의 센터가 grid cell로 빠지면(fall into),해당 그리드 셀이 인식할 책임이 생기는 것
- 각 그리드 셀은 B 바운딩 박스와 confidence score 를 예측
  - confidence score : 얼마나 모델이 그 박스에 물체가 포함되어 있는 게 확실한지 그리고 예측한 상자에 얼마나 정확한 지를 반영하는 변수 Pr(object)*IOU
  - IOU (Intersection over Union) : ground truth과 예측 값이 얼마나 겹치는지를 나타내는 지표 → area of overlap / area of union
  - 만약 박스 안에 물체가 없다면 confidence score = 0
- 각 바운딩 박스에는 5 predictions : x,y, w,h, confidence
  - (x,y) 좌표는 박스의 중심
  - weight, height는 이미지의 너비, 높이
  - confidence pred는 IOU



> red box : 7*7 grid
>
> yellow box : grid cell에서 예측한 경계 박스
>
> 그리드 셀이 7*7이므로 총 98개의 경계 박스가 만들어지고, 경계 박스 내부에 오브젝트가 있을 것 같으면 바운딩 박스를 굵게 그려줌 → 다수 바운딩 박스 중 threshhold보다 작으면 삭제해서 최종 하나의 경계 박스만 남김

- 각 그리드 셀은 C 조건부 클래스 확률 예측 Pr(Class i | object)
  - 그리드셀 별 한 세트 클래스 확률 예측
  - 테스트에서는 조건부 확률과 각 박스의 confidence 예측 곱하는데 이는 각 박스마다 class specific confidence store

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



These predictions are encoded as an S × S × (B * 5 + C) tensor.

2.1 Network Design

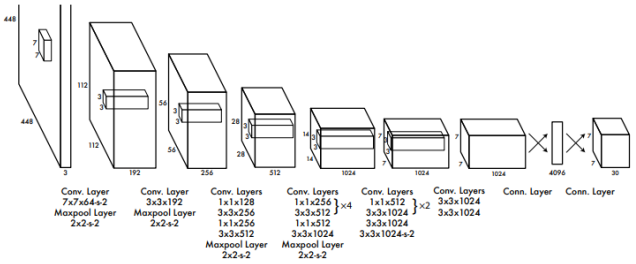


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

> inspired by GooLeNet (Inception 모듈 대신 1*1 reduction layer and 3*3 conv)
>
> 24 conv and 2 FC
>
> final output 7*7*30 tensors

2.2 Training

1. pretrain : first 20 conv layers and avg pooling and FC
2. convert to perform detection
   - 사전학습 시conv와 connected L를 모두 추가하는 것이 성능 향상에 도움 from Ren et al.
   - 4 conv + 2 FC w/ randomly initialized weights
   - input resolution 448*448 for fine grained visual info
3. final layer : class prob + bounding box 좌표 예측
   - w,h normalize (0~1 사이)
   - (x,y)도 0~1 사이로
4. etc
   - linear activation f

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

- optimize for SSE
  - 최적화 쉽지만
  - avg precision max 힘듦 → clf 에러와 loc 에러 가중치를 동일하게 부여하는 문제
  - 대부분 다수 그리드 셀에는 물체가 없는데 이는 confidence socre을 0으로 만들어서 종종 물체를 가진 셀의 그레디언트를 압도함 → model instability 문제

  ⇒ 바운딩 박스 좌표 예측 loss는 증가시키고, confidence 예측 loss는 감소시켜 완화
  - 두 파라미터 사용 : **λcoord = 5 and λnoobj = 0.5**
- responsible
  - predictor 중 가장 높은 current IOU 가진 애한테 물체를 할당
  - **specialization between the bounding box predictors**

loss function:

$$\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)
\end{aligned}$$

where $\mathbb{1}_i^{\text{obj}}$ denotes if object appears in cell i and $\mathbb{1}_{ij}^{\text{obj}}$ denotes that the jth bounding box predictor in cell i is "responsible" for that prediction.

- loss f는 그리드 셀에 물체가 있을 때에만 clf 에러에 패널티를 부여
- 해당 predictor가 responsible할 때만 바운딩 박스 좌표 에러에 패널티를 부여
- 배치 사이즈 64, 모멘텀 0.9, decay 0.0005
- 오버 피팅 피하기 위해
  - drop out 0.5
  - data augmentation : random scaling and translations up to 20% of the orginal img size
  - exposure and saturation 랜덤 조절

2.3 Inference

- 일부 큰 오브젝트 혹은 경계에 겹치는 물체가 well localized by multiple cells

⇒ non maximal suppression을 통해 다중 감지 문제 해결

2.4 Limitations

- 각 그리드 셀은 두 박스와 한 클래스만 예측 가능
- 군락 떼처럼 작은 오브젝트들에는 좋은 결과가 어려움
- 예측에 상대적으로 coarse(rough)한 피처들을 사용함
- 바운딩 박스 크기와 무관하게 에러를 같게 다룸 (작은 박스의 에러가 IOU에 더 큰 effect)

⇒ main src : incorrect localizations

4. Experiments

4.1. Comparison to Other Real-Time Systems

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

Table 1: Real-Time Systems on PASCAL VOC 2007. Comparing the performance and speed of fast detectors. Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real-time detector. YOLO is 10 mAP more accurate than the fast version while still well above real-time in speed.

- With 52.7% mAP, it is more than twice as accurate as prior work on real-time detection. YOLO pushes mAP to 63.4% while still maintaining real-time performance.
- YOLO using VGG-16 is more accurate but also significantly slower than YOLO.
- Fastest DPM still misses real-time performance by a factor of 2/ is limited by DPM's relatively low accuracy on detection compared to neural network approaches
- R-CNN minus R is much faster than R-CNN but it still falls short of real-time and takes a significant accuracy hit from not having good proposals.
- Faster R-CNN achieves 7 fps while a smaller, less accurate one runs at
  18 fps

4.2. VOC 2007 Error Analysis

- Fast R-CNN과의 비교

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

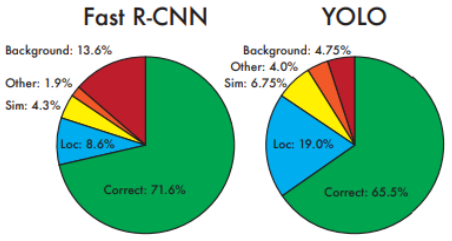


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

> YOLO는 localization 에러가 크고
> Fast R-CNN background 에러가 컸음(13.6% of it's top detections are false positives that don't contain any objects.)

4.3. Combining Fast R-CNN and YOLO

	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**

Table 2: Model combination experiments on VOC 2007. We examine the effect of combining various models with the best version of Fast R-CNN. Other versions of Fast R-CNN provide only a small benefit while YOLO provides a significant performance boost.

- 일부 성능 증가가 확인됨
- YOLO makes different kinds of mistakes at test time that it is so effective at boosting Fast R-CNN's performance

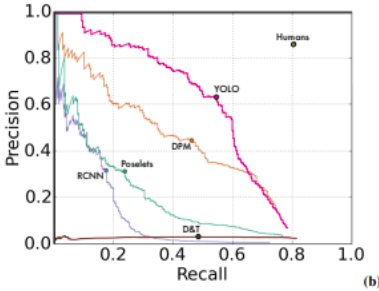
4.4. VOC 2012 Results

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

Table 3: PASCAL VOC 2012 Leaderboard. YOLO compared with the full comp4 (outside data allowed) public leaderboard as of November 6th, 2015. Mean average precision and per-class average precision are shown for a variety of detection methods. YOLO is the only real-time detector. Fast R-CNN + YOLO is the forth highest scoring method, with a 2.3% boost over Fast R-CNN.

- 타 모델에 비해 작은 물체들을 다루는 것에서 고전(병, 양,,,etc)
- cat, train category에서는 좋은 성과 (리더보드 spots up)

4.5. Generalizability: Person Dtection in Artwork



(a) Picasso Dataset precision-recall curves.

	VOC 2007 AP	Picasso AP	Picasso Best F_1	People-Art 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.

Figure 5: Generalization results on Picasso and People-Art datasets.

- the Picasso Dataset and the People-Art Dataset을 통해 예술 작품에서의 사람 인식 테스트
- R-CNN은 실제 이미지에 tune 되어진 Selective Search를 사용하기에 artwork 적용 결과 drop off/ clf가 작은 region만 보고 좋은 proposal 필요
- DPM은 물체에 대한 강한 spatial model이기에 R-CNN 만큼의 degrade는 아니었으나 낮아지긴 함
- YOLO는 물체의 사이즈와 쉐입 + 물체간 관계 + 주로 물체가 나타나는 곳이 모델되어 있기에 좋은 예측 보

5. Real-Time Detection in the Wild

Figure 6: Qualitative Results. YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.

- 결과 시스템은 interactive and engaging
- 트래킹 시스템처럼 웹캠에 부착하면 이미지를 개별적으로 처리하기에 물체가 움직이고 모양이 변하는 것을 잡아

6. Conclusion

> We introduce YOLO, a unified model for object detection. Our model is simple to construct and can be trained directly on full images. Unlike classifier-based approaches, YOLO is trained on a loss function that directly corresponds to detection performance and the entire model is trained jointly. Fast YOLO is the fastest general-purpose object detector in the literature and **YOLO pushes the state-of-the-art in real-time object detection**. YOLO also generalizes well to new domains making it ideal for applications that rely on fast, robust object detection .

7. More about model architecture (나중에 읽어보기)

https://iambeginnerdeveloper.tistory.com/180

https://dkssud8150.github.io/posts/YoloV5/