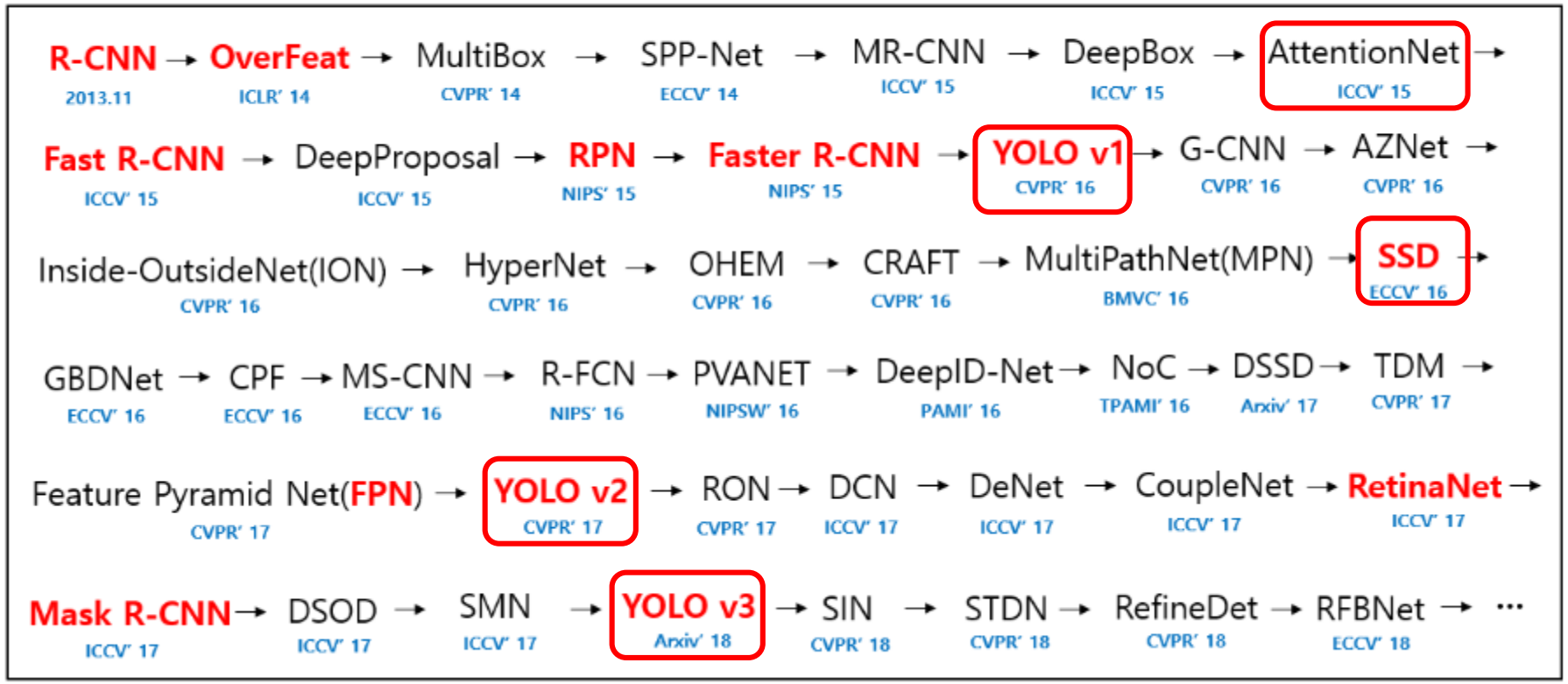
Image Detection 방법론: AttentionNet, SSD, YOLO, YOLOv2

참고자료

1. 논문으로 짚어보는 딥러닝의 맥, Image Detection 방법론: AttentionNet, SSD, YOLO, YOLOv2, https://www.edwith.org/deeplearningchoi/lecture/15579/
2. Yoo, Donggeun, et al. "Attentionnet: Aggregating weak directions for accurate object detection. " *Proceedings of the IEEE International Conference on Computer Vision*. 2015.
3. A paper list of object detection using deep learning., https://github.com/hoya012/deep_learning_object_detection
4. Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
5. What is Faster R-CNN, http://incredible.ai/deep-learning/2018/03/17/Faster-R-CNN/
6. PR-016: You only look once: Unified, real-time object detection, https://www.youtube.com/watch?v=eTDcoeqj1_w&t=56s
7. [분석] YOLO, https://curt-park.github.io/2017-03-26/yolo/
8. [Deeplearning] YOLO!, You Only Look Once : Unified, Real-Time object Detection, http://dhhwang89.tistory.com/51
9. YOLO: You only look once (How it works), https://www.youtube.com/watch?v=L0tzmv--CGY&feature=youtu.be
10. Liu, Wei, et al. "Ssd: Single shot multibox detector." *European conference on computer vision*. Springer, Cham, 2016.
11. [논문] SSD: Single Shot Multibox Detector 분석, https://taeu.github.io/paper/deeplearning-paper-ssd/
12. Deepsystem.io, SSD: Single Shot Multibox Detector, https://docs.google.com/presentation/d/1rtfeV_VmdGdZD5ObVVpPDPIODSDxKnFSU0bsN_rgZXc/pub?start=false&loop=false&delayms=3000&slide=id.g179f601b72_0_51
13. SSD: Single Shot MultiBox Detector (How it works), https://www.youtube.com/watch?v=P8e-G-Mhx4k
14. [SSD: Single Shot MultiBox Detector], https://www.youtube.com/watch?v=Gc233mo6r9c
15. Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." *arXiv preprint* (2017).
16. PR-023: YOLO9000: Better, Faster, Stronger, https://www.youtube.com/watch?v=6fdclSGgeio
17. [논문] YOLO9000: Better, Faster, Stronger 분석, https://taeu.github.io/paper/deeplearning-paper-yolov2/
18. PR-023: YOLO9000: Better, Faster, Stronger, https://www.slideshare.net/JinwonLee9/pr12-yolo9000
19. Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." *arXiv preprint arXiv:1804.02767* (2018).
20. [논문] YOLOv3: An Incremental Improvement 분석, https://taeu.github.io/paper/deeplearning-paper-yolov3/
21. [Deeplearning] Yolov3: An Incremental Improvement, http://dhhwang89.tistory.com/138
22. What's new in YOLO v3?, https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b
23. How to implement a YOLO (v3) object detector from scratch in PyTorch: Part 1, https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/

paper list from 2014 to now(2018)



https://github.com/hoya012/deep_learning_object_detection

Attentionnet: Aggregating weak directions for accurate object detection

Figure 1. Real detection examples of our detection framework. Starting from an image boundary (dark blue bounding box), our detection system iteratively narrows the bounding box down to a final human location (red bounding box).

논문의 특징 및 기여한 점

1. Cast an object detection problem as an iterative classification problem
   -> 지금까지 object detection problem은 바운딩 박스를 가지고 물체를 찾는 방식이였는데, 이 논문에서는 바운딩 박스의 크기를 순차적으로 변경하여 변경한 바운딩 박스의 크기가 물체와 만나게 하는 문제로 변경하였다.

2. 이 알고리즘의 전제 조건은 이미지안에 찾는 물체가 하나 있다고 가정한다.

3. 기존의 object detection method와는 다르게 object proposal, classifiers, post bounding box regressions 가 필요없다.

4. 이 논문이 공개된 시점에서 single class object detection tasks에서 SOTA 알고리즘이다. PASCAL VOC 2007/2012 데이터셋에서 65% (AP)을 달성

5. 제안된 방법론은 원래 DARPA challenge에서 로봇이 밸브를 정확히 인식하기 위해 만든 방법이라고 한다. 하지만, 실제 대회에서는 사용되지 않았다고 한다.

Attentionnet: Aggregating weak directions for accurate object detection

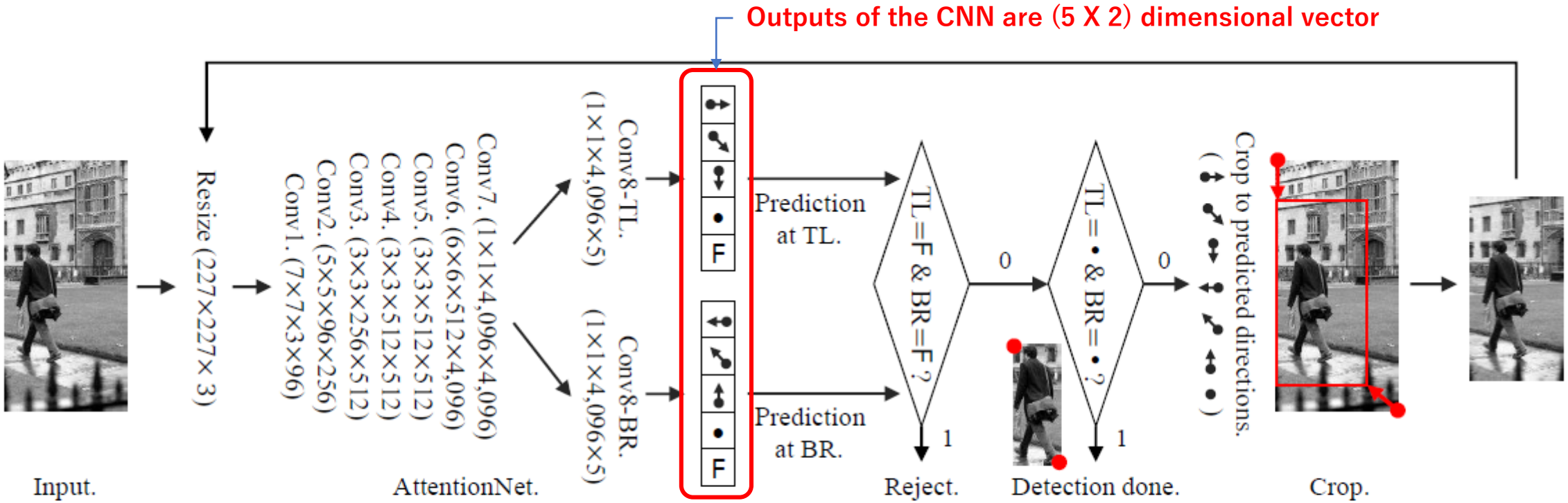


Figure 2. A pipeline of our detection framework. AttentionNet is composed of two final layers for top-left (TL) and bottom-right (BR) of the input image domain. Each of them outputs a direction (→ ↘ ↓ for TL, ← ↖ ↑ for BR) where each corner of the image should go to for the next step, or a "stop" sign (•), or "non-human" sign (F). When AttentionNet outputs "non-human" in both layers, the image is rejected. The image is cropped according to the weak directions and fed to AttentionNet again, until it meets "stop" in both layers.

Attentionnet: Aggregating weak directions for accurate object detection

Figure 3. Real examples of crop-augmentation for training AttentionNet. The target instance is the right man. Dashed cyan bounding boxes are ground-truths, and the blue bounding boxes are the augmented regions. Red arrows/dots denote their ground truths.

학습 단계에서는 아래의 3단계의 프로세스를 통해 랜덤으로 positive region을 설정한다.

1. 이미지안에서 랜덤 바운딩 박스를 생성한다.
   Positive region은 반드시 타켓(cyan bounding box)을 50% 이상 포함해야 한다.
   랜덤 바운딩 박스가 타켓을 50%이하로 포함하는 경우 negative region이 된다.

2. 그림3의 왼쪽 위의 이미지처럼 Positive region은 여러개의 인스턴스를 포함할 수도 있다.
   하지만 타켓 인스턴스는 반드시 가장 큰 영역을 차지하는 인스턴스로 설정한다.
   (그림의 경우 오른쪽 아저씨가 된다.)

3. Region은 다양한 종횡비와 스케일을 갖도록 구성한다.

위의 3단계를 거쳐 생성된 training dataset을 이용하여 학습을 진행한다.

Attentionnet: Aggregating weak directions for accurate object detection

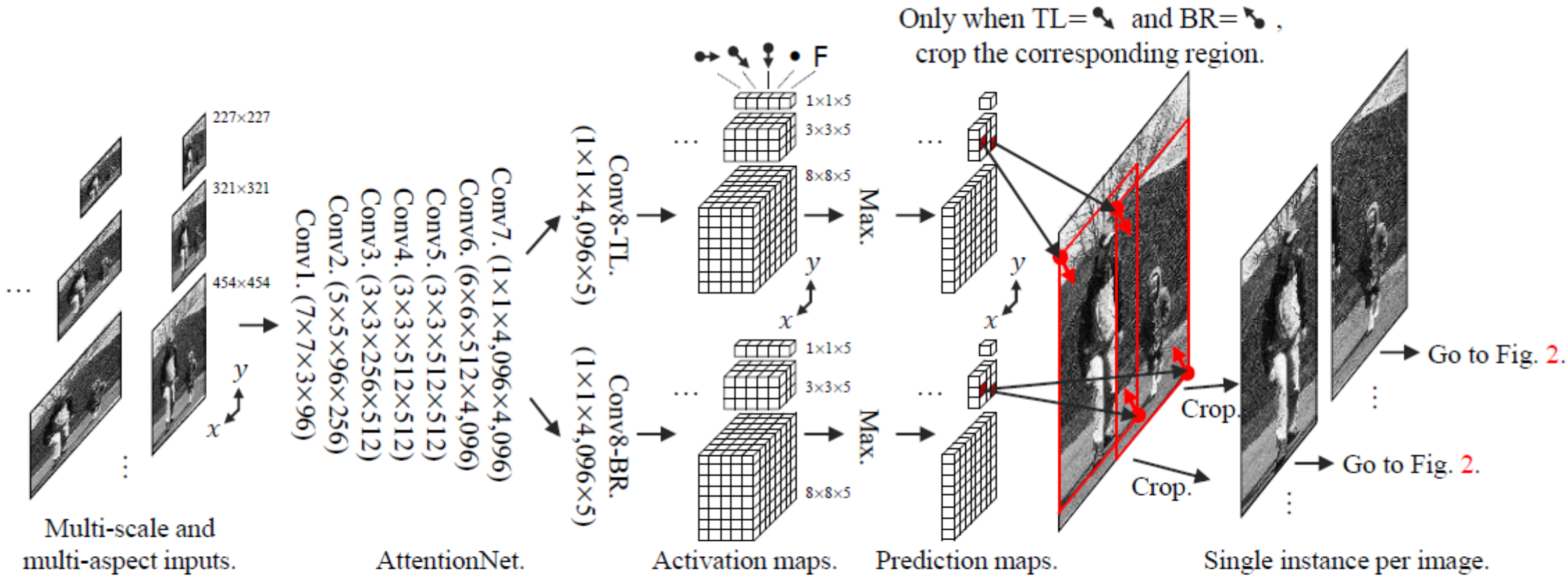
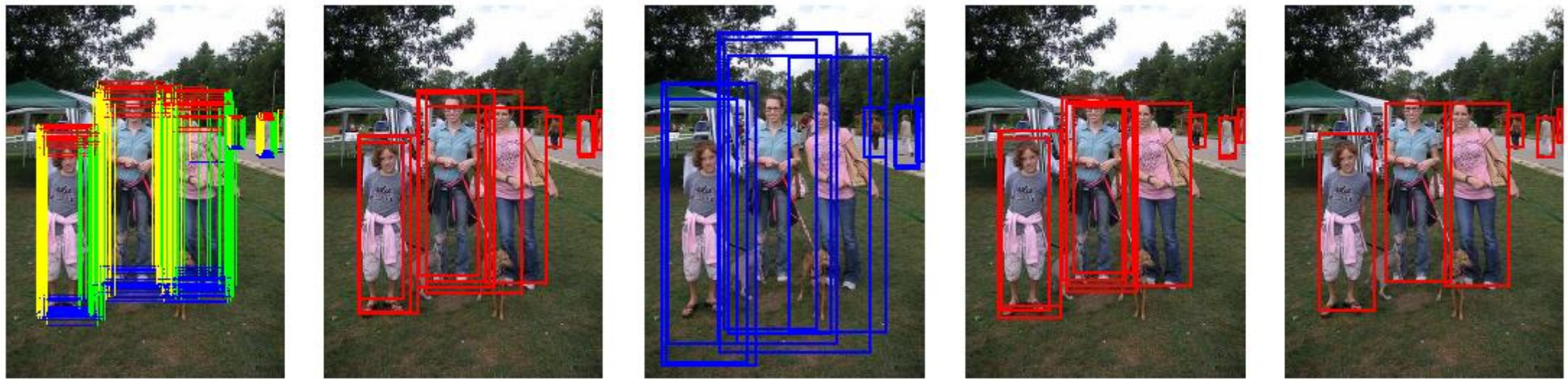


Figure 4. Extracting single-instance regions, where a single instance is included only. Multiple inputs with multiple scales/aspects are fed to AttentionNet, and prediction maps are produced. Only image regions satisfying $\{\searrow_{TL}, \nwarrow_{BR}\}$ are regarded as the single-instance regions. These regions are fed to AttentionNet again for final detection. Note, the CNN here and that in Fig. 2 are the same one, not separated.

Attentionnet: Aggregating weak directions for accurate object detection



(a) Initial detections. (b) Initial merge. (c) Re-initialize. (×2.5) (d) Re-detections. (e) Final merge.

Figure 6. Real examples of our detection procedure, including initial results (a∼b) and refinement (c∼e). Initially detected candidates come from Fig. 4 followed by Fig. 2 are merged by an intersection over union (IoU) of 0.8. We extend each merged box to 2.5-times larger size, and feed them to Fig. 2 again. Finally we merge the second results by an IoU of 0.5.

You only look once: Unified, real-time object detection

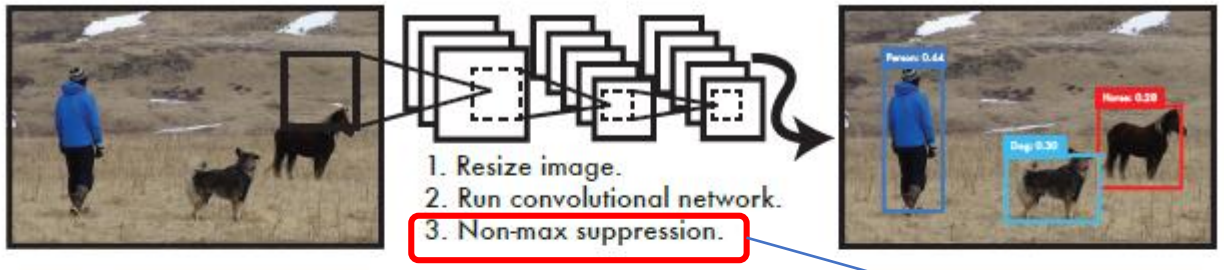


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

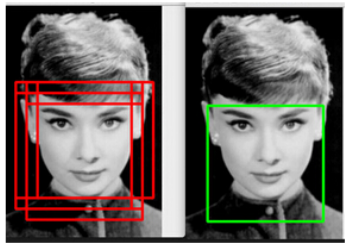
Non-Maximum Suppression

Faster R-CNN에 대한 학습이 완료된 후, RPN 모델을 예측시키면 아마도 한 객체당 여러개의 proposals (bounding boxes) 값을 얻을 것 입니다. 이유는 anchors자체가 어떤 객체에 중복이 되기 때문에 proposals 또한 여러개가 되는 것 입니다.

문제를 해결하기 위해서 non-maximum suppression (NMS) 알고리즘을 사용해서 proposals의 갯수를 줄이도록 합니다. NMS를 간단히 설명하면 먼저 IoU값으로 proposals을 모두 정렬시켜놓은뒤, RoI점수가 가장 높은 proposal과 다른 proposals에 대해서 overlapping을 비교한뒤 overlapping이 높은 것은 특정 threshold이상이면 지워버리며 이 과정을 iterate돌면서 삭제시킵니다.

직관적으로 설명하면 RoI가 높은 bounding box가 있는데.. 이 놈하고 overlapping 되는 녀석들중 특정 threshold이상이면 proposals에서 삭제시켜버리는 형태입니다. 그러면 서로 오버랩은 되지 않으면서 RoI가 높은 녀석들만 남게 됩니다.

일반적으로 threshold의 값은 0.6~0.9 정도로 합니다.



NMS 는 training 시에도 학습 속도를 높이기 위해서 사용 할 수 있습니다.

http://incredible.ai/deep-learning/2018/03/17/Faster-R-CNN/

논문의 특징 및 기여한 점

1. 기존의 object detection problem은 multi task object function으로 bounding box을 찾기 위한 regression 문제와, detection score을 높이는 classification 문제로 나누어 해결했는데, YOLO에서는 regression problem으로 풀어서 한번의 feedforward을 통해서 object detection을 할 수 있다.

2. 성능은 Faster R-CNN계열의 네트워크와 비슷하지만, 속도를 높여서 베이스 모델은 45FPS, Tiny 모델은 155FPS까지 처리속도를 높였다.

3. 기존 object detection 방법론의 장점들을 모아서 성능을 극대화 했다. RCNN계열의 region proposal method을 이용

You only look once: Unified, real-time object detection

Appendix: Intersection over Union (IOU)

Unified Detection

- All BBox, All classes

1) Image → **S x S** grids

2) grid cell

→ **B**: BBoxes and Confidence score

→ **C**: class probabilities w.r.t #classes

$\Pr(\text{Class}_i \mid \text{Object})$

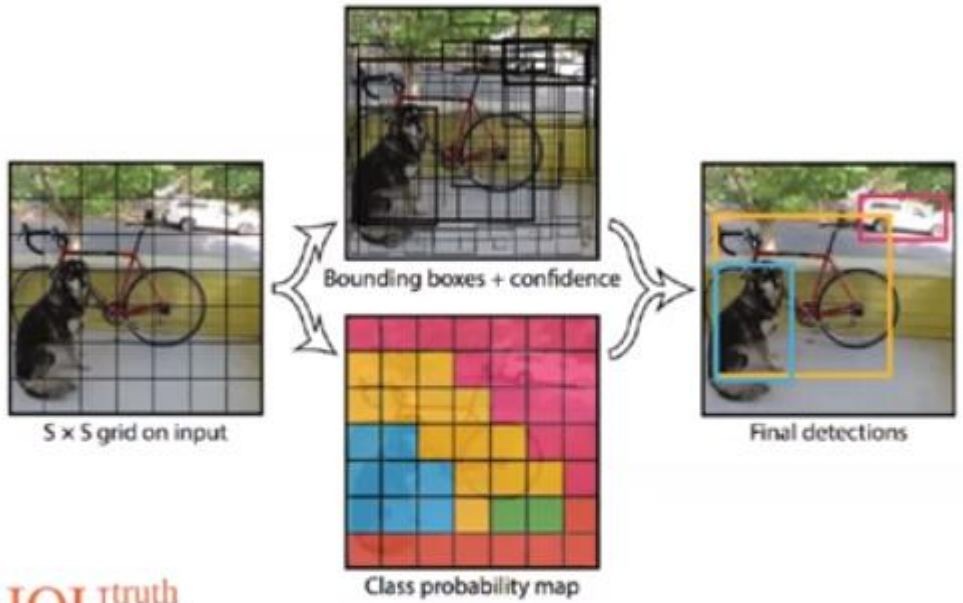


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

PR-016: You only look once: Unified, real-time object detection, https://www.youtube.com/watch?v=eTDcoeqj1_w&t=56s

You only look once: Unified, real-time object detection

Unified Detection

- Predict one set of class probabilities per grid cell, regardless of the number of boxes B.
- At test time, individual box confidence prediction

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

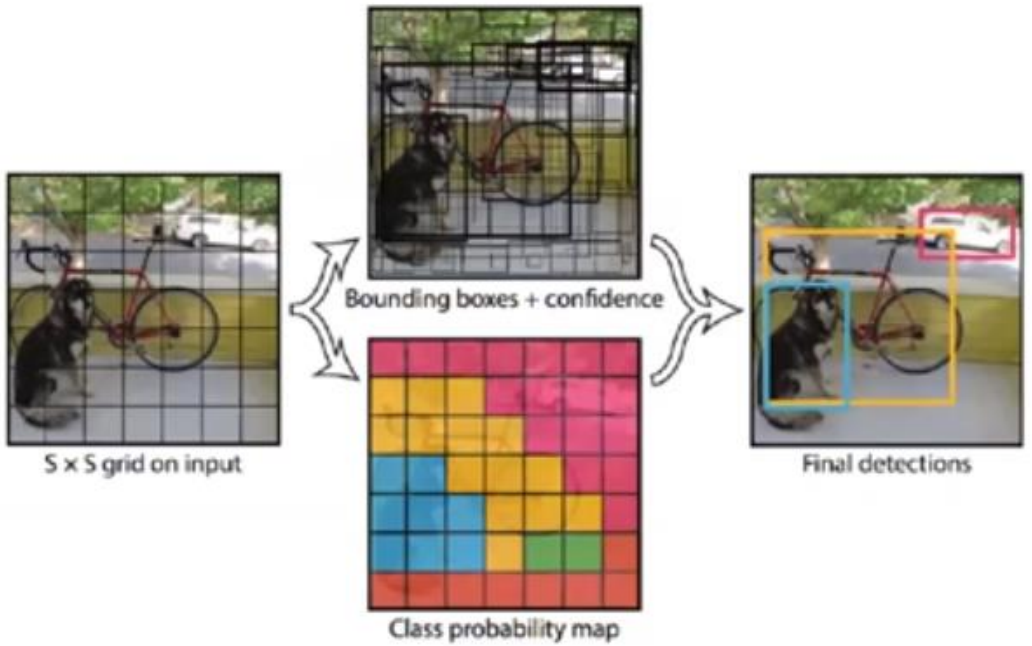
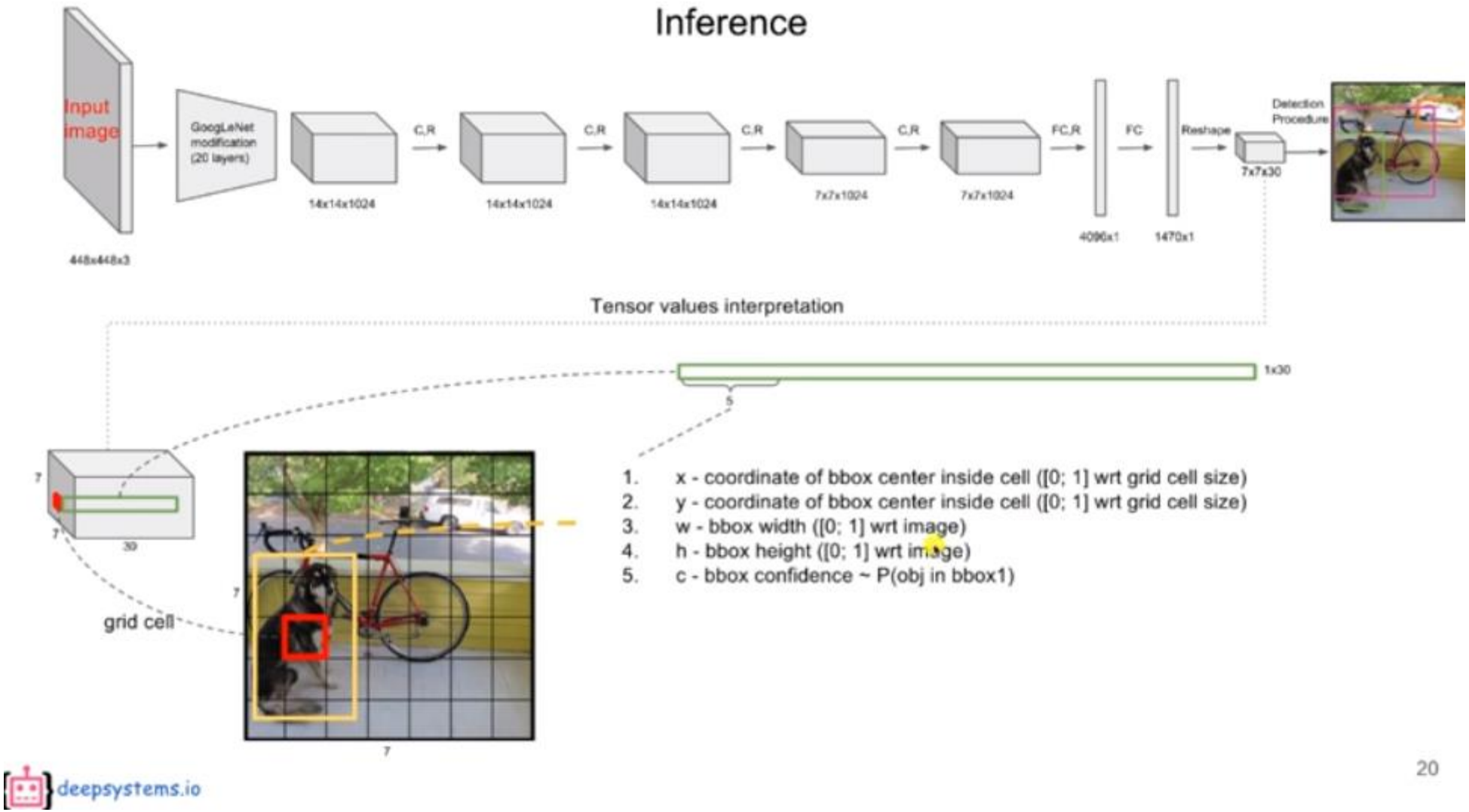


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

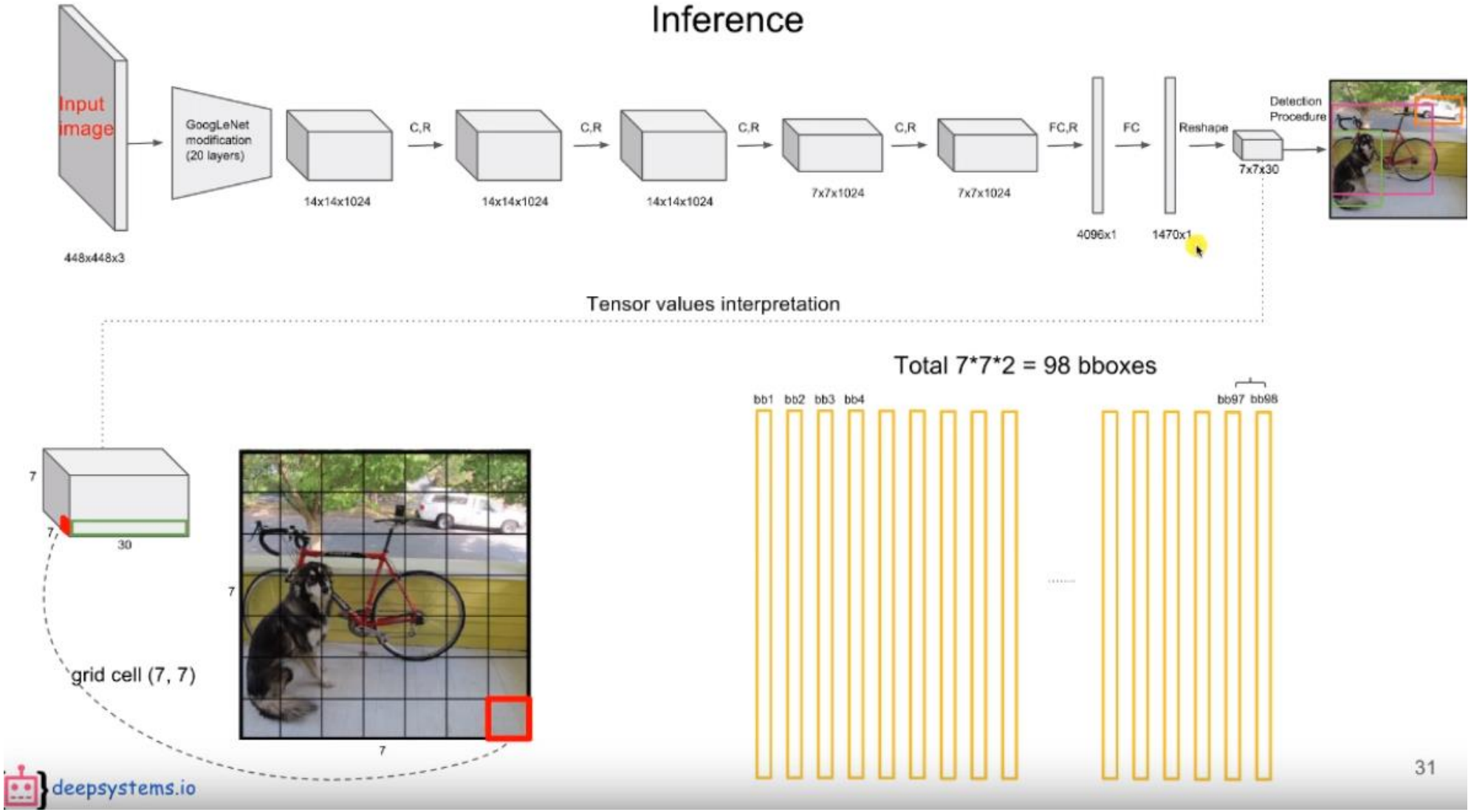
PR-016: You only look once: Unified, real-time object detection, https://www.youtube.com/watch?v=eTDcoeqj1_w&t=56s

You only look once: Unified, real-time object detection



YOLO: You only look once (How it works), https://www.youtube.com/watch?v=L0tzmv--CGY&feature=youtu.be

You only look once: Unified, real-time object detection



YOLO: You only look once (How it works), https://www.youtube.com/watch?v=L0tzmv--CGY&feature=youtu.be

You only look once: Unified, real-time object detection

Loss Function:

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \quad (1)$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \quad (2)$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \quad (3)$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \quad (4)$$

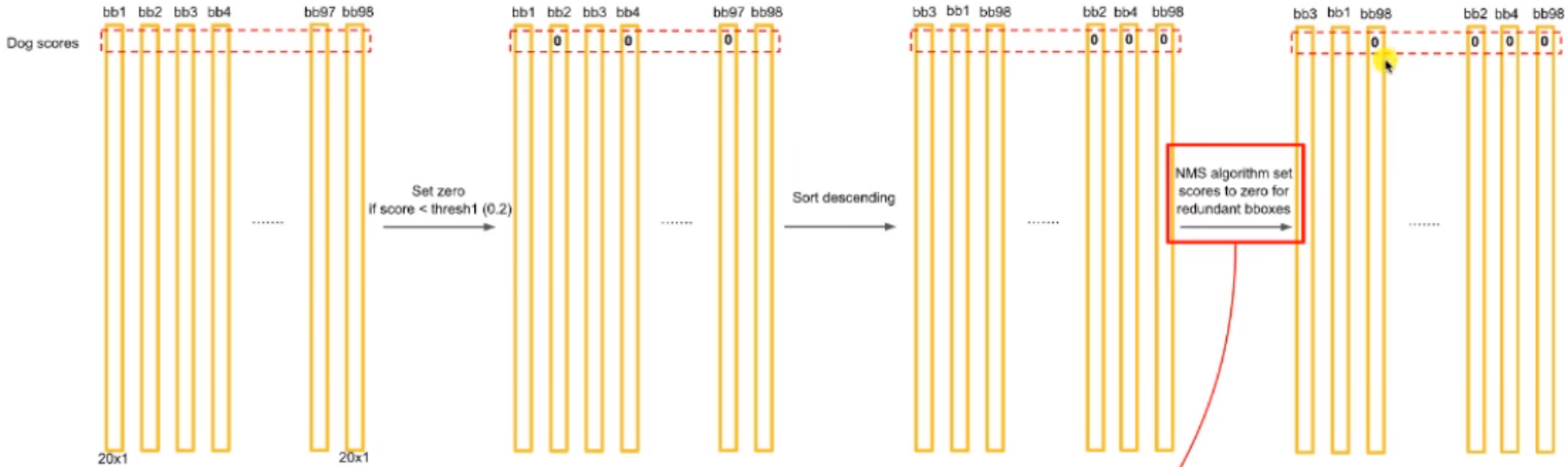
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \quad (5)$$

(1) Object가 존재하는 grid cell i의 predictor bounding box j에 대해, x와 y의 loss를 계산.
(2) Object가 존재하는 grid cell i의 predictor bounding box j에 대해, w와 h의 loss를 계산. 큰 box에 대해서는 small deviation을 반영하기 위해 제곱근을 취한 후, sum-squared error를 한다.(같은 error라도 larger box의 경우 상대적으로 IOU에 영향을 적게 준다.)
(3) Object가 존재하는 grid cell i의 predictor bounding box j에 대해, confidence score의 loss를 계산. (C_i = 1)
(4) Object가 존재하지 않는 grid cell i의 bounding box j에 대해, confidence score의 loss를 계산. (C_i = 0)
(5) Object가 존재하는 grid cell i에 대해, conditional class probability의 loss 계산. (Correct class c: $p_i(c)$=1, otherwise: $p_i(c)$=0)

λ_{coord}: coordinates(x,y,w,h)에 대한 loss와 다른 loss들과의 균형을 위한 balancing parameter.
λ_{noobj}: obj가 있는 box와 없는 box간에 균형을 위한 balancing parameter. (일반적으로 image내에는 obj가 있는 cell보다는 obj가 없는 cell이 훨씬 많으므로)

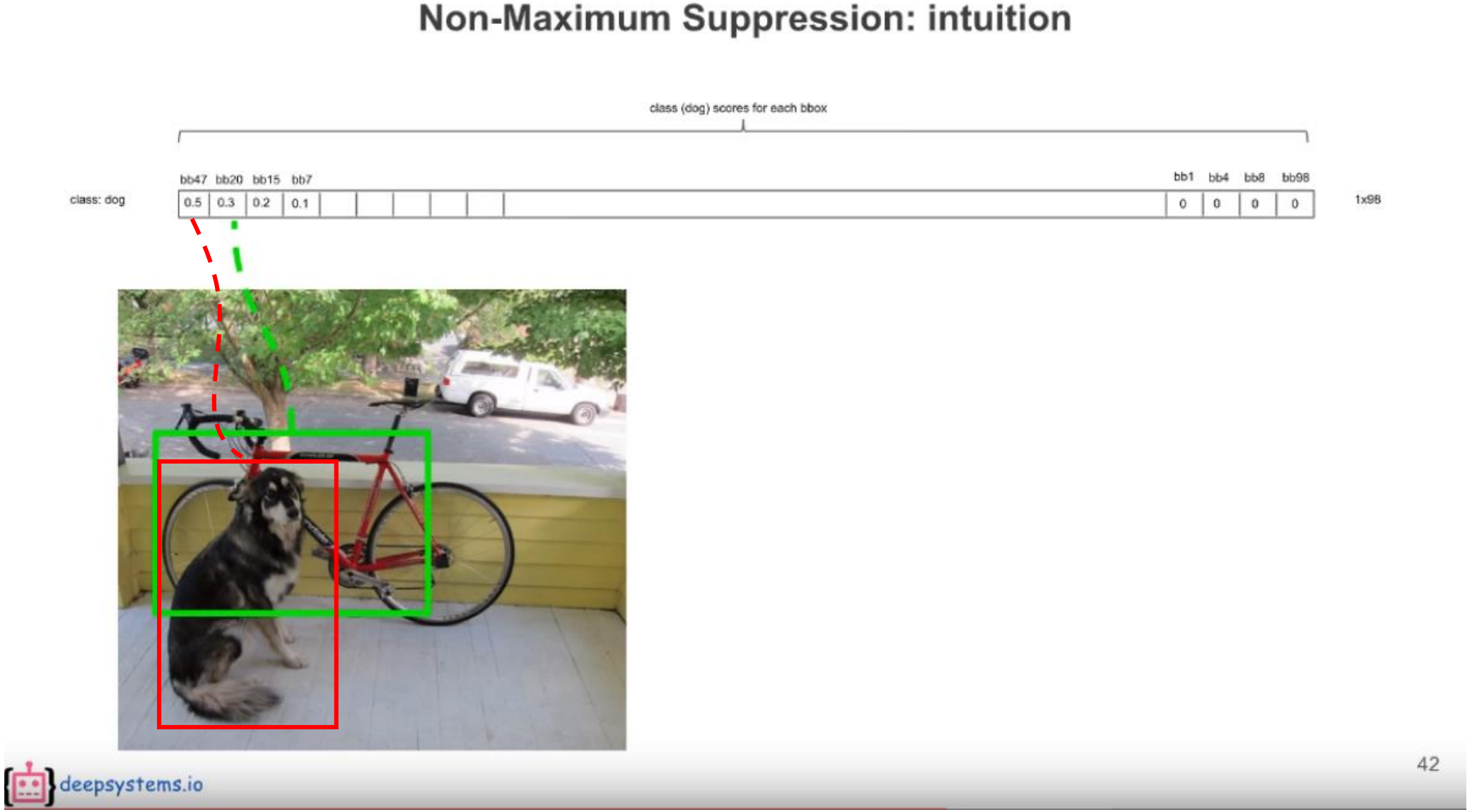
[분석] YOLO, https://curt-park.github.io/2017-03-26/yolo/

You only look once: Unified, real-time object detection



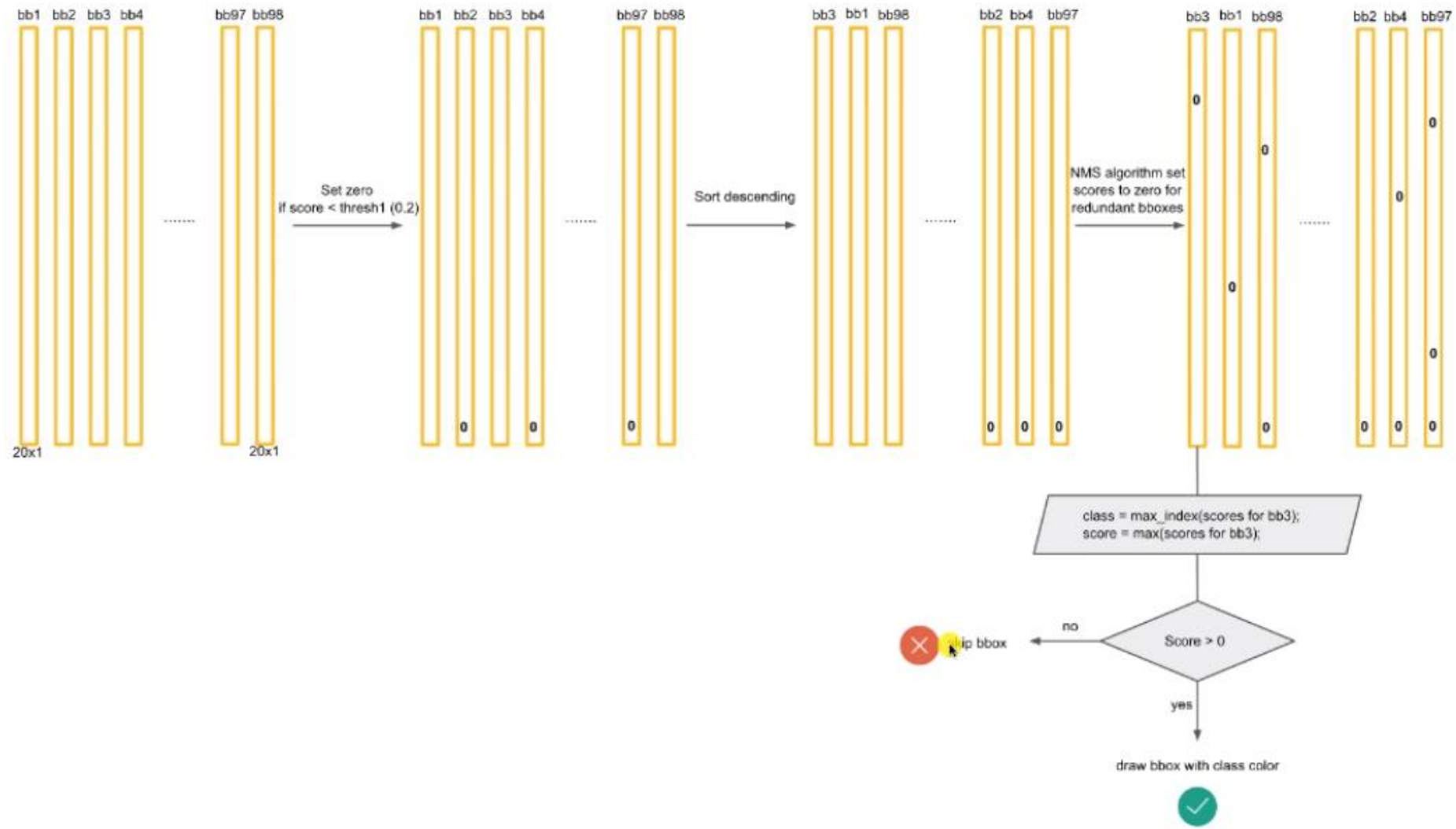
YOLO: You only look once (How it works), https://www.youtube.com/watch?v=L0tzmv--CGY&feature=youtu.be

You only look once: Unified, real-time object detection



YOLO: You only look once (How it works), https://www.youtube.com/watch?v=L0tzmv--CGY&feature=youtu.be

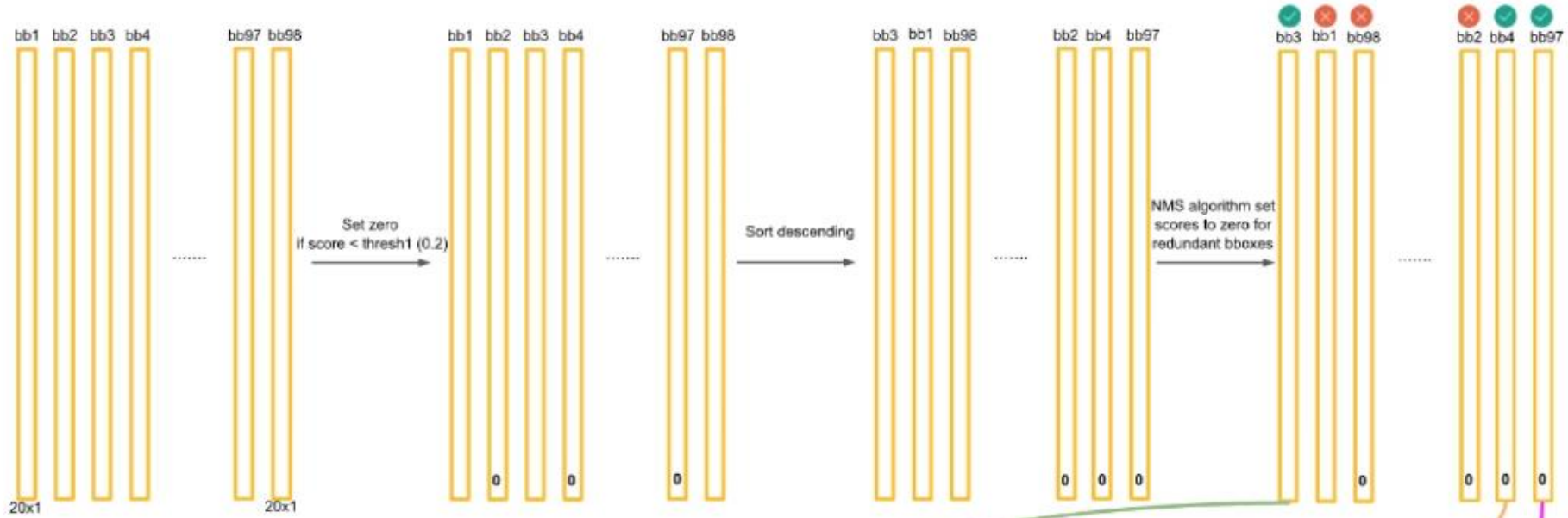
You only look once: Unified, real-time object detection

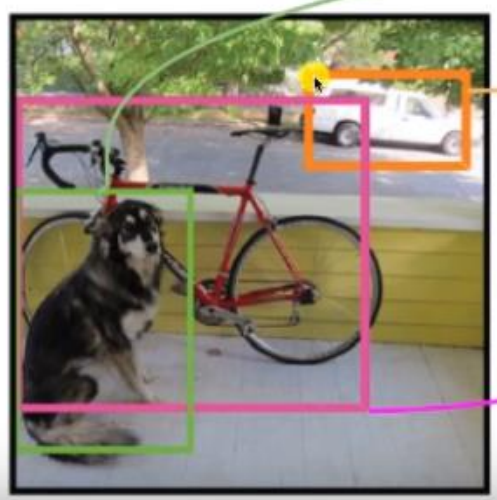


deepsystems.io

YOLO: You only look once (How it works), https://www.youtube.com/watch?v=L0tzmv--CGY&feature=youtu.be

You only look once: Unified, real-time object detection





YOLO: You only look once (How it works), https://www.youtube.com/watch?v=L0tzmv--CGY&feature=youtu.be

You only look once: Unified, real-time object detection

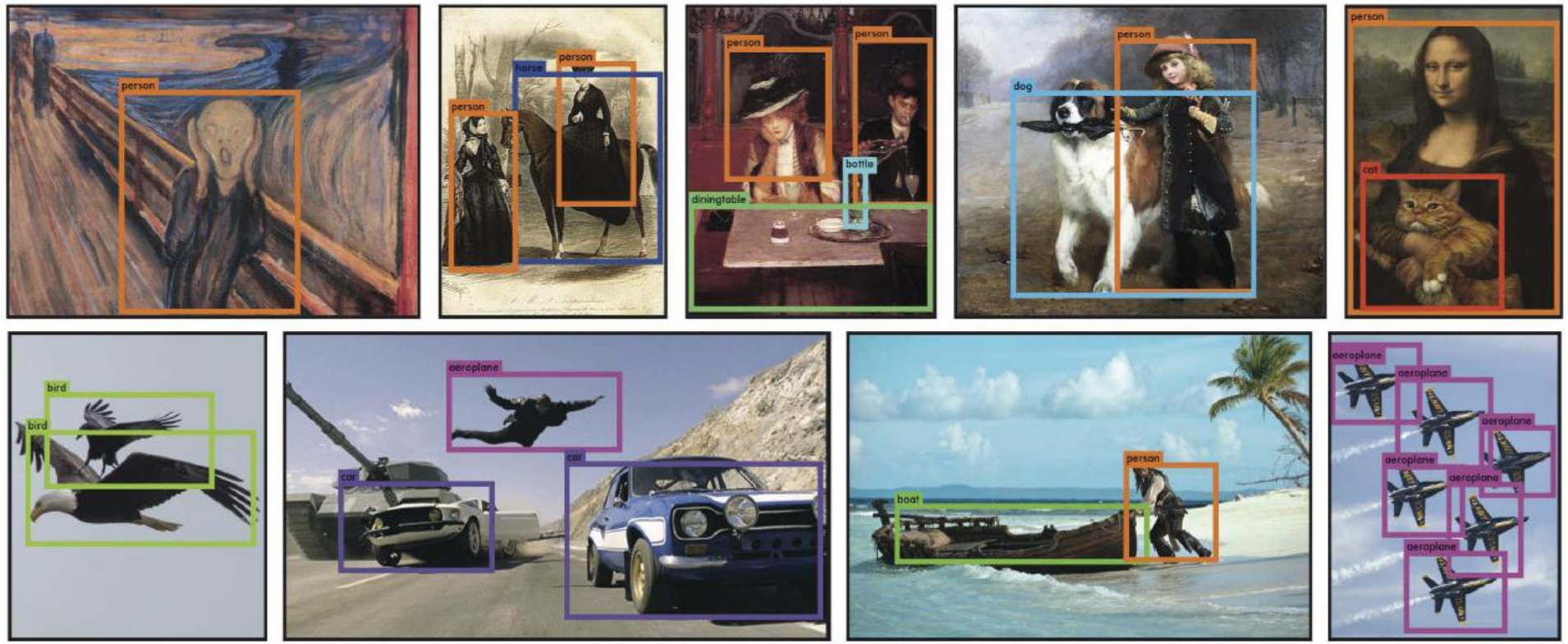


Figure 6: Qualitative Results. YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.

You only look once: Unified, real-time object detection

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	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[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	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.

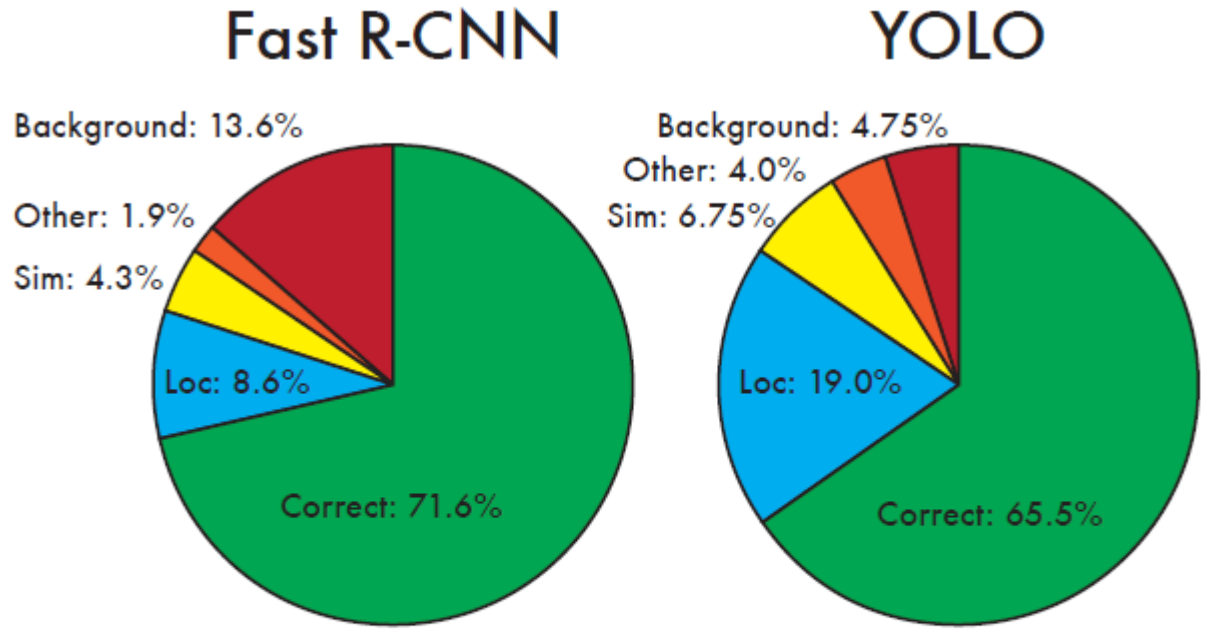


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

You only look once: Unified, real-time object detection

YOLO가 가지는 한계

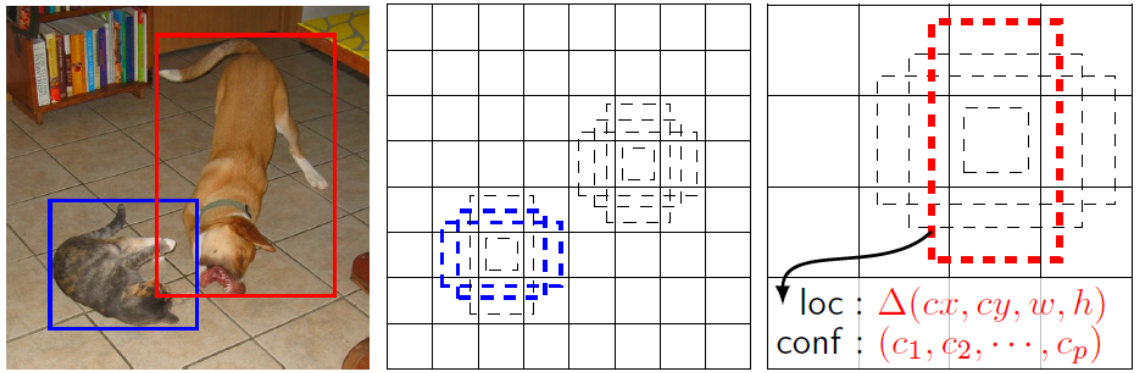
1. 논문에서 제안한 바와 같이 그리드셀을 7X7로 구성하여 Classification을 수행하면, 한개의 그리드 셀에 여러개의 오브젝트가 있는 경우 Classification이 제대로 안된다.

2. Bounding Box의 종횡비와 크기를 학습을 통해 구하므로, 학습하지 못한 새로운 형태의 Bounding Box 형태가 있을때 정확히 예측하기 어렵다.

3. Fast R-CNN계열의 알고리즘보다 Localization의 성능이 약간 떨어진다.
   CNN을 통과한 feature map을 대상으로 bounding box을 예측하기 때문에..

하지만.. 위의 한계점을 상쇄할만한 처리 속도와 성능을 충분히 보여줬다고 생각한다.

Ssd: Single shot multibox detector



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Fig. 1: **SSD framework.** (a) SSD only needs an input image and ground truth boxes for each object during training. In a convolutional fashion, we evaluate a small set (e.g. 4) of default boxes of different aspect ratios at each location in several feature maps with different scales (e.g. 8×8 and 4×4 in (b) and (c)). For each default box, we predict both the shape offsets and the confidences for all object categories ($(c_1, c_2, \cdots, c_p)$). At training time, we first match these default boxes to the ground truth boxes. For example, we have matched two default boxes with the cat and one with the dog, which are treated as positives and the rest as negatives. The model loss is a weighted sum between localization loss (e.g. Smooth L1 [6]) and confidence loss (e.g. Softmax).

논문의 특징 및 기여한 점

1. YOLO의 장점과 Fast R-CNN의 region proposal의 특징을 합쳐서 좋은 결과를 냈다.

   1) Predicting
   Category scores -> YOLO

   Box offset for a fixed set of default boxes
   -> Anchors in Fast R-CNN

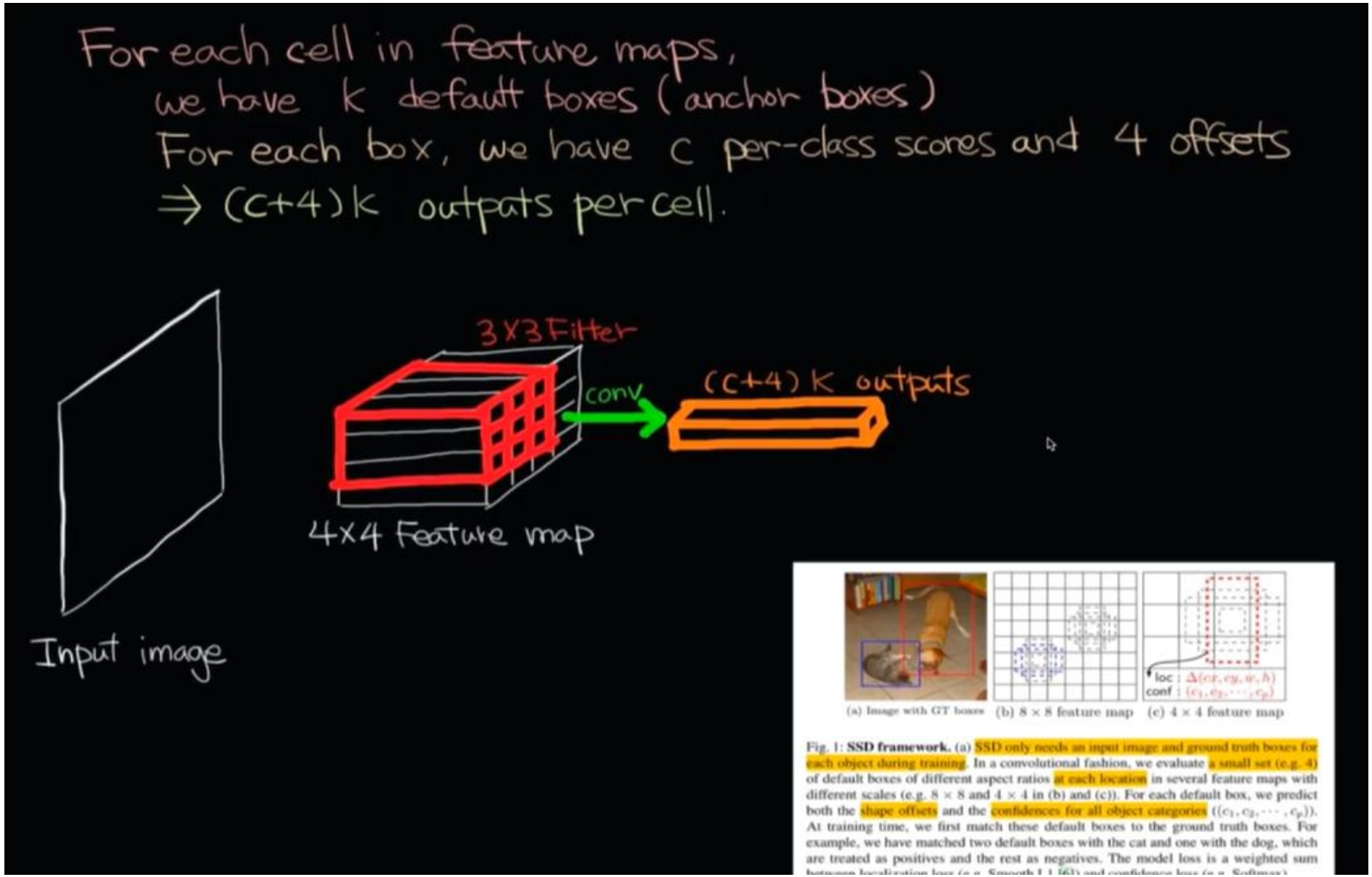
   2) From each cell(YOLO) of multiple(Deconvnet) convolutional feature maps.

2. 최종 레이어와 FC하지 않고, Conv을 이용(YOLO와 다른점)

3. 여러개의 피처맵을 사용함으로써, 다양한 그리드로 접근가능하기 때문에 다양한 크기의 물체를 detection 할 수 있다.
   -> 큰 피처맵에서는 작은 물체를 검출하고, 작은 피처맵에서는 큰 물체의 검출을 담당한다.

4. Bounding Box의 위치를 offset으로 표현(YOLO와 다른점)

Ssd: Single shot multibox detector



논문으로 짚어보는 딥러닝의 맥, Image Detection 방법론: AttentionNet, SSD, YOLO, YOLOv2, https://www.edwith.org/deeplearningchoi/lecture/15579/

Ssd: Single shot multibox detector

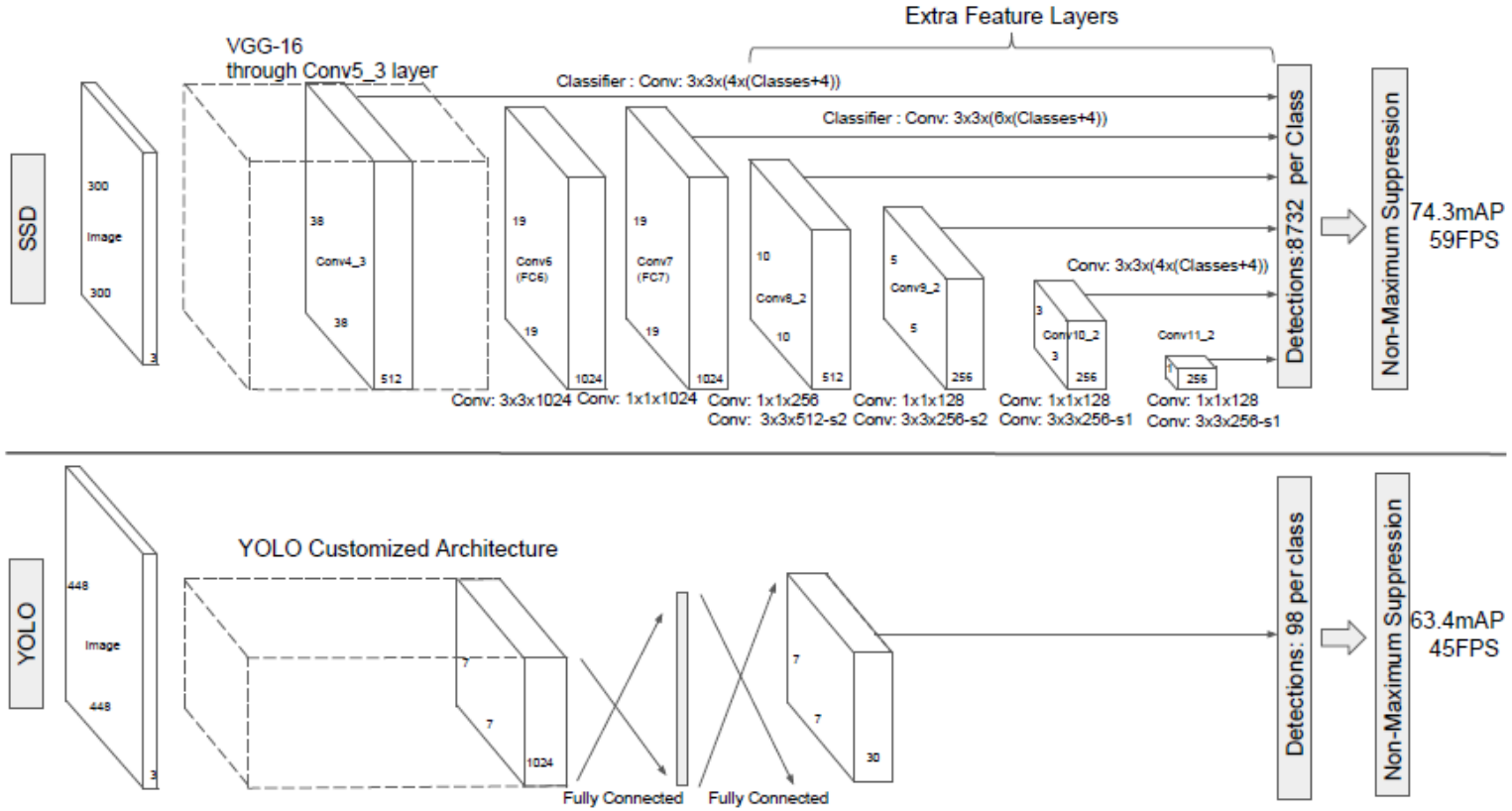
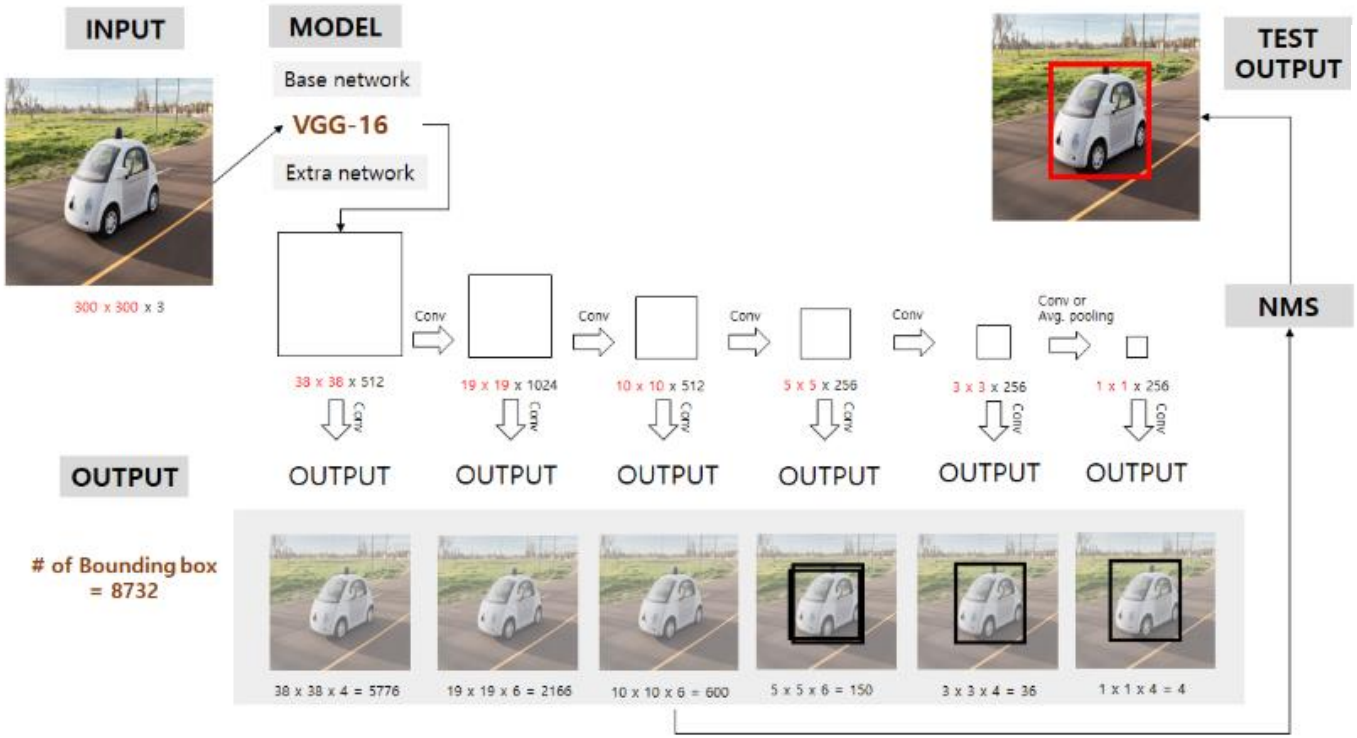


Fig. 2: A comparison between two single shot detection models: SSD and YOLO [5]. Our SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. SSD with a 300×300 input size significantly outperforms its 448×448 YOLO counterpart in accuracy on VOC2007 `test` while also improving the speed.

Ssd: Single shot multibox detector



Convolutional predictiors for detection

- 이미지부터 최종 피쳐맵까지는 Conv(3x3, s=2)로 연결
- Output과 연결된 피쳐맵은 3 x 3 x p 사이즈의 필터로 컨볼루션 연산. (Yolo v1은 Output과 Fully-Connected. 여기서 시간을 많이 단축시킴)
- 예측된 Output은 class, category 점수와, default box에 대응되는 offset을 구함

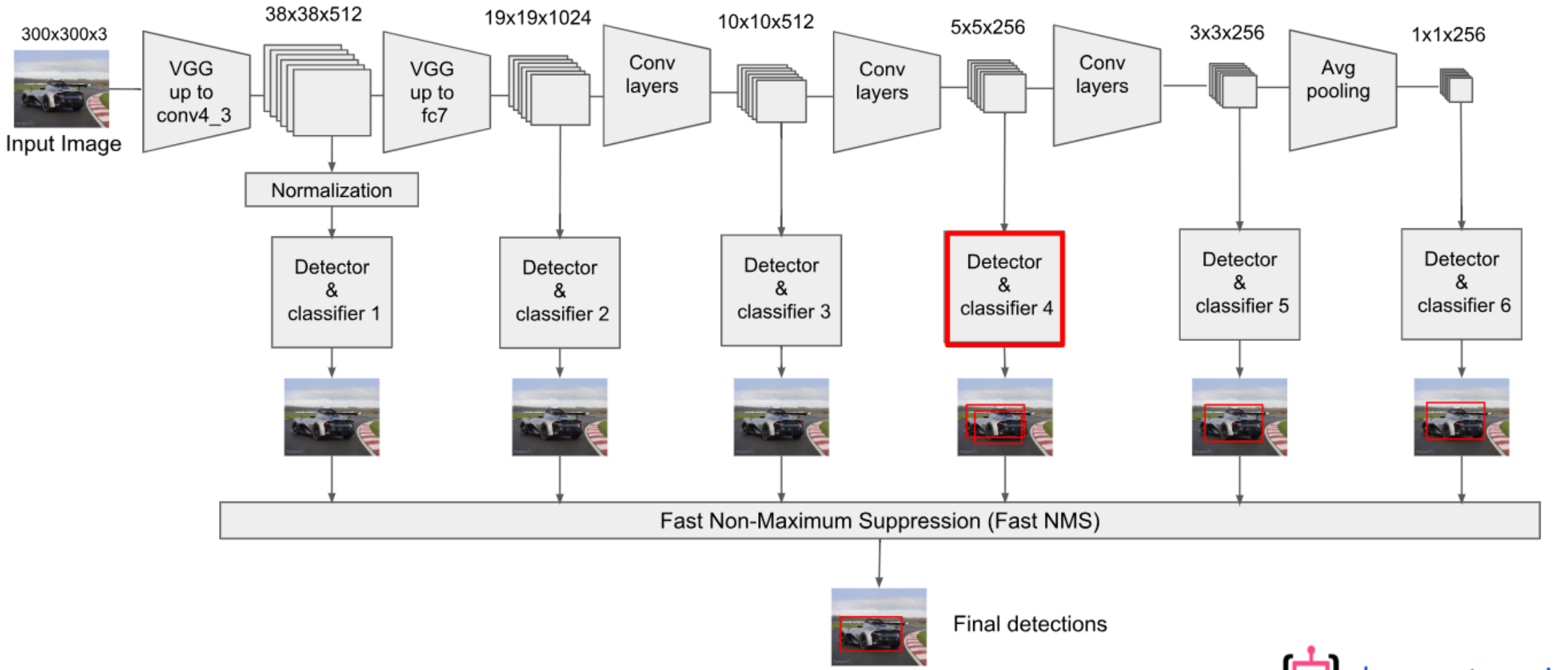
Multi-scale feature maps for detection

- 38x38, 19x19, 10x10, 5x5, 3x3, 1x1 의 피쳐맵들을 의미
- Yolo는 7x7 grid 하나뿐이지만 SSD는 전체 이미지를 38x38, 19x19, 10x10, 5x5, 3x3, 1x1의 그리드로 나누고 output과 연결
- 큰 피쳐맵에서는 작은 물체 탐지, 작은 피쳐맵에서는 큰 물체 탐지 (뒤의 2.2 training 부분에서 더 자세히 다룸)

[논문] SSD: Single Shot Multibox Detector 분석, https://taeu.github.io/paper/deeplearning-paper-ssd/

Ssd: Single shot multibox detector

Default box 가 3인 경우

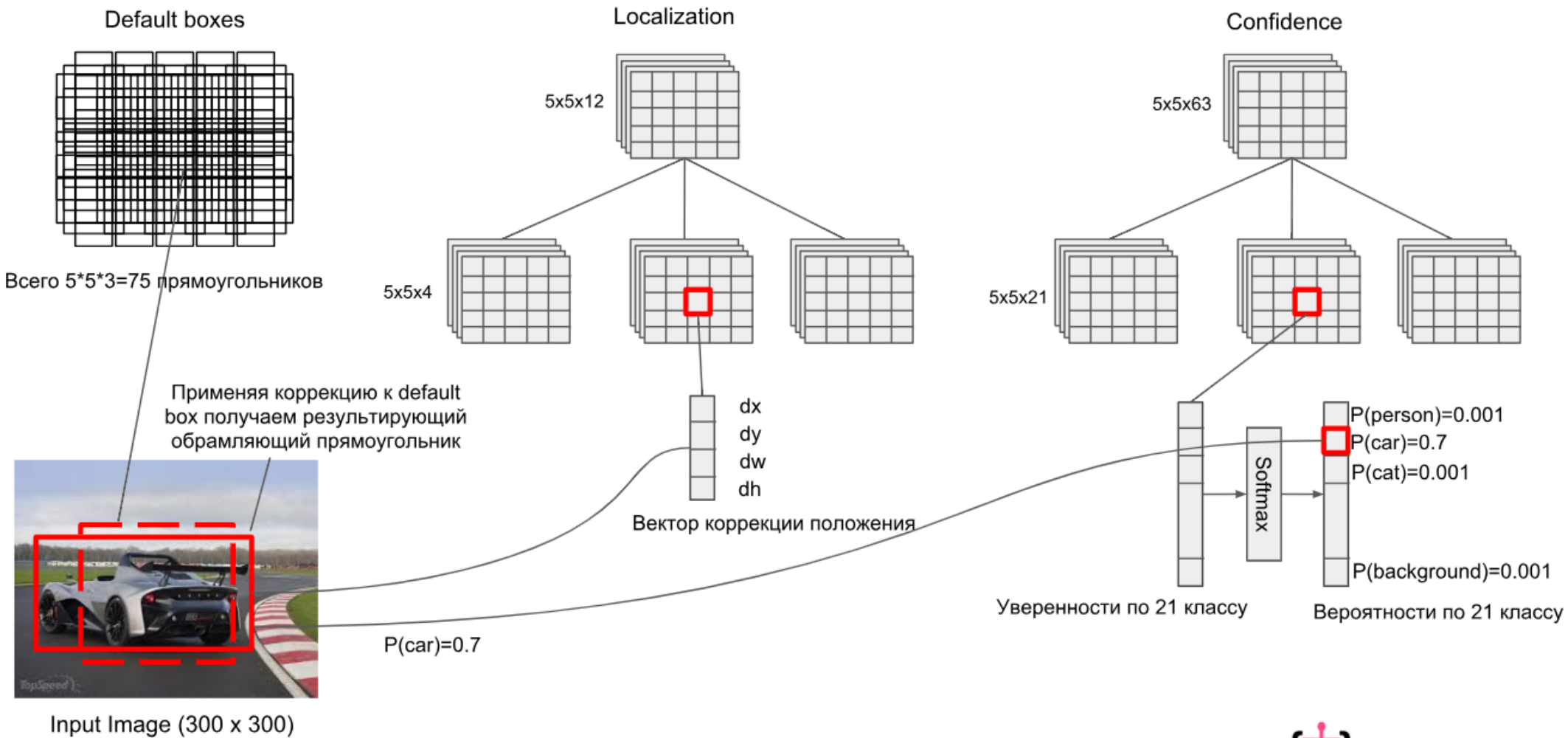


deepsystems.io

[SSD: Single Shot MultiBox Detector], https://www.youtube.com/watch?v=Gc233mo6r9c

Ssd: Single shot multibox detector

Default box 가 3인 경우



deepsystems.io

[SSD: Single Shot MultiBox Detector], https://www.youtube.com/watch?v=Gc233mo6r9c

Ssd: Single shot multibox detector

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

Class Score　　Bounding box Offset

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^{k} \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} x_{ij}^p log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

용어정리

- x^p_ij = {1,0} i번째 default box와 j번째 ground truth 박스의 category p에 물체 인식 지표. p라는 물체의 j번째 ground truth와 i번째 default box 간의 IOU 가 0.5 이상이면 1 아니면 0.
- N 은 # of matched default boxes
- l 은 predicted box (예측된 상자)
- g 는 ground truth box
- d 는 default box.
- cx, cy는 그 박스의 x, y좌표
- w, h는 그 박스의 width, heigth
- 알파는 1 (교차 검증으로부터 얻어진)
- loss fucntion 은 크게 2부분, 클래스 점수에 대한 loss와 바운딩 박스의 offset에 대한 loss로 나뉜다.

[논문] SSD: Single Shot Multibox Detector 분석, https://taeu.github.io/paper/deeplearning-paper-ssd/

Ssd: Single shot multibox detector

$$L_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^{k} \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

- 우리가 예측해야할 predicted box의 l^m_i(cx,cy,w,h)값들은 특이한 g햇 값들을 예측
- 이때 g햇의 cx,cy는 default box의 cx와 w,h로 normalize(?)된 것을 볼 수 있다.
- 이미 IOU가 0.5 이상만 된 것 부분에서 고려하므로, 상대적으로 크지 않은 값들을 예측해야하고 더불어 이미 0.5 이상 고려된 부분에서 출발하므로 비교적 빨리 수렴할 수 있을 것 같다.(이 부분은 주관적인 판단)
- 초기값은 default box에서 시작하지 않을까 싶음
- g햇의 w, h도 마찬가지
- 예측된 l 값들을 box를 표현할때(마지막 Test Output) 역시 default box의 offset 정보가 필요함.

[논문] SSD: Single Shot Multibox Detector 분석, https://taeu.github.io/paper/deeplearning-paper-ssd/

Ssd: Single shot multibox detector

$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

- positive(매칭 된) class에 대해서는 softmax
- negative(매칭 되지 않은, 배경) class를예측하는 값으 c햇^0_i 값이고 별다른 언급은 없지만 background이면 1, 아니면 0의 값을 가져야 함
- 최종적인 predicted class scores는 우리가 예측할 class + 배경 class 를 나타내는지표

[논문] SSD: Single Shot Multibox Detector 분석, https://taeu.github.io/paper/deeplearning-paper-ssd/

Ssd: Single shot multibox detector

Choosing scales ansd aspect ratios for default boxes

- default box를 위한 scale. 여래 크기의 default box 생성을 위해 다음과 같은 식 만듦.

$$s_k = s_{\min} + \frac{s_{\max} - s_{\min}}{m - 1}(k - 1), \quad k \in [1, m]$$

$$a_r \in \{1, 2, 3, \tfrac{1}{2}, \tfrac{1}{3}\}$$

$$(w_k^a = s_k\sqrt{a_r})$$

$$(h_k^a = s_k/\sqrt{a_r})$$

$$(\tfrac{i+0.5}{|f_k|}, \tfrac{j+0.5}{|f_k|})$$ $|f_k|$ is the size of the k-th square feature map, $i, j \in [0, |f_k|)$

- Smin = 0.2, Smax = 0.9
- 저 식에다 넣으면 각 feature map당 서로 다른 6개의 s 값들(scale 값들)이 나옴
- 여기에 aspect ratio = {1,2,3,1/2,1/3} 설정
- default box의 width는 s_k x 루트(a_r)
- a_r = 1 일경우 s'k = *루트(s_k x s*(k+1))
- default box의 cx,cy는 k 번째 피쳐맵 크기를 나눠 사용
- 근데 굳이 이렇게 스케일을 저렇게 나눈 이유는 잘 모르겠음.. 대략 예측되는 상자가 정사각형이나, 가로로 조금 길쭉한 상자 세로로 조금 길쭉한 상자이니 2,3으로 임의로 정해도 잘 학습이 될테지만, 특이한 경우, 예를들어 가로 방향으로 걸어가는 지네같은 경우 저 비율로하면 0.5 threshold로 지정했을때 학습안됨. 학습할 이미지에 따라 aspect ratio를 조정해야할 필요가 있을텐데, 임의로 정한다면 비효율적. knn 같은 알고리즘을 활용하면 더 좋을 것 같음.

[논문] SSD: Single Shot Multibox Detector 분석, https://taeu.github.io/paper/deeplearning-paper-ssd/

Ssd: Single shot multibox detector

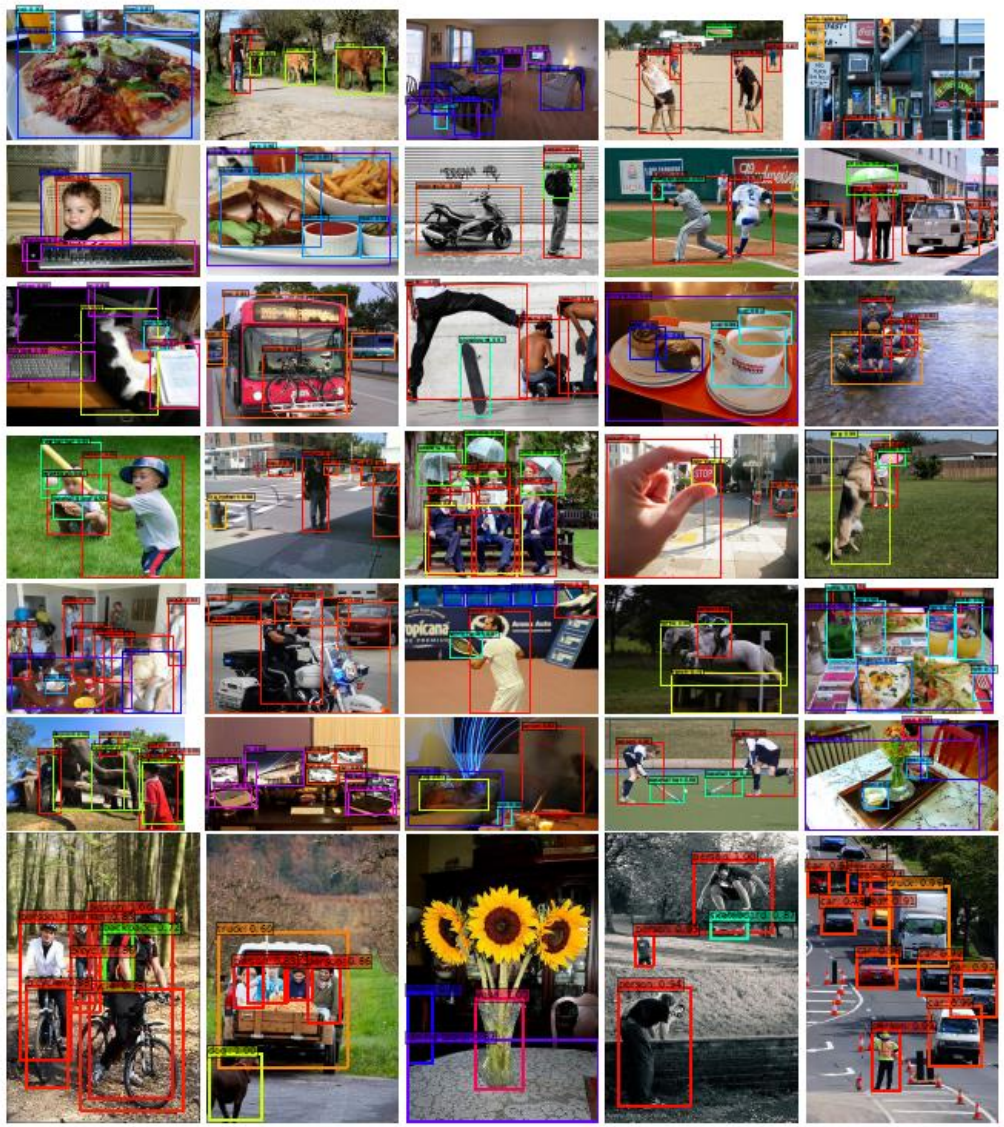


Fig. 5: **Detection examples on COCO `test-dev` with SSD512 model.** We show detections with scores higher than 0.6. Each color corresponds to an object category.

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast[6]	07++12	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
Faster[2]	07++12	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
Faster[2]	07++12+COCO	75.9	87.4	83.6	76.8	62.9	59.6	81.9	82.0	91.3	54.9	82.6	59.0	89.0	85.5	84.7	84.1	52.2	78.9	65.5	85.4	70.2
YOLO[5]	07++12	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
SSD300	07++12	72.4	85.6	80.1	70.5	57.6	46.2	79.4	76.1	89.2	53.0	77.0	60.8	87.0	83.1	82.3	79.4	45.9	75.9	69.5	81.9	67.5
SSD300	07++12+COCO	77.5	90.2	83.3	76.3	63.0	53.6	83.8	82.8	92.0	59.7	82.7	63.5	89.3	87.6	85.9	84.3	52.6	82.5	**74.1**	**88.4**	74.2
SSD512	07++12	74.9	87.4	82.3	75.8	59.0	52.6	81.7	81.5	90.0	55.4	79.0	59.8	88.4	84.3	84.7	83.3	50.2	78.0	66.3	86.3	72.0
SSD512	07++12+COCO	**80.0**	**90.7**	**86.8**	**80.5**	**67.8**	**60.8**	**86.3**	**85.5**	**93.5**	**63.2**	**85.7**	**64.4**	**90.9**	**89.0**	**88.9**	**86.8**	**57.2**	**85.1**	72.8	**88.4**	**75.9**

Table 4: **PASCAL VOC2012 `test` detection results.** Fast and Faster R-CNN use images with minimum dimension 600, while the image size for YOLO is 448×448. data: "07++12": union of VOC2007 `trainval` and `test` and VOC2012 `trainval`. "07++12+COCO": first train on COCO `trainval35k` then fine-tune on 07++12.

SSD가 가지는 한계

1. 여러개의 피처 맵의 검출을 전부 계산하기 때문에 YOLO에 비해 계산량이 많다.
2. Default Box와 박스의 종횡비는 이론적인 부분이 약하다. YOLO와 같이 특정 종횡비를 같는 물체의 검출에 취약할 수 밖에 없다.

   (이 부분은 여전히 Object Detection 문제에 중요한 과제로 남아있다.)

YOLO9000: better, faster, stronger

- Better
  - Batch normalization
  - High resolution classifier
  - Convolution with anchor boxes
  - Dimension clusters
  - Direct location prediction
  - Fine-grained features
  - Multi-scale training
- Faster
  - Darknet-19
  - Training for classification
  - Training for detection
- Stronger
  - Hierarchical classification
  - Dataset combination with Word-tree
  - Joint classification and detection

YOLOv2

YOLO9000

PR-023: YOLO9000: Better, Faster, Stronger, https://www.youtube.com/watch?v=6fdcISGgeio

YOLO9000: better, faster, stronger

	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**

Table 2: The path from YOLO to YOLOv2. Most of the listed design decisions lead to significant increases in mAP. Two exceptions are switching to a fully convolutional network with anchor boxes and using the new network. Switching to the anchor box style approach increased recall without changing mAP while using the new network cut computation by 33%.

YOLO9000: better, faster, stronger

Introduction & Motivation

- Detection frameworks have become increasingly fast and accurate → However, most detection methods are still constrained to a small set of objects.
- We would like detection to scale to level of object classification → However, labelling images for detection is far more expensive than labelling for classification.
- Maybe we cannot see detection datasets on the same scale as classification datasets in the near future.

PR-023: YOLO9000: Better, Faster, Stronger, https://www.youtube.com/watch?v=6fdcISGgeio

YOLO9000: better, faster, stronger

Introduction

- Authors propose a new method to expand the scope of current detection systems by using classification dataset we already have.
- Joint training algorithm is also proposed that allows to train object detectors on both detection and classification data
- First, improving upon the base YOLO detection system to produce YOLOv2 (SOTA real-time detector)
- Then, using dataset combination method and joint training algorithm to train model on more than 9000 classes!

PR-023: YOLO9000: Better, Faster, Stronger, https://www.youtube.com/watch?v=6fdcISGgeio

YOLO9000: better, faster, stronger

To Improve YOLO

- YOLO's shortcomings relative to SOTA detection systems
  - YOLO makes a significant number of localization errors
  - YOLO has relatively low recall compared to region proposal based method
- Focus mainly on improving recall and localization while maintain classification accuracy.
- It is still very important to detect FAST!

PR-023: YOLO9000: Better, Faster, Stronger, https://www.youtube.com/watch?v=6fdcISGgeio

YOLO9000: better, faster, stronger

1-1. Batch normalization. 모든 컨볼루션 네트워크에 배치 정규화 추가로 2%p mAP 향상.

1-2. High resolution classifier. 이미지넷 데이터로 앞단의 네트워크를 먼저 학습시키는데, 이때 448x448의 해상도로 10 epoch 동안 fine tuning함. 그리고 나머지 부분들을 디텍션하면서 tuning 시킨다. 고해상도로 학습된 앞단의 classification 네트워크 덕분에 4%p mAP 향상.

1-3. Convolutional with Anchor boxes. YOLO v1에서 사실 의문이기도 했던 detection을 위한 output을 연결시키는 마지막 레이어에서 Fully Connect 되었었는데, 이 fully connect를 제거하고 convolution으로 바꾸었다. 동시에 Anchor box도 적용했다고 한다. mAP는 0.3% p 만큼 줄어들었지만 recall이 81%에서 88%로 증가했다. FC를 convolutional connect로 바꿨으니 당연히 computation이 33%p 떨어졌다고한다.

> *사실 여기서 두개를 동시에 적용시켰기 때문에 어느 부분이 얼만큼의 효과를 내었는지는 모른다. 대략 FC -> convolutional는 computation을 많이 감소시켰을 거고, anchor box를 써서 recall 이 증가했지만 computation은 늘어났을거라 추측되지만 anchor box를 얼마나 잘 적용시켰는지는 나와있지 않으므로 물음표. 뭐 크게 중요하지 않을 수도 있는게 뒤에서 이 anchor box들을 잘 적용시키기위한 2가지 전략이 나오므로 뒤에서 더 살펴보도록 하자.*

[논문] YOLO9000: Better, Faster, Stronger 분석, https://taeu.github.io/paper/deeplearning-paper-yolov2/

YOLO9000: better, faster, stronger

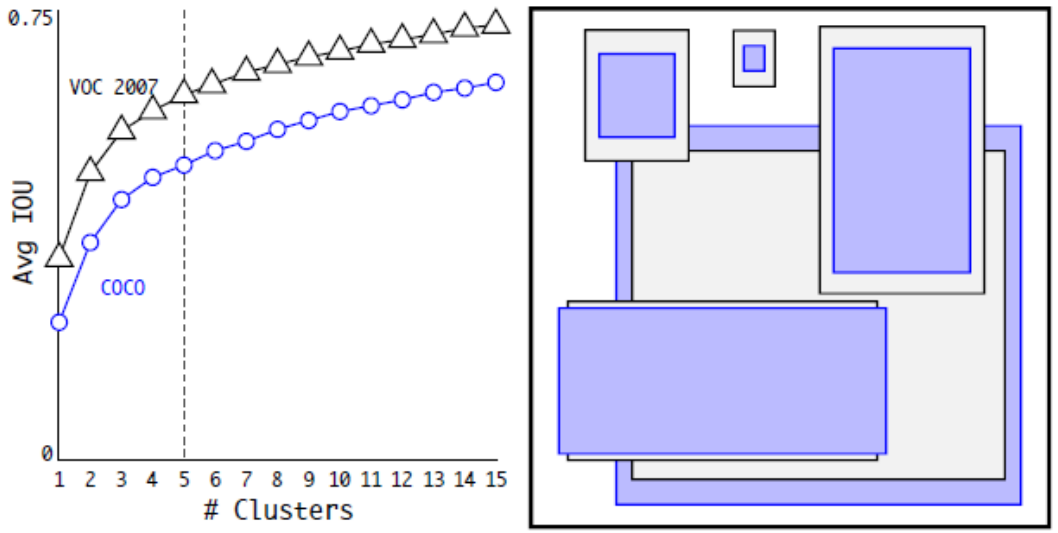


Figure 2: Clustering box dimensions on VOC and COCO. We run k-means clustering on the dimensions of bounding boxes to get good priors for our model. The left image shows the average IOU we get with various choices for k. $k = 5$ gives a good tradeoff for recall vs. complexity of the model. The right image shows the relative centroids for VOC and COCO. COCO has greater variation in size than VOC.

SSD나 Fast R-CNN의 경우 사람이 임의로 선정한 anchor box의 도움을 받아 predicted box를 구한다. 하지만 임의라는 말 역시 데이터의 특성에 따라 성능이 달라질 수 있고, 실제 실험할 때도 어느 정도의 anchor box의 비율을 두어야할지, 몇 개를 두어야 할지 애매해질 수 있다. YOLO v2에서는 이런 고민을 해결하기 위해 detection dataset에서 k-means 알고리즘을 활용해 이 하이퍼파라미터들을 설정했다. 이때 기존의 k-means에서 유클리디안 거리를 쓰지 않고 `d(box, centroid) = 1 - IOU(box,centroid)` 사용했다.

Box Generation	#	Avg IOU
Cluster SSE	5	58.7
Cluster IOU	5	61.0
Anchor Boxes [15]	9	60.9
Cluster IOU	9	67.2

Table 1: Average IOU of boxes to closest priors on VOC 2007. The average IOU of objects on VOC 2007 to their closest, unmodified prior using different generation methods. Clustering gives much better results than using hand-picked priors.

9개의 앵커박스를 두었을때 그만큼 다양한 anchor box로 잘 예측할 수 있지만, 5개를 두었을 때랑 크게 차이가 나지 않는다고 판단하고(computation은 2배가 되는데에 비해) YOLO v2에서는 5개의 anchor box를 사용하기로 결정.

[논문] YOLO9000: Better, Faster, Stronger 분석, https://taeu.github.io/paper/deeplearning-paper-yolov2/

YOLO9000: better, faster, stronger

1-5. Direct location prediction.

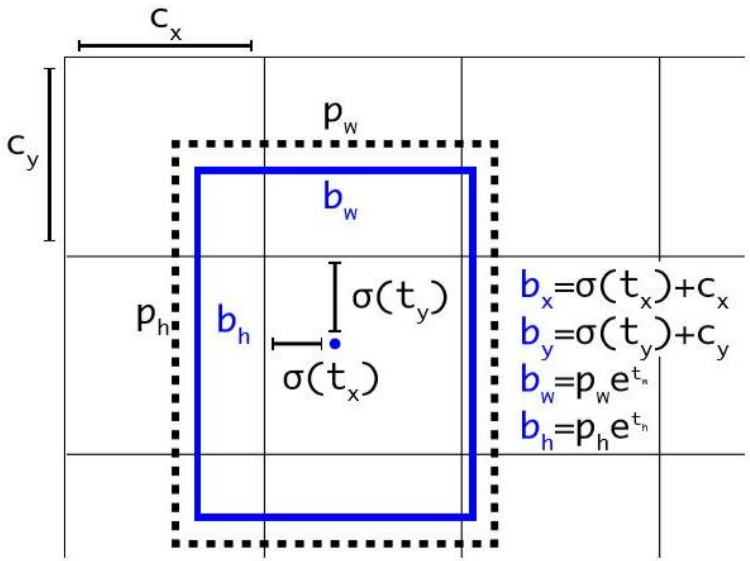


Figure 3: Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function.

The network predicts 5 bounding boxes at each cell in the output feature map. The network predicts 5 coordinates for each bounding box, t_x, t_y, t_w, t_h, and t_o. If the cell is offset from the top left corner of the image by (c_x, c_y) and the bounding box prior has width and height p_w, p_h, then the predictions correspond to:

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h} \\
Pr(\text{object}) * IOU(b, \text{object}) &= \sigma(t_o)
\end{aligned}
$$

- cx, cy는 그리드 셀의 좌상든 끝 offset
- pw, ph는 prior(우선순위 앵커박스)의 width, height
- tx, ty, tw, th가 우리가 예측해야할 값들
- bx, by, bw, bh는 각 값들을 조정하여 실제 GT와 IOU를 계산할 최종 bounding box의 offset 값들

이 부분은 SSD나 Fast R-CNN의 Loss fucntion 부분을 떠올리면 이해가 더 쉬울 것이다. anchor box를 예측하되 기존의 Predicted box의 중심좌표들이 초기 단계에 잘못 설정된다면 학습이 잘 안될 수 있다는 점을(예측해야되는 곳보다 멀어질 수 있음) 설명하고 있다. (SSD의 경우는 l, predict된 값이 default box로 normalize(?)된 값이라 상대적으로 잘 될 것) YOLO v2는 이런 값들이 범위를 크게 벗어나지 않고 적절히 학습시키게 하기 위해서 다음과 같은 설정을 한다.

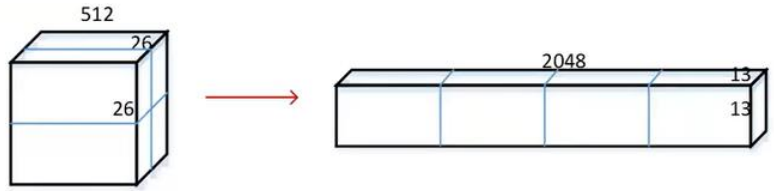
중심좌표는 시그모이드를 통해서 0으로 initialize되면 중심에 가게, 너비과 높이 역시 0으로 초기화 했을때 prior, anchor box의 값들에서 시작될 수 있게 한 조치이다. 이렇게 하면 안정적으로 학습이 잘 될 것.

최종적으로 1-4, 1-5번을 통해서 약 5%p mAP 향상.

[논문] YOLO9000: Better, Faster, Stronger 분석, https://taeu.github.io/paper/deeplearning-paper-yolov2/

YOLO9000: better, faster, stronger

Better



- Fine-Grained Features
  - Modified YOLO uses 13x13 feature maps to predict detections
    - Good at large object, finer grained features are required for small objects
  - Simply adding a passthrough layer that brings features from an earlier layer at 26x26 resolution
  - Turn 26x26x512 feature map into a 13x13x2048 feature map which can be concatenated with original features
  - Detector runs on top of this expanded features
  - 1% performance increase

PR-023: YOLO9000: Better, Faster, Stronger, https://www.youtube.com/watch?v=6fdcISGgeio

1-6. Fine-grained features. 이 부분 역시 SSD의 multi-scale feature map과 엮어 생각하면 좋은데, 13x13은 큰 이미지를 검출하기엔 충분한 feature map이지만, 작은 물체를 detect하기에는 약간 충분하지 않을 수 있다. 따라서 26x26 layer 에서 그다음 conv하지 않고 26x26x512의 특징맵을 13x13x(512x4)로 변환한다음(26x26에서 중심을 기준으로 2x2로 나눈 네 조각을 concatenate) detection을 위한 output으로 이어준다. 여기서 1%의 성능향상이 있음.

1-7. Multi-scale training. 기본 사이즈는 416x416을 이용한다.(마지막 부분이 홀수 grid를 갖게 하기 위해서) 거기에 추가로 한 네트워크로 다양한 해상도의 이미지도 잘 처리하게 만들기 위해 다양한 해상도를 골고루 학습시킨다. 10번의 배치마다 {320,352, … , 608}로 resize된다.

[논문] YOLO9000: Better, Faster, Stronger 분석, https://taeu.github.io/paper/deeplearning-paper-yolov2/

YOLO9000: better, faster, stronger

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.

2. Faster

2-1. Darknet. 많은 Image Detection Model에서 classifier netowrk(앞단의 네트워크)로 VGG Net을 많이 쓴다. 하지만 VGG-16 은 30.69 십억의 floating point 계산을 필요로 한다.(224x224 해상도 imgae의 경우) YOLO v2에서는 GoogleNet을 기반으로한 독자적인 Darkent을 만들어 30십억의 계산량을 8십억으로 줄였다. (accuracy는 88%로 VGG-16의 90% 성능과 크게 차이나지 않는다.)

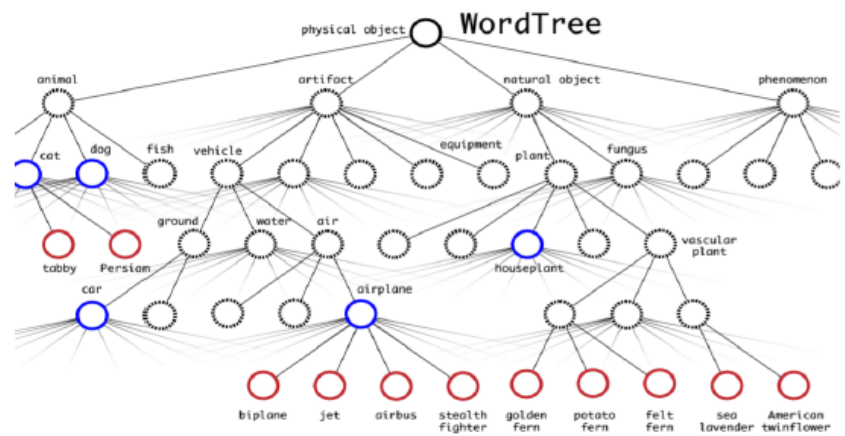
2-2. Training for classification. ImageNet 1000개 클래스 구별 데이터셋을 160 epoch동안 학습하면서 learning rate = 0.1, polynomia l rate decay = a power of 4(4로 나눈다는 뜻일듯), weight decay = 0.0005, momnetum = 0.9, 처음 튜닝은 224로 하다가 중간에 448로 fi ne tune.

2-3. Traininig for detection. 5 bounding box로 5개의 좌표(Confidence score + coordinate)와 20개의 class 점수를 예측하므로 한 그리드 셀에서는 총 5x(5+20) = 125개의 예측값을 가지게 된다. 또 중간에 passthrough layer로부터(26x26) concatenate된 예측값도 포함. 1 60 epoch동안 10^(-3)에서 시작하여 10, 60, 90 에폭마다 decay하고, weight decay = 0.0005, momentum = 0.9를 사용했다. data augm entation 역시 random crops, color shifting등을 이용했다.

[논문] YOLO9000: Better, Faster, Stronger 분석, https://taeu.github.io/paper/deeplearning-paper-yolov2/

Stronger

- Hierarchical classification
  - ImageNet labels are pulled from WordNet, a language database that structures concepts and how they relate
    - "Norfolk terrier" and "Yorkshire terrier" are both hyponyms of "terrier" which is a type of "hunting dog", which is a type of "dog", which is a "canine", etc.
  - WordNet is structured as a directed graph, not a tree, because language is complex
    - To make a tree not a graph, If a concept has tow paths to the root, choose the shorter path
  - Root note is a "Physical Object"



PR-023: YOLO9000: Better, Faster, Stronger, https://www.youtube.com/watch?v=6fdclSGgeio

Stronger

- Hierarchical classification
  - To perform classification with WordTree, predicting conditional probabilities at every node for the probability of each hyponym of that synset given that synset

$$Pr(\text{Norfolk terrier}|\text{terrier})$$
$$Pr(\text{Yorkshire terrier}|\text{terrier})$$
$$Pr(\text{Bedlington terrier}|\text{terrier})$$
$$\ldots$$

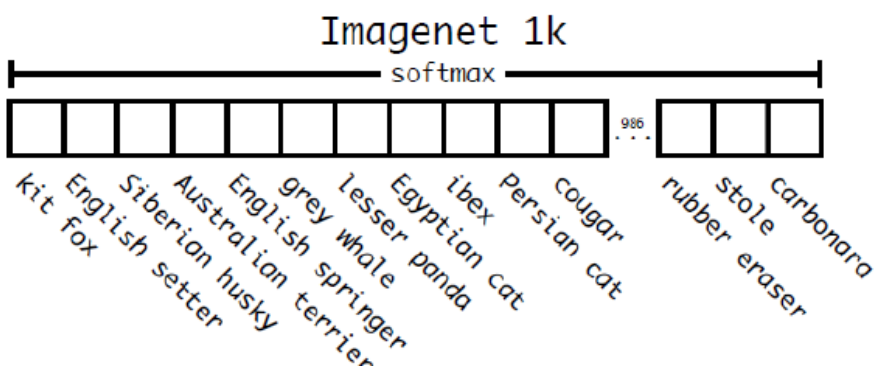
  - Then if a picture of a Norfolk terrier is encountered, its probability is calculated as below

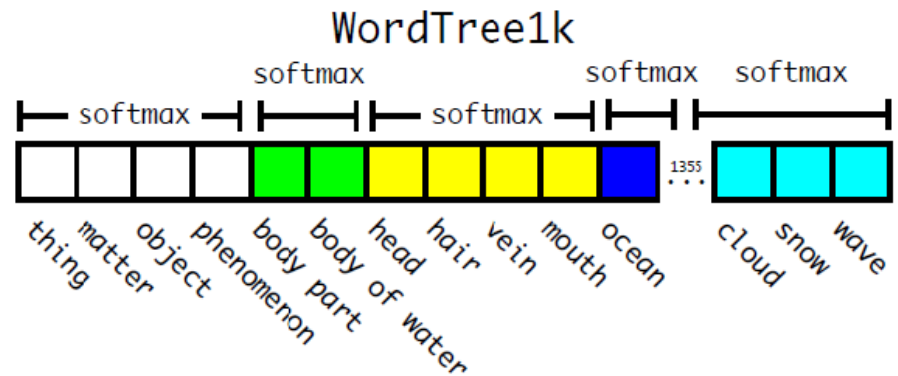
$$Pr(\text{Norfolk terrier}) = Pr(\text{Norfolk terrier}|\text{terrier}) * Pr(\text{terrier}|\text{hunting dog}) * \ldots * Pr(\text{mammal}|Pr(\text{animal}) * Pr(\text{animal}|\text{physical object})$$

PR-023: YOLO9000: Better, Faster, Stronger, https://www.youtube.com/watch?v=6fdclSGgeio

Stronger

- Hierarchical classification
  - To build WordTree1k, adding in all of the intermediate nodes which expands the label space from 1000 to 1369
  - During training, ground truth labels are propagated up
    - If an image is labelled as a "Norfolk terrier" it also gets labelled as a "dog" and a "mammal", etc.
  - To compute the conditional probabilities the model predicts a vector of 1369 values and computes the softmax over all synsets that are hyponyms of the same concept → 90.4% top-5 accuracy

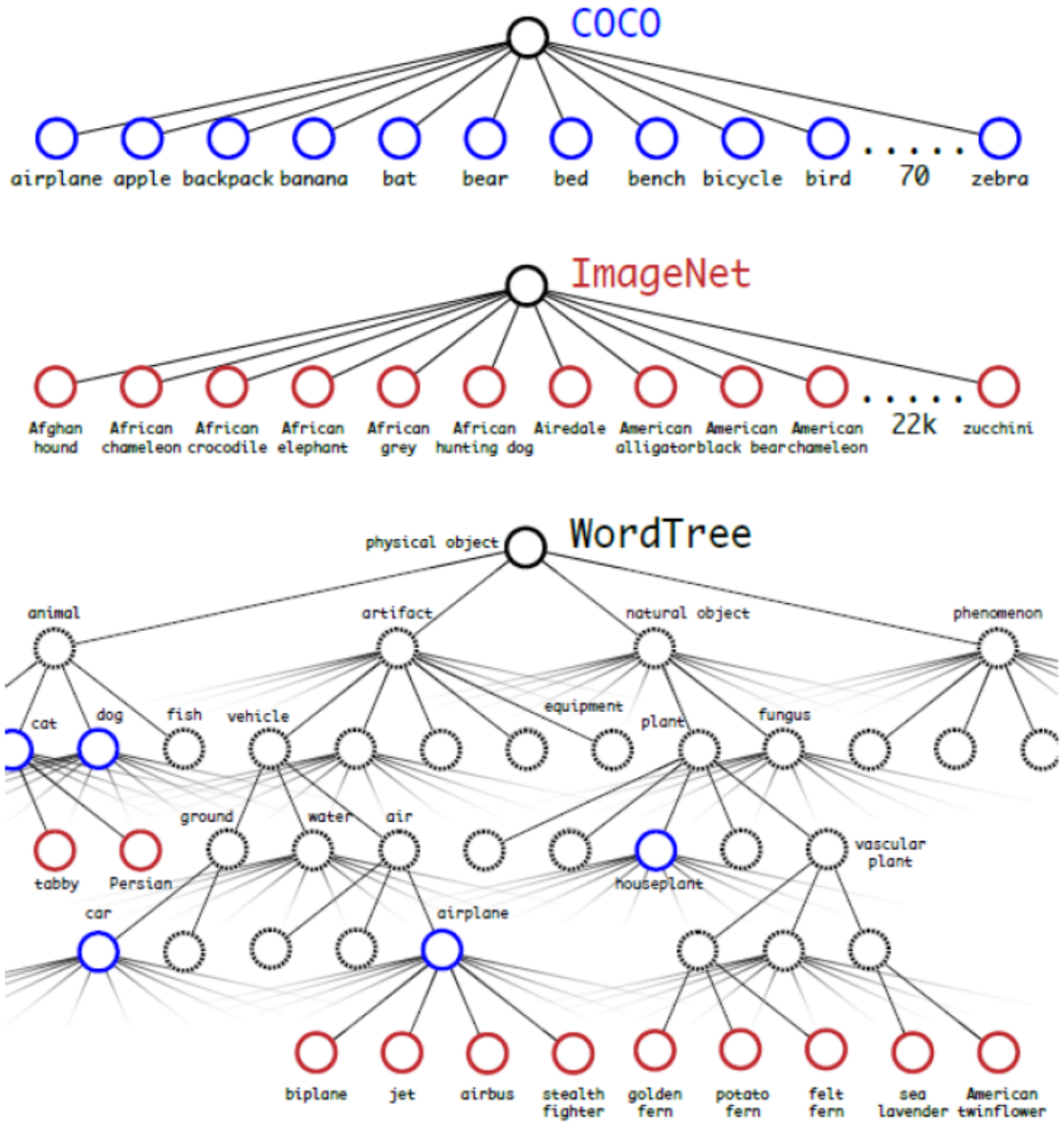




PR-023: YOLO9000: Better, Faster, Stronger, https://www.youtube.com/watch?v=6fdclSGgeio

Stronger

- Dataset Combination with WordTree
  - Example of using WordTree to combine the labels from ImageNet and COCO.
    - COCO – general concepts(higher nodes)
    - ImageNet – specific concepts(lower nodes and leaves)



PR-023: YOLO9000: Better, Faster, Stronger, https://www.youtube.com/watch?v=6fdcISGgeio

Stronger

- Joint Classification and Detection
  - New dataset created using the COCO detection dataset and the top 9000 classes from the full ImageNet release. ImageNet detection challenge dataset is also added for evaluation
  - Using YOLOv2 but only 3 priors instead of 5 to limit output size
  - The network sees a detection image, backpropagate loss as normal. For classification loss, only backpropagate loss at or above the corresponding level and only backpropagate classification loss
    - Simply find the b-box that predicts the highest probability for that class and compute the loss on just its predicted tree

PR-023: YOLO9000: Better, Faster, Stronger, https://www.youtube.com/watch?v=6fdclSGgeio

Stronger

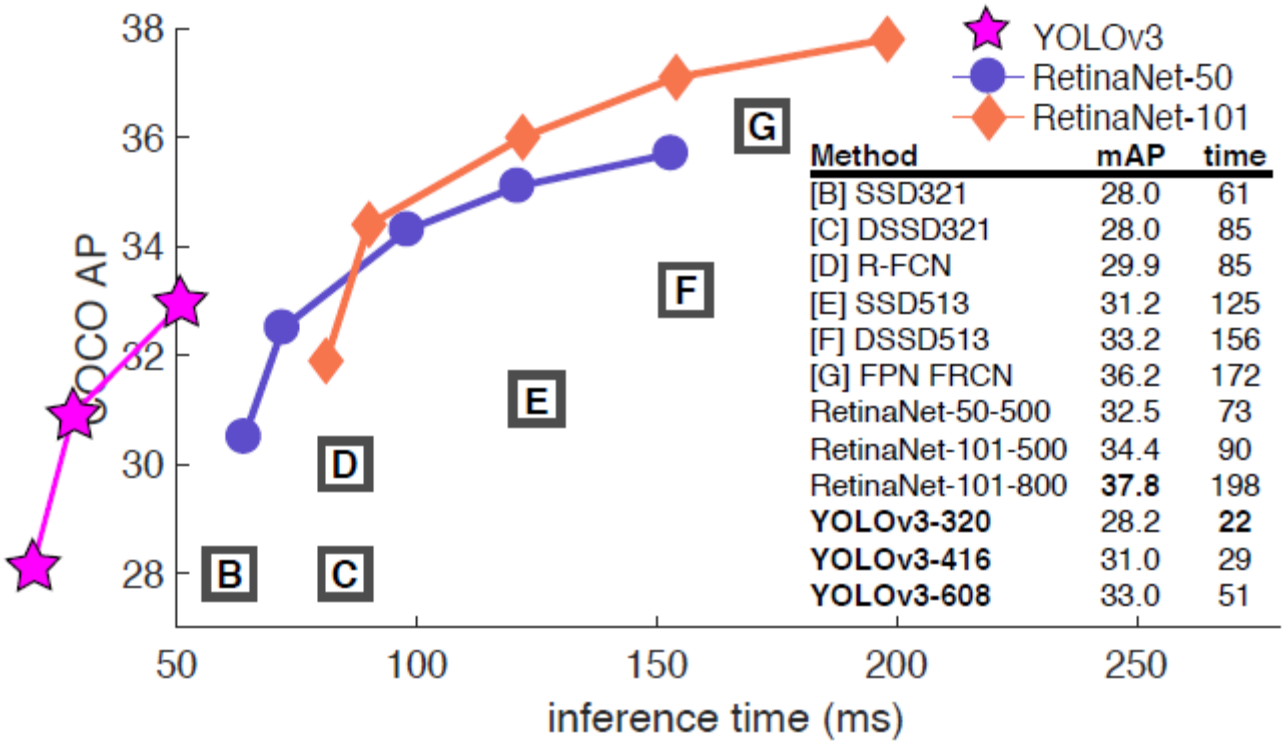
- Joint Classification and Detection
  - Evaluating YOLO9000 on the ImageNet detection task
    - ImageNet only share 44 object categories with COCO
    - 19.7mAP overall with 16.0mAP on the disjoint 156 object classes that it has never seen
  - YOLO9000 learns new species of animals well but struggles with learning categories like clothing and equipment

diaper	0.0
horizontal bar	0.0
rubber eraser	0.0
sunglasses	0.0
swimming trunks	0.0
...	
red panda	50.7
fox	52.1
koala bear	54.3
tiger	61.0
armadillo	61.7

Conclusion

- YOLOv2 is fast and accurate
- YOLO9000 is a strong step towards closing the dataset size gap between detection and classification
- Dataset combination using hierarchical classification would be useful in the classification and segmentation domains

YOLOv3: An Incremental Improvement



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

Figure 1. We adapt this figure from the Focal Loss paper [9]. YOLOv3 runs significantly faster than other detection methods with comparable performance. Times from either an M40 or Titan X, they are basically the same GPU.

YOLOv3: An Incremental Improvement

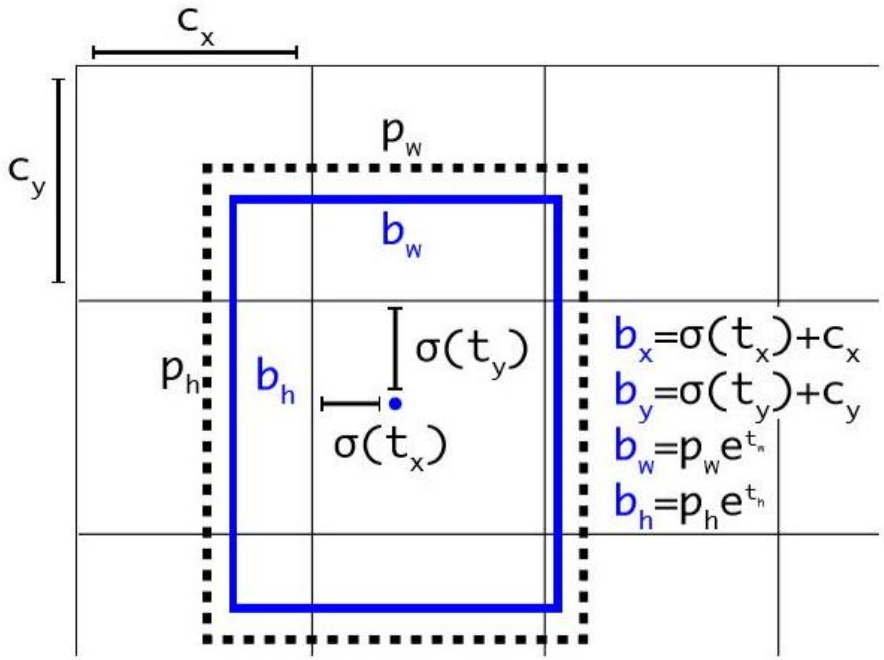
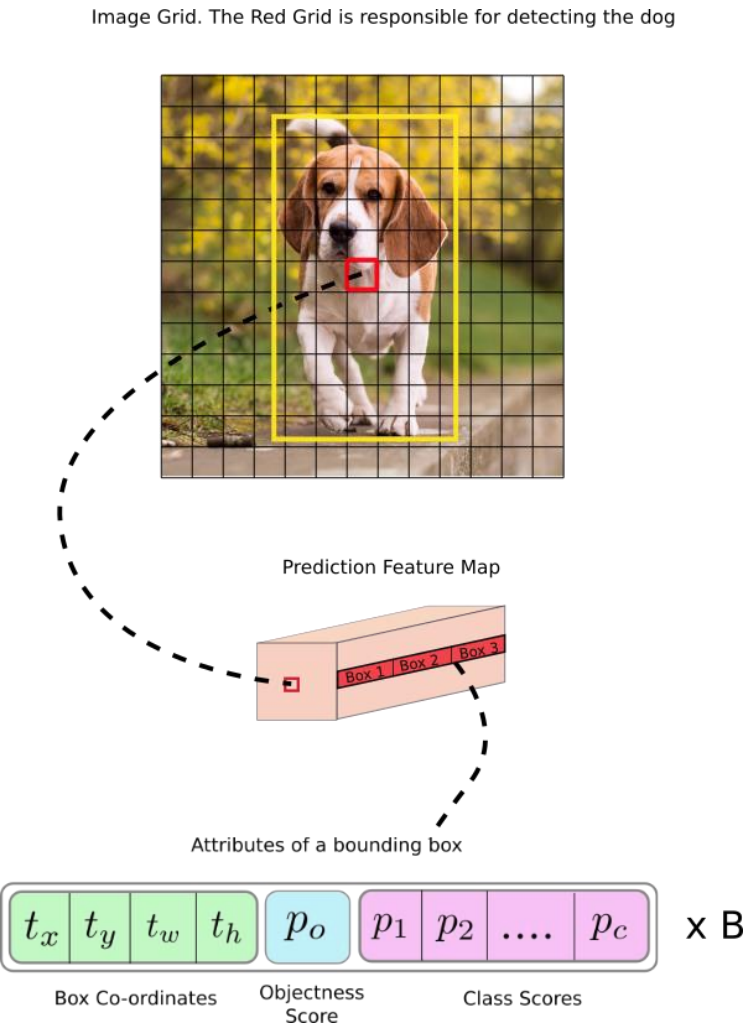


Figure 3: Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function.

YOLO v2

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

YOLOv3: An Incremental Improvement



1-2. Class Prediction

multi-label이 있을 수 있으므로 class prediction으로 softmax를 쓰지 않고 independent logistic classifiers를 썼다. 따라서 loss term도 binary cross-entropy로 바꾸었다. 이는 좀 더 복잡한 데이터셋(Open Image Dataset)을 학습하는데 도움이 되었다.(multi-label을 예측할 때 좋았다고 한다)

1-3. Predictions Accross Scales

- 3개의 bounding box
- 3개의 feature map 활용 (다른 scale, 각각 2배씩 차이)
- 한 feature map에서의 output 형태는 Grid x Grid x (#bb *(offset + objectiveness + class)) = NxNx(3x(4+1+80))
- 총 9개(3개 바운딩박스 x 3개 피쳐맵)의 anchor box는 k-means clustering을 통해 결정
- (10x13), (16x30), (33x23), (30x61), (62x45), (59x119), (116x90), (156x198), (373x326)

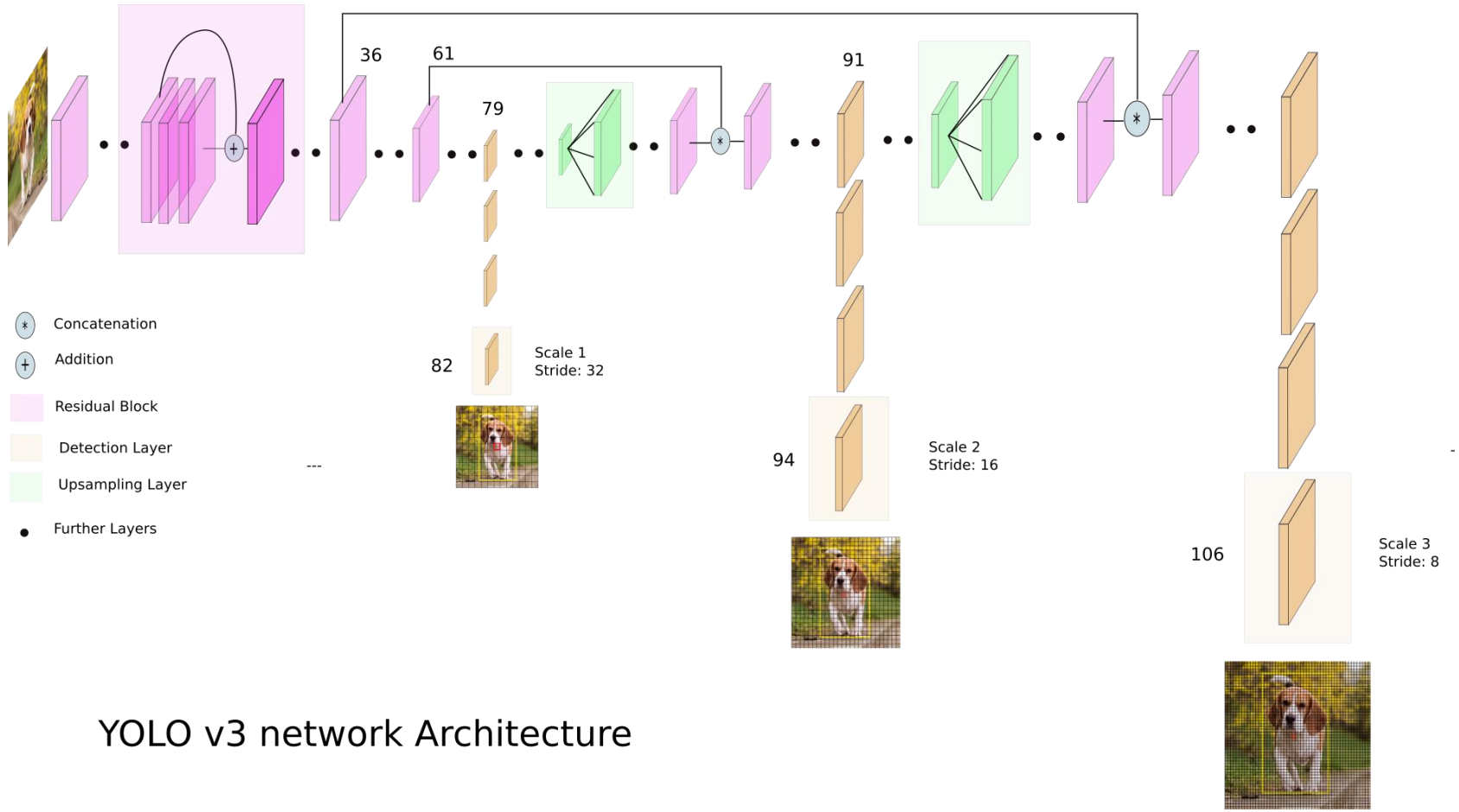
How to implement a YOLO (v3) object detector from scratch in PyTorch: Part 1, https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/

[논문] YOLOv3: An Incremental Improvement 분석, https://taeu.github.io/paper/deeplearning-paper-yolov3/

YOLOv3: An Incremental Improvement

	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.**



YOLO v3 network Architecture

What's new in YOLO v3?, https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b

YOLOv3: An Incremental Improvement

	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 $3.8\times$ 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.