Object Detection - Yolo

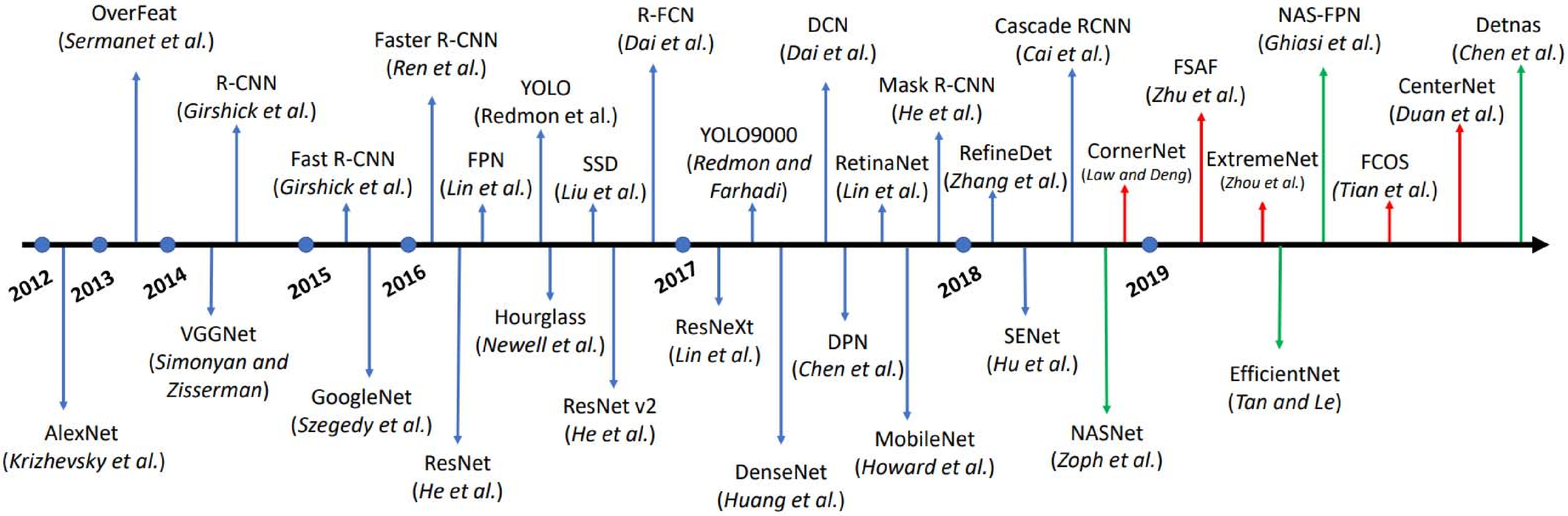
Artificial Intelligence

Creating the Future

Dong-A University

Division of Computer Engineering &
Artificial Intelligence

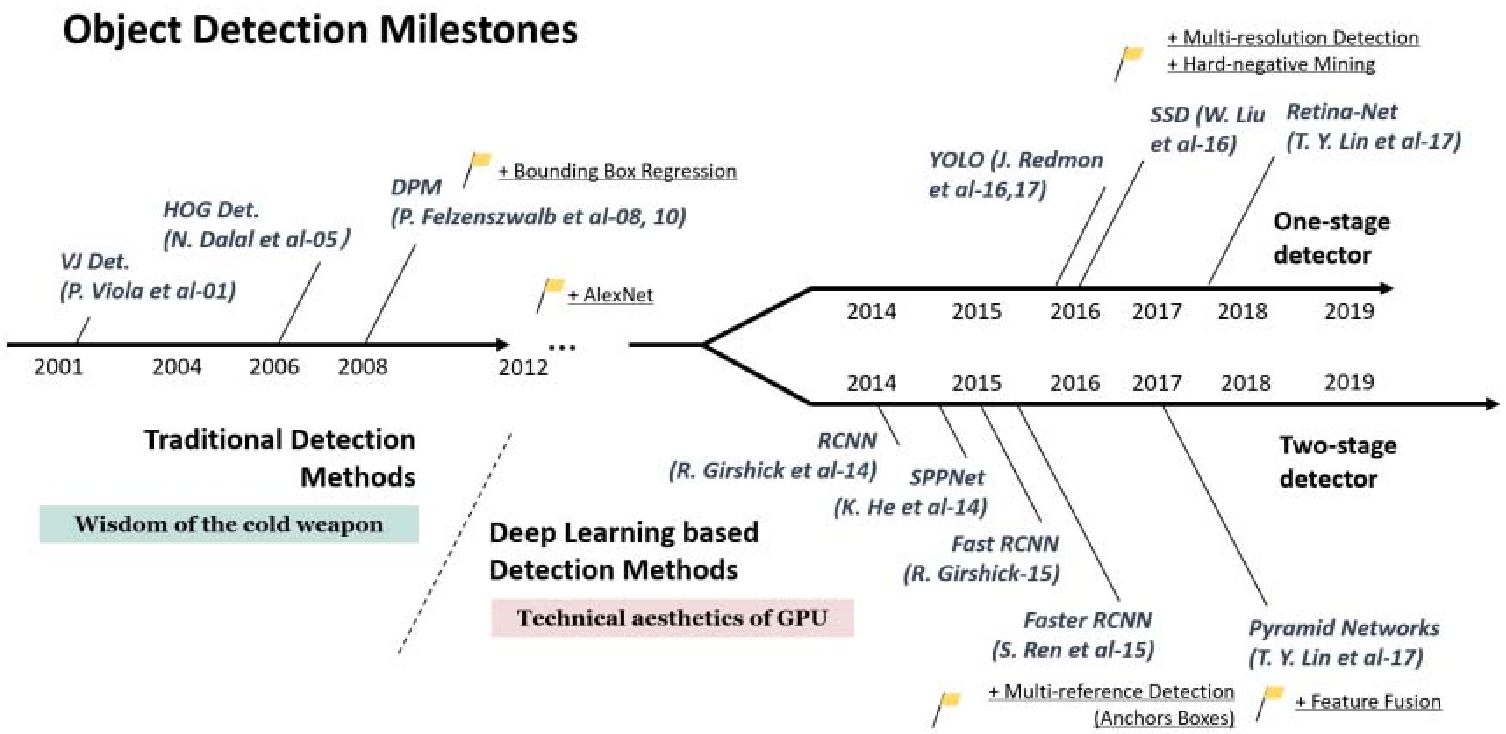
Object Detection



Major milestone in object detection research based on deep convolution neural networks since 2012. The trend in the last year has been designing object detectors based on anchor-free(in red) and AutoML(in green) techniques, which are potentially two important research directions in the future.
(X. Wu, et al, “Recent Advances in Deep Learning for Object Detection,” https://arxiv.org/pdf/1908.03673v1.pdf

Object Detection

- Two approaches for Object Detection



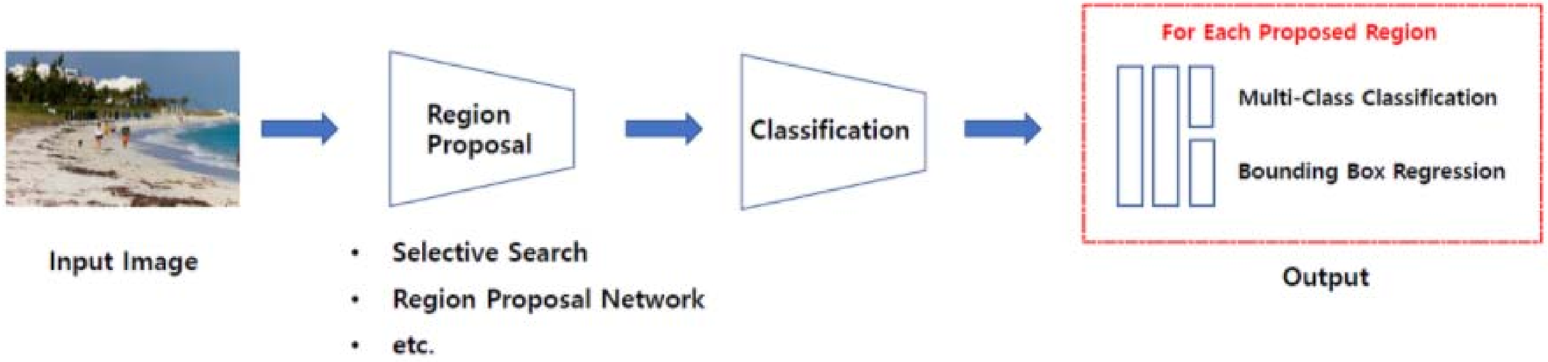
A road map of object detection. Milestone detectors in this figure: VJ Det. [10, 11], HOG Det. [12], DPM [13-15], RCNN [16], SPPNet [17], Fast RCNN [18], Faster RCNN [19], YOLO [20], SSD [21], Pyramid Networks [22], Retina-Net [23].
(Z. Zou, “Object Detection in 20 Years: A Survey,” arXiv:1905.05055v2.

Object Detection

❖ Two approaches for Object Detection

1) 2-stage Detector

- 특징 추출과 객체 분류의 두 가지 문제를 순차적으로 해결
- 정확도는 높으나, 속도가 느림

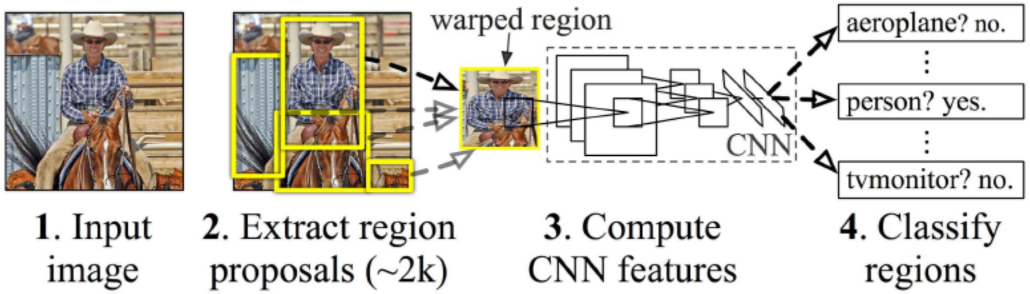


- Selective Search
- Region Proposal Network
- etc.

- Prior Arts like R-CNN, first generates 2K region proposals (bounding box candidates), then detect object within the each region proposal as below:

✓ Selective search를 통해 RoI(Region Of Interest)를 약 2000개 추출
✓ RoI 크기를 조절해 동일한 사이즈로 만듬.
✓ RoI를 CNN에 입력하여 feature를 추출
✓ CNN을 통해 나온 feature를 SVM에 넣어줘 classification 함
✓ CNN을 통해 나온 feature를 regression을 통해 bounding box를 예측함 (Bounding box regression)

R-CNN: *Regions with CNN features*

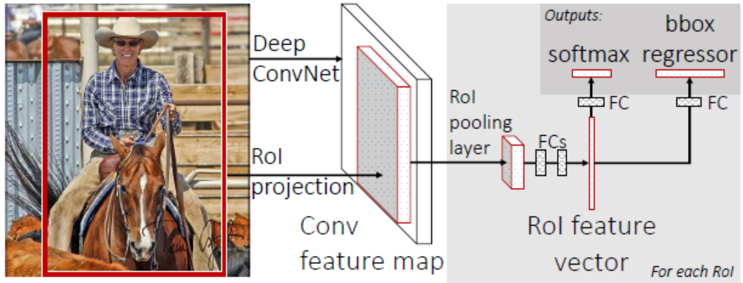


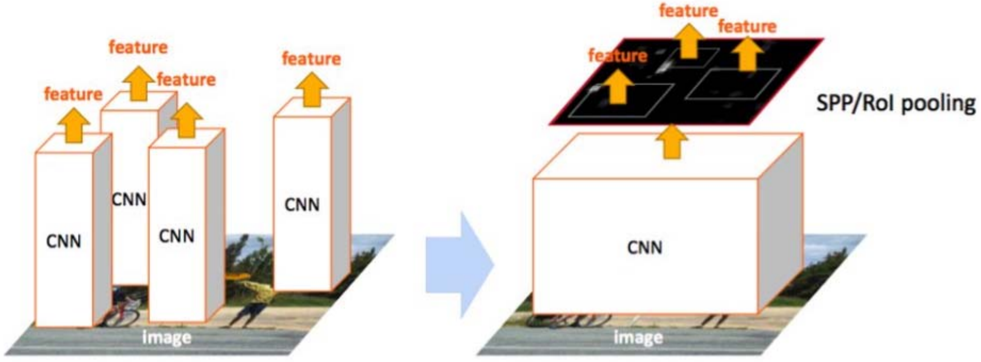
Object Detection

❖ Two approaches for Object Detection

1) 2-stage Detector

- Fast R-CNN





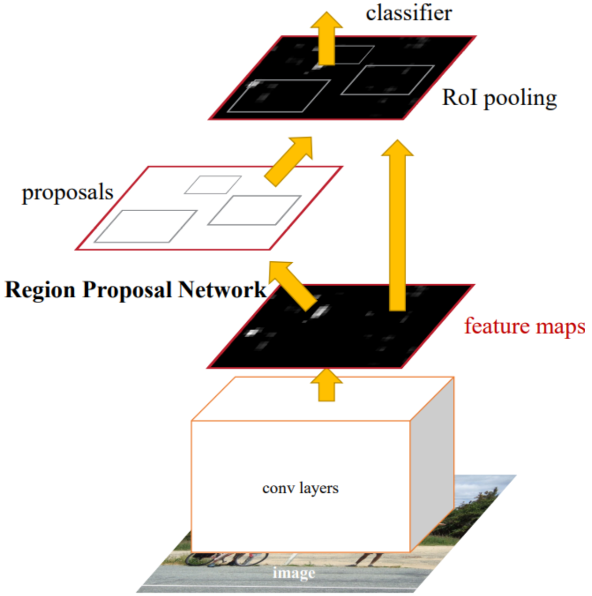
R-CNN

- Extract image regions
- 1 CNN per region (2000 CNNs)
- Classify region-based features
- Complexity: ~$\underset{H}{\boxed{224}} \times \underset{W}{\boxed{224}} \times 2000$

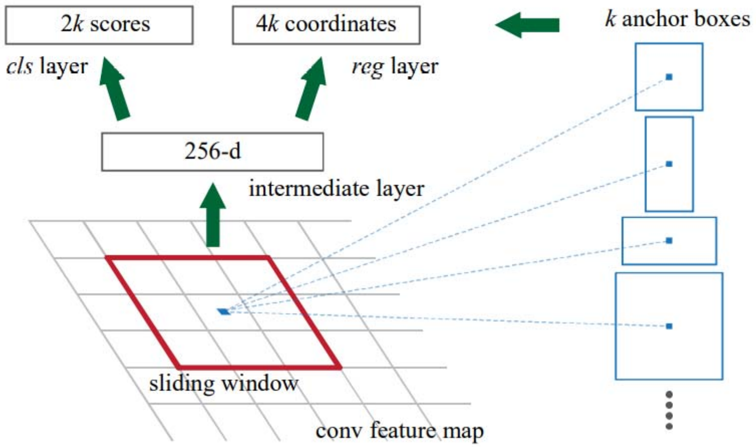
SPP-net & Fast R-CNN (the same forward pipeline)

- 1 CNN on the entire image
- Extract features from feature map regions
- Classify region-based features
- Complexity: ~$\underset{H}{\boxed{600}} \times \underset{W}{\boxed{1000}} \times 1$
- ~160x faster than R-CNN

- Faster R-CNN : a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.



Region Proposal Network (RPN) module

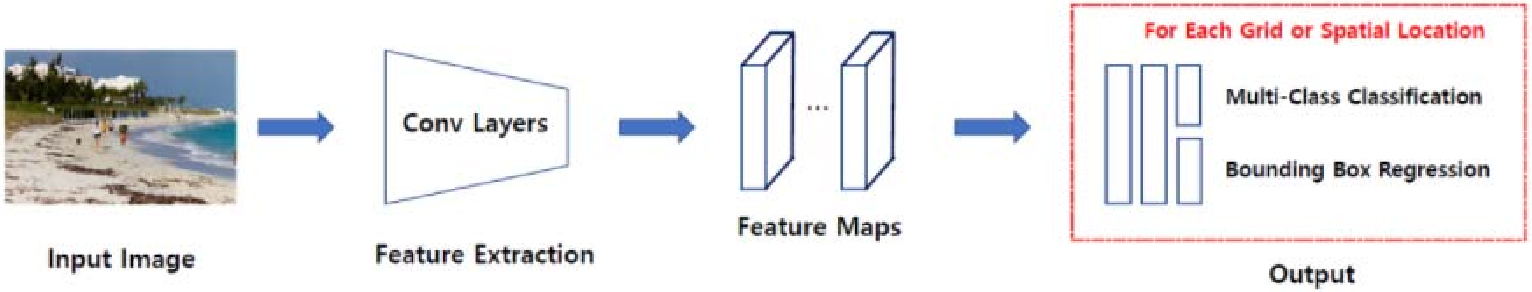


Object Detection

❖ Two approaches for Object Detection

2) 1-stage Detector

- 특징 추출과 객체 분류를 한 번에 처리 (두 문제를 단일 신경망으로 해결)
- 정확도는 높지 않으나, 속도가 빠름
- YOLO 계열과 SSD 계열



- **YOLO가 object detection을 single regression problem으로 재구성** 했다고 함.
- YOLO v1은 single convolutional network로 이미지를 입력받아, 여러 개의 Bounding Box와 각 Box의 class를 예측함. 그리고 non-max suppression을 통해 최종 Bounding Box를 선정함.

- Image -> bounding box coordinate and class probability
- Extremely fast
- Global reasoning
- Generalize representation

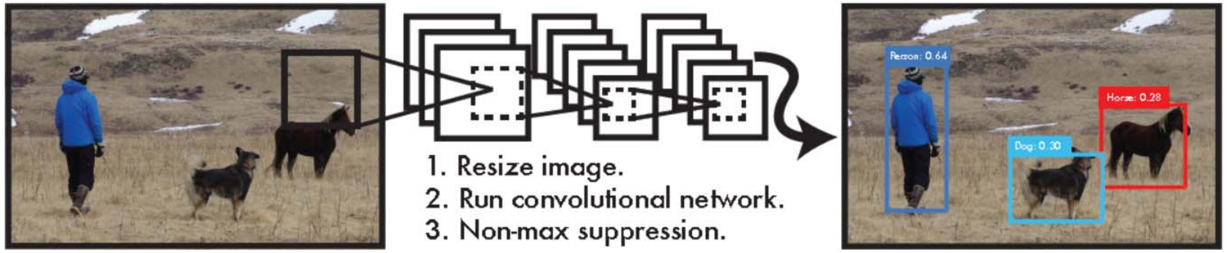
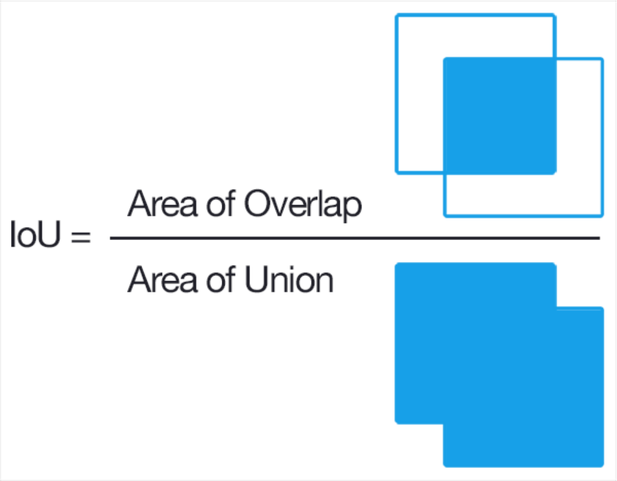


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

IoU (Intersection of Union)

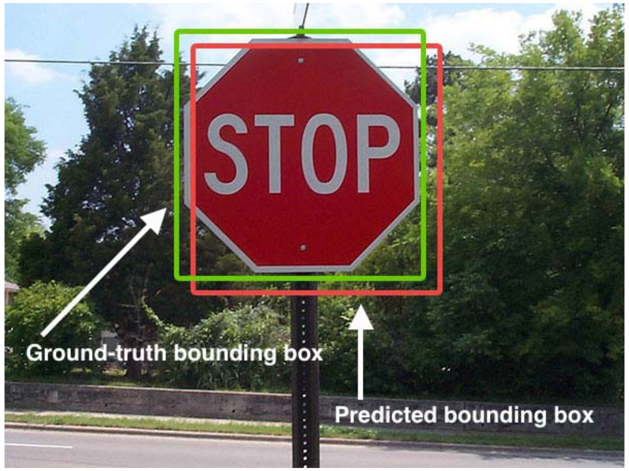
[발췌 자료] https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

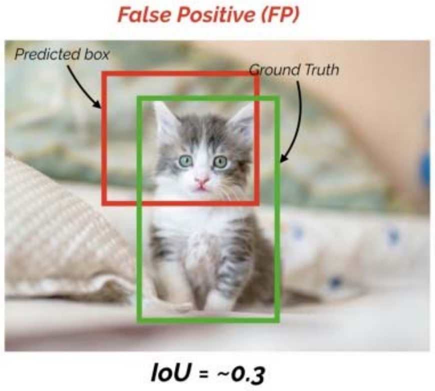
- ***IoU is an evaluation metric*** used to **measure the accuracy of an object detector** on a particular dataset.
- Any algorithm that provides predicted bounding boxes as output can be evaluated using IoU. Need two things;
  1) **The ground-truth bounding boxes** (i.e., the hand labeled bounding boxes from the testing set that specify where in the image our object is).
  2) **The predicted bounding boxes** from our model.

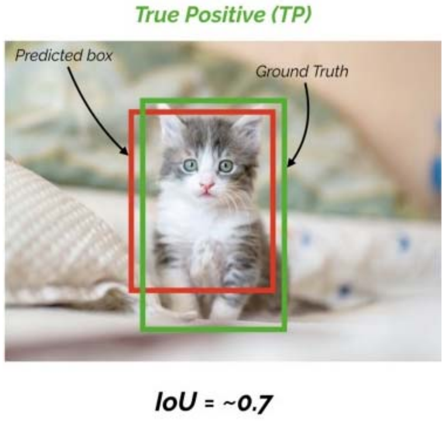


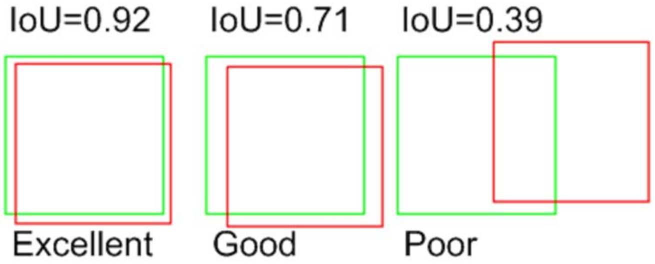
- *Area of overlap* between the predicted bounding box and the ground-truth bounding box.
- *Area of union* the area encompassed by both the predicted bounding box and the ground-truth bounding box.

If IoU Threshold = 0.5









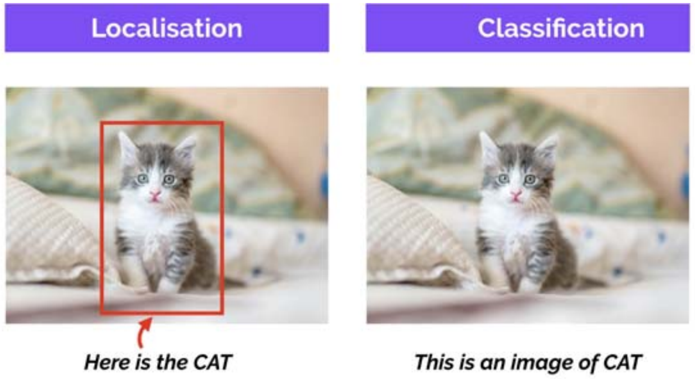
- *PASCAL VOC Challenge : An Intersection over Union score > 0.5 is normally considered a “good” prediction.*

mAP (mean Average Precision)

[발췌 자료] https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2

- **mAP** is a popular **evaluation metric** used for **object detection** (i.e. ***localization*** and ***classification***). Localization determines the location of an instance (e.g. bounding box coordinates) and classification tells you what it is (e.g. a dog or cat).

- ***Precision*** measures how accurate is your predictions. i.e. the percentage of your predictions are correct. (모델이 True라고 예측한 것 중 정답도 True인 것의 비율)
- ***Recall*** measures how good you find all the positives. For example, we can find 80% of the possible positive cases in our top K predictions. (실제 정답이 True인 것 중에서 모델이 True라고 예측한 것의 비율)



mAP is not calculated by taking the average of precision values.

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

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

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

example

$$Precision = \frac{TP}{\text{total positive results}}$$

$$Recall = \frac{TP}{\text{total cancer cases}}$$

Object detection systems make predictions in terms of *a bounding box* and *a class label*.

For each bounding box, we measure an overlap between the predicted bounding box and the ground truth bounding box. This is measured by ***IoU*** (intersection over union).

For object detection tasks, we calculate **Precision and Recall using IoU value for a given IoU threshold**.

mAP (mean Average Precision)

The general definition for the Average Precision (AP) is finding the area under the precision-recall curve above.

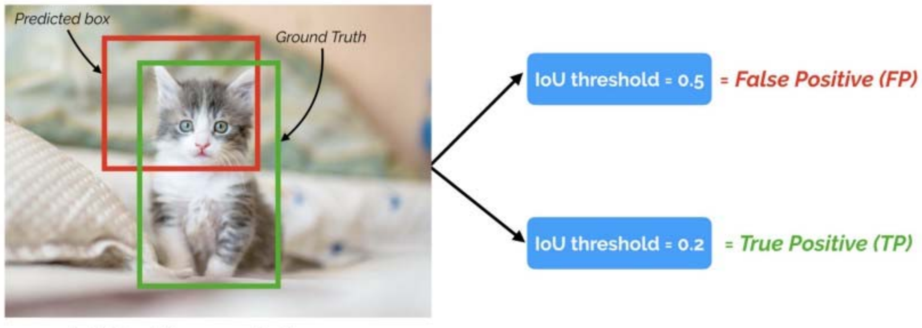
mAP (mean average precision) is the average of AP.

- AP is averaged over all categories. The mean Average Precision or mAP score is calculated by taking the mean AP over all classes and/or overall IoU thresholds, depending on different detection challenges that exist.

✓ *In PASCAL VOC2007 challenge, AP for one object class is calculated for an IoU threshold of 0.5. So the mAP is averaged over all object classes.*

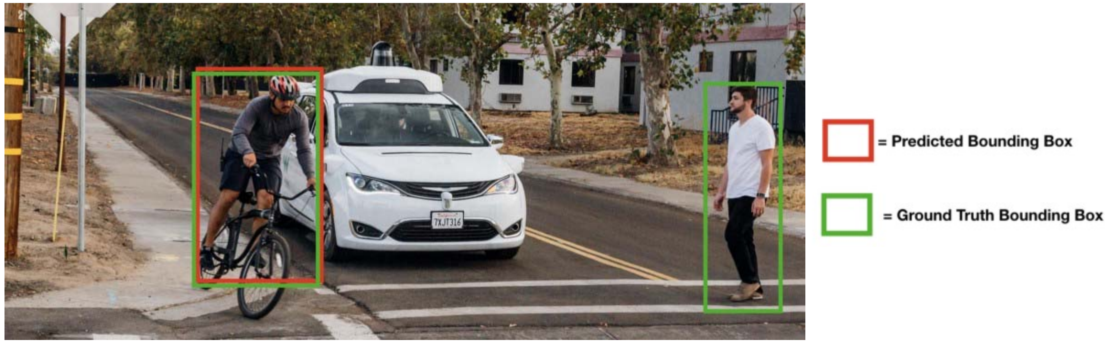
✓ *For the COCO 2017 challenge, the mAP is averaged over all object categories and 10 IoU thresholds.*

- For a prediction, we may get different binary TRUE or FALSE positives, by changing the IoU threshold.



IoU for the prediction = ~0.3

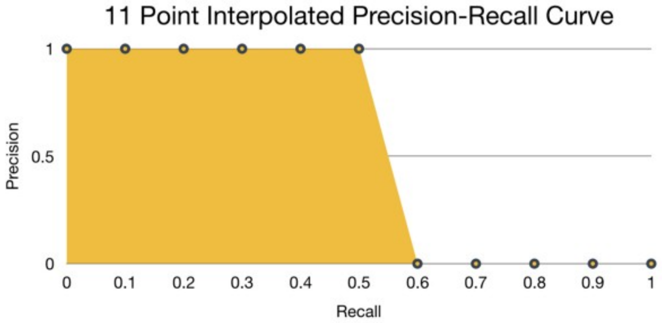
[발췌 자료] https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2



Calculate the AP at IoU threshold 0.5.   TP=1, FP=0. FN=1

$$\textbf{Precision} = \frac{TP}{TP+FP} = \frac{1}{1+0} = 1 \qquad \textbf{Recall} = \frac{TP}{TP+FN} = \frac{1}{1+1} = 0.5$$

Plot the 11 points interpolated Precision-Recall curve.



We now calculate AP by taking the area under the PR curve. This is done by segmenting the recalls evenly to 11 parts: {0,0.1,0.2,…,0.9,1}.

$$AP = \frac{1}{11}\sum_{Recall_i} Precision(Recall_i) = 1$$

$$AP = \frac{1}{11}\sum_{0,0.1,0.2,0.3,..0.9,1}(1*6)+(0*5) = 0.545$$

So mAP@0.5 for the image is 0.545, not 1.

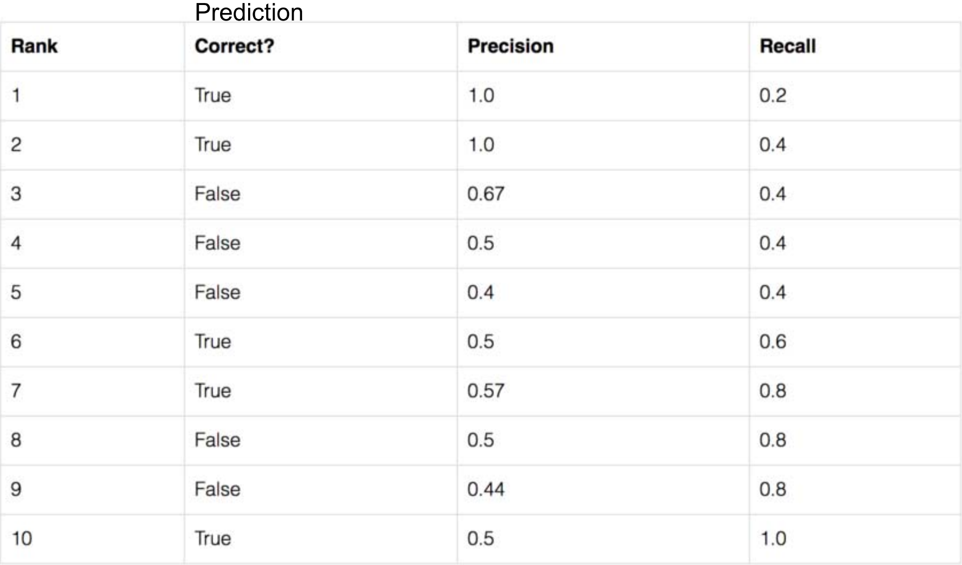
mAP (mean Average Precision)

AP

- The general definition for the Average Precision (AP) is finding the area under the precision-recall curve above.

$$AP = \int_0^1 p(r)dr$$

- 5개 사과가 포함되어 있는 이미지에 Detector가 10개의 사과를 검출했다고 가정함. 각각의 검출 결과를 confidence가 높은 순으로 정렬한 것임.

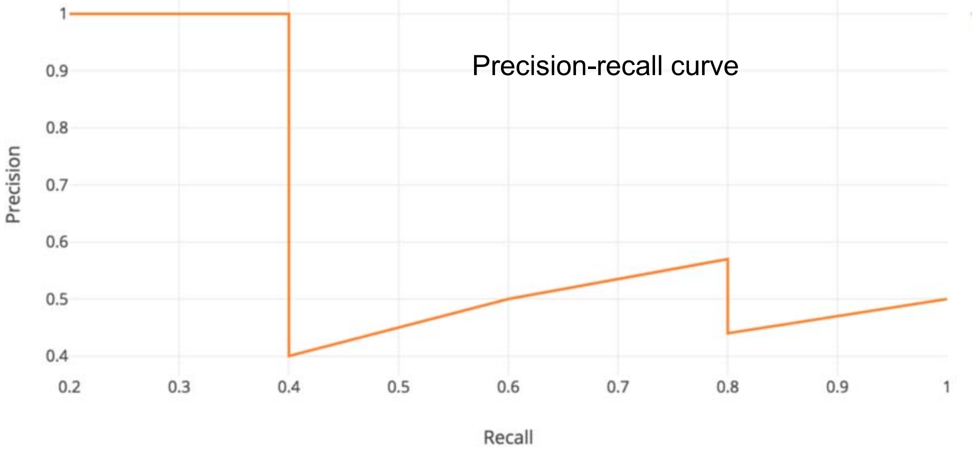


	Prediction		
Rank	**Correct?**	**Precision**	**Recall**
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

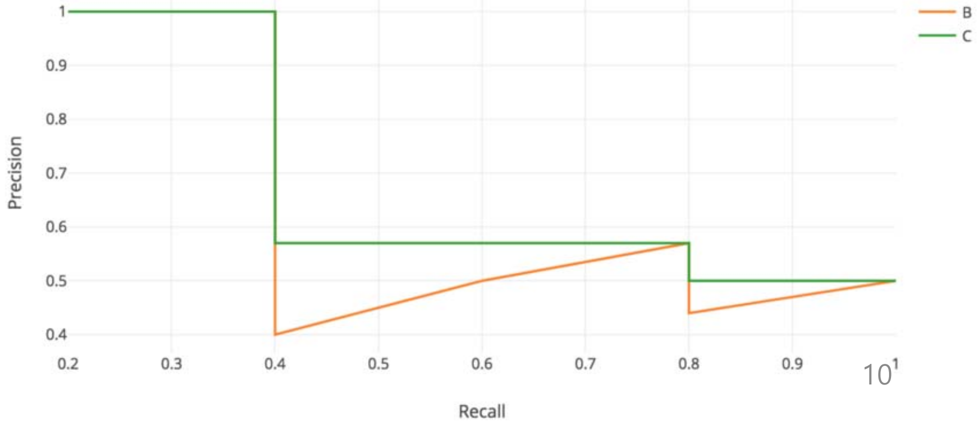
- recall은 서서히 증가하다가 rank 10에서 5번째 True때 1.0이 되었고,
- 반면에 Precision은 들쑥날쑥 한 것을 확인할 수 있으며 5개의 사과 중 10개의 사과를 검출했으므로 최종적으로 0.5의 precision가 됨.

[발췌 자료] https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

- Plot the precision against the recall value to see this zig-zag pattern



- Precision and recall are always between 0 and 1. Therefore, AP falls within 0 and 1 also. Before calculating AP for the object detection, we often smooth out the zigzag pattern first.

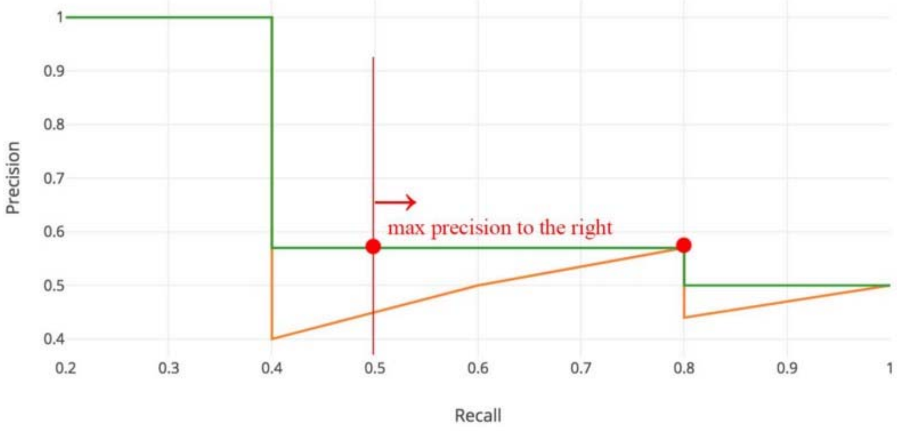


mAP (mean Average Precision)

[발췌 자료] https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

AP

- Graphically, at each recall level, we replace each precision value with the maximum precision value to the right of that recall level.

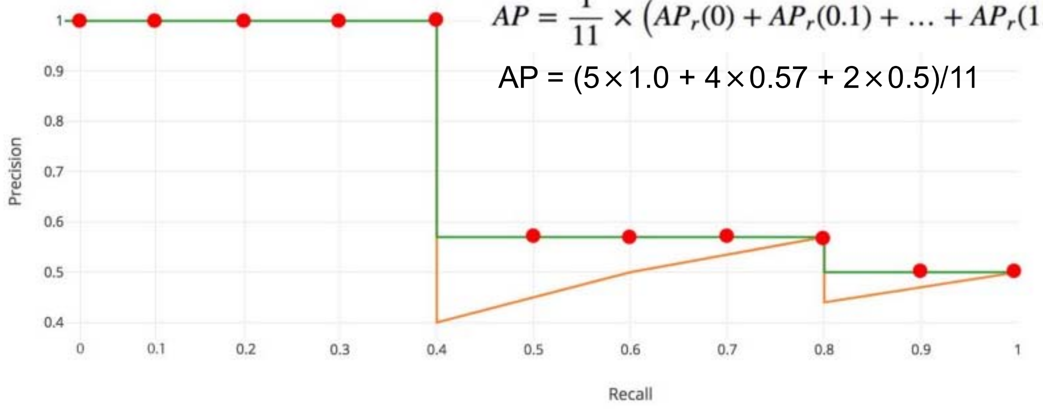


- So the orange line is transformed into the green lines and the curve will decrease monotonically instead of the zigzag pattern. The calculated AP value will be less suspectable to small variations in the ranking. Mathematically, we replace the precision value for recall r̂ with the maximum precision for any recall ≥ r̂.

$$p_{interp}(r) = \max_{\tilde{r} \geq r} p(\tilde{r})$$

Interpolated AP

- PASCAL VOC is a popular dataset for object detection. For the *PASCAL VOC challenge*, a prediction is positive if IoU ≥ 0.5. Also, if multiple detections of the same object are detected, it counts the first one as a positive while the rest as negatives.
- In *Pascal VOC2008*, ***an average for the 11-point interpolated AP*** is calculated.



$$AP = \frac{1}{11} \times \left(AP_r(0) + AP_r(0.1) + \ldots + AP_r(1.0)\right)$$

AP = (5×1.0 + 4×0.57 + 2×0.5)/11

$$AP = \frac{1}{11} \sum_{r \in \{0.0, \ldots, 1.0\}} AP_r \quad = \frac{1}{11} \sum_{r \in \{0.0, \ldots, 1.0\}} p_{interp}(r) \qquad p_{interp}(r) = \max_{\tilde{r} \geq r} p(\tilde{r})$$

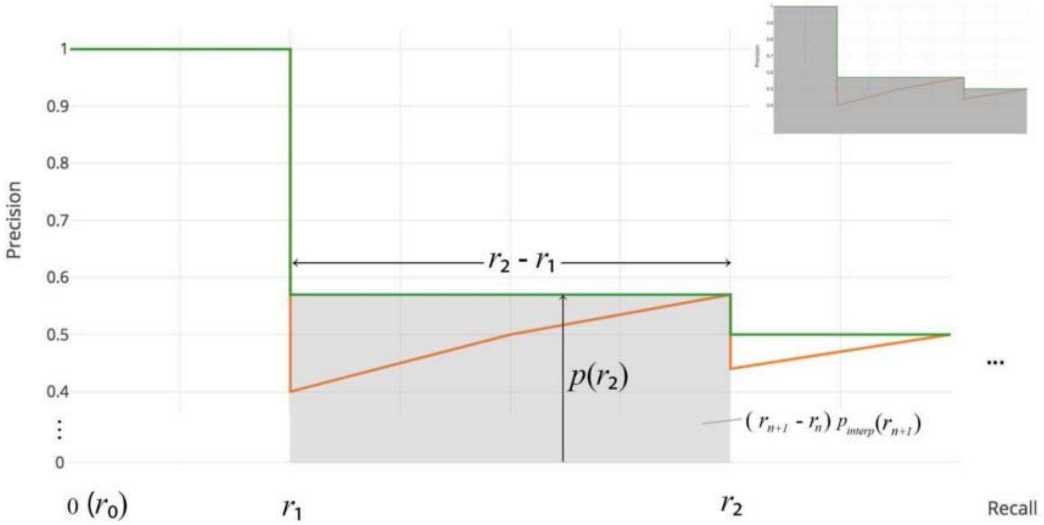
- This interpolated method is an approximation which suffers two issues. It is less precise. Second, it lost the capability in measuring the difference for methods with low AP. Therefore, a different AP calculation is adopted after 2008 for PASCAL VOC.

mAP (mean Average Precision)

[발췌 자료] https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

AP (Area under curve AUC)

- ***PASCAL VOC2010–2012*** samples the curve at all unique recall values (r_1, r_2, …), whenever *the maximum precision value drops*. With this change, we are measuring the exact area under the precision-recall curve after the zigzags are removed.



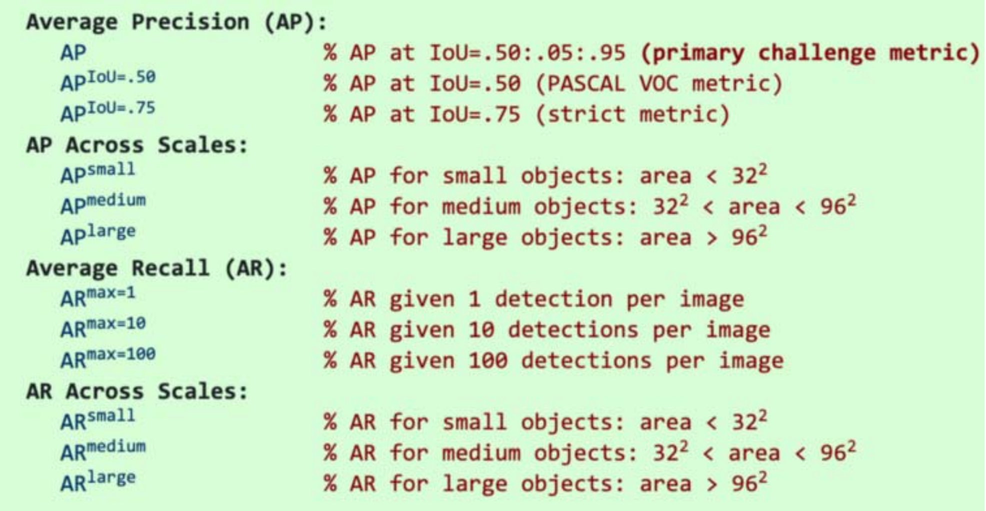
- No approximation or interpolation is needed. Instead of sampling 11 points, we sample $p(r_i)$ whenever it drops and computes AP as the sum of the rectangular blocks.

$$\mathrm{AP} = \Sigma\,(r_{n+1} - r_n)\, p_{interp}(r_{n+1}) \qquad p_{interp}(r_{n+1}) = \max_{\tilde{r} \geq r_{n+1}} p(\tilde{r})$$

- This definition is called **the Area Under Curve (AUC)**

COCO mAP

- In COCO mAP, a *101-point interpolated AP definition* is used in the calculation.
- For COCO, *AP is the average over multiple IoU (the minimum IoU to consider a positive match)*. AP@[.5:.95] corresponds to the average AP for IoU from 0.5 to 0.95 with a step size of 0.05. For the COCO competition, AP is the average over 10 IoU levels on 80 categories (AP@[.50:.05:.95]: start from 0.5 to 0.95 with a step size of 0.05).
- The following are some other metrics collected for the COCO dataset.



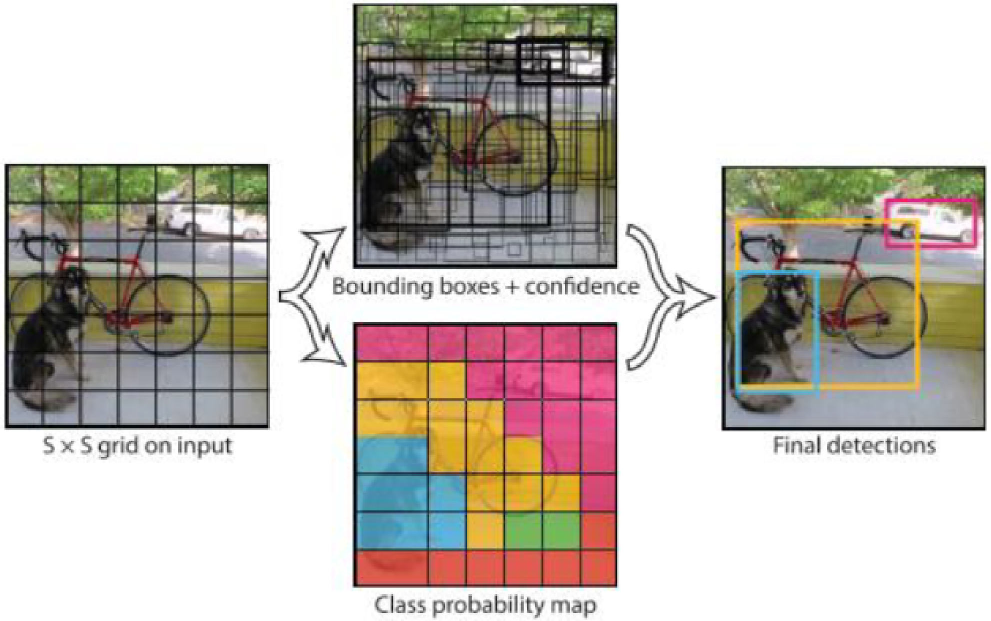
```
Average Precision (AP):
   AP                  % AP at IoU=.50:.05:.95 (primary challenge metric)
   AP^IoU=.50          % AP at IoU=.50 (PASCAL VOC metric)
   AP^IoU=.75          % AP at IoU=.75 (strict metric)
AP Across Scales:
   AP^small            % AP for small objects: area < 32^2
   AP^medium           % AP for medium objects: 32^2 < area < 96^2
   AP^large            % AP for large objects: area > 96^2
Average Recall (AR):
   AR^max=1            % AR given 1 detection per image
   AR^max=10           % AR given 10 detections per image
   AR^max=100          % AR given 100 detections per image
AR Across Scales:
   AR^small            % AR for small objects: area < 32^2
   AR^medium           % AR for medium objects: 32^2 < area < 96^2
   AR^large            % AR for large objects: area > 96^2
```

1 Quick Review

PR-249: YOLOv4: Optimal Speed and Accuracy of Object Detection, Ho Seong Lee
https://www.slideshare.net/HoseongLee6/yolov4-optimal-speed-and-accuracy-of-object-detection-review
https://www.youtube.com/watch?v=CXRlpsFpVUE

YOLO v1 ~ v3 quick review: YOLO v1

- Very fast one-stage approach!
- Image → bounding box coordinate and class probability



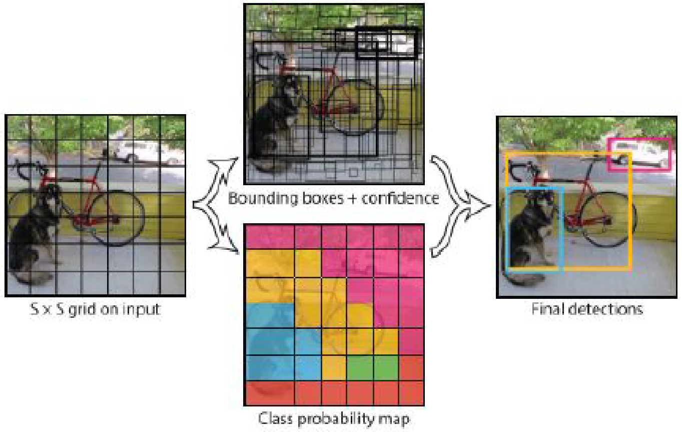
$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$
$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i}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

PR-249: YOLOv4: Optimal Speed and Accuracy of Object Detection, Ho Seong Lee
https://www.slideshare.net/HoseongLee6/yolov4-optimal-speed-and-accuracy-of-object-detection-review

YOLO v1 ~ v3 quick review: YOLO v2

- YOLO v1 + many algorithms

YOLO v1



+

Better

Batch Normalization
High resolution classifier
Anchor boxes
Dimension clusters
Direct location prediction
Fine-grained features
Multi-scale training

Faster

Darknet-19
Transfer learning

Stronger

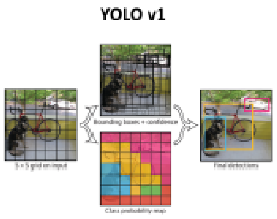
Hierarchical classification
Dataset combination with Word-tree
Joint classification and detection

PR-249: YOLOv4: Optimal Speed and Accuracy of Object Detection, Ho Seong Lee
https://www.slideshare.net/HoseongLee6/yolov4-optimal-speed-and-accuracy-of-object-detection-review

YOLO v1 ~ v3 quick review: YOLO v3

- YOLO v2 + many algorithms (**YOLOv3: An Incremental Improvement**)

YOLO v2



+

Better

Batch Normalization
High resolution classifier
Anchor boxes
Dimension clusters
Direct location prediction
Fine-grained features
Multi-scale training

Faster

Darknet-19
Transfer learning

Stronger

Hierarchical classification
Dataset combination with Word-tree
Joint classification and detection

+

Bounding box prediction → sum of squared loss

Class prediction → Multilabel classification

Predictions across scales

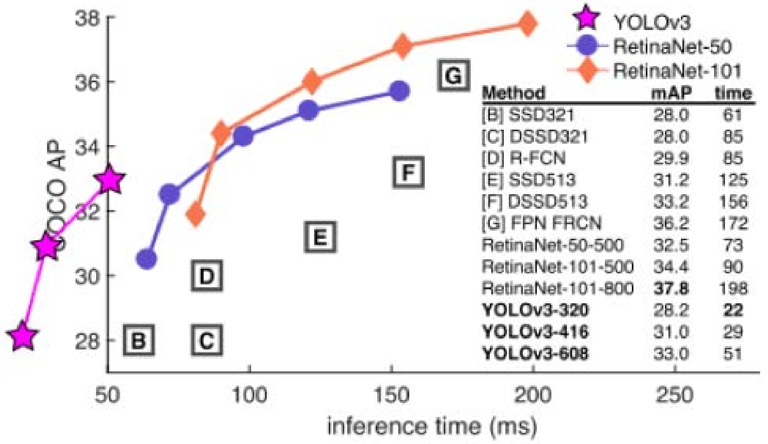
Darknet-53

PR-249: YOLOv4: Optimal Speed and Accuracy of Object Detection, Ho Seong Lee
https://www.slideshare.net/HoseongLee6/yolov4-optimal-speed-and-accuracy-of-object-detection-review

YOLOv4: Optimal Speed and Accuracy of Object Detection

- YOLO 3 + many algorithms

YOLO v3



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

+

Bag of Freebies

Bag of Specials

CSPDarknet53 + SPP, PAN

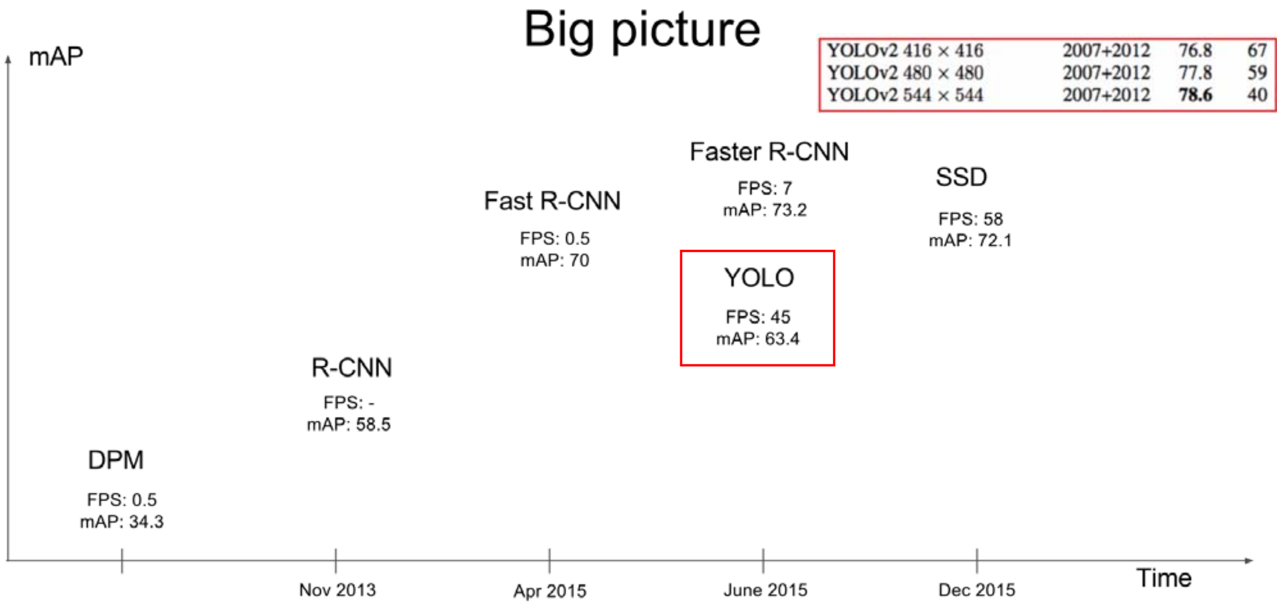
1 Yolo v1 – You Only Look Once

Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” arXiv:1506.02640, CVPR2016

[발췌 자료]
Sik-Ho Tsang, Review: YOLOv1 — You Only Look Once (Object Detection), https://towardsdatascience.com/yolov1-you-only-look-once-object-detection-e1f3ffec8a89

[PR12] You Only Look Once (YOLO): Unified Real-Time Object Detection, Taegyun Jeon
https://www.slideshare.net/TaegyunJeon1/pr12-you-only-look-once-yolo-unified-realtime-object-detection?from_action=save
https://www.youtube.com/watch?v=eTDcoeqj1_w

DeepSystem.io, “YOLO: You only look once Review”
https://www.youtube.com/watch?v=L0tzmv--CGY
https://docs.google.com/presentation/d/1aeRvtKG21KHdD5lg6Hgyhx5rPq_ZOsGjG5rJ1HP7BbA/pub?start=false&loop=false&delayms=3000&slide=id.g137784ab86_4_1318



YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

[발췌 자료] [PR12] You Only Look Once (YOLO): Unified Real-Time Object Detection, Taegyun Jeon

YOLO Main Concept

- **Object Detection**
  - Regression problem
- **Yolo**
  - Only One Feedforward
  - Global context
- **Unified (Real-time detection)**
  - YOLO : 45fps
  - Fast YOLO : 155fps
- **General representation**
  - Robust on various background
  - Other domain

Abstract

- Prior work on object detection repurposes classifiers to perform detection.
- Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities.
- A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.
- Our unified architecture is extremely fast. Our **base YOLO model** processes images in real-time at **45 fps**. A smaller version of the network, **Fast YOLO**, processes an astounding **155 fps** while still achieving ***double the mAP*** of other real-time detectors.
- Compared to state-of-the-art detection systems, **YOLO makes more localization errors but is less likely to predict false positives on background**. Finally, YOLO **learns very general representations of objects**. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

1 Yolo v1 – You Only Look Once

[발췌 자료] [PR12] You Only Look Once (YOLO): Unified Real-Time Object Detection, Taegyun Jeon

Unified Detection

YOLO suggests to have **unified network to perform all as once**. Also, **end-to-end training network** can be achieved.

All Box, All classes

- Image → S x S grids (S=7)
  - ✓ If the center of an object falls into a grid cell, that grid cell is responsible for detection that object.
- Each grid cell predicts B bounding boxes (B=2) and confidence scores for those boxes.
  - ✓ Each bounding box consists of 5 predictions;

$$\boxed{x, y, w, h, confidence}$$

$$confidence = \Pr(Object) \times IOU_{\text{pred}}^{\text{truth}}$$

- Each grid cell predicts C conditional class probabilities,

$\Pr(Class_i|Object)$ Total number of classes = 20

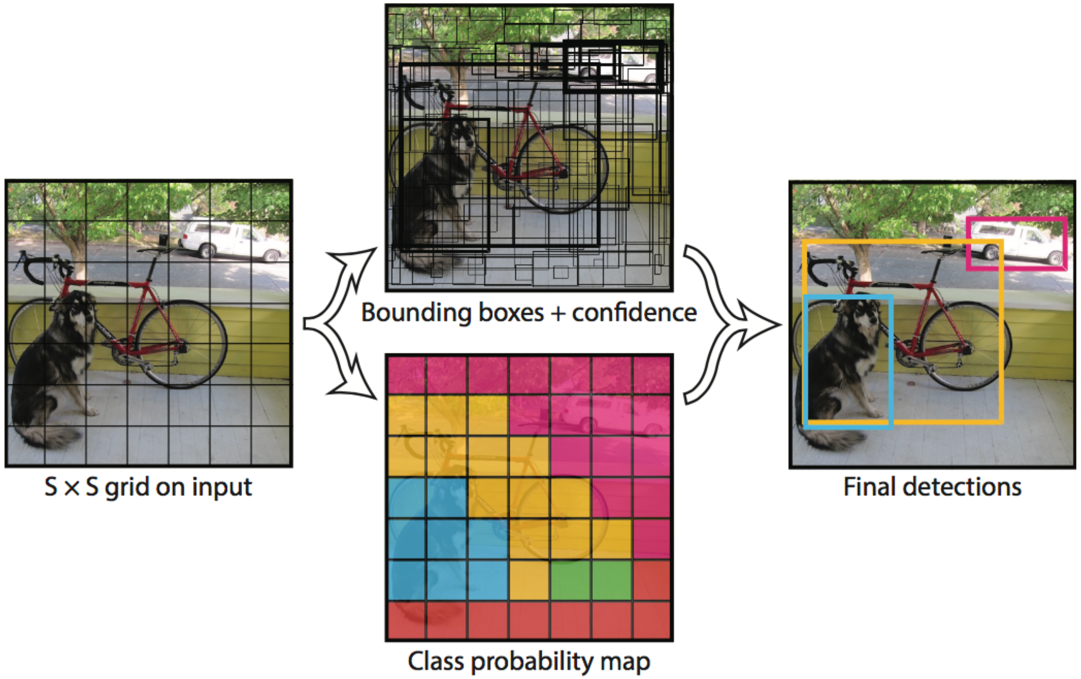


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

1 Yolo v1 – You Only Look Once

[발췌 자료] [PR12] You Only Look Once (YOLO): Unified Real-Time Object Detection, Taegyun Jeon

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

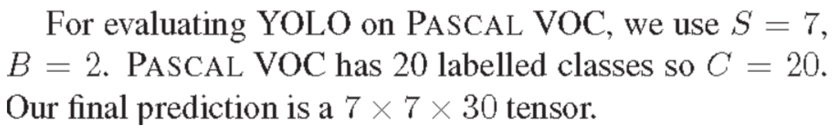
$$\Pr(Class_i|Object) \times \Pr(Object) \times IOU_{\text{pred}}^{\text{truth}}$$

$$=\Pr(Class_i) \times IOU_{\text{pred}}^{\text{truth}}$$

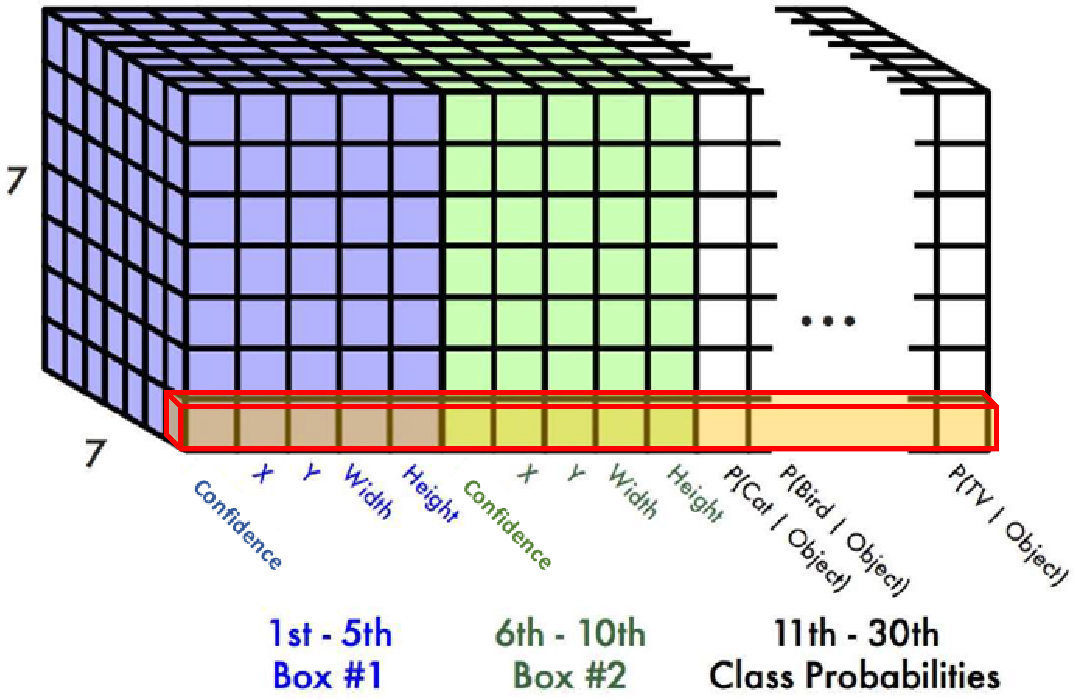
- The output size becomes: S x S x (B x 5 + C)

$7 \times 7 \times (2 \times 5+20)=1470$

(S=7, B=2, C=20 (total number of classes))



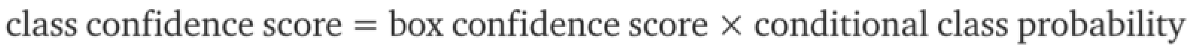
For evaluating YOLO on PASCAL VOC, we use $S = 7$, $B = 2$. PASCAL VOC has 20 labelled classes so $C = 20$. Our final prediction is a $7 \times 7 \times 30$ tensor.



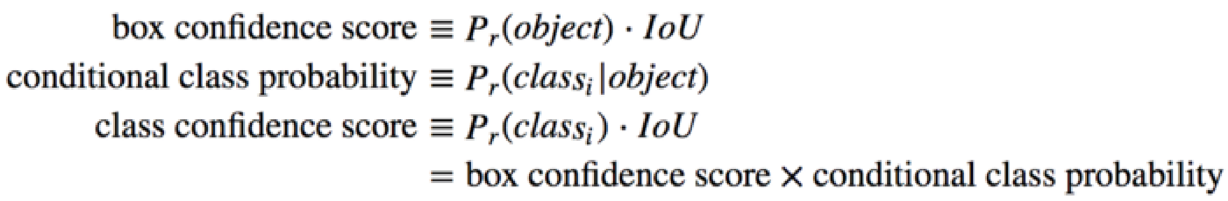
1 Yolo v1 – You Only Look Once

[발췌 자료] [PR12] You Only Look Once (YOLO): Unified Real-Time Object Detection, Taegyun Jeon

Unified Detection

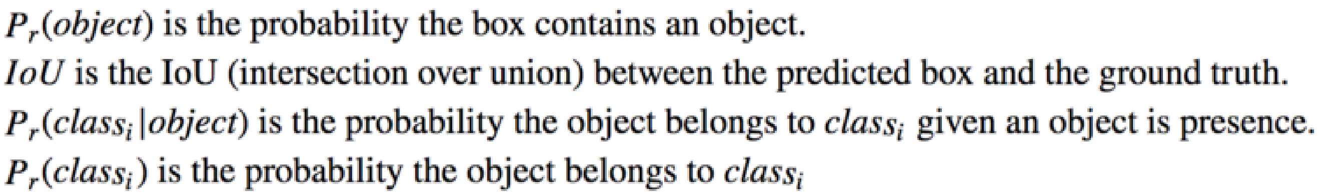


class confidence score = box confidence score × conditional class probability



$$\begin{aligned}
\text{box confidence score} &\equiv P_r(object) \cdot IoU \\
\text{conditional class probability} &\equiv P_r(class_i|object) \\
\text{class confidence score} &\equiv P_r(class_i) \cdot IoU \\
&= \text{box confidence score} \times \text{conditional class probability}
\end{aligned}$$

where



$P_r(object)$ is the probability the box contains an object.

IoU is the IoU (intersection over union) between the predicted box and the ground truth.

$P_r(class_i|object)$ is the probability the object belongs to $class_i$ given an object is presence.

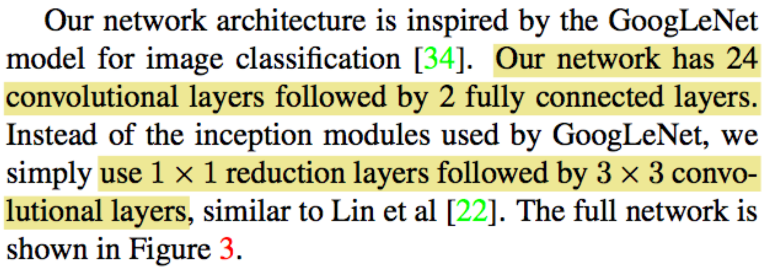
$P_r(class_i)$ is the probability the object belongs to $class_i$

1 Yolo v1 – You Only Look Once

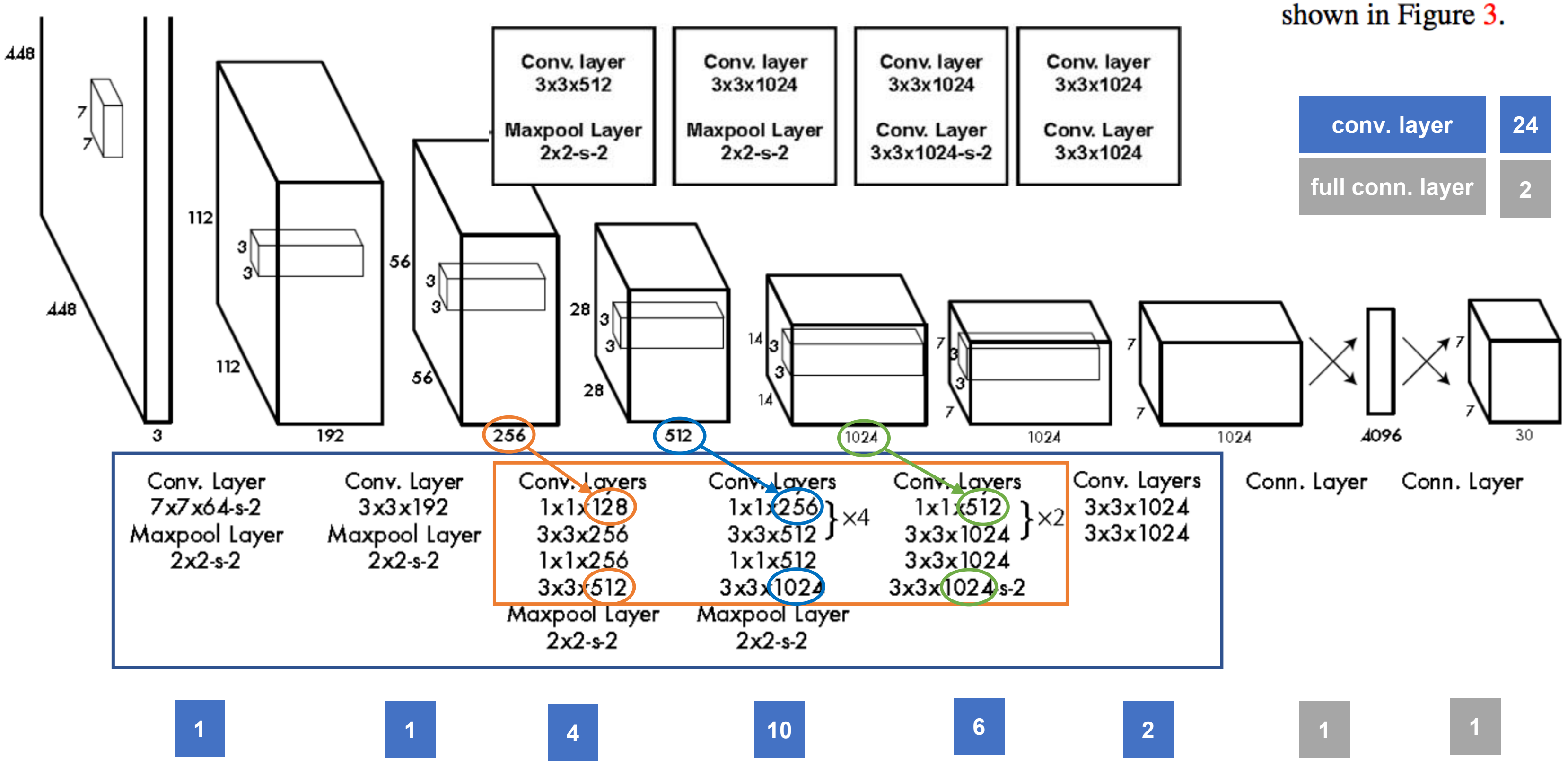
[발췌 자료] [PR12] You Only Look Once (YOLO): Unified Real-Time Object Detection, Taegyun Jeon

Network Design : Yolo

- Modified GoogLeNet
- 1x1 reduction layer ("Network in Network")



Our network architecture is inspired by the GoogLeNet model for image classification [34]. Our network has 24 convolutional layers followed by 2 fully connected layers. Instead of the inception modules used by GoogLeNet, we simply use 1 × 1 reduction layers followed by 3 × 3 convolutional layers, similar to Lin et al [22]. The full network is shown in Figure 3.


448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. layer 3x3x512 Maxpool Layer 2x2-s-2
Conv. layer 3x3x1024 Maxpool Layer 2x2-s-2
Conv. layer 3x3x1024 Conv. Layer 3x3x1024-s-2
Conv. layer 3x3x1024 Conv. Layer 3x3x1024
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 }×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 }×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer
1
1
4
10
6
2
1
1
conv. layer 24
full conn. layer 2

The Architecture. Our detection network has **24 convolutional layers** followed by **2 fully connected layers**.

Alternating **1x1 convolutional layers reduce the features space** from preceding layers.

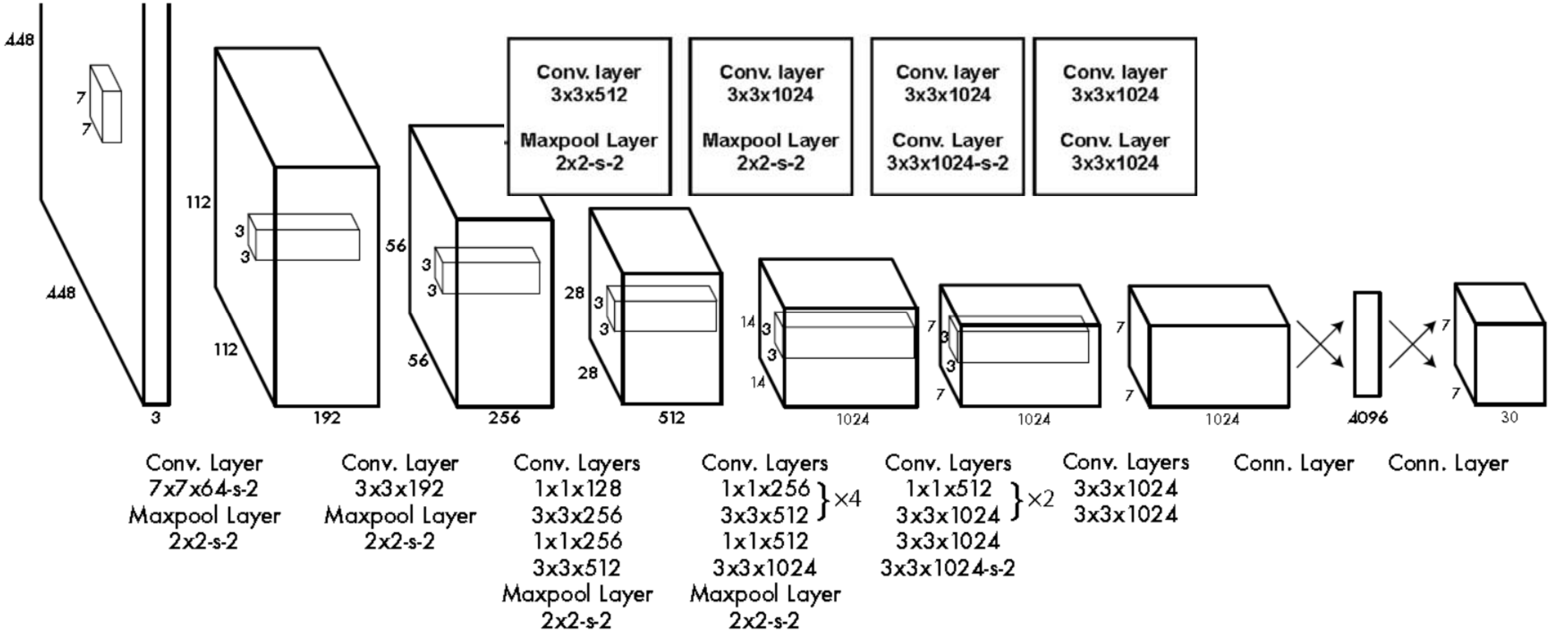
We **pretrain the convolutional layers on the ImageNet classification task at half the resolution (224x224 input image) and then double the resolution for detection.**

1 Yolo v1 – You Only Look Once

[발췌 자료] [PR12] You Only Look Once (YOLO): Unified Real-Time Object Detection, Taegyun Jeon

Network Design : Yolo-tiny (9 Conv. layers)

Yolo



Conv. layer 3x3x512
Maxpool Layer 2x2-s-2

Conv. layer 3x3x1024
Maxpool Layer 2x2-s-2

Conv. layer 3x3x1024
Conv. Layer 3x3x1024-s-2

Conv. layer 3x3x1024
Conv. Layer 3x3x1024

Conv. Layer
7x7x64-s-2
Maxpool Layer
2x2-s-2

Conv. Layer
3x3x192
Maxpool Layer
2x2-s-2

Conv. Layers
1x1x128
3x3x256
1x1x256
3x3x512
Maxpool Layer
2x2-s-2

Conv. Layers
1x1x256 }×4
3x3x512
1x1x512
3x3x1024
Maxpool Layer
2x2-s-2

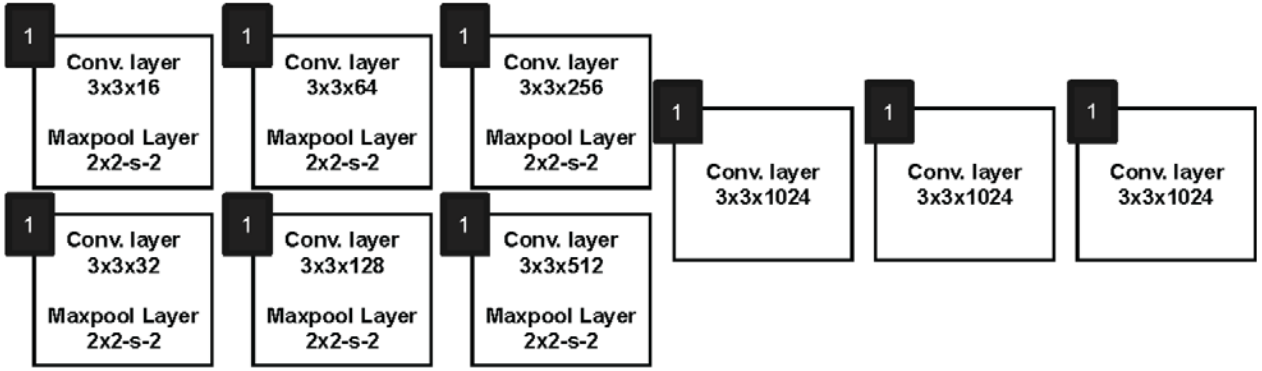
Conv. Layers
1x1x512 }×2
3x3x1024
3x3x1024
3x3x1024-s-2

Conv. Layers
3x3x1024
3x3x1024

Conn. Layer

Conn. Layer

Yolo - tiny



Conv. layer 3x3x16
Maxpool Layer 2x2-s-2

Conv. layer 3x3x32
Maxpool Layer 2x2-s-2

Conv. layer 3x3x64
Maxpool Layer 2x2-s-2

Conv. layer 3x3x128
Maxpool Layer 2x2-s-2

Conv. layer 3x3x256
Maxpool Layer 2x2-s-2

Conv. layer 3x3x512
Maxpool Layer 2x2-s-2

Conv. layer 3x3x1024

Conv. layer 3x3x1024

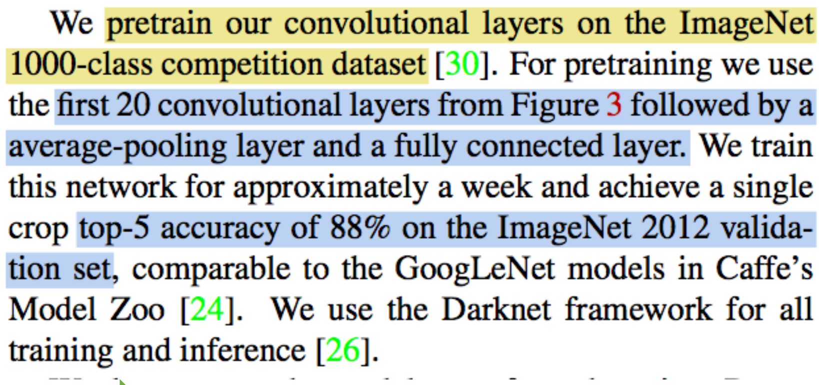
Conv. layer 3x3x1024

conv. layer	9
full conn. layer	2

1 Yolo v1 – You Only Look Once

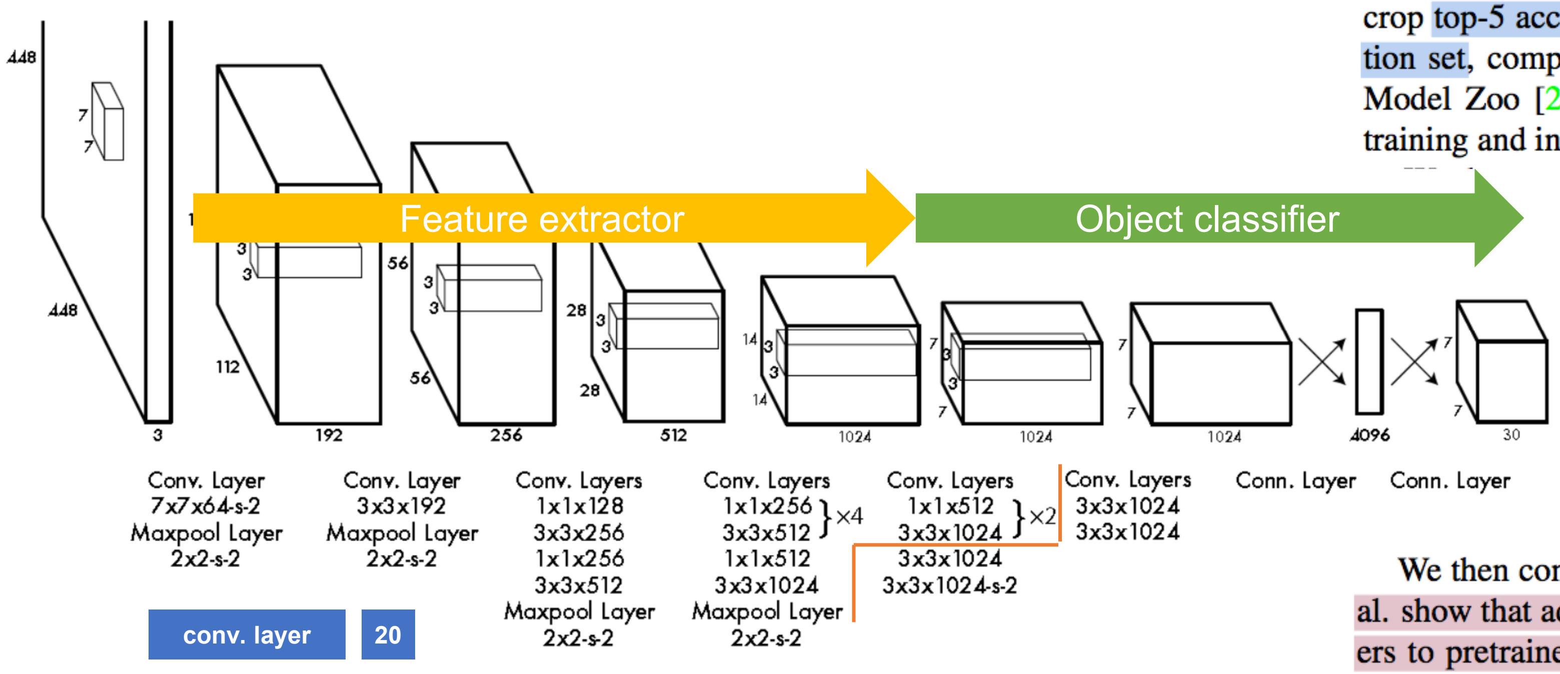
[발췌 자료] [PR12] You Only Look Once (YOLO): Unified Real-Time Object Detection, Taegyun Jeon

Training



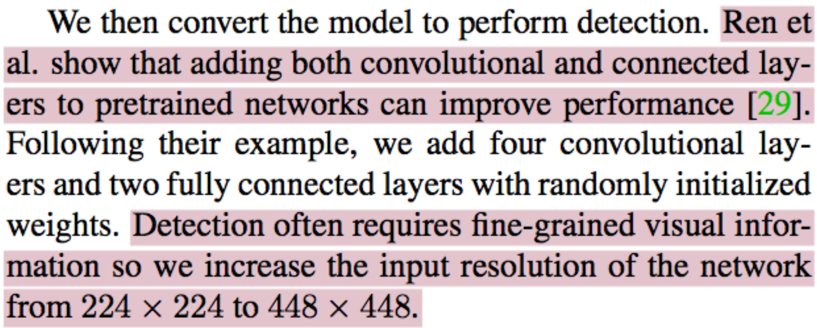
We pretrain our convolutional layers on the ImageNet 1000-class competition dataset [30]. For pretraining we use the first 20 convolutional layers from Figure 3 followed by a average-pooling layer and a fully connected layer. We train this network for approximately a week and achieve a single crop top-5 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].

1) Pretrain with ImageNet 1000-class competition dataset.



conv. layer 4

full conn. layer 2



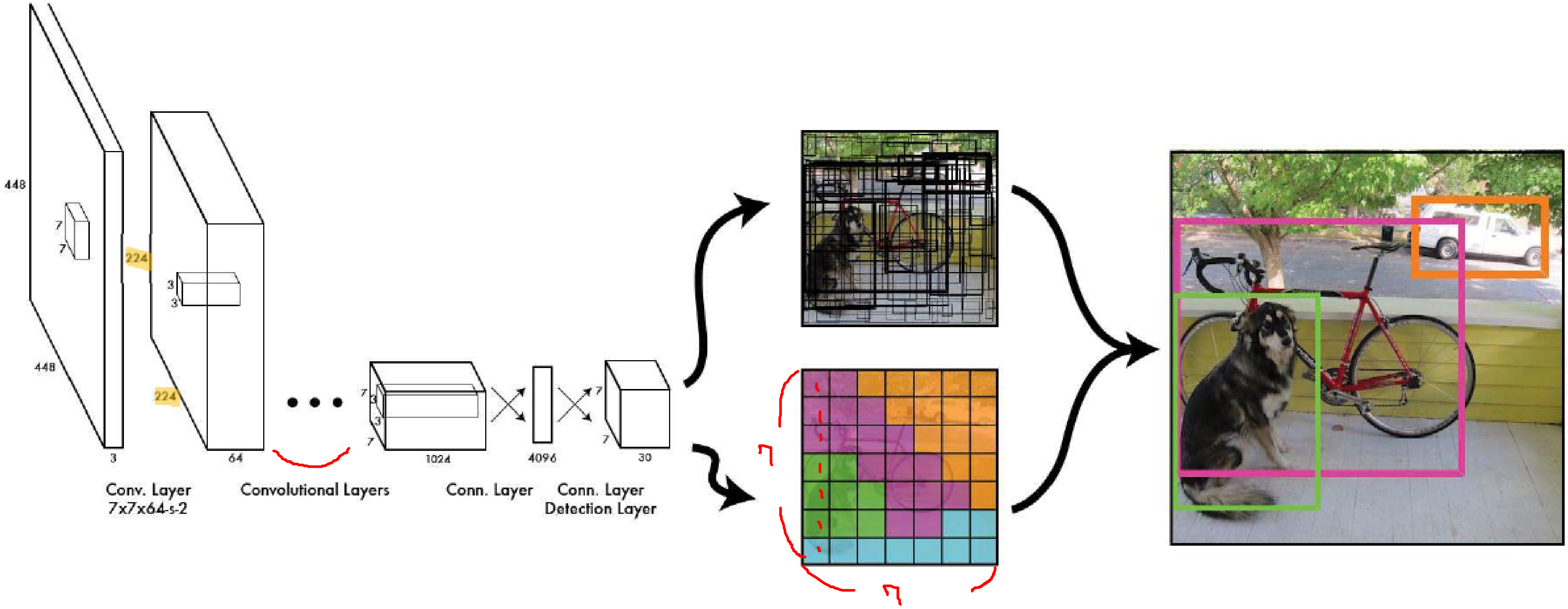
We then convert the model to perform detection. Ren et al. show that adding both convolutional and connected layers to pretrained networks can improve performance [29]. Following their example, we add four convolutional layers and two fully connected layers with randomly initialized weights. Detection often requires fine-grained visual information so we increase the input resolution of the network from 224 × 224 to 448 × 448.

2) "Network on Convolutional Feature Maps"

Increased input resolution (224 x 224) → (448 x 448)

1 Yolo v1 – You Only Look Once

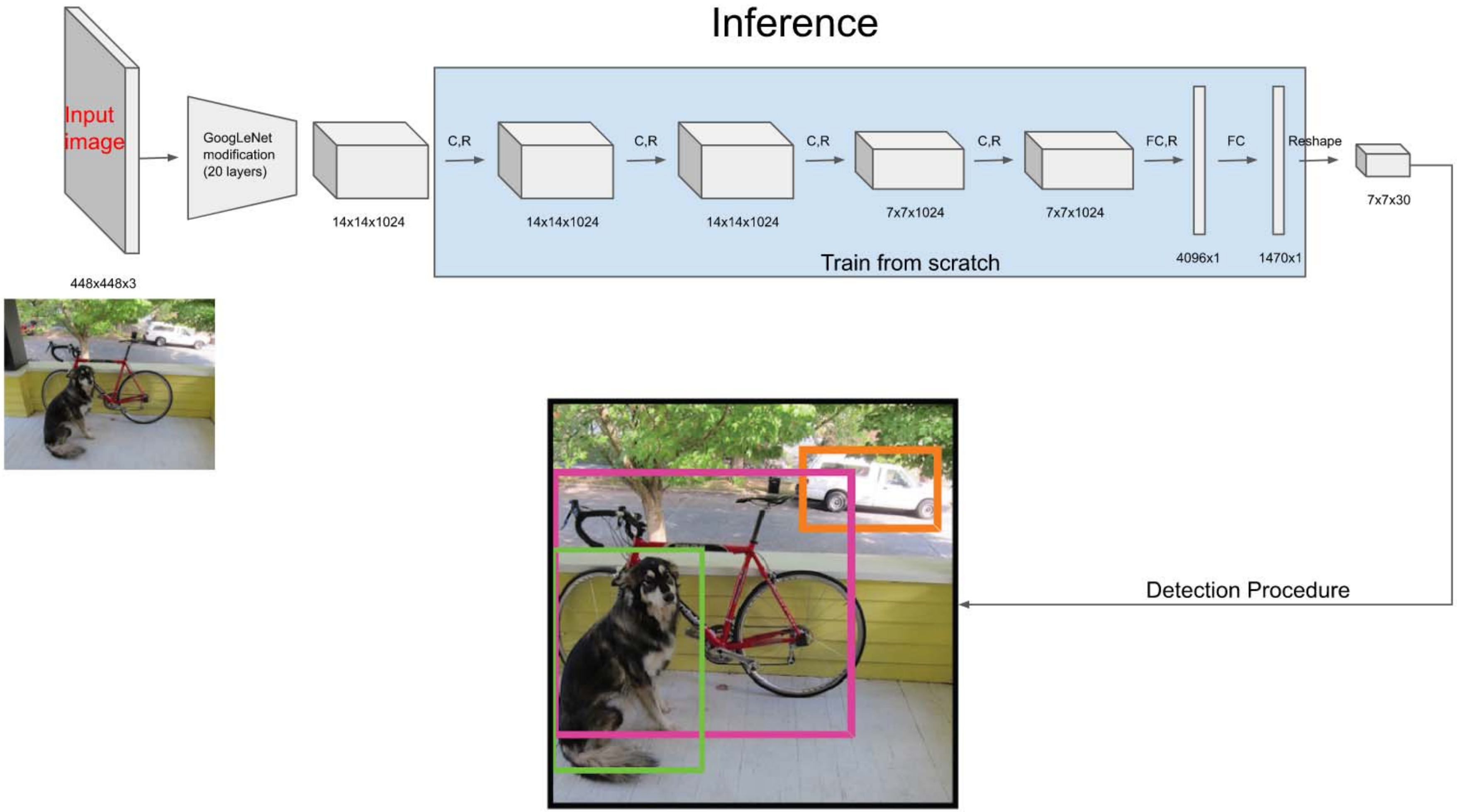
[발췌 자료] [PR12] You Only Look Once (YOLO): Unified Real-Time Object Detection, Taegyun Jeon



1

Yolo v1 – You Only Look Once

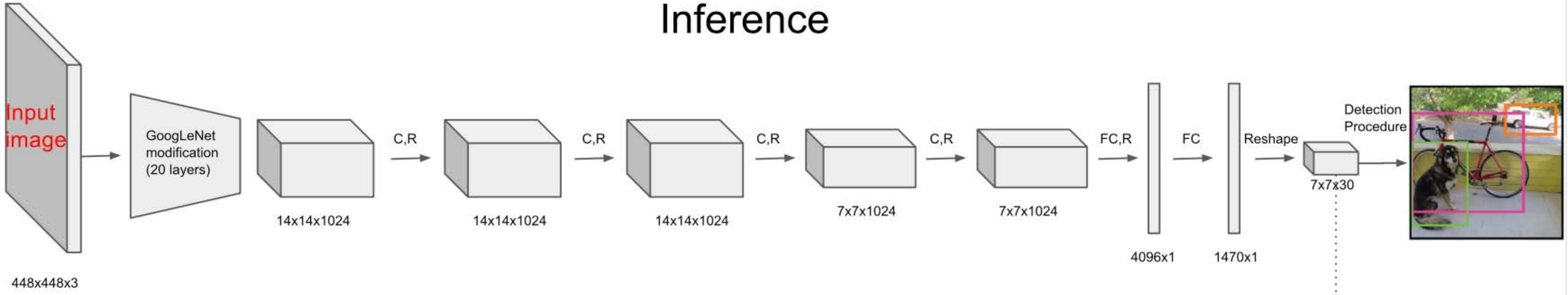
[발췌 자료] [PR12] You Only Look Once (YOLO): Unified Real-Time Object Detection, Taegyun Jeon



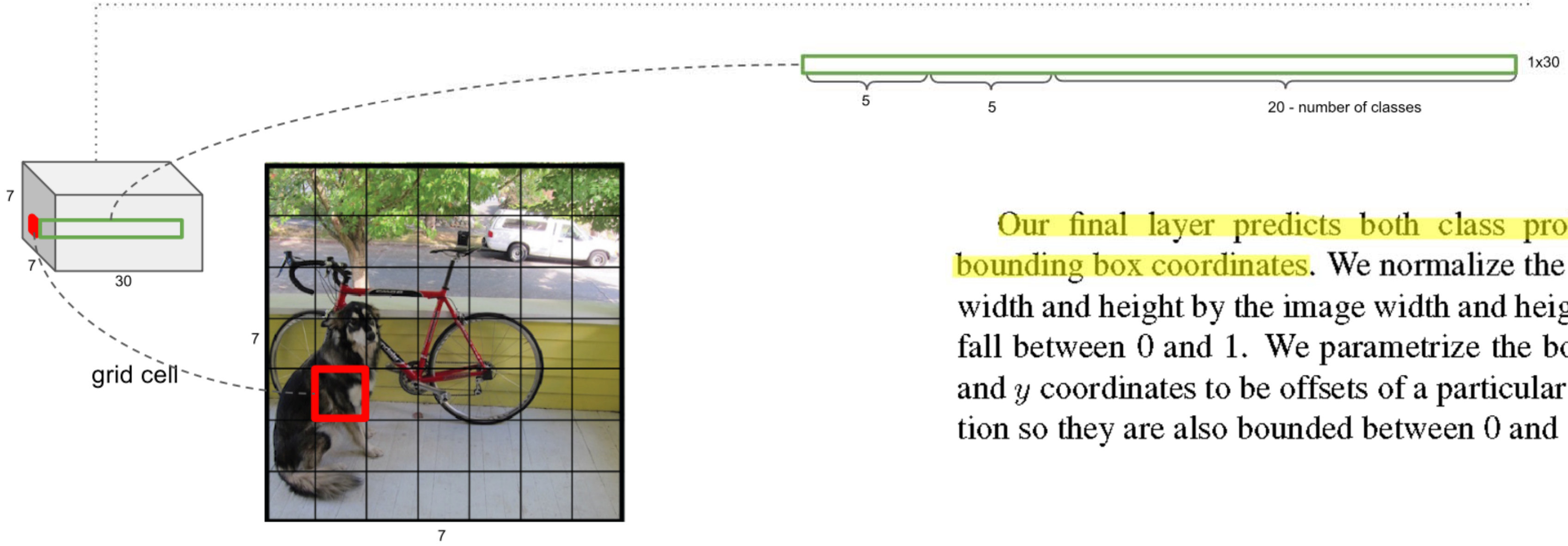
1 Yolo v1 – You Only Look Once

[발췌 자료] [PR12] You Only Look Once (YOLO): Unified Real-Time Object Detection, Taegyun Jeon

Inference



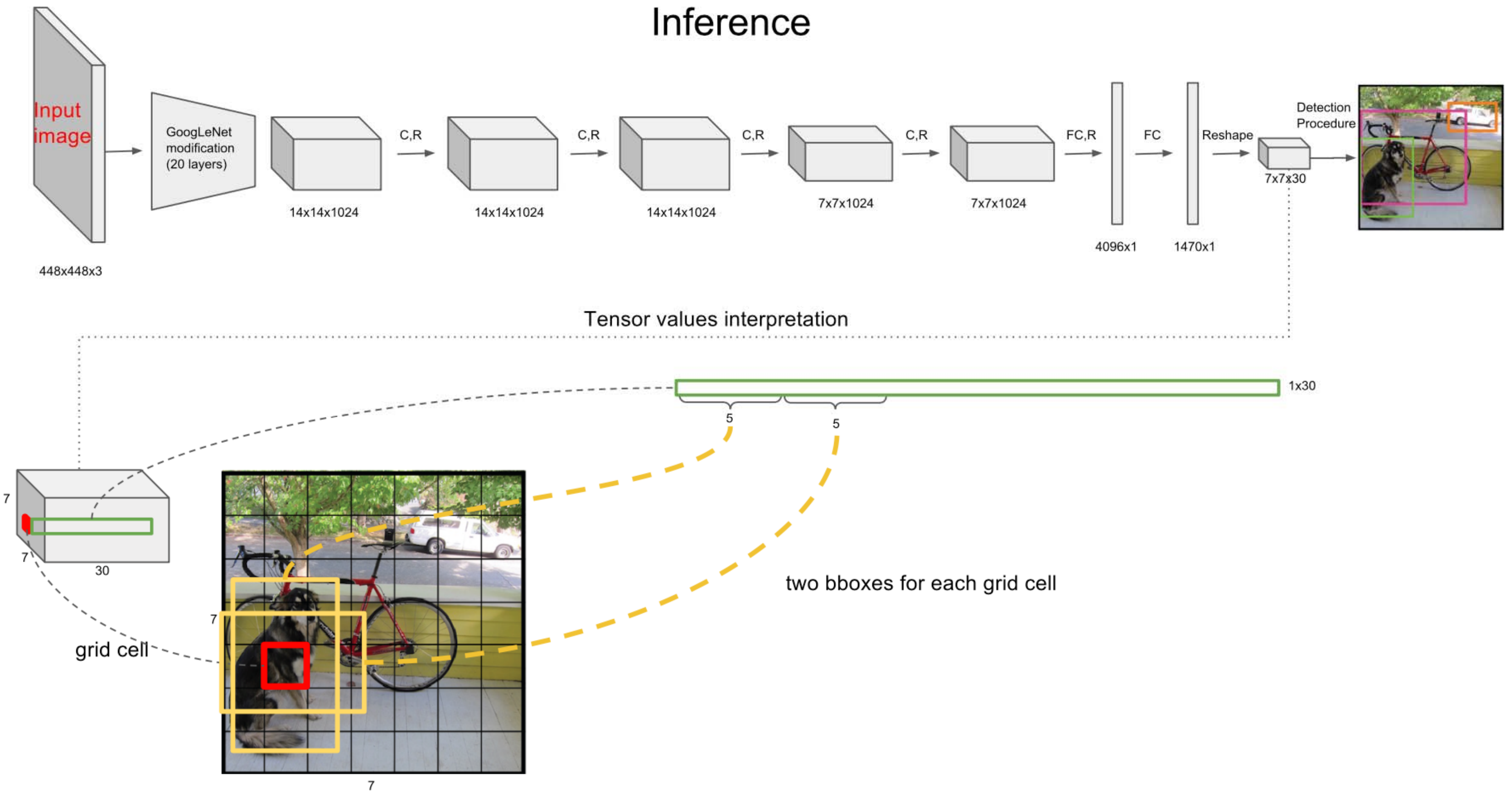
Tensor values interpretation



Our final layer predicts both class probabilities and bounding box coordinates. We normalize the bounding box width and height by the image width and height so that they fall between 0 and 1. We parametrize the bounding box x and y coordinates to be offsets of a particular grid cell location so they are also bounded between 0 and 1.

1 Yolo v1 – You Only Look Once

[발췌 자료] [PR12] You Only Look Once (YOLO): Unified Real-Time Object Detection, Taegyun Jeon

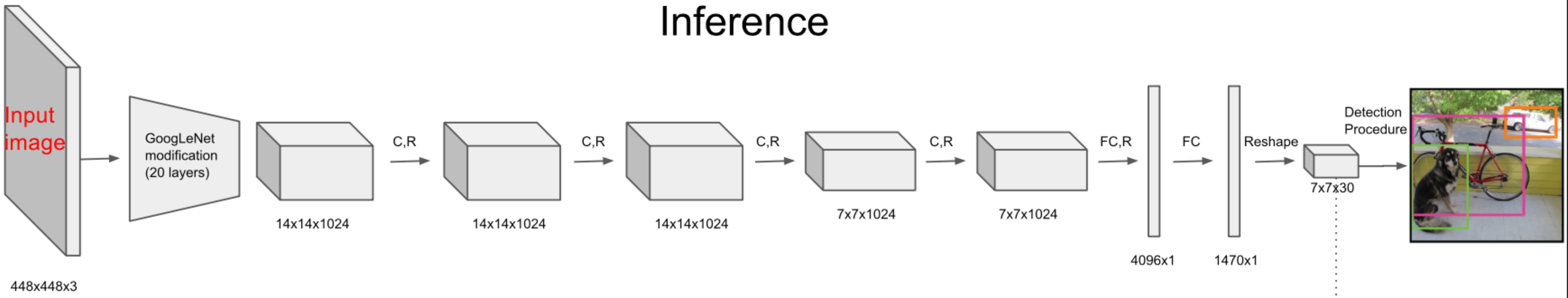


deepsystems.io

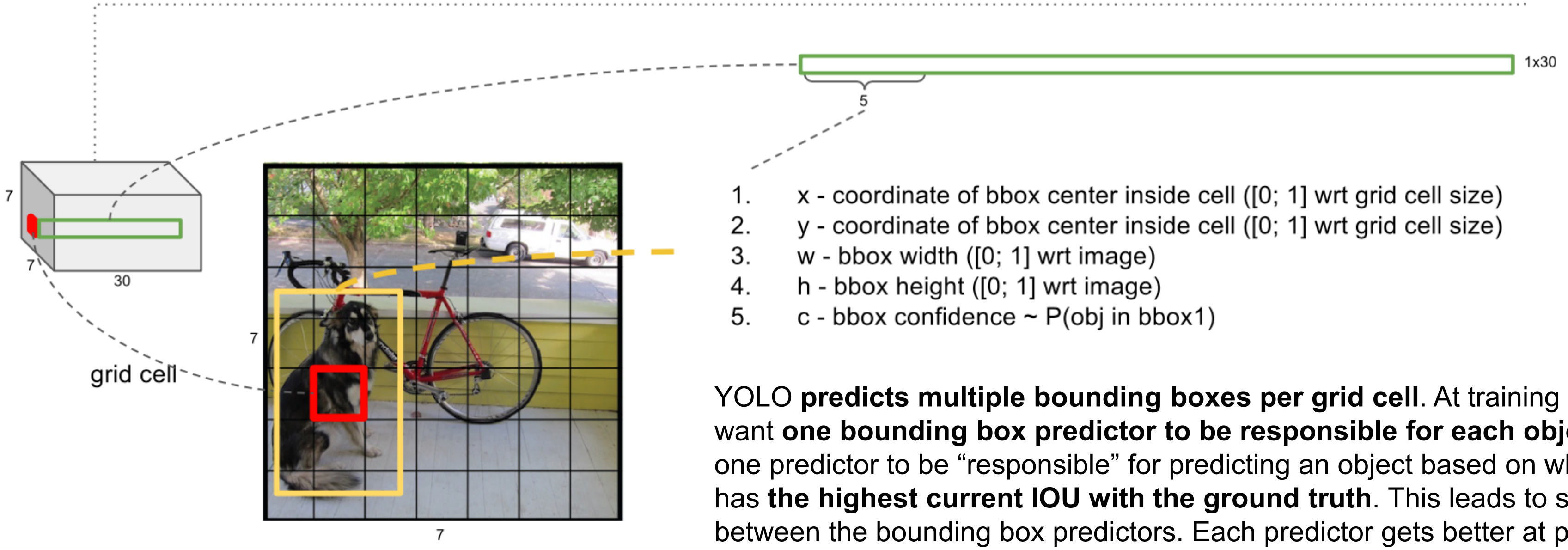
1 Yolo v1 – You Only Look Once

[발췌 자료] [PR12] You Only Look Once (YOLO): Unified Real-Time Object Detection, Taegyun Jeon

Inference



Tensor values interpretation



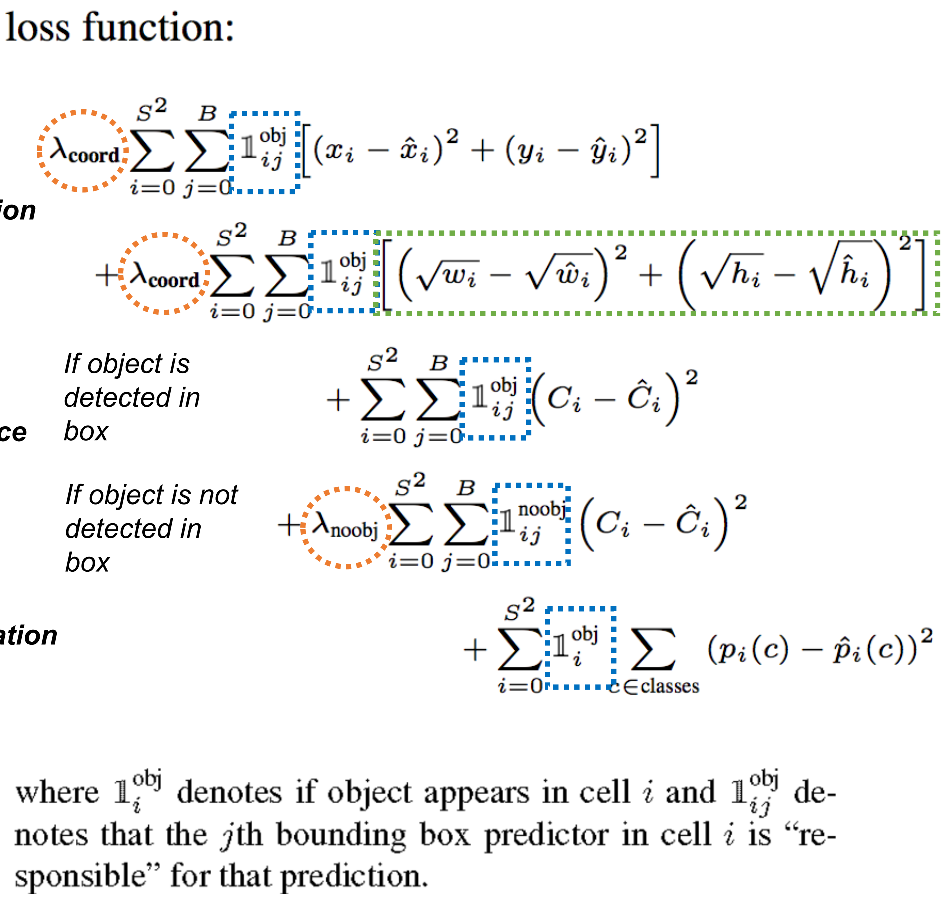
1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence ~ P(obj in bbox1)

YOLO **predicts multiple bounding boxes per grid cell**. At training time we only want **one bounding box predictor to be responsible for each object**. We assign one predictor to be “responsible” for predicting an object based on which prediction has **the highest current IOU with the ground truth**. This leads to specialization between the bounding box predictors. Each predictor gets better at predicting certain sizes, aspect ratios, or classes of object, improving overall recall.

Yolo v1 – You Only Look Once

[발췌 자료] [PR12] You Only Look Once (YOLO): Unified Real-Time Object Detection, Taegyun Jeon

Loss Function (Sum-squared error)



loss function:

Localization loss

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

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

Confidence loss

If object is detected in box

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

If object is not detected in box

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

Classification loss

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

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.

i : Object가 존재하는 Grid Cell

j : Predictor Bounding Box

$\mathbb{1}_{i,j}^{obj}$: Object가 존재하는 경우 grid cell의 predictor bounding box

$\mathbb{1}_{i,j}^{nobj}$: Object가 존재하지 않는 경우 grid cell의 predictor bounding box

$\mathbb{1}_{i}^{obj}$: Object가 존재하는 경우 grid cell

이미지 대부분에는 object가 없을 것이고 confidence는 전부 0으로 수렴하려고 할 것임. 그로 인해 발생되는 gradien가 너무 커지는 현상을 막아주기 위해서 추가 paramete를 사용함.

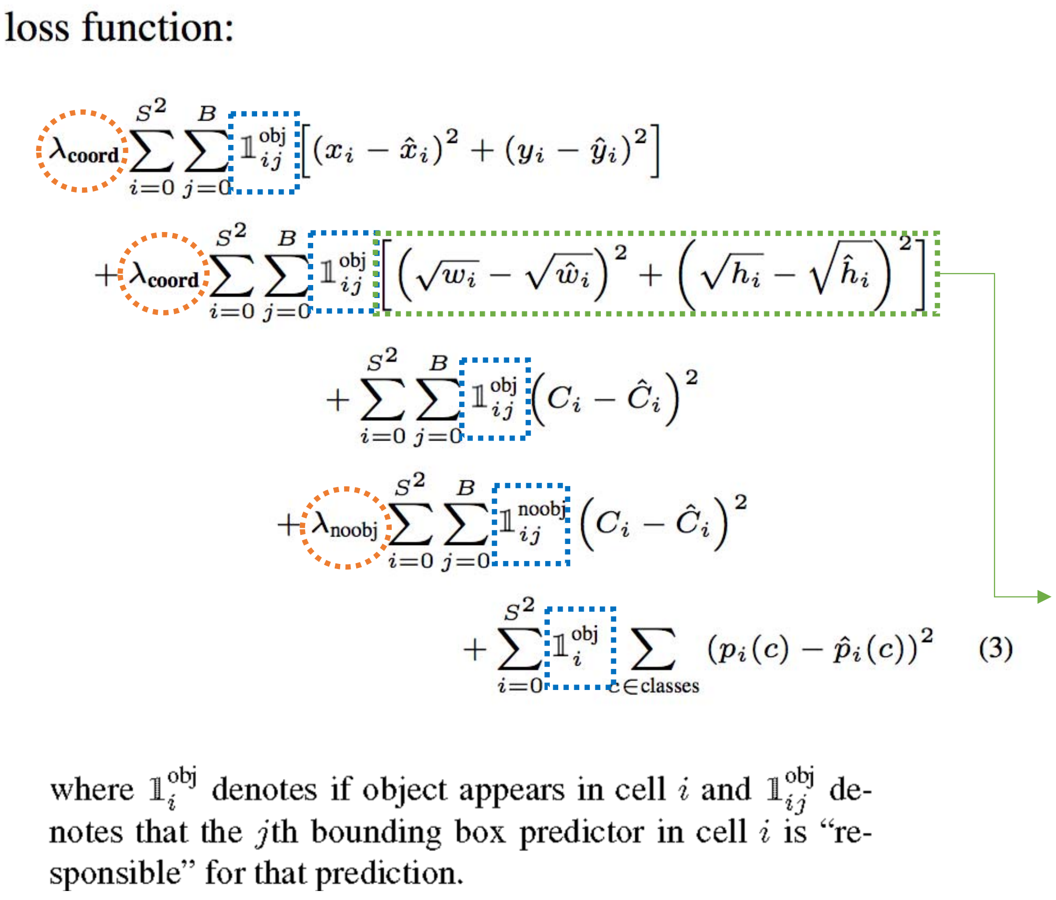
- λ_{coord} : x, y, w, h loss의 균형을 위한 parameter. (defalut : 5)
- λ_{nobjc} : object loss의 균형을 위한 parameter. (defalut : 0.5)
- x, y의 loss와 w, h의 loss를 구함 (가로, 세로의 제곱근을 예측함)
- confidence score의 loss를 구함 (C_i =1)
- confidence score의 loss를 구함 (C_i =0)
- conditional class probability의 loss를 구함

1

Yolo v1 – You Only Look Once

[발췌 자료] [PR12] You Only Look Once (YOLO): Unified Real-Time Object Detection, Taegyun Jeon

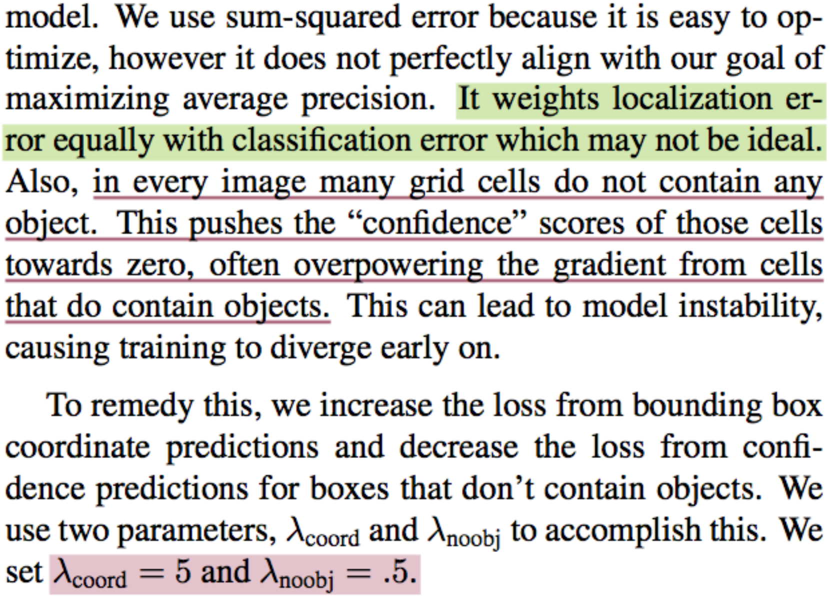
Loss Function (Sum-squared error)



loss function:

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

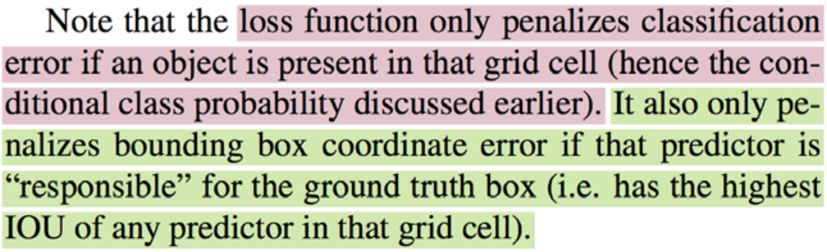
where $\mathbb{1}_i^{\text{obj}}$ denotes if object appears in cell i and $\mathbb{1}_{ij}^{\text{obj}}$ denotes that the jth bounding box predictor in cell i is "responsible" for that prediction.



model. We use sum-squared error because it is easy to optimize, however it does not perfectly align with our goal of maximizing average precision. It weights localization error equally with classification error which may not be ideal. Also, in every image many grid cells do not contain any object. This pushes the "confidence" scores of those cells towards zero, often overpowering the gradient from cells that do contain objects. This can lead to model instability, causing training to diverge early on.

To remedy this, we increase the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don't contain objects. We use two parameters, λ_{coord} and λ_{noobj} to accomplish this. We set $\lambda_{\text{coord}} = 5$ and $\lambda_{\text{noobj}} = .5$.

Sum-squared error also equally weights errors in large boxes and small boxes. Our error metric should reflect that small deviations in large boxes matter less than in small boxes. **To partially address this we predict the square root of the bounding box width and height instead of the width and height directly.**

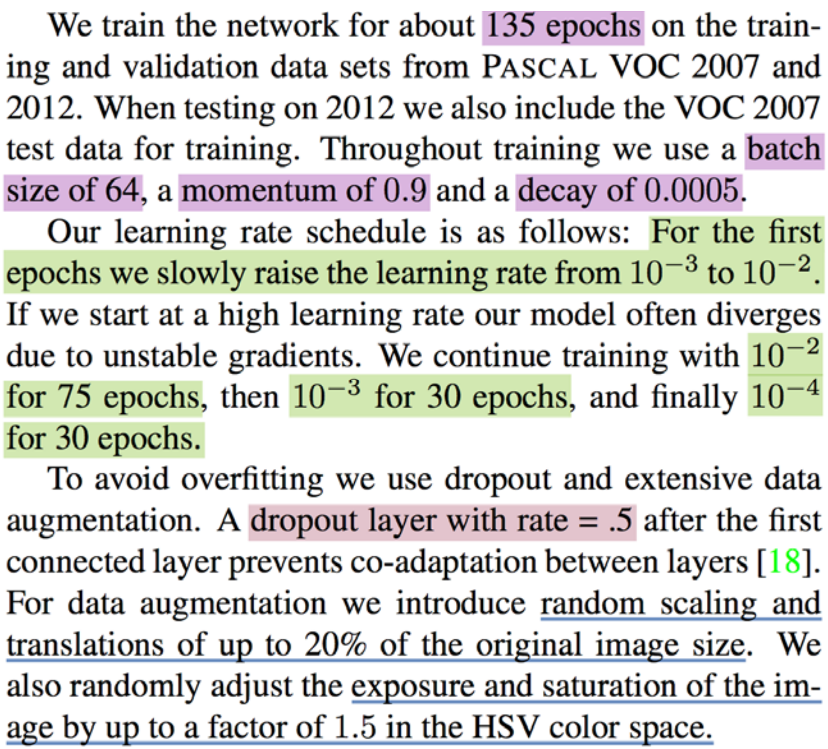


Note that the loss function only penalizes classification error if an object is present in that grid cell (hence the conditional class probability discussed earlier). It also only penalizes bounding box coordinate error if that predictor is "responsible" for the ground truth box (i.e. has the highest IOU of any predictor in that grid cell).

1 Yolo v1 – You Only Look Once

[발췌 자료] [PR12] You Only Look Once (YOLO): Unified Real-Time Object Detection, Taegyun Jeon

Train Strategy

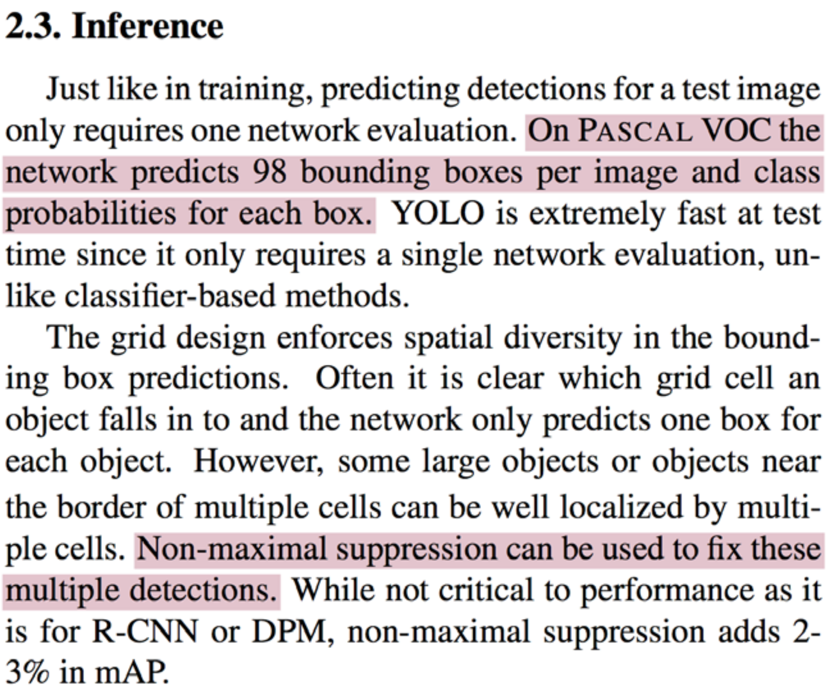


We train the network for about 135 epochs on the training and validation data sets from PASCAL VOC 2007 and 2012. When testing on 2012 we also include the VOC 2007 test data for training. Throughout training we use a batch size of 64, a momentum of 0.9 and a decay of 0.0005.

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.

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.

Inference



2.3. Inference

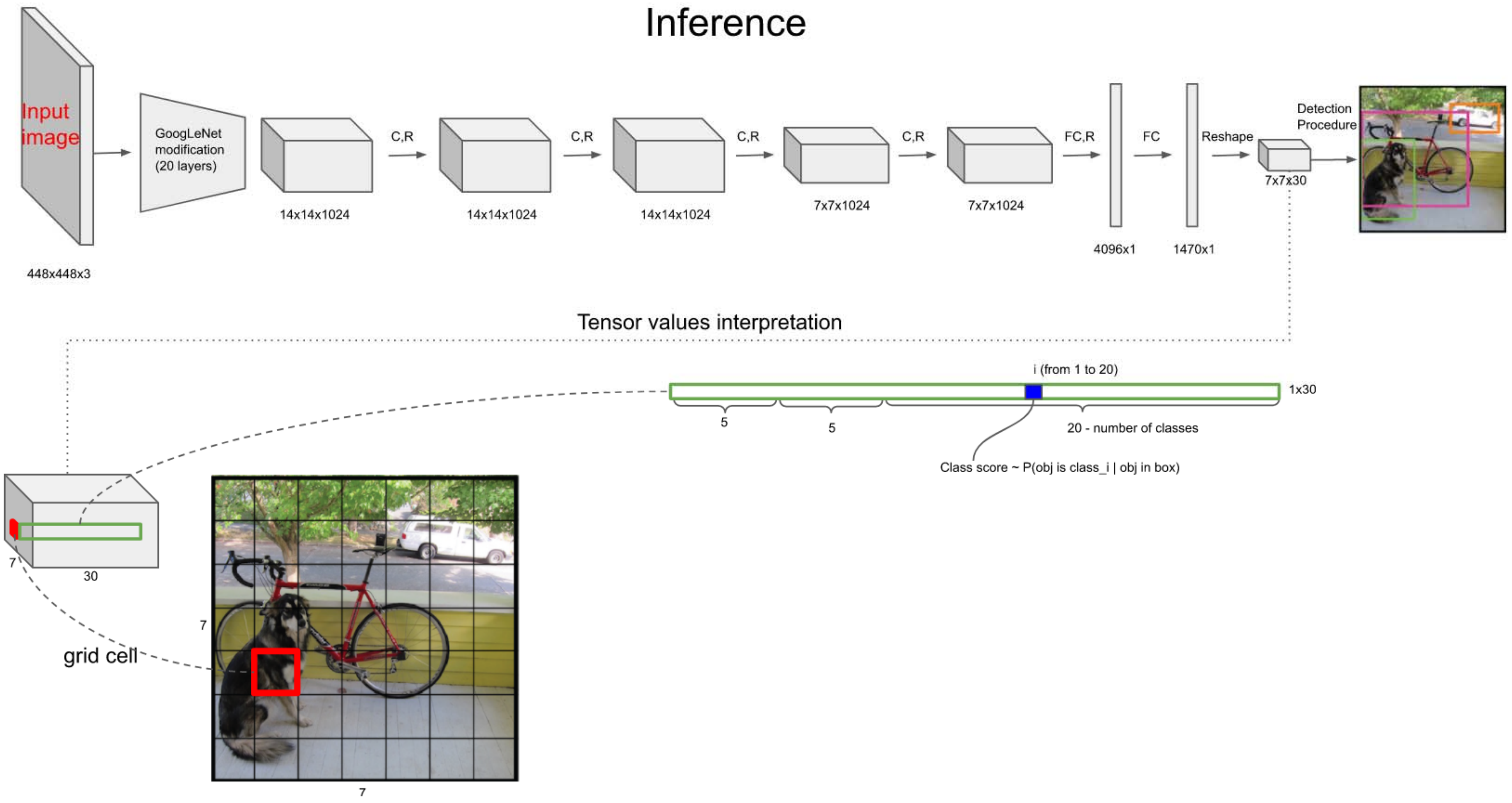
Just like in training, predicting detections for a test image only requires one network evaluation. On PASCAL VOC the network predicts 98 bounding boxes per image and class probabilities for each box. YOLO is extremely fast at test time since it only requires a single network evaluation, unlike classifier-based methods.

The grid design enforces spatial diversity in the bounding box predictions. Often it is clear which grid cell an object falls in to and the network only predicts one box for each object. However, some large objects or objects near the border of multiple cells can be well localized by multiple cells. Non-maximal suppression can be used to fix these multiple detections. While not critical to performance as it is for R-CNN or DPM, non-maximal suppression adds 2-3% in mAP.

Sum-squared error also equally weights errors in large boxes and small boxes. Our error metric should reflect that small deviations in large boxes matter less than in small boxes. **To partially address this we predict the square root of the bounding box width and height instead of the width and height directly.**

1 Yolo v1 – You Only Look Once

[발췌 자료]deepsystems.io deepsystems.io

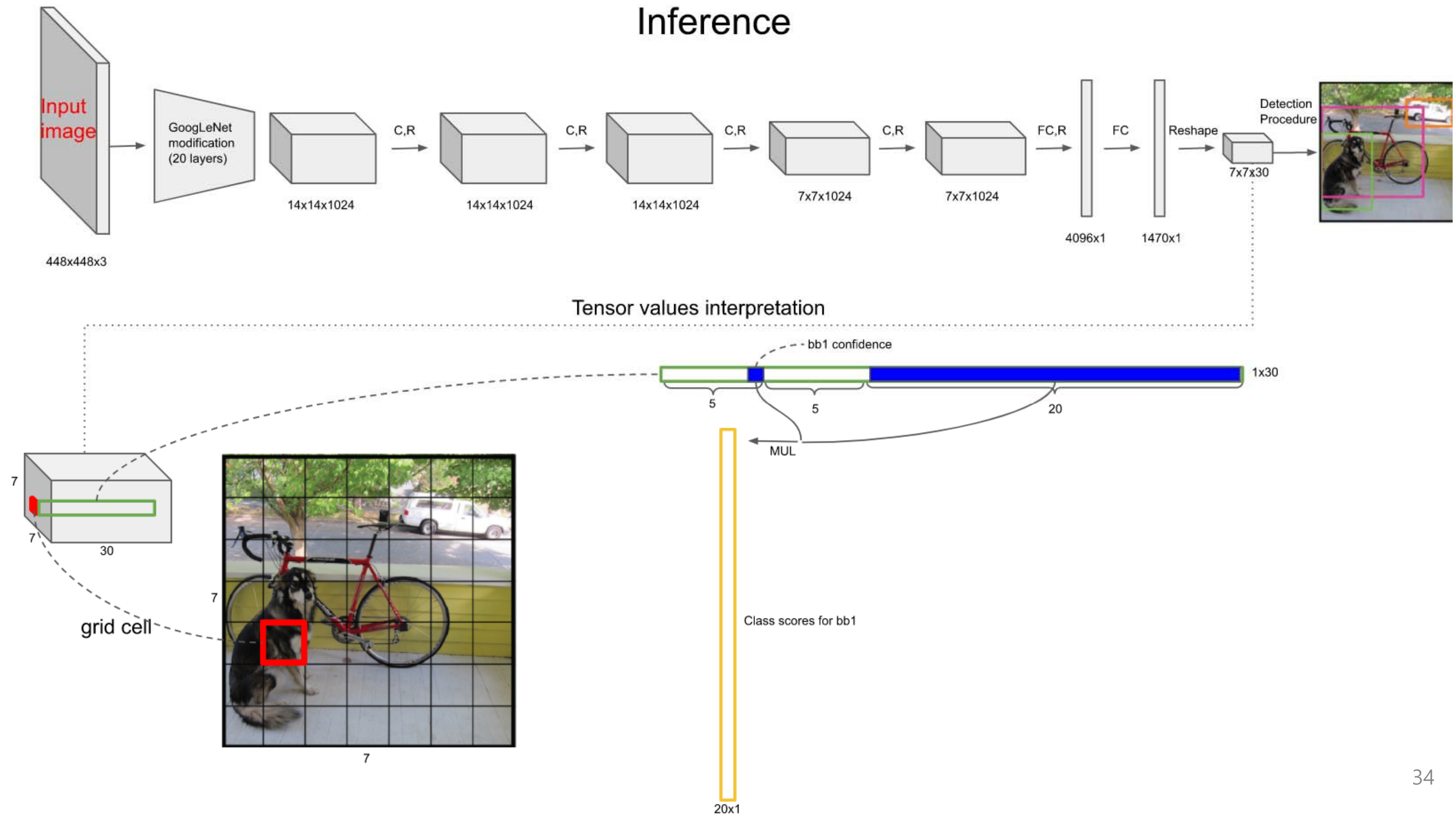


1

Yolo v1 – You Only Look Once

[발췌 자료]deepsystems.io

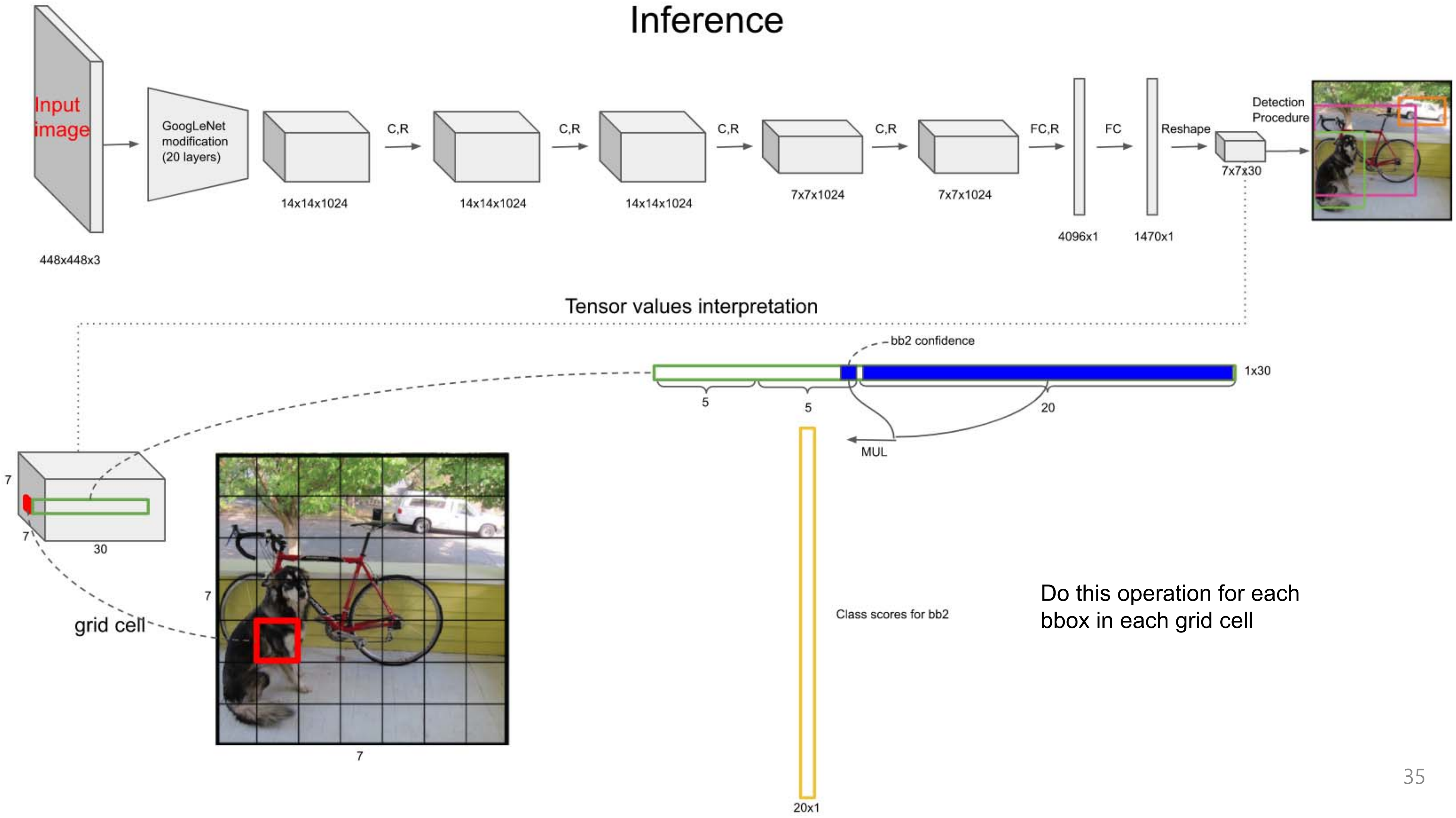
deepsystems.io



1 Yolo v1 – You Only Look Once

[발췌 자료]deepsystems.io

Inference



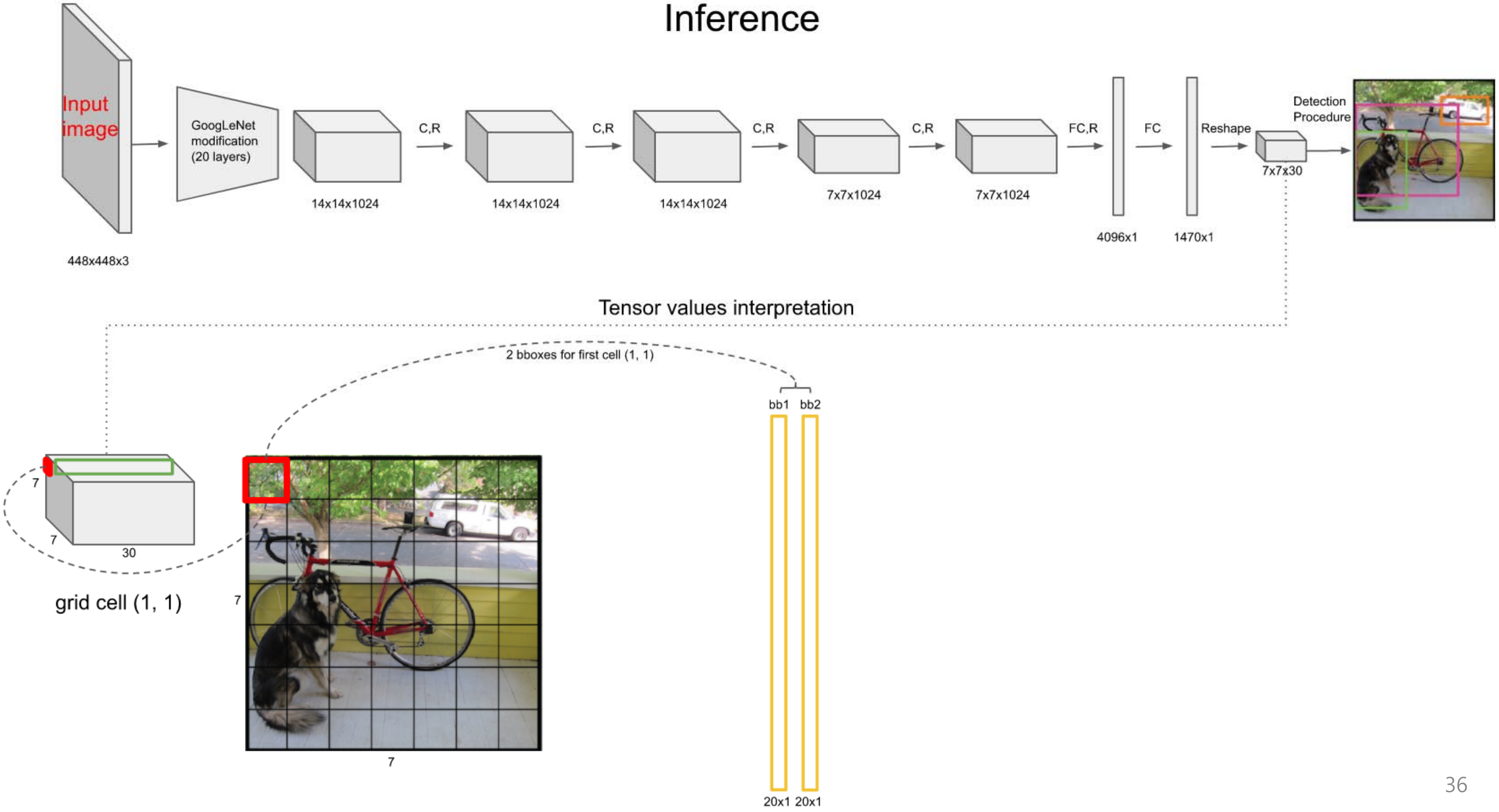
Do this operation for each bbox in each grid cell

1

Yolo v1 – You Only Look Once

[발췌 자료]deepsystems.io

deepsystems.io



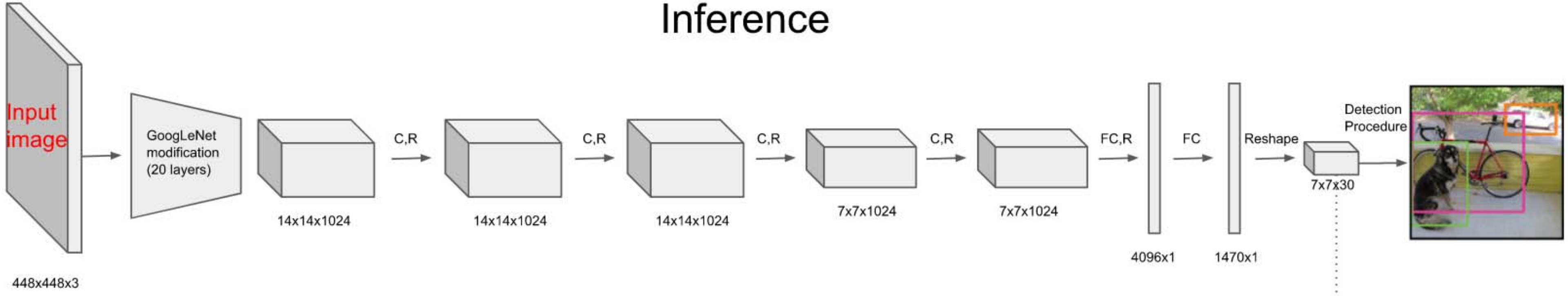
1

Yolo v1 – You Only Look Once

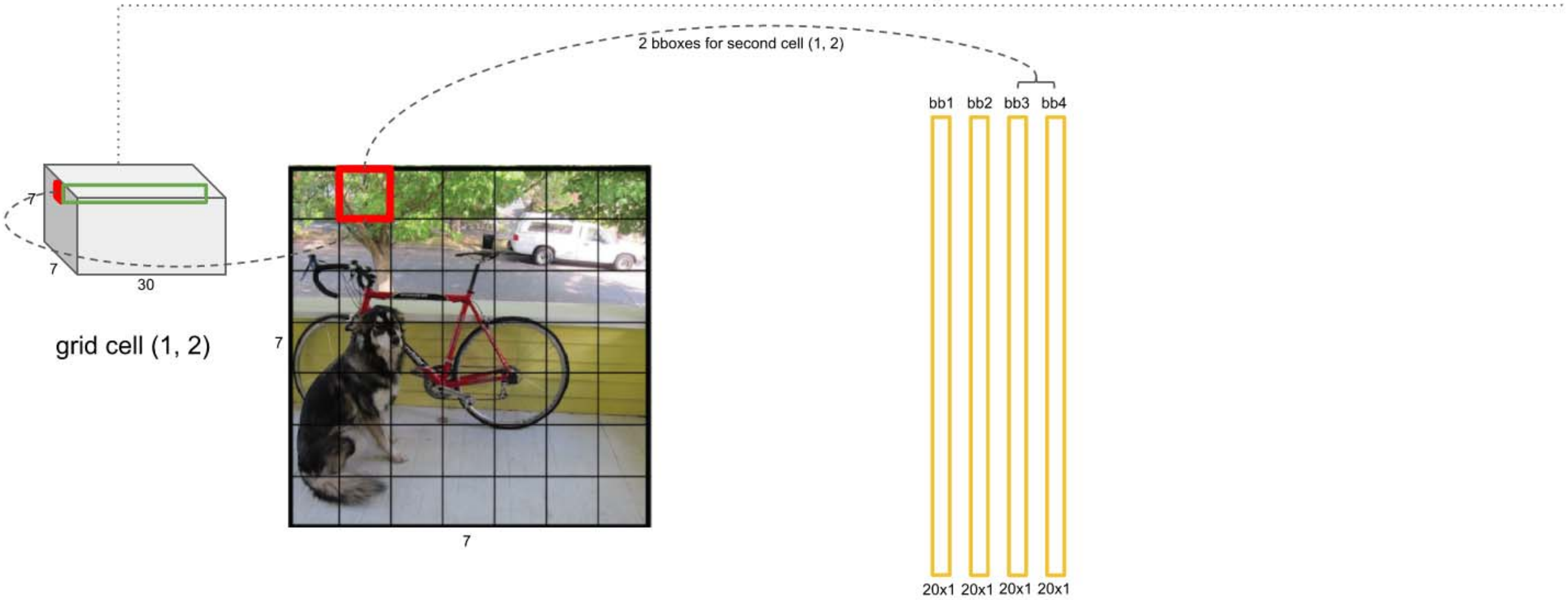
[발췌 자료]deepsystems.io

deepsystems.io

Inference



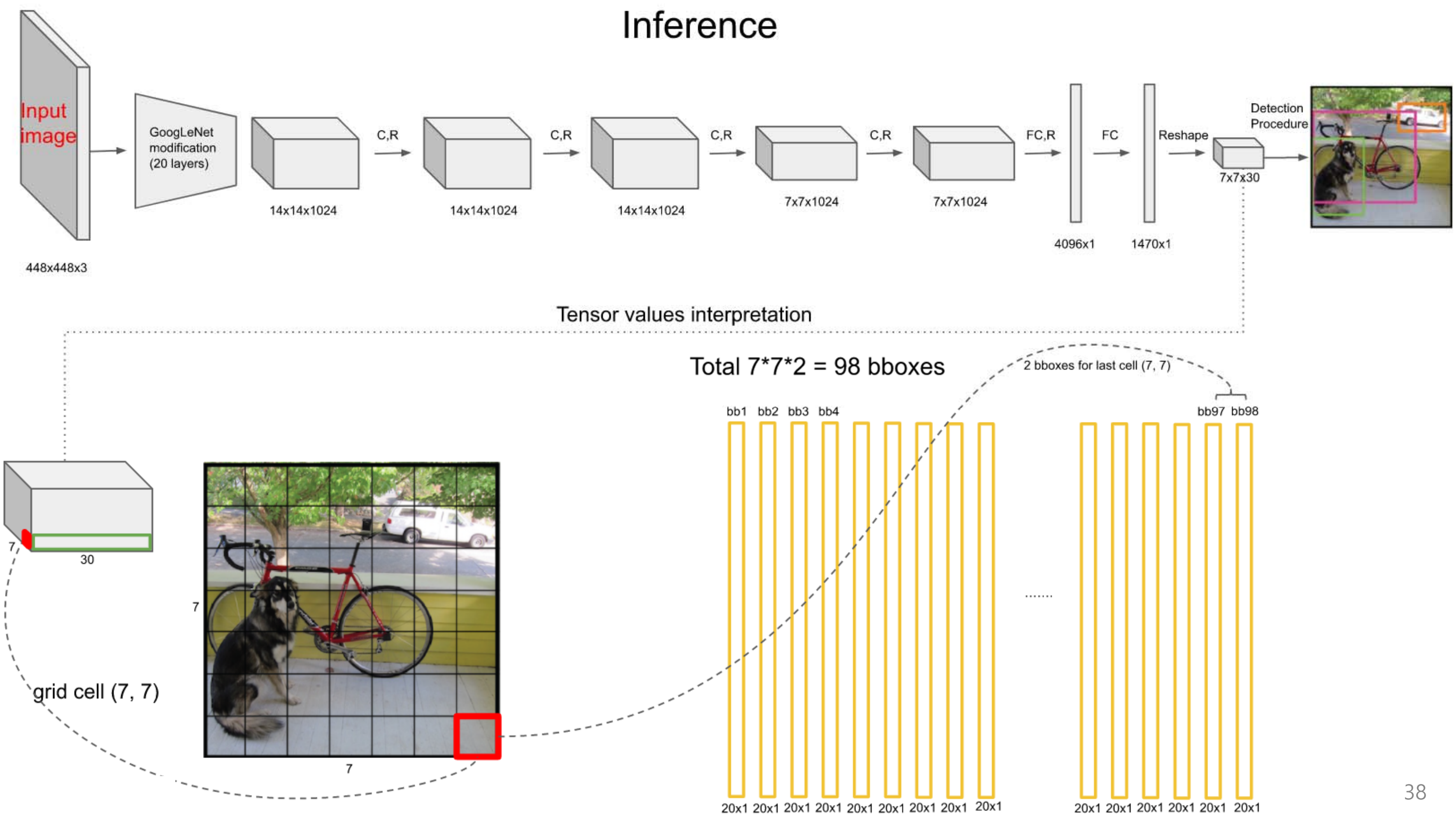
Tensor values interpretation



1

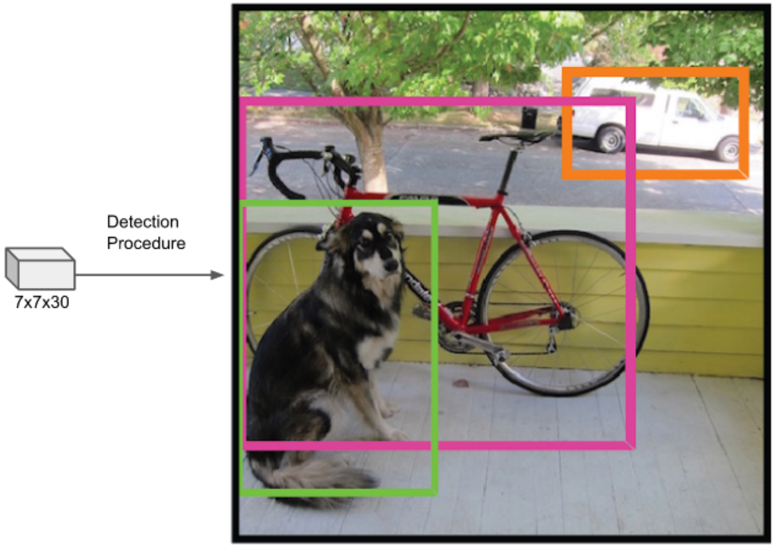
Yolo v1 – You Only Look Once

[발췌 자료]deepsystems.io



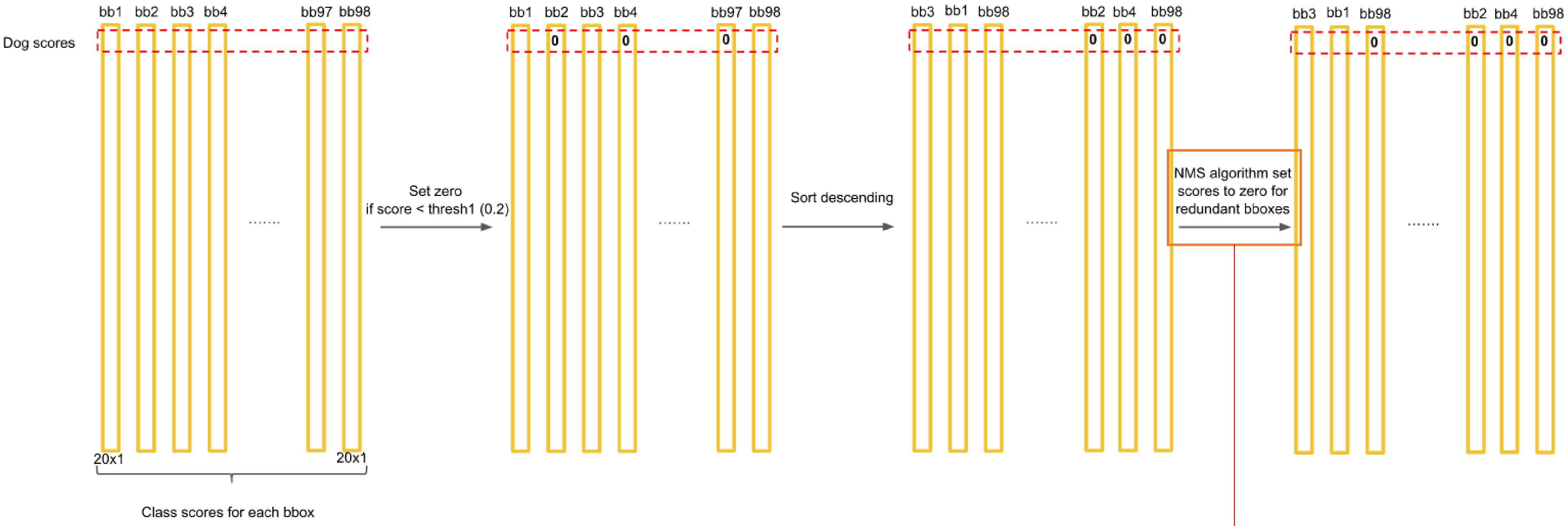
[발췌 자료]deepsystems.io

Look at detection procedure



1 Yolo v1 – You Only Look Once

[발췌 자료]deepsystems.io


bb1 bb2 bb3 bb4 bb97 bb98
Dog scores
....
20x1
20x1
Class scores for each bbox
Set zero
if score < thresh1 (0.2)
bb1 bb2 bb3 bb4 bb97 bb98
0 0 0
Sort descending
bb3 bb1 bb98 bb2 bb4 bb98
0 0 0
NMS algorithm set scores to zero for redundant bboxes
bb3 bb1 bb98 bb2 bb4 bb98
0 0 0 0

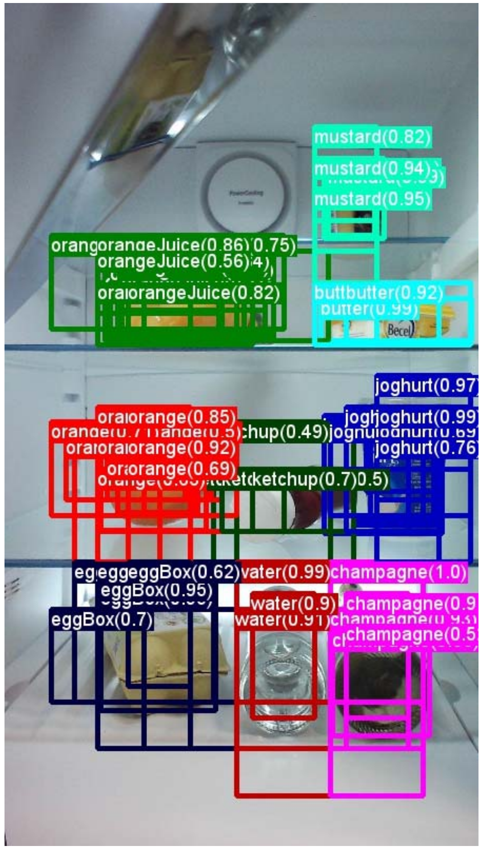
Get first class scores for each bbox

How it works

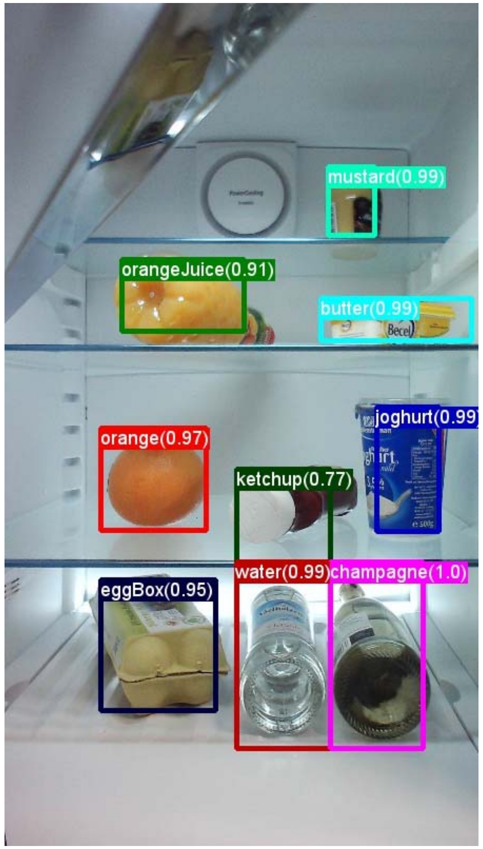
1 Yolo v1 – You Only Look Once

[발췌 자료] https://ctkim.tistory.com/98
https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c

NMS (Non-Maximum Suppression)



NMS 전



NMS 후

NMS(Non-maximum Suppression) : 일반적으로 Input image가 Object detection 알고리즘을 통과하면 Object에 bbox(Bounding box)가 그려지며 어떤 물체일 확률 값(Score)값을 가지게 된다. 이때 하나의 Object에 많은 bbox가 생긴다. 동일한 Object에 여러 개의 bbox가 있다면, 가장 score가 높은 bbox만 남기고 나머지를 제거하는 것임.

Input: A list of Proposal boxes B, corresponding confidence scores S and overlap threshold N.

Output: A list of filtered proposals D.

Algorithm:

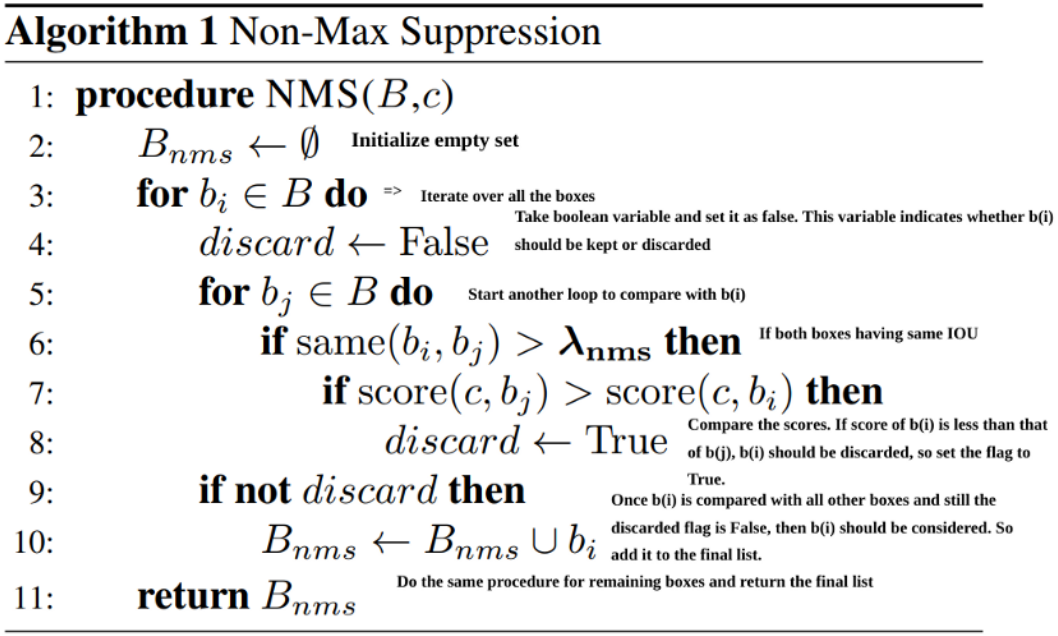
1) Select the proposal with highest confidence score, remove it from B and add it to the final proposal list D. (Initially D is empty).
2) Now compare this proposal with all the proposals — calculate the IOU (Intersection over Union) of this proposal with every other proposal. If the IOU is greater than the threshold N, remove that proposal from B.
3) Again take the proposal with the highest confidence from the remaining proposals in B and remove it from B and add it to D.
4) Once again calculate the IOU of this proposal with all the proposals in B and eliminate the boxes which have high IOU than threshold.
5) This process is repeated until there are no more proposals left in B.

1 Yolo v1 – You Only Look Once

[발췌 자료] https://ctkim.tistory.com/98
https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c

NMS (Non-Maximum Suppression)

- Pseudo code of NMS

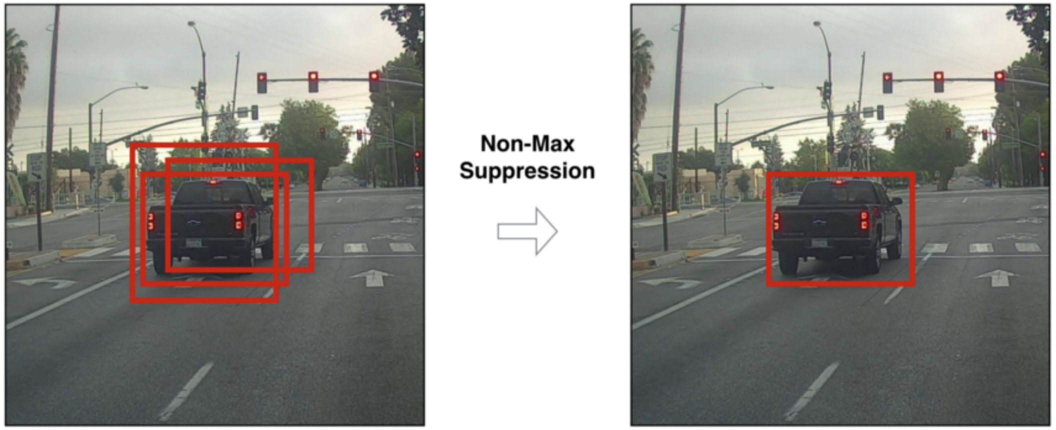


Algorithm 1 Non-Max Suppression

```
procedure NMS($B$,$c$)
    $B_{nms} \leftarrow \emptyset$    Initialize empty set
    for $b_i \in B$ do    => Iterate over all the boxes
        $discard \leftarrow$ False    Take boolean variable and set it as false. This variable indicates whether b(i) should be kept or discarded
        for $b_j \in B$ do    Start another loop to compare with b(i)
            if same$(b_i, b_j) > \lambda_{nms}$ then    If both boxes having same IOU
                if score$(c, b_j) >$ score$(c, b_i)$ then
                    $discard \leftarrow$ True    Compare the scores. If score of b(i) is less than that of b(j), b(i) should be discarded, so set the flag to True.
        if not $discard$ then    Once b(i) is compared with all other boxes and still the discarded flag is False, then b(i) should be considered. So
            $B_{nms} \leftarrow B_{nms} \cup b_i$    add it to the final list.
    return $B_{nms}$    Do the same procedure for remaining boxes and return the final list
```

Before non-max suppression

After non-max suppression



- Now if you observe the algorithm above, *the whole filtering process depends on single threshold value*. *So selection of threshold value is key for performance of the model*. However setting this threshold is tricky. Let us see this scenario.

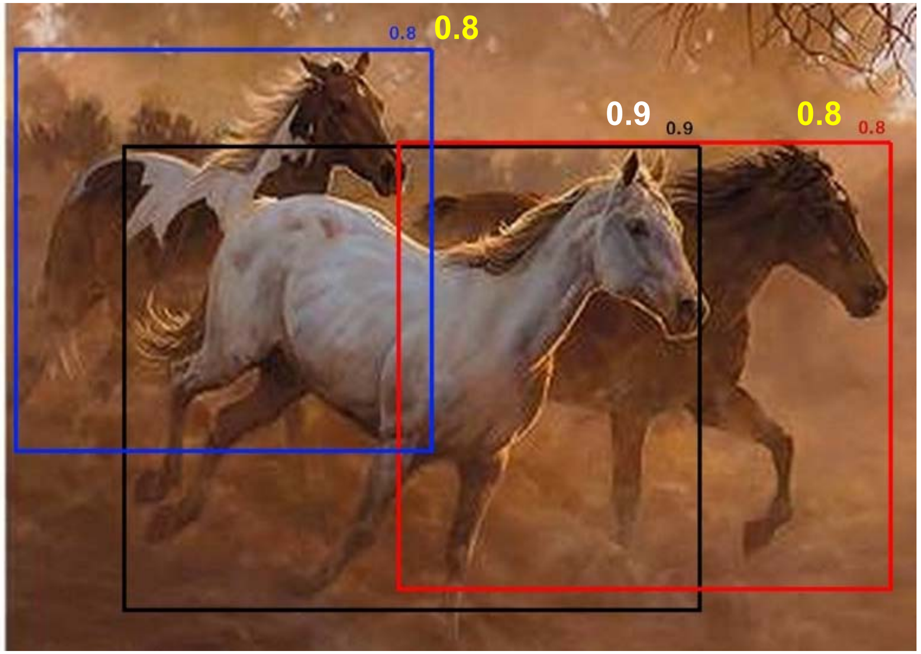
1 Yolo v1 – You Only Look Once

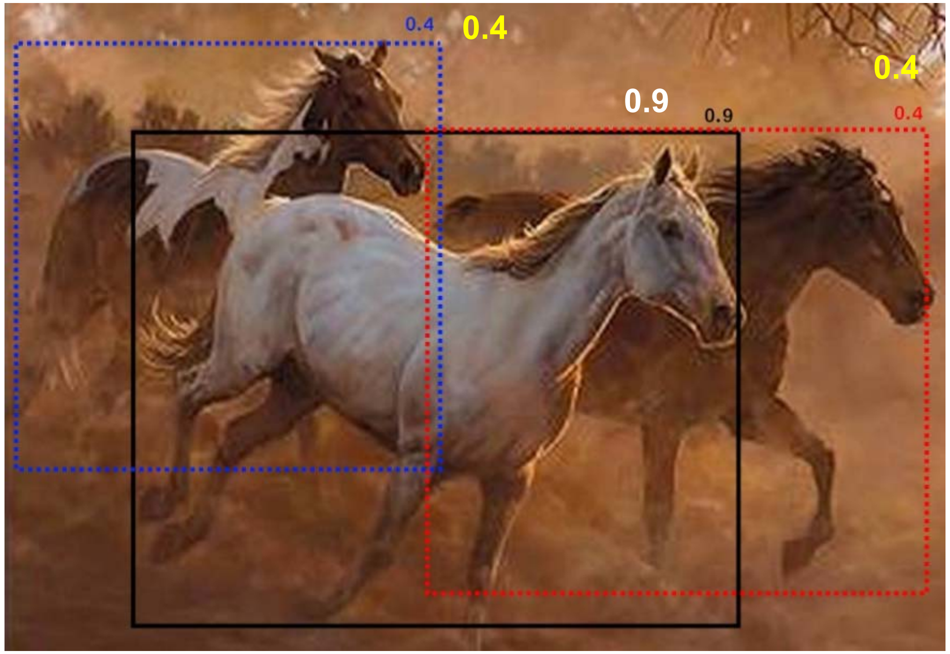
[발췌 자료] https://ctkim.tistory.com/98
https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c

NMS (Non-Maximum Suppression)

- 위의 알고리즘은 가장 높은 confidence를 가지는 bbox를 찾고, bbox들 중 겹치는 영역이 임계값 이상이며 제거해서 중복을 제거한다. 그러나 이 임계값을 설정하는 것은 매우 까다로우며 mAP가 낮아지는 문제가 있다.
- 예시로 아래 그림과 같은 상황이 있다. 아래 그림은 말들이 겹쳐져 있다. 이때 confidence는 0.8 / 0.9 / 0.8 이다. 가장 높은 0.9의 말을 제외하고 0.8의 말은 사라진다.

- 이런 문제점을 대처하는 방법은 **Soft-NMS**를 사용하는 것이다.
- 이 아이디어는 매우 간단하다. ***일정 비율 이상으로 겹쳐진 bbox들의 confidence를 0으로 만들어 제거하는 것이 아닌 confidence를 줄여서 유지하는 것***이다. 이 방법은 **최종 mAP를 향상**시킬 수 있다.





1 Yolo v1 – You Only Look Once

[발췌 자료] https://ctkim.tistory.com/98
https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c

NMS (Non-Maximum Suppression)

Soft-NMS

- The scores 0.4 of both the proposals are calculated based on the IOU values. The score calculation is as follows.

$$s_i = \begin{cases} s_i, & \text{iou}(\mathcal{M}, b_i) < N_t \\ s_i(1 - \text{iou}(\mathcal{M}, b_i)), & \text{iou}(\mathcal{M}, b_i) \geq N_t \end{cases},$$

s_i — score of proposal i,

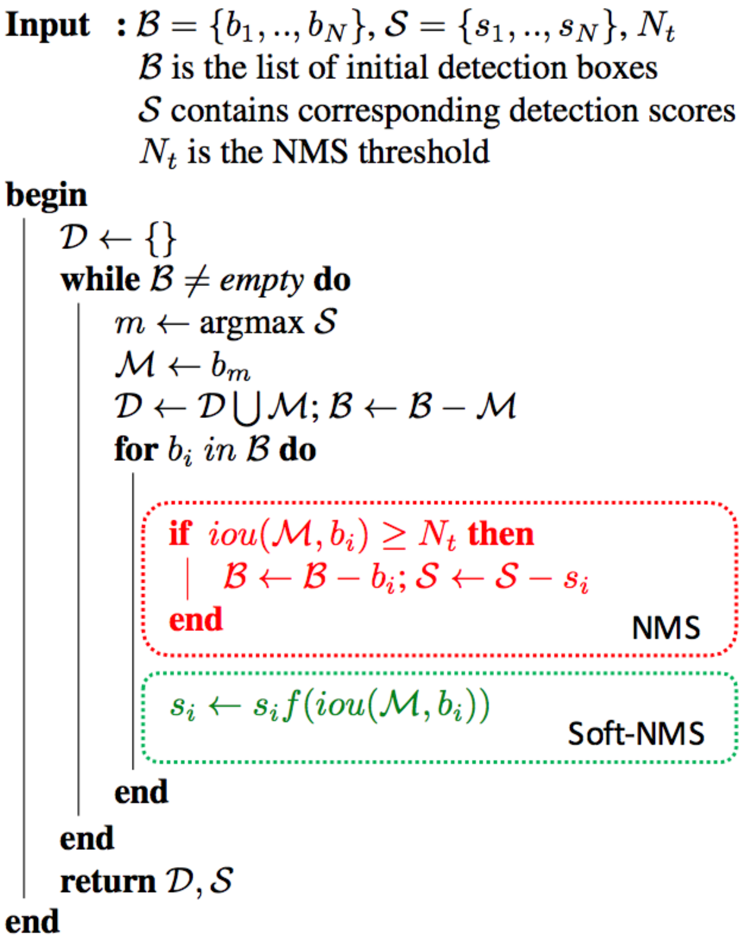
b_i — box corresponding to proposal i,

M — box corresponding to maximum confidence,

N_t — IOU threshold

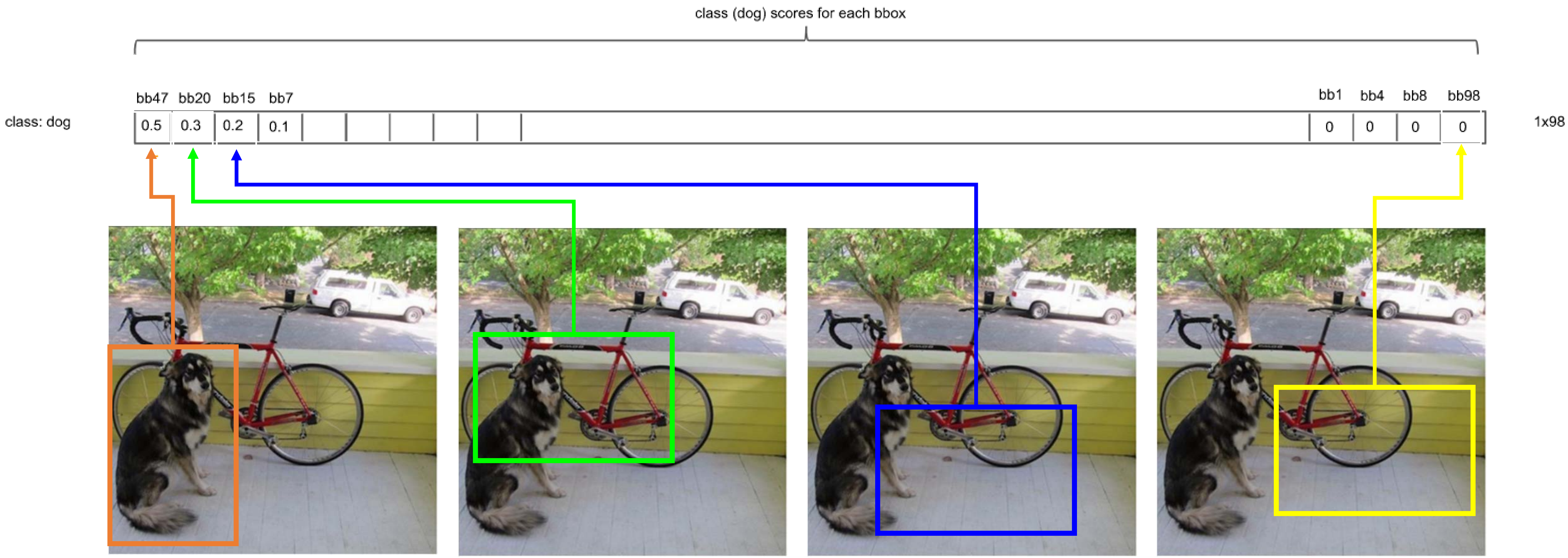
- These techniques works well for ***filtering predictions of a single model***, **What if you have predictions from multiple models?** **Weighted boxes fusion** is a novel method for combining predictions of object detection models.

➢ https://medium.com/analytics-vidhya/weighted-boxes-fusion-86fad2c6be16

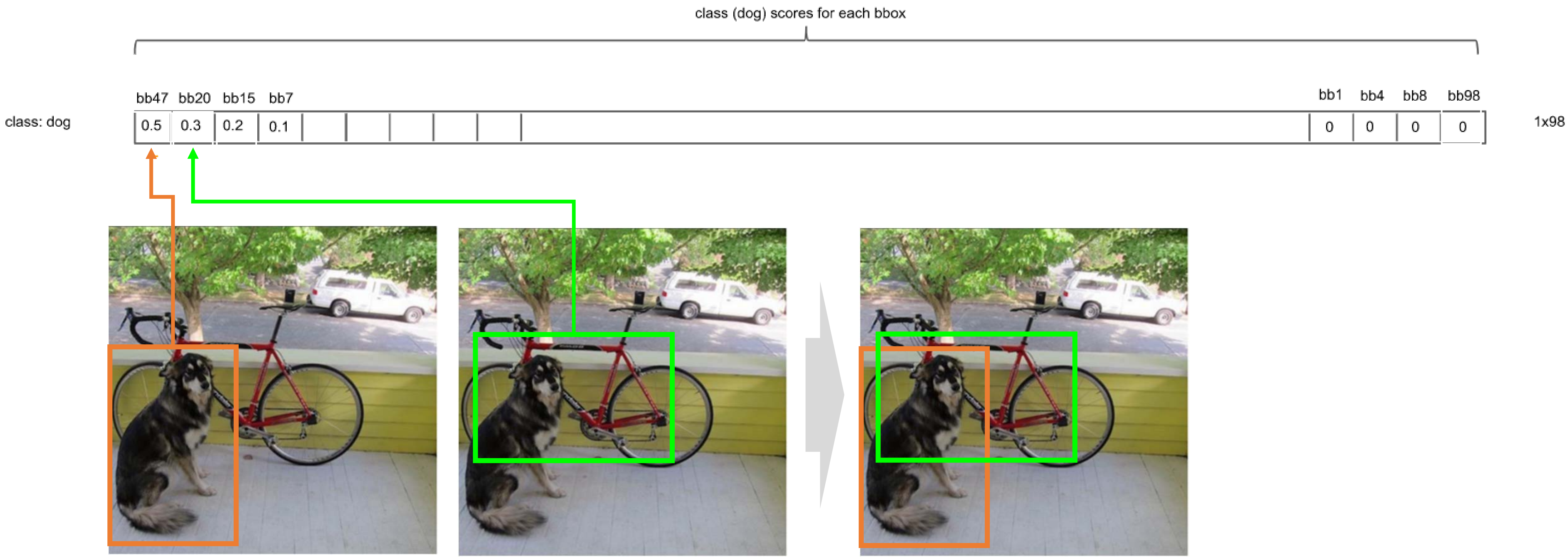


```
Input  : $\mathcal{B} = \{b_1, .., b_N\}$, $\mathcal{S} = \{s_1, .., s_N\}$, $N_t$
         $\mathcal{B}$ is the list of initial detection boxes
         $\mathcal{S}$ contains corresponding detection scores
         $N_t$ is the NMS threshold
begin
    $\mathcal{D} \leftarrow \{\}$
    while $\mathcal{B} \neq empty$ do
        $m \leftarrow$ argmax $\mathcal{S}$
        $\mathcal{M} \leftarrow b_m$
        $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{M}; \mathcal{B} \leftarrow \mathcal{B} - \mathcal{M}$
        for $b_i$ in $\mathcal{B}$ do
            if $iou(\mathcal{M}, b_i) \geq N_t$ then
                $\mathcal{B} \leftarrow \mathcal{B} - b_i; \mathcal{S} \leftarrow \mathcal{S} - s_i$
            end
                                                NMS
            $s_i \leftarrow s_i f(iou(\mathcal{M}, b_i))$
                                                Soft-NMS
        end
    end
    return $\mathcal{D}, \mathcal{S}$
end
```

NMS (Non-Maximum Suppression) : intuition



NMS (Non-Maximum Suppression) : intuition



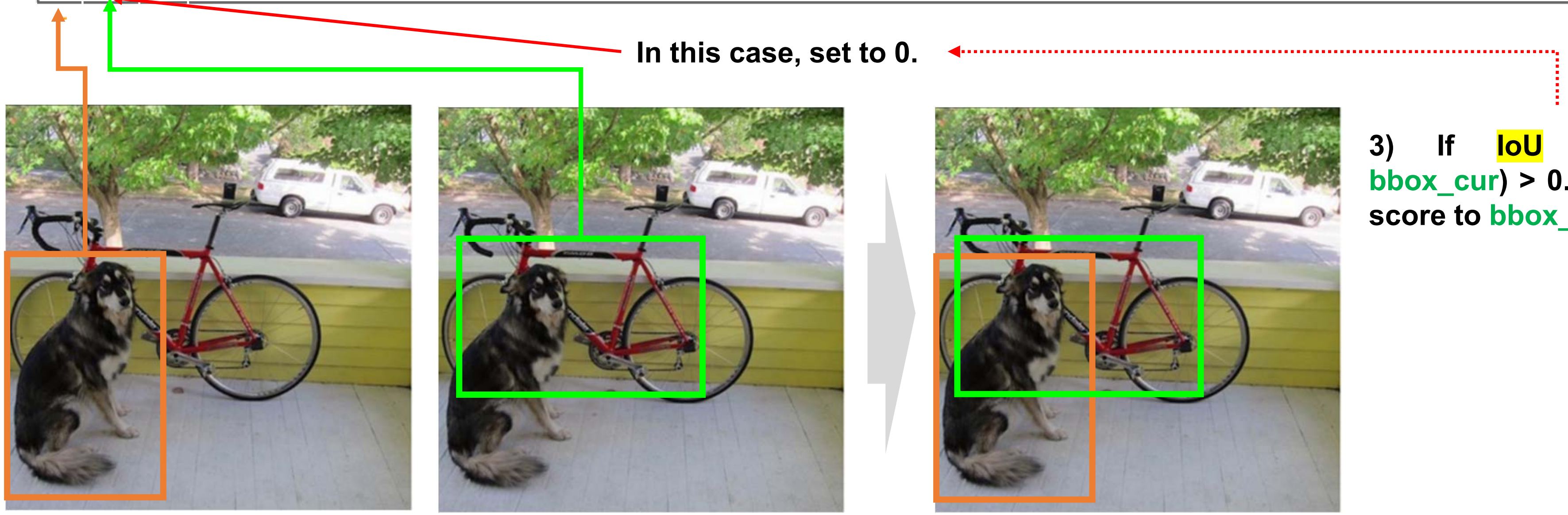
1) Get bbox with max score. Let's denote it "bbox_max"

2) Compare "bbox_max" with others less score (non-zero) bboxes. Let's denote it "bbox_cur"

NMS (Non-Maximum Suppression) : intuition

class (dog) scores for each bbox

	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0	0.2	0.1							0	0	0	0	1x98



In this case, set to 0.

3) If IoU (bbox_max, bbox_cur) > 0.5, then set 0 score to bbox_cur.

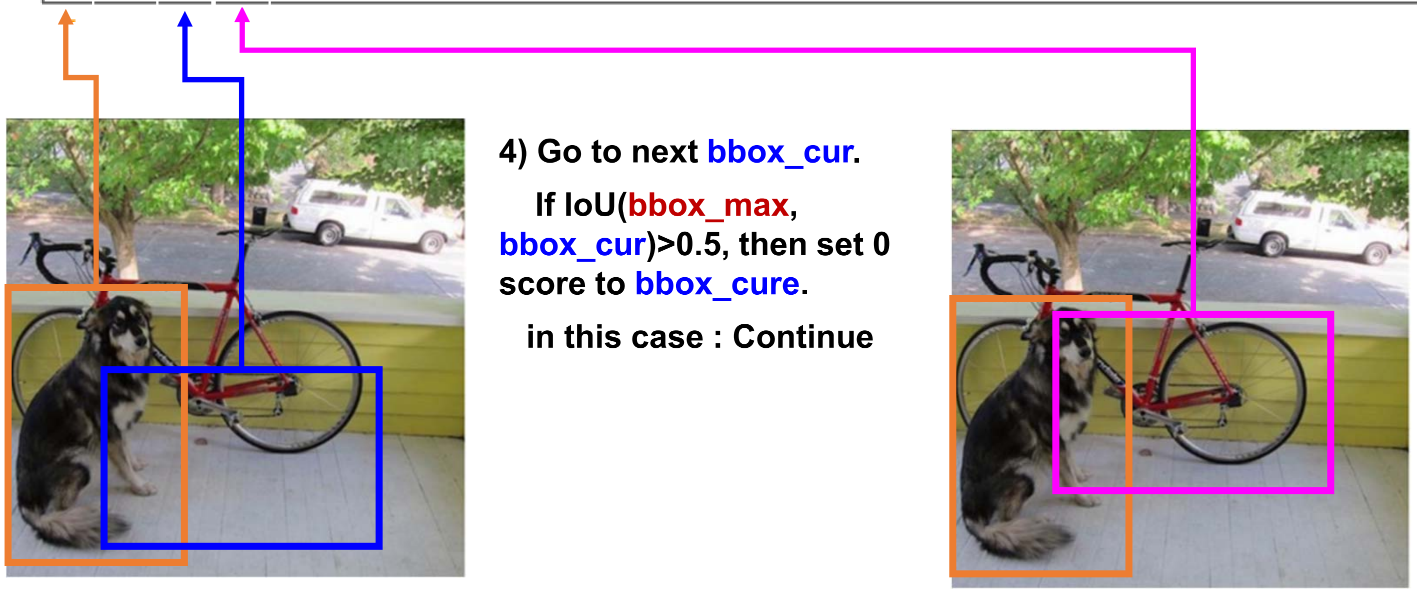
1) Get bbox with max score. Let’s denote it “bbox_max”

2) Compare “bbox_max” with others less score (non-zero) bboxes. Let’s denote it “bbox_cur”

NMS (Non-Maximum Suppression) : intuition

class (dog) scores for each bbox

	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0	0.2	0.1							0	0	0	0	1x98



4) Go to next bbox_cur.

If IoU(bbox_max, bbox_cur)>0.5, then set 0 score to bbox_cure.

in this case : Continue

5) Go to next bbox_cur.

If IoU(bbox_max, bbox_cur)>0.5, then set 0 score to bbox_cure.

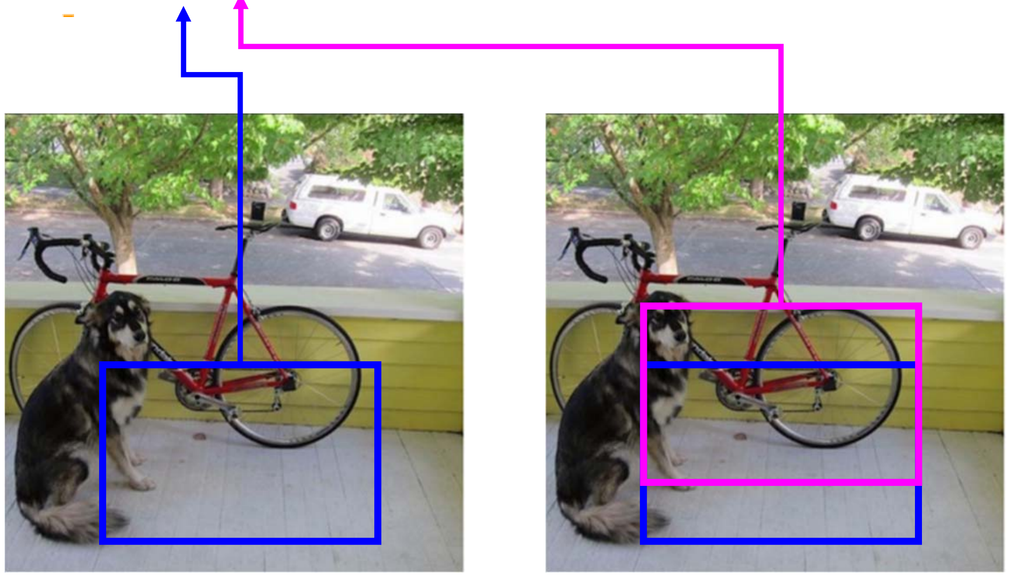
in this case : Continue

Do this procedure for other “bbox_cur”. After that....

NMS (Non-Maximum Suppression) : intuition

class (dog) scores for each bbox

	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	**0**	0.2	**0**							0	0	0	0	1x98



Go to next **bbox_cur**.

If IoU(**bbox_max**, **bbox_cur**)>0.5, then set 0 score to **bbox_cure**.

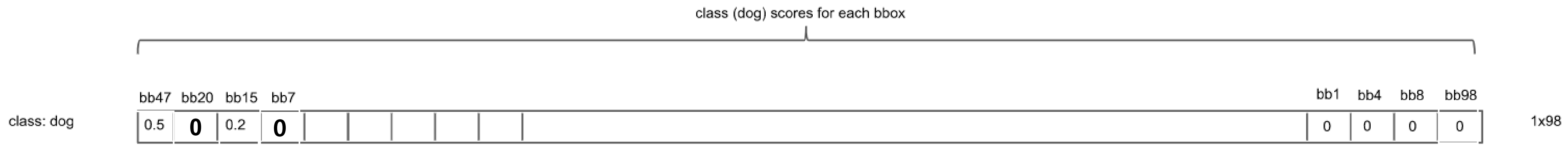
in this case : Set to 0

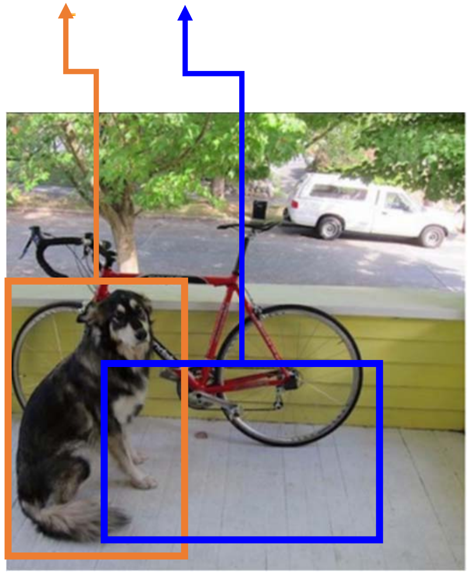
Do this procedure for other “bbox_max” and for other corresponding “bbox_cur”

Go to next bbox with big score.

Let’s denote it “**bbox_max**”

NMS (Non-Maximum Suppression) : intuition

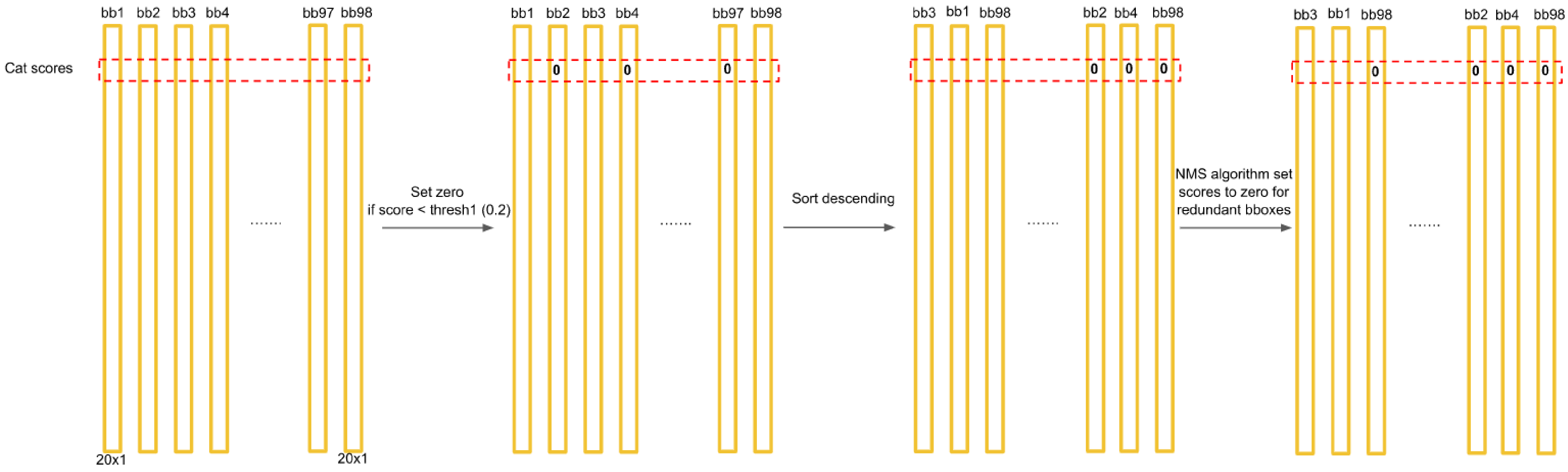




After comparison almost all pairs of bboxes the only two bboxes with non-zero class score value.

[발췌 자료]deepsystems.io

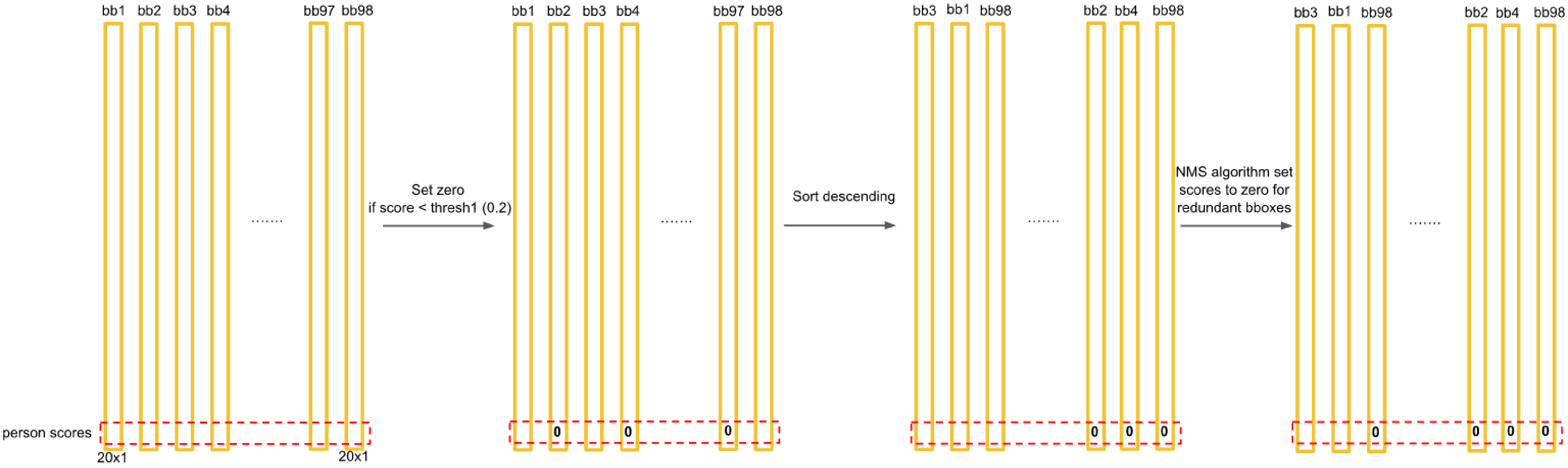
NMS (Non-Maximum Suppression) : intuition



Do this procedure for next class

[발췌 자료]deepsystems.io

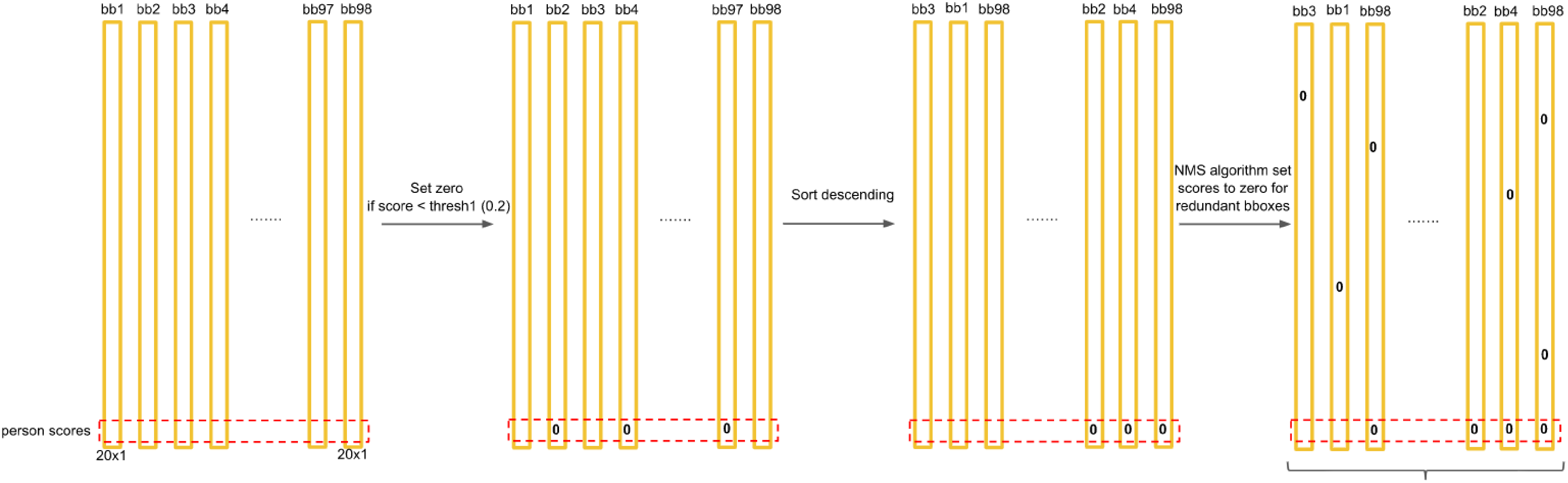
NMS (Non-Maximum Suppression) : intuition



Do this procedure for all classes

[발췌 자료]deepsystems.io deepsystems.io

NMS (Non-Maximum Suppression) : intuition

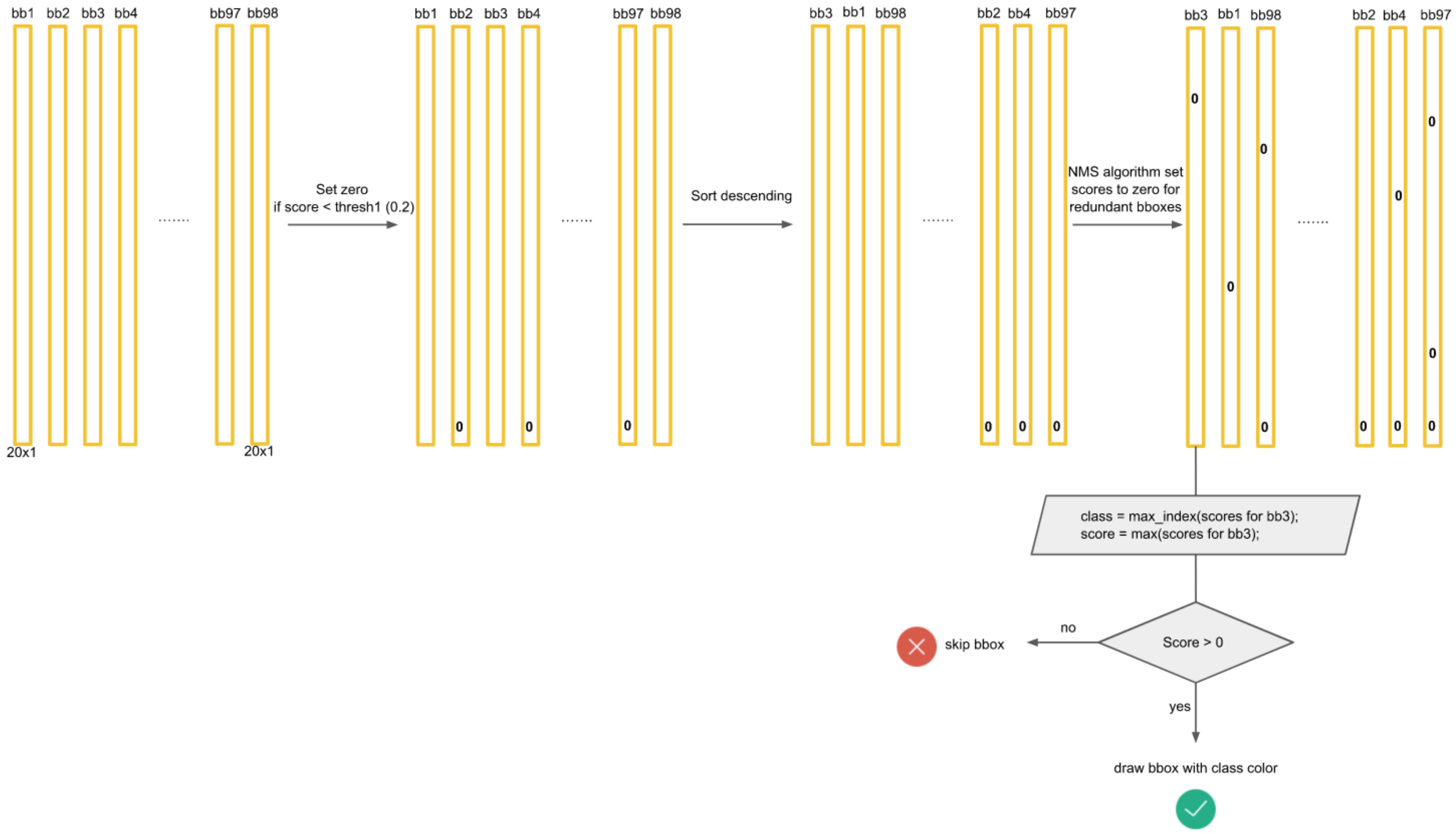


After this procedure –
a lot of zeros

Select bboxes to draw
by class score values

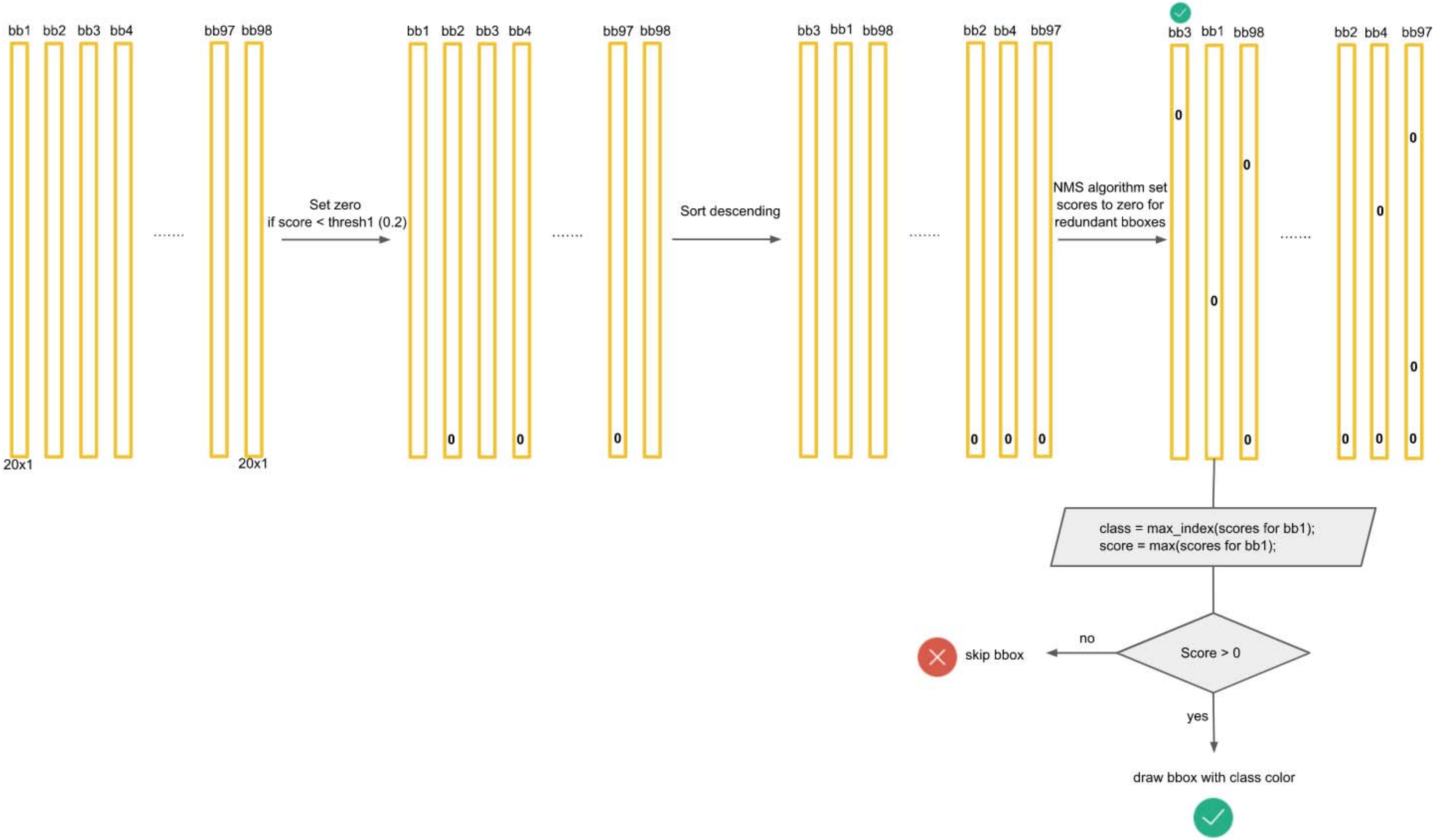
[발췌 자료]deepsystems.io

NMS (Non-Maximum Suppression) : intuition


bb1 bb2 bb3 bb4 .... bb97 bb98
20x1
20x1
Set zero
if score < thresh1 (0.2)
bb1 bb2 bb3 bb4 .... bb97 bb98
0
0
0
Sort descending
bb3 bb1 bb98 .... bb2 bb4 bb97
0 0 0
NMS algorithm set
scores to zero for
redundant bboxes
bb3 bb1 bb98 .... bb2 bb4 bb97
0
0
0
0
0
0
0
0
0
0
class = max_index(scores for bb3);
score = max(scores for bb3);
Score > 0
no
skip bbox
yes
draw bbox with class color

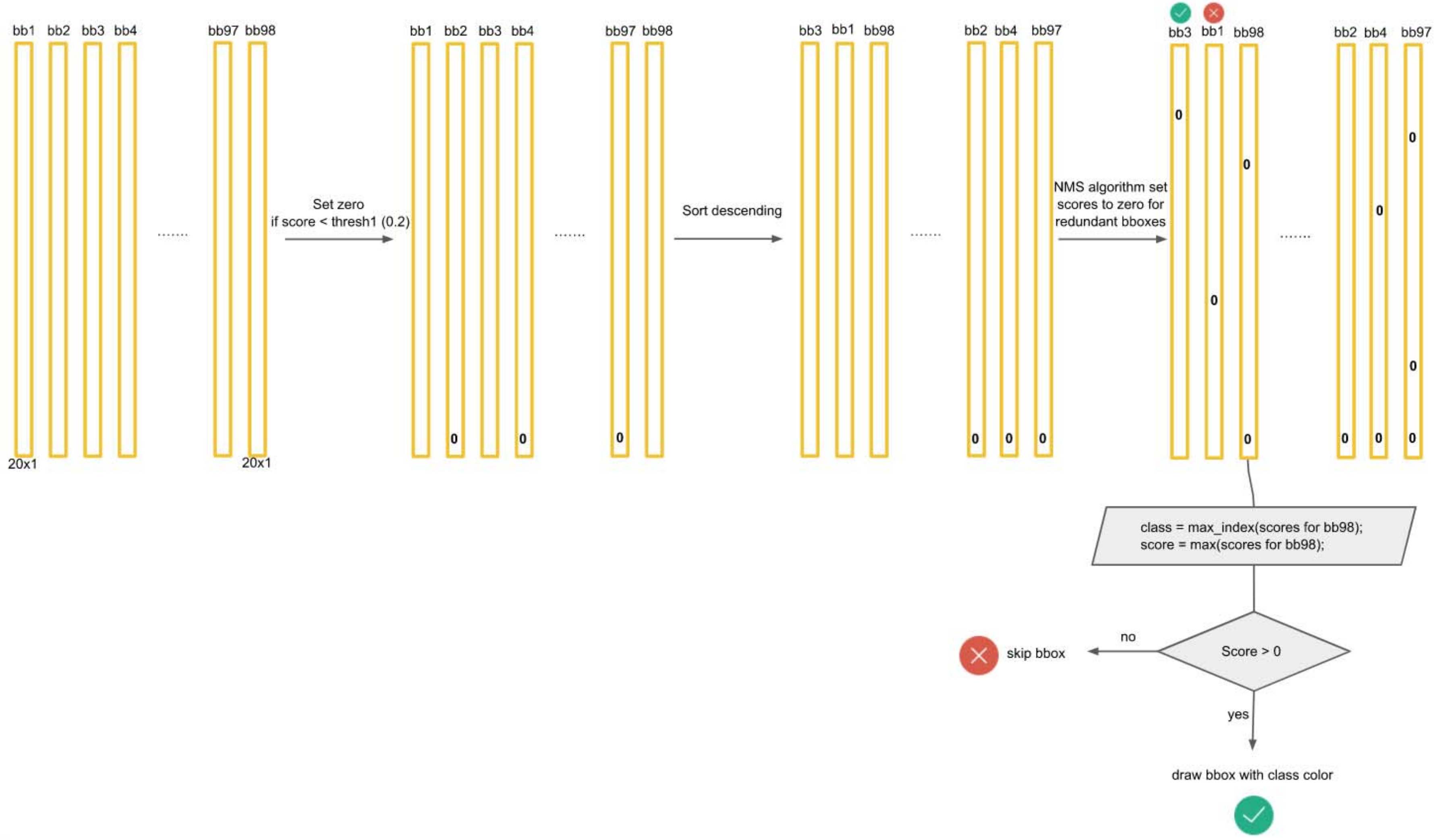
[발췌 자료]deepsystems.io

NMS (Non-Maximum Suppression) : intuition



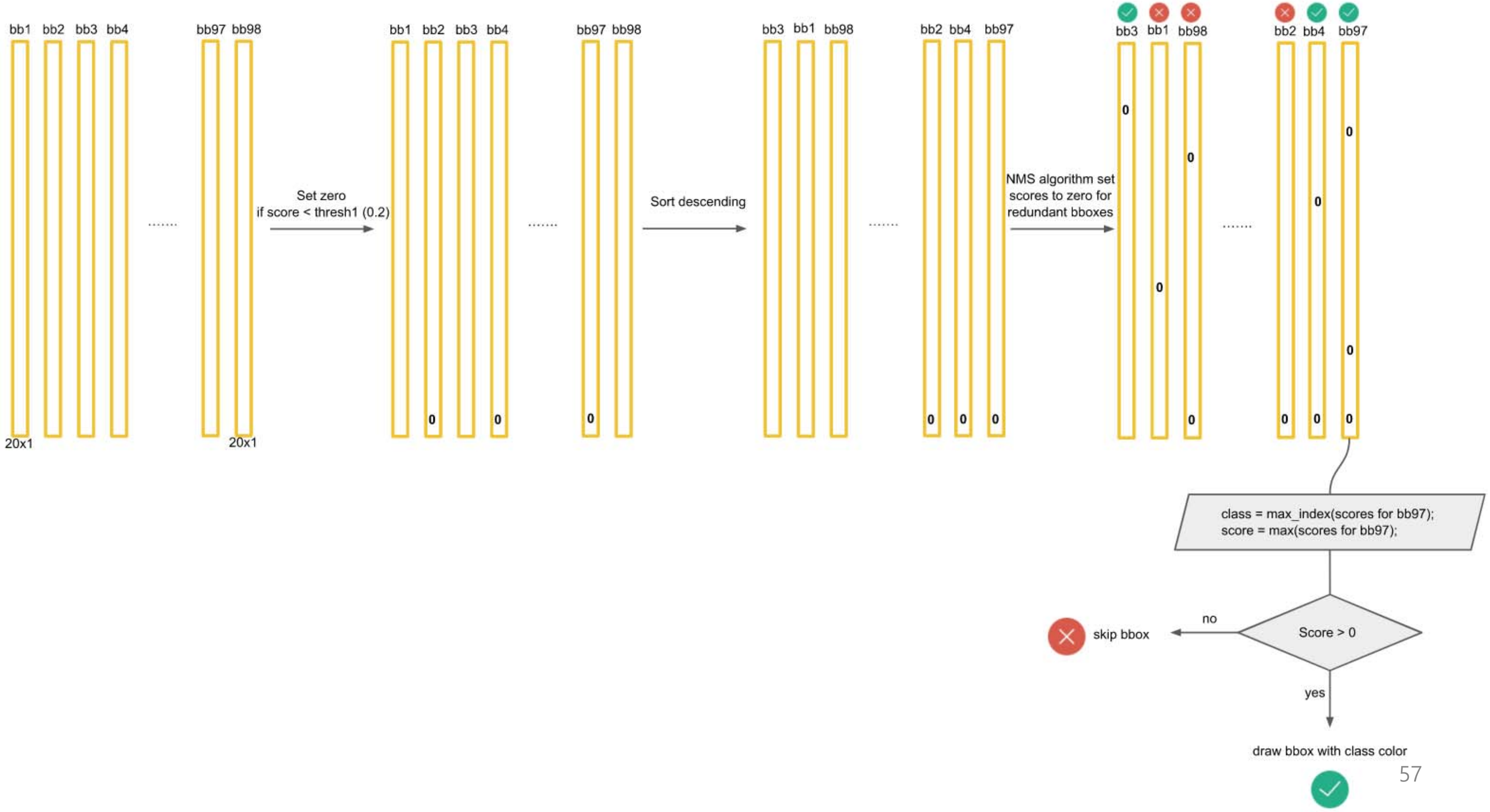
[발췌 자료]deepsystems.io

NMS (Non-Maximum Suppression) : intuition



[발췌 자료]deepsystems.io

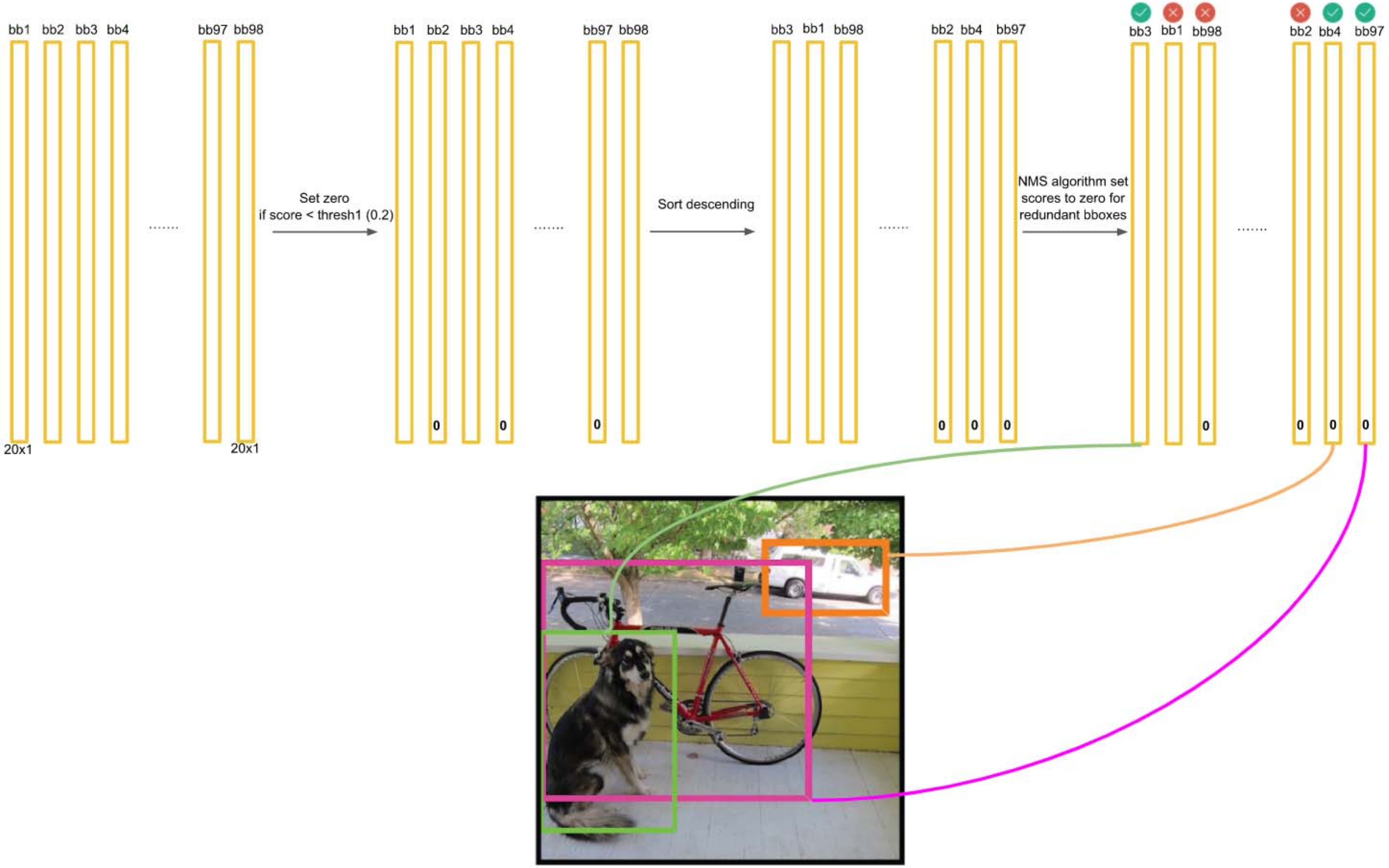
NMS (Non-Maximum Suppression) : intuition



[발췌 자료]deepsystems.io

deepsystems.io

NMS (Non-Maximum Suppression) : intuition



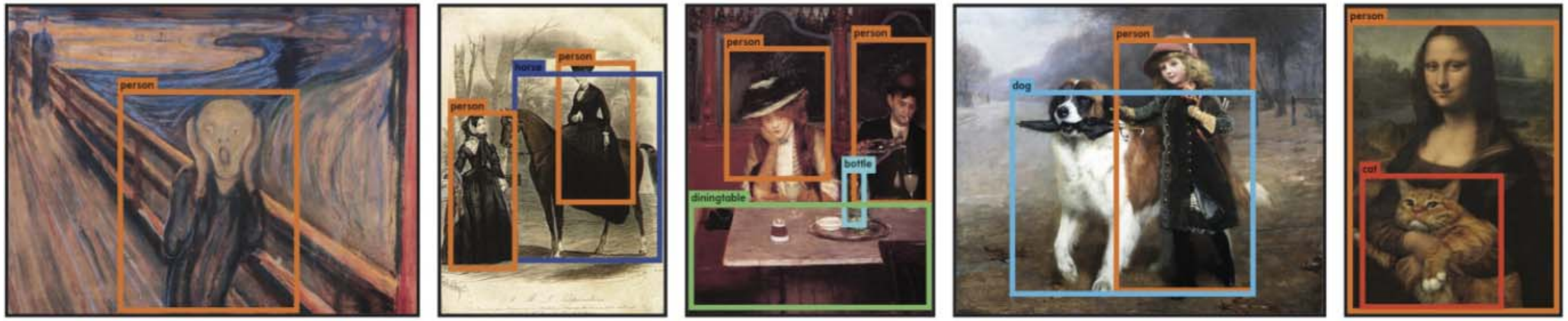
1

Yolo v1 – You Only Look Once

[발췌 자료]deepsystems.io

Key Points

1. Fast: YOLO - 45 fps, YOLO-tiny - 155 fps.
2. End-to-end training.
3. Makes more localization errors but is less likely to predict false positives on background
4. Performance is lower than the current state of the art.
5. Combined Fast R-CNN + YOLO model is one of the highest performing detection methods.
6. Learns very general representations of objects: it outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains


person
horse
person
person
person
person
bottle
diningtable
person
dog
person
cat