YOLO

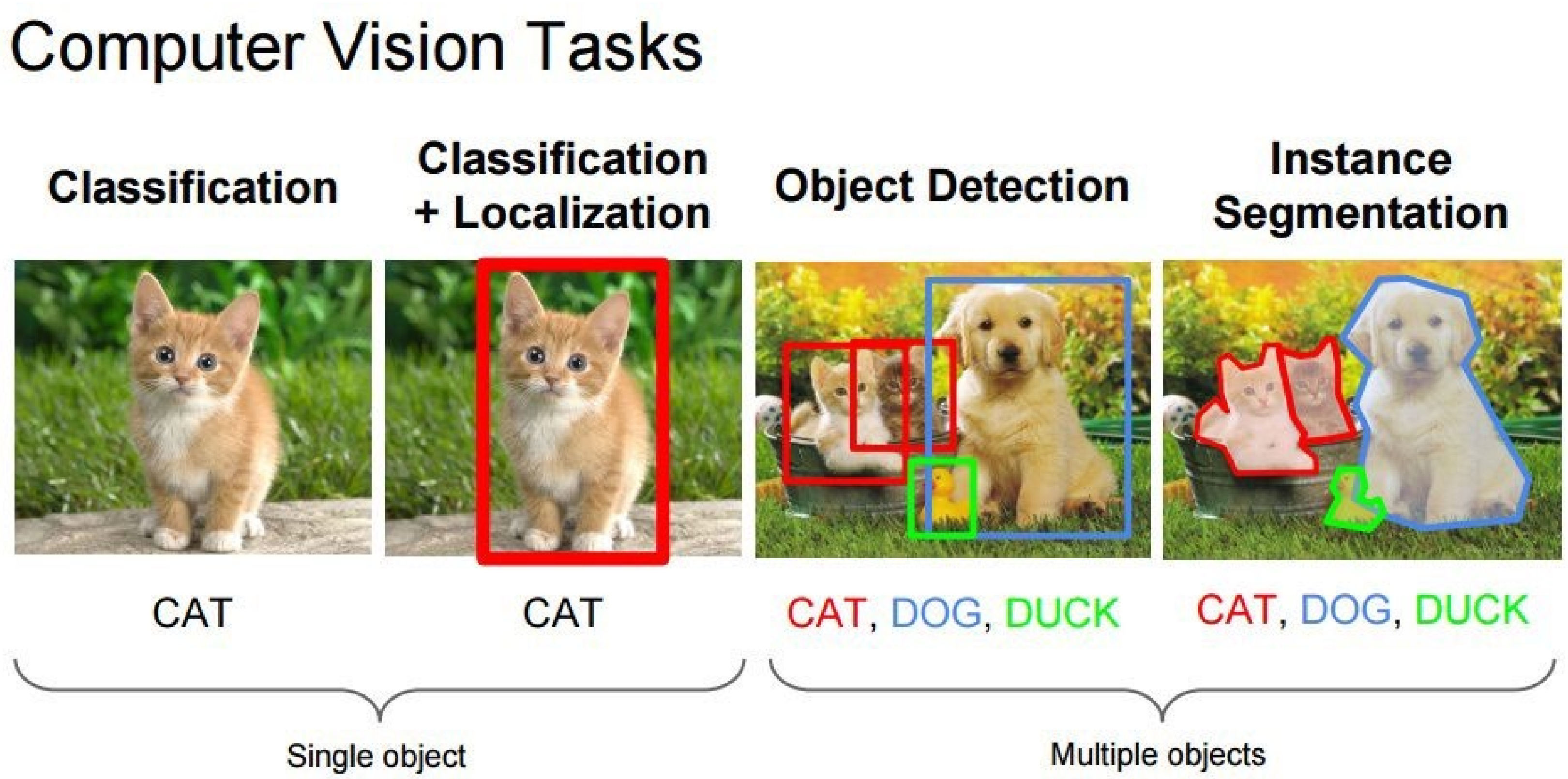
You Only Look Once: Unified, Real-Time Object Detection

Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi

University of Washington, Allen Institute for AI, Facebook AI Research

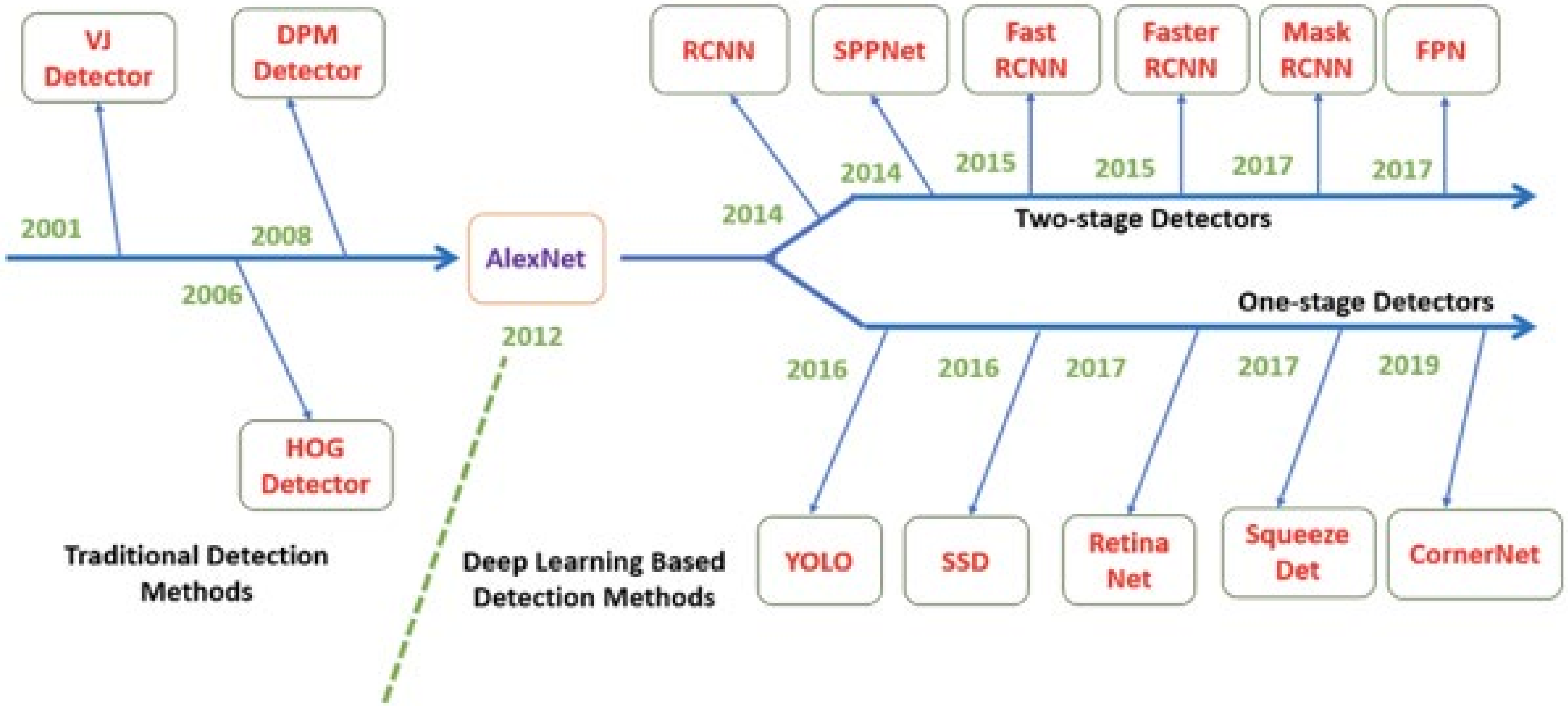
Younghoon Na
Email : nayounghoon0223@gmail.com
Github : younghoonNa@github.com

Computer Vision



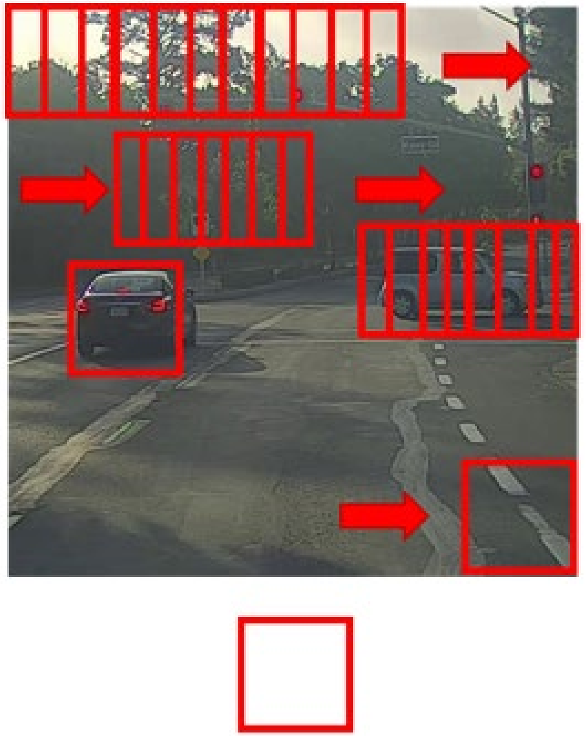
Obejct Detection -> (Multi-Labeled) Classification + Localization

Object Detection

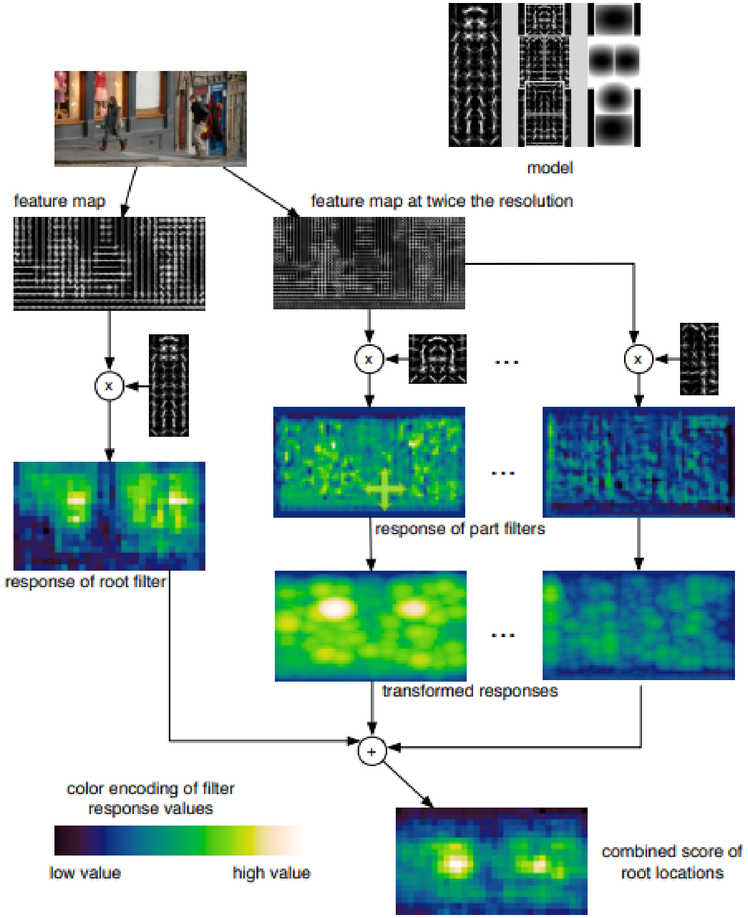


Deformable parts models (DPM)

Traditional Object Detection



Sliding Window

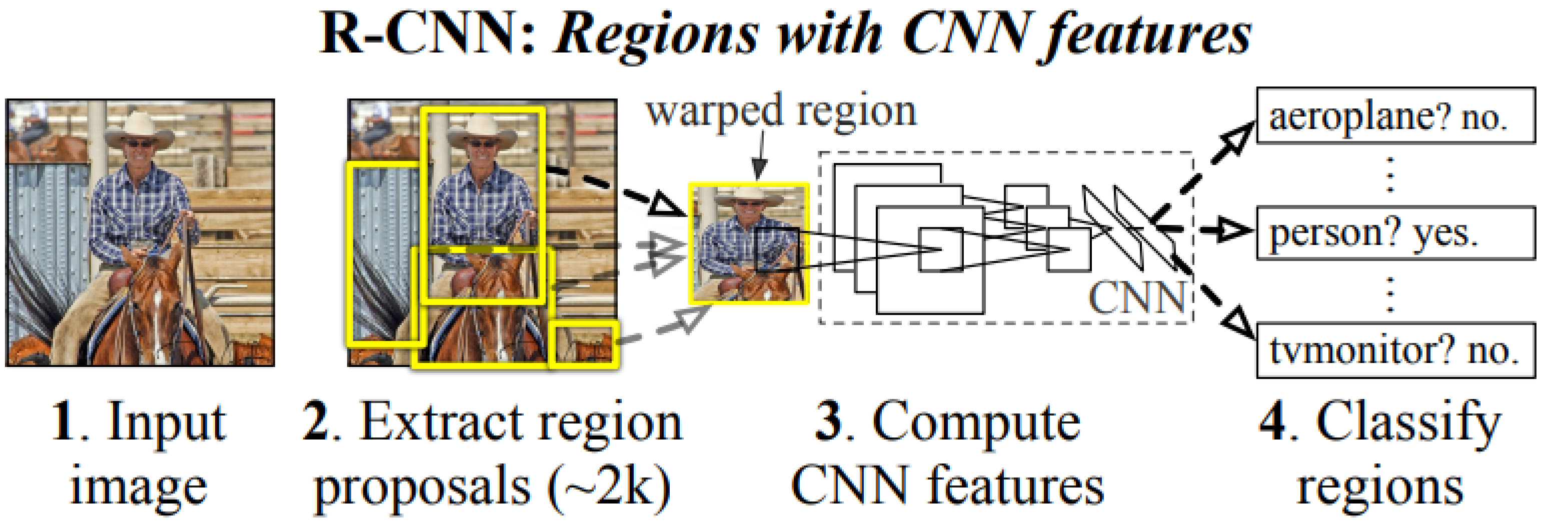


Disjoint pipeline

R-CNN (Regions Based Convolutional Neural Networks)

Deep Learning Based Object Detection

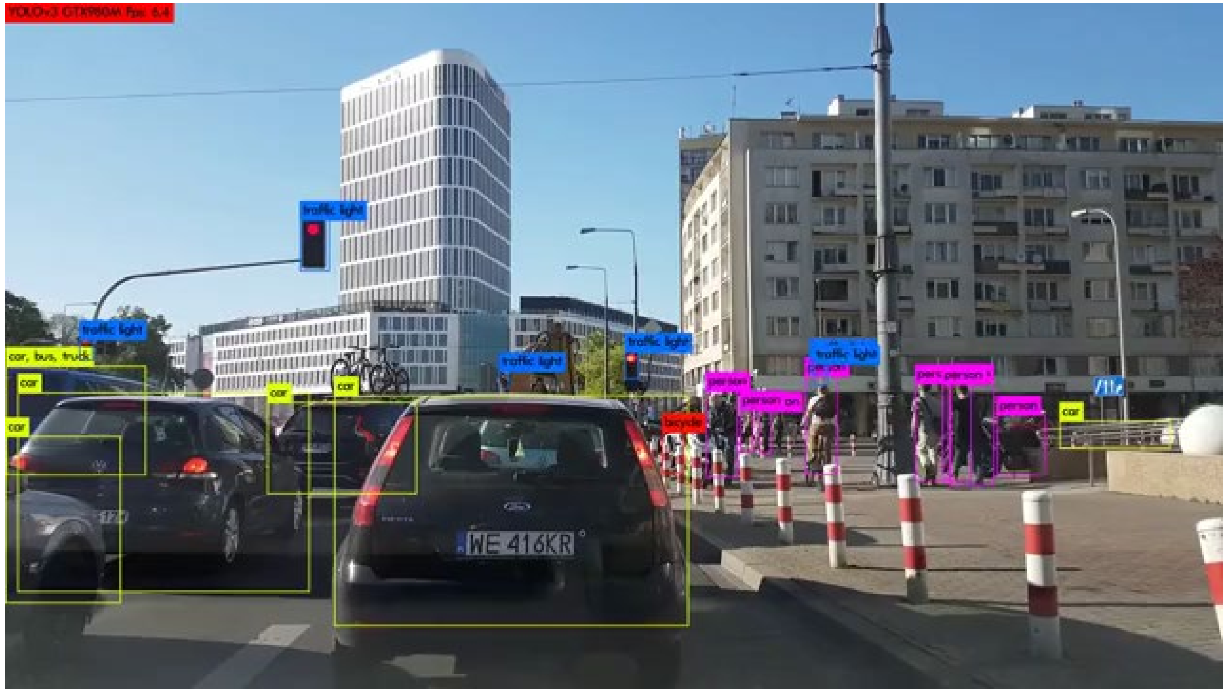
-> Two-Stage Detector



Region proposals

Region Classification

Real-time object detection



https://www.youtube.com/watch?v=MPU2HistivI

실시간 물체 탐지를 위해서는
초당 30프레임, 30FPS 이상 나와야 한다고 한다.

ImageNet & PASCAL VOC

ImageNet 2012
Class 수 : 1000

PASCAL VOC 2007
Class 수 : 20

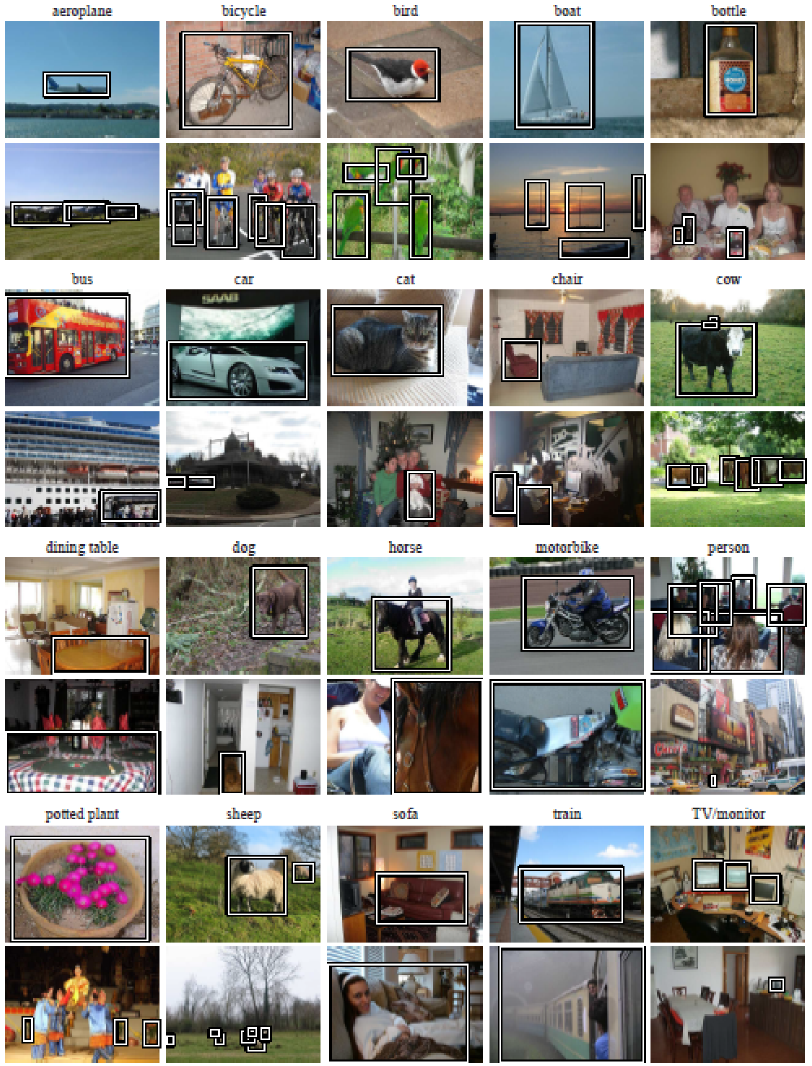


Fig. 1 Example images from the VOC2007 dataset. For each of the 20 classes annotated, two examples are shown. Bounding boxes indicate all instances of the corresponding class in the image which are marked as "non-difficult" (see Sect. 3.3) – bounding boxes for the other classes are available in the annotation but not shown. Note the wide range of pose, scale, clutter, occlusion and imaging conditions.

YOLO – Introduce

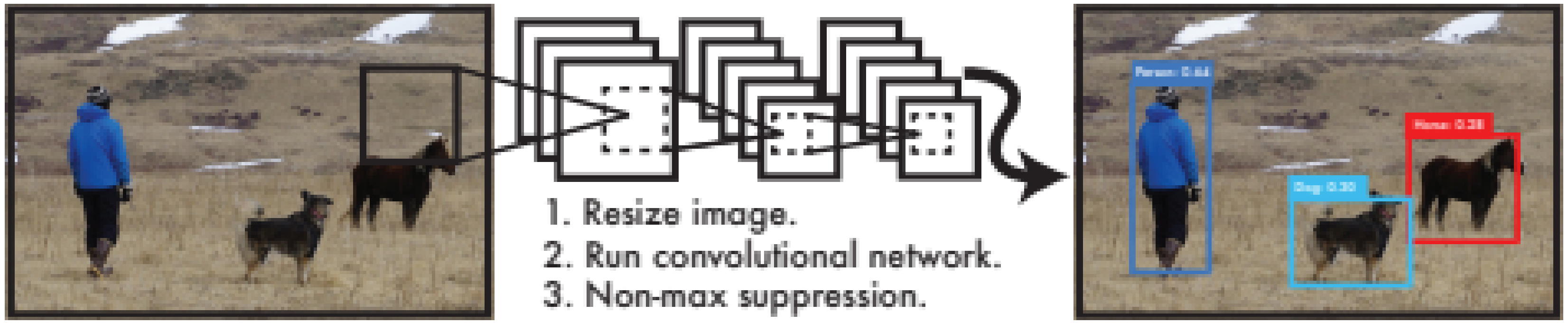


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448×448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

Predict Bounding Box & Class Probabilites -> Regression Problem

-> Simple & available Real-Time

YOLO – Unified Detection

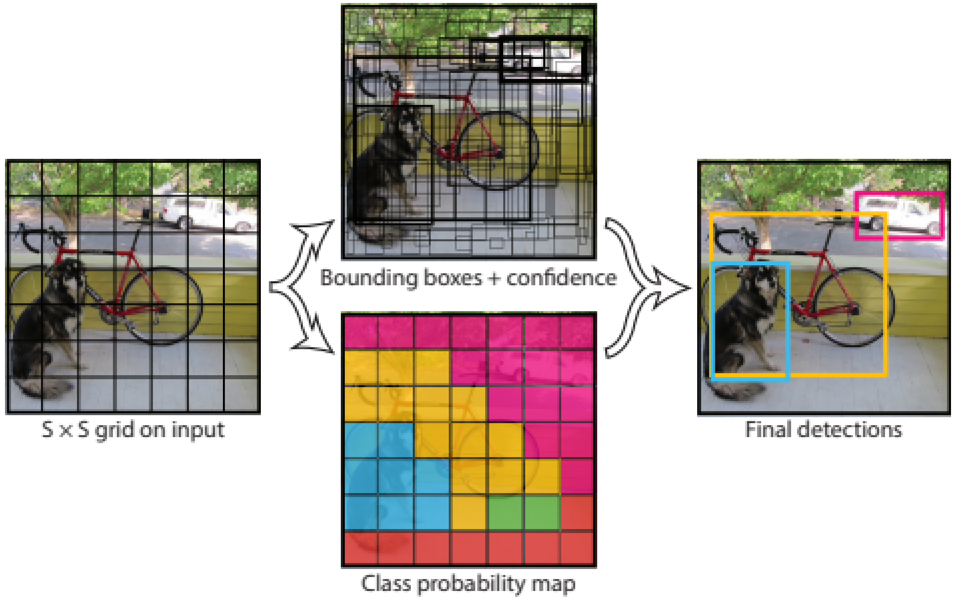
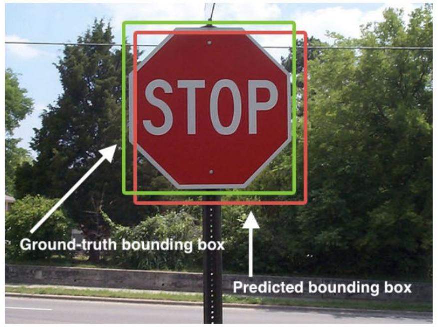


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

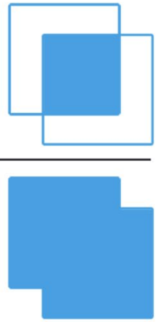
$\Pr(\text{Class}_i | \text{Object})$

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

$\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$



$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



YOLO – Class-specific confidence score

C 　　　　　　　confidence

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

If contain object: Pr(object) = 1
else : Pr(object) = 0

Test time

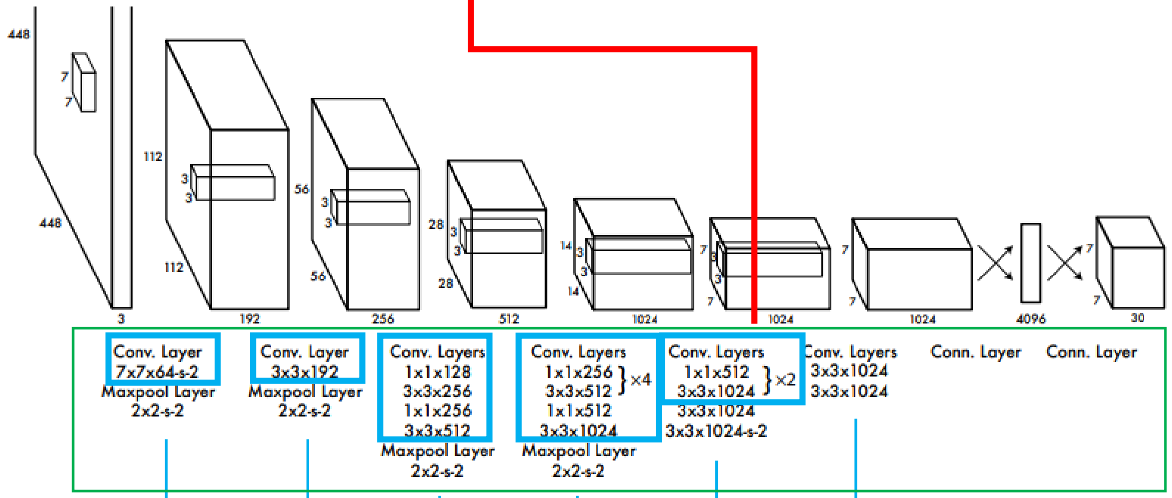
1. 박스가 클래스를 얼마나 잘 나타내는지 -> **Pr(Class_i)**
2. 박스가 얼마나 잘 맞는지 (Localization) -> **IOU**

Unified Detection – Network Design

Inspired by GoogLeNet model

Inception module -> 1x1 reduction layer + 3x3 conv layer

(similar to Network in Network)



1	1	4	10	6	2

Conv Layer : 24
Linear Layer : 2

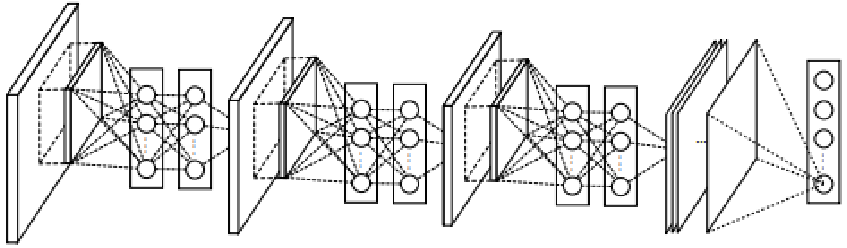


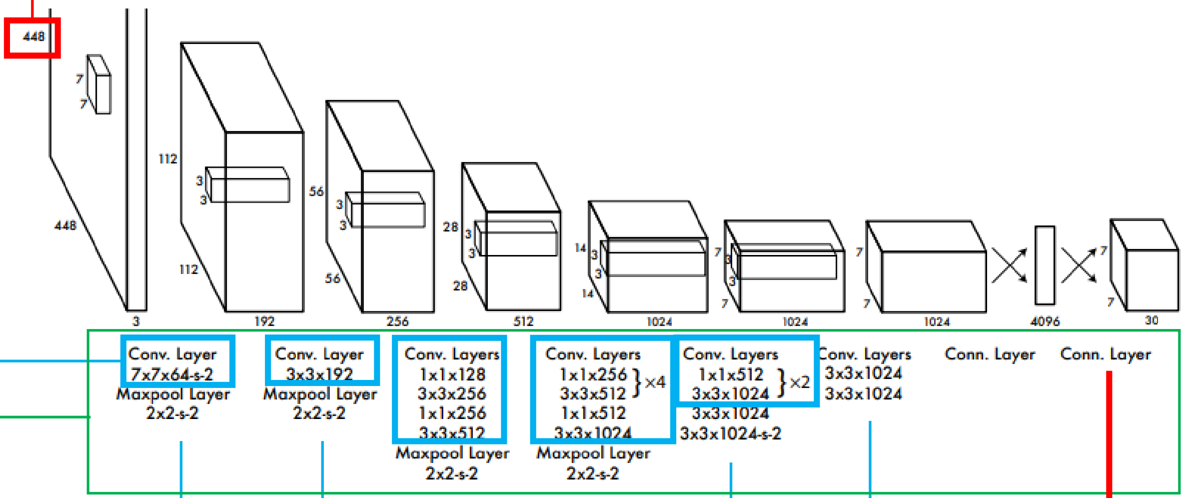
Figure 2: The overall structure of Network In Network. In this paper the NINs include the stacking of three mlpconv layers and one global average pooling layer.

Network in Network (NIN)
-> using 1x1 convolution layer

1. 행렬의 변화 없이 채널 크기 조절 가능
2. 학습해야 하는 Weight 수 감소
3. 비선형성을 부과 가능.

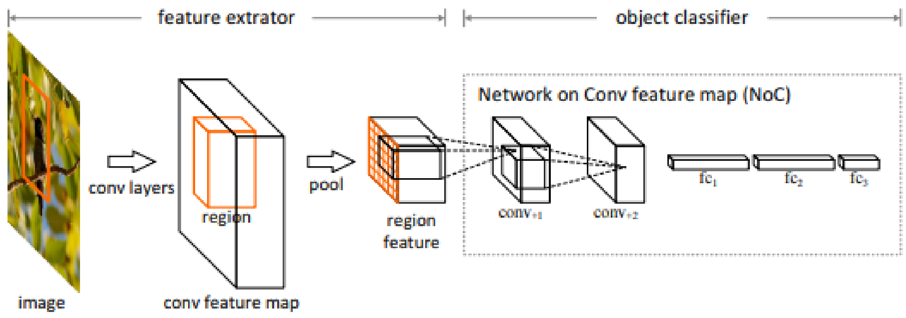
Unified Detection - Training

Pretrain : ImageNet 1000-Classification dataset (224x224)
Fine-tuning : VOC dataset 2007 and 2012 (448x448)
Detection : 448x448

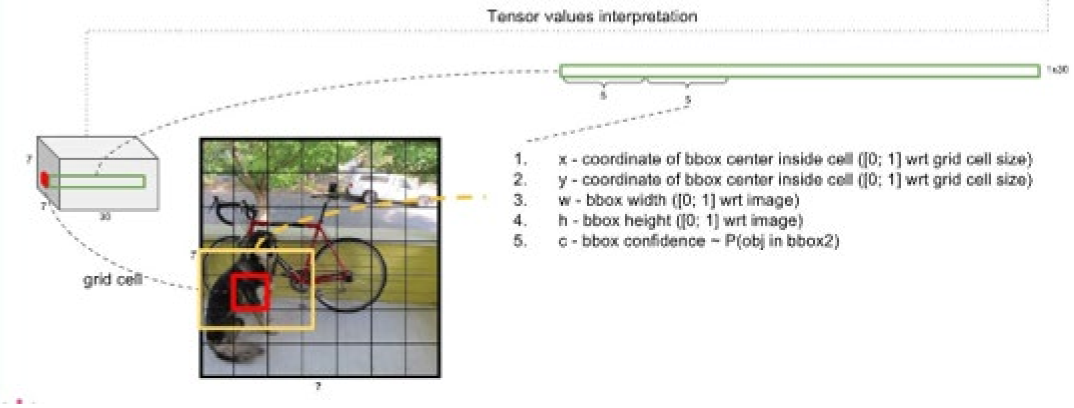


1	1	4	10	6	2

Conv Layer : 24
Linear Layer : 2



S. Ren, K. He, R. B. Girshick, X. Zhang, and J. Sun.
Object detection networks on convolutional feature maps. CoRR, abs/1504.06066, 2015.

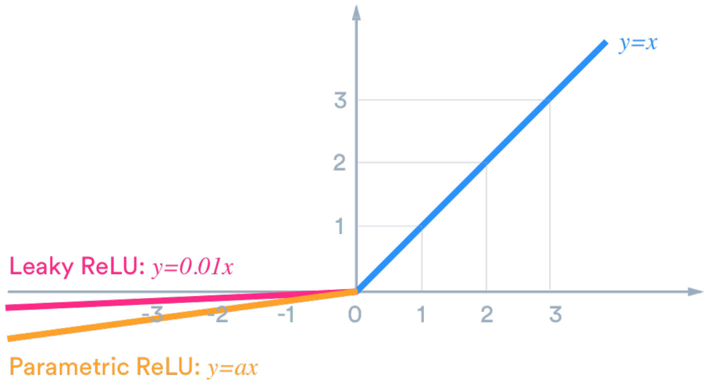


Class probabilities (C = 20)
Bounding box coordinates((x,y,w,h,c) = 5 x (B = 2))

Hyperparater

Linear Activation
-> leaky rectified linear activation

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



Loss Function
-> Sum-Squared-Error

$$\mathrm{SSR} = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2$$

1. **Easy to optimize**

2. **Weights localization equally with classification error**
3. **Training diverge early on**
   - **Confidence score will be zero (not contain object)**
   - **Overpowering the gradient (contain object)**
4. **Weight errors in large box equally small box**

Hyperparater

Loss Function

To solve the problem of equally reflecting weight errors
in large and small boxes

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

바운딩 박스에 대한 Loss의 가중치를 더 높이므로
더욱 정확한 바운딩 박스의 위치를 계산함
(Localization과 Classification 중 Localization의 비중 증가)

$\lambda_{\mathbf{coord}}$ **Increase loss from bounding box coordinates predictions**
Default = 5

λ_{noobj} **Decrease loss from Confidence predictions that don't contain objects for bounding boxes**
Default = 0.5

$\mathbb{1}_i^{\text{obj}}$ **Objects appears in cell I**

$\mathbb{1}_{ij}^{\text{obj}}$ **Jth bounding box predictor in cell i**

1. 바운딩 박스의 중심좌표에 대한 SSE 계산.
2. 바운딩 박스의 높이와 너비에 대해 SSE를 계산.
3. 물체가 있는 Cell의 Confidence Score SSE 계산
4. 물체가 없는 Cell의 Confidence Score SSE 계산
   (물체가 없을 때가 더 많으므로 noobj를 곱함)
5. 각 Grid Cell에 대한 조건부 확률 계산

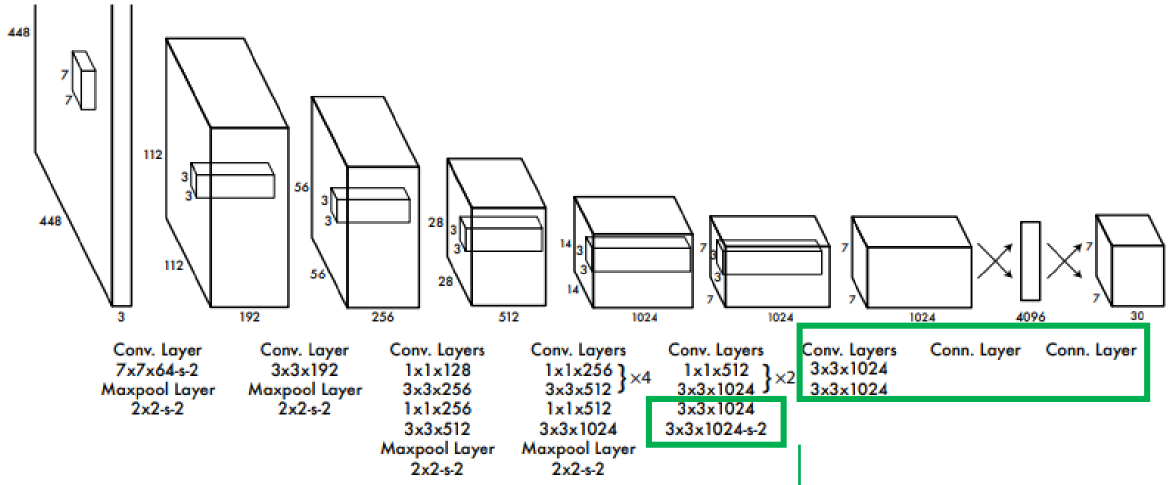
Hyperparater

Pretrain : ImageNet 1000-Classification dataset (224x224)

Fine-tuning : VOC dataset 2007 and 2012

Test : VOC dataset 2012 (also include 2007)

Detection : 448x448



Randomly initialized weighted

Epoch = 135
Batch size = 64
Momentum = 0.9
Weight decay = 0.0005
Dropout = 0.5

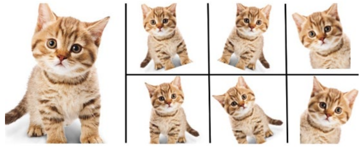
$$L(x,y) \equiv \sum_{i=1}^{n}(y_i - h_\theta(x_i))^2 + \boxed{\lambda \sum_{i=1}^{n} \theta_i^2}$$

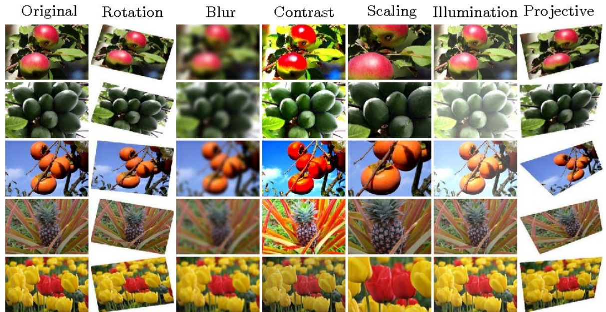
Learning Rate Schedule

Default = 0.001
~ 75 epochs = 0.001 -> 0.01
~ 105 epochs = 0.001
~ 135 epochs = 0.0001

Image Augmentation

Translations +
scaling = up to 20%
Random adjust
exposure/saturation
= factor of 1.5 HSV

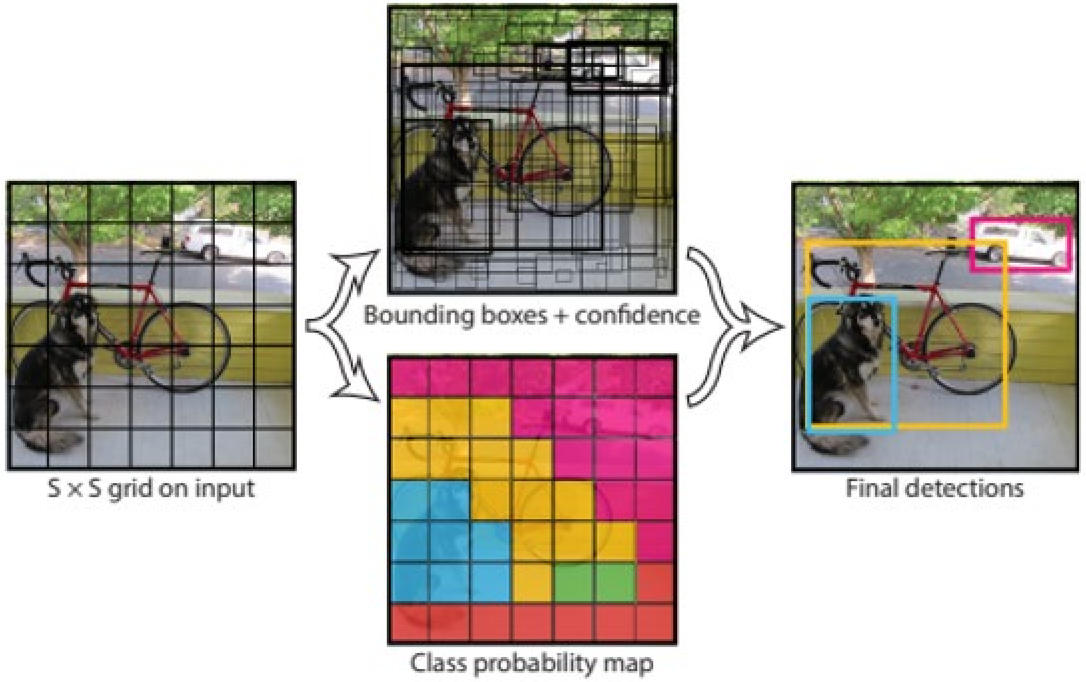


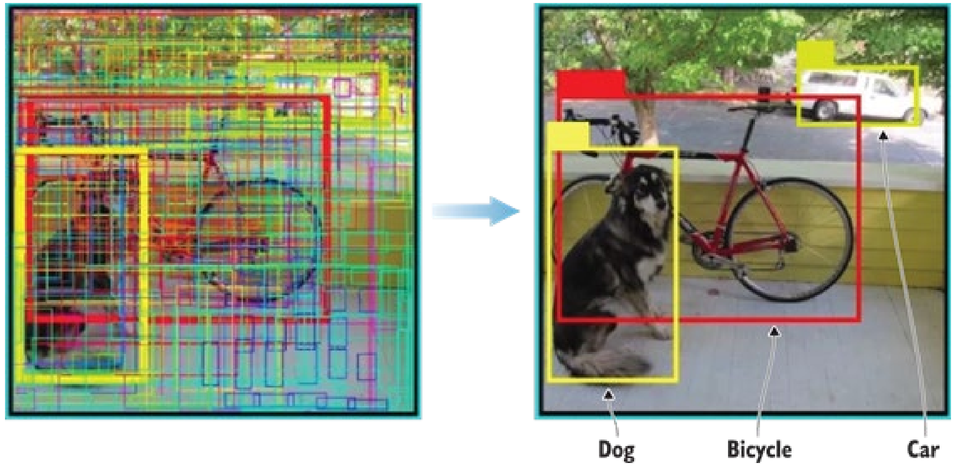


Inference

- Testing on PASCAL VOC, predict on 98 bounding boxes
- YOLO -> Fast
- Only requires a single network

- YOLO model의 디자인 단점 중 하나는 큰 물체가 여러 개의 Grid Cell에서 검출이 된다는 점입니다.
  -> multiple detections problem
- **Non-Maximal Suppression**

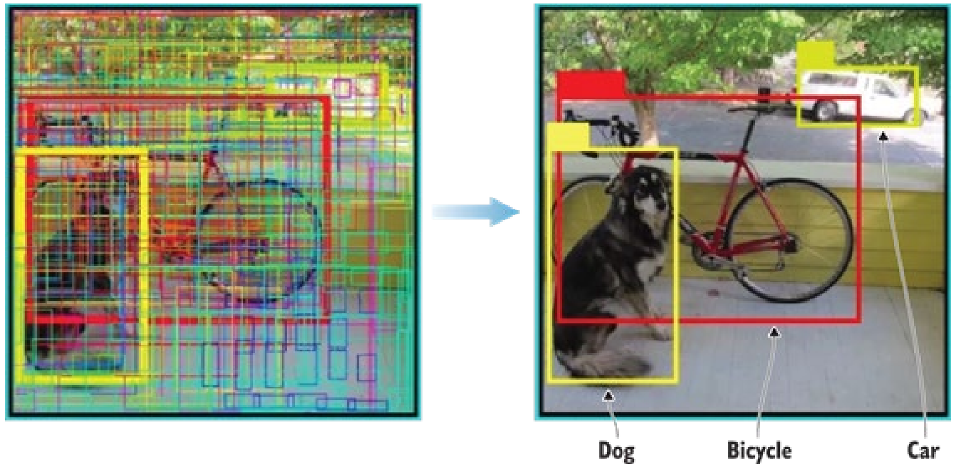




Non-Maximal Suppression

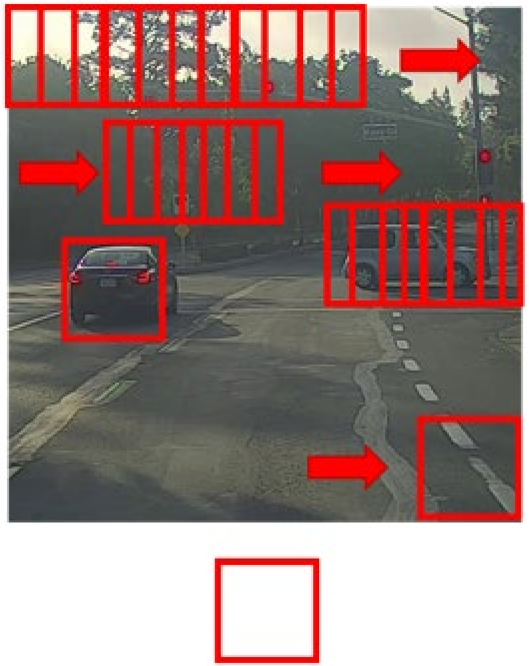
Execution process of NMS

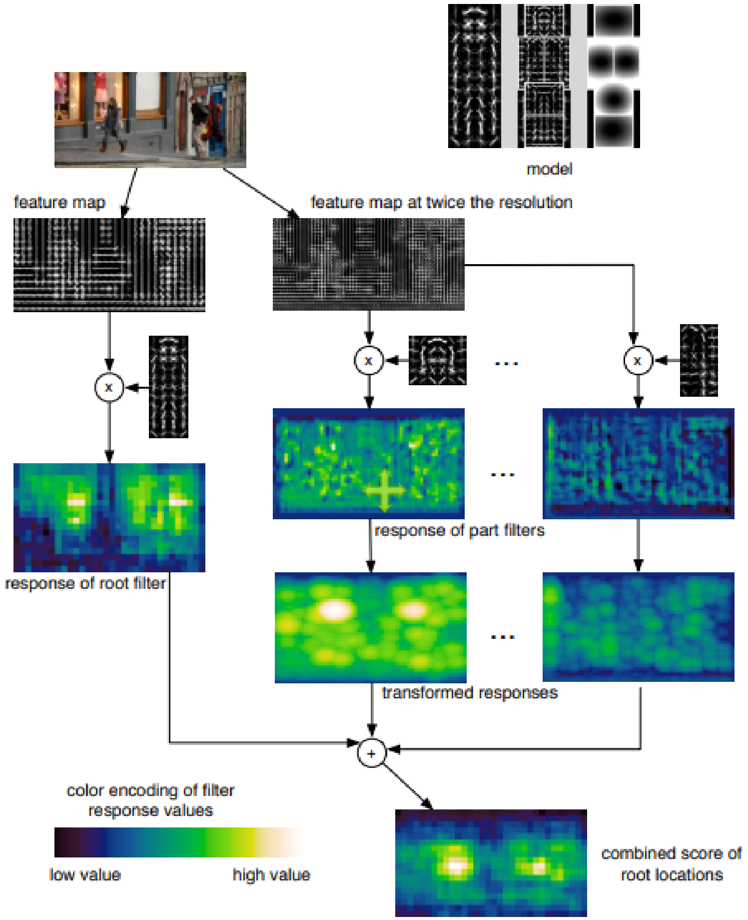
1. If “Confidence Score” less then threshold -> remove
2. Descending sort for “Confidence Score”
3. Select the box with highest “Confidence Score”
4. Compare the IOU this box with other boxes
5. Remove the bounding boxes with IOU higher then threshold(0.5)
6. Select the next box and Repeat 4-6



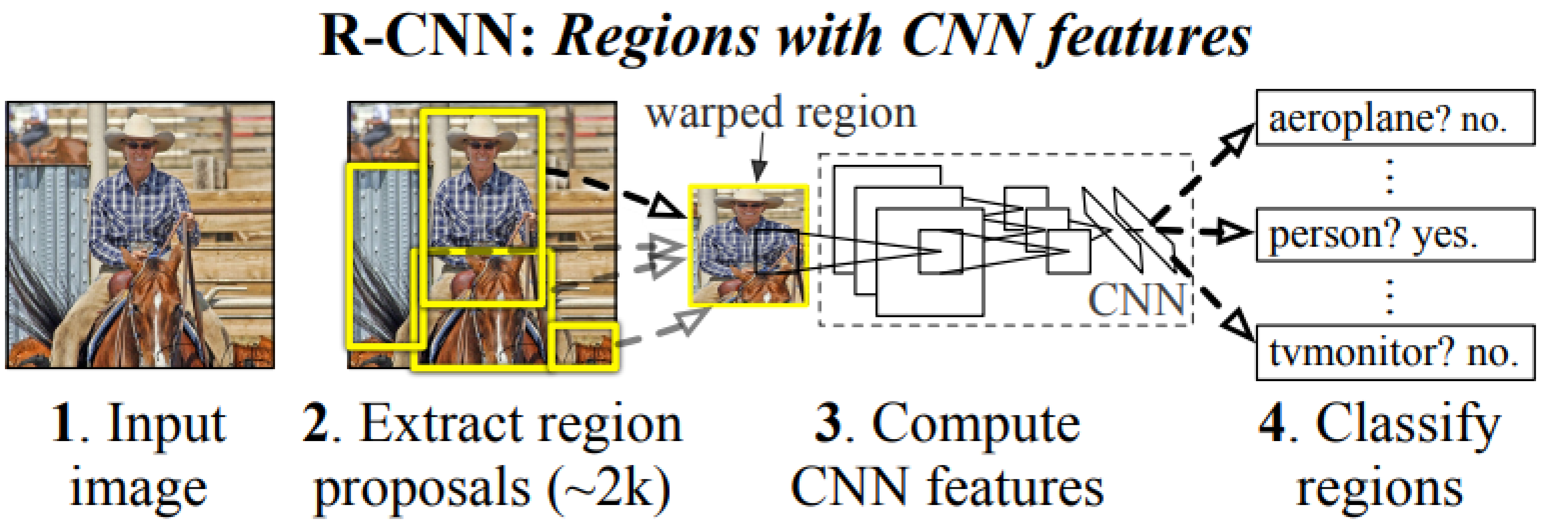
- NMS adds 2-3% in mAP (Mean Average Precision)

YOLO versus DPM





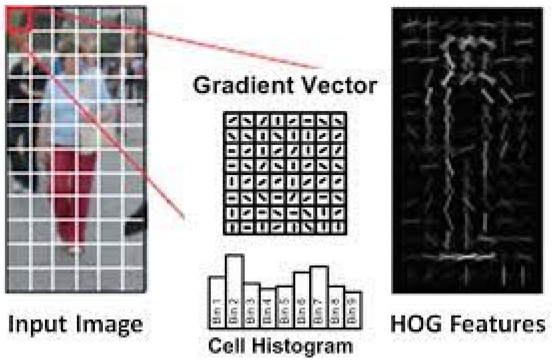
YOLO versus R-CNN



Selective Search

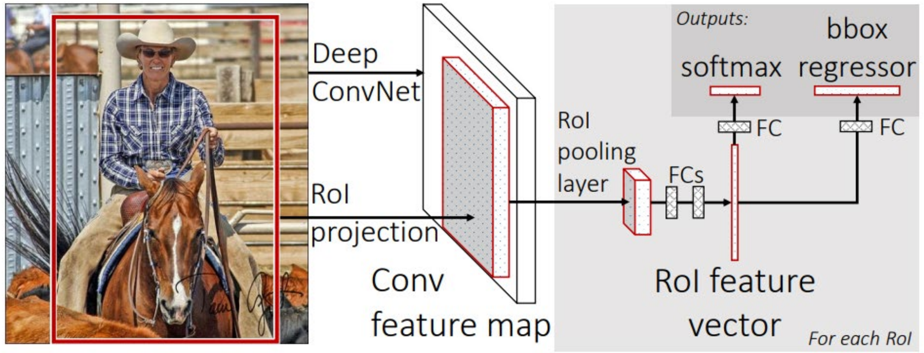
1. Generate potential bounding boxes
2. CNN extracts features
3. SVM score the boxes
4. Adjust bounding boxes and use NMS eliminates duplicate detections

YOLO versus Fast Detectors



"HOG Computation" for speeding up the DPM

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

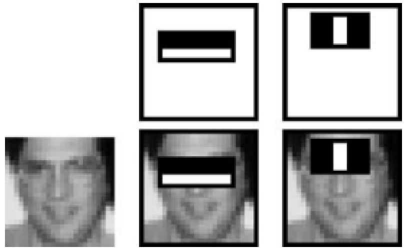


Fast R-CNN

- Speed up the classification stage of R-CNN
- Sharing computation

Faster R-CNN

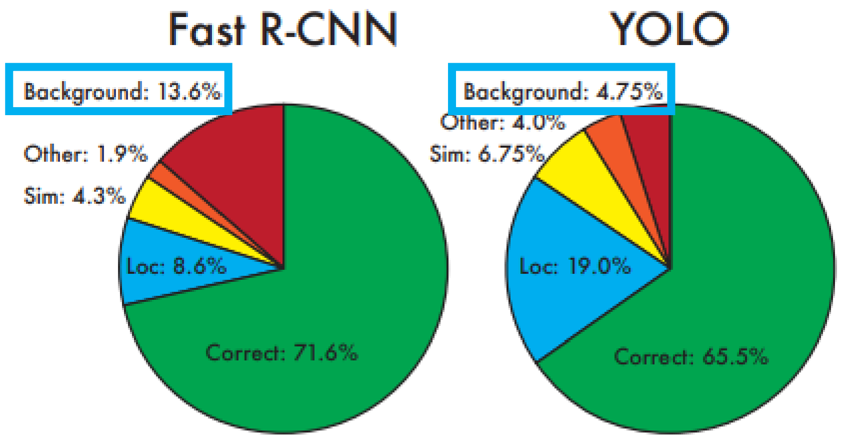
- Using Neural Network instead Selective Search



Robust Real-Time Face Detection

Comparison to Other Real-Time Systems

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21



- **FPS** mean Frame Rate
- GPU : TITAN X

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

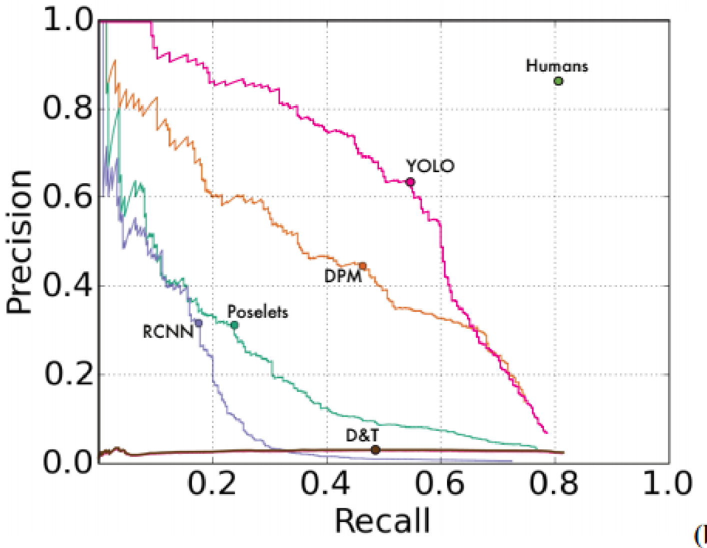
Combining Fast R-CNN and YOLO

	mAP	Combined	Gain
Fast R-CNN	71.8	-	-
Fast R-CNN (2007 data)	**66.9**	72.4	.6
Fast R-CNN (VGG-M)	59.2	72.4	.6
Fast R-CNN (CaffeNet)	57.1	72.1	.3
YOLO	63.4	**75.0**	**3.2**

VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR_CNN_MORE_DATA [11]	**73.9**	**85.5**	**82.9**	**76.6**	**57.8**	**62.7**	**79.4**	77.2	86.6	**55.0**	**79.1**	**62.2**	87.0	**83.4**	**84.7**	78.9	45.3	73.4	65.8	80.3	74.0
HyperNet_VGG	71.4	84.2	78.5	73.6	55.6	53.7	78.7	**79.8**	87.7	49.6	74.9	52.1	86.0	81.7	83.3	**81.8**	**48.6**	**73.5**	59.4	79.9	65.7
HyperNet_SP	71.3	84.1	78.3	73.3	55.5	53.6	78.6	79.6	87.5	49.5	74.9	52.1	85.6	81.6	83.2	81.6	48.4	73.2	59.3	79.7	65.6
Fast R-CNN + YOLO	70.7	83.4	78.5	73.5	55.8	43.4	79.1	73.1	**89.4**	49.4	75.5	57.0	**87.5**	80.9	81.0	74.7	41.8	71.5	68.5	**82.1**	67.2
MR_CNN_S_CNN [11]	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Faster R-CNN [28]	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
DEEP_ENS_COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	**68.8**	75.9	71.4
NoC [29]	68.8	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1
Fast R-CNN [14]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	**87.5**	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
UMICH_FGS_STRUCT	66.4	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2
NUS_NIN_C2000 [7]	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
BabyLearning [7]	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6
NUS_NIN	62.4	77.9	73.1	62.6	39.5	43.3	69.1	66.4	78.9	39.1	68.1	50.0	77.2	71.3	76.1	64.7	38.4	66.9	56.2	66.9	62.7
R-CNN VGG BB [13]	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3
R-CNN VGG [13]	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	80.4	71.6	74.0	60.0	30.8	63.4	52.0	63.5	58.7
YOLO	57.9	77.0	67.2	57.7	38.3	$\boxed{22.7}$	68.3	55.9	81.4	$\boxed{36.2}$	60.8	48.5	77.2	72.3	71.3	63.5	$\boxed{28.9}$	52.2	54.8	73.9	50.8
Feature Edit [33]	56.3	74.6	69.1	54.4	39.1	33.1	65.2	62.7	69.7	30.8	56.0	44.6	70.0	64.4	71.1	60.2	33.3	61.3	46.4	61.7	57.8
R-CNN BB [13]	53.3	71.8	65.8	52.0	34.1	32.6	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.9	59.3	54.1
SDS [16]	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7
R-CNN [13]	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6

Generalizability

Train Data : PASCAL VOC 2007
Precision : 모델이 True라 예측하였을 때 실제 True 비율
Recall : 실제 값이 True일 때 모델의 True 비율

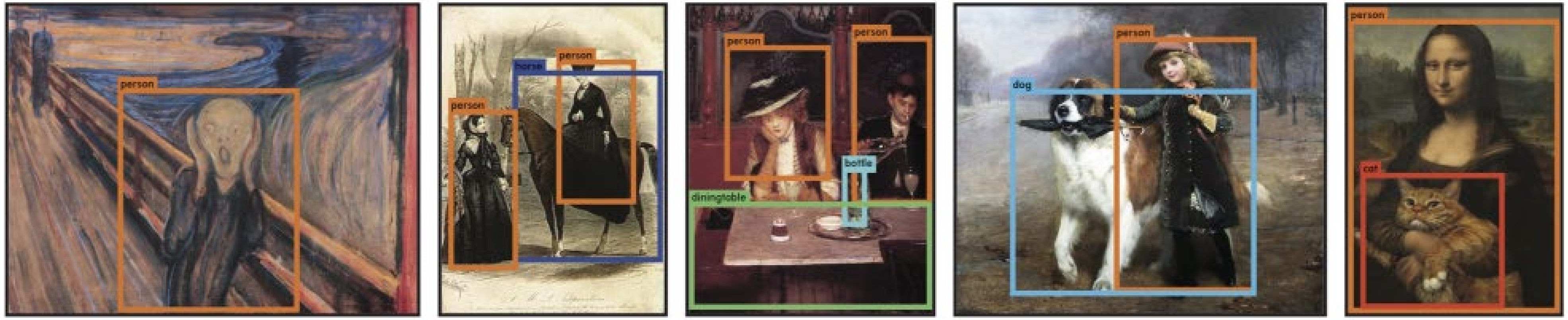


(a) Picasso Dataset precision-recall curves.

	VOC 2007	Picasso		People-Art
	AP	AP	Best F_1	AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32
Poselets [2]	36.5	17.8	0.271	
D&T [4]	-	1.9	0.051	

(b) Quantitative results on the VOC 2007, Picasso, and People-Art Datasets. The Picasso Dataset evaluates on both AP and best F_1 score.

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



Generalizability

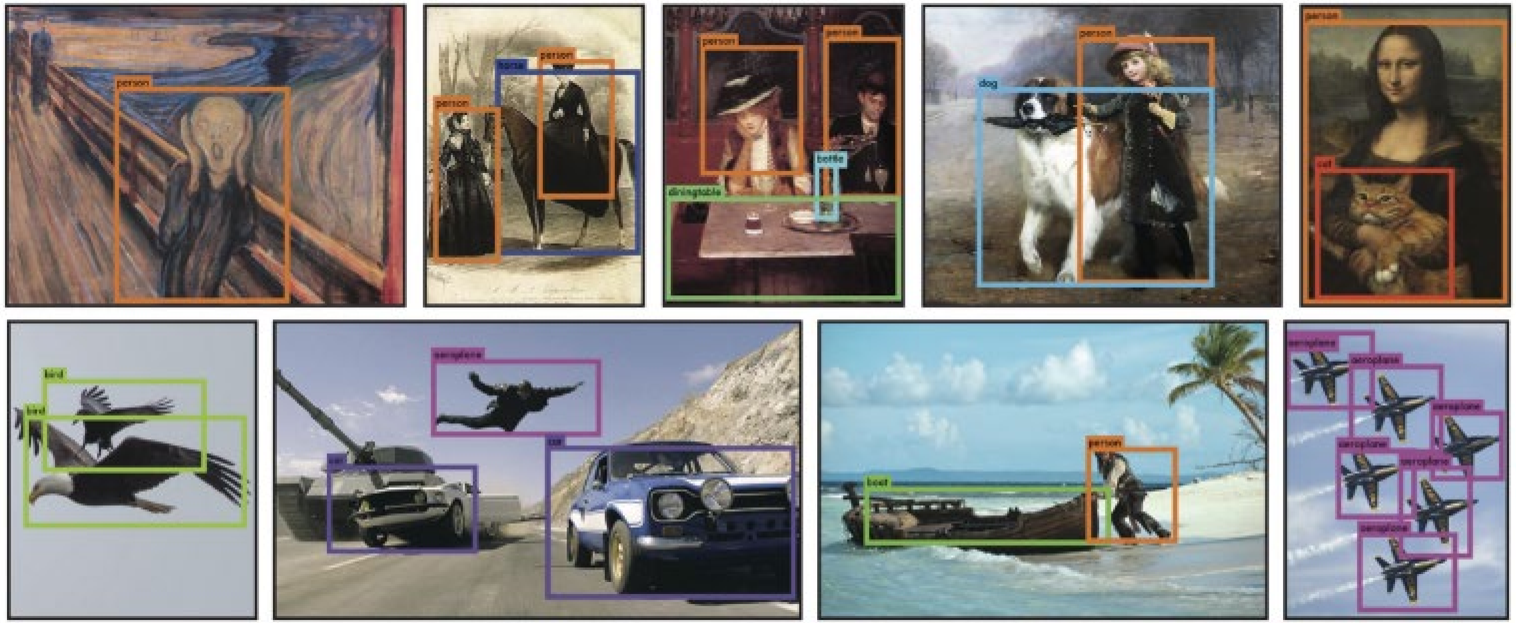


Figure 6: Qualitative Results. YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.

Limitations of YOLO

- Since each grid cell only predicts number of B boxes and only have one class
- 하나의 Grid cell이 각각 B개의 바운딩 박스와 하나의 클래스만을 예측.

  -> struggles with small objects that appear in groups

  -> 작은 물체가 그룹지어 나타나는 경우 어려움을 겪음

- model learns to predict bounding boxes
- YOLO는 바운딩 박스 예측을 학습함. (x, y, w, h, confidence Score가 Loss에 반영)

  -> struggles to generalize to objects in new or unusual aspect ratios or configurations

  -> 새로운 비율/배열을 가진 물체를 일반화 하기 어려워 함.

- Our loss function treats errors the same in small boxes versus large bounding boxes
- 우리의 Loss Function은 큰 박스와 작은 박스에 대해 동일한 Error를 취급.

  -> incorrect error about localizations

  -> 정확하지 않은 localization error 발생

Conclusion

- YOLO is simple and fast (Unified detection)
- YOLO enables real-time detection
- YOLO generalizes well to new domains