

Ch2. Useful model

VGG : Very Deep Convolutional Networks for Large-Scale Image Recognition

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

ResNet : Deep Residual Learning for Image Recognition

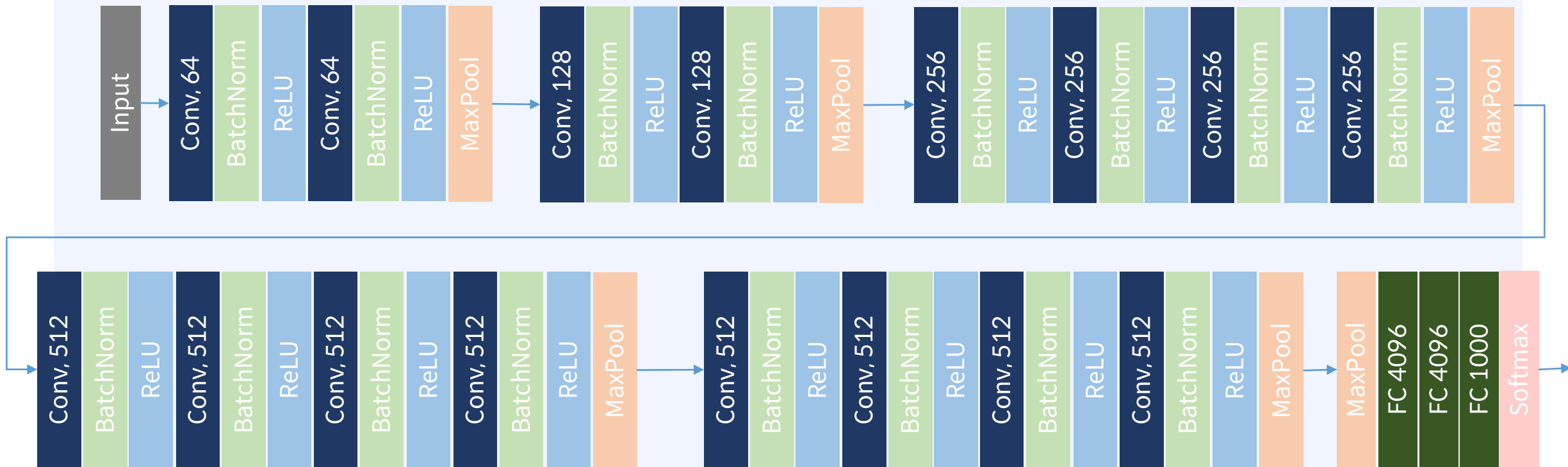
DenseNet : Densely Connected Convolutional Networks

VGG, ResNet, DensNet

classification

VGG

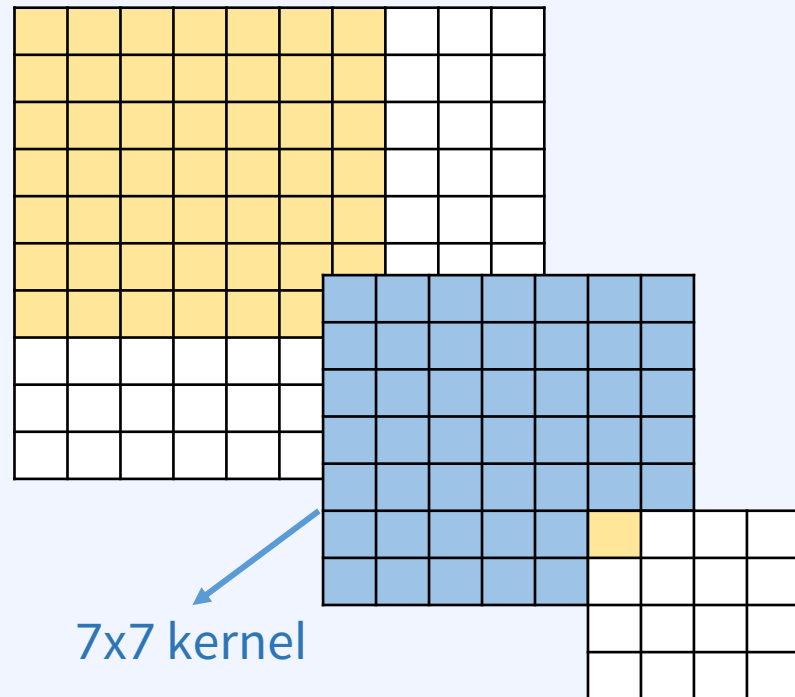
- 모든 Convolution layer에서 3x3 kernel 사용
- 16-19 에 달하는 깊은 신경망을 학습 가능
- VGG-19 -16 Convolution Layers + 3 Fully-connected Layers



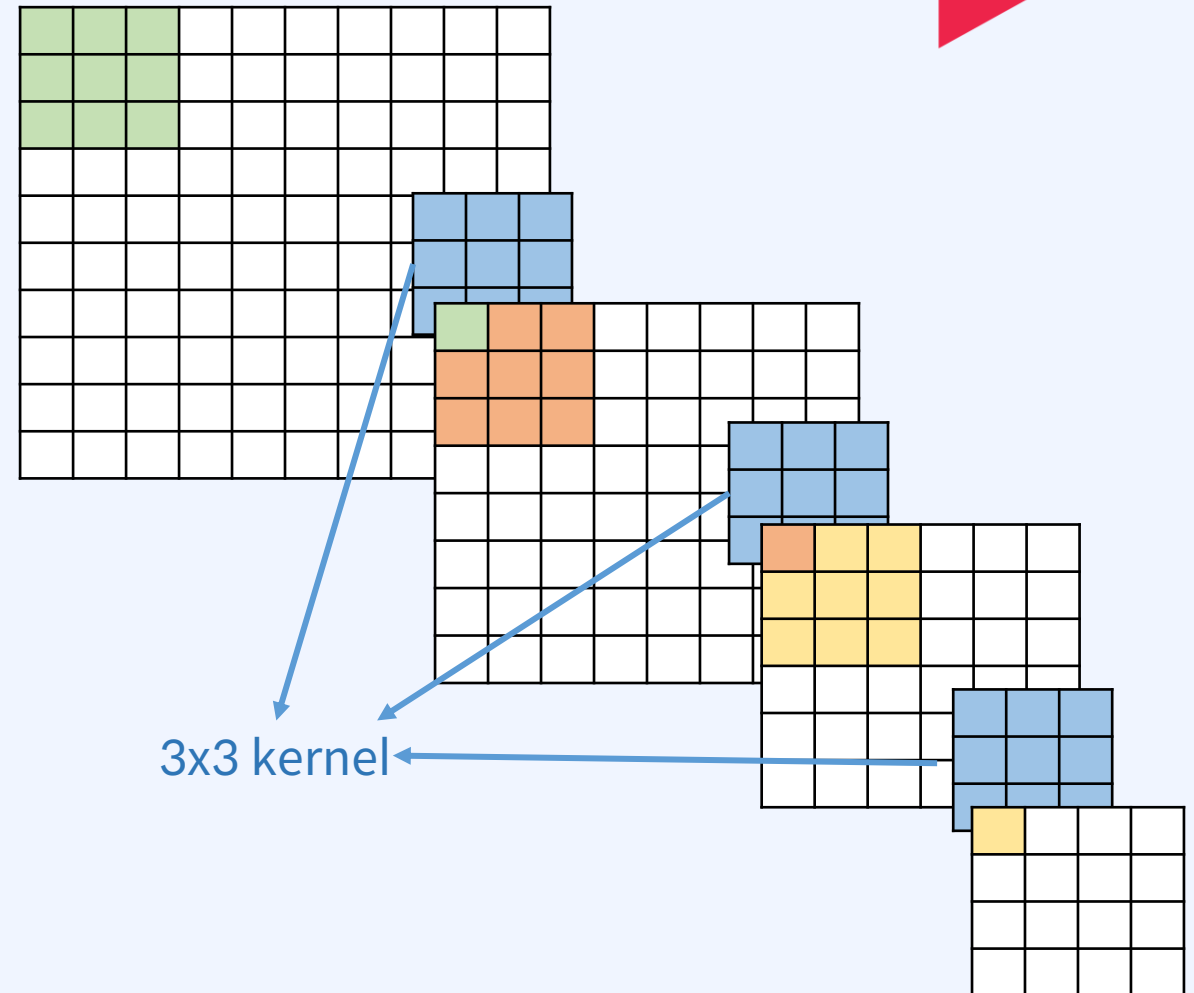
VGG-19 Architecture

3x3 convolution filtering

7x7 kernel에 의한 Convolution

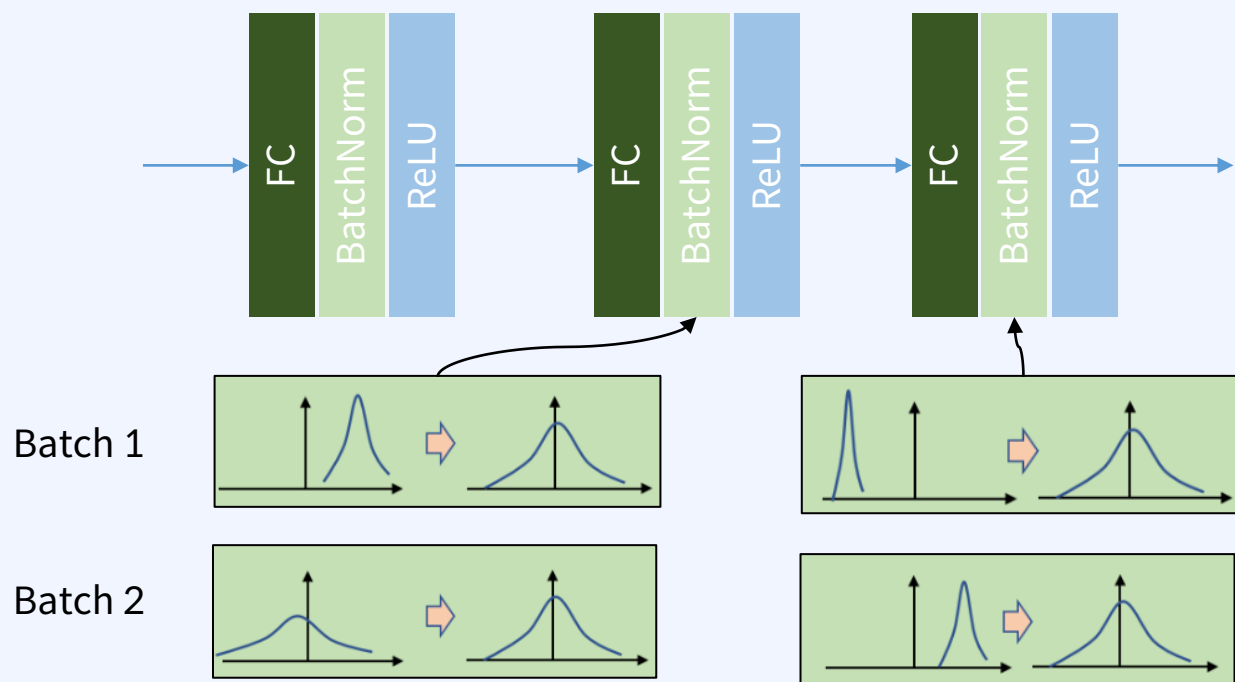


3x3 kernel에 의한 Convolution



Batch Normalization

- 학습속도를 빠르게 할 수 있음
- 가중치 초기화에 대한 민감도를 감소
- 모델의 regularization 효과



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots x_m\}$;

Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

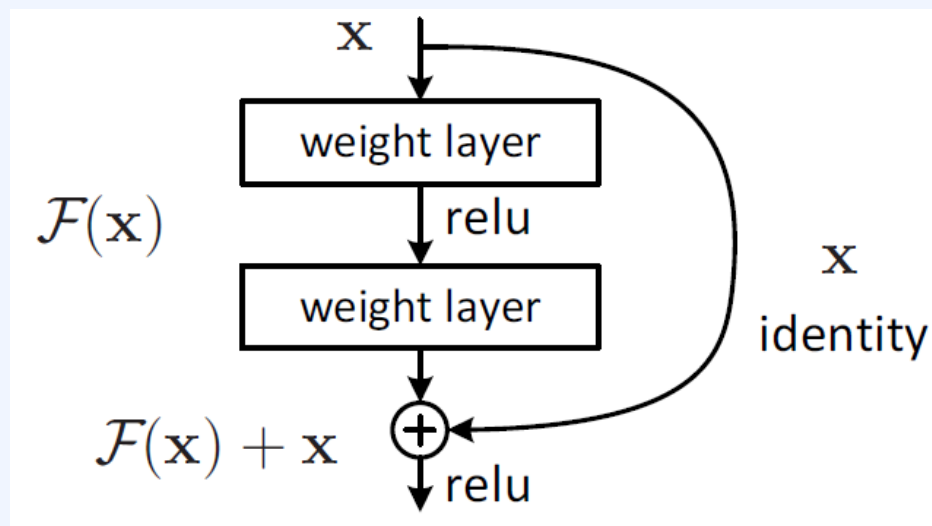
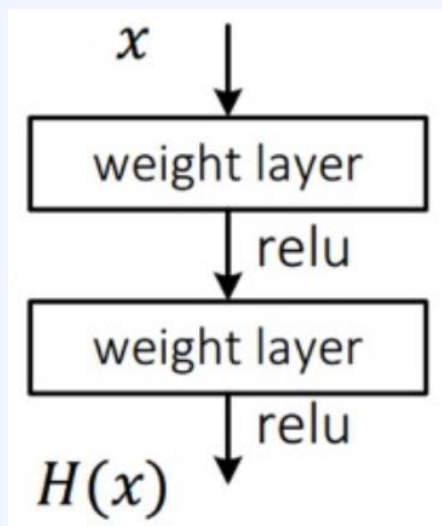
$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

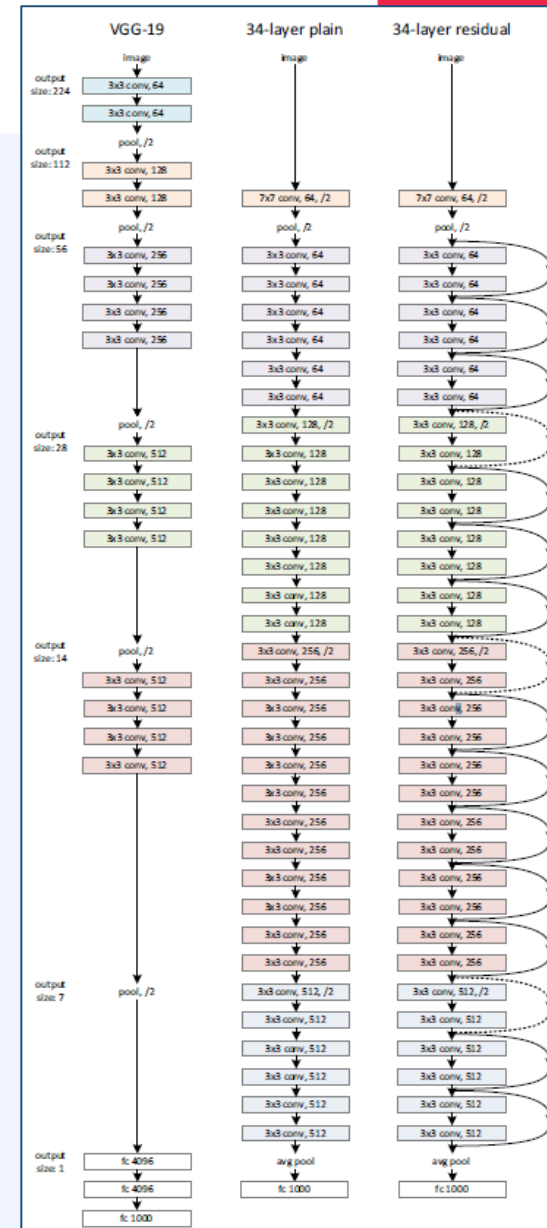
[2015' PMLR]

ResNet

- Residual function을 사용하여 쉽게 깊은 네트워크에서의 정확도 향상이 가능



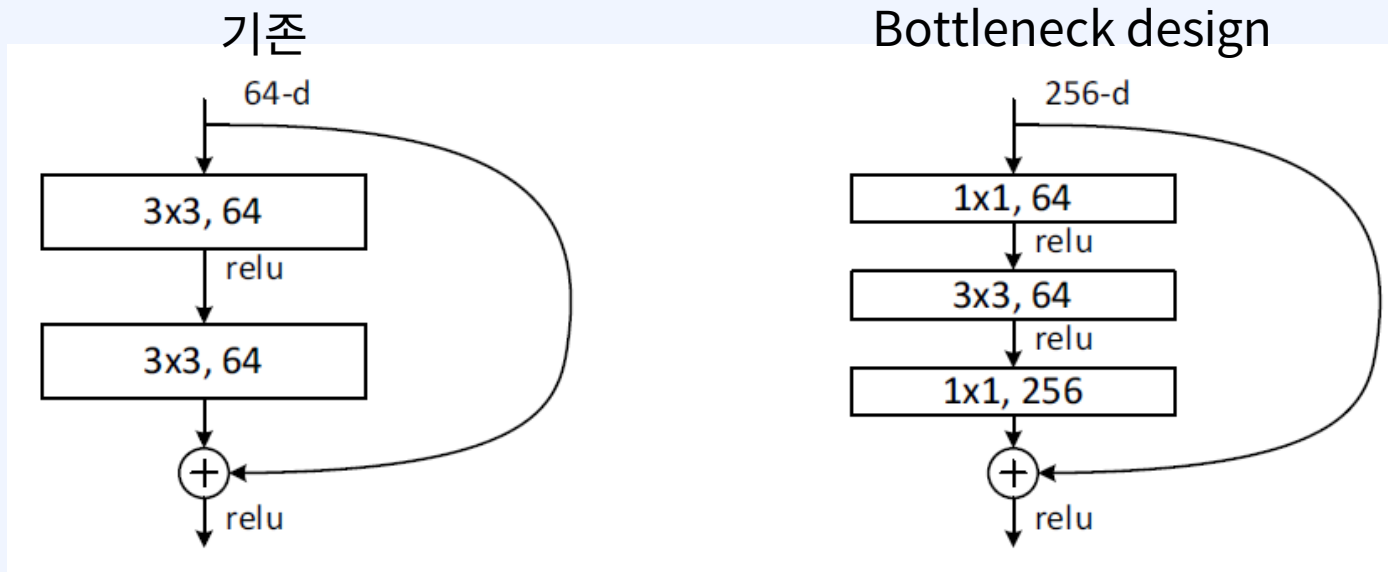
기존 네트워크와 ResNet의 구조



ResNet

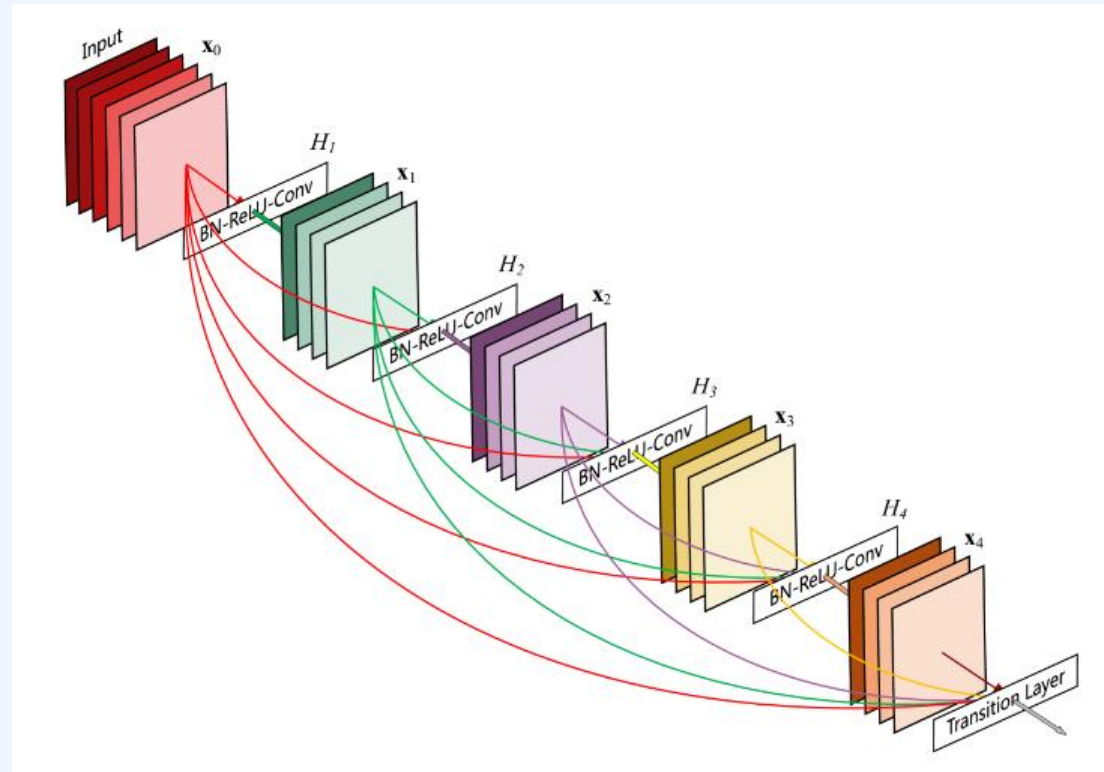
- bottleneck design

- 3-layer block(bottleneck design)을 사용하여 101-layer 및 152-layer ResNet을 구성



DenseNet

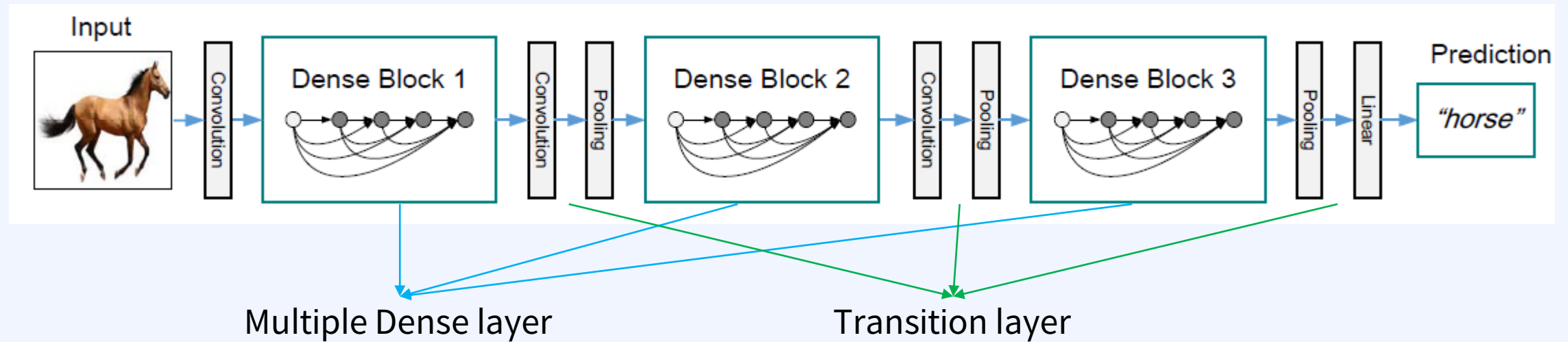
- Convolution 연산
 - $x_l = H_l(x_{l-1})$
- ResNet 연산
 - $x_l = H_l(x_{l-1}) + x_{l-1}$
- Dense connectivity
 - $x_l = H_l([x_0, x_1, \dots, x_{l-1}])$



5-layer dense block

DenseNet

- Pooling layer



Yolo

Object Detection

사물을 인식하는 다양한 문제 상황

- Object Detection
 - 다수의 Object가 존재하는 상황에서 각 Object의 위치와 클래스를 찾는 작업

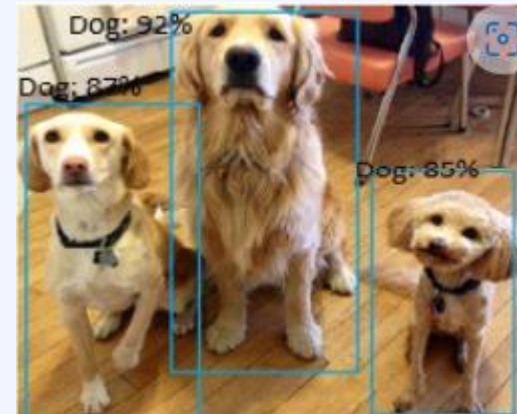
Classification



Classification
+ Localization



Object Detection



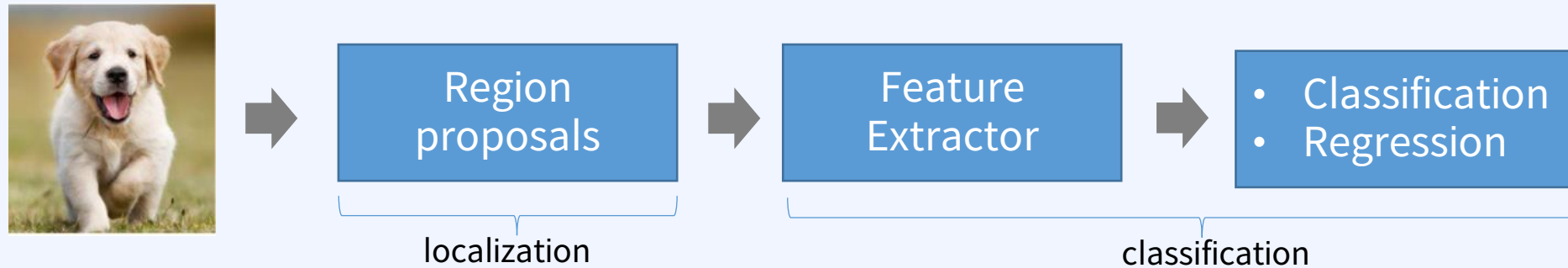
Single object

Multiple object

Object Detection 방식

- 2-Stage Detector

- 물체의 위치를 찾는 문제(localization)와 분류 문제(classification)를 순차적으로 해결

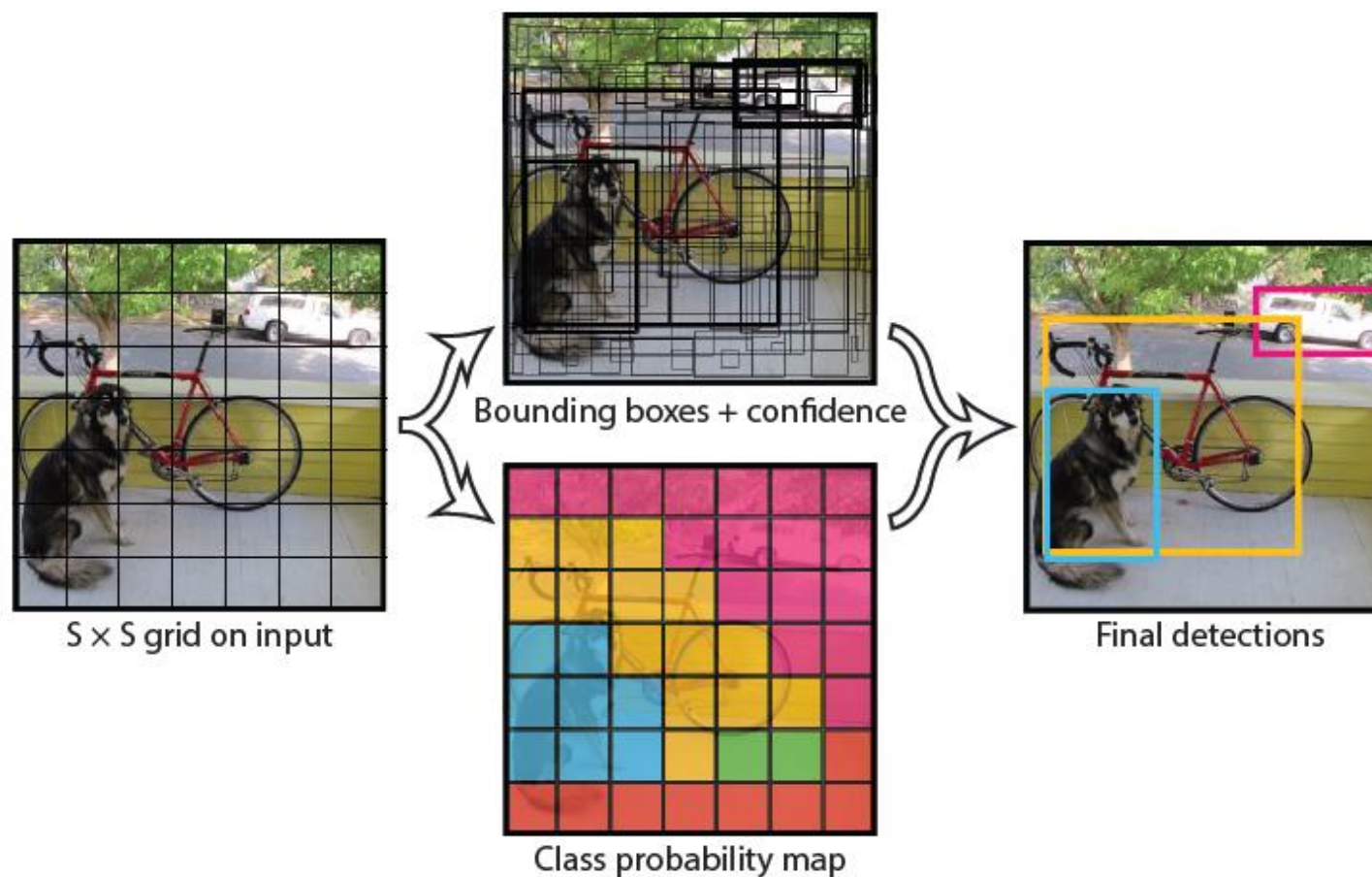


- 1-Stage Detector

- 물체의 위치를 찾는 문제와 분류 문제를 한번에 해결

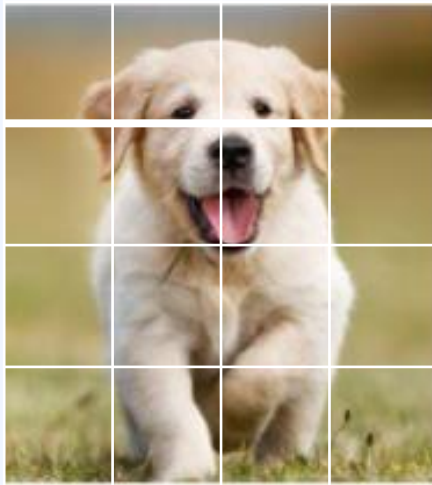


Unified Detection

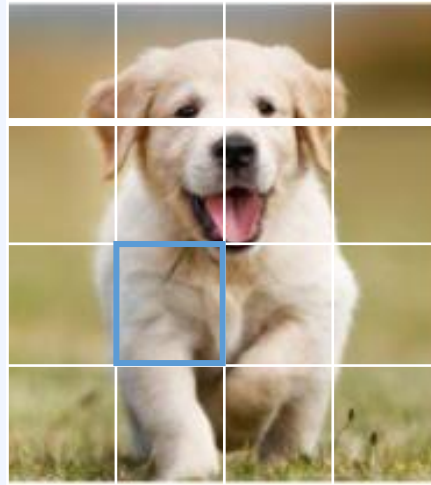


Unified Detection

예시) $S = 4$, $B = 2$, $C = 5$



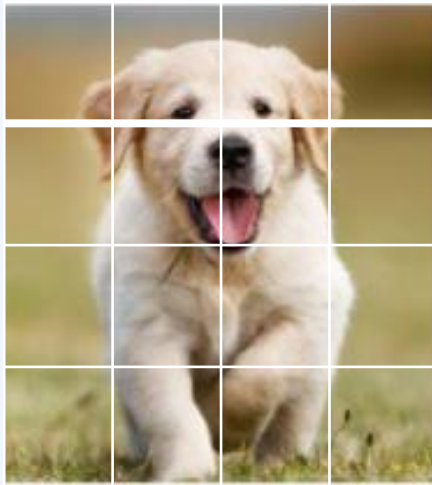
Resize image를
4x4 grid로 분할



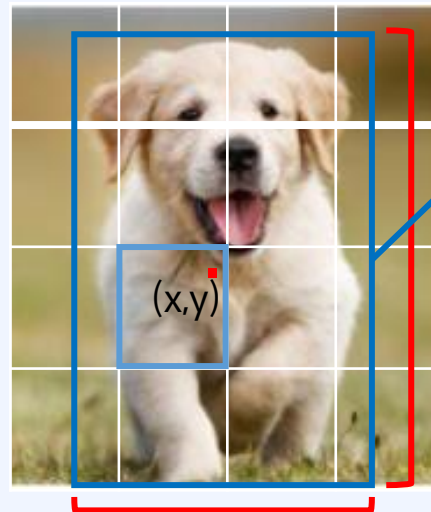
Grid cell 마다 bbox 2개씩 예측

Unified Detection

예시) $S = 4, B = 2, C = 5$



Resize image를
4x4 grid로 분할



Grid cell 마다 bbox 2개씩 예측

bbox #1

x
 y
 w
 h
 p_c

Bbox의 중심 좌표의 위치
(grid cell 기준)

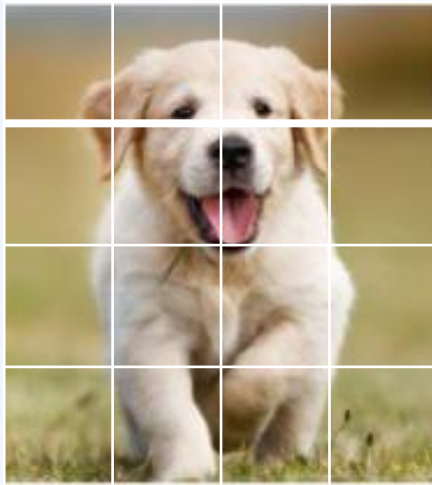
Input image W,H로 normalization

$p_c: \Pr(Object) \times IOU_{pre}^{truth}$

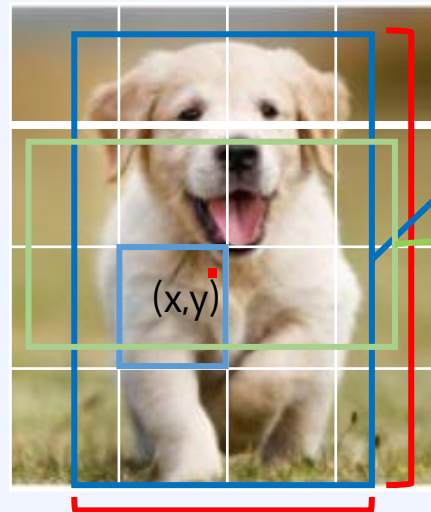
* $\Pr(Object)$: 물체가 bbox내에
있으면 1, 없으면 0

Unified Detection

예시) $S = 4, B = 2, C = 5$



Resize image를
4x4 grid로 분할



w/W

Grid cell 마다 bbox 2개씩 예측

bbox #1

bbox #2

h/H

(x,y)

x
 y
 w
 h
 p_c
 x
 y
 w
 h
 p_c

Bbox의 중심 좌표의 위치
(grid cell 기준)

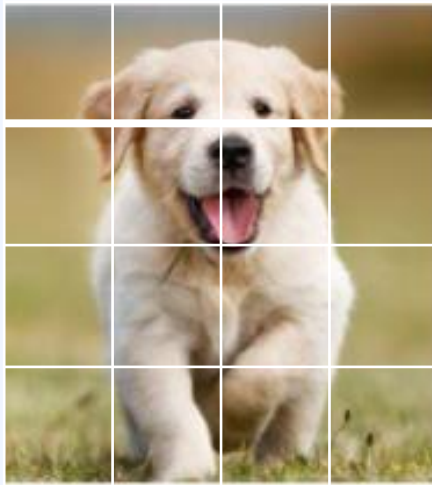
Input image W,H로 normalization

$p_c: \Pr(\text{Object}) \times IOU_{pre}^{truth}$

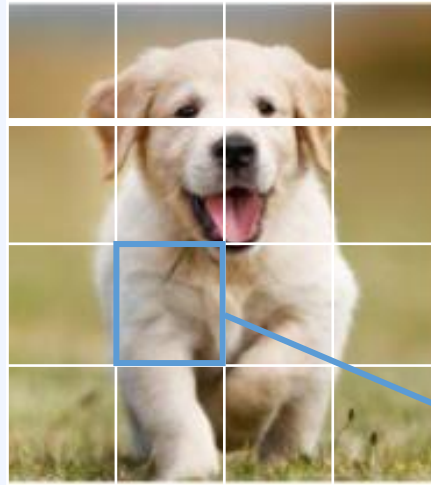
* $\Pr(\text{Object})$: 물체가 bbox내에
있으면 1, 없으면 0

Unified Detection

예시) $S = 4, B = 2, C = 5$



Resize image를
4x4 grid로 분할



Grid cell 마다 bbox 2개씩 예측

bbox #1

bbox #2

x
 y
 w
 h
 p_c
 x
 y
 w
 h
 p_c
 c_1
 c_2
 c_3
 c_4
 c_5

bbox의 중심 좌표의 위치
(grid cell 기준)

Input image W,H로 normalization

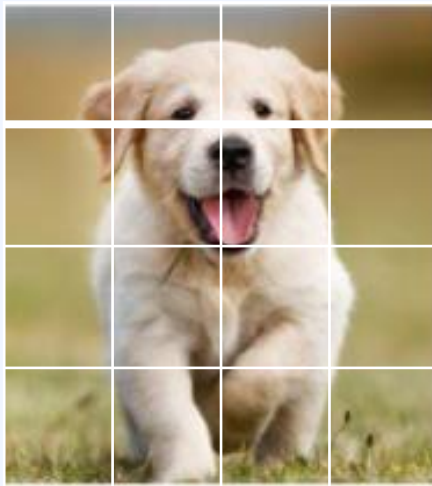
$p_c: \Pr(Object) \times IOU_{pre}^{truth}$

* $\Pr(Object)$: 물체가 bbox내에
있으면 1, 없으면 0

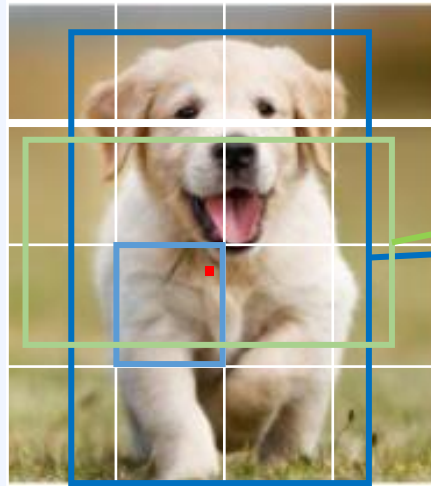
$\Pr(Class_i | Object)$: 객체가
bbox내에 있을 때, Grid cell에 있는
object가 i번째 class에 속할 확률

Unified Detection

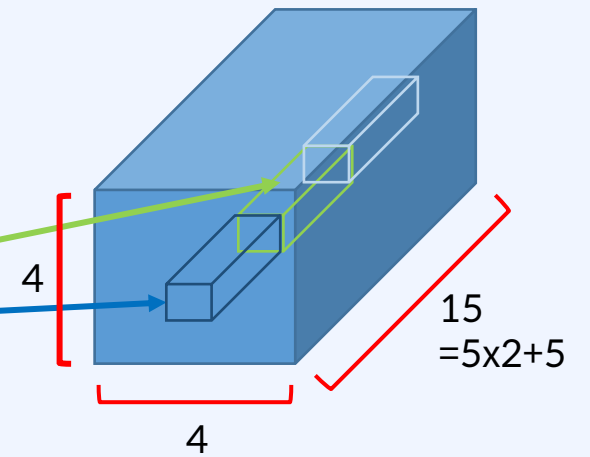
예시) $S = 4$, $B = 2$, $C = 5$



Resize image를
4x4 grid로 분할



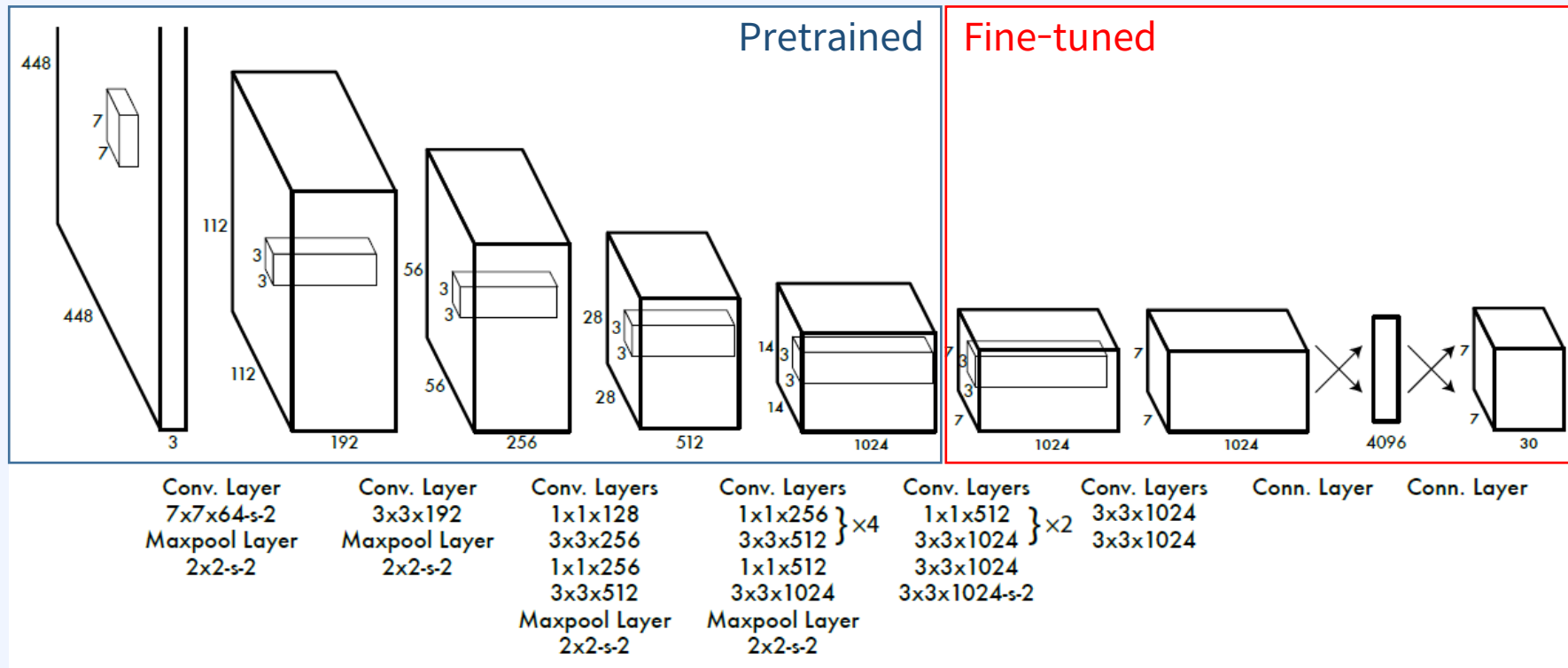
Grid cell 마다 bbox 2개씩 예측



Output

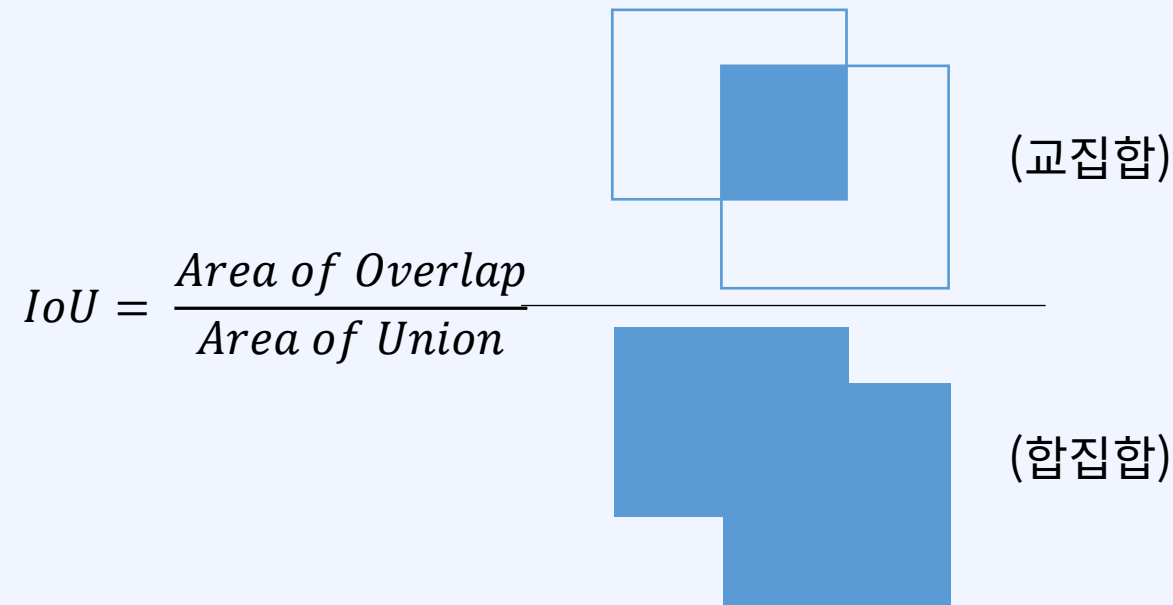
Yolo architecture

- 24 conv layer + 2FC layer
- 20 conv layer : Pretrained with 224x224 input image
- 4 conv layer + 2FC: Fine-tuned with 448x448 input image



Intersection over Union(IoU)

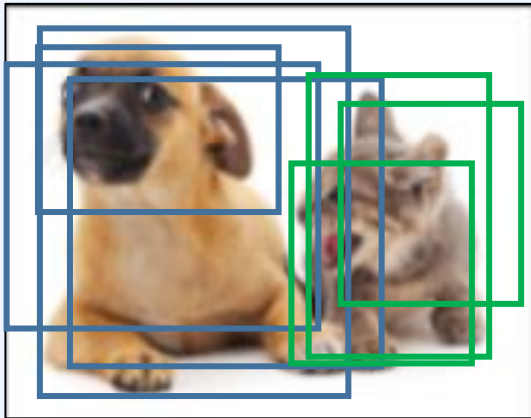
- IoU – 두 Bounding Box가 겹치는 비율
 - 성능 평가 예시 : mAP@0.5는 정답과 예측의 IoU가 50% 이상 때 정답으로 판정
 - NMS 계산 예시: 같은 class끼리 IoU가 50% 이상일 때 낮은 confidence의 box를 제거



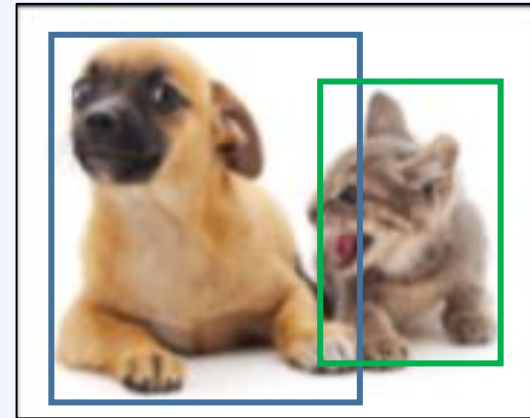
NMS(Non Maximum Suppression)

- Object Detection에서는 하나의 instance에 하나의 bounding box가 적용되어야 함
- 여러 개의 bounding box가 겹쳐있는 경우 하나로 합치는 방법이 필요

특정 threshold 이상인 중복 box



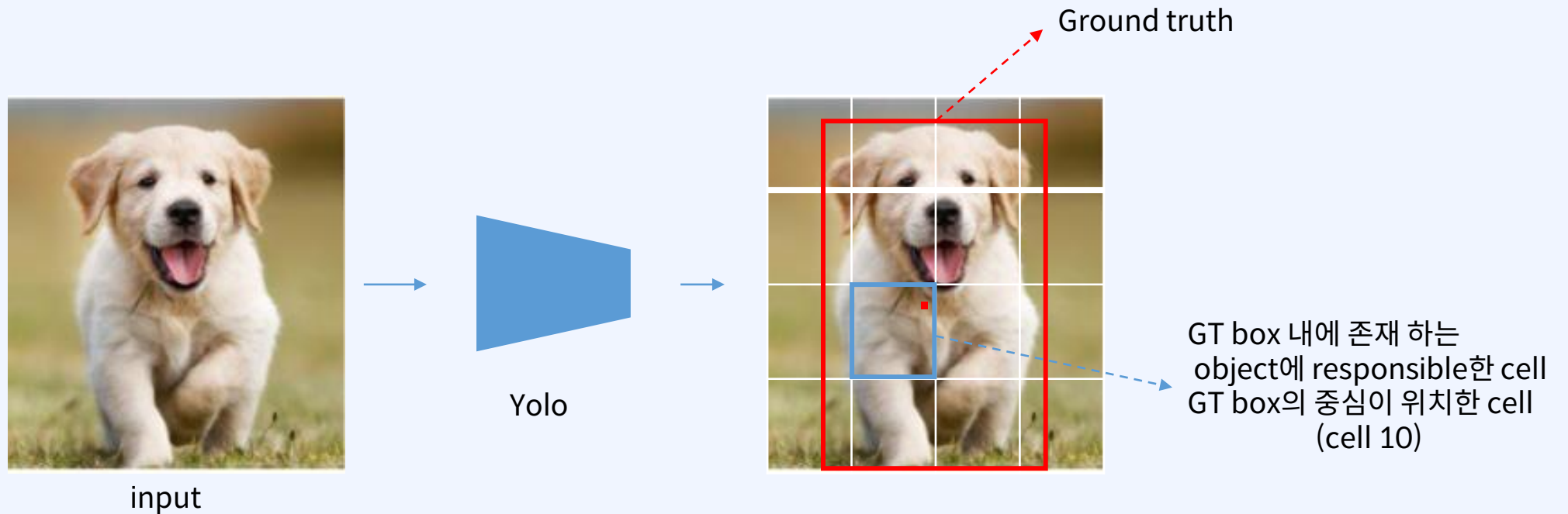
가장 높은 IoU를 갖는 box를 남김



NMS

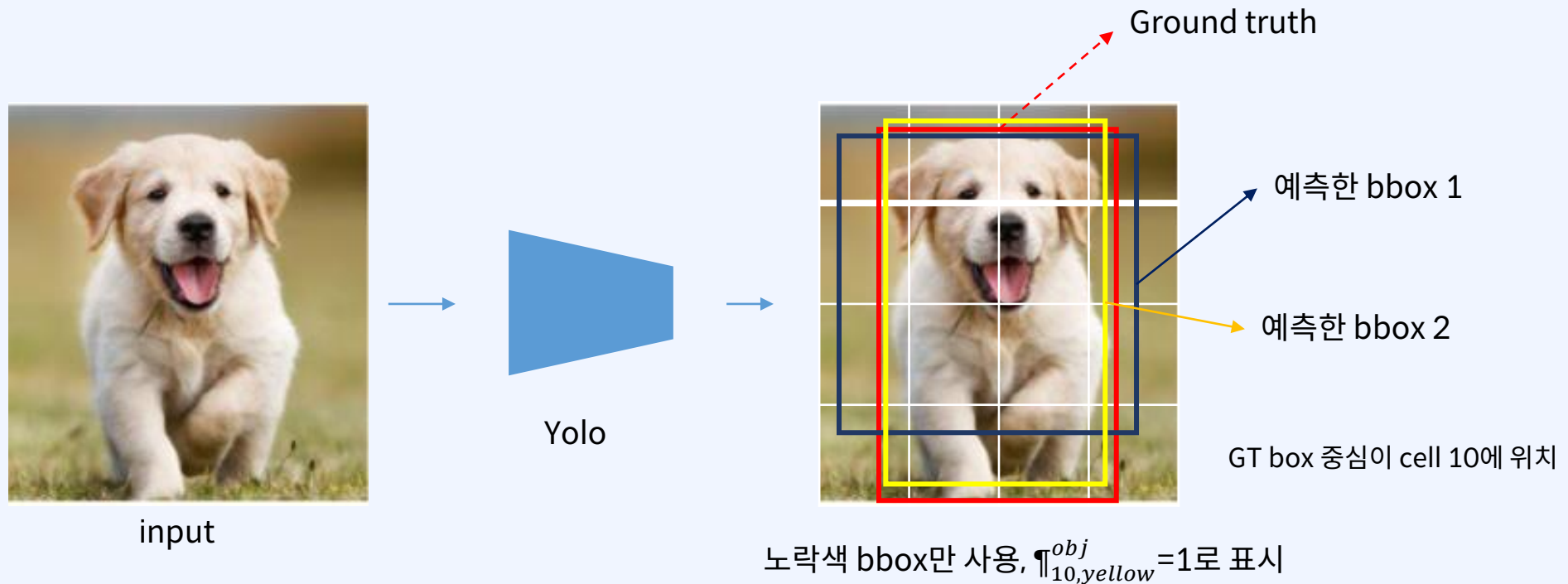
Training

- Object에 대한 responsible한 cell는 GT(ground truth) box의 중심에 위치한 cell로 할당



Training

- Yolo는 여러 bbox를 예측 \rightarrow 학습단계에서 IOU_{pred}^{true} 가 가장 높은 bbox 1개만 사용
 $\rightarrow \mathbb{1}_{ij}^{obj}$ 로 cell i에서 responsible한 j번째 bbox를 표시하여 loss function에 반영



$$IOU_{yellow}^{GT} > IOU_{blue}^{GT}$$

Loss function

$\mathbb{1}_{ij}^{obj}$ i번째 그리드 셀의 j번째 바운딩 박스가 객체 하나를 책임지고 있는지 여부(1 or 0)

λ_{coord} localization의 비중을 늘리기 위한 값(논문에서는 5)

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

바운딩박스의 크기에 따른 오차 차별을 없애기 위해 루트를 씌움

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

Confidence score = $\text{Pr}(\text{Object}) \times IOU_{pred}^{true}$

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

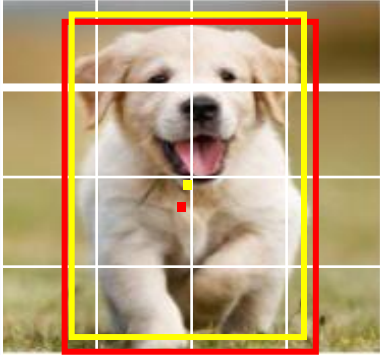
λ_{noobj} 객체가 없는 부분의 비중을 줄이기 위한 값(논문에서는 0.5)

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

$\mathbb{1}_i^{obj}$ i번째 그리드 셀에 객체가 존재하는지 여부

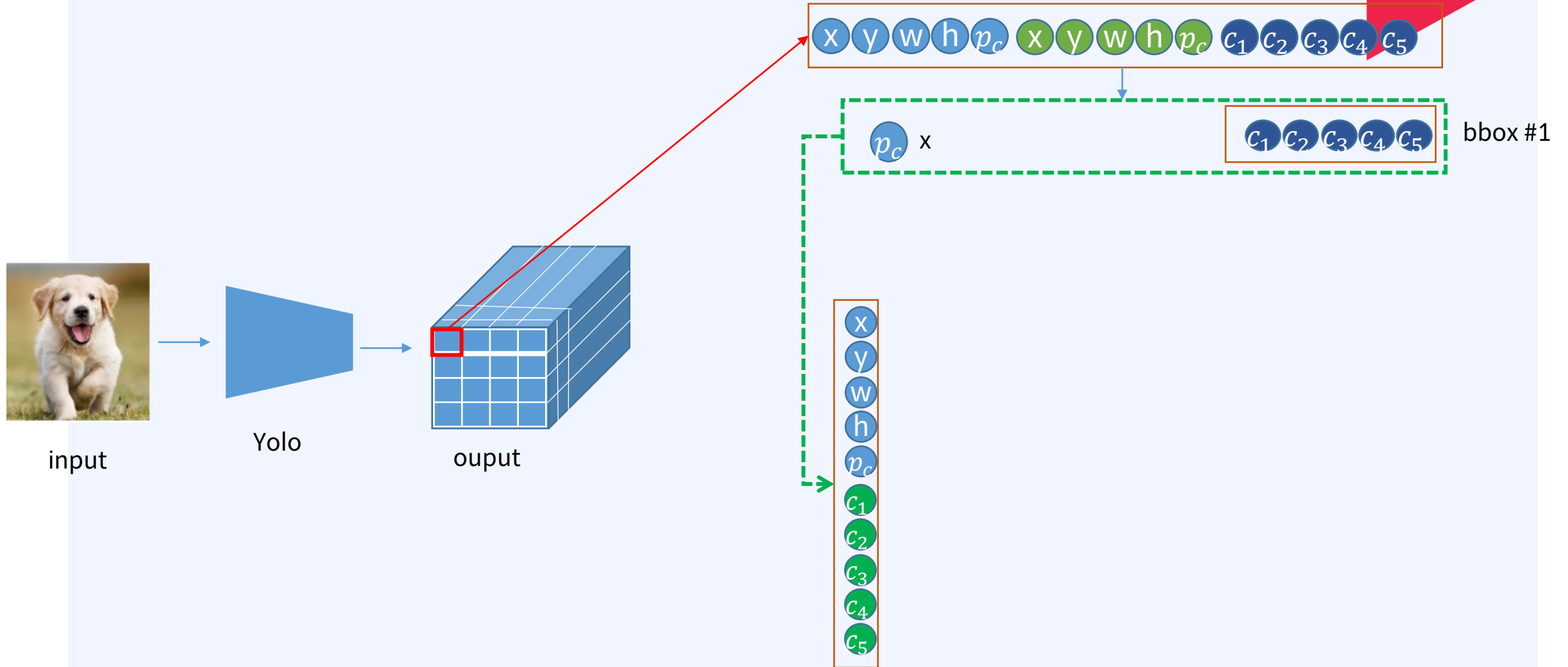
$$+ \sum_{i=0}^{S^2} \mathbb{1}_i^{obj} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

Conditional class probability = $\text{Pr}(\text{Class}_i | \text{Object})$

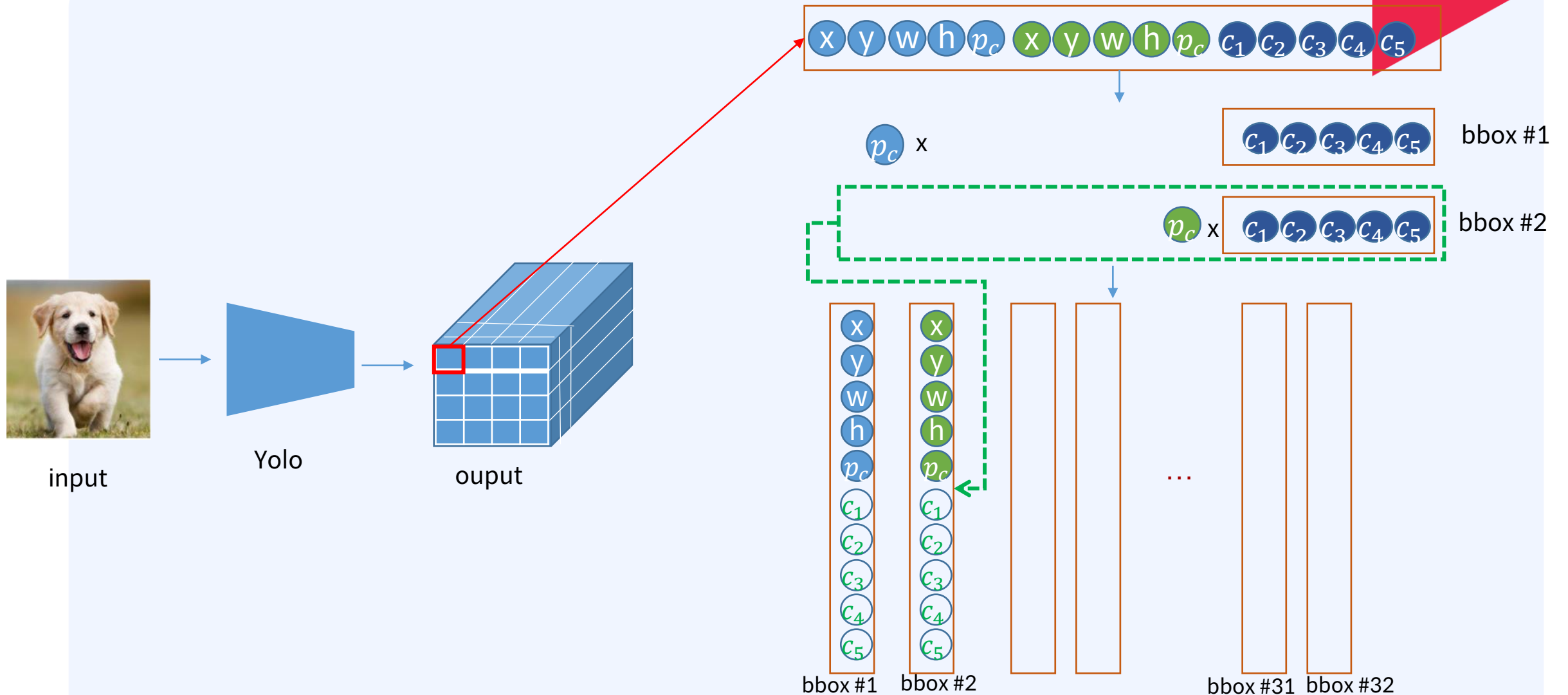


Grid cell에 object 존재하는 경우의 오차 & predictor box로 선정된 경우의 오차만 학습

Inference

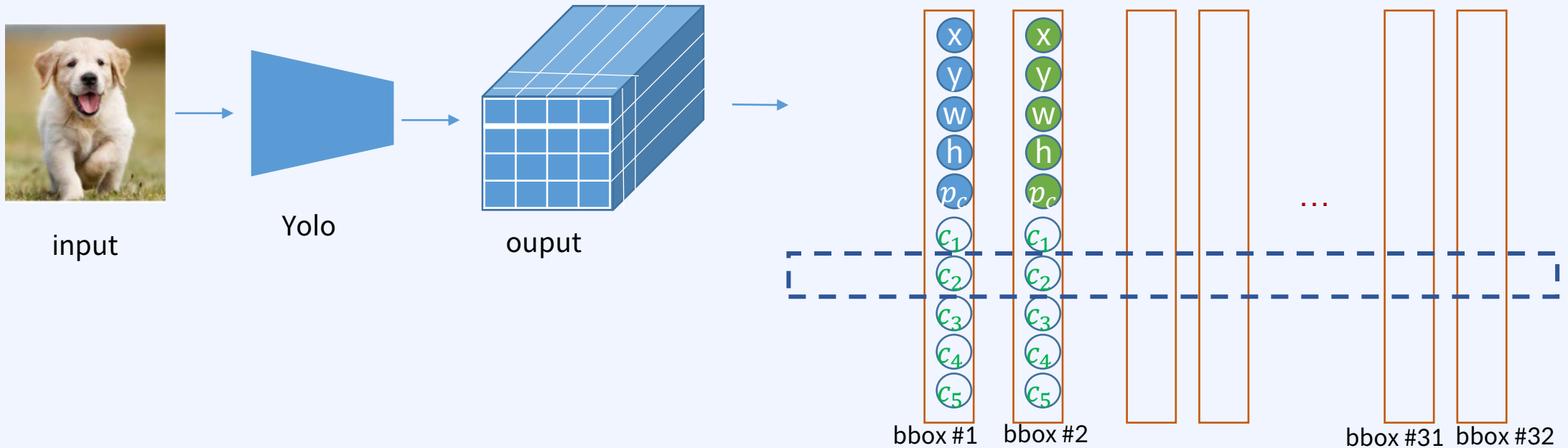


Inference



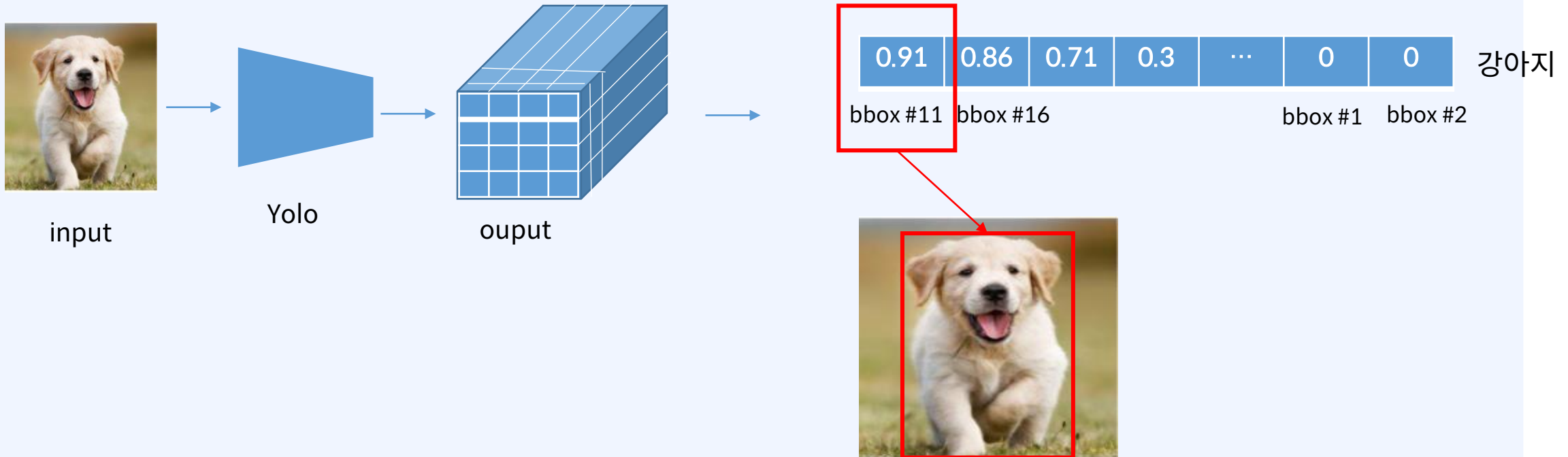
Inference

NMS를 적용 → 각 object에 대해 예측한 여러 bbox중에서 가장 예측력이 좋은 bbox만을 남기기 위함



Inference

NMS를 적용 → 각 object에 대해 예측한 여러 bbox중에서 가장 예측력이 좋은 bbox만을 남기기 위함



Summary