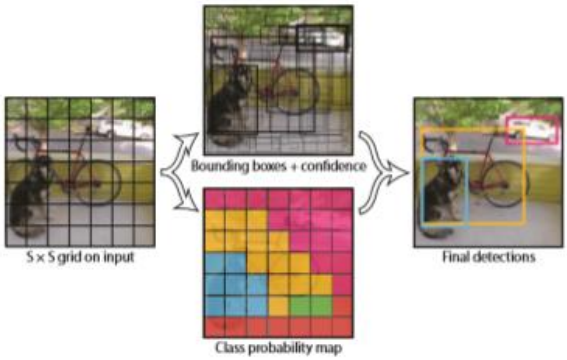
Abstract

We present YOLO, a new approach to object detection. 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 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second 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.



Purpose: Improving structure like human visual system.

Method: using convolutional network, calculating class probability of multiple bounding boxes.

Problem: lower precision (especially about a small object), occlusion problem, new type of bbox cannot be predicted correctly

Suggestion: proposing **a new detection method names YOLO.** It's fast, reasons globally about the image when making prediction, learns generalizable representations of objects.

(간단한 처리과정으로 속도가 매우 빠름, image 전체를 한번에 바라보는 방식으로 class에 대한 맥락적 이해도가 높음 -> background error가 낮음(false positive), object에 대한 좀 더 일반화된 특징을 학습)

1. Divide image into S x S grid cell
2. Each grid cell has # of B bounding box and confidence score of bounding box.
    A. Confidence Score:

        Pr(Object) * IOU_truth_predict
3. Each grid cell has # of C conditional class probability
    A. Conditional Class Probability:

        Pr(Class_i | Object)
4. Each bounding box has {x, y, w, h, confidence}
    A. (x, y): relative center position
    B. (w, h): relative width, height

Test time에는 conditional class probability와 bounding box의 confidence score를 곱하여 class specific confidence score를 얻는다.

$$
\begin{aligned}
ClassSpecificConfidenceScore &= ConditionalClassProbability * ConfidenceScore \\
&= Pr(Classi|Object) * Pr(Object) * IOU_{pred}^{truth} \\
&= Pr(Classi) * IOU_{pred}^{truth}
\end{aligned}
$$

- Network

YOLO network architecture은 GoogLeNet for image classification 모델 기반(24 Conv layers & 2 Fully Connected Layer)

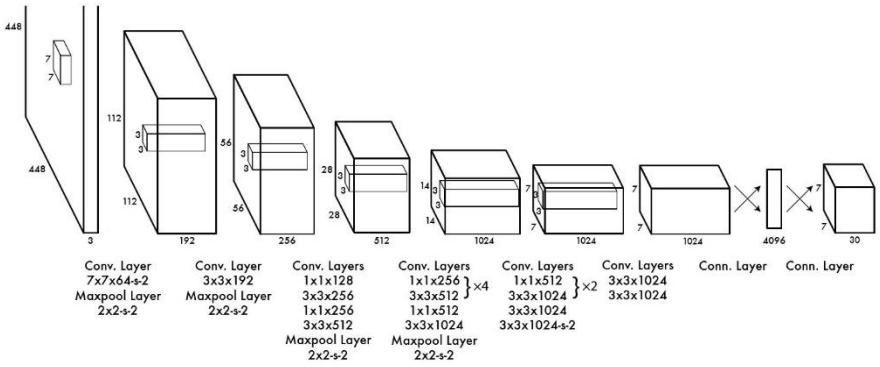
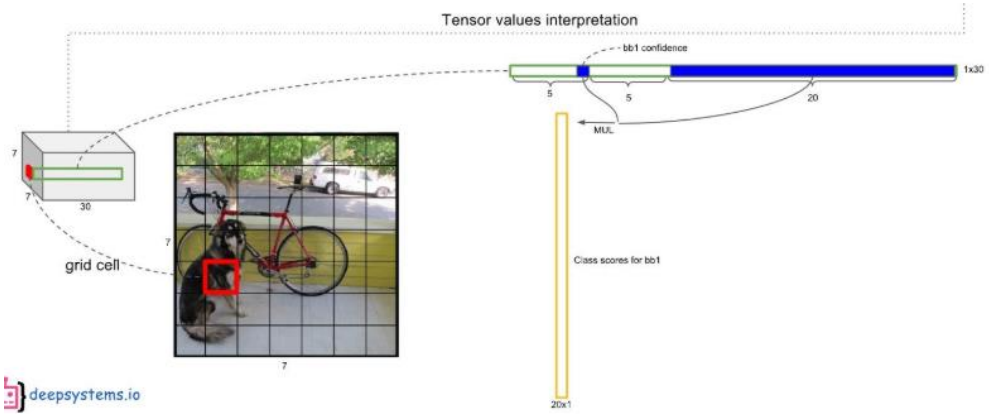
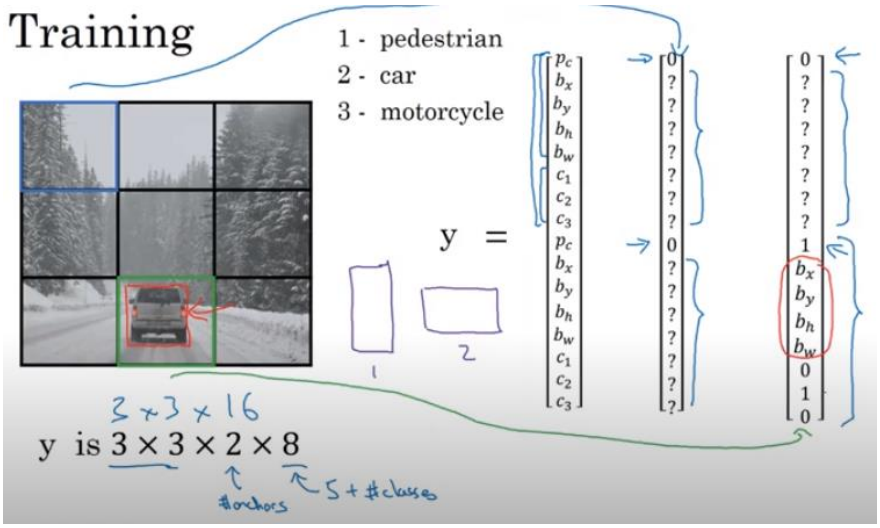


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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



- 첫번째 bounding box confidence score와 각 conditional class probability를 곱하면 첫번째 bounding box의 class specific confidence score가 나옴



- 이 계산을 각 bounding box에 대해 하게 되면 총 98개의 class specific confidence score를 얻게 됨
- 이에 대해 각 20개의 class기준으로 non-maximum suppression을 통해 object class/bounding box location을 결정
- 또한 Threshold by Object Confidence Score를 통해 box를 무시

Training process

1. **Grid cell의 여러 bbox중 ground-truth box와 IOU가 가장 높은 bounding box predictor로 설정**

$$\mathbb{1}_{ij}^{\text{obj}} \quad (1)$$

$$\mathbb{1}_{ij}^{\text{noobj}} \quad (2)$$

$$\mathbb{1}_{i}^{\text{obj}} \quad (3)$$

(1) Object가 존재하는 grid cell i의 predictor bbox j

(2) Object가 존재하지 않는 grid cell i의 bbox j

(3) Object가 존재하는 grid cell i

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

2. Loss Function

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

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

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

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

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

2.1

Object가 존재하는 grid cell i의 predictor bbox j에 대해 x, y loss 계산

2.2

Object가 존재하는 grid cell i의 predictor bbox j에 대해 w, h loss 계산

2.3

Object가 존재하는 grid cell i의 predictor bbox j에 대해 confidence score loss 계산

2.4

Object가 존재하지 않는 grid cell i의 bbox j에 대해 confidence score loss 계산

2.5

Object가 존재하는 grid cell i에 대해, conditional class probability loss 계산

3. Training

- **imgNet 1000 class dataset으로 20개의 conv layer pretraining**
- **pretraining이후 4 conv layer, 2 fc layer 추가**
- **bbox w, h를 image w, h로 normalize(0-1)**
- **bbx x, y는 특정 grid cell 위치 offset값 사용(0-1)**
- $\lambda_{coord}=5$, $\lambda_{noobj}=0.5$
- batch size = 64
- momentum = 0.9, decay = 0.0005
- lr: 0.001 ~ 0.01 천천히 상승
- drop rate: 0.5
- leaky LeRU 사용

출처) https://curt-park.github.io/2017-03-26/yolo/