YOLO Final-report

ByeongGeun Shin[1],
[1]Pusan National University.

1 You Only Look Once(YOLO) Paper Summary

1.1 Intro

You Only Look Once(YOLO)논문은 Object Detection 모델에 대한 논문이다. YOLO이전의 R-CNN[2]은 이미지 안에서 bounding box를 생성하기 위해 sliding window를 사용하여 region proposal 방식으로 사진의 부분부분을 확인한다. bounding box에 classifier를 적용하여 분류한 뒤 bounding box를 조정하고, 중복된 검출을 제거하고, 객체에 따라 box의 점수를 재산정하기 위해 후처리를 한다. 반면 YOLO는 이미지 전체를 한눈에 보고 regression으로 multi task를 한번에 처리한다. region proposal 같은 과정이 없다. 이렇게 이미지 전체를 놓고 한 번에 통과시키기 때문에, fps가 더 빨라졌다. 하지만 속도에서는 엄청난 향상을 이뤘지만 아직 정확도에서는 한계가 있다고 지적하고 있다.

*fps = frames per second. 초당 처리하는 frame 수.

1.2 Background

Object Detection에는 Image Classification과 Localization을 구분 짓는 Two-stage Detector와 두 과정을 구분 짓지 않는 One-stage Detector가 존재한다. Two-stage Detector는 성능이 뛰어나지만, 속도가 느리다. 대표적인 모델로 Faster R-CNN[1]이 있는데, 실행 속도가 5FPS이다. 반면, YOLO는 One-Stage Detector로 그 속도가 45FPS(Fast-YOLO는 150FPS)로 매우 빠르다. 두 방식의 차이는 Figure-1 에서 한눈에 볼 수 있다.

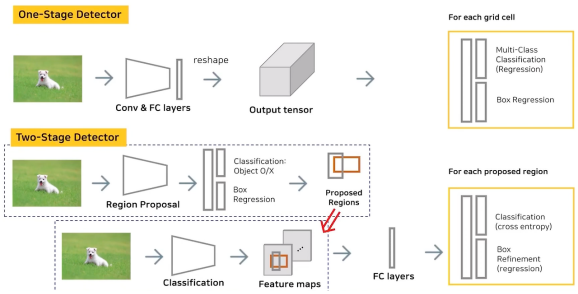


Figure 1: Object Detection Stage.

1.3 Unified Detection

YOLO는 region proposal, feature extraction, classification, box regression 과정들을 one-stage detection에 끝낸다.

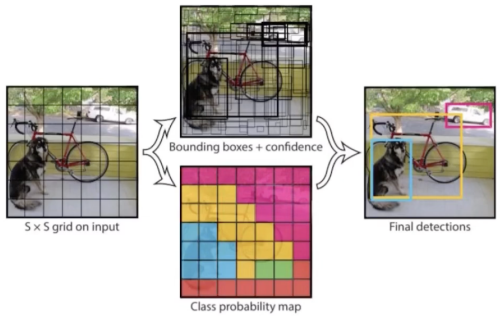


Figure 2: YOLO Detection.

우선 Input 이미지를 S x S 크기의 Grid로 나누고, 한 Grid Cell 안에 어떤 객체의 중심이 포함되면, 해당 Grid Cell이 그 객체에 대해 책임을 지고 탐지를 수행한다. 각 Grid Cell은 B 개의 Bounding Box와 Confidence Score를 예측한다. Confidence Score는 박스 안에 객체가 포함될 확률과, 박스가 객체를 얼마나 잘 담고 있는지에 대한 곱으로 표현된다. Pr(Object)는 객체가 포함되면 1, 없으면 0. 그래서 객체가 포함되지 않은 경우 전체가 0이 된다. 점수가 높기 위해서는 예측한 Bounding Box가 최대한 Ground truth box와 겹치도록 하여 1에 가까워야 한다.

Bounding Box는 4개의 값에 대한 예측으로 이루어져 있다. x, y는 Bounding Box의 중심 값으로 Grid Cell 안에서의 상대적인 좌표로 0에서 1 사이의 값을 갖게 된다.

각 Grid Cell은 C 개 클래스에 대한 Conditional Class Probabilities를 계산한다. 각 Bounding Box 안에 있는 객체가 어떤 클래스일지에 대한 조건부 확률이다.

이를 사용해서 최종적으로 구하는 값은 Class-specific Confidence Score이다. Class-specific Confidence Score는 Bounding Box안에 특정 객체가 속할 확률과 Bounding Box가 객체를 얼마나 잘 담고 있는지에 대한 정보를 표현하는 값이며, Equation-1과 같이 계산된다.

$$P(\text{Class}_i) = P(\text{Class}_i \mid \text{Object}) \times P(\text{Object}) \times \text{IOU}^{\text{truth}}_{\text{pred}} \quad (1)$$

1.4 Architecture

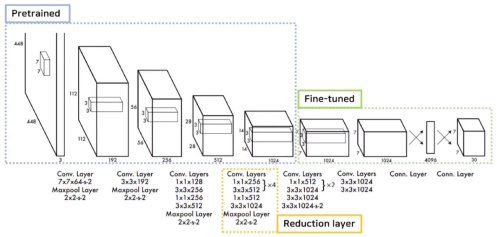


Figure 3: YOLO Architecture.

YOLO는 GoogLeNet[3]의 구조를 채택하여 총 24개의conv layers과 2개의 fully connected layers으로 구성되어 있다. 차이점은, GoogLeNet의 인셉션 구조 대신 YOLO는 1 x 1 축소 계층과 3 x 3 컨볼루션 계층의 결합을 사용했다. 1 x 1 축소 계층과 3 x 3 컨볼루션 계층의 결합이 인셉션 구조를 대신한다.

1.5 Training

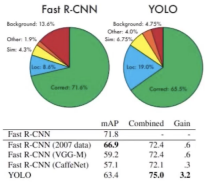
YOLO는 처음부터 모든 layer를 학습시키지 않는다.처음 20개의 convolution layer만 따로 떼어 뒷단에 2개의 layer만 붙인 뒤 ImageNet 2012 데이터셋에 대한 classification 문제를 학습시킵니다.이 과정을 통해 이미지에 대한 general한 특징들을 학습할 수 있게 될 것임. activation function으로는 leaky relu를 사용했으며 loss는 sum-squared error을 사용함. sum-squared error을 사용한 이유는 optimize를 하는 과정이 비교적 단순하기 때문이라고 소개함. 하지만 YOLO에서 단순히 sum-squared error을 사용했을 때에는 문제점이 생기게 되는데, 첫번째로 localization error(bounding box의 위치 정보에 대한 loss)와 classification error에 대한 weight를 동일하게 주는 것은 이상적이지 않음. 두번째는 모든 grid cell에 물체가 포함된 것은 아니다. 이것은 종종 confidence score 계산에 부정적인 영향을 끼침. 마지막으로 큰 박스의 오차와 작은 박스의 오차의 weight를 동일하게 주는 것은 이상적이지 않음. 이를 해결하기 위해 bounding box 위치 정보에 대한 loss에 대해 weight를 크게 주고, 물체가 포함되어 있지 않은 부분에 대한 confidence prediction에 대한 weight는 작게 주며, bounding box의 width와 height는 값을 그대로 쓰는 것이 아닌 square root(제곱근) 적용한 값으로 error을 처리함. Yolo는 여러 bbox를 예측하지만, 학습단계에서는 IOU(truth pred)가 가장 높은 bbox 1개만 사용한다. 1(obj ij)라는 스칼라 값으로 cell i번째에서 responsible 한 j번째 bbox 를 표시하여 loss function 에 반영한다. 정리하면, grid cell에 object 존재하는 경우의 오차와 predictor box로 선정된 경우의 오차만 학습하게 된다.

1.6 Inference

훈련 단계와 마찬가지로, 추론 단계에서도 테스트 이미지로부터 객체를 검출하는 데에는 하나의 신경망 계산만 하면 되기 때문에 굉장히 빠르다. 하지만 객체의 크기가 크거나 여러 객체가 겹쳐있는 경우 다중 검출 문제가 생기는데, 이를 NMS(non-maximal-suppression) 기법을 통해 mAP를 2 3% 개선하였다.

1.7 Experiment

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 [31]	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



	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**

Figure 4: YOLO Result.

다른 모델과 비교했을때, mAP 정확도는 약간 낮을 수 있지만 비슷한 성능에 추론 속도가 훨씬 높은 것을 볼 수 있다. Fast R-CNN은 localization을 잘하는 대신, backgroud error가 많다. 그렇기 때문에 이 둘을 합쳐서 사용하기도 한다. 이 둘을 어떻게 합치는 과정을 ensemble이라고 한다. 다른 R-CNN과 비교했을 때보다 YOLO를 합칠 경우 mAP의 증가가 크다.

한계점으로는, 하나의 grid cell 안에 bounding box의 수를 2개로 설정했기 때문에 YOLO는 작은 물체를 검출하는데 어려움을 겪는다고 한다. bounding box 내에서 물체를 예측하도록 학습 받았기 때문에, 조잡한 특징들로 물체를 검출할 수 밖에 없다. 그렇게 되니, 새로운 환경이나 다른 모습의 같은 물체를 검출할 때 어려움이 있다고 한다. 마지막으로는 또, 큰 상자와 작은 상자 간의 error이 같게 반영이 되는 점인데, 작은 상자일수록 error에 더 민감해야 한다. 하지만, 현재의 loss function에서는 이게 미미하게 반영이 된다고 한다.

2 You Only Look Once(YOLO) Practice

2.1 Build Model Architecture

이 실험에서는 PyTorch로 YOLO 모델을 구축하고 성능을 측정하였다. 그러나 오리지널 YOLO 네트워크와 달리 이 실험의 네트워크는 다른 구조적 변화를 포함하고 있다. 주된 차이점은 다음과 같다.

첫째, 오리지널 YOLO는 24개의 합성곱 층과 2개의 완전 연결 층으로 구성되어 있다. 반면, 이 모델은 22개의 합성곱 층과 2개의 완전 연결 층으로 구성되어 있으며, 일부 합성곱 층이 stride=2를 사용하여 출력 크기를 절반으로 줄인다.

둘째, 오리지널 YOLO는 주로 패딩이 없는 경우가 많지만, 이 모델은 대부분의 층에 padding을 사용하여 출력 크기를 조정한다.

셋째, 오리지널 YOLO는 다양한 크기의 필터를 사용하지만, 이 실험에서는 고정된 크기 (7x7 또는 3x3)의 필터만을 사용한다. 이는 오리지널 YOLO보다 연산 복잡도를 줄이는 데 기여할 수 있다.

마지막으로, 모든 활성화 함수로 Leaky ReLU(negative slope=0.1)를 사용하여 음수 입력값의 기울기를 조정하였다.

2.2 Data Set

데이터셋의 이미지를 불러오고, 다양한 데이터 증강 기법과 변환을 적용하여 모델이 다양한 조건에서 더 나은 일반화 성능을 발휘하도록 하였다. 최종적으로 모델의 입력 형식에 맞춘 이미지와 타겟 텐서를 생성하여 학습에 사용된다. 이미지의 크기를 YOLO 입력에 맞게 448x448로 설정하였고, 이미지 반전 (Horizontal Flip), 이미지 스케일링 (Scaling), 이미지 이동 (Translation), 이미지 자르기 (Random Cropping) 증강 기법이 적용되었다.

또한 모델 학습을 위해, 각 이미지를 YOLO 모델의 타겟 텐서로 변환한다. 타겟 텐서는 7x7 그리드로 나뉘며, 각 그리드 셀에 객체 존재 여부와 바운딩 박스 좌표 및 클래스 레이블이 포함된다.

2.3 Loss Function

YOLO 손실 함수는 크게 다음 네 가지 구성 요소로 이루어진다. 이 항목은 각 셀(cell)에서 예측된 클래스 확률이 실제 클래스 확률과 얼마나 차이가 있는지를 계산한다. 셀에 객체가 있는 경우 target의 4번째 채널 값이 양수이며, compute prob error 함수에서는 이 값이 0보다 큰 셀의 예측 클래스 확률과 실제 클래스 확률을 비교한다. 사용한 손실 함수는 MSE이다. 이미지의 특정 셀에 객체가 없는 경우에는 예측 확률을 0에 가깝게 예측해야 한다. not contain obj error 함수는 이러한 셀에서 예측한 확률이 0과 얼마나 차이나는지를 계산한다. contain obj error 함수는 예측된 바운딩 박스가 실제 바운딩 박스와의 IoU(Intersection over Union)가 가장 높은 예측 박스에 대해 오차를 계산하여 셀에 객체가 포함된 경우 해당 객체의 바운딩 박스 좌표 예측 오차를 측정한다. not contain box pred는 객체가 포함되지 않은 셀에 대한 두 번째 예측 바운딩 박스의 confidence 값을 0으로 맞추는 역할을 수행한다. 최종 손실 함수는 앞서 정의한 모든 오차를 가중합하여 최적화한다.

$$\text{total\_loss} = l_{\text{coord}} \cdot (\text{xy\_loss} + \text{wh\_loss}) +$$

$$\text{obj\_loss} + l_{\text{noobj}} \cdot (\text{noobj\_loss1} + \text{noobj\_loss2}) + \text{prob\_loss}(2)$$

2.4 Training

제공된 Pre-Trained 된 사전 가중치와 learning rate = 1e-4, momentum=0.9, weight decay=5e-4의 SGB Optimizer를 이용해 15에포크 훈련을 수행한다. 최종 결과는 Loss:0.5974, average loss: 0.6870 이다.

2.5 Architecture

```
Epoch [15/15], Iter [360,430] Loss:0.9356, average_loss: 0.6774
Epoch [15/15], Iter [370,430] Loss:0.5735, average_loss: 0.6808
Epoch [15/15], Iter [380,430] Loss:0.7204, average_loss: 0.6836
Epoch [15/15], Iter [390,430] Loss:0.7028, average_loss: 0.6844
Epoch [15/15], Iter [400,430] Loss:0.5275, average_loss: 0.6836
Epoch [15/15], Iter [410,430] Loss:0.7305, average_loss: 0.6828
Epoch [15/15], Iter [420,430] Loss:0.9444, average_loss: 0.6869
Epoch [15/15], Iter [430,430] Loss:0.5974, average_loss: 0.6870
```

Figure 5: YOLO Training (15Epoch Result).

2.6 Inference

클래스를 'aeroplane', 'bicycle', 'bus', 'car', 'cat', 'dog'의 여섯 가지로 한정하고 이미지 크기를 YOLO 모델 입력 크기인 448x448로 조정하고, RGB 형식으로 변환한 후 정규화 과정을 거쳤다. 이후 interpret target 함수를 사용하여 Bounding Box 좌표, 클래스 인덱스, 확률을 반환받았다. 이 결과를 원본 이미지 크기로 복원하여 결과 배열(result)에 저장하고 원본 이미지에 예측된 Bounding Box와 클래스 정보를 시각화하였다.

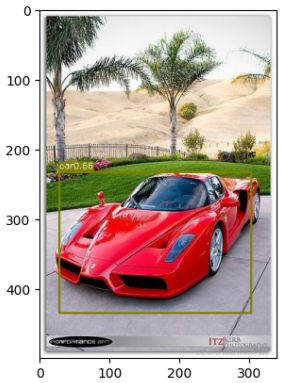


Figure 6: Car Image Object Detection Result.

2.7 Result

객체 탐지에서 모델 성능을 평가하기 위한 mAP(Mean Average Precision)를 각 클래스별로 계산하고, 평균 결과를 확인한다. mAP(mean Average Precision)은 객체 검출 성능을 평가하는 지표로, 여러 클래스에 걸쳐 평균 정밀도(AP)를 계산하여 정확한 위치와 클래스 예측 여부를 평가한다. AP는 각 클래스의 Precision-Recall 곡선 아래 면적을 구한 값이며, mAP는 이를 모든 클래스에 대해 평균낸 값이다.

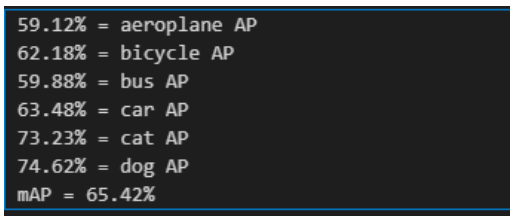


Figure 7: YOLO Object Detection mAP.

[1] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[2] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[3] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.