**\* YOLO**

: You Only Look Once의 약자로 한 번에 보고 바로 처리한다는 뜻을 가지며, 처음으로 one-stage-detection 방법을 고안해 실시간으로 object detection이 가능하게 만들었음

- 특징

* 이미지 전체를 한 번만 보는 것

-> YOLO 이전의 RCNN은 이미지를 여러 장으로 분할하고, CNN 모델을 이용해 이미지를 분석했기 때문에 이미지 한 장에서 object detection을 해도 실제로는 여러 장의 이미지를 분석하는 것과 같았음

* 통합된 모델을 사용하는 것(간단함)

-> 기존 object detection 모델은 다양한 전처리 모델과 인공신경망을 결합해서 사용했음

* 실시간으로 객체를 탐지할 수 있는 것

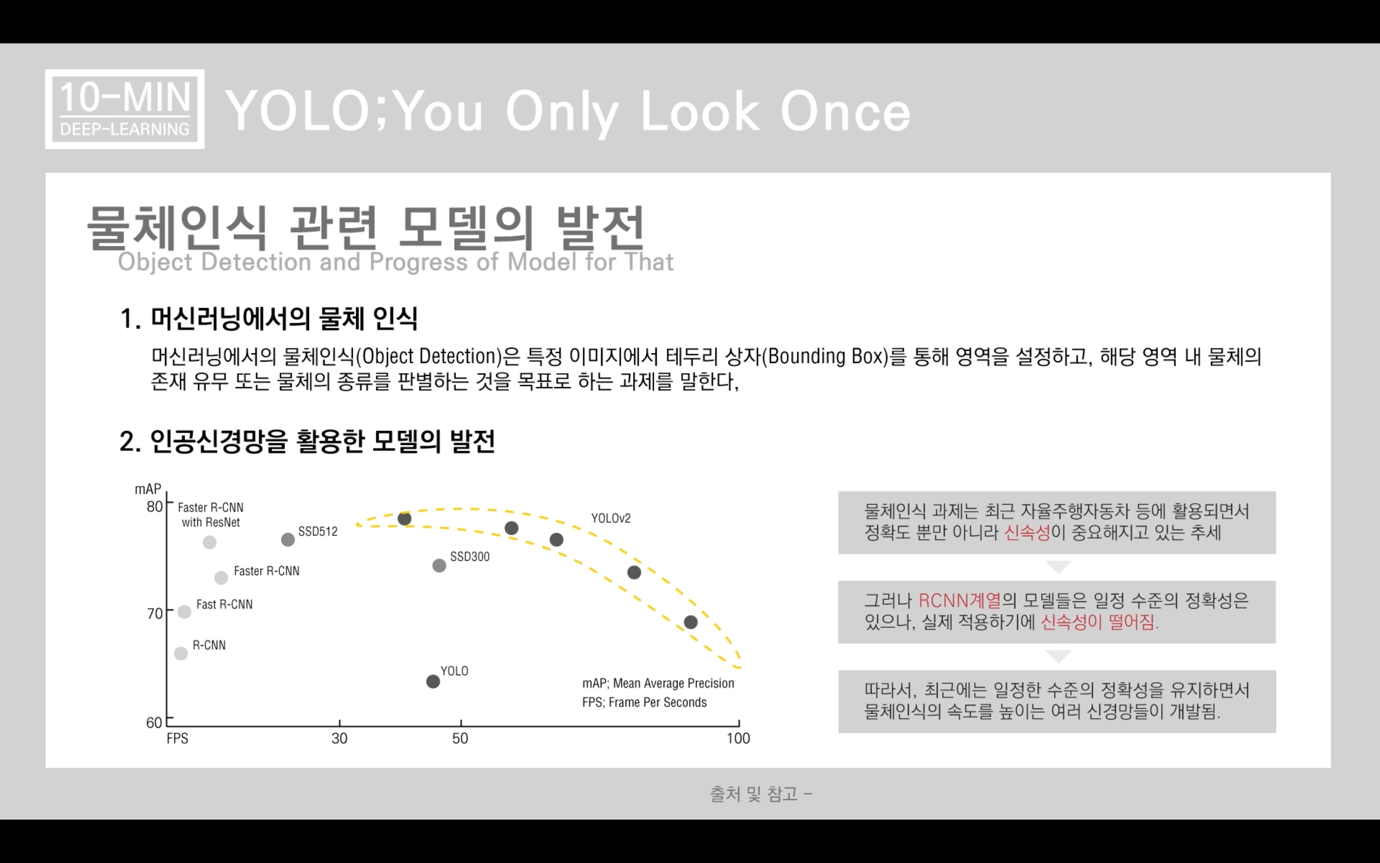
-> 높은 성능은 아니더라도 준수한 성능으로 실시간으로 object detection이 가능했기 때문

- 강점

* 간단한 처리 과정으로 속도가 매우 빠르며 기존의 실시간 object detection 모델들과 비교하면 2배 정도 높은 mAP를 보임
* 이미지 전체를 한 번에 바라보는 방식을 이용하므로 class에 대한 맥락적 이해도가 다른 모델에 비해 높아 낮은 false-positive를 보임
* 일반화된 object 학습이 가능하여 자연 이미지로 학습하고 이를 그림과 같은 곳에 테스트 해도 다른 모델에 비해 훨씬 높은 성능을 보여줌

- 단점

* 특히 다른 모델에 비해 작은 개체에 대해 정확도가 낮음



**\* 객체 검출(Object Detection) 관련 모델의 발전**

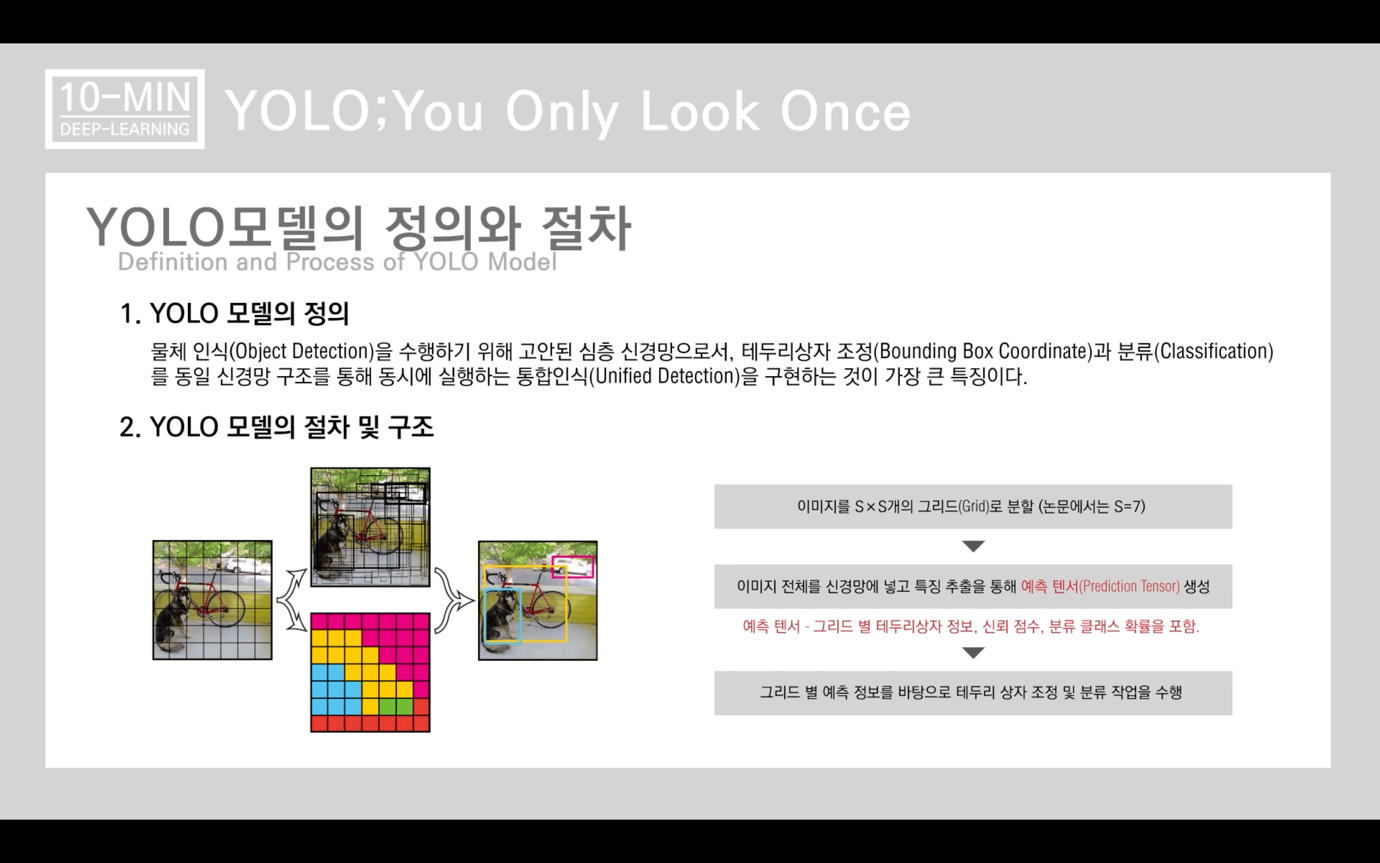
1. 머신러닝에서의 객체 검출(Object Detection)

: 특정 이미지에서 Bounding Box를 통해 영역 설정을 한 후, Bounding Box의 해당 영역 내 물체의 존재 유무 또는 물체의 종류를 판별하는 것을 말한다.

2. 인공신경망을 활용한 모델의 발전

: 객체 검출은 최근 자율주행자동차 등에 활용되면서 정확도 뿐만 아니라 신속성(속도)도 중요해지고 있는 추세이다. 그러나, RCNN 계열의 모델들은 일정 수준의 정확성은 있으나, 실제 적용하기엔 신속성이 떨어진다. (그래프를 보면 RCNN 계열의 모델들은 30 FPS도 안되는 것을 볼 수 있음) 따라서, 최근에는 일정한 수준의 정확성을 유지하면서 객체 검출의 속도를 높이는 여러 신경망들이 개발되었다. (ex. SSD300, YOLO 등)

* mAP(Mean Average Precision) : 정확성
* FPS(Frame Per Seconds) : 신속성(속도)



**\* YOLO 모델의 정의와 절차**

1. YOLO 모델의 정의

: 객체 검출(Object Detection)을 수행하기 위해 고안된 심층 신경망으로서, 테두리 상자 조정 (Bounding Box Coordinate)와 분류(Classification)을 동일 신경망 구조를 통해 동시에 실행하는 통합인식 (Unified Detection)을 구현하는 것이다.

2. YOLO 모델의 절차 및 구조

- YOLO Process

1. S x S grid on input

: 이미지를 S x S개의 그리드(Grid)로 분할한다.(논문에서는 S=7)

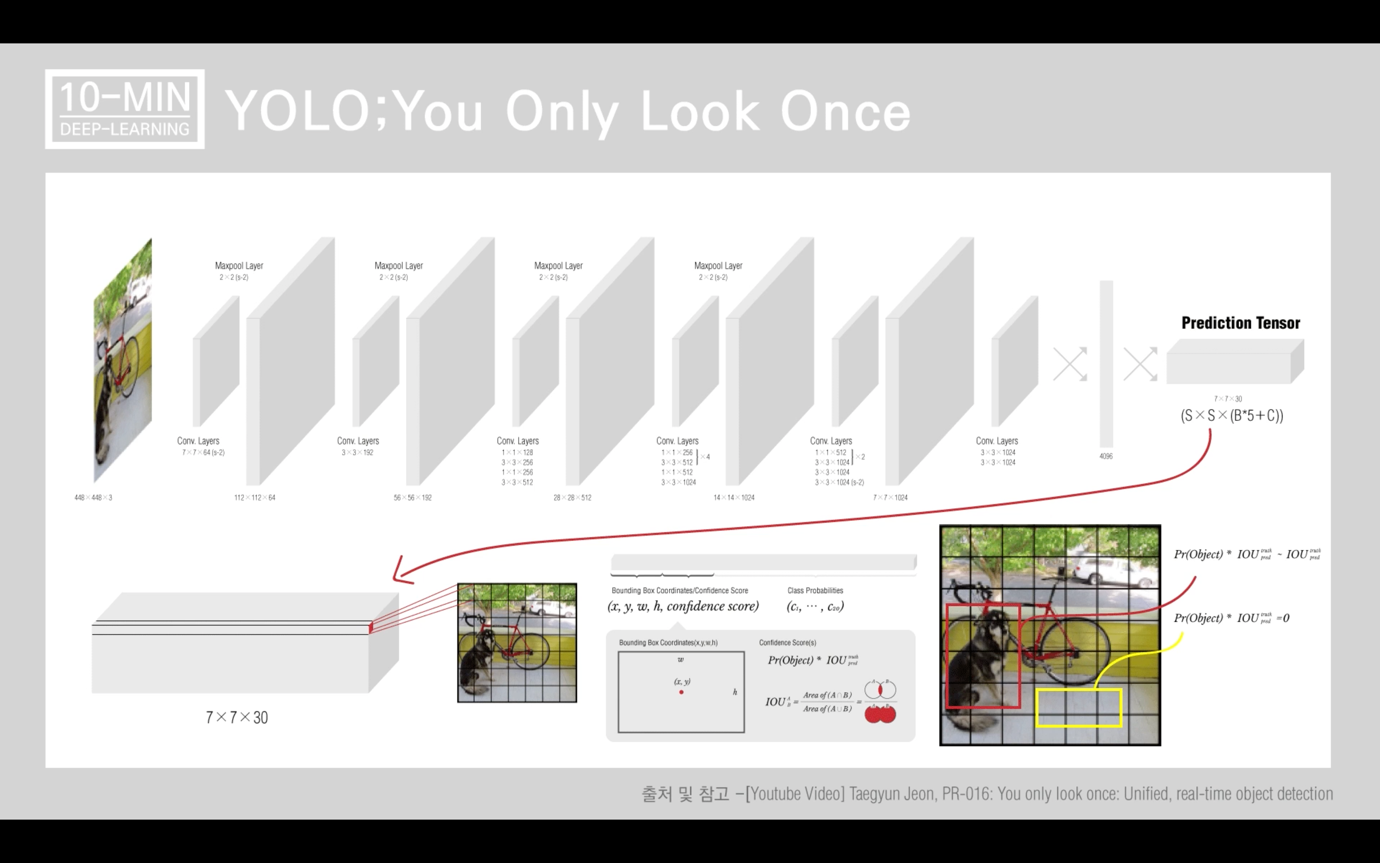
2. Class probability map -> bounding boxes + confidence

: 이미지 전체를 신경망에 넣고 특징 추출을 통해 예측 텐서(Prediction Tensor)를 생성한다.

* 예측 텐서 : grid별 Bounding Box 정보, 신뢰 점수(Confidence Score), 분류 클래스 확률을 포함

3. Final detections

: grid별 예측 정보를 바탕으로 Bounding Box 조정 및 분류 작업을 수행한다.



- YOLO Network

: Input Image -> CNN -> FC -> Prediction Tensor -> Input Image

* CNN : Convolution filter를 거쳐 feature map을 뽑아내고, 마지막 Convolution 단계에서 flatten을 통해 FC와 매핑함
* FC(Fully-Connected) : 전연결 노드
* Prediction Tensor : image regression 및 분류 작업 조정

- Prediction Tensor

* (S x S x (B \* 5 + C))

-> S : grid 개수

-> B : grid별로 갖는 bounding box의 개수

ex) bounding box 중앙점(x, y)로 2개이므로 B = 2

-> C : class 개수(논문에서는 ImageNet이 쓰였으므로 20개) – grid cell에 포함된 물체가 각 class에 해당될 확률들의 벡터 크기

* (7, 7, 30)

-> 7, 7 : grid

-> 30 : 길이

b.b1, b.b2, 20

(Bounding box coordinates/ (Class probabilities)

Confidence score)

* Bounding box coordinates(x, y, w, h)

-> (x, y) : bounding box의 중심점을 의미하며, grid cell의 범위에 대한 상대값이 입력됨

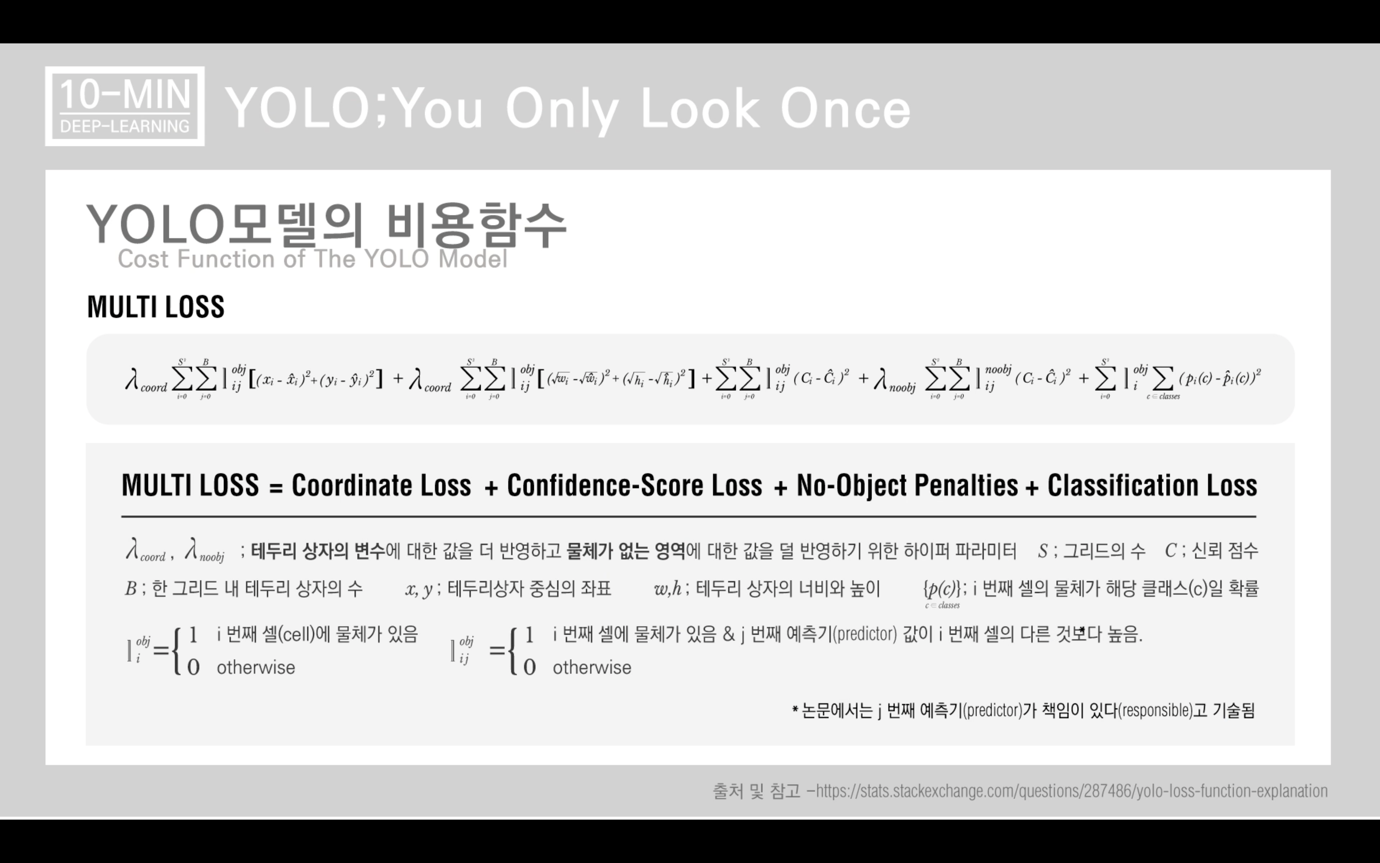
-> w, h : 전체 이미지의 넓이, 높이에 대한 상대값이 입력됨

* Confidence score(s) : 이 시스템이 물체를 포함한다는 예측을 얼마나 확신하는지, 박스에 대한 예측이 얼마나 정확할지를 의미한다. (예측 box와 실제 정답 사이의 IOU를 의미함)

-> Pr(Object) \* IOU(Truth/Pred) : 객체가 있을 확률 \* (교집합/합집합)

- Result

* Pr(Object) \* IOU ~ IOU : Object가 올바르게 학습이 되었다면 1이 되게끔 해서 1 \* IOU = IOU로 IOU 높은 것들의 bounding box만 선정함
* Pr(Object) \* IOU = 0 : Cell에 object가 존재하지 않는다면 confidence score는 0이 됨



**\* YOLO 모델의 비용함수**

: Prediction Tensor에서 값을 조정해 학습할 때, cost function 이용

- Multi Loss

: [x, y + w, h + s + s + c]

: Coordinate Loss + Confidence-Score Loss + No-Object Penalties + Classification Loss

* Coordinate Loss : x, y, w, h
* No-Object Penalties : object가 없을 때, 패널티 부여
* 함수coord : bounding box의 변수에 대한 값을 더 반영하기 위한 하이퍼 파라미터

함수noobj : 물체가 없는 영역에 대한 값을 덜 반영하기 위한 하이퍼 파라미터

* obj-i : 1(i번째 셀에 물체가 있음, 없음), 0(otherwise)
* obj-ij : 1(i번째 셀에 물체가 있음, 없음 & j번째 predictor 값이 i번째 셀의 다른 것보다 높음, 낮음), 0(otherwise)

- Result

뽑아낸 bounding box에서 물체가 있고 다른 셀보다 confidence score 값이 높으면 대표적인 bounding box로 선정한다.

-> why?

Grid별로 2개를 뽑아내기 때문에 7 x 7 x 2 면 98개의 bounding box가 나오는데 실제 객체는 그것보다 적어서 대표적인 bounding box를 선정해야 하기 때문이며, 이를 위해서 함수 파라미터가 붙는다.

**\* YOLO 실습**

: YOLOv5, Pytorch 및 Python을 사용한 깊은 졸음 감지

(Deep Drowsiness Detection using YOLOv5, Pytorch and Python)

- Object : 우리가 졸릴 때와 완전히 깨어 있을 때 특정 모델을 감지하는 데 사용할 것이며, 웹 캠을 활용하여 우리가 졸린 지의 여부를 실시간으로 테스트 할 것임

- Process

1. Ultralytics YOLOv5 설치

2. 이미지에서 물체 감지

3. 사전 녹화된 비디오에서 물체 감지

4. OpenCV를 사용하여 실시간으로 객체 감지

5. YOLOv5 및 Pytorch를 사용하여 졸음 모델 미세 조정

6. 실시간 졸음 감지 수행

- Code

coco dataset에 대해 훈련된 baseline model을 사용하여 몇 가지 탐지를 수행함

(coco dataset : 기본적으로 약 80개의 다른 클래스가 있음)

이 구현은 미세 조정 없이도 일부 감지를 볼 수 있도록 시도할 수 있지만, 우리는 표준 이미지 수집 코드를 사용해 일부 이미지를 수집하여 우리가 졸리거나 그렇지 않은 것을 분류할 것

- Result

