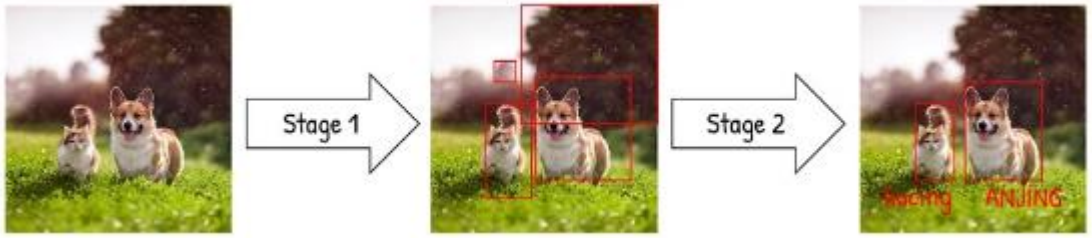
객체탐지 YOLO 분석

최석재 *lingua@naver.com*

Two-stage detector

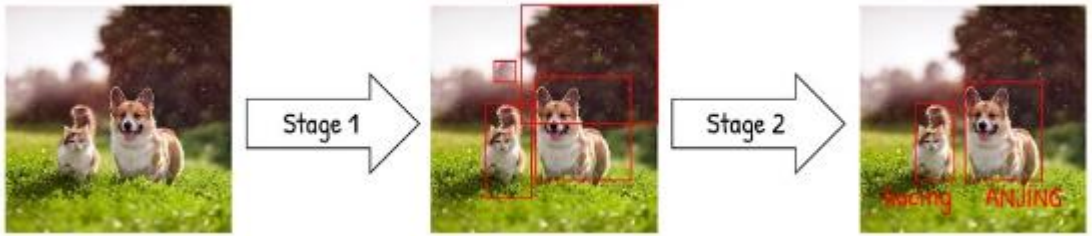


- 객체 탐지(Object Detection)란 이미지에서 특정 객체를 탐지하고 영역을 인식하는 기술
- 객체 탐지 기술은 two-stage detector와 one-stage detector로 나뉜다
- 먼저 two-stage detector는 다음과 같은 프로세스를 거친다

1. 영역 제안 (Region Proposal)
- 이미지 내에서 객체가 존재할 가능성이 있는 후보 영역을 추출한다
- 후보 영역은 복수가 될 수 있다

2. 분류(Classification)와 바운딩 박스 회귀(Bounding Box Regression)
- 추출된 후보 영역 각각을 딥러닝 모델에 입력으로 제공하여, 영역 내 객체를 분류한다
- 정확한 객체의 위치를 나타내기 위해 바운딩 박스의 사이즈와 위치를 조절한다

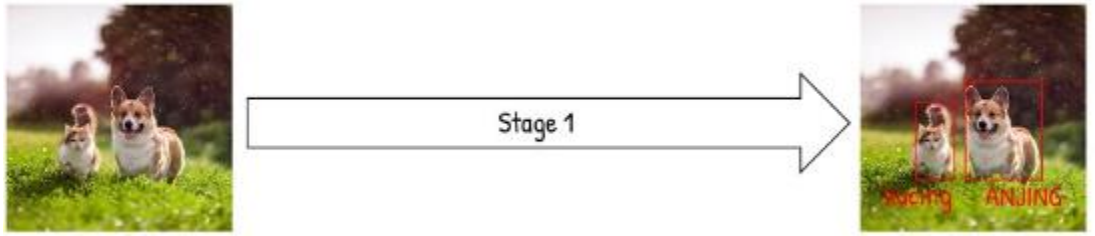
Two-stage detector

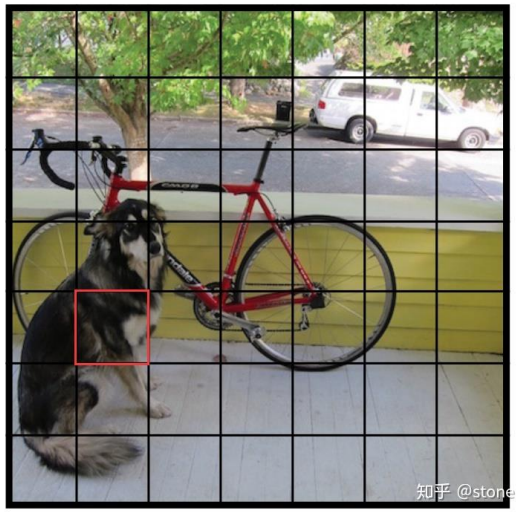


- 이 방식의 장점은 객체의 영역을 확인하고 분류하므로 높은 정확도를 달성할 수 있다는 것

- 그러나 Two-stage detector는 일단 영역 제안을 받고,
- 해당 부분에 대한 특징 정보만 보고 객체를 추출하는 방식이므로
- 객체 주변의 정보들을 무시하여 전체 맥락에 대한 이해도가 낮다는 점이 한계이다
- 그 결과 사람이 나무나 기둥에 가려져 있을 때 나무는 인식하고, 가려져 있는 사람은 잘 인식하지 못하는 것과 같은 문제가 발생한다

One-stage detector

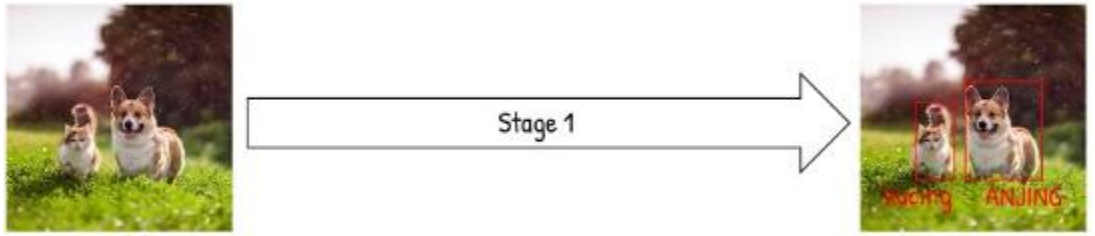




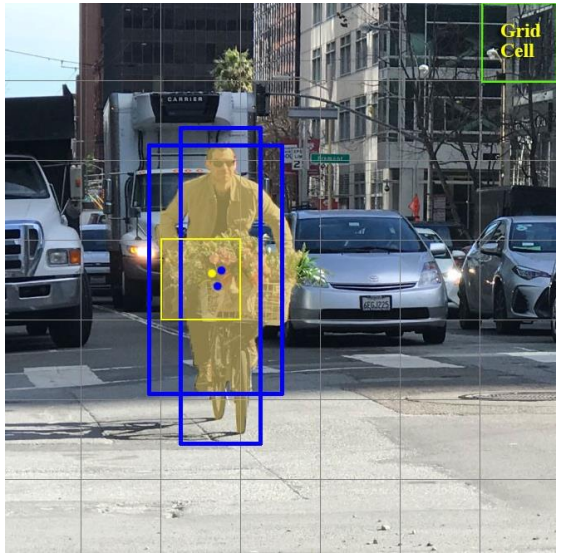
- One-stage detector는 객체의 정보와 위치를 한 번에 예측하는 방식이다
- YOLO가 이 방식으로 처음 구현되었다 (You Only Look Once)
- One-stage detector는 다음과 같이 그리드 분할 방법으로 예측한다

- 그리드 분할 (Grid Cell Division)
- 먼저 이미지를 7x7과 같은 사이즈의 그리드로 분할한다
- 객체의 중심이 특정 그리드 셀 내에 위치하면 그 셀이 해당 객체를 탐지하는 역할을 한다
- 해당 그리드 셀은 서로 다른 형태의 복수 개(보통 2~5개) 바운딩 박스를 상정하여 각각의 바운딩 박스에 대해 객체가 존재할 위치, 크기, 확률을 구한다

One-stage detector



- 복수 개의 바운딩 박스를 사용하는 이유는
- 높이와 너비의 비율이 다른 바운딩 박스로 객체를 더 잘 탐지하기 위해서이다
- 이를 위하여 미리 정의된 다양한 크기와 비율의 바운딩 박스를 사용한다
- 이를 앵커 박스(anchor boxes)라고 한다
- 또한 객체가 밀집해 있는 경우 각 그리드 셀이 여러 개의 객체를 탐지할 수 있다



- 장점은 이미지 전체 맥락에 대한 이해가 높아 객체가 가려져 있어도 비교적 잘 탐지하며,
- 속도가 빨라서 실시간 탐지에 적합하다

- 그러나 작은 객체를 포착할 수 있는 학습된 앵커 박스가 많지 않은 경우,
- 이미지 내에서 차지하는 비율이 작은 객체는 잘 포착하지 못하는 단점이 있다

Local 환경 설정

- Local PC에 저장되어 있는 동영상을 대상으로 사용해본다
- pytest_img > YOLO 폴더에 있는 sample.mp4 파일을 다운로드

- from ultralytics import YOLO
- model = YOLO("../models/yolov8m.pt")
  예: C:/Users/lingu/Downloads/models/yolov8m.pt

모델의 종류는
yolov8n, yolov8s, yolov8m, yolov8l, yolov8x의 다섯 가지 모델이 있다

- seconds = 1 # 동영상 읽을 시간(초 단위)

비디오 경로 설정 및 결과 저장 경로 설정
- video_path = 'C:/Users/lingu/Downloads/sample.mp4' # video path
- output_dir = 'C:/Users/lingu/Downloads/results' # 결과를 저장할 폴더 (자동 생성)

예측 함수 정의

- YOLO 모델을 이용하여 예측하기 위한 함수를 정의한다

- import torch

- def predict(frame, iou=0.7, conf=0.25):

```
    results = model(
        source = frame,
        device = "0" if torch.cuda.is_available() else "cpu",
        iou = 0.7,                  # 바운딩 박스 필터링 신뢰도 기준
        conf = 0.25,                # 모델이 탐지한 객체에 대한 최소 신뢰도 기준
        verbose = False,            # 추가 정보 출력 여부
    )
    result = results[0]             # 첫 번째 프레임의 이미지 (현재 1개씩의 프레임만 전달됨)
    return result
```

iou: Intersection Over Union
예측된 경계 상자와 실제 경계 상자 사이의 겹치는 부분을 의미
여러 바운딩 박스 후보 중 신뢰도가 0.7 이상인 것만 남긴다

바운딩 박스 함수 정의

- import cv2
- import numpy as np

- def draw_boxes(result, frame):

```
    for boxes in result.boxes:                                              # 바운딩 박스 분석 결과 순회
        x1, y1, x2, y2, score, classes = boxes.data.squeeze().cpu().numpy()
        label = model.names[int(classes)]                                   # 정수형 클래스 이름을 문자형으로 받음
        cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 0, 255), 3)      # 바운딩 박스 생성
        cv2.putText(frame, f'{label} {score:.2f}', (int(x1), int(y1) - 10), cv2.FONT_HERSHEY_SIMPLEX, 1.2, (255, 0, 0), 2)
```

클래스 이름과 신뢰도를 바운딩 박스 영역에 맞춰 입력　　OpenCV 기본 폰트 사용, 텍스트 크기 비율, 텍스트 색상 RGB, 텍스트 두께

```
    return frame, x1, y1, x2, y2, score, classes                # 처리된 이미지와 좌표, 신뢰도, 클래스 정보를 return
```

squeeze()는 크기가 1인 모든 차원을 제거한다. 배치 크기가 1이므로 배치 차원이 없어지게 된다
이렇게 하는 이유는 데이터를 더 간단하고 일반적인 형태로 만들기 위함
예를 들어 [1, n, 4] (배치크기, 탐지된 객체수, 각 객체의 바운딩 박스 좌표)일 때 → [n, 4]로 바뀐다

출력 프레임 계산

- import cv2

- capture = cv2.VideoCapture(video_path)

비디오가 마지막 프레임에 도달한 경우, 처음부터 다시 시작하기
- if capture.get(cv2.CAP_PROP_POS_FRAMES) == capture.get(cv2.CAP_PROP_FRAME_COUNT):
    capture.set(cv2.CAP_PROP_POS_FRAMES, 0)

- fps = capture.get(cv2.CAP_PROP_FPS) # 초당 프레임 수 얻기
- total_frames = int(fps * seconds) # seconds 시간 동안의 프레임 수
- print('total 프레임:', total_frames) # 출력하게 되는 전체 프레임 수. 25

실행 (1/2)

- import os

- if not os.path.exists(output_dir):
    os.makedirs(output_dir)

- frame_count = 0
- while frame_count < total_frames:
    ok, frame = capture.read()
    if not ok:
        print("프레임 읽기에 실패했습니다. 종료.")
        break
    result = predict(frame)
    results = draw_boxes(result, frame)
    cv2.imshow("VideoFrame", results[0])

현재 프레임의 이미지를 예측

각 프레임의 예측 결과에 대하여 바운딩 박스 생성

frame을 담고 있는 0번만 사용. 이미지, 좌표, 신뢰도, 클래스 정보

실행 (2/2)

```
import os
from google.colab.patches import cv2_imshow

if not os.path.exists(output_dir):
  os.makedirs(output_dir)

frame_count = 0
while frame_count < total_frames:
  ok, frame = capture.read()
  if not ok:
    print("프레임 읽기에 실패했습니다. 종료.")
    break
  result = predict(frame)
  results = draw_boxes(result, frame)
  cv2_imshow(results[0])

  output_path = os.path.join(output_dir, f'frame_{frame_count:04d}.jpg')
  cv2.imwrite(output_path, results[0])
  print("Saved to:", output_path)

  frame_count += 1

  key = cv2.waitKey(10)
  if key==ord('q'):
    print("사용자가 종료를 요청했습니다.")
    break

capture.release()
cv2.destroyAllWindows()
```

while 문 안에 있음

```
output_path = os.path.join(output_dir, f'frame_{frame_count:04d}.jpg')
cv2.imwrite(output_path, results[0])          # 파일로 저장
print("Saved to:",  output_path)

frame_count += 1

key = cv2.waitKey(10)                         # 10 밀리초 동안 키 입력을 대기하고 입력된 키의 값을 반환
if key == ord('q'):                           # 입력된 key 값이 'q' 라면
    print("사용자가 종료를 요청했습니다.")
    break
```

- capture.release() # 메모리에서 자원 해제
- cv2.destroyAllWindows() # 모든 창 닫기

```
Saved to: /content/gdrive/MyDrive/pytest_img/YOLO/results/frame_0000.jpg
Saved to: /content/gdrive/MyDrive/pytest_img/YOLO/results/frame_0001.jpg
Saved to: /content/gdrive/MyDrive/pytest_img/YOLO/results/frame_0002.jpg
Saved to: /content/gdrive/MyDrive/pytest_img/YOLO/results/frame_0003.jpg
Saved to: /content/gdrive/MyDrive/pytest_img/YOLO/results/frame_0004.jpg
Saved to: /content/gdrive/MyDrive/pytest_img/YOLO/results/frame_0005.jpg
Saved to: /content/gdrive/MyDrive/pytest_img/YOLO/results/frame_0006.jpg
Saved to: /content/gdrive/MyDrive/pytest_img/YOLO/results/frame_0007.jpg
```

분석결과 확인

- 앞에서 지정한 결과 저장 경로를 확인한다
- 사람은 잘 인식했으나, 그물은 제대로 인식하지 못했다

Colab에서의 사용

Colab에서 YOLO v8 사용하기

- Colab에는 기본 환경이 구축되어 있으므로 몇 가지 명령어만으로 사용이 가능하다
- 설정 부분 정도를 제외하고는 동일하다

구글 드라이브와 연결하기

- from google.colab import drive
- drive.mount('/content/gdrive')

토치비전 설치

- !python -m pip install torch torchvision

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: torch in /usr/local/lib/python3.9/dist-packages (2.0.0+cu118)
Requirement already satisfied: torchvision in /usr/local/lib/python3.9/dist-packages (0.15.1+cu118)
Requirement already satisfied: sympy in /usr/local/lib/python3.9/dist-packages (from torch) (1.11.1)
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.9/dist-packages (from torch) (2.0.0)
Requirement already satisfied: networkx in /usr/local/lib/python3.9/dist-packages (from torch) (3.1)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.9/dist-packages (from torch) (4.5.0)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.9/dist-packages (from torch) (3.1.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.9/dist-packages (from torch) (3.11.0)
Requirement already satisfied: lit in /usr/local/lib/python3.9/dist-packages (from triton==2.0.0->torch) (16.0.1)
Requirement already satisfied: cmake in /usr/local/lib/python3.9/dist-packages (from triton==2.0.0->torch) (3.25.2)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages (from torchvision) (2.27.1)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.9/dist-packages (from torchvision) (8.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages (from torchvision) (1.22.4)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-packages (from jinja2->torch) (2.1.2)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests->torchvision) (2.0.12)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests->torchvision) (2022.12.7)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist-packages (from requests->torchvision) (1.26.15)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests->torchvision) (3.4)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.9/dist-packages (from sympy->torch) (1.3.0)
```

ultralytics 설치

- !pip install ultralytics

```
Requirement already satisfied: py-cpuinfo in /usr/local/lib/python3.10/dist-packages (from ultralytics) (9.0.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (4.41.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (1.4.4)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (1.22.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (23.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (3.1.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1.4->ultralytics) (2022.7.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (2023.7.22)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (3.4)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch>=1.7.0->ultralytics) (3.12.2)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from torch>=1.7.0->ultralytics) (4.7.1)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch>=1.7.0->ultralytics) (1.11.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch>=1.7.0->ultralytics) (3.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch>=1.7.0->ultralytics) (3.1.2)
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dist-packages (from torch>=1.7.0->ultralytics) (2.0.0)
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch>=1.7.0->ultralytics) (3.25.2)
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch>=1.7.0->ultralytics) (16.0.6)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=3.2.2->ultralytics) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.7.0->ultralytics) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch>=1.7.0->ultralytics) (1.3.0)
Installing collected packages: ultralytics
Successfully installed ultralytics-8.0.145
```

모델 저장 경로 설정

- %cd /content/gdrive/MyDrive/pytest_img/YOLO/

```
/content/gdrive/MyDrive/pytest_img/YOLO
```

경로 확인

- !ls # sample.mp4 파일이 있으면 정상

모델 다운로드

- from ultralytics import YOLO

- model = YOLO("../models/yolov8m.pt")

```
Downloading https://github.com/ultralytics/assets/releases/download/v8.2.0/yolov8m.pt to '../models/yolov8m.pt'...
100%|██████████| 49.7M/49.7M [00:00<00:00, 109MB/s]
```

모델의 종류는
yolov8n, yolov8s, yolov8m, yolov8l, yolov8x의 다섯 가지 모델이 있다

경로 설정

- seconds = 1 # 동영상을 읽을 시간을 초 단위로 설정

- video_path = '/content/gdrive/MyDrive/pytest_img/YOLO/sample.mp4' # video path
- output_dir = '/content/gdrive/MyDrive/pytest_img/YOLO/results' # 결과 저장 폴더

- 이후 내용은 Local에서와 동일하다

예측 함수 정의

- YOLO 모델을 이용하여 예측하기 위한 함수를 정의한다

- import torch
- def predict(frame, iou=0.7, conf=0.25):

```
    results = model(
        source = frame,
        device = "0" if torch.cuda.is_available() else "cpu",
        iou = 0.7,                  # 바운딩 박스 필터링 신뢰도 기준
        conf = 0.25,                # 모델이 탐지한 객체에 대한 최소 신뢰도 기준
        verbose = False,            # 추가 정보 출력 여부
    )
    result = results[0]             # 첫 번째 프레임의 이미지 (현재 1개씩의 프레임만 전달됨)
    return result
```

iou: Intersection Over Union
예측된 경계 상자와 실제 경계 상자 사이의 겹치는 부분을 의미
여러 바운딩 박스 후보 중 신뢰도가 0.7 이상인 것만 남긴다

바운딩 박스 함수 정의

- import cv2
- import numpy as np

- def draw_boxes(result, frame):

```
    for boxes in result.boxes:                                              # 바운딩 박스 분석 결과 순회
        x1, y1, x2, y2, score, classes = boxes.data.squeeze().cpu().numpy()
        label = model.names[int(classes)]                                   # 정수형 클래스 이름을 문자형으로 받음
        cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 0, 255), 3)      # 바운딩 박스 생성
        cv2.putText(frame, f'{label} {score:.2f}', (int(x1), int(y1) - 10), cv2.FONT_HERSHEY_SIMPLEX, 1.2, (255, 0, 0), 2)
```

클래스 이름과 신뢰도를 바운딩 박스 영역에 맞춰 입력　　OpenCV 기본 폰트 사용, 텍스트 크기 비율, 텍스트 색상 RGB, 텍스트 두께

```
    return frame, x1, y1, x2, y2, score, classes                # 처리된 이미지와 좌표, 신뢰도, 클래스 정보를 return
```

squeeze()는 크기가 1인 모든 차원을 제거한다. 배치 크기가 1이므로 배치 차원이 없어지게 된다
이렇게 하는 이유는 데이터를 더 간단하고 일반적인 형태로 만들기 위함이다
예를 들어 [1, n, 4] (배치크기, 탐지된 객체수, 각 객체의 바운딩 박스 좌표)일 때 → [n, 4]로 바뀐다

출력 프레임 계산

- import cv2

- capture = cv2.VideoCapture(video_path)
- if capture.get(cv2.CAP_PROP_POS_FRAMES) == capture.get(cv2.CAP_PROP_FRAME_COUNT):
  capture.set(cv2.CAP_PROP_POS_FRAMES, 0)

현재 비디오의 위치가 비디오의 전체 프레임 수와 같은지,
즉, 비디오의 마지막 프레임에 도달했는지 검사

마지막 프레임에 도달했다면,
프레임의 위치를 비디오의 시작점으로 재설정하여 다시 재생할 준비를 함

- fps = capture.get(cv2.CAP_PROP_FPS) # 초당 프레임 수 얻기
- total_frames = int(fps * seconds) # seconds 시간 동안의 프레임 수
- print('total 프레임:', total_frames) # 출력하게 되는 전체 프레임 수. 25

실행 (1/2)

- import os
- from google.colab.patches import cv2_imshow

- if not os.path.exists(output_dir):
    os.makedirs(output_dir)

- frame_count = 0
- while frame_count < total_frames:
    ok, frame = capture.read()
    if not ok:
        print("프레임 읽기에 실패했습니다. 종료.")
        break
    result = predict(frame)
    results = draw_boxes(result, frame)
    cv2_imshow(results[0]) # frame을 담고 있는 0번만 사용

실행 (2/2)

```
import os
from google.colab.patches import cv2_imshow

if not os.path.exists(output_dir):
  os.makedirs(output_dir)

frame_count = 0
while frame_count < total_frames:
  ok, frame = capture.read()
  if not ok:
    print("프레임 읽기에 실패했습니다. 종료.")
    break
  result = predict(frame)
  results = draw_boxes(result, frame)
  cv2_imshow(results[0])

  output_path = os.path.join(output_dir, f'frame_{frame_count:04d}.jpg')
  cv2.imwrite(output_path, results[0])
  print("Saved to:", output_path)

  frame_count += 1

  key = cv2.waitKey(10)
  if key==ord('q'):
    print("사용자가 종료를 요청했습니다.")
    break

capture.release()
cv2.destroyAllWindows()
```

while 문 안에 있음

```
output_path = os.path.join(output_dir, f'frame_{frame_count:04d}.jpg')
cv2.imwrite(output_path, results[0])          # 파일로 저장
print("Saved to:",  output_path)

frame_count += 1

key = cv2.waitKey(10)                         # 10 밀리초 동안 키 입력을 대기하고 입력된 키의 값을 반환
if key == ord('q'):                           # 입력된 key 값이 'q' 라면
    print("사용자가 종료를 요청했습니다.")
    break
```

- capture.release() # 메모리에서 자원 해제
- cv2.destroyAllWindows() # 모든 창 닫기

```
Saved to: /content/gdrive/MyDrive/pytest_img/YOLO/results/frame_0000.jpg
Saved to: /content/gdrive/MyDrive/pytest_img/YOLO/results/frame_0001.jpg
Saved to: /content/gdrive/MyDrive/pytest_img/YOLO/results/frame_0002.jpg
Saved to: /content/gdrive/MyDrive/pytest_img/YOLO/results/frame_0003.jpg
Saved to: /content/gdrive/MyDrive/pytest_img/YOLO/results/frame_0004.jpg
Saved to: /content/gdrive/MyDrive/pytest_img/YOLO/results/frame_0005.jpg
Saved to: /content/gdrive/MyDrive/pytest_img/YOLO/results/frame_0006.jpg
Saved to: /content/gdrive/MyDrive/pytest_img/YOLO/results/frame_0007.jpg
```

분석 결과 확인

- 앞에서 지정한 결과 저장 경로를 확인한다
- 사람은 잘 인식했으나, 그물은 제대로 인식하지 못했다

객체 위치 정보 Pandas로 정리 (1/3)

- import cv2
- import pandas as pd

프레임 시작 지점 초기화

- capture = cv2.VideoCapture(video_path)
- if capture.get(cv2.CAP_PROP_POS_FRAMES) == capture.get(cv2.CAP_PROP_FRAME_COUNT):

    capture.set(cv2.CAP_PROP_POS_FRAMES, 0)

- frame_count = 0
- data = []

객체 위치 정보 Pandas로 정리 (2/3)

- while frame_count < total_frames:

```
    ok, frame = capture.read()
    if not ok:
        print("프레임 읽기에 실패했습니다. 종료")
        break
    result = predict(frame)
    results = draw_boxes(result, frame)

    xmin = results[1]                    # x1
    ymin = results[2]                    # y2
    xmax = results[3]                    # x2
    ymax = results[4]                    # y2
    confidence = results[5]              # score
    class_id = results[6]                # classes

    data.append([xmin, ymin, xmax, ymax, confidence, class_id])      # data 리스트에 결과를 추가
    frame_count += 1
```

객체 위치 정보 Pandas로 정리 (3/3)

- capture.release()
- cv2.destroyAllWindows()

- df = pd.DataFrame(data, columns=['xmin', 'ymin', 'xmax', 'ymax', 'confidence', 'class'])
- print(df)

	xmin	ymin	xmax	ymax	confidence	class
0	358.814789	473.679657	1661.984009	826.175232	0.861612	8.0
1	359.286438	477.869659	1670.956787	832.348022	0.870794	8.0
2	360.762115	482.152679	1654.847168	819.409485	0.863247	8.0
3	365.622437	484.062744	1672.197144	817.899902	0.859944	8.0
4	361.093811	488.134918	1598.921265	819.122925	0.891443	8.0
5	885.452271	218.280670	1185.341797	495.739655	0.458125	24.0
6	885.941711	217.941299	1189.287720	500.989014	0.540155	24.0
7	891.040283	218.308014	1192.860352	503.791992	0.506911	24.0
8	896.101501	218.789612	1198.618774	506.189392	0.452767	24.0
9	898.903015	218.935104	1206.374390	508.780792	0.410287	24.0
10	905.549561	220.926544	1213.283569	509.202118	0.300703	24.0
11	369.714844	518.252808	1615.651611	812.976929	0.801592	8.0
12	370.712952	528.532898	1594.235596	809.511536	0.856819	8.0
13	370.914734	538.990051	1665.358887	811.067200	0.781377	8.0
14	916.808167	221.476410	1230.544678	531.069092	0.420519	24.0
15	919.719727	220.617569	1235.096924	528.987854	0.394348	24.0
16	365.870178	607.571045	1582.191650	795.432312	0.682366	8.0
17	931.334534	221.268265	1249.663696	538.715881	0.302369	24.0
18	930.758484	223.951324	1252.840210	535.794067	0.469738	24.0
19	788.850037	49.741516	1398.598022	732.219421	0.915446	0.0
20	790.003235	48.702049	1401.224609	748.203735	0.876010	0.0
21	368.623901	688.287781	1722.298584	795.281982	0.367186	37.0
22	372.782043	690.149841	1720.324951	797.510620	0.425239	37.0
23	793.929749	37.537560	1406.919556	746.963562	0.894975	0.0
24	383.311554	655.857605	1710.128174	781.728516	0.294600	37.0

전체 코드

```python
import cv2
import pandas as pd

capture = cv2.VideoCapture(video_path)

if capture.get(cv2.CAP_PROP_POS_FRAMES) == capture.get(cv2.CAP_PROP_FRAME_COUNT):
    capture.set(cv2.CAP_PROP_POS_FRAMES, 0)

frame_count = 0
data = []

while frame_count < total_frames:
    ok, frame = capture.read()
    if not ok:
        print("프레임 읽기에 실패했습니다. 종료")
        break
    result = predict(frame)
    results = draw_boxes(result, frame)

    xmin = results[1]
    ymin = results[2]
    xmax = results[3]
    ymax = results[4]
    confidence = results[5]
    class_id = results[6]

    data.append([xmin, ymin, xmax, ymax, confidence, class_id])  # data 리스트에 결과를 추가
    frame_count += 1

capture.release()
cv2.destroyAllWindows()

df = pd.DataFrame(data, columns=['xmin', 'ymin', 'xmax', 'ymax', 'confidence', 'class'])
print(df)
```