YOLOv8

: 2023년 Ultralytic에서 출시한 YOLO(You Only Look Once) 시리즈의 실시간 객체 탐지를 위한 대표적인 딥러닝 모델

특징

1. Anchor-Free 구조

: 기존 YOLOv5/YOLOv7 등의 anchor-based 방식과 달리 YOLOv8에서는 **anchor 없이 동작**

- **Anchor: 객체 탐지 모델에서 각 클래스/스케일별로 사전에 정의된 기준 박스**
- 장점: 더 나은 정확도, 효율적인 탐지 프로세스에 기여, 학습 안정성 향상

2. New Export Formats (새로운 내보내기 포맷 지원)

: 다양한 내보내기(export) 포맷 지원

- 예: ONNX, TensorRT, CoreML, OpenVINO, Paddle, TFLite 등

3. Task 통합 (Unified Task Architecture)

: 객체 탐지(Object Detection), 분할(Segmentation), 분류(Classification) 모델을 하나의 구조 안에서 관리 가능

- 코드 재사용 및 확장성 개선

4. 모델 아키텍처

```
[입력 이미지]
    ↓
  Backbone (특징 추출)
    ↓
   Neck (특징 연결/정리)
    ↓
   Head (박스/클래스 예측)
```

BackBone: 이미지에서 **특징(feature)** 을 추출해주는 **기초 뼈대 네트워크**

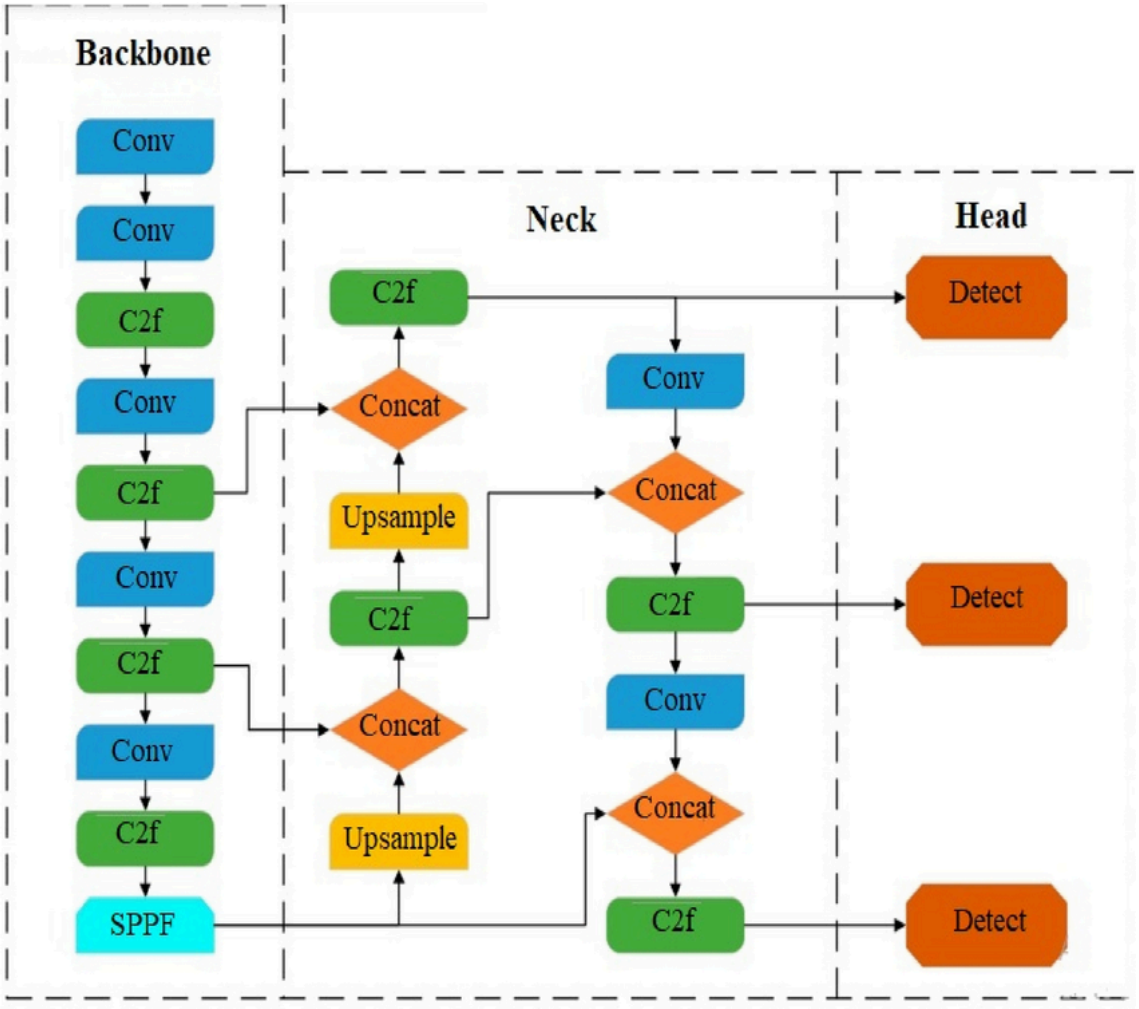
- CSPDarknet(Cross Stage Partial — 입력을 둘로 나눠 일부만 연산한 후 다시 합치는 방식 + Darknet — CNN 계열의 특징 추출기)계열이 아닌 **C2f(Concatenate to fused** — 여러 개의 **BottleNeck block**을 통과한 출력을 전부 **연결**한 후, 마지막에만 한번에 **합쳐**서 출력**) 구조 도입**

Neck: Backbone이 추출한 특징들을 **정리, 결합, 강화**해서 Head로 보내주는 중간 단계

- PAN 구조(Path Aggregation Network — 특징 맵을 **아래에서 위로도 전달**하여 작은 물체도 잘 탐지) 유지 혹은 간소화

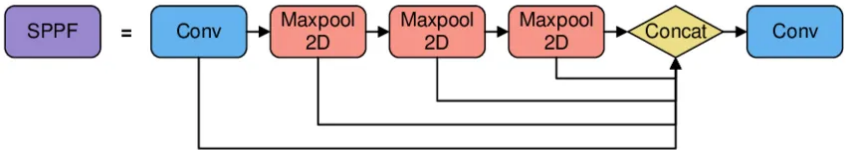
Head: Backbone과 Neck을 거쳐 나온 특징맵(feature map)을 바탕으로 **최종 결과(=박스, 클래스 등)를 예측**하는 부분

- **Detect Head(물체탐지 헤드)가 Anchor-Free 구조에 맞게 변경**



💡 **Backbone**에서 Conv + C2f + SPPF로 특징 추출 → **Neck**에서 Upsample + Concat + C2f로 다양한 해상도 결합 → **Head**의 Detect 블록에서 박스 위치, 클래스 예측

SPPF(Spatial Pyramid Pooling - Fast): 공간피라미드 풀링



Upsample: 해상도(크기)를 ↑ 키움

5. Loss Function 변경

- **BCE(Binary Cross Entropy) + DFL(Distribution Focal Loss** — YOLOv8에서 **좌표 회귀(x, y, w, h)**를 더 정밀하게 하기 위해 **박스 좌표를 정수값이 아닌, 분포(distribution)**로 표현하고 학습**)** 등 개선된 손실 함수 적용

💡 **BCE로 클래스와 객체 존재 여부를 예측**하고, **DFL로 박스 위치를 분포로 예측**

- **CIoU → SIoU**로 대체되기도 함 (보다 정확한 위치 학습)
  - 정답 박스 (Ground Truth Box): 라벨링된 객체의 진짜 위치
  - 예측 박스 (Predicted Box): 모델이 추정한 객체의 위치

항목	설명
IoU	단순 겹침만
CIoU	IoU + 중심 거리 + 비율
SIoU	IoU + 중심 거리 + **방향** + **형태 보정**

▼ 손실함수 부연 설명

- IoU (Intersection over Union): 예측 박스와 정답 박스가 **얼마나 겹치는지 정도**

$$IoU = \frac{\text{예측 박스} \cap \text{정답 박스}}{\text{예측 박스} \cup \text{정답 박스}}$$

$$IoUCost = 1 - IoU$$

- CIoU (Complete IoU): IoU 손실보다 더 정확하게 박스의 **거리, 크기, 비율**까지 고려

$$CIoU = 1 - IoU + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \alpha v$$

  - 박스의 **중심점 사이 거리,** 박스의 **너비와 높이 비율 차이**

항목	의미
IoU	예측 박스와 겹치는 정도
$\rho^2(b, b^{gt})$	중심점 거리 (Euclidean distance)
c	두 박스를 감싸는 최소 박스의 대각선 길이
v	너비/높이 비율 차이 보정 항
α	v 항의 가중치 조절 (IoU에 따라 바뀜)

- SIoU (Scylla IoU): CIoU 손실에 **방향 정보, 형태보정**까지 추가

$$SIoU = \underbrace{\left(1 - \cos(\theta) + \frac{\Delta x^2 + \Delta y^2}{c^2}\right)}_{\text{Distance}} + \underbrace{\left(1 - e^{-\lambda\left(\frac{|w-w^*|}{w^*} + \frac{|h-h^*|}{h^*}\right)}\right)}_{\text{Shape}} + \underbrace{(1 - IoU)}_{\text{Overlap}}$$

  - Distance Cost (거리 비용) (+ Angle Penalty) + Shape Cost (크기 차이 손실) + IoU Cost

항목	의미
θ	예측 박스 중심과 정답 박스 중심이 이루는 **각도**
Δx, Δy	중심 좌표 차이
c	전체 정규화 거리 (예: 박스 대각선 길이)

항목	의미
w, h	예측 박스의 너비와 높이
w^*, h^*	정답 박스의 너비와 높이
λ	조절 계수
IoU	예측 박스와 겹치는 정도

6. 학습 및 추론 코드 개선

- ultralytics 패키지 구조 개선
- CLI와 Python API에서 **매우 간단하게 학습/추론 가능**

7. 자동 하이퍼파라미터 튜닝 지원

- `-optimizer` (옵티마이저), `-lr0` (초기학습률) 등의 자동 설정 기능 향상
- **모델 크기(small, medium 등)** 자동 설정도 지원

8. Yolov8에서 요구하는 데이터 구조

```
dataset/
├── train/
│   ├── images/
│   │   ├── image1.jpg
│   │   ├── image2.jpg
│   │   └── ...
│   └── labels/
│       ├── image1.txt
│       ├── image2.txt
│       └── ...
├── val/
│   ├── images/
│   │   ├── image101.jpg
│   │   └── ...
│   └── labels/
│       ├── image101.txt
│       └── ...
└── data.yaml
```

- 루트 디렉토리 dataset
- train, val 각각의 폴더에 images/, labels/가 함께 존재

기타 응용/비교

1. YOLO-NAS, RT-DETR 등 다른 최신 모델과의 비교

특징/모델	YOLOv8	YOLO-NAS	RT-DETR
기반 구조	CNN + Anchor-free	CNN + NAS 기반 최적화	Transformer 기반

특징/모델	YOLOv8	YOLO-NAS	RT-DETR
실시간 성능	매우 우수 (높은 FPS)	우수 (경량화에 특화)	보통 (속도는 상대적으로 느림)
정확도	우수	경쟁력 있음	복잡 장면에서 매우 우수
모델 크기	다양 (스몰~라지)	매우 경량 모델 가능	비교적 큼
복잡한 장면 처리	제한적 (작은 객체 한계)	중간	뛰어남
사용 편의성	매우 좋음	개선 중	복잡 (Transformer 이해 필요)

2. YOLOv8 한계점

1. **복잡하게 겹쳐 있는 객체들을 정확히 분리하기 어려움**
2. **작은 객체 탐지 어려움**
   - YOLO 구조 특성상 작은 크기의 객체는 특징 추출이 어렵고 탐지 정확도가 떨어질 수 있음
3. **실시간 처리 시 자원 부담**
   - 최신 버전이라 효율적이긴 하나, 고해상도 영상이나 복잡한 모델을 사용할 경우 GPU 메모리와 연산량이 많이 필요함
   - 저사양 환경에서는 실시간 성능 유지가 어려울 수 있음
4. **데이터 편향 문제**
   - 훈련 데이터에 따라 특정 클래스나 환경에 편향되어 잘 작동하지 않는 경우가 있음
   - 예를 들어, 특정 조명, 배경, 카메라 앵글에 대해서는 일반화가 덜 될 수 있음
5. **정확한 위치 보정 한계**
   - 박스 위치나 크기 조정 시 오차가 존재할 수 있으며, 매우 정밀한 위치 검출이 필요한 작업에는 한계가 있음
6. **복잡한 객체의 세분화 불가**
   - YOLOv8은 기본적으로 객체 검출(바운딩박스) 중심이기 때문에, 객체 내 세부 구조(세그멘테이션 등)에는 한계가 있음
   - 물론 세그멘테이션 버전도 있지만, 복잡도와 연산량이 크게 늘어남
7. **외부 환경 변화에 취약**
   - 조명 변화, 날씨, 영상 품질 저하 등 환경 변화에 따라 성능이 급격히 떨어질 수 있음

3. YOLOv8 응용 방향

1. **실시간 객체 탐지 및 추적**
   - CCTV, 드론 영상, 자율주행차 등
2. **산업 현장 자동화 및 품질 검사**
   - 제조 라인에서 불량품 검출, 부품 유무 확인
   - 산업용 로봇의 작업 대상 인식 및 위치 파악
3. **의료 영상 분석**
   - 엑스레이, MRI 등에서 병변이나 이상 부위 탐지
   - 내시경 영상에서 이상 조직 자동 인식
4. **소매 및 유통 분야**

- 매장 내 상품 인식 및 재고 관리
- 고객 행동 분석, 무인 계산대 자동 결제 시스템

5. **농업 환경 모니터링**
    - 농작물 상태 진단, 병충해 탐지
    - 산림, 해양 등 자연 환경 내 특정 동식물 탐지
6. **스마트 시티 및 공공 안전**
    - 공공장소에서 위험 상황(싸움, 낙상 등) 실시간 감지
    - 교통 사고, 불법 주차 등 자동 감지 시스템
7. **증강현실(AR) 및 가상현실(VR)**
    - 실제 환경 내 객체를 인식하여 AR 콘텐츠와 연동
    - 인터랙티브 게임, 교육 콘텐츠 제작
8. **로봇 비전**
    - 서비스 로봇의 주변 환경 인식 및 상호작용 대상 탐지
    - 물류 로봇의 물품 분류 및 위치 파악
9. **교통 관리 및 분석**
    - 차량 번호판 인식, 교통량 분석, 속도 측정
    - 교통 법규 위반 감시