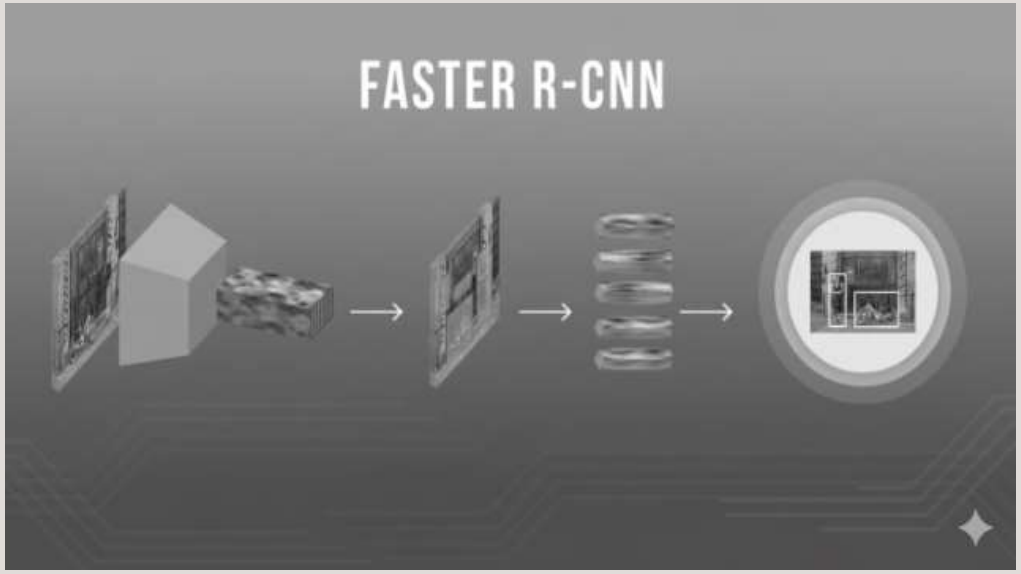
Part 3

모델 연구

”모델 선택 방향

YOLOv8

Faster R-CNN



Part 3

모델 연구

” YOLOv8

```python
from A04 import Execute_Train

if __name__ == "__main__":
    # data_dir = r"D:\01.project\EntryPrj\data\oraldrug\2.drug_no_image_ok_Anno"
    # data_dir = r"D:\01.project\EntryPrj\data\oraldrug\3.drug_ok_Image_no_Anno"

    data_dir = r"D:\01.project\EntryPrj\data\oraldrug\1.drug_Image_annotation_allOK"

    trans_type = ["default", "A", "B"]
    for transform_type in trans_type:
        #
        # 예제 : YOLOv8 Nano - 모든 파라미터 명시
        #
        Execute_Train(
            model_type="yolov8",            # 모델 타입: "yolov8"
            data_dir=data_dir,              # 데이터 디렉토리 경로
            model_size="n",                 # YOLOv8 모델 크기: "n", "s", "m", "l", "x"
            epochs=1,                       # 학습 에포크 수
            batch_size=16,                  # 배치 크기
            lr=0.001,                       # 학습률 (YOLOv8 권장: 0.001)
            bBestLoad=True,                 # Best 모델 로드 여부 (yolobest.pt)
            imgsz=640,                      # 이미지 크기
            patience=10,                    # Early stopping patience (에포크 수)
            train_ratio=0.8,                # 학습/검증 데이터 분할 비율
            num_workers=4,                  # 데이터 로더 워커 수
            transform_type=transform_type   # 데이터 증강 타입: "default", "A", "B"
        )
```

장점

1. 추론 속도 우수 (Real-time 가능) : **단일 단계 구조**로 지연 시간이 짧고 처리량이 높음.
2. 배포 및 운영 용이 : 모델 구조와 추론 파이프라인이 비교적 **단순 ⇒ 서비스 적용(웹/모바일)에 유리**함.
3. 튜닝 포인트가 명확하여 개선 사이클이 빠름 : 개선 방향을 실험적으로 빠르게 반복하기 용이함.

단점

1. 정밀한 박스 품질에서의 한계 : 전반적으로 빠르지만, 객체 경계를 타이트하게 맞춰야 하는 상황에는 2-Stage 계열보다 박스 정밀도가 불리할 수 있음.
2. 객체 밀집 장면에서의 성능 저하 기능
3. 촬영 환경 변화(조명/배경)에 민감 ⇒ **도메인 다양화와 증강 전략 필요.**

Part 3

모델 연구

Faster R-CNN

```python
from A04 import Execute_Train

if __name__ == "__main__":
    # data_dir = r"D:\01.project\EntryPrj\data\oraldrug\2.drug_no_image_ok_Anno"
    # data_dir = r"D:\01.project\EntryPrj\data\oraldrug\3.drug_ok_Image_no_Anno"

    data_dir = r"D:\01.project\EntryPrj\data\oraldrug\1.drug_image_annotation_allOK"

    # ──────────────────────────────────────
    # 예제 : FasterRCNN - 모든 파라미터 명시
    # ──────────────────────────────────────
    trans_type = ["default", "A", "B"]
    for transform_type in trans_type:
        Execute_Train(
            model_type="faster",            # 모델 타입: "faster"
            data_dir=data_dir,              # 데이터 디렉토리 경로
            backbone="resnet50",            # FasterRCNN 백본: "resnet50" 또는 "mobilenet"
            epochs=50,                      # 학습 에포크 수
            batch_size=16,                  # 배치 크기
            lr=0.005,                       # 학습률 (FasterRCNN 권장: 0.005)
            bBestLoad=True,                 # Best 모델 로드 여부 (fasterbest.pt)
            imgsz=540,                      # 이미지 크기
            patience=10,                    # Early stopping patience (에포크 수)
            gubun="partial",                # 파인튜닝 방식: "freeze", "partial", "all"
            train_ratio=0.8,                # 학습/검증 데이터 분할 비율
            num_workers=4,                  # 데이터 로더 워커 수
            transform_type=transform_type   # 데이터 증강 타입: "default", "A", "B"
        )
```

장점

1. 높은 검출 정확도 : **2-Stage 구조**⇒ 전반적으로 **안정적인 정확도 확보 가능.**
2. 작은 물체 검출에 유리한 설계 가능, 복잡한 배경/조명 변화에서 상대적으로 견고.
3. 확장성 : 동일한 프레임워크에서 Mask R-CNN(세그멘테이션), Cascade(정밀도 강화) 등으로 확장 가능하여, 요구 성능에 맞춰 **고도화**하기 좋음.

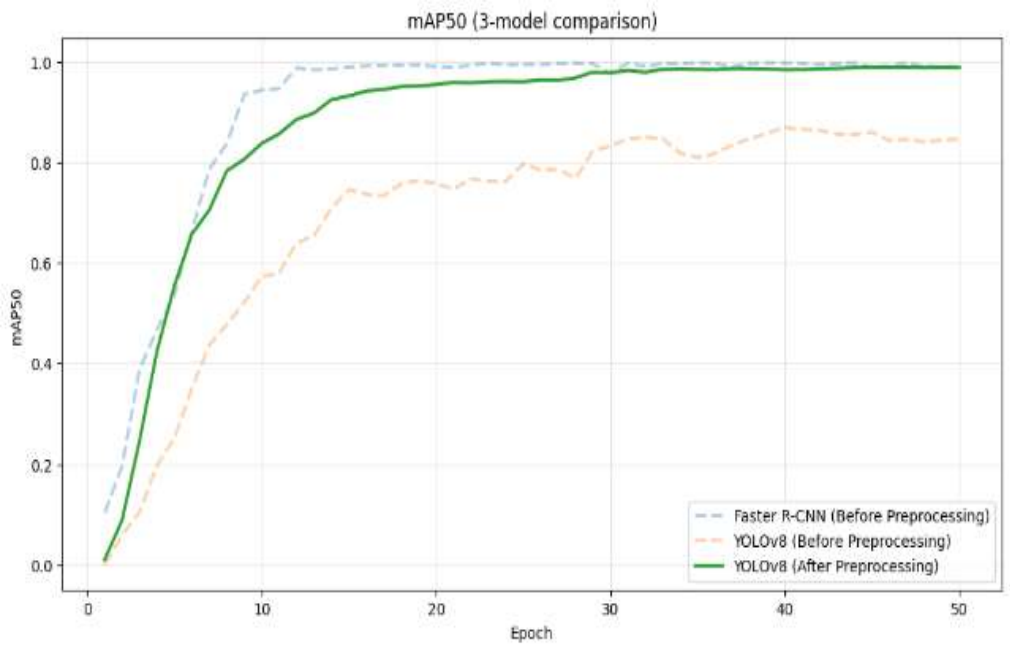
단점

1. 추론 속도 및 연산 비용 부담 : YOLO 대비 추론 지연과 운영 비용이 큼
2. 파이프라인 복잡도 및 튜닝 난이도
3. 실시간 서비스 제약 : 실시간 처리 요구가 높은 서비스(모바일/웹 실시간)에서는 **단독 메인 검출기**로 쓰기 어려움.

Part 3

모델 연구

”모델 선택 전략



- YOLO는 추론 속도와 경량성이 뛰어나 실시간 서비스 및 대량 이미지 처리에서 1차 알약 검출기로 적합하다.
- 반면 Faster R-CNN은 2-Stage 구조로 후보 영역을 정밀하게 판별하여, 작은 알약이나 복잡한 배경에서 정확도 향상을 기대할 수 있다.
- 따라서 운영 관점에서는 **YOLO를 기본 검출기로 사용**하고, **신뢰도가 낮거나 겹침·난이도가 높은 사례에 한해 Faster R-CNN으로 2차 검증** 하는 구성이 효율적이라고 판단된다. ⇒ 이 방식은 전체 처리 속도를 유지하면서도, 어려운 샘플에서의 오탐·미탐을 줄여 최종 성능을 끌어올릴 수 있다.
- 본 프로젝트에서는 제공된 데이터에 따라 **속도 기반 YOLO와 정확도 보강용 Faster R-CNN 두가지 모델을 비교하여 최적의 결과를 도출할 수 있는 모델을 선택하였다.**