KDT 스마트팩토리 과정 7기(CodingOn)

자동차 불량 이미지 데이터를 이용한 자동차 외관 하자 자동 검사 프로그램

팀명 I Team 3조(20세기 청년들)
조원 I 유승태 김다운 한용찬 신예지
멘토 I Ian, Jcastle 리더님

Table of Contents

목차

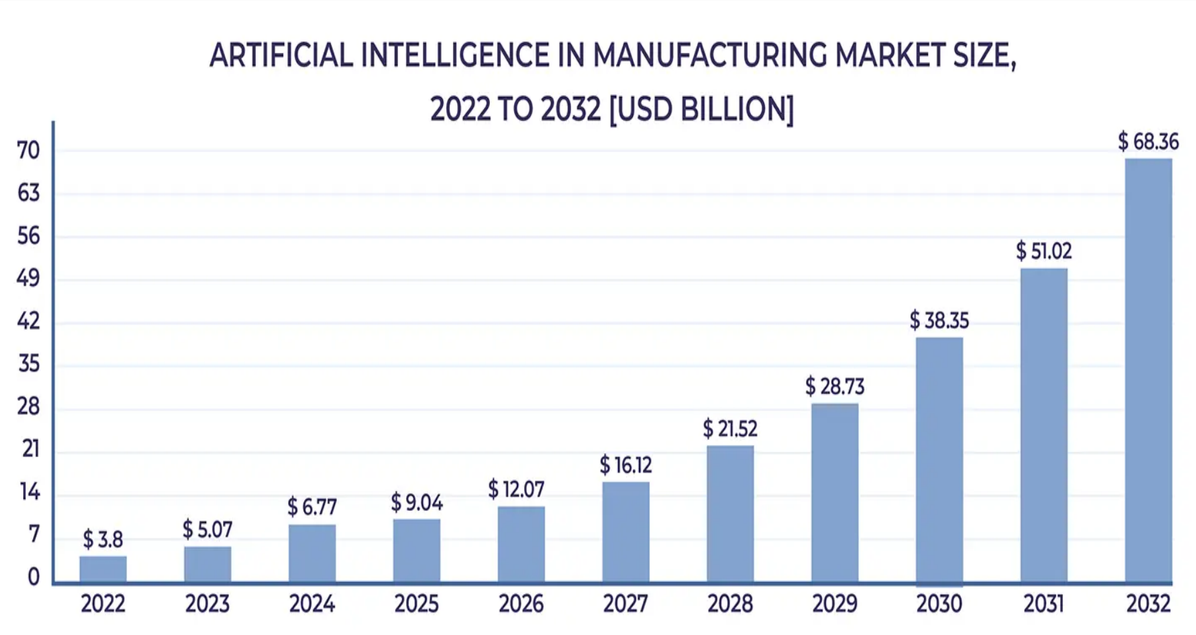
01

프로젝트 개요

Project Introduction

CHAPTER 01

선행연구



AI 제조 시장 규모 예측(2022-2032)

- 글로벌 시장 조사 기관 Market Research Intellect의 조사 결과

- 제조시장에서의 인공지능 기술 도입이 증가하고 있으며, 증가할 것이라는 전망

https://www.marketresearchintellect.com/ko/blog/ai-visual-inspection-systems-driving-precision-and-efficiency-in-automobile-and-transportation-markets/?utm_source=chatgpt.com

CHAPTER 01

주제 및 데이터 선정 배경

1. 프로젝트 주제

자동차 불량 이미지 데이터 기반 머신러닝 모델을 활용한 자동차 외관 하자 검출

2. 선정 이유

1. 트랜드 반영

: 제조 산업 분야에서 머신러닝, AI 기술이 빠르게 확산, 이상감지는 스마트 팩토리 구축에 중요 요소

2. 검사 공정 효율성 증가

: 제조 공정에서 하자 검사는 중요함, 자동화 기술 도입으로 정확도 향상, 비용 절감 및 시간 단축

3. 다방면 산업에 활용 가능

:렌터카, 공유차량 서비스 이용 시 손상 여부 자동 판별하여 운영 효율성 증가

CHAPTER 01

프로젝트 내용

1. 프로젝트 구현 내용

자동차 하자를 판별하는 모델을 제작하고, 웹캠으로 실시간 판별 시스템을 구축

자동차 불량 이미지를 이용해 인공지능 모델(YOLO) 학습 시킴

→

입력된 이미지 차량의 하자 여부를 판별하는 모델 제작

→

OpenCV를 통해 자동차 이미지를 촬영해 하자를 판별하는 시스템 구축

2. 훈련내용과의 연관성

- Python 기반 프로젝트 수행
- OpenCV 기반 프로그램 구현과 이미지 전처리
- 인공지능 모델 YOLO를 이용해 학습

CHAPTER 01

활용 장비 및 재료

1. 활용 장비

- 팀원 개인노트북 4대(운영체제: Window11 2대, Window10 1대, MAC13 1대)
- GPU: NDIVIA RTX 3050 1대 , RTX 4050 1대

2. 개발 환경

- 프로젝트 관리: Git, Notion
- 데이터 전처리: Anaconda 기반 Vscode(Python 3. 12. 0.)
- YOLO 모델 실험 및 프로그램 구현: Anaconda 기반 Vscode(2대), Google Colab(2대)

CHAPTER 01

프로젝트 구조

1. 데이터 수집('25. 1. 15.(수) ~ 1. 17.(금))

- ai허브에서 다운로드(500GB → 25GB 축소) 및 Feature 선정(4개)
- pandas을 통하여 이미지 불러오기

2. 데이터 전처리('25. 1. 18.(토) ~ 1. 21.(화))

- Train, Validity, Test 데이터 6:2:2 비율로 구성
- 라벨데이터 파일 JSON를 CSV 파일로 변환
- YOLO에서 요구하는 데이터 파일 구성

CHAPTER 01

프로젝트 구조

3. YOLO 분류 모델 학습('25. 1. 20.(월) ~ 1. 23.(목))

- YOLO 버전 8 or 11을 대상으로 학습 모델 실험 시행
- 각 팀원 로컬에서 실험을 하여 최적의 모델을 선정

4. OpenCV 기반 프로그램 구현('25. 1. 22.(수) ~ 1. 24.(금))

- 웹캠을 통하여 이미지를 입력받아 분류하는 프로그램 구축
- 실제 차량 대상 실험

CHAPTER 01

활용방안 및 기대효과

1. 스마트 팩토리 분야에 도입 가능

- 센서를 통해 이미지를 입력 받아 인공지능에 접목시켜 분류하는 기술은 스마트 팩토리 구현에 중요한 요소

2. 자동차 분야 뿐만 아니라 데이터에 따라 여러 분야의 검사공정에 적용 가능

- 공정 내 검사 공정에 들어가는 인적 자원 소요 감소 가능
- 그 외 중고를 취급하는 회사에서도 응용 가능

3. 관련 주제 발전 가능성

- 데이터 수집 여부에 따라 타 분야에서도 활용 가능
- GPU 등 리소스 추가를 통하여 고도화 가능

02

프로젝트 팀 구성 및 역할

Project Team Composition and Roles

CHAPTER 02

프로젝트 팀 구성 및 역할

훈련생	역할	담당 업무
유승태	팀장	PM, 모델 학습 및 구축, 성능평가 지표 구현, PPT 작성, 발표
김다운	팀원	데이터 수집 및 전처리, 모델 학습 및 구축, openAI 구현
한용찬	팀원	선행 연구, 데이터 선정, 모델 학습 및 구축, PPT 작성
신예지	팀원	선행 연구, 주제 발의, 모델 학습 및 구축, PPT 작성
이안, 제케 리더님	멘토	주제 선정 간 방향성 제시, 중간발표 피드백

03

프로젝트 수행 절차 및 방법

Project Execution Process and Methods

CHAPTER 03

프로젝트 팀 수행 절차 및 방법

구분	기간	활동	비고
사전 기획	1/15(수)	프로젝트 기획 및 주제 선정	아이디어 선정, 조장 선발
데이터 수집 및 전처리	1/15(수) ~ 1/21(화)	AI허브 내 데이터 다운로드 및 전처리	총 500GB -> 25GB 라벨데이터 변환(JSON-> CSV)
YOLO 분류 모델 학습	1/20(월) ~ 1/23(목)	YOLO 8 or 11 기반 학습 시행	
OpenCV 기반 프로그램 구현	1/22(수) ~ 1/24(금)	웹캠을 통해 이미지를 입력받아 분류하는 프로그램 구축	실제 차량 대상 실험
PPT 작성	1/23(목) ~ 1/24(금)	자료 종합 및 발표준비	발표: 유승태
총 개발기간	1/15(수) ~ 1/24(금) (총 10일)		

04

프로젝트 수행 경과

Project Progress

CHAPTER 04

프로젝트 개요

환경

- 파이썬: 3.11., 3.12.
- 프레임워크: YOLO(이미지 데이터 학습 알고리즘)
- 패키지: Pandas(데이터 전처리), OpenCV(이미지 데이터 입력), scikit-learn(성능 측정)

데이터 개요

- AI허브 내 부품 품질 검사 이미지 데이터(자동차) 503.23GB 중 일부
- 주소: https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&dataSetSn=578

프로젝트 개요

- YOLO 기반 학습 모델 학습
- 차량 불량 데이터 기반 하자 자동 인식 프로그램 구현
- Git주소: https://github.com/smartfactory-project-2/smartfactory_project_2

CHAPTER 04

데이터 전처리

1. 데이터 수집

- AI허브에서 데이터 수집
- 용량의 한계에 따라 일부 다운로드

CHAPTER 04

데이터 전처리

2. 데이터 feature 선정

- 외부의 하자를 표현한 데이터 선택
- 총 4개의 feature 선정

데이터 분류		품질상태			비율
대분류(부품)	소분류(불량유형)	양품	불량품	합계	
도어	스크래치	4,000	6,000	10,000	5.0%
	외관 손상	3,000	3,000	6,000	3.0%
라디에이터 그릴	단차	3,000	3,000	6,000	3.0%
루프사이드	장착 불량	6,000	6,000	12,000	6.0%
배선	고정 불량	6,000	6,000	12,000	6.0%
범퍼	스크래치	4,000	10,000	14,000	7.0%
카울커버	고정핀 불량	6,000	6,000	12,000	6.0%
	연계 불량	6,000	6,000	12,000	6.0%
커넥터	유격 불량	6,000	9,000	15,000	7.5%
	체결 불량	6,000	12,000	18,000	9.0%
테일 램프	단차	7,000	7,000	14,000	7.0%
프레임	외관 손상	3,000	4,000	7,000	3.5%
	실링 불량	2,000	4,000	6,000	3.0%
	헤밍 불량	5,000	5,000	10,000	5.0%
	홀 변형	4,000	4,000	8,000	4.0%
헤드 램프	단차	7,000	7,000	14,000	7.0%
휀더	외관 손상	4,000	4,000	8,000	4.0%
	단차	8,000	8,000	16,000	8.0%
합계		90,000	110,000	200,000	100.0%

CHAPTER 04

데이터 전처리

3. YOLO 학습모델용 데이터 전처리

- Pandas을 통하여 이미지 불러오기
- Label 데이터 json -> csv
- 텍스트 파일 만들기
- YAML 파일 생성
- YOLO의 BBOX는 중앙값 좌표 2개, 길이, 높이로 구성되어 있어 변환이 필요

```
dataset/
├── train/                  # 학습 데이터 디렉토리
│   ├── images/             # 학습 이미지 폴더
│   │   ├── image1.jpg
│   │   ├── image2.jpg
│   │   └── ...
│   └── labels/             # 학습 라벨 폴더
│       ├── image1.txt
│       ├── image2.txt
│       └── ...
├── val/                    # 검증 데이터 디렉토리
│   ├── images/             # 검증 이미지 폴더
│   │   ├── image1.jpg
│   │   ├── image2.jpg
│   │   └── ...
│   └── labels/             # 검증 라벨 폴더
│       ├── image1.txt
│       ├── image2.txt
│       └── ...
└── data.yaml               # 데이터셋 설정 파일
```

CHAPTER 04

데이터 전처리

3. YOLO 학습모델용 데이터 전처리

- Pandas을 통하여 이미지 불러오기
- Label 데이터 json -> csv
- 텍스트 파일 만들기
- YAML 파일 생성
- YOLO의 BBOX는 중앙값 좌표 2개, 길이, 높이로 구성되어 있어 변환이 필요

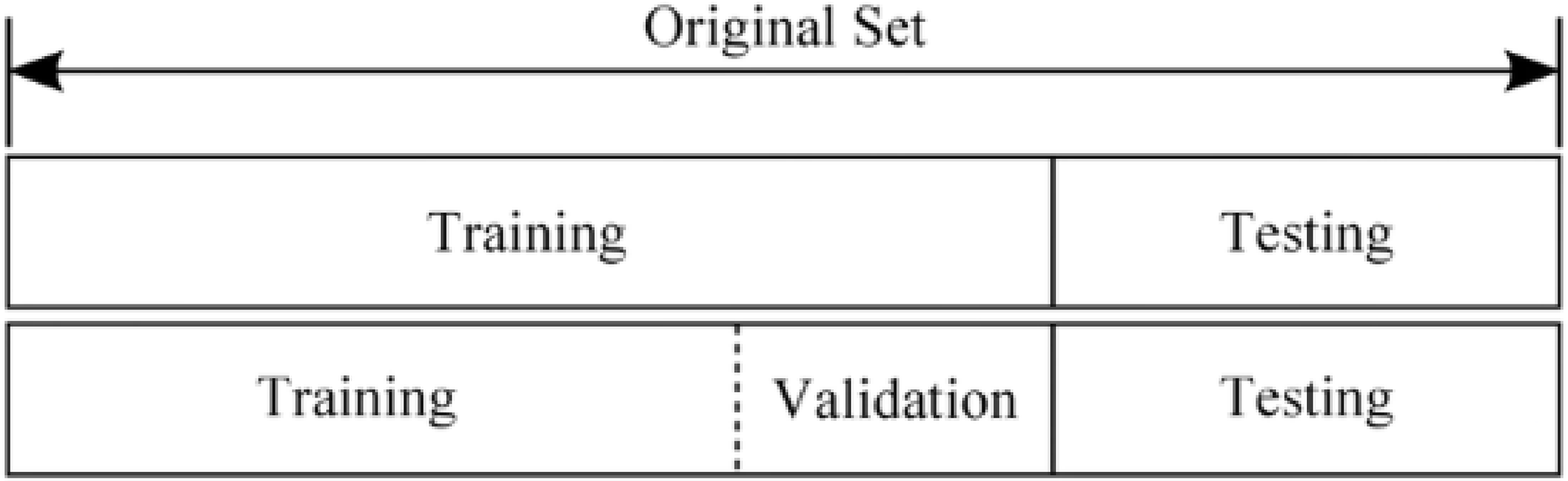
```yaml
import yaml

# YAML 파일에 저장할 데이터
yaml_data = {
    "train": "C:/Users/Administrator/sf_project_2/train/images",
    "test": "C:/Users/Administrator/sf_project_2/test/images",
    "val": "C:/Users/Administrator/sf_project_2/valid/images",
    "nc": 2,
    "names": ["ok", "faulty_frame",],
    "augment": {
        "flipud": 0.3,
        "fliplr": 0.3,
        "rotate": 5,  # 회전 범위 증이기
        "mosaic": 0.5,  # 모자이크 비율 증이기
        "translate": 0.2,
        "scale": 0.7,
        "random_shadow": 0.4,  # 그림자 효과를 증가시켜 밝기 변화에 강하게
        "random_fog": 0.3,  # 안개 효과를 늘려서 다양한 시나리오 학습
        "crop": 0.3,  # 잘라내기 영역을 증가시켜 모델의 일반화 성능 향상
        "brightness": 0.4,  # 밝기 변화 범위 증가
        "contrast": 0.4,  # 대비 범위 증가
        "sharpen": 0.3,  # 이미지 샤프닝 강도를 높임
        "mixup": 0.2,  # mixup 비율 증이기
        "cutmix": 0.1,  # cutmix 비율 증이기
        "motion_blur": 0.2,  # motion blur 적용 강도 증이기
        "gauss_noise": 0.2,  # 노이즈 강도 증이기
        "hsv_h": 0.015,  # 색조 변화 증가
        "hsv_s": 0.7,  # 채도 변화 증가
        "hsv_v": 0.4,  # 명도 변화 증가
        "shear": 20,  # 전단 변환 강도 증가
        "perspective": 0.2,  # 투시 변환 추가
    },
}

# YAML 파일 생성
with open("data.yaml", "w") as yaml_file:
    yaml.dump(yaml_data, yaml_file, default_flow_style=False, sort_keys=False)

print("YAML 파일이 성공적으로 생성되었습니다.")
```

CHAPTER 04

데이터 전처리

4. 데이터 분리

- Train, Validation, Test 데이터를 각각 6:2:2 비율로 나눠서 학습 시행



https://neocarus.tistory.com/entry/%EB%8D%B0%EC%9D%B4%ED%84%B0-%EB%B6%84%ED%95%A0-%ED%95%99%EC%8A%B5%EB%8D%B0%EC%9D%B4%ED%84%B0-%EA%B2%80%EC%A6%9D%EB%8D%B0%EC%9D%B4%ED%84%B0-%ED%85%8C%EC%8A%A4%ED%8A%B8%EB%8D%B0%EC%9D%B4%ED%84%B0%EC%9D%98-%EA%B0%81-%EC%97%B4%ED%95%A0%EC%9D%80

CHAPTER 04

YOLO 모델 학습

1. 모델 구상

中文 | 한국어 | 日本語 | Русский | Deutsch | Français | Español | Português | Türkçe | Tiếng Việt | العربية

Ultralytics CI passing | downloads 71M | DOI 10.5281/zenodo.7347926 | Discord 728 online | Forums 256 users | Reddit 304

Run on Gradient | Open in Colab | Open in Kaggle | launch binder

소개 Ultralytics YOLO11는 호평을 받고 있는 실시간 물체 감지 및 이미지 분할 모델의 최신 버전입니다. YOLO11 는 딥 러닝과 컴퓨터 비전의 최첨단 발전을 기반으로 구축되어 속도와 정확성 측면에서 비교할 수 없는 성능을 제공합니다. 간소화된 디자인으로 다양한 애플리케이션에 적합하며, 엣지 디바이스에서 클라우드 API에 이르기까지 다양한 하드웨어 플랫폼에 쉽게 적용할 수 있습니다.

Ultralytics 문서에서 기능을 이해하고 활용하는 데 도움이 되는 종합적인 리소스를 살펴보세요. 이 허브는 숙련된 머신 러닝 실무자이든 이 분야를 처음 접하는 사람이든 프로젝트에서 YOLO 의 잠재력을 극대화하는 것을 목표로 합니다.

https://docs.ultralytics.com/ko

CHAPTER 04

YOLO 모델 학습

1. 모델 구상

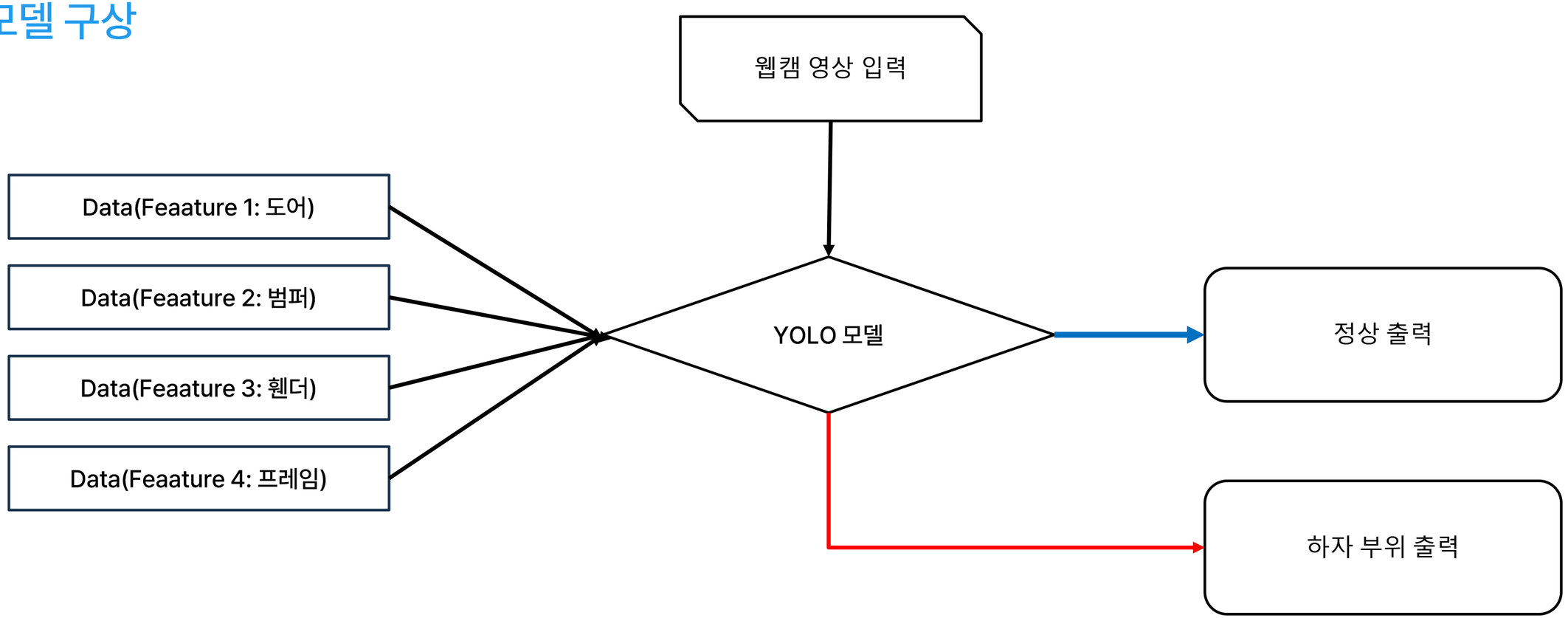
1. **YOLOv3**: The third iteration of the YOLO model family, originally by Joseph Redmon, known for its efficient real-time object detection capabilities.
2. **YOLOv4**: A darknet-native update to YOLOv3, released by Alexey Bochkovskiy in 2020.
3. **YOLOv5**: An improved version of the YOLO architecture by Ultralytics, offering better performance and speed trade-offs compared to previous versions.
4. **YOLOv6**: Released by Meituan in 2022, and in use in many of the company's autonomous delivery robots.
5. **YOLOv7**: Updated YOLO models released in 2022 by the authors of YOLOv4.
6. **YOLOv8**: The latest version of the YOLO family, featuring enhanced capabilities such as instance segmentation, pose/keypoints estimation, and classification.
7. **YOLOv9**: An experimental model trained on the Ultralytics YOLOv5 codebase implementing Programmable Gradient Information (PGI).
8. **YOLOv10**: By Tsinghua University, featuring NMS-free training and efficiency-accuracy driven architecture, delivering state-of-the-art performance and latency.
9. **YOLO11** 🚀 **NEW**: Ultralytics' latest YOLO models delivering state-of-the-art (SOTA) performance across multiple tasks.
10. **Segment Anything Model (SAM)**: Meta's original Segment Anything Model (SAM).
11. **Segment Anything Model 2 (SAM2)**: The next generation of Meta's Segment Anything Model (SAM) for videos and images.
12. **Mobile Segment Anything Model (MobileSAM)**: MobileSAM for mobile applications, by Kyung Hee University.

https://docs.ultralytics.com/models/#featured-models

CHAPTER 04

YOLO 모델 학습

1. 모델 구상



CHAPTER 04

YOLO 모델 학습

2. 모델 평가 지표

- 혼동행렬, 정확도, 정밀도, 재현율을 선정

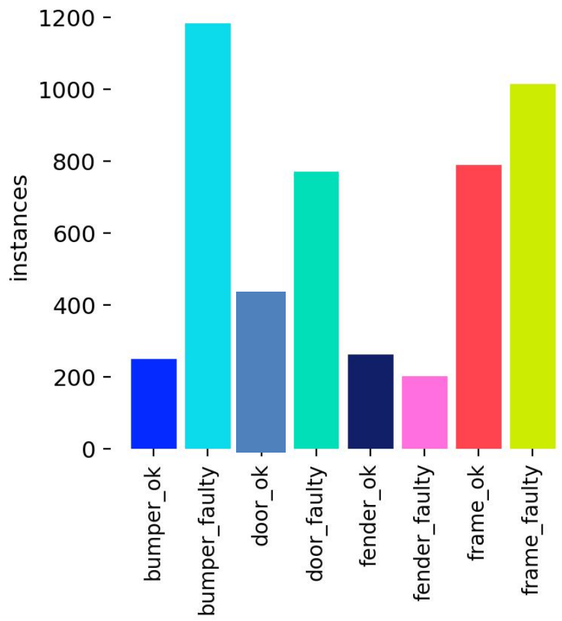
```python
# 혼동 행렬
confmat = confusion_matrix(y_true=y_test, y_pred=pred)
print('혼동 행렬: ', confmat, sep='\n')

# 정확도
accuracy = accuracy_score(y_true=y_test, y_pred=pred)
print('정확도:', accuracy)

# 정밀도(precision)
precision = precision_score(y_true=y_test, y_pred=pred, average='macro')
print('정밀도: ', precision)

# 재현율(recall)
recall = recall_score(y_true=y_test, y_pred=pred, average='macro')
print('재현율: ', recall)
```

```
혼동 행렬:
[[194   6]
 [ 22 178]]
정확도: 0.93
정밀도:  0.932769726247987
재현율:  0.9299999999999999
```

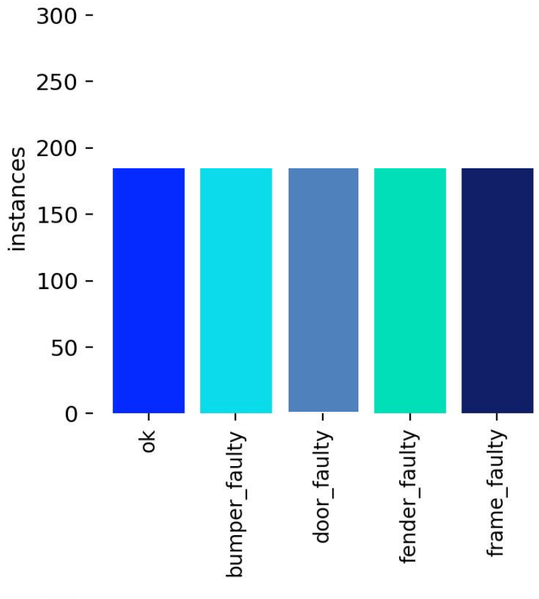
CHAPTER 04

YOLO 모델 학습

3. 모델 최적화 – 데이터 수 불균형 해소



초기 학습했던 불균형한 데이터



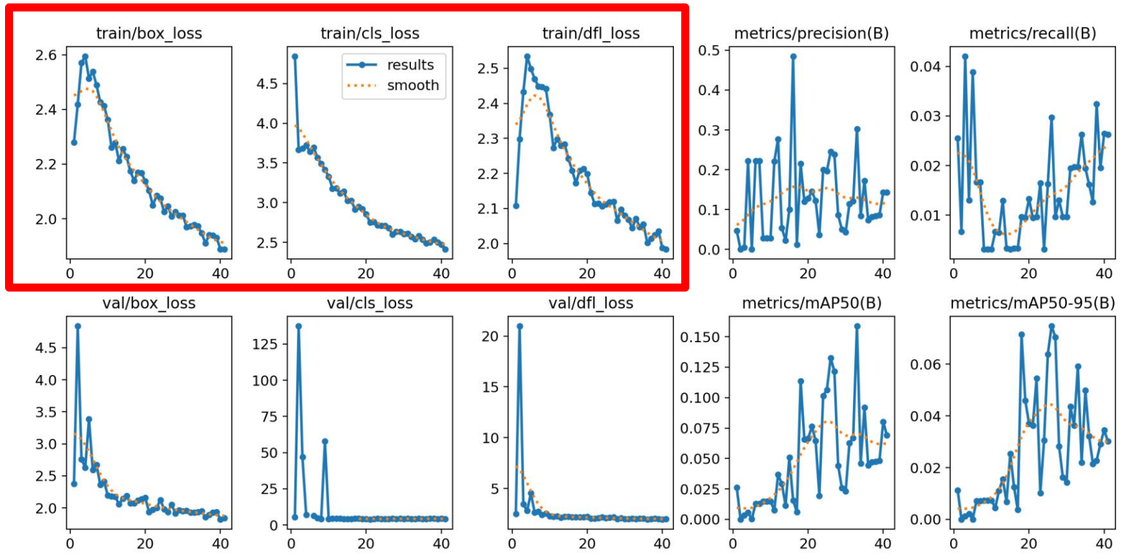
개선한 데이터

각 Feature마다의 데이터 수를 균형있게 배치 -> 유의미한 성능 개선 확인

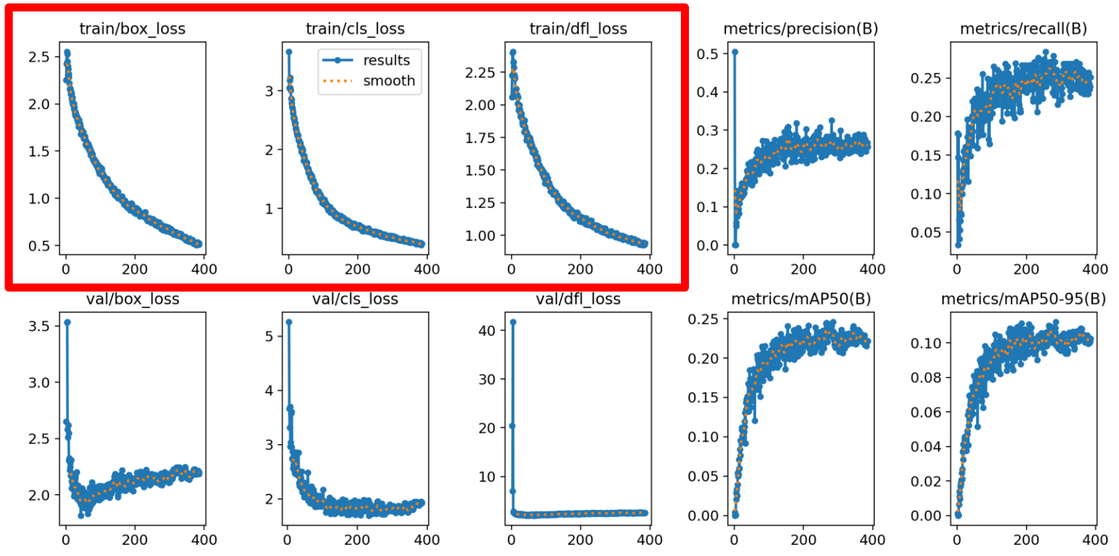
CHAPTER 04

YOLO 모델 학습

3. 모델 최적화 – 반복 수 증가



epoch 40



epoch 400

반복수 증가를 통하여 Loss값 감소 및 성능 개선 확인 BUT 학습에 너무 많은 시간 소요

CHAPTER 04

YOLO 모델 학습

3. 모델 최적화 – 하이퍼파라미터 조정

- 임계값(Confidence Threshold) -> (0.2 ~ 0.5)
- 초기 학습률
- 학습률 범위
- 배치 크기
- 입력 이미지 크기

하이퍼파라미터를 조정하며 성능 개선
-> Threshold가 성능에 미치는 영향이 제일 컸음

```python
from ultralytics import YOLO

# 모델 로드
model = YOLO("yolo11s.pt")

# 학습 설정
model.train(
    data="data.yaml",  # 데이터 설정 파일 경로
    epochs=50,  # 에폭 수를 늘려서 더 많은 학습 진행
    batch=20,  # 배치 크기 늘리기
    imgsz=640,  # 입력 이미지 크기 512로 조정 (성능 및 메모리 효율 고려)
    device="cuda",  # CUDA GPU 사용 (GPU를 사용하려면 'cuda'로 설정)
    workers=4,  # 데이터 로드 병렬 처리 워커 수
    project="C:/Users/Administrator/sf_project_2",     # 결과 저장 디렉토리
    name="model_11_s_50_20_640_frame(thres0.2)",
    lr0=0.001,  # 초기 학습률0.0001
    lrf=0.1,      # 학습률 범위
    conf=0.2,     # Confidence threshold
#    patience=10,  # 성능 개선이 없을 때 조기 종료 (10 에포크 대기)
    augment=True,  # 데이터 증강 활성화
    optimizer="Adam",  # AdamW 옵티마이저 사용
    cos_lr=True,  # 코사인 학습률 스케줄링
    momentum=0.6,  # 모멘텀
    cache=True,  # 데이터 로딩 속도 개선
)

# 학습 완료 후 모델 저장
model.save('model_11_s_50_20_640_frame(thres0.2).pt')
```

CHAPTER 04

YOLO 모델 학습

3. 모델 최적화 – 이진 분류 시도

- Feature 중 두개를 선정하여 이진 분류 실험

	도어 외장손상	도어 스크래치
최적 임계값	0.5	0.35

데이터마다 특성이 달라 최적의 임계값 또한 다름

↓

데이터의 분포 차이가 있어 결정경계를 구하는데 어려움이 있었다고 판단

↓

임계값 차이,정규화 부족

그러나

이를 해결하기 위한 다양한 전처리를 대입하는데 시간이 부족하였음

CHAPTER 04

YOLO 모델 학습

4. 앙상블 모델

- 각각 feature 별 이진 분류 실험

범퍼	정상	불량품
도어 스크래치	정상	불량품
도어 외장손상	정상	불량품
프레임	정상	불량품
휀더	정상	불량품

범퍼	정상	불량품
도어 스크래치	정상	불량품
도어 외장손상	정상	불량품
프레임	정상	불량품
휀더	정상	불량품

총 5종류의 데이터를 각각의 모델에 학습시킴

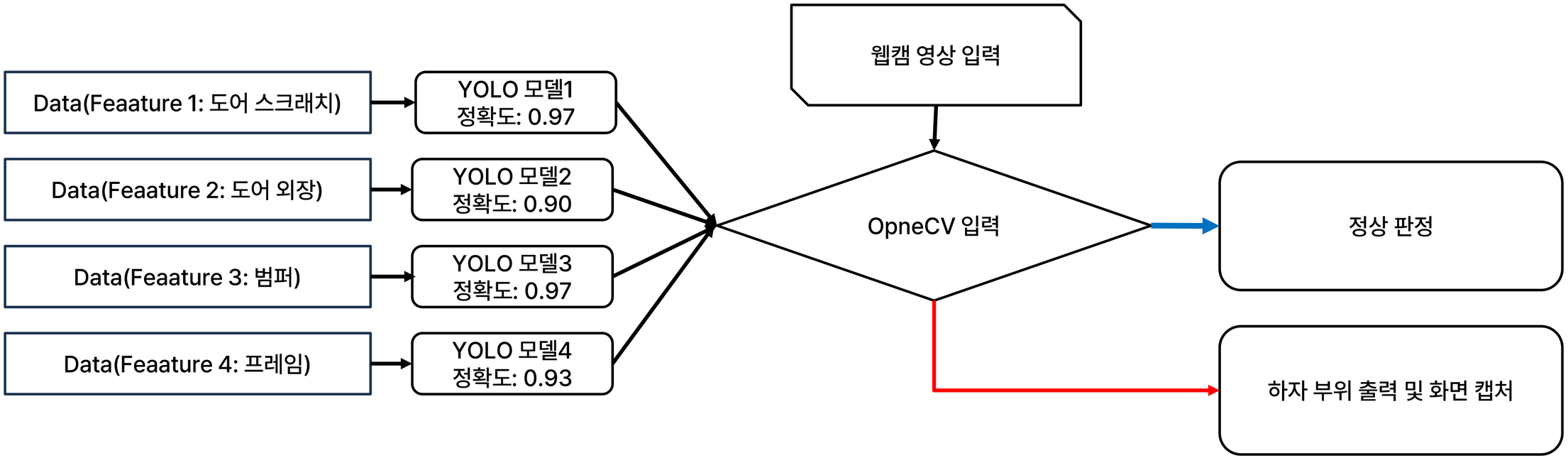
-> 그 중 4개의 모델에서 정확도 0.9이상의 성능을 보임

CHAPTER 04

YOLO 모델 학습

4. 앙상블 모델

- 모델 4개로 앙상블 모델 구성
- 하나라도 불량을 감지하면 불량으로 탐지(OR 방식의 Hard Voting)



CHAPTER 04

프로그램 구현

CHAPTER 04

고찰

1. 리소스 및 시간의 부족에 따른 실험의 한계
2. 데이터 전처리 경험 부족
3. 실제 현장 도입시에는 더 높은 성능이 필요함

05

자체 평가 의견

Self-Assessment and Feedback

CHAPTER 05

자체 평가 의견

유승태	직접 기획하고 데이터를 가져와서 모델을 돌리는데 있어 가장 힘들었던 부분은 모델을 실험하는 것이었습니다. 머신러닝이라는 블랙박스를 해석하는데 어려움이 있었으나, 이를 생각해볼 수 있는 경험을 해보았다는 것에 큰 의의를 가질 수 있다고 생각합니다.
김다운	초반 모델 학습 과정에서 성능 변화를 정확히 파악하지 못해, 학습 중에 파라미터를 적절히 조절하는 데 어려움을 겪었으며, 부품 별 임계값 설정의 중요성을 놓쳐 최종 모델의 성능을 개선하지 못한 점이 아쉬웠습니다.
한용찬	Colab 환경에서 데이터 추출에 많은 시간이 소모되어 모델 학습 과정에서 어려움을 겪었다. 제한된 시간적 여건으로 인해 최적의 모델을 구현하지 못한 점이 아쉬웠습니다.
신예지	예상보다 낮은 모델 성능으로 어려움을 겪었는데 클래스 간의 특징이 달라 성능 향상에 문제가 생긴 것으로 판단하였고, 훈련데이터 패턴에 맞는 전처리를 하거나 모델 구조를 개선에 여지가 발견했으나 시간이 부족해 아쉬웠습니다.

감사합니다.