X-ray검사장비 AI 데이터셋 (yolov3 성능 테스트)

COF(정형준, 조석현)

AI 데이터셋 개요

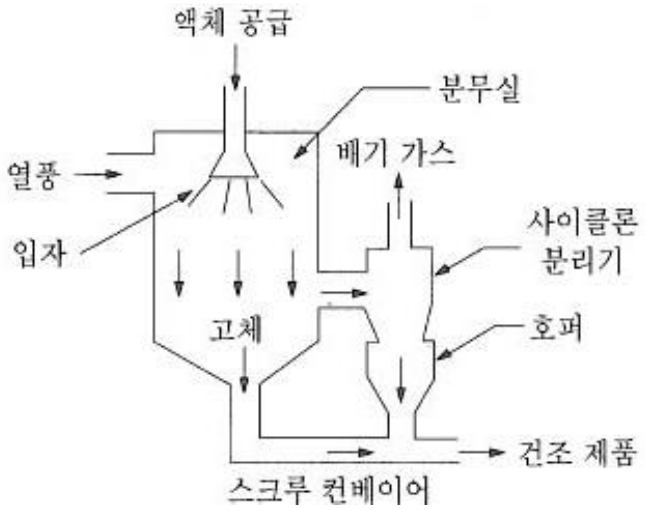
업종 : 식품가공(분무건조공법을 이용한 분말유크림 제조)

목적 : 품질 보증

유형 : bmp

알고리즘 : YOLOv3

제공기관 : KAIST



배경 및 분석 목적

포장 공정의 마지막 단계에 완제품 내 금속 및 이물 포함 여부를 검출하기 위해 진행되는 X-RAY 비전 검사에서 미검(불량이 양품)과 과검(양품이 불량)의 문제를 해결 하기위해 AI 분석을 통해 장비 셋팅, 검사기준 또는 여타 스펙을 개선 할 수 있는 학습 모델이 필요하게 되었다.

검사장비를 교체하지 않는 한 해결될 수 없는 근본적 기술적 한계를 보완하기 위해 검사장비에서 수집되는 X-RAY 사진이미지를 라벨링하고 학습한 객체 탐지 분석 모델을 개발하여 이를 바탕으로 포장제품의 양품/불량품을 정확하게 판정하고자 한다.

사용 프로그램

Python(3.9.18)

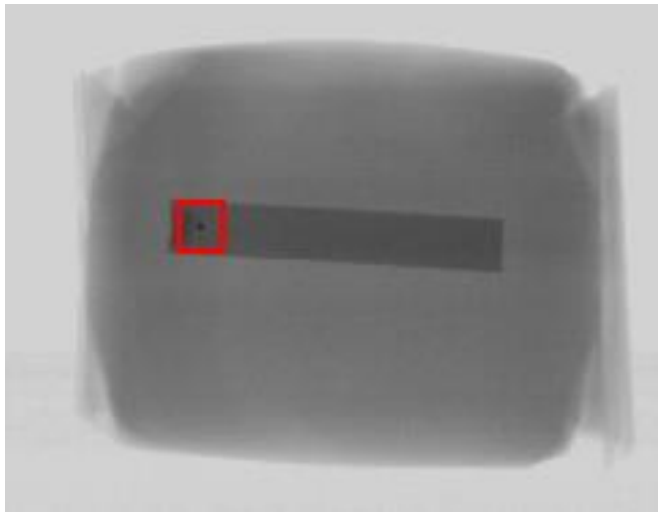
(anaconda)Jupyter notebook(7.0.8)

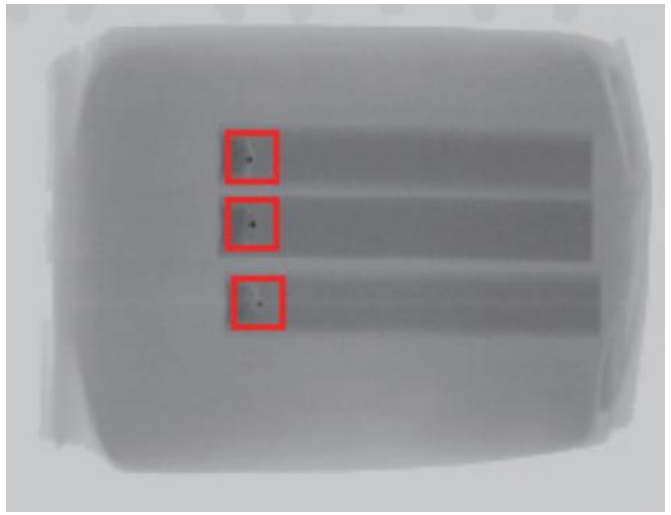
YOLOv3(Cartucho/OpenLabeling: Label images and video for Computer Vision applications (github.com))

Pytorch(1.12.1)

데이터

- 데이터 크기, 데이터 수량 (확장자.bmp)
  - [1호기] 폴더 – 1,031개, 112MB
  - [2호기] 폴더 – 920개, 92.9MB
  - [3호기] 폴더 – 858개, 210MB





AI 분석모델 YOLOv3 선정이유

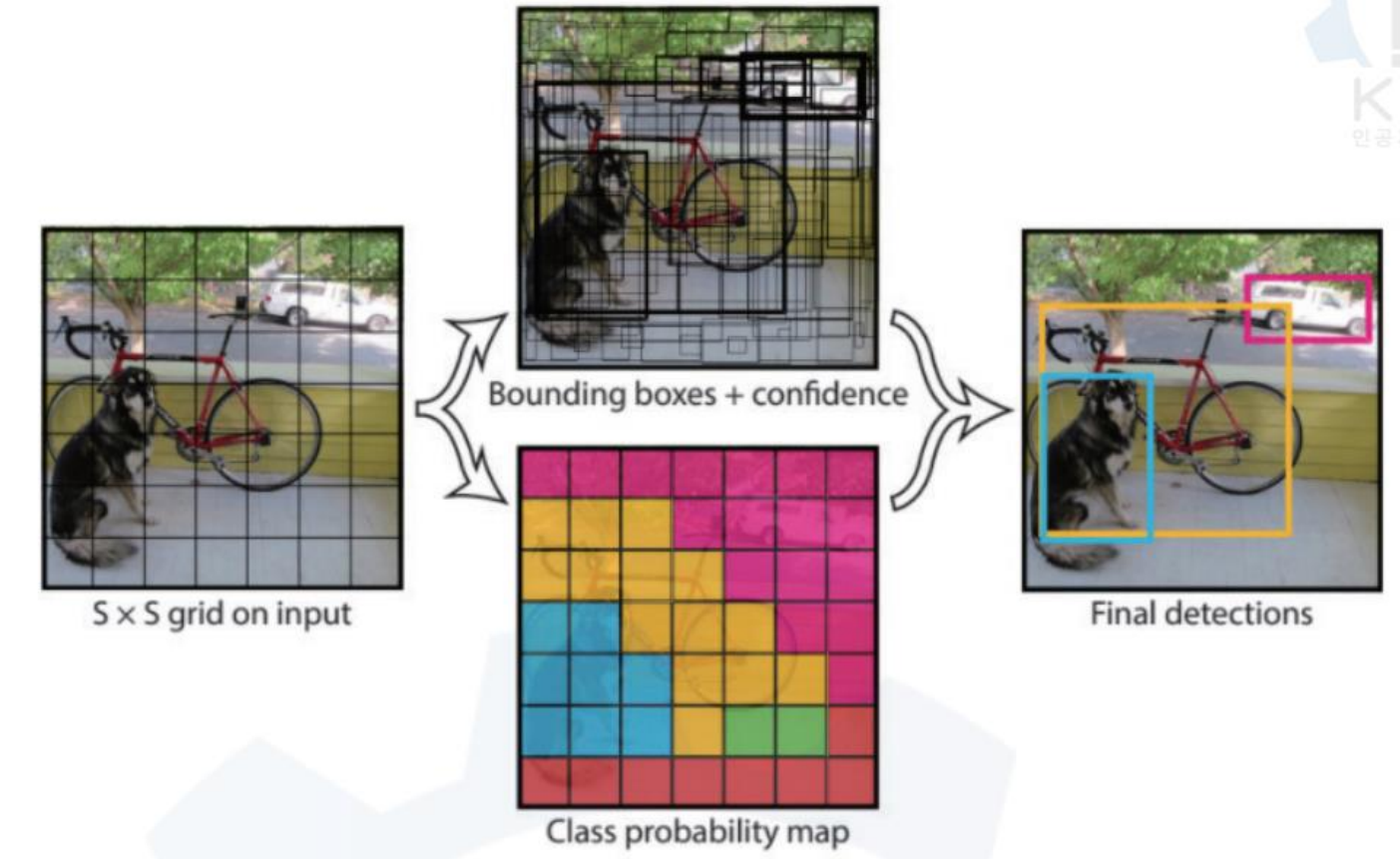
이중 삼중 검사 시 라인 흐름의 버퍼와 작업 동선등의 변화로 비용 낭비를 막기위한 타 모델보다 정확하면서 신속한 AI 모델 이여아 한다.

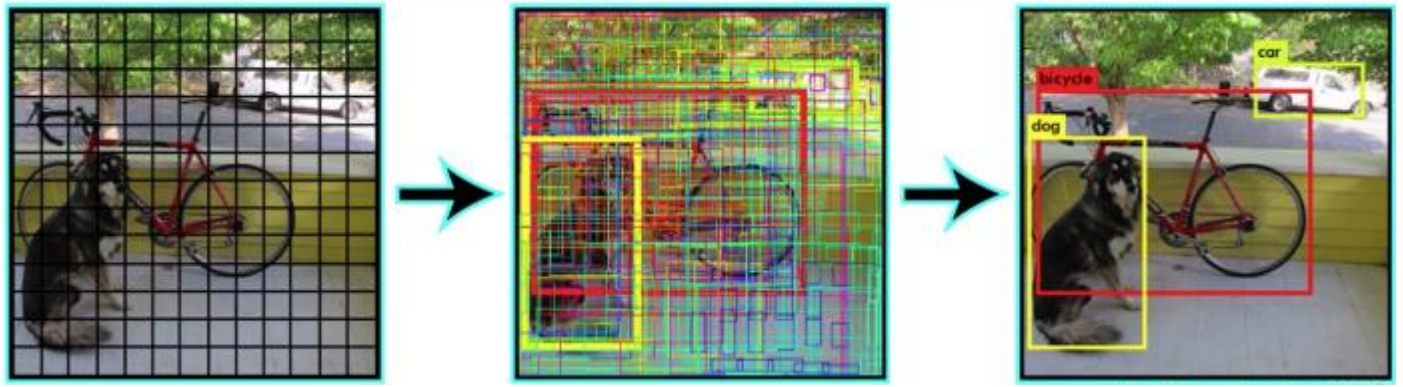
각 이미지에는 검출된 결함이 표시되어 있으며, 이러한 이미지에 대 하여 객체 탐지 방법론을 적용하여 X-RAY 이물검출기에서 확인된 결함을 객체로 탐 지하여 불량을 구분해 낼 수 있어야 한다.

객체를 둘러싸고 있는 경계 상자(bounding box)와 그 내부의 객체에 대한 종류 (class)를 예측하는 회귀(regression) 문제로 객체 탐지(object detection)를 수행하는 모델이기 때문이다.

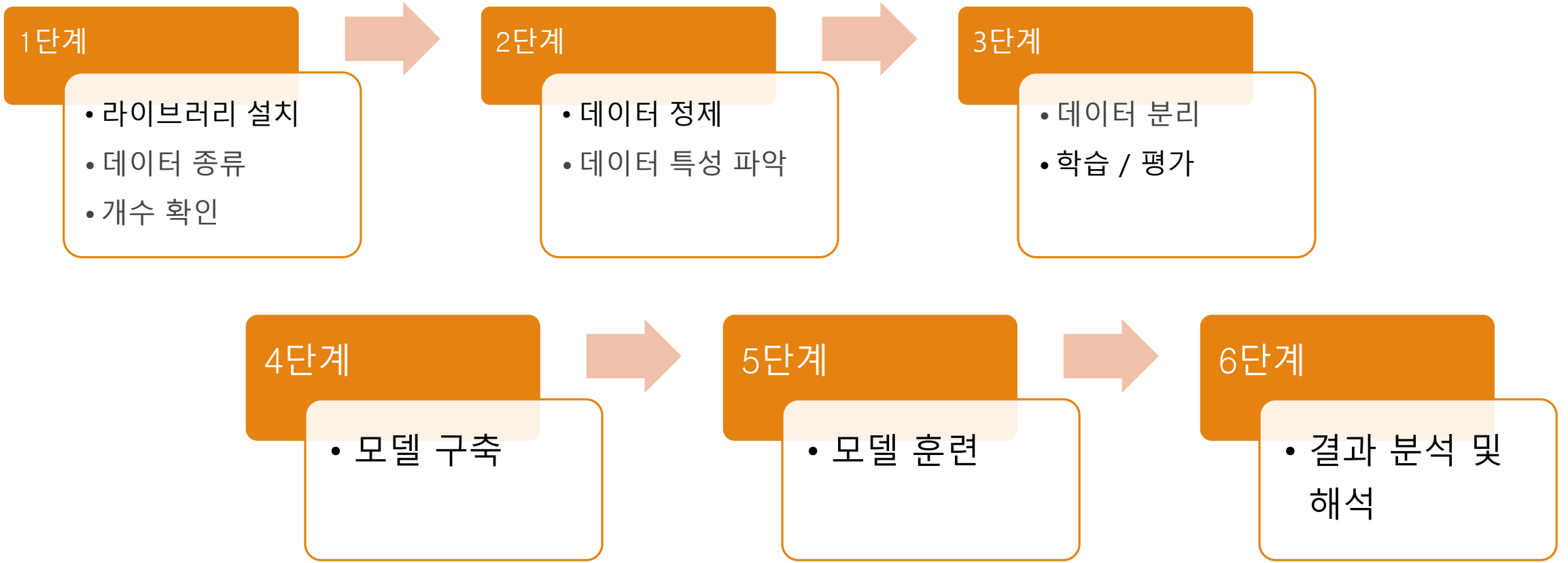
YOLOv3 방식

그리드로 나누어 그리드 별 작업

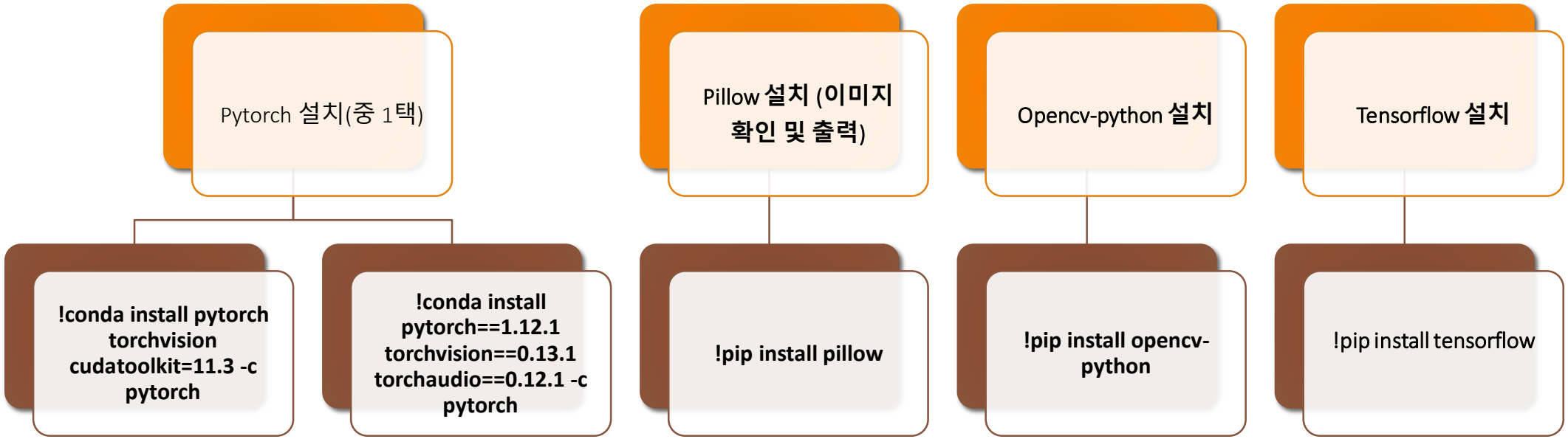




분석 단계

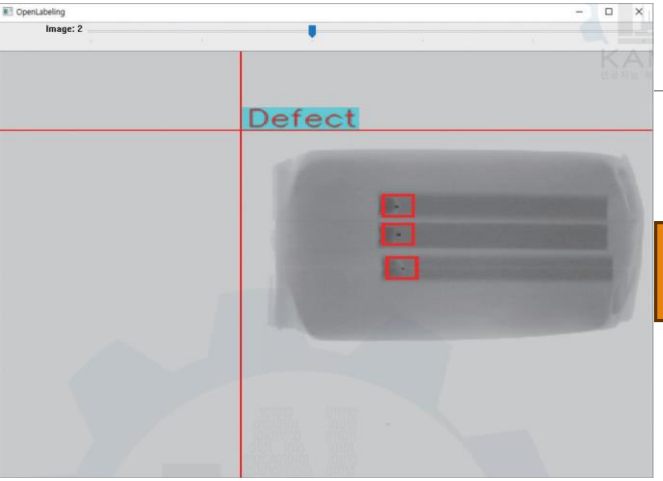

1단계
• 라이브러리 설치
• 데이터 종류
• 개수 확인
2단계
• 데이터 정제
• 데이터 특성 파악
3단계
• 데이터 분리
• 학습 / 평가
4단계
• 모델 구축
5단계
• 모델 훈련
6단계
• 결과 분석 및 해석

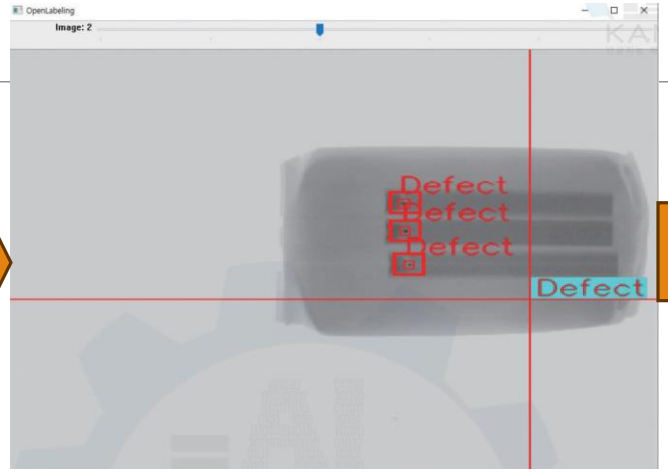
1단계 - 라이브러리 설치

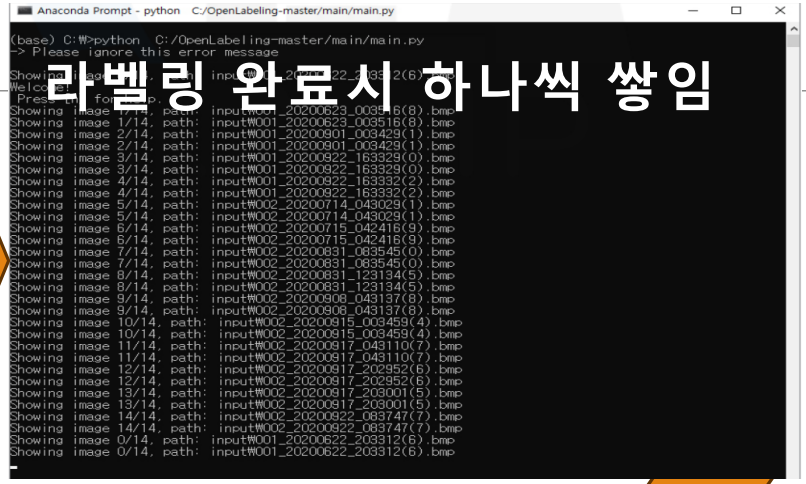

Pytorch 설치(중 1택)
!conda install pytorch torchvision cudatoolkit=11.3 -c pytorch
!conda install pytorch==1.12.1 torchvision==0.13.1 torchaudio==0.12.1 -c pytorch
Pillow 설치 (이미지 확인 및 출력)
!pip install pillow
Opencv-python 설치
!pip install opencv-python
Tensorflow 설치
!pip install tensorflow

2단계 - 데이터 정제(전처리)

결함 데이터 좌표입력을 위한 Labeling 실행

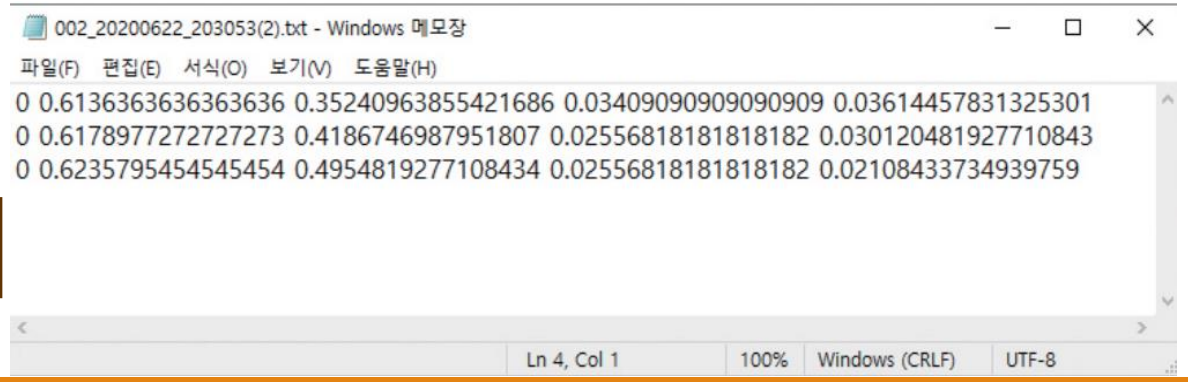






라벨링 완료시 하나씩 쌓임

라벨링 된 이미지 내 사각형 좌표



Train과 test를 나눌때
splitdata.py 가 jpg 만 지원 됨

```
# 디렉토리 경로 설정
%cd C:/test1/yolov3/images
!ren *.* *.jpg*
```

3단계-학습/평가 데이터 분리

특징은 이미지 파일 위치 데이터만 train.txt, test.txt에 들어가서 나뉜다는것 입니다.

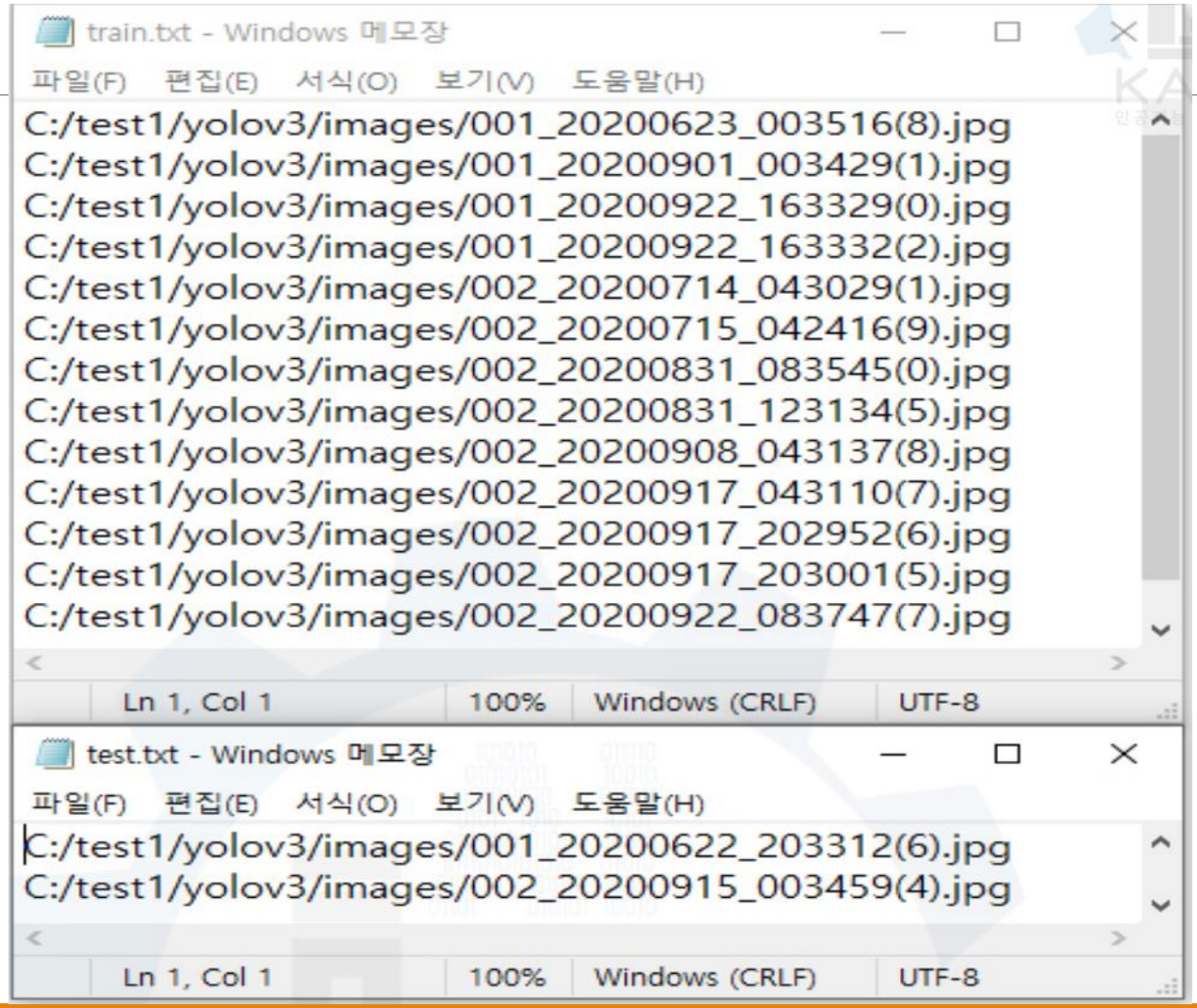
```
import os
import random

def split_data_set(image_dir):
    # 테스트 세트를 저장할 파일을 'test.txt'로, 훈련 세트를 저장할 파일을 'train.txt'로 엽니다.
    f_val = open("test.txt", 'w')
    f_train = open("train.txt", 'w')

    # 지정된 디렉토리에서 파일 목록을 가져옵니다.
    path, dirs, files = next(os.walk(image_dir))
    data_size = len(files)  # 전체 데이터 크기를 계산합니다.

    ind = 0
    data_test_size = int(0.2 * data_size)  # 전체 데이터의 20%를 테스트 세트 크기로 설정합니다.
    test_array = random.sample(range(data_size), k=data_test_size)  # 테스트 세트에 사용할 파일 인덱스를 무작위로 선택합니다.

    # 디렉토리 내의 모든 파일을 순회합니다.
    for f in os.listdir(image_dir):
        if f.endswith('.jpg'):  # 파일 확장자가 .jpg인 경우만 처리합니다.
            ind += 1

            # 파일 인덱스가 테스트 배열에 포함되어 있으면 테스트 세트로, 그렇지 않으면 훈련 세트로 분류합니다.
            if ind in test_array:
                f_val.write(image_dir + '/' + f + '\n')  # 테스트 세트 파일에 이미지 경로를 기록합니다.
            else:
                f_train.write(image_dir + '/' + f + '\n')  # 훈련 세트 파일에 이미지 경로를 기록합니다.

    # 파일을 닫아 리소스를 해제합니다.
    f_val.close()
    f_train.close()
```

- 분석에 쓰일 이미지 데이터셋이 저장되어 있는 디렉토리 지정
- 데이터 나누기
- 이미지를 읽어 각각의 이미지 위치를 입력하고 각각의 데이터셋 저장

```
split_data_set(image_dir)
```

- 학습 및 평가 데이터 셋이 분리된 것을 확인하면 'train.txt', 'test.txt'로 나누어져있다.

3단계-학습/평가 데이터 분리



4단계-모델 구축

```
classes= 1
train= 'C:/test1/yolov3/images/train.txt'
valid= 'C:/test1/yolov3/images/valid.txt'
names= 'C:/test1/yolov3/data/classes.names'
```

클래스 개수 설정

학습 데이터셋 위치 설정

평가(검증) 데이터셋 위치 설정

클래스 name 위치 설정

```
!pip install -r C:/test1/yolov3/requirements.txt
```

```
Requirement already satisfied: Cython in c:\users\untjr\anaconda3\lib\site-packages (from -r C:/test1/yolov3/requirements.txt (line 4)) (3.0.8)
Requirement already satisfied: matplotlib>=3.2.2 in c:\users\untjr\anaconda3\lib\site-packages (from -r C:/test1/yolov3/requirements.txt (line 5)) (3.
8.0)
Requirement already satisfied: numpy>=1.18.5 in c:\users\untjr\anaconda3\lib\site-packages (from -r C:/test1/yolov3/requirements.txt (line 6)) (1.26.4)
Requirement already satisfied: opencv-python>=4.1.2 in c:\users\untjr\anaconda3\lib\site-packages (from -r C:/test1/yolov3/requirements.txt (line 7))
(4.9.0.80)
Requirement already satisfied: pillow in c:\users\untjr\anaconda3\lib\site-packages (from -r C:/test1/yolov3/requirements.txt (line 8)) (10.2.0)
Requirement already satisfied: PyYAML>=5.3 in c:\users\untjr\anaconda3\lib\site-packages (from -r C:/test1/yolov3/requirements.txt (line 9)) (6.0.1)
Requirement already satisfied: scipy>=1.4.1 in c:\users\untjr\anaconda3\lib\site-packages (from -r C:/test1/yolov3/requirements.txt (line 10)) (1.11.4)
Requirement already satisfied: tensorboard>=2.2 in c:\users\untjr\anaconda3\lib\site-packages (from -r C:/test1/yolov3/requirements.txt (line 11)) (2.1
0.1)
Requirement already satisfied: torch>=1.6.0 in c:\users\untjr\anaconda3\lib\site-packages (from -r C:/test1/yolov3/requirements.txt (line 12)) (1.12.1)
Requirement already satisfied: torchvision>=0.7.0 in c:\users\untjr\anaconda3\lib\site-packages (from -r C:/test1/yolov3/requirements.txt (line 13))
(0.13.1)
Requirement already satisfied: tqdm>=4.41.0 in c:\users\untjr\anaconda3\lib\site-packages (from -r C:/test1/yolov3/requirements.txt (line 14)) (4.65.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\untjr\anaconda3\lib\site-packages (from matplotlib>=3.2.2->-r C:/test1/yolov3/requirements.
txt (line 5)) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\untjr\anaconda3\lib\site-packages (from matplotlib>=3.2.2->-r C:/test1/yolov3/requirements.txt
```

4단계-모델 구축(requirements)

- **Cython**: C 언어로 작성된 Python 확장 모듈을 쉽게 생성할 수 있게 해주는 프로그래밍 언어입니다.
- **matplotlib**: Python에서 데이터를 시각화할 수 있는 라이브러리입니다.
- **numpy**: 대규모 다차원 배열 및 행렬 연산에 최적화된 라이브러리입니다.
- **opencv-python**: 오픈소스 컴퓨터 비전 및 머신 러닝 소프트웨어 라이브러리입니다.
- **pillow**: Python 이미징 라이브러리(PIL)의 포크로, 이미지 파일 처리 기능을 제공합니다.
- **PyYAML**: YAML 파일 형식을 읽고 쓸 수 있는 Python 라이브러리입니다.
- **scipy**: 과학 계산을 위한 Python 라이브러리입니다.
- **tensorboard**: TensorFlow의 시각화 도구입니다.
- **torch**: PyTorch, 오픈소스 머신 러닝 라이브러리입니다.
- **torchvision**: PyTorch와 함께 이미지 및 비디오 데이터 작업을 위한 라이브러리 및 유틸리티입니다.
- **tqdm**: 루프 및 작업 진행 상황 표시를 위한 라이브러리입니다.
- **pycocotools**: COCO 데이터셋을 다루기 위한 도구 모음입니다.
- **packaging**: 버전 관리 및 패키지 정보 파싱을 위한 라이브러리입니다.
- **coremltools**: Apple의 Core ML 모델 포맷으로 변환하기 위한 도구입니다.
- **onnx**: 여러 프레임워크 간 모델 호환성을 제공하는 오픈 뉴럴 네트워크 교환 포맷입니다.
- **scikit-learn**: 데이터 마이닝 및 데이터 분석을 위한 라이브러리입니다.
- **thop**: 모델의 FLOPS를 계산하기 위한 도구입니다.

이 제조 데이터 의 특징은 파일화 되어있어서 간단하게 줄일 수있다

5단계-모델 훈련

```
!python train.py --epochs 15 --weights weights/last.pt --batch-size 3 --cfg yolov3-spp.cfg --data custom.data --nosave --device cpu
```

--epochs 15: 모델을 15 에폭(epoch) 동안 학습시킵니다. 에폭은 전체 데이터 세트를 한 번 학습하는 것을 의미합니다.

--weights weights/last.pt: 학습을 시작할 때 사용할 가중치 파일입니다. 'last.pt' 파일을 초기 가중치로 사용합니다.

--batch-size 3: 각 학습 배치에서 처리할 샘플의 수를 3으로 설정합니다. 배치 크기는 메모리 사용량과 학습 속도에 영향을 미칩니다.

--cfg yolov3-spp.cfg: 모델의 구조를 정의하는 구성 파일입니다. 'yolov3-spp.cfg'는 모델 구조에 대한 설정을 포함합니다.

--data custom.data: 학습에 사용될 데이터와 관련된 설정을 포함한 파일입니다. 데이터 경로, 클래스 수, 클래스 이름 등의 정보를 포함할 수 있습니다.

--nosave: 이 옵션을 사용하면 학습 도중 생성되는 중간 가중치 파일을 저장하지 않습니다. 주로 테스트나 디버깅 목적으로 사용됩니다.

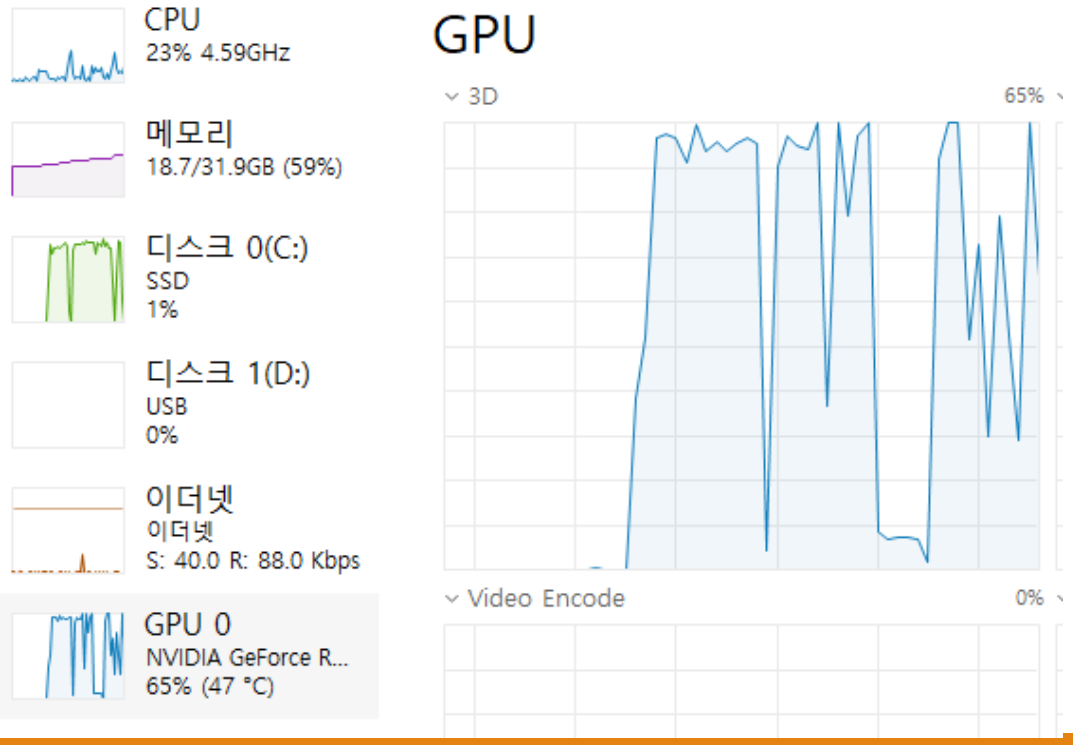
--device cpu: 학습을 CPU에서 수행하도록 설정합니다. GPU를 사용할 수 있는 경우, '--device gpu' 또는 '--device 0'과 같이 GPU ID를 지정하여 GPU에서 학습을 수행할 수 있습니다

5단계-모델 훈련(GPU, CPU 비교)

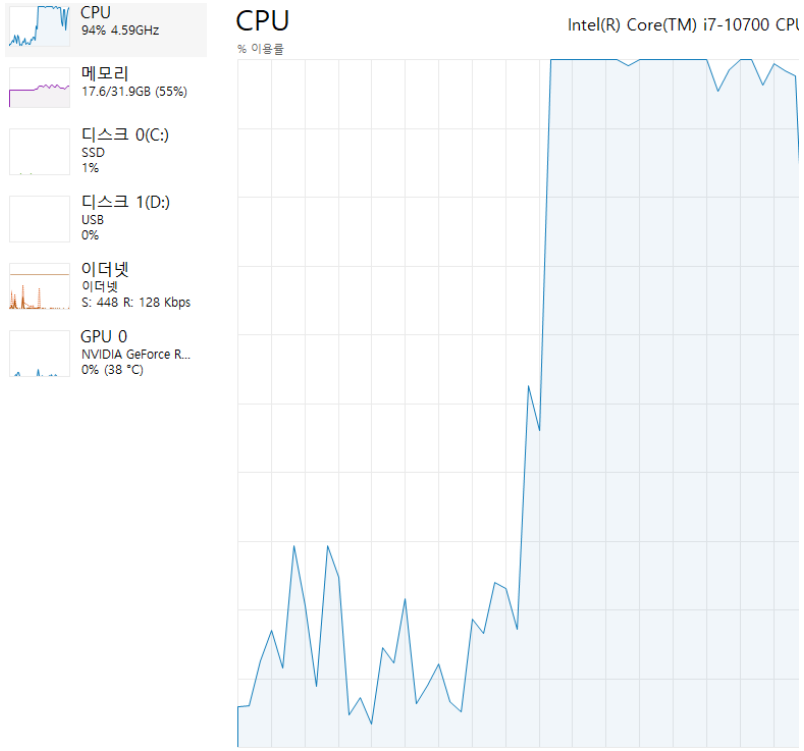
!python train.py --epochs 15 --weights weights/last.pt --batch-size 3 --cfg yolov3-spp.cfg --data custom.data --nosave --device cpu

---

GPU로 훈련 : 빠름
다른작업도 여유있게 가능



CPU로 훈련 : 느리고pc 죽는소리남;;
다른작업시 랙이 걸림



5단계-모델 훈련

```
873/876      0G      1.08    0.0202         0       1.1         6       512:  75%|#######5  | 3/4 [00:16<00:04,  4.19s/it]
873/876      0G      1.08    0.0202         0       1.1         6       512: 100%|##########| 4/4 [00:16<00:00,  4.00s/it]
873/876      0G      1.08    0.0202         0       1.1         6       512: 100%|##########| 4/4 [00:17<00:00,  4.28s/it]

           Class    Images   Targets         P         R   mAP@0.5        F1:   0%|          | 0/1 [00:00<?, ?it/s]
           Class    Images   Targets         P         R   mAP@0.5        F1: 100%|##########| 1/1 [00:03<00:00,  3.71s/it]
           Class    Images   Targets         P         R   mAP@0.5        F1: 100%|##########| 1/1 [00:03<00:00,  3.99s/it]

0%|          | 0/4 [00:00<?, ?it/s]
874/876      0G     0.631     0.011         0     0.642         2       512:   0%|          | 0/4 [00:05<?, ?it/s]
874/876      0G     0.631     0.011         0     0.642         2       512:  25%|##5       | 1/4 [00:05<00:16,  5.50s/it]
874/876      0G     0.843    0.0118         0     0.854         6       512:  25%|##5       | 1/4 [00:09<00:16,  5.50s/it]
874/876      0G     0.843    0.0118         0     0.854         6       512:  50%|#####     | 2/4 [00:09<00:09,  4.52s/it]
874/876      0G     0.799    0.0136         0     0.812         5       512:  50%|#####     | 2/4 [00:13<00:09,  4.52s/it]
874/876      0G     0.799    0.0136         0     0.812         5       512:  75%|#######5  | 3/4 [00:13<00:04,  4.28s/it]
874/876      0G      1.05     0.014         0      1.06         4       512:  75%|#######5  | 3/4 [00:17<00:04,  4.28s/it]
874/876      0G      1.05     0.014         0      1.06         4       512: 100%|##########| 4/4 [00:17<00:00,  4.10s/it]
874/876      0G      1.05     0.014         0      1.06         4       512: 100%|##########| 4/4 [00:17<00:00,  4.36s/it]

           Class    Images   Targets         P         R   mAP@0.5        F1:   0%|          | 0/1 [00:00<?, ?it/s]
           Class    Images   Targets         P         R   mAP@0.5        F1: 100%|##########| 1/1 [00:03<00:00,  3.76s/it]
           Class    Images   Targets         P         R   mAP@0.5        F1: 100%|##########| 1/1 [00:04<00:00,  4.03s/it]

0%|          | 0/4 [00:00<?, ?it/s]
875/876      0G      1.01    0.0295         0      1.04        10       512:   0%|          | 0/4 [00:05<?, ?it/s]
875/876      0G      1.01    0.0295         0      1.04        10       512:  25%|##5       | 1/4 [00:05<00:16,  5.63s/it]
875/876      0G      1.02    0.0245         0      1.05         5       512:  25%|##5       | 1/4 [00:09<00:16,  5.63s/it]
875/876      0G      1.02    0.0245         0      1.05         5       512:  50%|#####     | 2/4 [00:09<00:09,  4.60s/it]
875/876      0G      1.05    0.0233         0      1.07         5       512:  50%|#####     | 2/4 [00:13<00:09,  4.60s/it]
875/876      0G      1.05    0.0233         0      1.07         5       512:  75%|#######5  | 3/4 [00:13<00:04,  4.23s/it]
875/876      0G      1.06    0.0224         0      1.09         6       512:  75%|#######5  | 3/4 [00:17<00:04,  4.23s/it]
875/876      0G      1.06    0.0224         0      1.09         6       512: 100%|##########| 4/4 [00:17<00:00,  4.15s/it]
875/876      0G      1.06    0.0224         0      1.09         6       512: 100%|##########| 4/4 [00:17<00:00,  4.40s/it]

           Class    Images   Targets         P         R   mAP@0.5        F1:   0%|          | 0/1 [00:00<?, ?it/s]
           Class    Images   Targets         P         R   mAP@0.5        F1: 100%|##########| 1/1 [00:03<00:00,  3.79s/it]
           Class    Images   Targets         P         R   mAP@0.5        F1: 100%|##########| 1/1 [00:04<00:00,  4.09s/it]

0%|          | 0/4 [00:00<?, ?it/s]
876/876      0G     0.897    0.0134         0     0.911         3       512:   0%|          | 0/4 [00:05<?, ?it/s]
876/876      0G     0.897    0.0134         0     0.911         3       512:  25%|##5       | 1/4 [00:05<00:16,  5.53s/it]
876/876      0G     0.737    0.0149         0     0.752         4       512:  25%|##5       | 1/4 [00:09<00:16,  5.53s/it]
876/876      0G     0.737    0.0149         0     0.752         4       512:  50%|#####     | 2/4 [00:09<00:09,  4.53s/it]
876/876      0G     0.884    0.0131         0     0.897         2       512:  50%|#####     | 2/4 [00:13<00:09,  4.53s/it]
876/876      0G     0.884    0.0131         0     0.897         2       512:  75%|#######5  | 3/4 [00:13<00:04,  4.18s/it]
876/876      0G         1    0.0145         0      1.02         4       320:  75%|#######5  | 3/4 [00:15<00:04,  4.18s/it]
876/876      0G         1    0.0145         0      1.02         4       320: 100%|##########| 4/4 [00:15<00:00,  3.31s/it]
876/876      0G         1    0.0145         0      1.02         4       320: 100%|##########| 4/4 [00:15<00:00,  3.84s/it]

           Class    Images   Targets         P         R   mAP@0.5        F1:   0%|          | 0/1 [00:00<?, ?it/s]
           Class    Images   Targets         P         R   mAP@0.5        F1: 100%|##########| 1/1 [00:03<00:00,  3.55s/it]
```

6단계-결과 분석 및 해석

```
!python detect.py --weights weights/last.pt --source images --cfg yolov3-spp.cfg --names classes.names --output result
```

```
Model Summary: 225 layers, 6.25733e+07 parameters, 6.25733e+07 gradients
image 1/15 images\001_20200622_203312(6).jpg: 416x512 Done. (0.404s)
image 2/15 images\001_20200623_003516(8).jpg: 416x512 Done. (0.395s)
image 3/15 images\001_20200901_003429(1).jpg: 416x512 Done. (0.386s)
image 4/15 images\001_20200922_163329(0).jpg: 416x512 Done. (0.386s)
image 5/15 images\001_20200922_163332(2).jpg: 416x512 Done. (0.396s)
image 6/15 images\002_20200714_043029(1).jpg: 512x512 1 defects, Done. (0.559s)
image 7/15 images\002_20200715_042416(9).jpg: 512x512 Done. (0.500s)
image 8/15 images\002_20200831_083545(0).jpg: 512x512 Done. (0.479s)
image 9/15 images\002_20200831_123134(5).jpg: 512x512 Done. (0.466s)
image 10/15 images\002_20200908_043137(8).jpg: 512x512 Done. (0.458s)
image 11/15 images\002_20200915_003459(4).jpg: 512x512 Done. (0.460s)
image 12/15 images\002_20200917_043110(7).jpg: 512x512 Done. (0.464s)
image 13/15 images\002_20200917_202952(6).jpg: 512x512 Done. (0.477s)
image 14/15 images\002_20200917_203001(5).jpg: 512x512 1 defects, Done. (0.570s)
image 15/15 images\002_20200922_083747(7).jpg: 512x512 1 defects, Done. (0.500s)
Results saved to C:\test1\yolov3\result
Done. (7.072s)
```

제품의 결함이 발견됨

6단계-결과 분석 및 해석

```
!python test.py --cfg yolov3-spp.cfg --batch-size 3 --data custom.data --weights weights/last.pt
```

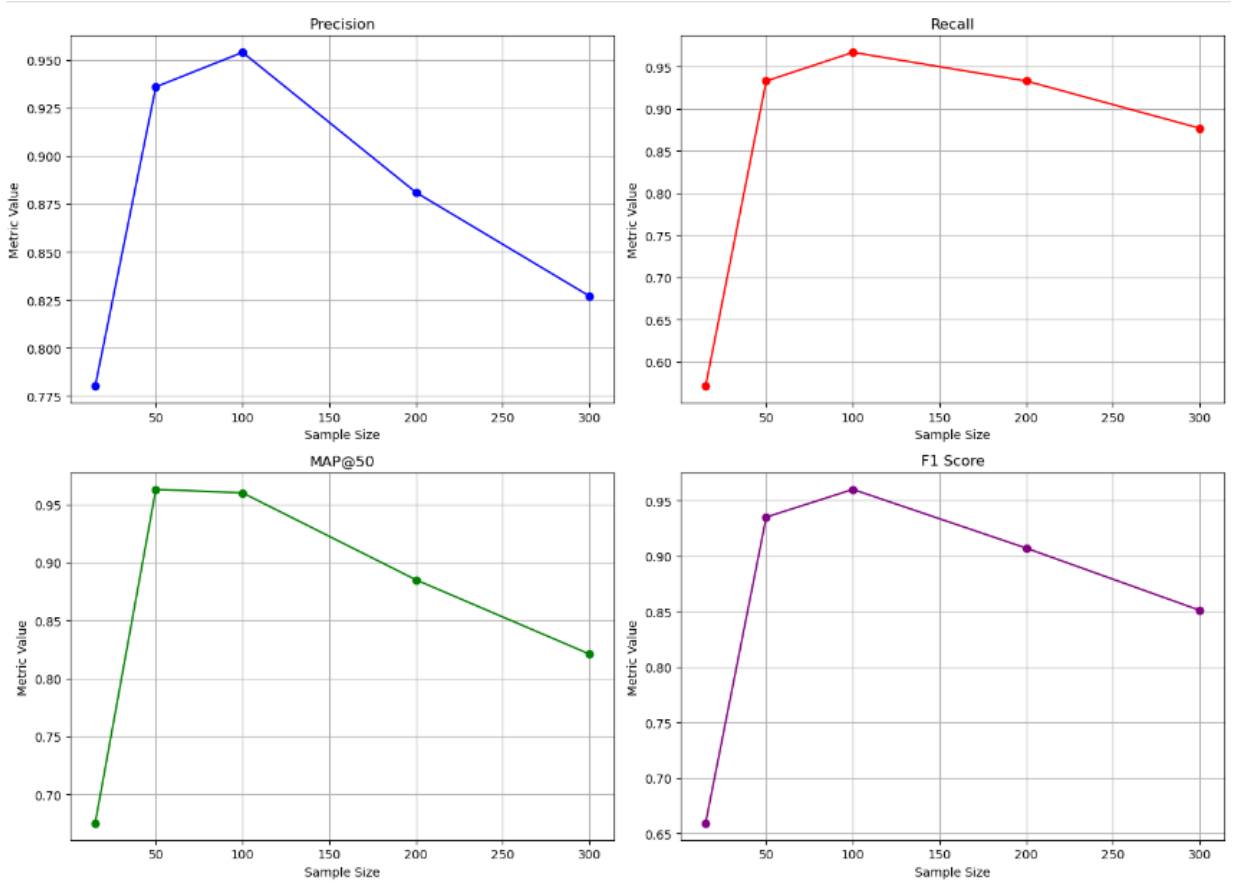
```
WARNING: smart bias initialization failure.
WARNING: smart bias initialization failure.
WARNING: smart bias initialization failure.
Model Summary: 225 layers, 6.25733e+07 parameters, 6.25733e+07 gradients
Fusing layers...
Model Summary: 152 layers, 6.25465e+07 parameters, 6.25465e+07 gradients
                 all           3           5       0.967           1       0.995       0.983
Speed: 441.8/1.0/442.8 ms inference/NMS/total per 512x512 image at batch-size 3
```

P (정밀도)　R (재현율)　mAP@0.5 (평균 정밀도)　F1

결과-발표

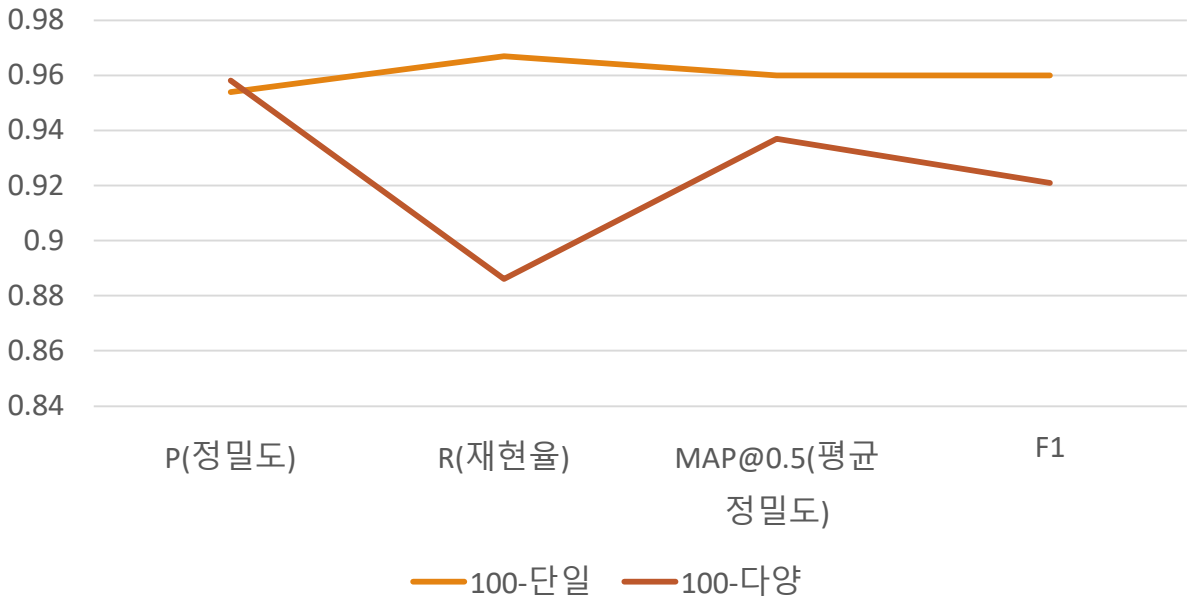
학습 이미지 수량	P(정밀도)	R(재현율)	MAP@0.5 (평균 정밀도)	F1
15	0.78	0.571	0.675	0.659
50	0.936	0.933	0.963	0.935
100	0.954	0.967	0.96	0.96
200	0.881	0.933	0.885	0.907
300	0.881	0.933	0.885	0.907

결과-발표



새로운 문제
1. 과적합?
2. 데이터 전처리 문제?

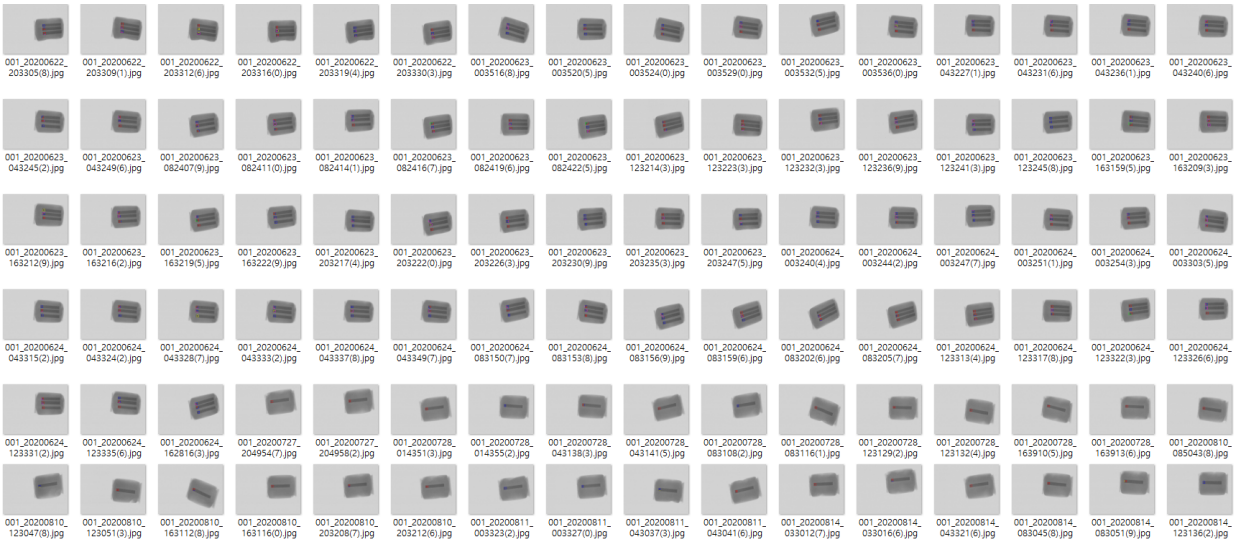
결과-발표



문제 해결 : 데이터 전처리(데이터 다양성)

학습 이미지 수량	P(정밀도)	R(재현율)	MAP@0.5 (평균 정밀도)	F1
100-a(단일)	0.954	0.967	0.96	0.96
100-b(다양성)	0.958	0.886	0.937	0.921

데이터 다양성이란?



다른 라인(다양성 높음)

같은 라인(다양성 낮음)

학습 데이터 이미지 수(100개) 는 같으나 사진의 각도와 불량품의 종류의 다양성이 다름

결론

Yolov3로 성능을 올리기 위해서는 일반적으로 학습데이터 개수를 많이 하는게 일반적이나,

그보다 중요한건 학습데이터 선별에서 데이터의 다양성을 확보해서 학습시키는것이 중요하다.

이 프로젝트를 하면서....

데이터 다양성을 위해 다른 라인에서 섞어서 학습을 시켜야 하는데 그러지 못했다.(시간부족)

이러한 경우(다양하게 섞는경우) 학습데이터 양을 늘릴 수 록 성능이 개선 될것이라 예측됨

감사합니다.