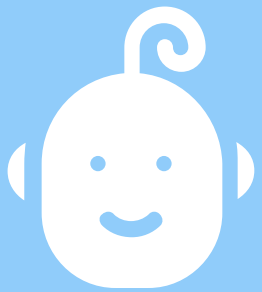
BOUNDING BOXER

딥러닝 기반
반도체 불량 검출 AI 시스템

이현우

김형중

윤예은



이주성

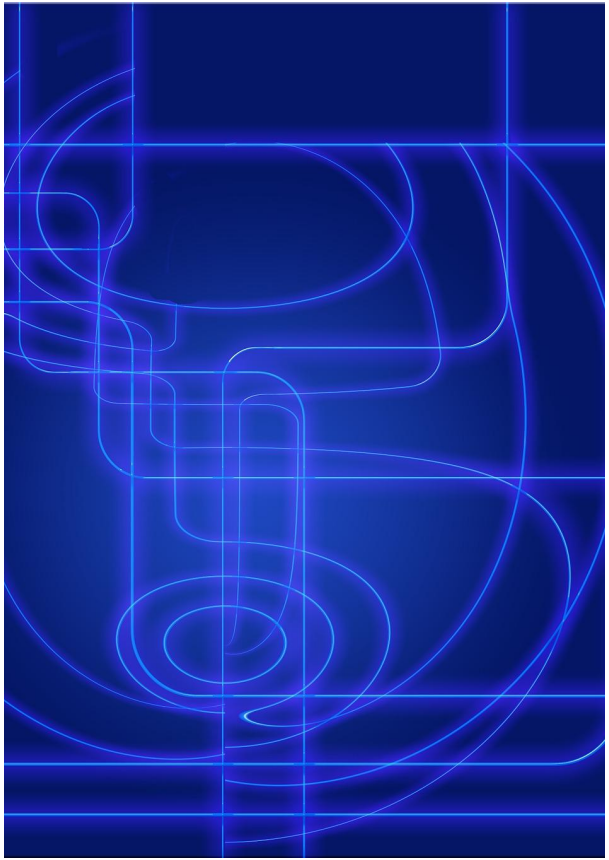
임지현

Bounding boxer

Contents

1. 프로젝트 개요

Deep learning-based
semiconductor
EDS AI system

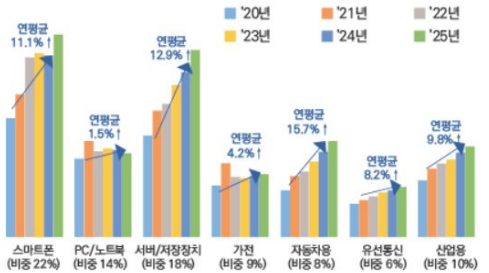


반도체 산업 동향

한국의 우수한 기술력과 경쟁력

지속적인 **연구개발(R&D)** 투자

인공지능, 5G, 자율주행 등
기술 발전에 따라 **반도체의 수요 증가 예상**



반도체 8대 공정

EDS 공정이란?

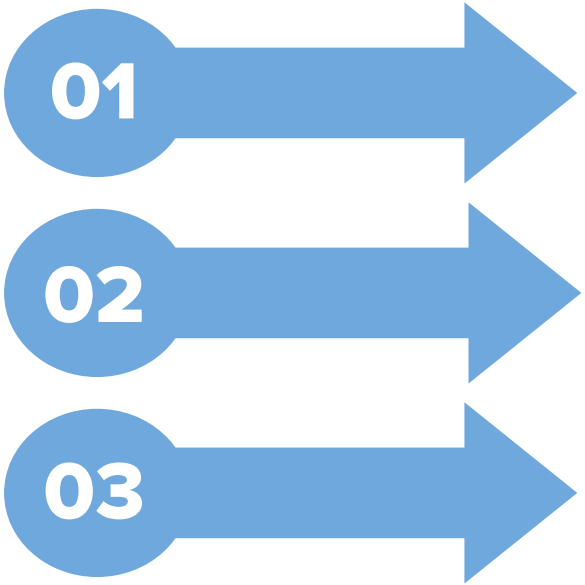
웨이퍼가 포함하고 있는 반도체 칩의 불량을 선별하는 공정
수율을 높이기 위해 필요한 공정

기존 반도체 산업의 문제점

수작업
작업자의 검사력 차이
2차 결함 발생

제조 및 제조과정
공정, 설비상의 문제
공정의 step수 증가

높은 비용
레이저 장비 비용
인건비



프로젝트의 목적

컴퓨터 비전을 이용
객체탐지 AI 모델 구축
-> EDS 공정 문제 해결

수율▲ 비용▼
솔루션 제시

01 프로젝트 개요

Skill Set

개발 환경

colab

Visual Studio Code

공유 문서

GitHub

Google Drive

Pretrained Model

YOLO

Detectron2

Web 프레임워크

django

Flask

Bootstrap

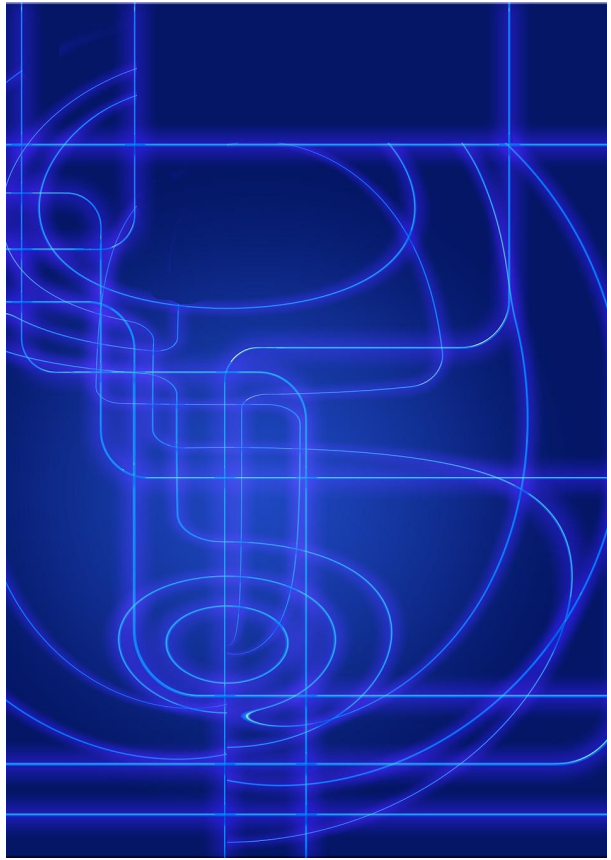
Annotation Tool

roboflow

GIMP

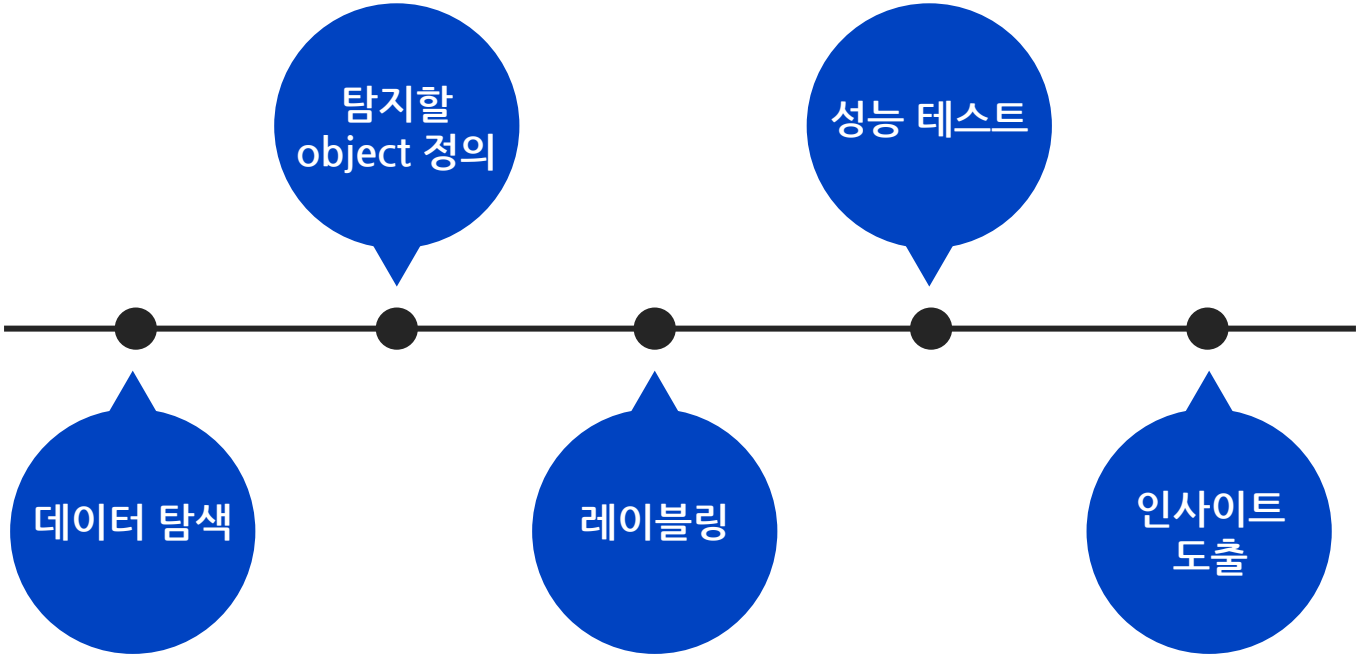
OpenCV

2. 데이터 탐색 & 전처리



Deep learning-based
semiconductor
EDS AI system

데이터 처리과정



02 데이터 탐색&전처리

원본 데이터 탐색 및 평가

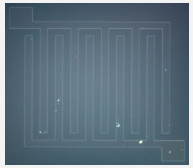
data

train

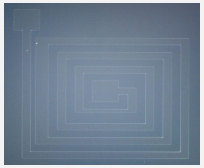
valid

test

images



pattern 1



pattern 2

labels

```
3 0.4095 0.886 0.0375 0.036
3 0.6365 0.1375 0.0365 0.035
2 0.7235 0.1425 0.005 0.005
3 0.5615 0.3515 0.0365 0.035
3 0.559 0.3875 0.0365 0.0335
3 0.55 0.4225 0.035 0.0315
3 0.546 0.4585 0.036 0.035
3 0.5065 0.5195 0.0365 0.034
3 0.5055 0.557 0.036 0.034
3 0.4785 0.628 0.0365 0.036
3 0.465 0.665 0.0365 0.0365
3 0.464 0.7025 0.0365 0.0365
3 0.45 0.7405 0.0365 0.0375
2 0.185 0.7775 0.0035 0.0035
3 0.439 0.781 0.041 0.0415
3 0.427 0.827 0.0375 0.0375
```

행 : 결함 개수 / 열
[Class, X, Y, W, H]

data.yaml

```
train: ../train/images
val: ../valid/images
test: ../test/images

nc: 4
names: ['0', '1', '2', '3']
```

4개의 class

YOLOv5 TEST

Class	mAP50	mAP50-95:
all	0.474	0.286
0	0.901	0.687
1	0.249	0.0746
2	0.272	0.0952

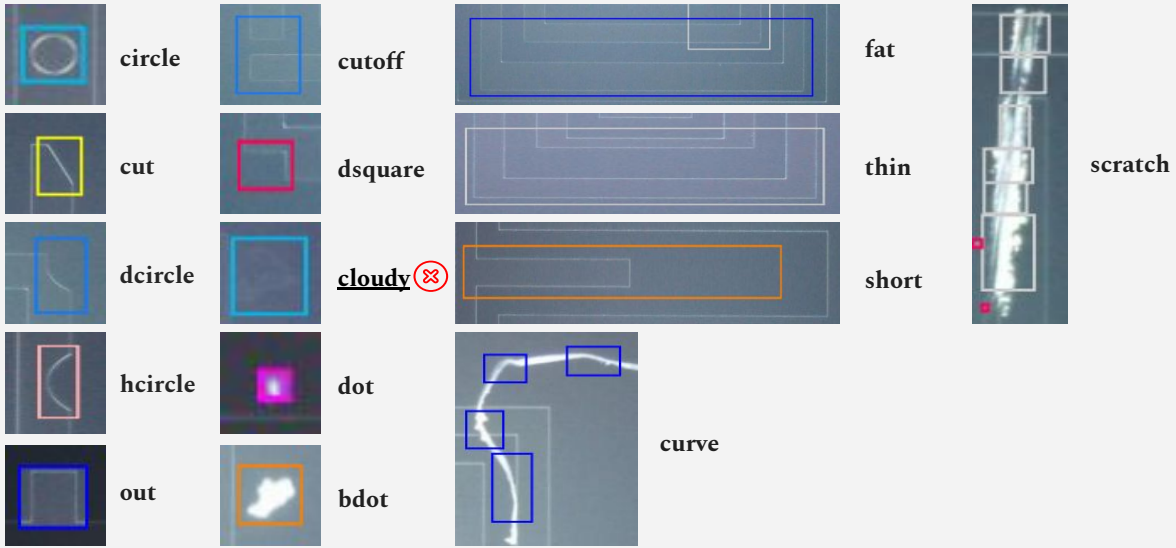
Result

- mAP 지표 저조
- ‘straight’ class 검출 불가

>> **새로운 Labeling 필요**

02 데이터 탐색&전처리

도메인 탐색을 통한 결함(object) 정의



Class 재설정

원본데이터

0 : bdot
1 : curve
2 : dot
3 : straight

↓

15개 class

↓

'cloudy' 삭제

↓

최종 14개 class 확정

02 데이터 탐색&전처리

다양한 데이터셋 생성 후 성능 평가

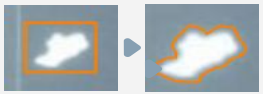
Dataset 1

· num_classes : 15

Dataset 2

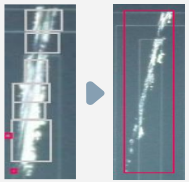
· num_classes : 14
· bdot 폴리곤 처리
· curve, scratch 바운딩박스 변경

bdot



curve

scratch



*성능 TEST에는 속도가 빠른 YOLOv5 모델 사용

성능TEST 결과

Dataset 1 (mAP 0.753)

· dot, bdot 개수가 가장 많지만 낮은 AP
· curve, scratch 가장 낮은 AP
· cloudy 개수가 적고 AP가 낮아 mAP 하락
  >> 일부 dot으로 변경, 나머지 삭제
  >> **mAP 0.808**

Dataset 2 (mAP 0.185)

· dot 0.223
· bdot 0.96
· curve 0.0117
· scratch 0.224

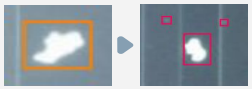
02 데이터 탐색&전처리

다양한 데이터셋 생성 후 성능 평가

Dataset 3

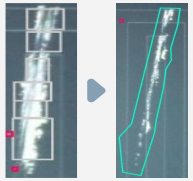
· num_classes : 14
· bdot 일부 dot 처리
· curve 폴리곤 처리

bdot



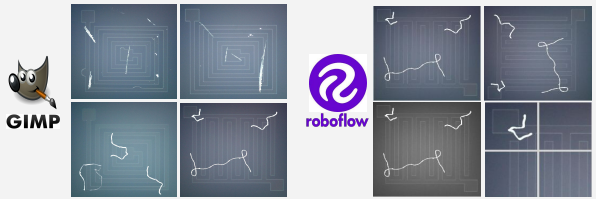
curve

scratch



Dataset 4

· 김프에서 curve, scratch 합성하여
각 20장씩 총 40장 증식
· roboflow 활용하여
Grayscale, Rotation, Tiling 이미지 증식

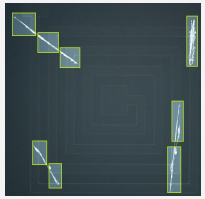


· bdot, dot 미세조정

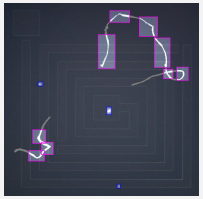
bdot

dot

scratch



curve



*성능 TEST에는 속도가 빠른 YOLOv5 모델 사용

성능TEST 결과

Dataset 3 (mAP 0.74)

· curve 0.0276
· dot 0.511
· scratch 0.521

Dataset 4 (mAP 0.896)

· curve 0.641
· dot 0.633
· bdot 0.652
· scratch 0.635

02 데이터 탐색&전처리

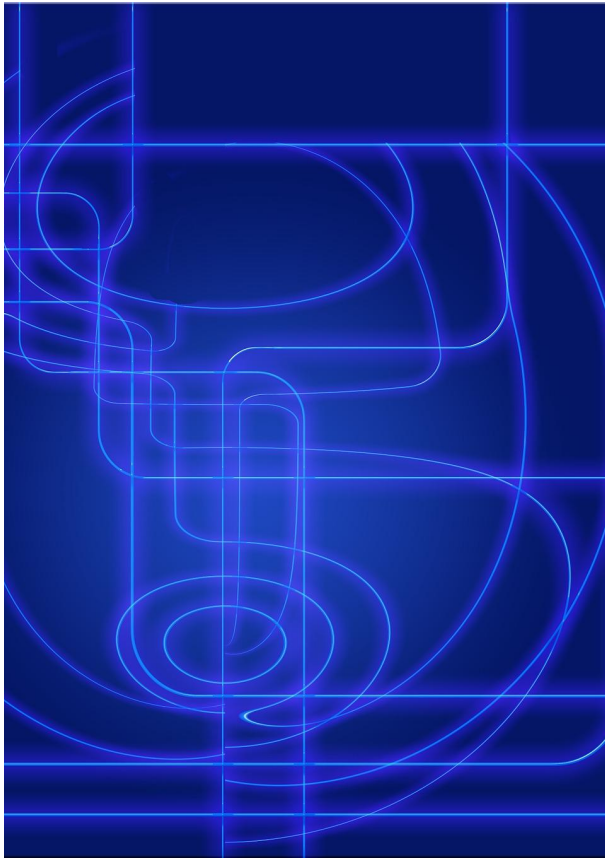
다양한 데이터셋 생성

insight

- **성능향상** 원인분석
  - 분명한 특징
  - 일정한 크기, 모양, 위치
  - 여유있는 bounding box
- **성능하락** 원인분석
  - 적합하지 않은 폴리곤 처리
  - 여러명이 함께 **labeling**

3. 모델링 및 분석

Deep learning-based
semiconductor
EDS AI system



03 모델링 및 분석

사전훈련된 모델 비교

	mAP	Training Time / 1 epoch total
YOLOv5s	0.911	12s/1 epoch 1.998h
YOLOv6l 6finetuned	0.917	55s/1 epoch 6.989
YOLOv7s	0.875	1m7s/1 epoch 6.349h
YOLOv8s	0.868	42s/1 epoch 2.428h
DETECTRON	0.886	3s/1 epoch 54m

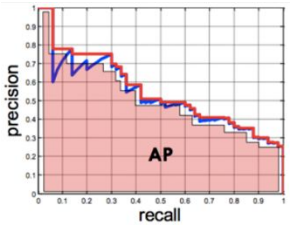
성능 비교

mAP와 소요시간을 비교한 결과
YOLOv5가 가장 우수하다!

왜 mAP를 지표로 사용할까요?

mAP 지표

mAP = AP의 합 / 클래스 개수



정밀도와 재현율은 **반비례 관계**를 갖기 때문에 이 **두 값을 모두 고려하여 정확도를 평가**하는 것이 좋다. 그래서 나온 것이 precision-recall 곡선 및 AP 개념이다. 정밀도와 재현율 모두 우수하다면 면적(AP)이 넓어 정확도가 높다고 평가한다.

03 모델링 및 분석

사전훈련된 모델 비교

YOLOv4

- YOLOv3 + CSP Darknet53 (Backbone) + SPP + PAN + BoF + BoS
- YOLOv5는 Darknet이 아닌 PyTorch를 이용하여 환경구성 및 구현이 쉽다.

YOLOv6

- Yolov5에 비해 더 깊어지고 head 부분의 scale이 하나 더 생김
- 더 다양한 size의 object를 잘 detection 한다

YOLOv7

- inference 시에 추가적인 비용없이 네트워크 성능 향상을 위한 방법이다.
- 전체적인 성능은 높은 편이지만 YOLOv5에 비해 낮다.

YOLOv8

- 모델 활용이 쉽고 전체적인 성능이 높은 편이지만 YOLOv5에 비해 낮다.

Detectron

- Facebook에서 개발한 object detection, segmentations PyTorch 프레임워크
- Detectron은 데이터 이미지 수가 적은 경우 유리
- 기존 데이터의 양이 충분하고 증식이 쉬운 이미지 이기에 Detectron보다는 YOLOv5가 적합하다.

YOLOv5 선정 이유

1. YOLOv5의 Backbone을 **PyTorch**로 구현하여 **접근성이 좋다**

2. YOLOv5는 YOLOv6, YOLOv7에 비해 **속도가 빠르다**

3. Detectron은 속도는 빠르나 리소스를 많이 사용하여 YOLOv5에 비해 **범용성** 면에서 아쉬운 점이 있다.

03 모델링 및 분석

선정된 YOLOv5 모델의 구조

Backbone

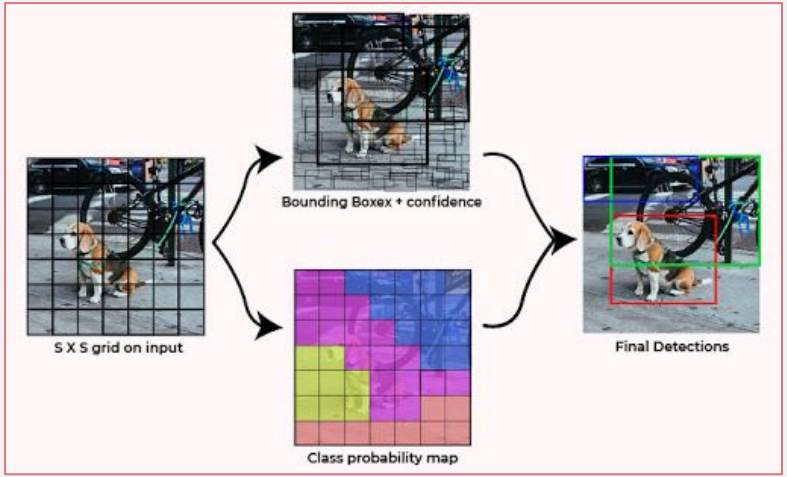
- 다양한 피처맵 추출
- Convolution, Pooling layers
- YOLOv5 - BottleneckCSP

Neck

- 피처맵 융합
- PAN을 사용하여 성능 향상

Head

- 피처맵 바탕으로 위치 탐지
- 9개의 anchor box 사용
- 최종 bounding box 생성



〈 YOLOv5 작동원리 그림 예시〉

YOLOv5 작동원리

어떤 원리로 작동되나요?

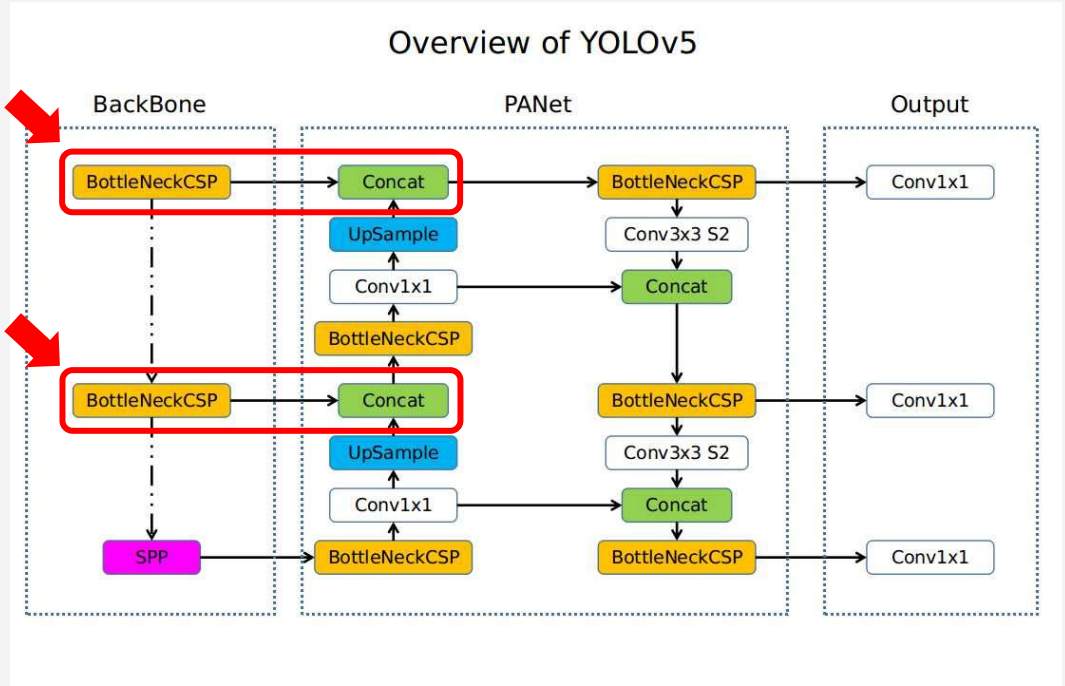
우선 **Backbone**에서 이미지를 받아 **위치 정보가 담긴 Tensor로 변환**합니다.
그리고 여러 번의 텐서연산을 거쳐서 **다양한 해상도의 Feature map을 추출**합니다.

Neck에서는 **Feature map을 받아 융합**합니다.

마지막으로 **Head**에서 **Anchor box를 이용하여 Bounding box를 구성**하고 **객체 탐지를 실시**합니다.

03 모델링 및 분석

왜 YOLOv5가 가장 적합한가?



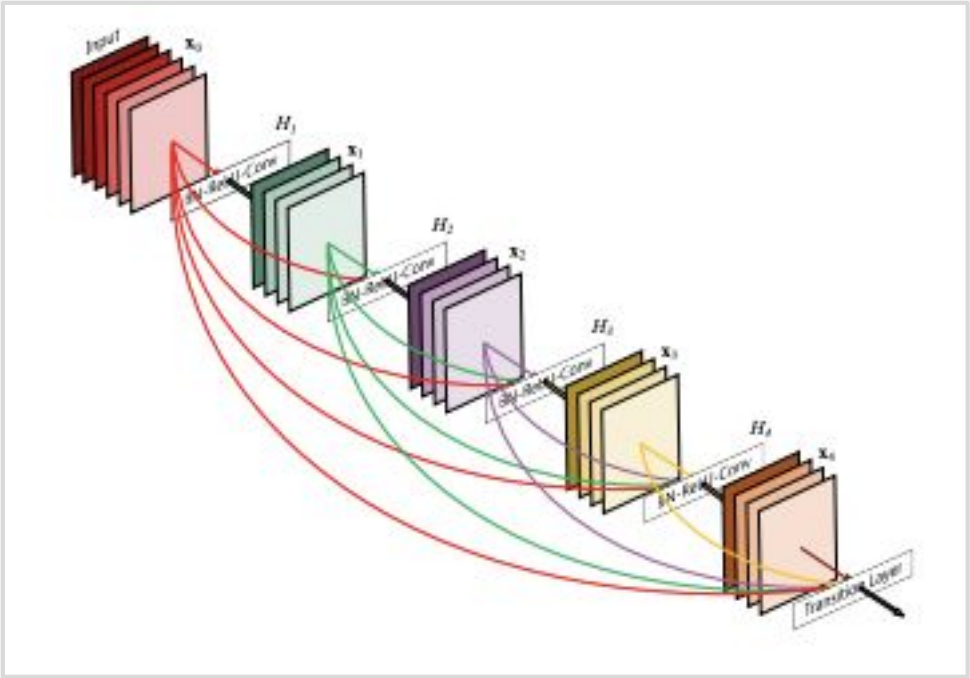
1 속도

BottleneckCSP 기술

데이터를 나누어 따로 처리하고 다시 합치는 방식으로 처리량을 줄여 속도를 높인다.

03 모델링 및 분석

왜 YOLOv5가 가장 적합한가?



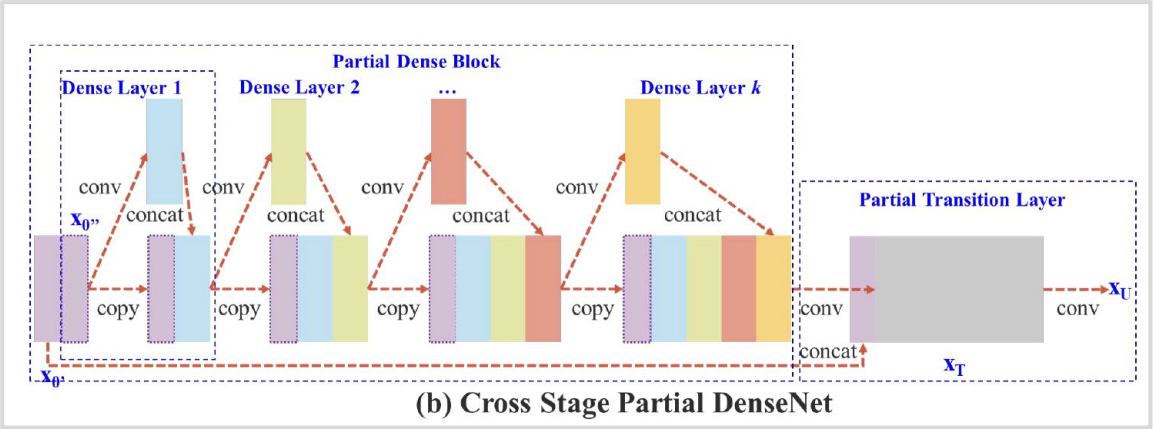
2 성능

53개의 깊은 층

DarkNet53기반의 Backbone으로 **53개의 Convolution 층**을 가진 **깊은층의 모델**로 더욱 정확하고 복잡한 객체 검출이 가능하고 GPU 사용이 가능

03 모델링 및 분석

왜 YOLOv5가 가장 적합한가?



(b) Cross Stage Partial DenseNet

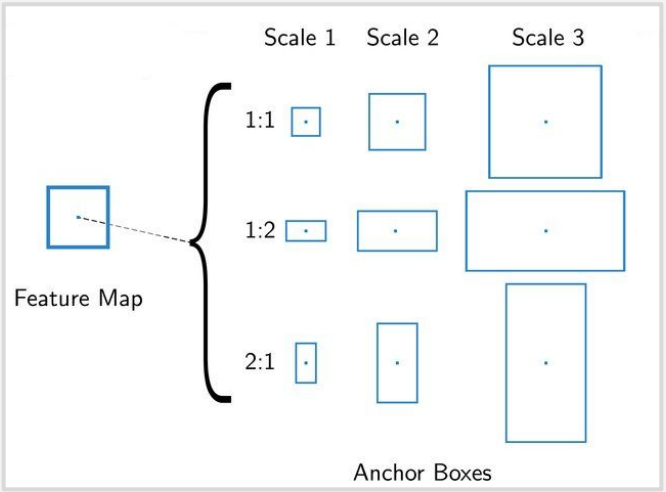
3 성능

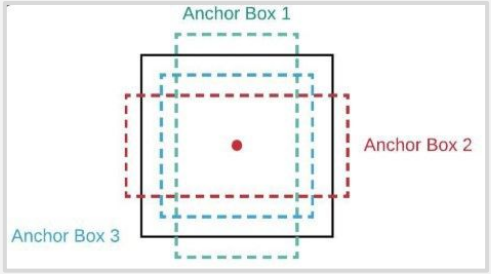
BottleneckCSP 모듈 사용

BottleneckCSP 모듈의 Bottleneck의 역할로 **잔차연결을 통해 하위층의 데이터 손실 방지**하여 성능을 향상

03 모델링 및 분석

왜 YOLOv5가 가장 적합한가?





4 성능

9개의 Anchor box

스케일의 크기에 따라 각기 다른 Bounding Box 예측으로 **더 세밀한 예측이 가능**

03 모델링 및 분석

YOLOv5 s, m, l 성능비교

	mAP	training time / 1 epoch total
YOLOv5s	0.911	12s/1 epoch 1.998h
YOLOv5m	0.902	21s/ 1epoch 2.7h
YOLOv5l	0.915	1m7s/1 epoch 6.349h

YOLOv5 Large

YOLOv5 Large 모델의 장점

크고 복잡한 데이터 세트 처리 시 s, m에 비해 높은 mAP 점수!

그러나 성능 처리 시간이 길어지고 계산 요구사항이 높아짐

03 모델링 및 분석

옵티마이저 비교

	mAP	training time / 1 epoch total
SGD (default)	0.915	34s/1 epoch 5.08h
Adam	0.903	35s/1 epoch 5.226h

SGD vs Adam

작은 데이터셋에 적합한 SGD

- SGD가 비교적 작은 데이터 세트에, Adam은 큰 데이터 세트에 효과적
- SGD가 모델 가중치를 더 자주 업데이트 하고 비교적 **작은 데이터 세트의 특정 특성에 더 잘 적응**

SGD의 간단한 최적화 알고리즘

SGD가 Adam보다 간단한 최적화 알고리즘으로 조정해야 할 하이퍼파라미터가 적음

03 모델링 및 분석

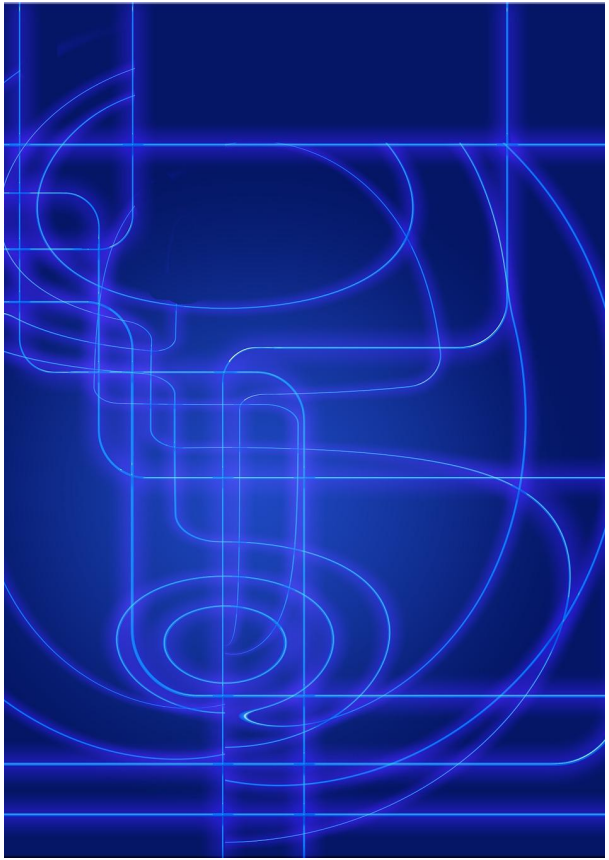
모델 분석을 통해 얻은 인사이트

?

?

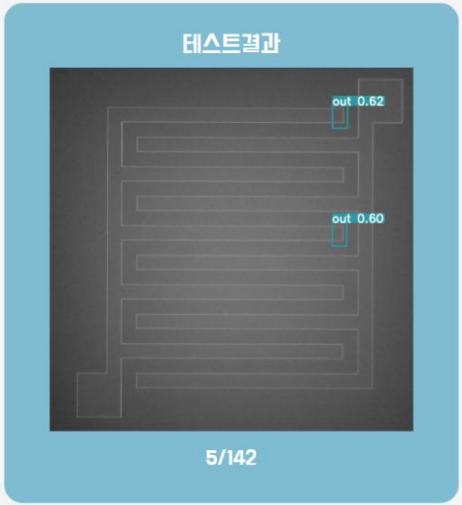
4. Solution DEMO

Deep learning-based
semiconductor
EDS AI system



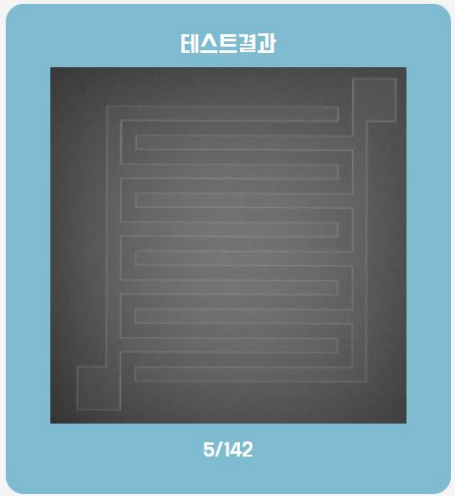
04 Solution DEMO

옵티마이저 비교



정상회로에서 결함이 탐지됨

새로 훈련



결함을 탐지하지 않음

detect 오류 1

정상회로 이미지를 rotation 했을 뿐인데 **결함이 있는것으로 detect** 되는 문제 발견

원인분석

- OpenCV를 활용해 copy, flip, rotation 처리하여 test dataset 생성
- **그러나 train dataset에 rotation, flip 처리한 이미지가 없음**

해결

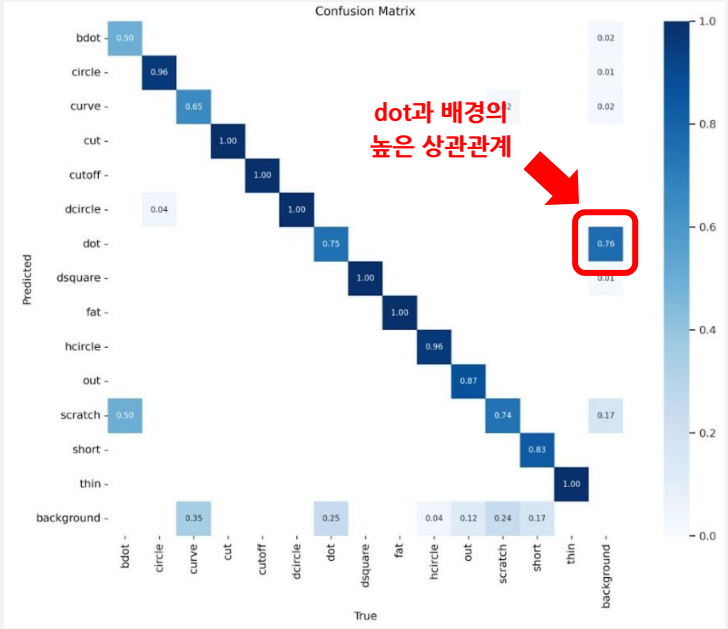
rotation, flip한 이미지를 새로 학습시켜서 문제 해결

04 Solution DEMO
옵티마이저 비교

미세한 노이즈를 결함으로 탐지

conf = 0.55

결함을 탐지하지 않음



detect 오류 2

정상회로 이미지에서 **미세한 노이즈를 dot으로 detect** 하는 문제 발견

원인분석

Confusion Matrix 확인 결과 **dot과 배경의 상관관계가 높아서** detect 됨

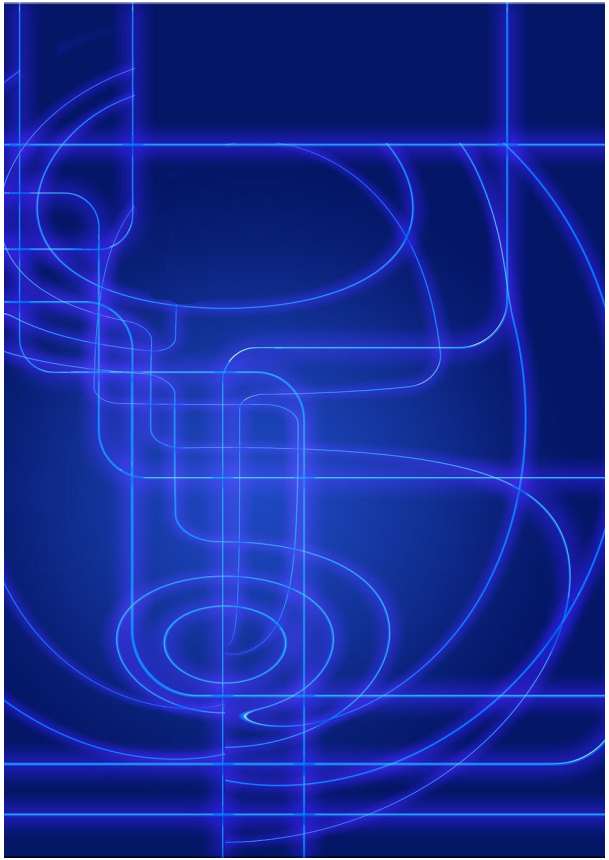
해결

배경과 dot이 구분될수있게 model **confidence를** 0.45에서 **0.55로** 높혀서 해결

04 Solution DEMO

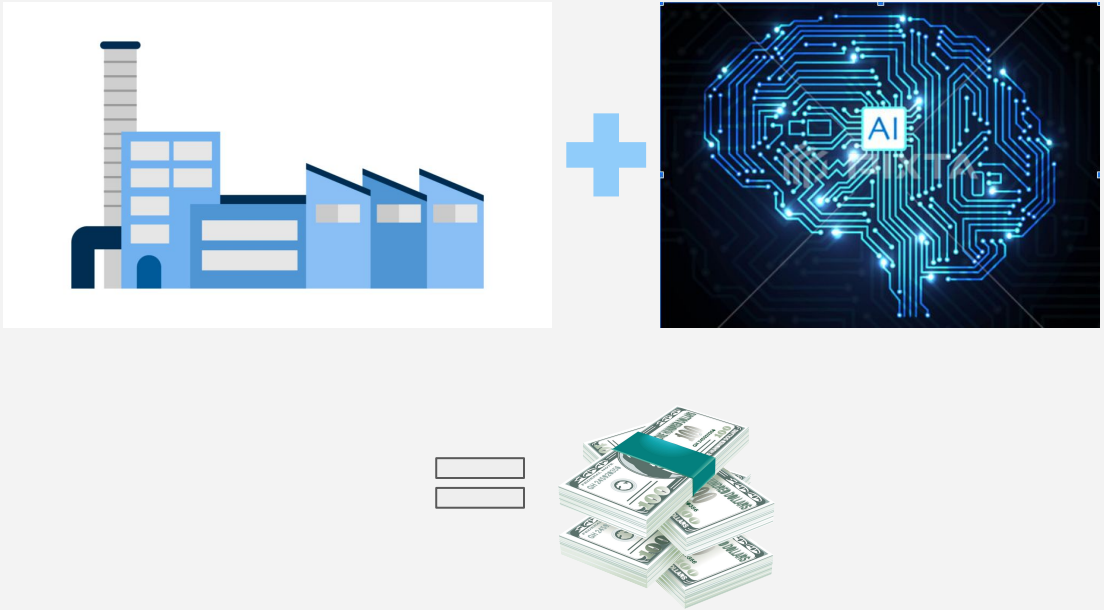
5. 활용방안 및 기대효과

Deep learning-based
semiconductor
EDS AI system



05 활용방안 및 기대효과

옵티마이저 비교



활용방안

AI 알고리즘을 통해

· **대량의 데이터**를 **빠르게** 처리, 분석 가능
· **24시간 제조** 운영 가능
· 수작업으로 인해 발생하는 2차 손실 감소
· **제품 신뢰도 증가**

>> 비용 절감, 생산성 향상, 불량품 감소

기대효과

하루 $20,000 손실을 50% 감소

제조 공장이 하루에 $10,000 가치의 1,000개의 웨이퍼를 처리하면 2%의 수율 손실 발생 추정 [하루 $20,000의 손실]

· AI를 이용하면 **수율 손실의 50%▼**
· 잠재적 비용 **약 300만~3000만 달러** 절약 가능

BOUNDING BOXER

Thanks!

Does **anyone** have any questions?