

반도체 불량 AI 자동 검출을 위한 딥러닝 프로젝트

Presentation for Deep Learning Project

Data Universe 박주경, 김성호, 김용훈, 윤예은, 이승주

목차

01 데이터 목차에 데이터 소개, ANNOTATION 및 증식

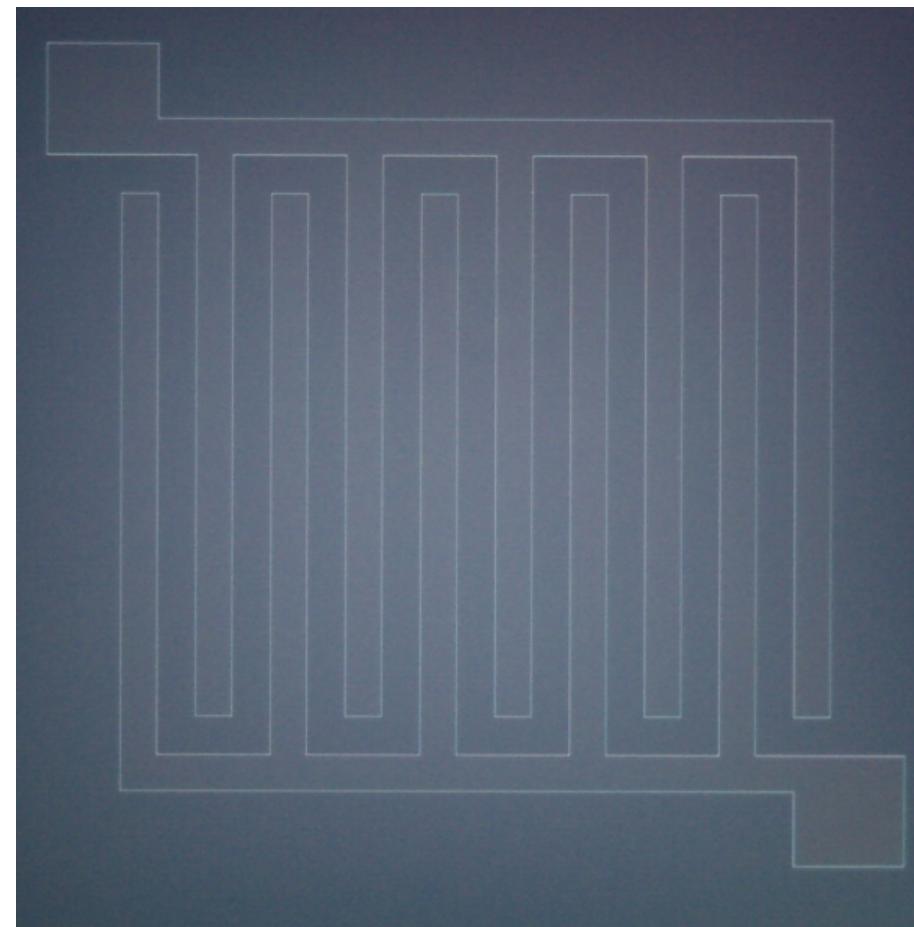
02 모델링 모델의 특징 및 장단점

03 중간평가

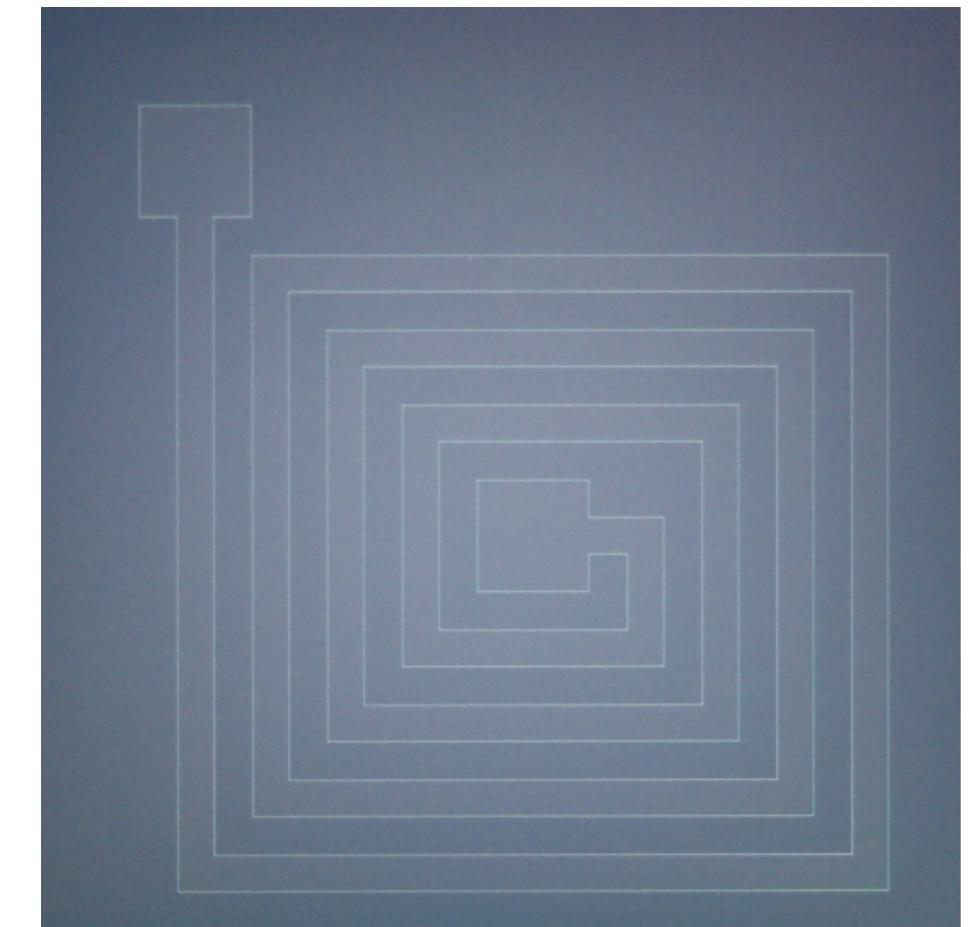
04 사용기술

01 데이터 소개 원본 데이터 총 337개

패턴	총 개수	Defect type
패턴 1	136개	9종류
패턴 2	201개	9종류



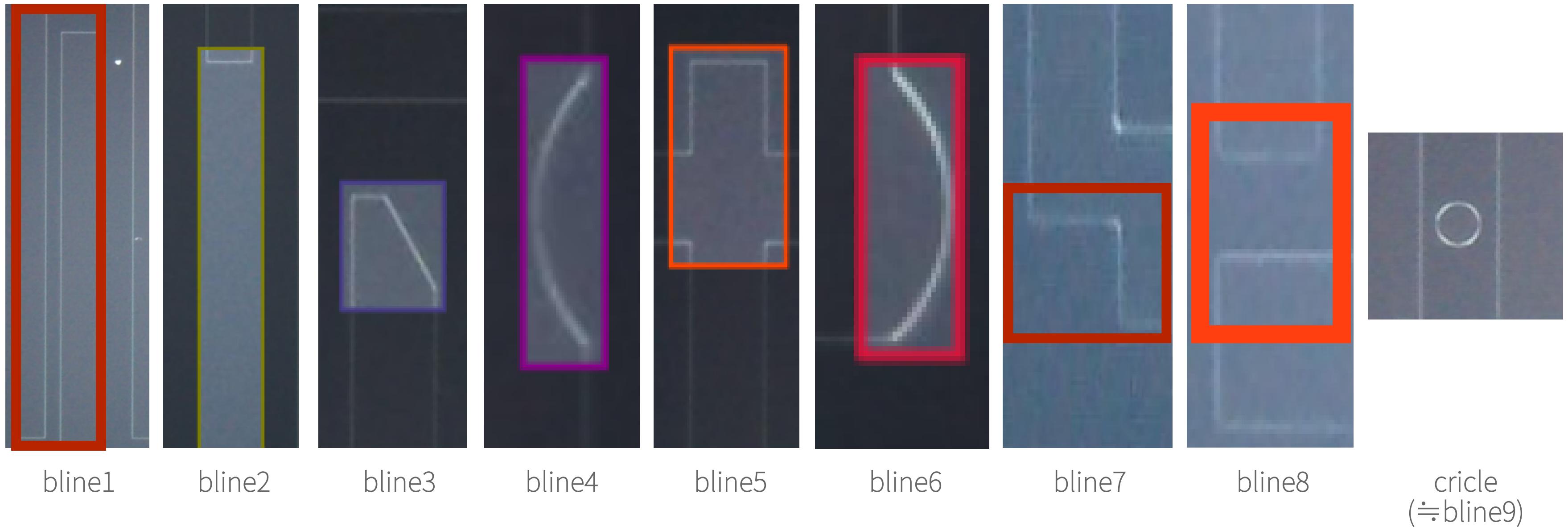
정상 pattern 1



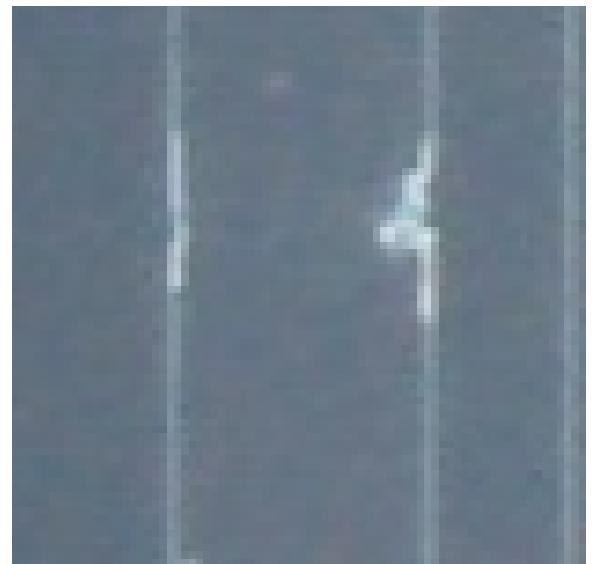
정상 pattern 2

2가지의 패턴 존재 / 9개의 패턴 불량 존재 -> 점, 스크래치, 얼룩 등의 불량이 겹쳐 있음

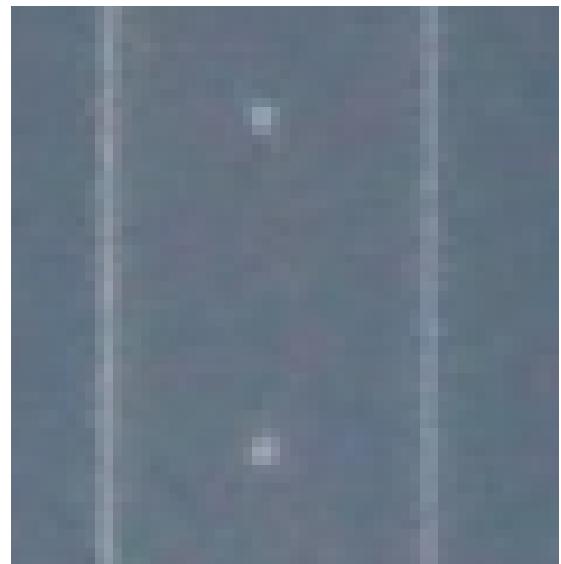
01 ANNOTATION 결함 패턴 유형 14개



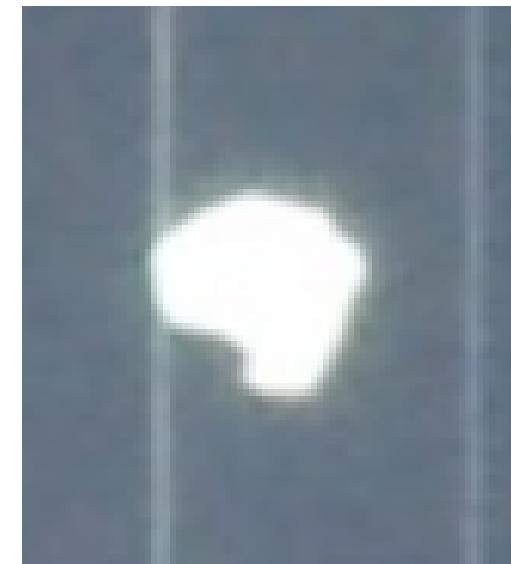
01 ANNOTATION 결함 패턴 유형 14개



break
line break



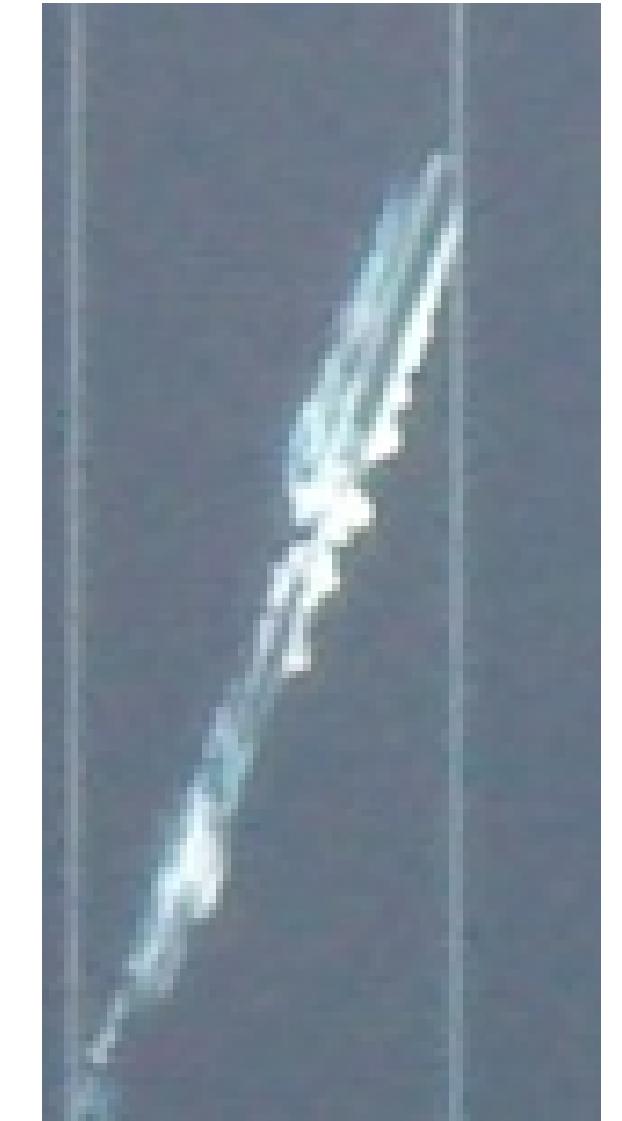
sdot
작은 점



bdot
큰 점



stain
얼룩



scratch
스크래치

01 데이터 증식

배분	증식 전 (337)	증식 후 (1016)
Training Set	97	776
Validation Set	136	59
Testing Set	104	181

1. Rotation → $97 \times 3 = 291$

rotation_range
90, 180, 270



2. Filp → $97 \times 4 = 388$

vertical_flip

horizontal_flip



Train Set $97 + 291 + 388 = 776$ *총 8배

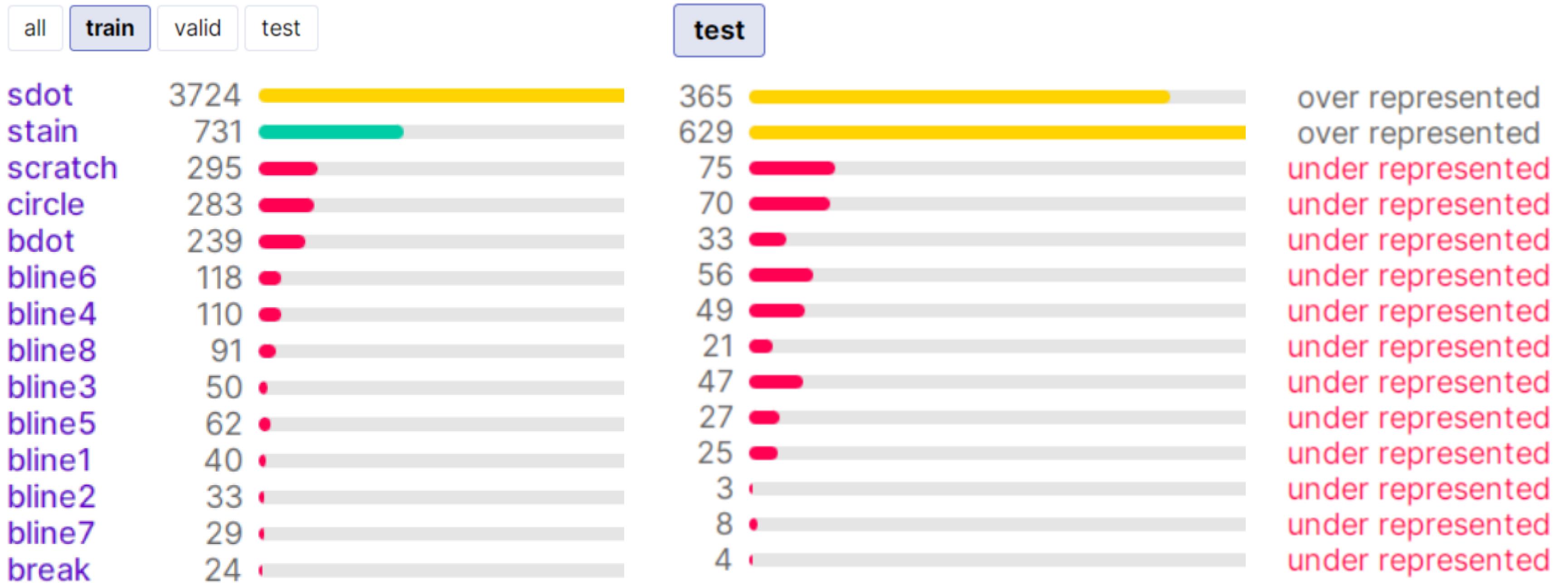
01 ANNOTATION 결함 패턴 유형 14개

Class Balance



01 ANNOTATION 결함 패턴 유형 14개

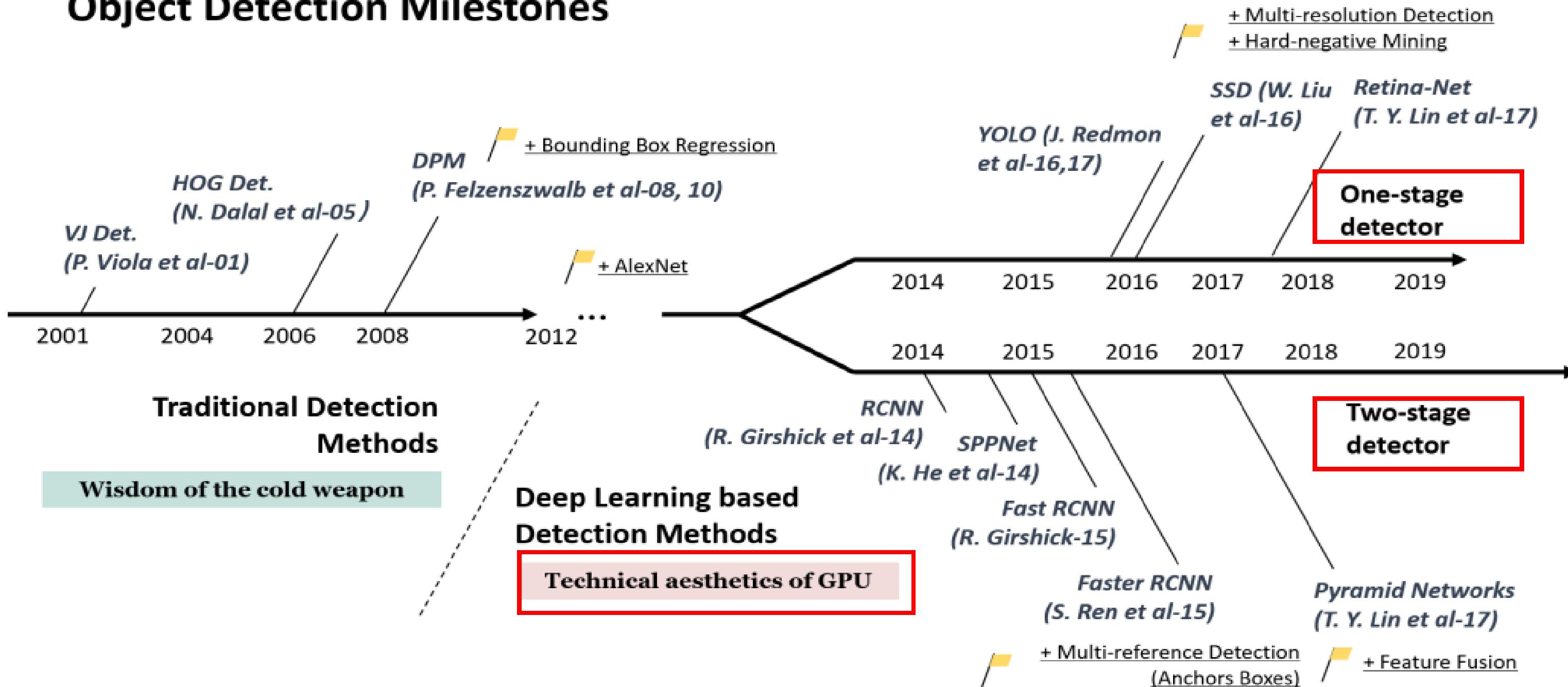
Class Balance



02 모델링

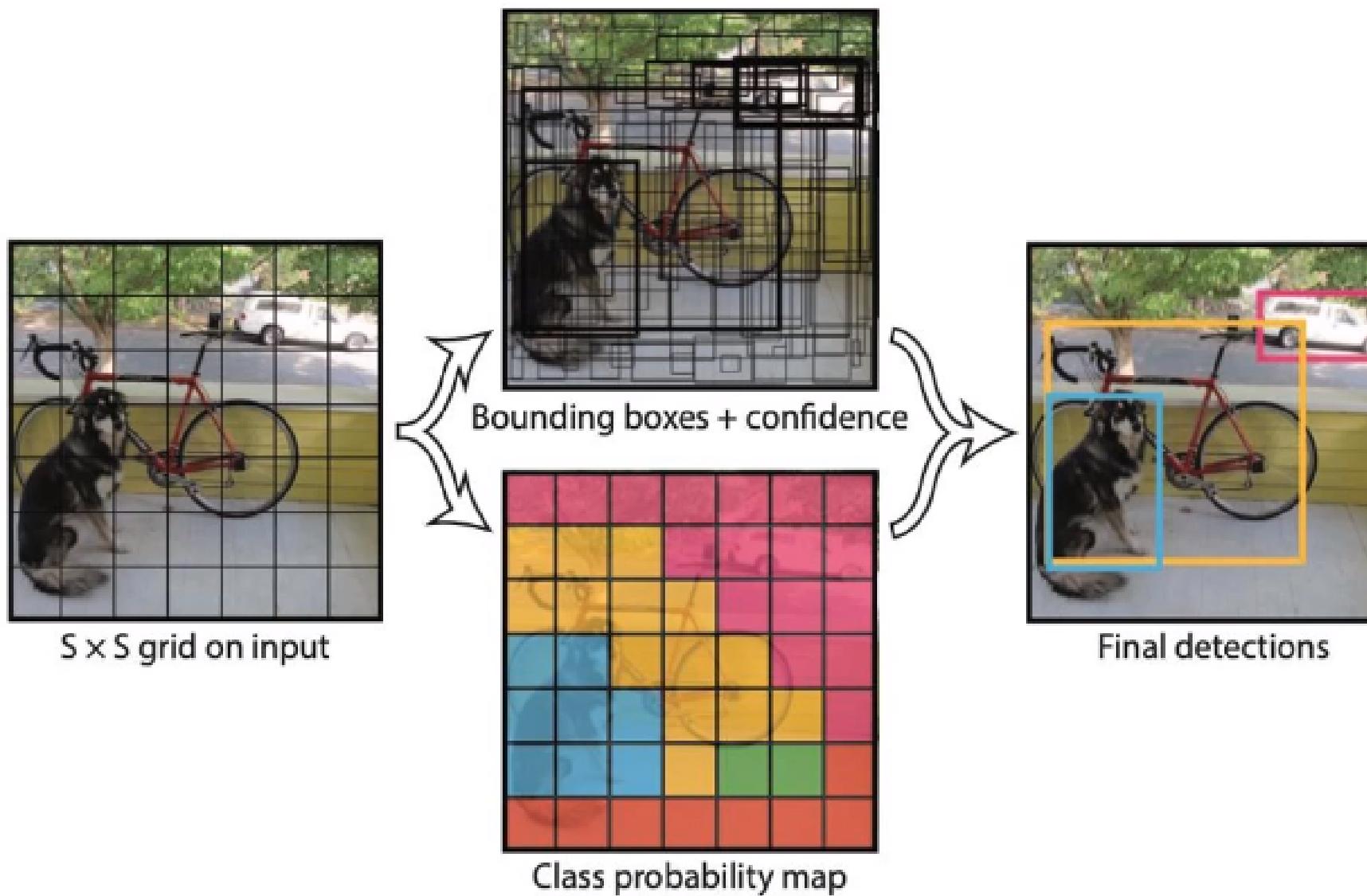
1-stage detector vs 2-stage detector

Object Detection Milestones



02 모델링 YOLO 객체탐지 과정

물체 인식(Object Detection)을 수행하기 위해 고안된 심층 신경망 테두리상자(Bounding Box)조정과 분류(Classification)를 동일 신경망 구조를 통해 동시에 실행하는 통합인식(Unified Detection)을 구현



입력이미지를 $S \times S$ 개의 그리드로 분할

- 그리드 영역에서 물체가 있을 영역으로 생각되는 해당 B개의 Bounding Box를 예측
(중심점 x,y 너비w 높이h)

- 해당 Bounding Box에 물체가 있을 확률
Confidence(C) 계산

- 각각의 그리드마다 해당 Bounding Box의 물체가 특정클래스(특정물체)일 확률 계산

출처- <https://arxiv.org/abs/1506.026>

02 모델링 프로젝트에서 다룰 YOLO 버전별 특징

1. YOLOv5

- 2020년 6월 발표
- 전버전 대비 객체 검출 정확도 10% 향상, 더 빠른 속도, 더 작은 모델
- PyTorch로 implementation
- CSPNet 기반의 backbone을 설계하여 사용(다크넷)
- 모델 크기인 s m l x 별로 속도와 정확도가 트레이드 오프 관계임

2. YOLOv5u

- YOLOv8의 anchor-free와 objectness-free split head 통합

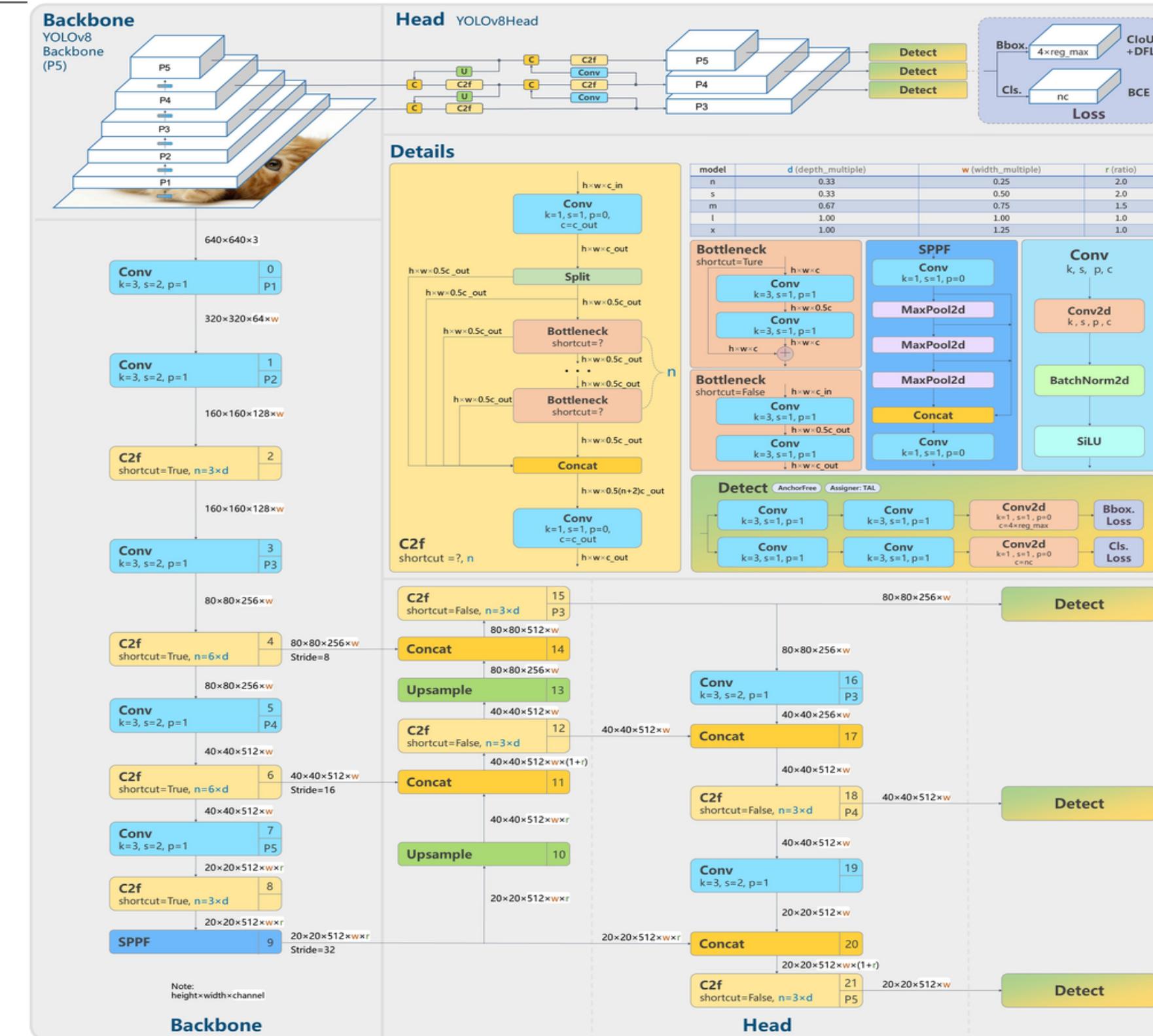
02 모델링 프로젝트에서 다룰 YOLO 버전별 특징

3. YOLOv8

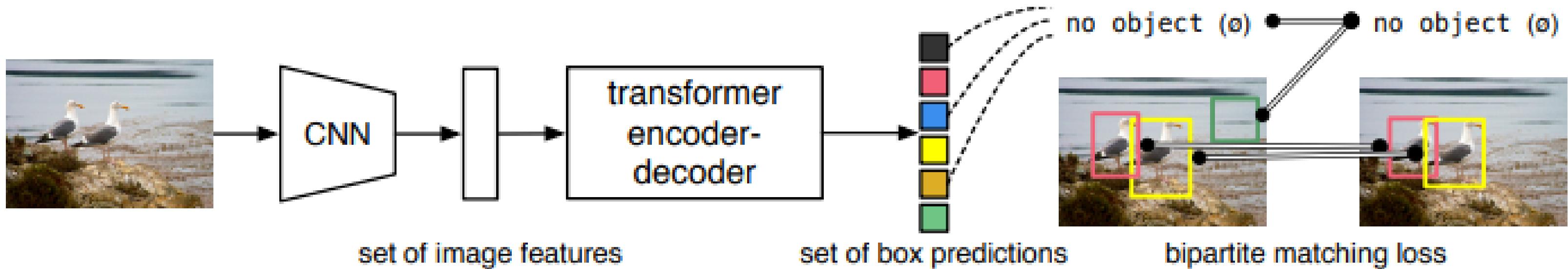
- 2023년 1월 발표
- 인스턴스 세그멘테이션 (Instance Segmentation) 새롭게 지원
- 사용자 친화적인 API 및 파이썬 명령 프롬프트 지원
- 속도와 정확성 향상
- 이전 버전보다 용량이 커짐
- 새로운 고급 백본 아키텍처 사용
- 새로운 손실함수 사용
- New Anchor-Free head

02 모델링

프로젝트에서 다룰 YOLO 버전별 특징



02 모델링 YOLO

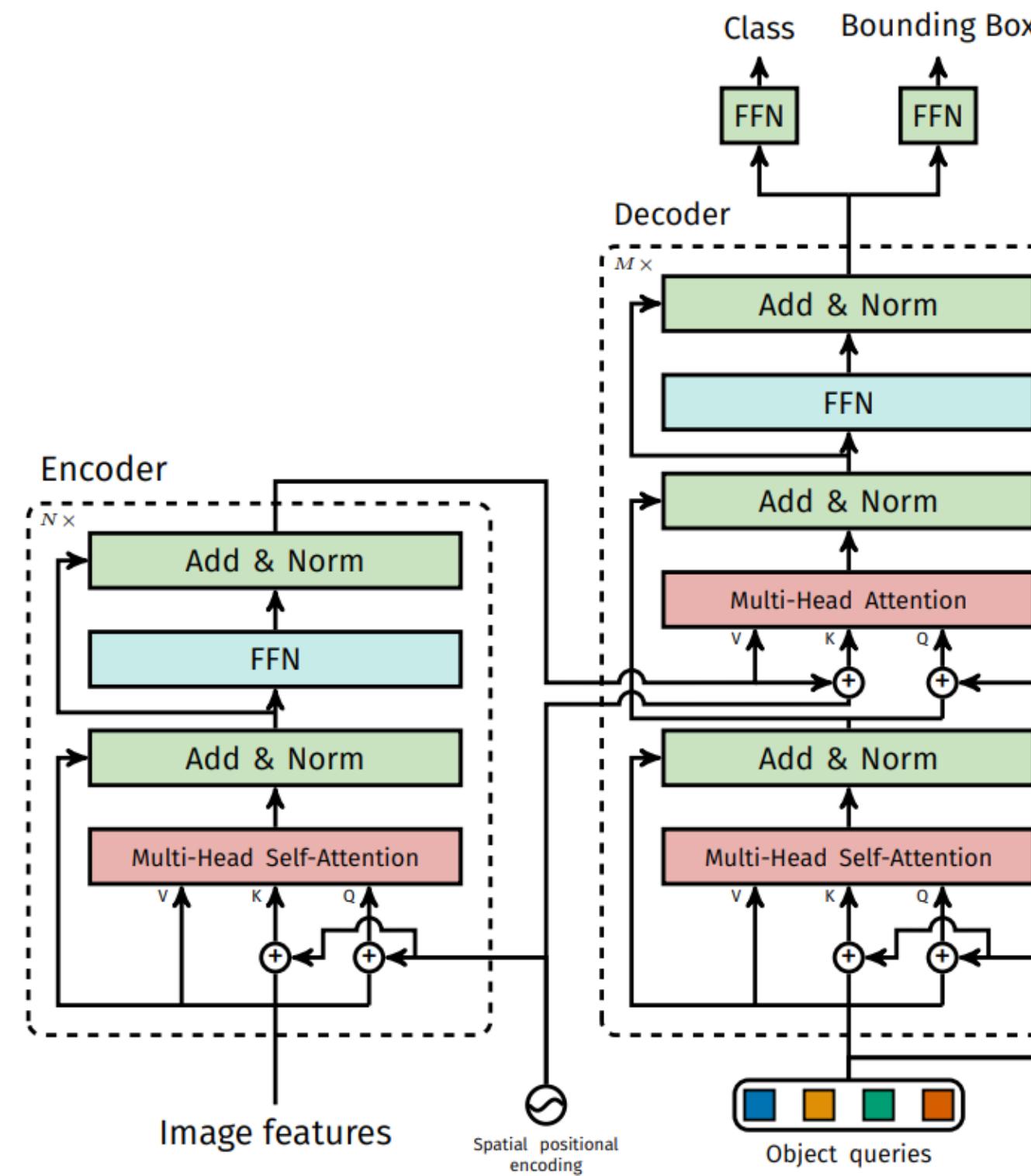


4. DE:TR

- End-to End Object Detection with Transformers
- 유명한 컴퓨터 비전 학회 ECCV에서 2020년 8월에 논문으로 발표된 모델
- 특징
 - 1) 이분 매칭 손실 함수
 - 2) 트랜스포머

02 모델링

DE:TR Architerture



03 중간평가

YOLOv5와 YOLOv8 비교

Model	mAP50	Time	Model	mAP50	Time
YOLOv5s	0.786	42분	YOLOv8n	0.788	23분
YOLOv5m	0.824	45분	YOLOv8s	0.814	26분
YOLOv5l	0.837	49분	YOLOv8m	0.828	43분
YOLOv5x	0.853	85분	YOLOv8l	추후 학습 예정	
			YOLOv8x	추후 학습 예정	

- 작은 모델의 경우 v8이 빠름, 큰 모델의 경우 v5가 빠름
- 모델의 크기가 커질수록 성능이 향상되는 경향 보임

03 중간평가

Valid Set class별 balancing 전 평가수치 vs balancing 후 평가수치

class balance	Model	mAP50
조정 전	YOLOv8m	0.754
조정 후	YOLOv8m	0.828

DE:TR

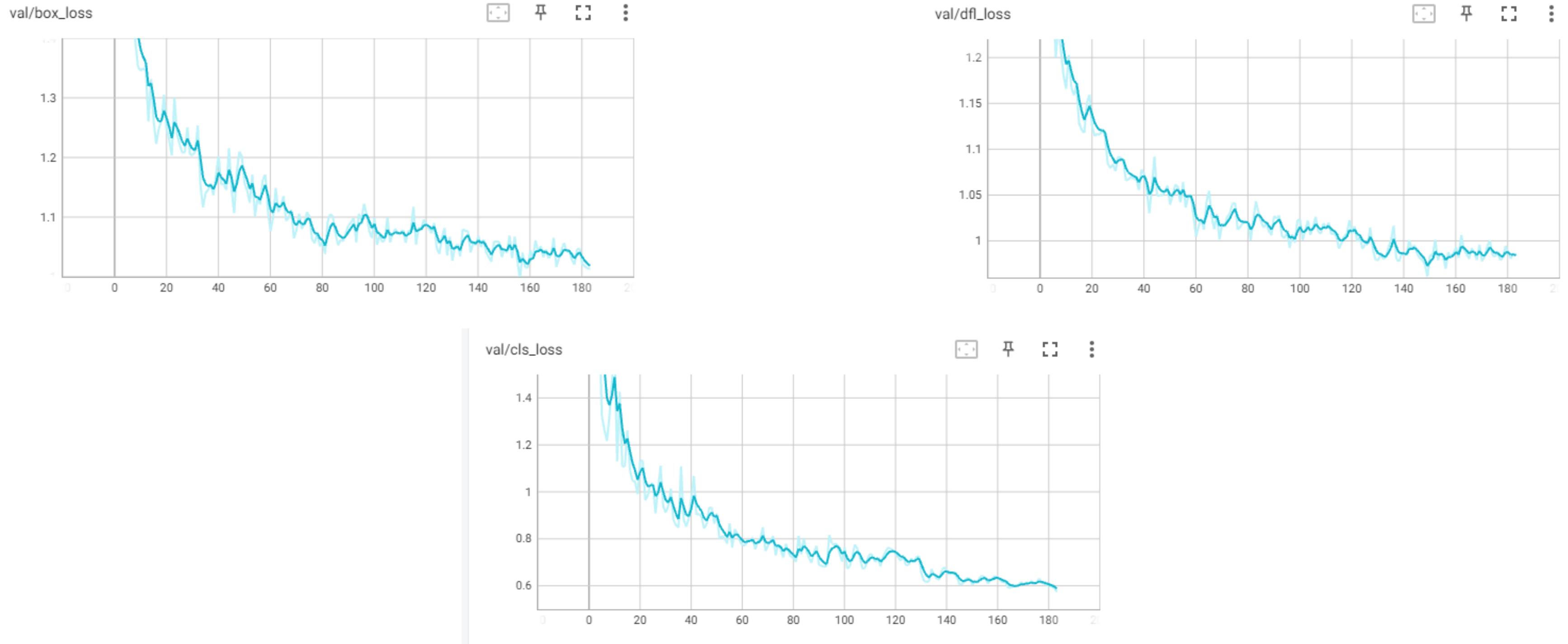
epoch: 100
IoU metric: bbox
학습시간: 3시간 13분

DE:TR	IoU=0.50	0.653
-------	----------	-------

03 중간평가 추후 진행 방향

하이퍼 파라미터 계속 수정 예정, 모델별 공부 및 파악하여 계속 실험
-> 최종 모델 선정 및 최종 튜닝 예정

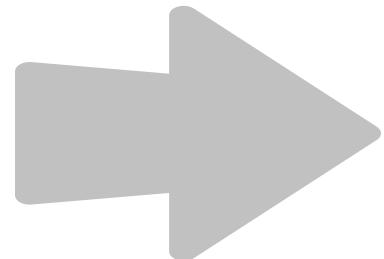
03 중간평가 추후 진행 방향



03 중간평가 epoch에 따른 수치 변화

YOLO8n과 YOLO8s의 mAP50 수치변화(증가)

epoch = 100	
	mAP50
YOLOv8n	0.788
YOLOv8s	0.814



epoch = 200	
	mAP50
YOLOv8n	0.815
YOLOv8s	0.821

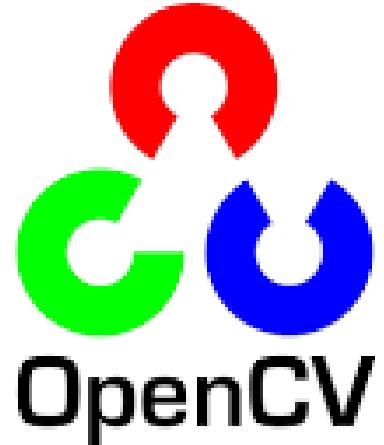
03 YOLOv5x 클래스별 평가수치

ex) YOLOv5x --img 640 --batch 16 --epoch 100

				t4 고Ram	85분	
Class	Images	Instances	P	R	mAP50	mAP50-95
all	60	179	0.818	0.791	0.853	0.577
bdot	60	13	0.9	1	0.995	0.719
bline1	60	15	1	0.914	0.991	0.67
bline2	60	5	0.893	1	0.995	0.867
bline3	60	14	0.914	0.759	0.898	0.55
bline4	60	15	0.931	0.902	0.958	0.756
bline5	60	14	0.863	1	0.995	0.745
bline6	60	12	0.659	1	0.95	0.709
bline7	60	4	0.884	1	0.995	0.672
bline8	60	15	0.843	1	0.995	0.541
break	60	2	1	0	0.532	0.11
circle	60	14	1	0.985	0.995	0.882
scratch	60	13	0.752	0.467	0.619	0.33
sdot	60	22	0.31	0.864	0.626	0.304
stain	60	21	0.496	0.188	0.397	0.219

04 사용기술

라이브러리 및 프레임워크

**roboflow****gradio**PyTorch
Lightning**Transformers**

개발환경

**GitHub****python**{**JSON**}

분석언어

YAML

Data Universe

Q & A

질문 사항

- Q1. 바운딩 박스를 칠 때 타이트하게 치는 것과 여유 있게 치는 것 중 더 좋은 방식은?
- Q2. 14개의 클래스가 valid와 test에 골고루 들어가게 하는 것 외에 train valid test 개수, 비율 등을 어떻게 해야 할지 (특히 데이터가 적을 때)
- Q3. train set을 증식해서 train -> valid set에까지 써도 되는지 (test에는 쓰면 안되는 것으로 알고 있음)
- Q4. 흑백이 아닌 컬러 이미지도 정규화 등을 하는지? 정규화를 한다면, 그 정규화한 이미지로 train
- Q5. 및 valid를 한 다음 정규화하지 않은 이미지로 test를 해야 하는지?
- Q6. 과적합 문제를 조심하면서 epoch을 200, 300 늘려서 성능 개선을 할 때, 실무에서는 어떤 선에서 정하는지? (시간과 비용 등도 고려해야 할 것 같아서)

감사합니다

Data Universe

박주경 <https://github.com/likespike>

김성호 <https://github.com/RTYYYY>

김용훈 <https://github.com/dydgns94>

윤예은 <https://github.com/yenny2>

이승주 <https://github.com/llukaLee>
